Developpez

Le Mag

Édition de octobre - novembre 2013.
Numéro 48.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Éditorial

Comme tous les mois, retrouvez vos rubriques préférées dans ce magazine qui vous est intégralement destiné.

Profitez-en bien !

La rédaction

Article Ruby

RubyMotion et les entrées utilisateur

Aujourd'hui nous allons nous pencher sur la mise en place du traitement d'un formulaire basique.

par **Synbioz**
Page 10

Article Android

Créer sa première bibliothèque Android pour Unity

Dans cet article, nous allons voir comment créer, ajouter et utiliser une bibliothèque JAR pour la plateforme Android à un projet Unity.

par **Jonathan Odul**
Page 39

Ce que personne ne vous a jamais dit à propos de z-index

Le problème de la propriété CSS z-index, c'est que peu de développeurs savent comment elle fonctionne réellement. Ce n'est pas compliqué, mais si vous n'avez jamais pris le temps de lire sa spécification, il y a très certainement des aspects cruciaux que vous ne connaissez pas.

Vous ne me croyez pas ? Alors, voyons si vous saurez résoudre le problème qui vient.

Cet article est la traduction de What No One Told You About Z-Index publié sur le blog de Philip Walton.

1. Le problème

Dans le code HTML suivant, vous avez trois balises chacune contenant une balise . Chaque possède une couleur de fond, respectivement rouge, vert et bleu. Ces sont aussi en position absolute vers le bord haut et gauche du document et se chevauchent légèrement, vous pouvez ainsi voir lesquelles sont au-dessus des autres. Le premier a un z-index valant 1, les autres n'en ont pas encore.
Voici les codes HTML et CSS pour ces éléments :

```
<div>
  <span class="red">Red</span>
</div>
<div>
  <span class="green">Green</span>
</div>
<div>
  <span class="blue">Blue</span>
</div><div>
  <span class="red">Red</span>
</div>
<div>
  <span class="green">Green</span>
</div>
<div>
  <span class="blue">Blue</span>
</div>
```

```
.red, .green, .blue {
  position: absolute;
}
.red {
  background: red;
  z-index: 1;
}
.green {
  background: green;
}
.blue {
  background: blue;
}.red, .green, .blue {
  position: absolute;
}
.red {
  background: red;
  z-index: 1;
}
.green {
  background: green;
}
.blue {
  background: blue;
}
```

Votre défi :
Essayez de faire passer le rouge en dessous des bleu et vert en respectant les règles suivantes :

- vous ne devez pas modifier le code HTML ;
- vous ne pouvez modifier la propriété z-index d'aucun élément ;
- vous ne pouvez modifier la propriété position d'aucun élément.

Pour effectuer vos essais, cliquez sur le lien « edit on Codepen » de l'exemple ci-dessus. Le résultat doit ressembler à l'exemple ci-dessous.

2. La solution

La solution consiste à affecter une opacité inférieure à 1 pour la première (l'élément parent du rouge). Voici le CSS qui a été ajouté dans le code du résultat :

```
div:first-child {
  opacity: .99;
}div:first-child {
  opacity: .99;
}
```

Si vous vous arrachez les cheveux en refusant de croire que l'opacité joue un rôle pour déterminer l'ordre d'empilement des éléments, alors bienvenu au club ! J'ai eu la même réaction lorsque j'ai constaté cela la première fois.
Heureusement, la suite de cet article devrait vous aider à mieux comprendre le pourquoi du comment.

3. L'ordre d'empilement

La propriété z-index semble pourtant si simple à comprendre : les éléments avec un z-index plus élevé sont placés au-dessus de ceux ayant une plus petite valeur, c'est bien ça ? En fait, pas exactement et c'est bien le problème avec z-index. Son fonctionnement semble si simple que la

plupart des développeurs ne prennent pas le temps de lire les spécifications.
Lorsque ni z-index ni position ne sont définis, la règle est simple : l'ordre d'empilement des éléments correspond à leur ordre d'apparition dans le code HTML. En fait, c'est un peu plus compliqué que cela (Lien 01), mais tant que vous n'utilisez pas de marges négatives pour le chevauchement d'éléments de type inline, vous ne rencontrerez probablement pas les effets de bord.
Lorsque vous utilisez la propriété position, tout élément positionné (et ses enfants) sera affiché au-dessus de tout élément non positionné. Pour rappel, un élément est dit positionné si sa propriété position vaut autre chose que static (la valeur par défaut), c'est-à-dire relative, absolute ou fixed.
Enfin, lorsque vous utilisez la propriété z-index, les choses se compliquent un peu. Au premier abord, il parait naturel de considérer que les éléments ayant un z-index plus élevé se placent au-dessus de ceux en ayant un moins élevé et que les éléments ayant un z-index se placent au-dessus de ceux qui n'en ont pas, mais ce n'est pas aussi simple. Tout d'abord, z-index ne fonctionne qu'avec des éléments positionnés. Si vous essayez d'affecter un z-index à un élément non positionné, cela n'aura aucun effet. Ensuite, z-index peut créer des contextes d'empilement et soudainement, ce qui semblait simple devient beaucoup plus compliqué.

4. Le contexte d'empilement

Des groupes d'éléments ayant un parent commun qui se déplace en avant ou en arrière dans l'ordre d'empilement créent ce que l'on appelle un contexte d'empilement (stacking context). Bien comprendre la notion de contexte d'empilement est la clé pour maitriser parfaitement le fonctionnement de z-index et de l'ordre d'empilement.
Chaque contexte d'empilement possède un unique élément racine. Lorsqu'un contexte d'empilement est créé sur un élément, tous ses éléments enfants sont alors confinés à un emplacement défini (celui de l'élément racine) de l'ordre d'empilement. Cela signifie que si un élément se situe dans un contexte d'empilement tout au fond de la pile d'affichage, il ne pourra jamais se placer au-dessus d'un élément d'un autre contexte se situant au-dessus dans l'ordre d'empilement, même en lui affectant un z-index dépassant le milliard !
Il y a trois façons de créer un contexte d'empilement :

- lorsqu'un élément est l'élément racine du document (la balise <html>) ;
- lorsqu'un élément a une propriété position différente de static et un z-index différent de auto ;
- lorsqu'un élément a une opacité inférieure à 1.

Les deux premiers points semblent logiques et sont habituellement bien compris des développeurs Web (même s'ils ne connaissent pas la notion de contexte d'empilement).
Le troisième point (l'opacité) n'est quasiment jamais évoqué en dehors des spécifications du W3C.

5. Déterminer la position d'un élément dans la pile

En fait, déterminer l'ordre d'empilement complet pour tous les éléments de la page (en incluant les bordures, arrière-plans, nœuds textes, etc.) est extrêmement compliqué et va très au-delà du cadre de cet article (encore une fois, je vous renvoie aux spécifications : Lien 01).
Cependant, dans la plupart des cas, une compréhension simplifiée peut mener suffisamment loin et aider à rendre le développement CSS prévisible. Essayons donc de suivre la logique d'ordonnancement pour un contexte d'empilement.

5.1. Ordre d'empilement dans un même contexte

Voici les règles principales pour déterminer l'ordre d'empilement au sein d'un même contexte (du bas vers le haut).

- L'élément racine du contexte.
- Les éléments positionnés (et leurs enfants) avec des valeurs de z-index négatives (les valeurs les plus hautes étant placées au-dessus des valeurs les plus basses ; les valeurs égales sont placées selon leur ordre d'apparition dans le code HTML).
- Les éléments non positionnés (placés selon l'ordre d'apparition dans le code HTML).
- Les éléments positionnés (et leurs enfants) ayant un z-index valant auto (placés selon leur ordre d'apparition dans le code HTML).
- Les éléments positionnés (et leurs enfants) avec des valeurs de z-index positives (les valeurs les plus hautes étant placées au-dessus des valeurs les plus basses ; les valeurs égales sont placées selon leur ordre d'apparition dans le code HTML).

Note : les éléments positionnés ayant des valeurs de z-index négatives sont placés en premier dans la pile du contexte, ce qui signifie qu'ils apparaissent en dessous des autres éléments. À cause de cela, il est possible pour un élément d'être en dessous de son propre parent, ce qui est normalement impossible. Cela n'est toutefois possible que si le parent en question se trouve dans le même contexte et qu'il n'est pas l'élément racine de ce contexte. Vous pouvez voir une magnifique illustration de cela dans l'article de Nicolas Gallagher CSS drop-shadows without images : Lien 02.

5.2. Ordre d'empilement global

En comprenant bien quand et comment est créé un nouveau contexte d'empilement, ainsi que la façon dont est ordonnée la pile au sein d'un contexte, vous devriez vous en sortir pour prévoir où se situeront les éléments.
Le point-clé pour pouvoir s'y retrouver est d'être capable de détecter chaque nouvelle formation de contexte d'empilement. Si vous affectez un z-index d'un milliard et qu'il n'apparait pas en haut de la pile, inspectez ses éléments ancêtres pour vérifier si l'un d'eux ne crée pas un nouveau contexte. Si c'est le cas, alors aucune valeur de z-index ne pourra vous aider.

6. Conclusion

Pour revenir au problème exposé au début de l'article, j'ai réécrit la structure HTML avec en commentaire de chaque balise sa position dans la pile. Cet ordre correspond au code CSS initial.

```
<div><!-- 1 -->
  <span class="red"><!-- 6 --></span>
</div>
```

```
<div><!-- 2 -->
  <span class="green"><!-- 4 --><span>
</div>
<div><!-- 3 -->
  <span class="blue"><!-- 5 --></span>
</div><div><!-- 1 -->
  <span class="red"><!-- 6 --></span>
</div>
<div><!-- 2 -->
  <span class="green"><!-- 4 --><span>
</div>
<div><!-- 3 -->
  <span class="blue"><!-- 5 --></span>
</div>
```

Lorsque nous ajoutons une opacité à la première , l'ordre devient :

```
<div><!-- 1 -->
  <span class="red"><!-- 1.1 --></span>
</div>
<div><!-- 2 -->
  <span class="green"><!-- 4 --><span>
</div>
<div><!-- 3 -->
  <span class="blue"><!-- 5 --></span>
</div><div><!-- 1 -->
  <span class="red"><!-- 1.1 --></span>
</div>
<div><!-- 2 -->
  <span class="green"><!-- 4 --><span>
</div>
<div><!-- 3 -->
  <span class="blue"><!-- 5 --></span>
</div>
```

span.red est passé de 6 à 1.1. J'ai utilisé un point pour indiquer qu'un nouveau contexte était créé et span.red est le premier élément de ce contexte.
J'espère que vous comprenez mieux pourquoi le carré rouge est passé en dessous des autres éléments. Le premier exemple comprenait seulement deux contextes : la racine du document et celui créé par span.red. En ajoutant une opacité au parent de span.red, nous avons formé un troisième contexte, ainsi le z-index défini sur span.red ne s'applique qu'à l'intérieur de ce contexte. Comme la première et ses éléments frères ne sont pas positionnés et n'ont pas de z-index, leur ordre d'empilement correspond à leur ordre d'apparition dans le code HTML, ce qui signifie que la première , ainsi que tous les éléments inclus dans son contexte d'empilement, sont affichés en dessous des suivantes.

Cet article a été publié avec l'aimable autorisation de Philip Walton. L'article original (What No One Told You About Z-Index : Lien 03) peut être vu sur le blog de l'auteur : Lien 04.

Retrouvez l'article de Philip Walton traduit par Didier Mouronval en ligne : Lien 05

Introduction à LESS

LESS est un langage de définition de feuille de style très similaire à CSS par la syntaxe, mais avec plus de fonctionnalités :

variables ;
expressions ;
fonctions ;
structuration du code.

Rien de moins.
Mais beaucoup plus. Vous y trouverez tout ce dont vous avez besoin !

Cet article est une traduction enrichie de An Introduction to Less publié sur le site de l'auteur.

1. En deux mots…

Comme CSS, LESS est un langage de définition de feuilles de style.
LESS est plus facile à utiliser que CSS et offre beaucoup de fonctionnalités très intéressantes.
LESS doit être compilé pour générer du code CSS :

- soit automatiquement au chargement de la page (pratique en phase de développement) ;
- soit durant une phase de compilation pour produire un fichier CSS (recommandé en production).

Attention : si vous utilisez LESS pendant plus de 10 minutes, vous deviendrez dépendant et souffrirez beaucoup si vous venez par la suite à devoir écrire ou maintenir du CSS. Continuez à vos risques et périls !

2. Fonctionnalités principales

Voici un aperçu des fonctionnalités les plus intéressantes autour desquelles s'articulent la plupart des feuilles de style LESS. Pour une liste exhaustive, reportez-vous à la documentation de LESS : Lien 06

2.1. Variables

Tout d'abord, avec LESS, vous pouvez déclarer des variables…

```
@themeColor: #6dd;
@themeBackground: #cfe;@themeColor: #6dd;
@themeBackground: #cfe;
```

… que vous pouvez utiliser partout où vous en avez besoin :

```
body {
    color: @themeColor;
    background-color: @themeBackground;
}

.floating-box {
    border: 1px solid @themeColor;
}

a, a:hover, a:focus, a:active, a:visited {
    color: @themeColor;
}

a:focus, button:focus {
    outline-color: @themeColor;
}body {
    color: @themeColor;
    background-color: @themeBackground;
}

.floating-box {
    border: 1px solid @themeColor;
}

a, a:hover, a:focus, a:active, a:visited {
    color: @themeColor;
}

a:focus, button:focus {
    outline-color: @themeColor;
}
```

En plus de rendre le code plus lisible et maintenable, les variables permettent de faire une distinction claire entre ce qui est destiné à être changé pour obtenir un style différent et ce qui est de l'ordre du comportement en dur. En pratique, vous pouvez avoir une zone « variables » en haut du fichier LESS (ou même dans un fichier à part), dans laquelle le thème est décrit de manière simple et centralisée. C'est par exemple ce que fait Bootstrap (Lien 07) avec le fichier variables.less (Lien 08).
L'utilité des variables ne se limite pas à la définition des couleurs du thème. Par exemple, on peut écrire :

```
@space: @10px;
@tinySpace: @2px;@space: @10px;
@tinySpace: @2px;
```

et réutiliser la variable ainsi définie pour spécifier des marges, des espacements…

```
h2, h3 {
    margin-top: @space;
}h2, h3 {
    margin-top: @space;
}
```

… et même d'autres variables :

```
@defaultPadding: @tinySpace @space;

.notification {
    padding: @defaultPadding;
}

.popup .header {
    padding: @defaultPadding;
}@defaultPadding: @tinySpace @space;

.notification {
    padding: @defaultPadding;
}

.popup .header {
    padding: @defaultPadding;
}
```

2.2. Fonctions et expressions

LESS fournit un ensemble de fonctions permettant d'écrire des choses très intuitives, comme ceci :

```
a:hover {
    color: darken(@themeColor, 20%);
}a:hover {
    color: darken(@themeColor, 20%);
}
```

ou ceci :

```
h2 {
    margin-top: (@space * 2);
}h2 {
    margin-top: (@space * 2);
}
```

De nombreuses fonctions apportent à peu près tout ce qui est nécessaire pour définir le style d'une application Web.

2.3. Structuration

La possibilité d'imbriquer les règles permet de structurer le code. Cette fonctionnalité permet d'écrire un code plus clair, car chaque fichier peut être décomposé en sections, sous-sections, etc. Typiquement, chaque classe CSS décrivant un rôle dans l'interface graphique, ou n'importe quel sous-ensemble du document identifiable par un sélecteur CSS peut être une section. Les règles figurant à l'intérieur de la section s'appliquent uniquement aux descendants.
Par exemple, le code CSS suivant :

```
#header {
    ...
}

#header ul {
    ...
}

#header ul > li {
    ...
}

#header ul > li.divider {
    ...
}
#header ul > li:hover {
    ...
}
#logout-link {
    ...
}#header {
    ...
```

```
}

#header ul {
    ...
}

#header ul > li {
    ...
}

#header ul > li.divider {
    ...
}
#header ul > li:hover {
    ...
}
#logout-link {
    ...
}
```

s'écrirait ainsi avec LESS :

```
#header {
...
    ul {
        ...
        > li {
            ...
            &.divider {
                ...
            }
            &:hover {
                ...
            }
        }
    }
    #logout-link {
        ...
    }
}#header {
...
    ul {
        ...
        > li {
            ...
            &.divider {
                ...
            }
            &:hover {
                ...
            }
        }
    }
    #logout-link {
        ...
    }
}
```

Chacun jugera de la syntaxe la plus agréable… et si le nom LESS se justifie.

Note : un bloc sert aussi de périmètre de visibilité pour les variables.

2.4. Mixins

Un mixin peut être vu comme une sorte de macro acceptant éventuellement des paramètres. Pour chaque utilisation qui en est faite, le contenu du mixin sera copié lors de la compilation en remplaçant les variables par les valeurs spécifiées. Cette fonctionnalité aide beaucoup à diminuer la redondance du code pour certaines constructions. Exemple :

```
.shadow(@spread, @color) {
    -moz-box-shadow: 0px 0px @spread @color
    -webkit-box-shadow: 0px 0px @spread @color
    -ms-filter:
filter:progid:DXImageTransform.Microsoft.Shadow(c
olor=@color, Direction=135, Strength=@spread)
    box-shadow: 0px 0px @spread @color
}

.popup {
    .shadow(10, #000);
    ...
}

.dashboard .widget {
    .shadow(5, #888);
    ...
}.shadow(@spread, @color) {
    -moz-box-shadow: 0px 0px @spread @color
    -webkit-box-shadow: 0px 0px @spread @color
    -ms-filter:
filter:progid:DXImageTransform.Microsoft.Shadow(c
olor=@color, Direction=135, Strength=@spread)
    box-shadow: 0px 0px @spread @color
}

.popup {
    .shadow(10, #000);
    ...
}

.dashboard .widget {
    .shadow(5, #888);
    ...
}
```

À première vue, cette fonctionnalité peut sembler être du sucre syntaxique, mais il ne faut pas la sous-estimer.
Elle permet de factoriser beaucoup de code qui ne pourrait pas l'être autrement, ainsi que de simplifier et de rendre le code plus lisible en donnant un nom unique aux attributs déclinés par type de vendeur.
Elle a également un bénéfice sur la pureté du document HTML. Par exemple, si on veut avoir des ombres sur plusieurs composants dans une application Web, on est quasiment obligé d'avoir une classe CSS shadow sous peine de forte duplication du code CSS implémentant l'ombre. Ceci n'est pas souhaitable (même si c'est un moindre mal) car de l'information de style (avoir une ombre) se retrouve dans le document HTML et non dans le code CSS. Dans l'exemple ci-dessus, l'utilisation du mixin shadow() dans les blocs décrivant un composant devant avoir une ombre permet de remettre cette information dans la feuille de style et non dans le document HTML, et ce sans duplication de code.

3. Comment essayer rapidement ?

Pour démarrer rapidement avec LESS, vous pouvez utiliser la compilation à la volée au chargement. Cette méthode ne demande aucune configuration et est plus pratique en développement. Voici un exemple :

```
<!--  notez l'attribut rel : "stylesheet/less"
-->
<link rel="stylesheet/less" type="text/css"
href="chemin/vers/le/fichier/less/file.less" />
<!-- Chargement de LESS depuis un CDN, le script
doit se trouver après les feuilles de style LESS
    bien sûr, si vous ne souhaitez pas le charger
depuis un CDN, vous pouvez le télécharger
    une fois pour toutes et le placer directement
sur votre site -->
<script
src="http://cdnjs.cloudflare.com/ajax/libs/less.j
s/1.3.3/less.min.js"
type="text/javascript"></script><!--  notez
l'attribut rel : "stylesheet/less" -->
<link rel="stylesheet/less" type="text/css"
href="chemin/vers/le/fichier/less/file.less" />
<!-- Chargement de LESS depuis un CDN, le script
doit se trouver après les feuilles de style LESS
    bien sûr, si vous ne souhaitez pas le charger
depuis un CDN, vous pouvez le télécharger
    une fois pour toutes et le placer directement
sur votre site -->
<script
src="http://cdnjs.cloudflare.com/ajax/libs/less.j
s/1.3.3/less.min.js"
type="text/javascript"></script>
```

Vous n'avez plus qu'à saisir du code LESS dans la feuille de style.

4. Utilisation en production

Pour une utilisation en production, il est souhaitable de précompiler le code LESS afin d'obtenir de meilleures performances du chargement des pages. Cependant cette étape peut être repoussée tant que le volume de code LESS reste faible. Il existe plusieurs façons de compiler le code LESS, voici les deux principales :

- compiler avec les utilitaires fournis en ligne de commande : Lien 09 ;
- compiler avec maven :

```
<plugin>
    <groupId>org.lesscss</groupId>
    <artifactId>lesscss-maven-plugin</artifactId>
    <version>1.3.3</version>
    <configuration>
        <sourceDirectory>$
{project.basedir}/src/main/less</sourceDirectory>
        <outputDirectory>$
{project.basedir}/target/generated-
css</outputDirectory>
        <compress>false</compress>
        <includes>
            <include>style.less</include>
        </includes>
    </configuration>
    <executions>
        <execution>
            <goals>
                <goal>compile</goal>
            </goals>
        </execution>
    </executions>
</plugin><plugin>
    <groupId>org.lesscss</groupId>
    <artifactId>lesscss-maven-plugin</artifactId>
    <version>1.3.3</version>
    <configuration>
        <sourceDirectory>$
{project.basedir}/src/main/less</sourceDirectory>
        <outputDirectory>$
{project.basedir}/target/generated-
css</outputDirectory>
        <compress>false</compress>
        <includes>
            <include>style.less</include>
        </includes>
    </configuration>
    <executions>
        <execution>
            <goals>
                <goal>compile</goal>
            </goals>
        </execution>
    </executions>
</plugin>
```

5. Conclusion

LESS facilite vraiment la vie lors de l'écriture de feuilles de style et vaut vraiment la peine d'être essayé…
Oh… et en parlant d'essayer, n'importe quel fichier CSS valide est également un fichier LESS valide. La seule chose à faire est donc de le renommer et commencer à utiliser les fonctionnalités apportées par LESS.

Retrouvez l'article de Samuel Rossile en ligne : Lien 10

Créer un menu animé en CSS

Dans cet article nous allons apprendre à créer un menu animé en utilisant les transitions CSS.

1. Traduction

Ce tutoriel est la traduction la plus fidèle possible du tutoriel original de Paul Underwood, Create An Animated CSS Box Menu : Lien 11.

2. Introduction

Dans cet article, nous allons utiliser les transitions CSS pour créer un nouveau style de menu. Le but à atteindre est d'avoir plusieurs « boites » de navigation et lorsque l'on passera sur l'une d'elles avec la souris la boite en question grossira et les autres rétréciront, ce qui aura pour effet de la mettre en avant. On pourra également ajouter des icônes qui représenteront la page cible de chaque boite de menu.

3. Le code HTML

En premier lieu nous allons commencer par écrire le code HTML pour notre menu. Nous allons créer un élément div pour chaque boite et lui associer un texte ainsi qu'une image.

```
<div class="nav">
<div class="box home">
    <a href="#home">HOME
    <span><img src="./images/home.png" alt=""
/></span></a>
     </div>
<div class="box about">
    <a href="#about">ABOUT
    <span><img src="./images/person.png"
alt="" /></span></a>
      </div>
<div class="box portfolio">
    <a href="#portfolio">PORTFOLIO
    <span><img src="./images/folder.png"
alt="" /></span></a>
      </div>
<div class="box services">
    <a href="#services">SERVICES
    <span><img src="./images/screw-driver.png"
alt="" /></span></a>
     </div>
<div class="box contact">
    <a href="#contact">CONTACT
    <span><img src="./images/mail-back.png"
alt="" /></span></a>
     </div>
</div>
```

4. Ajouter du style sur les boites

Le code CSS va nous fournir toutes les fonctionnalités nécessaires pour ajouter du style à notre menu. Nous allons débuter par ajouter du style pour chacune des boites, on définit une hauteur et une largeur que l'on souhaite aux boites afin de remplir l'espace disponible. On ajoute également un display : inline-block afin de les afficher côte à côte plutôt que l'une en dessous de l'autre.
Comme nous allons également afficher une icône (qui sera animée), nous voulons nous assurer que rien ne viendra masquer notre contenu en utilisant la propriété overflow: hidden.
Enfin, nous pouvons ajouter la transition pour la largeur de la boite de manière à modifier cette largeur au survol de la souris.

```
.box
{
    display: inline-block;
        float:left;
    height:200px;
    overflow: hidden;
    width:20%;
    -webkit-transition: width 1s;
    -moz-transition: width 1s;
        transition: width 1s;
}
```

Nous pouvons maintenant ajouter une couleur d'arrière-plan à nos boites de menu.

```
.box.home      { background-color: #2d89ef; }
.box.about     { background-color: #00a300; }
.box.portfolio { background-color: #e3a21a; }
.box.services  { background-color: #9f00a7; }
.box.contact   { background-color: #ee1111; }
```

Comme nous voulons que toute la zone des boites soit cliquable, nous devons modifier l'ancre qui sert de lien pour la transformer en élément de type block en utilisant la propriété display: block et en définissant une hauteur de 100 %.

```
.box a
{
    color:#FFF;
    text-decoration: none;
    text-align: center;
    vertical-align: middle;
    height:100%;
    display:block;
        padding-top: 20px;
}
```

Nous voulons également ajouter une icône animée qui s'affichera aussi au survol de la souris. Nous allons une fois de plus utiliser les transitions CSS pour modifier la propriété top de l'élément span à l'intérieur duquel sera stockée une image que nous utiliserons comme icône. Au

début nous avons besoin de déplacer l'image en dehors de l'élément, ce qui aura pour effet de la masquer puisque nous avons défini un overflow: hidden. Puis nous pouvons utiliser la propriété position: relative avec un top défini à 100 % pour déplacer l'élément span en dehors.
Ajouter une transition sur la propriété top va animer l'icône et la déplacer à l'intérieur de la boite de menu qui sera survolée par la souris.

```
.box span
{
    display:block;
    position:relative;
    top:100%;
    text-align: center;
    -webkit-transition: top 1s;
    -moz-transition: top 1s;
    transition: top 1s;
}
```

5. L'événement survol de la souris

Créer l'événement qui va se déclencher au survol de la souris va nous permettre trois choses :

- tout d'abord toutes les boites de menu vont se réduire jusqu'à 10 % ;
- la boite survolée par le curseur de la souris va s'agrandir jusqu'à une largeur de 60 % ;
- puis nous modifions la valeur de la propriété top de l'élément span contenu dans la boite de menu, ce qui va avoir pour incidence de le faire apparaitre et ainsi afficher l'icône.

```
.nav:hover .box { width:10%; }
.nav .box:hover { width: 60%; }
.box:hover span{ top:25%; }
```

6. Conclusion

Dans cet article nous avons vu comment créer de manière très simple un menu animé agréable pour un utilisateur. Nous avons pour cela utilisé les transitions CSS. Vous pouvez bien sûr choisir vos propres couleurs d'arrière-plan, taille de boite de menu, etc. N'hésitez pas à le personnaliser !

Retrouvez l'article de Paul Underwood traduit par Sébastien Germez en ligne : Lien 12

RubyMotion et les entrées utilisateur

Dans le premier article à propos de RubyMotion, nous avons vu les bases de la création d'une application avec l'utilisation de Rake puis du REPL. Nous avons ensuite mis en place l'affichage de labels, le traitement de date/heure ainsi que la personnalisation des images de fond.

Nous avons vu qu'il est très facile et assez concis de mettre en place le traitement d'informations, mais aussi l'UI via RubyMotion.

Aujourd'hui nous allons nous pencher sur la mise en place du traitement d'un formulaire basique. L'idée est de permettre à l'utilisateur de sélectionner une heure ainsi qu'un fuseau horaire pour lui permettre de convertir cette heure vers le fuseau horaire concerné.

Voyons comment procéder.

1. Création de l'application

Commençons par créer le projet :

```
$ motion create TimeZones
Create TimeZones
Create TimeZones/.gitignore
Create TimeZones/app/app_delegate.rb
Create TimeZones/Rakefile
Create TimeZones/resources/Default-568h@2x.png
Create TimeZones/spec/main_spec.rb$ motion create
TimeZones
Create TimeZones
Create TimeZones/.gitignore
Create TimeZones/app/app_delegate.rb
Create TimeZones/Rakefile
Create TimeZones/resources/Default-568h@2x.png
Create TimeZones/spec/main_spec.rb
```

Nous allons, pour les besoins de cette présentation, demander à l'utilisateur d'entrer son nom, de choisir une heure et un fuseau horaire cible. Nous aurons donc besoin d'un champ texte, d'une liste de fuseaux horaires, d'un sélecteur de date et d'un label pour afficher le résultat.
Commençons par le plus simple, l'ajout du champ texte, du bouton de validation et du label.

2. Formulaire de récupération du nom

Ouvrez le projet dans votre éditeur de texte préféré pour vous diriger dans le fichier app/app_delegate.rb pour y instancier la fenêtre principale :

```
class AppDelegate
  def application(application,
didFinishLaunchingWithOptions:launchOptions)
    @window =
UIWindow.alloc.initWithFrame(UIScreen.mainScreen.
bounds)
    @window.rootViewController =
RootViewController.alloc.init
    @window.makeKeyAndVisible
  end
endclass AppDelegate
```

```
  def application(application,
didFinishLaunchingWithOptions:launchOptions)
    @window =
UIWindow.alloc.initWithFrame(UIScreen.mainScreen.
bounds)
    @window.rootViewController =
RootViewController.alloc.init
    @window.makeKeyAndVisible
  end
end
```

Comme vous pouvez le voir, nous utilisons comme contrôleur principal une instance de RootViewController. Il va donc falloir créer cette classe :

app/controllers/root_view_controller.rb

```
class RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor
  end
endclass RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor
  end
end
```

Vous pouvez déjà compiler le projet via la commande rake et vous verrez apparaitre la fenêtre principale avec en fond la texture grisée.

Nous pouvons maintenant passer à l'ajout du champ texte et de son label de description :

```
class RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    view.addSubview name_label
    view.addSubview name_text_field
  end

  private

  def name_label
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"
    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)

    textField
  end
endclass RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    view.addSubview name_label
    view.addSubview name_text_field
  end

  private

  def name_label
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"
    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)

    textField
  end
end
```

Nous avons ici ajouté deux méthodes privées qui nous permettent de générer un label avec fond transparent et texte blanc puis un champ texte avec une police de caractères à 15 px.
Ces deux méthodes sont appelées l'une après l'autre au chargement de la vue pour que les éléments soient ajoutés à la vue principale.

Si vous testez, vous verrez qu'il n'y a pour le moment pas grand-chose de fonctionnel, d'ailleurs une fois le clavier virtuel déployé, impossible de revenir en arrière.

Traitons ce problème avant de passer à la suite.

2.1. Masquage du clavier virtuel

```
def textFieldShouldReturn(text_field)
  text_field.resignFirstResponder
enddef textFieldShouldReturn(text_field)
  text_field.resignFirstResponder
end
```

La définition de la méthode textFieldShouldReturn permet de définir ce qui doit se passer lorsque l'utilisateur demande explicitement la fermeture du clavier via un appui sur « entrée » par exemple.
Il faudra ensuite préciser à qui nous déléguons cette tâche, en l'occurrence ce sera notre contrôleur principal qui va s'en charger.
Nous allons donc modifier notre code pour qu'il fonctionne comme attendu :

```
class RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    view.addSubview name_label
    view.addSubview name_text_field
  end

  def textFieldShouldReturn(text_field)
    text_field.resignFirstResponder
  end

  private

  def name_label
```

```
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"

    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)
    textField.delegate = self

    textField
  end
endclass RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    view.addSubview name_label
    view.addSubview name_text_field
  end

  def textFieldShouldReturn(text_field)
    text_field.resignFirstResponder
  end

  private

  def name_label
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"

    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)
    textField.delegate = self

    textField
  end
end
```

Nous avons donc ajouté le textField.delegate = self dans la méthode name_text_field qui permet de préciser le contrôleur qui doit gérer le comportement du champ texte, puis nous avons ajouté la méthode dédiée à la gestion de l'appui sur la touche « retour » dans laquelle nous demandons de fermer le clavier pour le champ texte à l'origine de la demande.
Pour peaufiner le comportement, il serait pratique de faire en sorte que le clavier se ferme également lorsque nous cliquons ailleurs que sur le clavier lui-même.
Pour arriver à nos fins, nous allons attraper l'événement de « tap » simple sur l'écran. Lorsqu'un « tap » est effectué, nous fermerons le clavier si ce dernier est visible. Lors d'un « tap » nous n'aurons pas connaissance du champ texte en cours d'utilisation, nous allons donc devoir stocker notre champ texte dans une variable d'instance pour pouvoir y faire référence par la suite. Voici donc le code modifié :

```
class RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    @text_field = name_text_field

    view.addSubview name_label
    view.addSubview @text_field

    single_tap =
UITapGestureRecognizer.alloc.initWithTarget(self,
action: :'handle_single_tap')
    view.addGestureRecognizer(single_tap)
  end

  def textFieldShouldReturn(text_field)
    text_field.resignFirstResponder
  end

  def handle_single_tap
    @text_field.resignFirstResponder
  end

  private

  def name_label
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"

    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)
    textField.delegate = self

    textField
  end
endclass RootViewController < UIViewController
  def viewDidLoad
    view.backgroundColor =
UIColor.scrollViewTexturedBackgroundColor

    @text_field = name_text_field

    view.addSubview name_label
    view.addSubview @text_field

    single_tap =
UITapGestureRecognizer.alloc.initWithTarget(self,
action: :'handle_single_tap')
    view.addGestureRecognizer(single_tap)
  end

  def textFieldShouldReturn(text_field)
```

```
    text_field.resignFirstResponder
  end

  def handle_single_tap
    @text_field.resignFirstResponder
  end

  private

  def name_label
    label = UILabel.alloc.initWithFrame [[10,
10], [100, 30]]
    label.backgroundColor = UIColor.clearColor
    label.textColor = UIColor.whiteColor
    label.text = "Votre nom"

    label
  end

  def name_text_field
    textField = UITextField.alloc.initWithFrame
[[110, 10], [170, 30]]
    textField.borderStyle =
UITextBorderStyleRoundedRect
    textField.font = UIFont.systemFontOfSize(15)
    textField.delegate = self

    textField
  end
end
```

Nous avons donc ajouté la reconnaissance du « tap » simple sur notre vue et associé ce tap à la méthode handle_single_tap qui va explicitement demander à notre champ texte de masquer son clavier s'il est visible.

2.2. Mise à jour de l'affichage

Lorsque notre utilisateur valide sa saisie, nous allons ajouter un message dans un label pour le saluer. Ce label servira également à lui indiquer l'heure du fuseau horaire sélectionné par la suite.
Le code va être assez simple puisqu'en nous basant sur l'existant, il ne nous reste qu'à récupérer le texte lors de la validation de la saisie par l'utilisateur pour l'ajouter à notre label nouvellement créé.
Nous ajoutons donc une méthode privée pour générer le label :

```
def remote_time_label
  label = UILabel.alloc.initWithFrame [[0, 350],
[view.frame.size.width, 30]]
  label.backgroundColor = UIColor.clearColor
  label.textColor = UIColor.whiteColor
  label.textAlignment = NSTextAlignmentCenter

  label
enddef remote_time_label
  label = UILabel.alloc.initWithFrame [[0, 350],
[view.frame.size.width, 30]]
  label.backgroundColor = UIColor.clearColor
  label.textColor = UIColor.whiteColor
  label.textAlignment = NSTextAlignmentCenter

  label
end
```

On prend ici soin de créer un label qui prend toute la largeur de la vue et d'aligner le texte au centre, puis nous ajoutons cette vue à notre vue principale via viewDidLoad :

```
@remote_time_label = remote_time_label

view.addSubview
@remote_time_label@remote_time_label =
remote_time_label

view.addSubview @remote_time_label
```

Nous pouvons maintenant faire en sorte que le label soit mis à jour suite à la validation de l'utilisateur :

```
def textFieldShouldReturn(text_field)
  text_field.resignFirstResponder
  @remote_time_label.text = "Bonjour
#{text_field.text} !"
enddef textFieldShouldReturn(text_field)
  text_field.resignFirstResponder
  @remote_time_label.text = "Bonjour
#{text_field.text} !"
end
```

Nous avons donc bouclé notre première étape et nous allons pouvoir passer à la suivante qui va consister à permettre à l'utilisateur de choisir son fuseau horaire cible.

3. Conversion de l'heure

Il va nous falloir récupérer une liste des fuseaux horaires disponibles pour ensuite les afficher dans une liste déroulante. L'utilisateur pourra ainsi faire son choix et on utilisera la valeur sélectionnée pour faire la conversion de l'heure.
Dans un premier temps, concentrons-nous sur la mise en place de cette liste déroulante pour simplement afficher le fuseau horaire sélectionné.

3.1. Sélection du fuseau horaire

En tout premier lieu, à l'initialisation de la vue, nous allons créer une variable d'instance qui contiendra les fuseaux horaires. Cocoa nous permet d'obtenir cette liste très facilement :

```
@timezones =
NSTimeZone.knownTimeZoneNames@timezones =
NSTimeZone.knownTimeZoneNames
```

Nous pouvons ensuite ajouter une méthode privée qui nous servira à générer notre vue pour la liste déroulante :

```
def timezone_picker
  picker = UIPickerView.alloc.init
  picker.showsSelectionIndicator = true
  picker.center = self.view.center
  picker.dataSource = self
  picker.delegate = self

  picker
enddef timezone_picker
  picker = UIPickerView.alloc.init
  picker.showsSelectionIndicator = true
  picker.center = self.view.center
  picker.dataSource = self
  picker.delegate = self
```

```
  picker
end
```

Nous initialisons donc un UIPickerView. Nous précisons que l'on souhaite avoir un indice visuel pour l'élément sélectionné. On place la liste déroulante au centre de la fenêtre, puis on définit le délégué et la source de données. Une fois encore, c'est notre contrôleur principal qui se chargera de ça.
Nous pouvons maintenant ajouter cette vue à la vue principale :

```
view.addSubview timezone_pickerview.addSubview
timezone_picker
```

Il ne nous reste plus qu'à implémenter les méthodes requises par l'interface de UIPickerView :

```
def numberOfComponentsInPickerView(pickerView)
  1
end

def pickerView(pickerView,
numberOfRowsInComponent:component)
  @timezones.size
end

def pickerView(pickerView, titleForRow:row,
forComponent:component)
  @timezones[row]
end

def pickerView(pickerView, didSelectRow:row,
inComponent:component)
  @remote_time_label.text = "#{@text_field.text},
vous avez choisi #{@timezones[row]}"
enddef numberOfComponentsInPickerView(pickerView)
  1
end

def pickerView(pickerView,
numberOfRowsInComponent:component)
  @timezones.size
end

def pickerView(pickerView, titleForRow:row,
forComponent:component)
  @timezones[row]
end

def pickerView(pickerView, didSelectRow:row,
inComponent:component)
  @remote_time_label.text = "#{@text_field.text},
vous avez choisi #{@timezones[row]}"
end
```

La méthode numberOfComponentsInPickerView permet de définir le nombre de listes déroulantes qu'on aura au sein de la vue, dans notre cas, nous n'avons qu'une liste à afficher.
La méthode pickerView(pickerView, numberOfRowsInComponent:component) nous permet d'indiquer au composant le nombre total d'éléments qui seront dans la liste. Pour nous, c'est le nombre d'éléments dans notre tableau de fuseaux horaires.
La méthode pickerView(pickerView, titleForRow:row, forComponent:component) permet de définir le contenu de chaque élément de la liste, le titre de l'élément en quelque sorte, on aura par exemple « Europe/Paris ». Nous devons donc tout simplement retourner l'élément de notre tableau à l'index demandé, ici disponible via la variable row.
La méthode pickerView(pickerView, didSelectRow:row, inComponent:component) permet quant à elle de définir le comportement à adopter lorsqu'une sélection est faite dans la liste. Nous décidons ici d'utiliser notre label pour y afficher le nom de l'utilisateur ainsi que le fuseau horaire choisi.
Voici un exemple du résultat obtenu :

3.2. Choix de l'heure

La dernière étape en termes d'interaction avec l'utilisateur est la possibilité de lui laisser choisir la date et l'heure à convertir, pour cela nous allons utiliser un élément d'UI appelé UIDatePicker.
Nous avons utilisé pour le UIPickerView la taille par défaut, nous pourrions faire de même avec le UIDatePicker mais tous les éléments ne tiendraient pas à l'écran. Nous allons donc devoir personnaliser la taille et le positionnement du UIPickerView, les UIDatePicker ne permettant pas ce genre de manipulation.
Nous ajoutons donc une méthode privée pour générer la vue dont nous avons besoin :

```
def date_picker
  picker = UIDatePicker.alloc.init
  picker.center = [view.frame.size.width / 2,
320]

  picker
enddef date_picker
  picker = UIDatePicker.alloc.init
  picker.center = [view.frame.size.width / 2,
320]

  picker
end
```

Il nous suffit ensuite de l'ajouter à la vue principale lors de son initialisation :

```
view.addSubview date_pickerview.addSubview
date_picker
```

Finalement, nous devons modifier les méthodes existantes pour le UIPickerView et notre label afin qu'ils ne se

recouvrent pas les uns les autres :

```
def remote_time_label
  label = UILabel.alloc.initWithFrame [[0, 430],
[view.frame.size.width, 30]]
  label.backgroundColor = UIColor.clearColor
  label.textColor = UIColor.whiteColor
  label.textAlignment = NSTextAlignmentCenter

  label
end

def timezone_picker
  picker = UIPickerView.alloc.initWithFrame [[0,
50], [320, 120]]
  picker.showsSelectionIndicator = true
  picker.dataSource = self
  picker.delegate = self

  picker
enddef remote_time_label
  label = UILabel.alloc.initWithFrame [[0, 430],
[view.frame.size.width, 30]]
  label.backgroundColor = UIColor.clearColor
  label.textColor = UIColor.whiteColor
  label.textAlignment = NSTextAlignmentCenter

  label
end

def timezone_picker
  picker = UIPickerView.alloc.initWithFrame [[0,
50], [320, 120]]
  picker.showsSelectionIndicator = true
  picker.dataSource = self
  picker.delegate = self

  picker
end
```

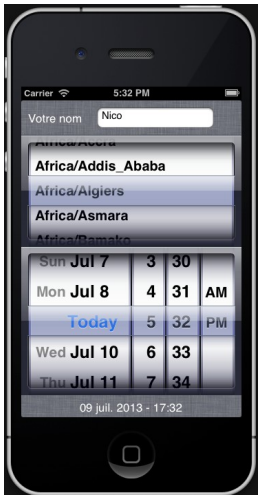
Nous pouvons maintenant mettre en place un mécanisme similaire à celui du UIPickerView pour réagir lors de la sélection d'une date par l'utilisateur. Pour cela, nous allons devoir stocker notre sélecteur de date dans une variable d'instance puis observer ses événements pour savoir quand la valeur sélectionnée change.
Dans le viewDidLoad, on aura :

```
@date_picker = date_picker
view.addSubview @date_picker

@date_picker.addTarget(self,
action: :'handle_date_change',
forControlEvents:UIControlEventValueChanged)@date
_picker = date_picker
view.addSubview @date_picker

@date_picker.addTarget(self,
action: :'handle_date_change',
forControlEvents:UIControlEventValueChanged)
```

On demande ici a être prévenu à chaque changement de valeur dans le UIDatePicker et que la méthode handle_date_change soit appelée à ce moment-là. Il ne nous reste donc plus qu'à implémenter cette méthode pour afficher la date sélectionnée :

```
def handle_date_change
  fr_FR = NSLocale.alloc.initWithLocaleIdentifier
"fr_FR"

  format = NSDateFormatter.alloc.init
  format.locale = fr_FR
  format.setDateFormat("dd MMM yyyy - HH:mm")

  # Conversion de la date en chaine
  dateString =
format.stringFromDate(@date_picker.date)

  @remote_time_label.text = dateString
enddef handle_date_change
  fr_FR = NSLocale.alloc.initWithLocaleIdentifier
"fr_FR"

  format = NSDateFormatter.alloc.init
  format.locale = fr_FR
  format.setDateFormat("dd MMM yyyy - HH:mm")

  # Conversion de la date en chaine
  dateString =
format.stringFromDate(@date_picker.date)

  @remote_time_label.text = dateString
end
```

On récupère donc la date via @date_picker.date, date qu'on prend soin de formater pour l'affichage comme vu dans le précédent article. On l'affiche ensuite dans le label.



3.3. Conversion vers le fuseau horaire choisi

Dernière étape de cet article, convertir la date/heure choisie vers le fuseau horaire choisi juste au-dessus. Nous allons donc devoir modifier notre méthode handle_date_change. Pour avoir accès au fuseau horaire sélectionné à tout instant, nous allons stocker la vue dans une variable d'instance dans le viewDidLoad :

```
@timezone_picker = timezone_picker
view.addSubview @timezone_picker@timezone_picker
= timezone_picker
view.addSubview @timezone_picker
```

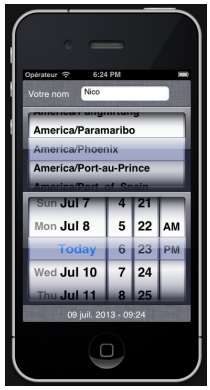
Nous pouvons maintenant modifier la méthode handle_date_change :

```
def handle_date_change
  selected_row =
@timezone_picker.selectedRowInComponent(0)
```

```
  selected_tz = @timezones[selected_row]

  fr_FR = NSLocale.alloc.initWithLocaleIdentifier
"fr_FR"

  format = NSDateFormatter.alloc.init
  format.locale = fr_FR
  format.timeZone =
NSTimeZone.timeZoneWithName(selected_tz)
  format.setDateFormat("dd MMM yyyy - HH:mm")

  dateString =
format.stringFromDate(@date_picker.date)

  @remote_time_label.text = dateString
enddef handle_date_change
  selected_row =
@timezone_picker.selectedRowInComponent(0)
  selected_tz = @timezones[selected_row]

  fr_FR = NSLocale.alloc.initWithLocaleIdentifier
"fr_FR"

  format = NSDateFormatter.alloc.init
  format.locale = fr_FR
  format.timeZone =
NSTimeZone.timeZoneWithName(selected_tz)
  format.setDateFormat("dd MMM yyyy - HH:mm")

  dateString =
format.stringFromDate(@date_picker.date)

  @remote_time_label.text = dateString
end
```

Nous n'avons finalement pas modifié grand-chose puisqu'on récupère simplement le fuseau horaire sélectionné et on s'en sert ensuite sur notre formateur de date via format.timeZone = NSTimeZone.timeZoneWithName(selected_tz).
Pour finaliser le fonctionnement, on remplace le code de pickerView(pickerView, didSelectRow:row, inComponent:component) pour qu'il appelle handle_date_change. On a donc la date qui se met à jour dès que l'on change le fuseau horaire ou l'heure :

```
def pickerView(pickerView, didSelectRow:row,
inComponent:component)
  handle_date_change
enddef pickerView(pickerView, didSelectRow:row,
inComponent:component)
  handle_date_change
end
```



Nous avons maintenant une application capable de convertir une heure donnée vers tous les fuseaux horaires !

4. Conclusion

Nous avons vu dans cet article plusieurs composants qui à eux seuls peuvent largement suffire à créer une application complète et fonctionnelle.
Il n'y a aucun piège à éviter, la seule chose bloquante peut être d'avoir à se rappeler des signatures des méthodes déléguées de UIPickerView.
Vous trouverez le code de cet article sur GitHub : Lien 13
Dans le prochain article concernant RubyMotion, nous verrons comment mettre en place des models, des interactions multi-controllers. Nous découvrirons de nouveaux éléments d'UI que nous personnaliserons. Nous verrons également comment créer des transitions entre les différentes vues root des controllers.

Retrouvez l'article de Synbioz en ligne : Lien 14.

AMD loader pour un code JavaScript organisé et performant

L'écosystème JavaScript grandit. JavaScript côté serveur avec Node.js, JavaScript côté client avec des frameworks de plus en plus élaborés (jQuery, Dojo, Mootools…) et des projets associés pour répondre à nos problématiques récurrentes (backbone.js et consorts pour du MVC, raphael.js et ses amis pour les graphismes…). Bref JavaScript devient incontournable et peu de pages Web sauraient s'en affranchir. Revers de la médaille, souvent mal maitrisé et mal utilisé dans son contexte de page Web, il s'attire les foudres de nombreux développeurs. Qui n'a jamais pesté contre ses dizaines de kilooctets de scripts qui sont emmenés avec une page Web, ralentissant d'autant son téléchargement, obligeant à déplacer les balises <script> en fin de page ? Qui n'a jamais perdu quelques cheveux lors de la gestion des dépendances entre les scripts et leur ordre de chargement ? Qui n'a jamais transpiré quand on lui a annoncé que sur une même page devront cohabiter jQuery, Mootools et Prototype ? Ou pire, deux versions différentes de jQuery ? Dans cet article nous allons découvrir une API en plein essor, solution globale à tous ces soucis, indispensable, mais trop peu utilisée quotidiennement par nos soins : Asynchronous Module Definition (AMD loader). Ou comment définir des modules JavaScript (techniquement chaque module est un fichier .js) qui pourront être ensuite chargés en parallèle en mode asynchrone, dans leur propre scope - donc sans conflit, en même temps que leurs dépendances, offrant ainsi une scalabilité souvent décriée.

1. AMD loader en deux mots : define, require

L'API est constituée de deux fonctions : define qui définit un module en renvoyant une valeur ou une fonction et require qui est similaire mais se contente d'effectuer un simple callback.

```
//Code écrit dans le fichier package1/moduleC.js
//définition ici du module package1/moduleC
//moduleC dépend de moduleA défini dans le
package2 et de moduleB défini dans package1
define(["package2/moduleA", "./moduleB"],
function(depA, depB) {
  //utilisation de depA et depB

// Renvoyer ce que le module définit : Objet,
Classe, variable etc.
  return function() {
    //...
  }
});//Code écrit dans le fichier
package1/moduleC.js
//définition ici du module package1/moduleC
//moduleC dépend de moduleA défini dans le
package2 et de moduleB défini dans package1
define(["package2/moduleA", "./moduleB"],
function(depA, depB) {
  //utilisation de depA et depB

// Renvoyer ce que le module définit : Objet,
Classe, variable etc.
  return function() {
    //...
  }
});
```

La notion de package (ou de namespace) correspond à celle en Java : une arborescence. Dans l'exemple précédent, le loader AMD va donc charger en asynchrone les fichiers package2/moduleA.js et package1/moduleB.js et une fois ces dépendances résolues, les injectera sous la forme de paramètres dans la fonction de rappel (callback), laquelle renverra la « valeur » (au sens large) du module. À noter qu'on peut aussi donner un nom optionnel au module (dans ce cas c'est le premier paramètre de define) mais que cette pratique n'est pas recommandée pour des raisons d'unicité.
Quand on souhaite ensuite utiliser un ou plusieurs modules pour mener des actions (par exemple au sein d'une page Web), on utilise alors la fonction require et un objet de configuration qui indique notamment les chemins possibles pour trouver les modules (en très gros des liens vers les ressources…) :

```
//Objet de config, exemple du loader Curl
curl = {
    paths: {
        curl: '../src/curl/',
        package1: '../projet/package1'
    },
    locale: 'fr'  /* sert pour les aspects i18n,
voir plus bas */
};
//on "requiert" le module package1/moduleC, qui
sera chargé depuis
//../projet/package1/moduleC.js, ce qui va
induire l'exécution
//du code précédent donc le chargement des
dépendances, etc.
require(["package1/moduleC"], function(depC) {

   //utilisation ici de depC
});//Objet de config, exemple du loader Curl
curl = {
    paths: {
        curl: '../src/curl/',
        package1: '../projet/package1'
    },
    locale: 'fr'  /* sert pour les aspects i18n,
voir plus bas */
};
```

```
//on "requiert" le module package1/moduleC, qui sera chargé depuis
//../projet/package1/moduleC.js, ce qui va induire l'exécution
//du code précédent donc le chargement des dépendances etc.
require(["package1/moduleC"], function(depC) {

   //utilisation ici de depC
});
```

On saisit rapidement les avantages d'une telle architecture modulaire : code correctement organisé, assurance de voir les dépendances résolues, chargement facilité des seuls modules concernés (chaque module n'est évidemment chargé qu'une seule fois), dynamiquement, voire dans l'ordre souhaité. En effet, pourquoi charger l'ensemble d'une bibliothèque ou d'un framework si on n'en utilise qu'une fraction ? Avec AMD, on gagne en dynamique et on réduit les temps de chargement d'un facteur pouvant aller jusqu'à 10…
Pas encore complètement convaincu ? Cela va venir. Prenons par exemple le footer d'une page Web : il peut correspondre à cela (exemple d'un snapshot de l'excellent Tatami à un moment de son développement) :

```
<script src="http://code.jquery.com/jquery-1.7.2.min.js"></script>
..
<script src="/assets/js/bootstrap/2.0.2/bootstrap-dropdown.js"></script>
...
<script src="/assets/js/raphael/2.1.0/raphael-min.js"></script>
<script src="/assets/js/mustache/mustache.js"></script>

<script src="/assets/js/tatami/constants.js"></script>
<script src="/assets/js/tatami/standard/tatami.utils.js"></script>
<script src="/assets/js/tatami/standard/tatami.tweets.js"></script>
<script src="/assets/js/tatami/standard/tatami.users.js"></script>
<script src="/assets/js/tatami/standard/tatami.ajax.js"></script>
<script src="/assets/js/tatami/standard/tatami.js"></script><script src="http://code.jquery.com/jquery-1.7.2.min.js"></script>
..
<script src="/assets/js/bootstrap/2.0.2/bootstrap-dropdown.js"></script>
...
<script src="/assets/js/raphael/2.1.0/raphael-min.js"></script>
<script src="/assets/js/mustache/mustache.js"></script>

<script src="/assets/js/tatami/constants.js"></script>
<script src="/assets/js/tatami/standard/tatami.utils.js"></script>
<script src="/assets/js/tatami/standard/tatami.tweets.js"></script>
<script src="/assets/js/tatami/standard/tatami.users.js"></script>
<script src="/assets/js/tatami/standard/tatami.ajax.js"></script>
<script src="/assets/js/tatami/standard/tatami.js"></script>
```

Notons déjà que tous ces scripts se chargent en mode **synchrone** (on cumule les temps de chargement). Ensuite, comme au sein de la page Web Tatami certaines fonctionnalités peuvent être gérées dynamiquement, sans rechargement de la page (graphismes pour les statistiques, etc.), **il est donc nécessaire de charger aussi** les bibliothèques de Raphael et Mustache. On décèle ici le hic : ces dépendances sont chargées inutilement si l'utilisateur ne demande pas à utiliser les fonctionnalités associées. En utilisant un loader AMD, non seulement tous ces scripts seraient chargés en **asynchrone** (donc chargements en parallèle), mais en plus, on chargerait Mustache et Raphael **uniquement** en cas de besoin.
Convaincus ?

2. Les loaders AMD existants

Il existe plusieurs implémentations, chacune pouvant prendre en charge complètement les modules qui respectent l'API. La plus connue est **RequireJS**, puis viennent **Curl**, **Backdraft**… Cas particulier : s'il n'a pas échappé à certains que **node.js** porte aussi une notion de modules, sachez que celui-ci utilise une autre forme de loader, **CommonJS** (précurseur, initiateur de AMD) fonctionnant sur le même principe, mais moins bien adapté au Web. À noter que nombre de loaders AMD proposent généralement une couche qui les rend compatibles avec **node.js**. Pour la suite nous utiliserons **Curl**, qui mérite d'être connu, car il présente l'avantage de proposer en complément des chainages et des deferred. Pour autant, même si **Curl** expose la fonction require, ces extensions légères imposent de disposer d'une autre fonction pour qu'il n'y ait pas de confusion, ce sera donc la fonction curl. Retenons que curl = require.
Tous les exemples sont consultables dans ce dépôt Github (Lien 15) ou dans l'archive jointe (Lien 16).

3. Premier exemple

À l'aide de trois modules, affichons un « Hello world » — pour commencer sagement.
Créons un premier module client/world dans le fichier client/world.js, qui ne fait que renvoyer une chaine de caractères :

```
//module client/world
define("world");//module client/world
define("world");
```

Créons maintenant un second module client/hello dans le fichier client/hello.js, qui dépend de client/world :

```
//module client/hello
//d'abord le module world est chargé, puis sa valeur renvoyée
```

```
//est passée comme paramètre txt
define(['./world'], function (txt) {
    return "hello " + txt;
});//module client/hello
//d'abord le module world est chargé, puis sa
valeur renvoyée
//est passée comme paramètre txt
define(['./world'], function (txt) {
    return "hello " + txt;
});
```

Créons maintenant un module utils/string dans le fichier utils/string.js, qui fournit quelques fonctions de manipulation de String indispensables…

```
//module utils/string
//aucune dépendance pour ce module, on pourrait
omettre
//le tableau vide en premier paramètre
define([], function() {
    //ce module retourne un objet directement
exploitable
    return {
         capitalizeFirst : function (str) {
             return str.charAt(0).toUpperCase() +
str.slice(1);
         },
         isEmpty : function(str) {
             return /^[\s\xa0]*$/.test(str);
         }
    };
});//module utils/string
//aucune dépendance pour ce module, on pourrait
omettre
//le tableau vide en premier paramètre
define([], function() {
    //ce module retourne un objet directement
exploitable
    return {
         capitalizeFirst : function (str) {
             return str.charAt(0).toUpperCase() +
str.slice(1);
         },
         isEmpty : function(str) {
             return /^[\s\xa0]*$/.test(str);
         }
    };
});
```

Reste maintenant à créer une page Web qui va afficher un « Hello world » après son chargement.

```
<html>
<head>
<script>
    //on configure curl (son répertoire de base
pour les extensions et plugins
    //(voir plus loin)
    curl = {
        paths: {   curl: '../src/curl/' }
    };
</script>
<script src="../src/curl.js"
type="text/javascript"></script>

<script type="text/javascript">

curl(
    [
     'utils/string',
     'client/hello',
     //on attend la fin de chargement du DOM pour
exécuter la fonction,
     //ce module ne renvoie pas de valeur
     'domReady!'
    ],
    /**
    * fn: l'objet renvoyé par le module string
    * hello: la valeur (chaine de caractères)
renvoyée par le module hello
    */
    function (fn, hello) {

document.getElementById("welcome").innerHTML =
"Hello world apparait ci-dessous: "

+ "<br> "

+ fn.capitalizeFirst(hello);
    }
 );
</script>
</head>
<body>
    <p id="welcome"></p>
</body>
</html><html>
<head>
<script>
    //on configure curl (son répertoire de base
pour les extensions et plugins
    //(voir plus loin)
    curl = {
        paths: {   curl: '../src/curl/' }
    };
</script>
<script src="../src/curl.js"
type="text/javascript"></script>

<script type="text/javascript">

curl(
    [
     'utils/string',
     'client/hello',
     //on attend la fin de chargement du DOM pour
exécuter la fonction,
     //ce module ne renvoie pas de valeur
     'domReady!'
    ],
    /**
    * fn: l'objet renvoyé par le module string
    * hello: la valeur (chaine de caractères)
renvoyée par le module hello
    */
    function (fn, hello) {

document.getElementById("welcome").innerHTML =
"Hello world apparait ci-dessous: "

+ "<br> "

+ fn.capitalizeFirst(hello);
    }
 );
</script>
</head>
<body>
    <p id="welcome"></p>
</body>
```

```
</html>
```

Notons le module spécial préexistant, domReady!, qui s'assure que le DOM a bien fini d'être chargé. Notre fonction de rappel sera donc appelée uniquement quand toutes les dépendances auront été résolues **ET** le DOM chargé (finalement domReady! est une dépendance comme une autre). Pas belle la vie ? Si nous avons un module utils/string qui grossit, on peut très facilement le segmenter en plusieurs modules (des sous-modules) et ne charger que ceux qui sont **réellement** nécessaires.

```
curl(
    [
     'utils/string/substitutions',
     'utils/string/regexp',
     'utils/string/display',
     'client/hello',
     'domReady!'
    ],
    function (substr, regexp, disp, hello) {
       ...
    }
 );curl(
    [
     'utils/string/substitutions',
     'utils/string/regexp',
     'utils/string/display',
     'client/hello',
     'domReady!'
    ],
    function (substr, regexp, disp, hello) {
       ...
    }
 );
```

4. Les plugins sont prévus dans l'API

L'API prévoit la possibilité d'ajouter des plugins. Sincèrement, c'est aussi là que la magie opère et offre de nombreuses possibilités pour construire proprement ses applications. L'utilisation d'un plugin se déclare ainsi : 'nom_plugin!ressource_cible!options'. Les plugins « de fait », fournis par tous les loaders :

- 'js!librairie.js' : chargement d'une bibliothèque JavaScript non mise au format de module AMD ; le fichier est chargé en asynchrone, exécuté, et si la bibliothèque dispose d'une variable globale, alors elle peut être exportée comme valeur du module. Par exemple le moteur de template Mustache est écrit ainsi :

```
var Mustache =  {}; //ai simplifié cette ligne,
variable exportée
(function (exports) {
    exports.name = "mustache.js";
    exports.render = function(...) {...}
    ...
})(Mustache);var Mustache =  {}; //ai simplifié
cette ligne, variable exportée
(function (exports) {
    exports.name = "mustache.js";
    exports.render = function(...) {...}
    ...
})(Mustache);
```

- Une fois Mustache copié dans un répertoire utils, nous pouvons l'utiliser de cette façon :

```
curl(
    ['client/hello',
     'js!utils/mustache.js!order!
exports=Mustache'
    ],
    function (hello, Mustache /*reçoit la
variable exportée */) {
        ...
    }
 );curl(
    ['client/hello',
     'js!utils/mustache.js!order!
exports=Mustache'
    ],
    function (hello, Mustache /*reçoit la
variable exportée */) {
        ...
    }
 );
```

- Voir l'exemple : Lien 17 ;
- 'css!stylesheet.css' : chargement d'un module feuille de style (dépendance) injectée ensuite directement dans la page en cours. Avant d'injecter les styles, le plugin pourra les normaliser selon le navigateur cible (les hacks connus pour IE, etc.).

```
curl(
    ['client/hello',
     'css!css/widget.css'
    ],
    function (hello) {
        ...
    }
 );curl(
    ['client/hello',
     'css!css/widget.css'
    ],
    function (hello) {
        ...
    }
 );
```

- Voir l'exemple : Lien 18 ;
- 'text!template/widget.html' : chargement d'un module de texte (dépendance) injecté ensuite sous la forme d'un paramètre String.

```
curl(
    ['client/hello',
     'text!template/widget.html'
    ],
    function (hello, txtWidget ) {
        ...
    }
 );curl(
    ['client/hello',
     'text!template/widget.html'
    ],
    function (hello, txtWidget ) {
        ...
    }
 );
```

- Voir l'exemple : Lien 19 ;
- 'i18n!nls/application' : chargement d'un module i18n (dépendance) injecté ensuite sous la forme d'un objet (clé, valeur) disposant des traductions pour la locale déclarée dans la configuration ou celle configurée dans le navigateur si elle existe ;
- mais aussi d'autres loaders moins consensuels : json, cs (CoffeeScript), less, font, image…

5. Et si on mixait l'ensemble ?

Maintenant que nous avons vu le fonctionnement, reste à voir ce que cela donne concrètement dans le cadre d'un exemple, simple, mais relativement complet. Je vous propose de construire un widget de type Timeline Twitter qui affiche les n derniers tweets d'un compte Twitter.
Caractéristiques et contraintes imposées de la démo :

- code organisé en modules (avec la possibilité d'utiliser un module utils/console pour effectuer quelques sorties, tant sur IE que sur les autres navigateurs) ;
- le widget est autonome (template, CSS) ;
- le widget utilise jQuery, Mustache pour le système de templates (voir l'article de Ludovic CHAN WON IN : Lien 20), une bibliothèque JavaScript pour le formatage des dates ;
- le widget doit être chargé à la demande, pas avec la page de départ ;
- une page Web avec un formulaire (une seule zone de texte pour saisir le libellé du compte Twitter et un bouton pour obtenir l'affichage du widget), initialement sans widget affiché.

Pour vous faire une idée du résultat attendu, la démo est visible par ici : Lien 21.
Allez, hop c'est parti !
Réglons tout de suite le problème de la console : il n'existe pas de console sous IE, on crée donc un module qui crée la console du pauvre (un simple alert).

```
//module utils/console
define([], function () {
 // mock console pour IE
 if (!window.console) console = {};
 if (!('log' in console)) {
    console._msg = [];
    console.log = function (msg) {
        var _msg = this._msg;
        _msg.push([].join.call(arguments, ' '));
        clearTimeout(this._timeout);
        this._timeout = setTimeout(function () {
                          alert(_msg.join('\n'));
                        }, 100);
    };
 }

return window.console || console;
});//module utils/console
define([], function () {
 // mock console pour IE
 if (!window.console) console = {};
 if (!('log' in console)) {
    console._msg = [];
    console.log = function (msg) {
        var _msg = this._msg;
        _msg.push([].join.call(arguments, ' '));
        clearTimeout(this._timeout);
        this._timeout = setTimeout(function () {
                          alert(_msg.join('\n'));
                        }, 100);
    };
 }

return window.console || console;
});
```

Préparons maintenant la page HTML twitter.html

```
<html>
<head>
<script>
    curl = {
      paths: {
         curl : '../src/curl/',
         jquery :
'http://ajax.googleapis.com/ajax/libs/jquery/1.7.
1/jquery.min'
      }
    };
</script>
<script src="../src/curl.js"
type="text/javascript"></script>

<script type="text/javascript">
    var start = new Date();
    //on s'assure du chargement du DOM et de la
console
    curl(['utils/console', 'domReady!'],
           function (console) {
                console.log('Temps de
chargement:', new Date() - start);

document.getElementById("read").onclick =
function() {
                    //ICI LE CODE POUR AFFICHER
NOTRE WIDGET DANS LA DIV tweetsNode
                };
           }
    );

</script>
</head>
<body>
  <div>
     <label>Lire les 8 derniers tweets de
@</label>
     <input type="text" id="user"
value="ippontech" />
     <button id="read">Et hop !</button><br>
  </div>
  <div id="tweetsNode" style="margin-
top:20px"></div>
</body>
</html><html>
<head>
<script>
    curl = {
      paths: {
         curl : '../src/curl/',
         jquery :
'http://ajax.googleapis.com/ajax/libs/jquery/1.7.
1/jquery.min'
      }
    };
</script>
<script src="../src/curl.js"
type="text/javascript"></script>
```

```
<script type="text/javascript">
    var start = new Date();
    //on s'assure du chargement du DOM et de la
console
    curl(['utils/console', 'domReady!'],
           function (console) {
                console.log('Temps de
chargement:', new Date() - start);

document.getElementById("read").onclick =
function() {
                    //ICI LE CODE POUR AFFICHER
NOTRE WIDGET DANS LA DIV tweetsNode
                };
           }
    );

</script>
</head>
<body>
  <div>
     <label>Lire les 8 derniers tweets de
@</label>
     <input type="text" id="user"
value="ippontech" />
     <button id="read">Et hop !</button><br>
  </div>
  <div id="tweetsNode" style="margin-
top:20px"></div>
</body>
</html>
```

Nous pouvons noter qu'à aucun moment nous ne faisons référence à notre widget, il ne sera donc pas chargé initialement. Ensuite, nous n'avons pas besoin de jQuery pour l'exécution de cette page (c'est un exemple, bien sûr…), donc aucune raison de charger la bibliothèque. En revanche nous déclarons ligne 7 que tout module référencé par « jquery » pointe vers le CDN Google, permettant de charger/référencer la bibliothèque. Enfin, nous utilisons le module domReady! pour être certain de disposer de l'élément read quand on ajoute le onclick (ligne 19).
Reste maintenant à créer le widget. Dans notre architecture, les widgets sont déposés dans un répertoire widget. On définit alors cette arborescence et les modules associés :

- widget/twitter/timeline (.js), module principal du widget, c'est lui qui définit le widget qui sera instancié depuis la page Web ;
- widget/twitter/css/widget.css, « module » de feuille de style propre au widget ;
- widget/twitter/template/template.html, « module » de texte représentant le template Mustache.

Commençons par le template Mustache qui prend en entrée un objet JavaScript comprenant notamment un tableau de tweets, chaque tweet intégrant le nom du compte Twitter, son avatar, le texte et sa date ; on souhaite obtenir pour chaque tweet ce style :

Le fichier widget/twitter/template/template.html :

```
{{#tweets}}
    <div class="{{baseClass}}">
        <h3>{{name}}</h3>
        <img class="avatar" src="{{avatar}}" >
        <p>{{&text}}</p>
        <p class="date">{{date}}</p>
    </div>
{{/tweets}}{{#tweets}}
    <div class="{{baseClass}}">
        <h3>{{name}}</h3>
        <img class="avatar" src="{{avatar}}" >
        <p>{{&text}}</p>
        <p class="date">{{date}}</p>
    </div>
{{/tweets}}
```

Continuons par la feuille de style widget/twitter/css/widget.css :

```
/* twitterWidget est la class de base du widget
*/
.twitterWidget .avatar {
    ...
}
.../* twitterWidget est la class de base du
widget */
.twitterWidget .avatar {
    ...
}
...
```

Nous disposons du rendu, reste à coder le widget. Techniquement, il n'y a aucune difficulté, c'est un simple JSONP exposé par Twitter (l'appel est en cross-domain) qui renvoie une liste de tweets. Nous filtrerons cette liste pour n'en conserver que les informations nécessaires au template…
Définissons notre module : nous devons donner les dépendances attendues et la valeur renvoyée doit être une classe qui pourra être instanciée, en bref une fonction. Nommons la classe Timeline.

```
define(['utils/console',
        'jquery',
        'js!utils/mustache.js!exports=Mustache',
        'text!./template/template.html',
        'css!./css/widget',
        //la class de formatage des dates: elle
ajoute une fonction format
        //au prototype de Date
        'js!
http://stevenlevithan.com/assets/misc/date.format
'],
        function (console, $, mustache,
txtTemplate) {
               //Notre class à renvoyer
               function Timeline(/* le nom du
compte twitter */ twitterName,
                                 /* le nombre de
tweets à afficher */ nbTweets,
                                 /* le container
où afficher le widget */ container) {
                        this.twitterName =
twitterName;
                        this.nbTweets = nbTweets;
                        this.container = container;
               }
              //La fonction de rendu de notre
```

```
widget
            Timeline.prototype.render =
function() {

$.getJSON("http://twitter.com/statuses/user_timel
ine.json?callback=?",
                            {
                               screen_name:
this.twitterName,
                               count:
this.nbTweets
                            },
                            (function(scope) {
                                return
function(data) { scope._display(data); };
                            })(this)
                        );
              };
              //la fonction dédiée à
l'affichage, "privée"
              Timeline.prototype._display =
function(data) {
                        var tweets =
$.map(data,function(tweet) {
                           tweet.text =
tweet.text.replace(/* regexps en chaine
                             (omises) pour
faire le ménage dans le texte reçu */);
                           return {'name':
tweet.user.name,

'avatar':tweet.user.profile_image_url,
                                    'text':
tweet.text,
                                    'date': (new
Date (tweet.created_at))

.format("dd/mm/yyyy HH:MM:ss")
                                };
                        });
                        //Un peu de travail pour
Mustache...
                        var view = { baseClass:
"twitterWidget", tweets : tweets };
                        var output =
mustache.render(txtTemplate, view);
                        //on affiche le widget
avec tous ses tweets
                        $
(this.container).html(output);
                };

                //Notre class est maintenant
définie, on la renvoie
                return Timeline;
        }
);define(['utils/console',
        'jquery',
        'js!utils/mustache.js!exports=Mustache',
        'text!./template/template.html',
        'css!./css/widget',
        //la class de formatage des dates: elle
ajoute une fonction format
        //au prototype de Date
        'js!
http://stevenlevithan.com/assets/misc/date.format
'],
        function (console, $, mustache,
txtTemplate) {
                //Notre class à renvoyer
                function Timeline(/* le nom du
compte twitter */ twitterName,
                                  /* le nombre de
tweets à afficher */ nbTweets,
                                  /* le container
où afficher le widget */ container) {
                        this.twitterName =
twitterName;
                        this.nbTweets = nbTweets;
                        this.container = container;
                }
                //La fonction de rendu de notre
widget
                Timeline.prototype.render =
function() {

$.getJSON("http://twitter.com/statuses/user_timel
ine.json?callback=?",
                                {
                                   screen_name:
this.twitterName,
                                   count:
this.nbTweets
                                },
                                (function(scope) {
                                    return
function(data) { scope._display(data); };
                                })(this)
                            );
                };
                //la fonction dédiée à
l'affichage, "privée"
                Timeline.prototype._display =
function(data) {
                            var tweets =
$.map(data,function(tweet) {
                               tweet.text =
tweet.text.replace(/* regexps en chaine
                                 (omises) pour
faire le ménage dans le texte reçu */);
                               return {'name':
tweet.user.name,

'avatar':tweet.user.profile_image_url,
                                        'text':
tweet.text,
                                        'date': (new
Date (tweet.created_at))

.format("dd/mm/yyyy HH:MM:ss")
                                    };
                            });
                            //Un peu de travail pour
Mustache...
                            var view = { baseClass:
"twitterWidget", tweets : tweets };
                            var output =
mustache.render(txtTemplate, view);
                            //on affiche le widget
avec tous ses tweets
                            $
(this.container).html(output);
                    };

                    //Notre class est maintenant
définie, on la renvoie
                    return Timeline;
            }
);
```

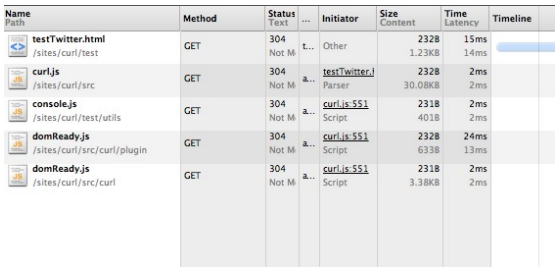
Enfin, il faut afficher notre widget dans la page HTML ; n'oublions pas que le widget est chargé à la demande (une seule fois, la première), lors du clic sur le bouton.

```
document.getElementById("read").onclick =
```

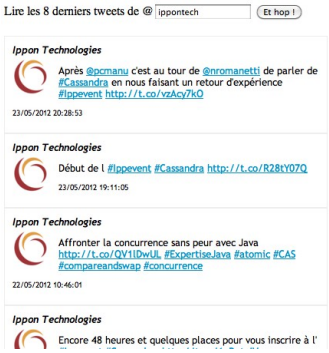
```
function() {
    //on charge notre widget (module). Il ne sera
chargé que sur le premier clic

curl(['widget/twitter/timeline'],function(Timelin
eTwitter /* la class renvoyée par
                                                le
module */) {
        var user =
document.getElementById("user").value;
        //On crée une instance du widget puis on
l'affiche
        var tm = new TimelineTwitter(user, 8,
document.getElementById("tweetsNode"));
        tm.render();
    });
};document.getElementById("read").onclick =
function() {
    //on charge notre widget (module). Il ne sera
chargé que sur le premier clic

curl(['widget/twitter/timeline'],function(Timelin
eTwitter /* la class renvoyée par
                                                le
module */) {
        var user =
document.getElementById("user").value;
        //On crée une instance du widget puis on
l'affiche
        var tm = new TimelineTwitter(user, 8,
document.getElementById("tweetsNode"));
        tm.render();
    });
};
```

Pour vous faire une idée du résultat, la démo est visible par ici.
Et… c'est terminé ! Tout est mis sous la forme de modules, correctement organisé et optimisé pour le chargement. Vérifions…
1) Arrivée sur la page : pas de widget affiché :

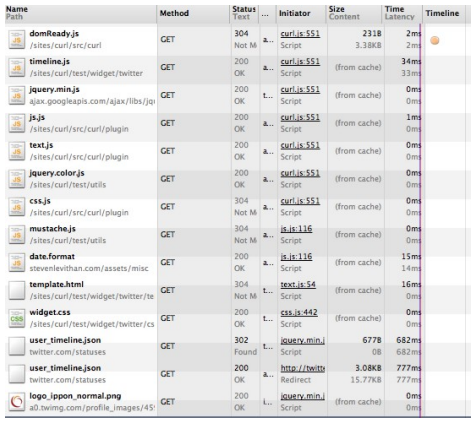
2) Les modules chargés : ni jQuery chargé, ni widget, le strict minimum :



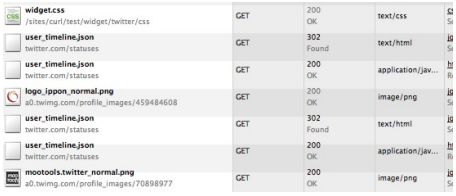
3) On lance la recherche (donc sans recharger la page) :



4) Les modules chargés en complément des précédents : jQuery et les autres dépendances déclarées par le widget, plus la timeline JSON et l'avatar du compte Twitter concerné :



5) Si on effectue une nouvelle recherche (bien sûr toujours sans recharger la page), on constate bien que seuls la nouvelle timeline et l'avatar associé au compte Twitter sont chargés



6. Si c'est si bien, pourquoi tous les projets ne sont-ils pas compatibles AMD ?

La communauté JavaScript y vient progressivement. jQuery 1.7 est compatible AMD, Mootools 2 l'est, Dojo aussi, etc. tirant vers le haut les projets liés. Si on prend par exemple le cas de **Backbone.js** (MVC basé sur jQuery), il est quasi indispensable d'avoir cette notion de modules pour associer des widgets MVC. Le projet backbone-aura.js s'en occupe.

7. La multiplication de modules ne nuit-elle pas aux performances ?

C'est un souci à prendre en considération. Effectivement chaque module étant normalement chargé individuellement, on peut craindre une dégradation des performances si le nombre de modules croit. Des dizaines, voire des centaines de modules induiront un temps de latence important dû à l'établissement de la connexion

HTTP pour chacun. Heureusement, les outils à notre disposition évacuent élégamment ce souci. Ainsi les loaders proposent des outils de builds qui assemblent plusieurs modules en un seul fichier qui passera ensuite dans un compilateur (aussi appelé shrinker ou compressor) tel que Google Closure Compiler. Finalement, on obtient un seul fichier minimisé qui regroupe n modules (chaque module conserve son nommage initial incluant son package). C'est très efficace et cela s'inclut parfaitement au sein d'un POM Maven ou d'une usine d'Intégration Continue telle que Jenkins.

8. D'autres idées d'utilisation des modules ?

- Lors d'une phase du développement, on peut facilement fournir un module mock :

```
curl = {
  paths: {
     curl   : '../src/curl/',
     jquery :
'http://ajax.googleapis.com/ajax/libs/jquery/1.7.
1/jquery.min',
     //on redirige le package widget vers ce path
qui héberge des mocks
     widget : '../mock/widget/'
  }
};

curl(['widget/timeline', 'domReady!'],
       function (TimeLine) {
         //Timeline est ici le mock
         ...
       }
);curl = {
  paths: {
     curl   : '../src/curl/',
     jquery :
'http://ajax.googleapis.com/ajax/libs/jquery/1.7.
1/jquery.min',
     //on redirige le package widget vers ce path
qui héberge des mocks
     widget : '../mock/widget/'
  }
};

curl(['widget/timeline', 'domReady!'],
       function (TimeLine) {
         //Timeline est ici le mock
         ...
       }
);
```

- Un module peut être remplacé sans délai par un autre module, sans impact pour les développeurs si les fonctions exportées portent le même nom.
- Création simplifiée de tests unitaires.
- Création d'un ensemble de ressources utilisables dans tous les projets.

9. Les « petits plus » de Curl

Curl introduit quelques fonctionnalités intéressantes issues notamment des Deferred : chainage de modules (next), fonctions de rappel en cas de succès ou d'erreur de chargement.

```
curl(['js!has.js'])
    //on est assuré du chargement préalable de la
bibliothèque has.js
    //avant d'enchainer...
   .next(['model/client', 'utils/string',
'utils/console'],
          function (client, fn, console) {
        // on prépare les données...
          }
   )
   //... on attend maintenant que le DOM soit
prêt et le composant timeline chargé
   .next(['widget/twitter/timeline',
'domReady!'])
   /* on va détecter si une erreur de chargement
de module a eu lieu, auquel cas
      on pourra intervenir */
   .then(www.hotm
       function (Timeline) {
           // tout est OK, Timeline chargée et
DOM prêt
       },
       function (ex) {
           // en cas de problème, code exécuté
       }
    );curl(['js!has.js'])
    //on est assuré du chargement préalable de la
bibliothèque has.js
    //avant d'enchainer...
   .next(['model/client', 'utils/string',
'utils/console'],
          function (client, fn, console) {
        // on prépare les données...
          }
   )
   //... on attend maintenant que le DOM soit
prêt et le composant timeline chargé
   .next(['widget/twitter/timeline',
'domReady!'])
   /* on va détecter si une erreur de chargement
de module a eu lieu, auquel cas
      on pourra intervenir */
   .then(
       function (Timeline) {
           // tout est OK, Timeline chargée et
DOM prêt
       },
       function (ex) {
           // en cas de problème, code exécuté
       }
    );
```

10. Et maintenant ?

Le cout de la mise en œuvre de cette forme de développement est ridicule par rapport à la qualité et aux performances obtenues. Il y a fort à parier que les loaders AMD vont se généraliser dans les projets et que la modularité et la gestion des conflits entre les frameworks vont rapidement progresser, permettant aisément de mélanger des composants techniques ou UI hétérogènes. À titre d'exemple le composant UI dGrid (Lien 22) prévu initialement pour le framework Dojo est complètement exploitable en environnement jQuery.
Les exemples sont consultables dans ce dépôt Github (Lien 15) ou dans l'archive jointe (Lien 16).

Retrouvez l'article d'Emmanuel Remy en ligne : Lien 23

Apple présente l'iPhone 5S et iPhone 5C et la sortie d'iOS 7

Apple a présenté comme à son habitude du mois de septembre, l'arrivée des nouveaux iPhone et de son OS. Au programme de ce keynote, la sortie d'iOS 7, de quelques applications iWorks et iLife gratuites pour iOS, un peu de couleur avec l'arrivée de l'iPhone 5C disponible en cinq couleurs et de l'iPhone 5S avec une version "Or".

iOS 7

Apple proposera donc la nouvelle version d'iOS 7 au public le 18 septembre pour iPhone, iPad et iPod touch (5G).
Pour connaitre toutes les nouveautés d'iOS 7, vous pouvez consulter le compte-rendu de la keynote de la WWDC du mois de juin (Lien 24) ou directement le site d'Apple.

Lien 25

Apps Apple gratuites

La société a annoncé que toutes ces applications pour iOS des suites iWorks et iLife sont maintenant gratuites.
Il sera donc maintenant possible d'avoir les applications iPhoto, iMovie, Keynote, Pages et Numbers gratuitement sur iOS.

iPhone 5C

L'iPhone 5C, « C » comme « color », qui est tout simplement un iPhone 5, mais disponible en cinq couleurs (vert, bleu, jaune, rose et gris).

- 5 couleurs : bleu, gris, jaune, vert, rose ;
- capacité : 16 Go et 32 Go ;
- processeur A6 ;
- appareil photo iSight 8 Mpx.

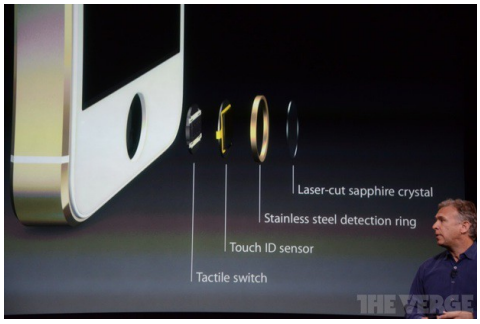
L'iPhone 5C est en précommande à partir du 13 septembre et disponible en vente le 20 septembre.
Le prix est de 599 € pour la version 16 Go et de 699 € pour la version 32 Go.

Lien 26

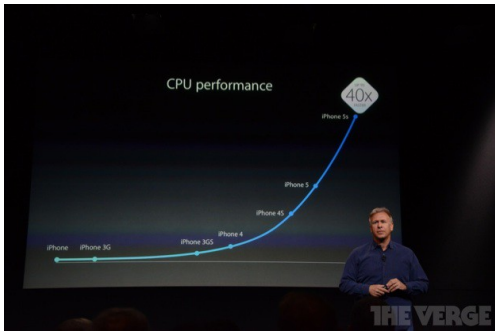
iPhone 5S

L'iPhone 5S est le nouveau téléphone haut de gamme d'Apple avec une mise à jour hardware de qualité.

Un nouveau processeur A7 avec architecture 64 bits, un coprocesseur de mouvement M7, support de l'OpenGL 3.0, nouvelle caméra et appareil photo et d'une nouvelle fonctionnalité appelée Touch ID qui est en fait un capteur d'empreinte digitale.



Des performances annoncées comme un CPU et GPU 2x plus rapide, et une puce M7 qui gère séparément toutes les données de mouvement comme l'accéléromètre, gyroscope, boussole et permettra à toutes les applications d'activité physique de l'utiliser sans solliciter la puce A7.



- trois couleurs : gris sidéral, or, argent ;
- capacité : 16 Go, 32 Go et 64 Go ;
- processeur A7 avec architecture 64 bits ;
- coprocesseur de mouvement M7 ;
- appareil photo iSight 8 Mpx ;
- meilleure autonomie ;
- capteur d'identité par empreinte digitale.

L'iPhone 5S sera disponible à la vente le 20 septembre de 699 à 899 € selon la capacité de stockage choisie.

Lien 27

Commentez la news d'Aurélien Gaymay en ligne : Lien 28

Introduction aux POSIX MQ

Explications sur l'usage (avec des exemples complets) des POSIX MQ et leurs équivalents dans l'industrie.

1. Introduction

Les POSIX MQ, ou POSIX Message Queues, également traduisibles en Files de Messages, sont un des quelques mécanismes d'IPC (InterProcess Communications).
Le terme file, MQ ou POSIX MQ sera employé dans l'article pour les désigner.
Ses principaux homologues seraient les Named Pipe/FIFO (tubes nommés) et les sockets.
Les POSIX MQ sont apparues pour standardiser les précédentes MQ de AT&T System V (cf. msgget, msgctl, msgsnd, msgrcv).
Contrairement aux tubes ou aux sockets, elles utilisent des modules « temps réel » du noyau : en pratique dans l'industrie, elles sont utilisées pour acheminer des informations importantes entre processus (et même entre machines), cela explique l'importance de la réactivité malgré le fonctionnement global asynchrone.

2. Description

Les MQ fonctionnent, comme leur nom l'indique, comme un service de gestion de files. Les files contiennent des messages avec des priorités variables, et les messages les plus prioritaires et les plus anciens sont lus en premiers. Les files sont approvisionnées par un ou des processus, et sont lues par un ou des processus.
Les MQ contiennent des propriétés qui peuvent être locales à la file, ou globales à leur fonctionnement sur le système. Par exemple, on peut fixer localement le nombre maximum de messages que la file pourra contenir, mais c'est le système qui fixe la priorité maximale qu'un message peut avoir.
Les files contiennent des messages dont la longueur maximale est fixe. Cette dernière information nous permettra une implémentation beaucoup plus facile avec un buffer de taille fixe, ou des chaines de tailles fixes.
POSIX impose quelques règles lors de la création des MQ quant à leur nom. Celles-ci doivent « au moins » commencer par un slash (« / »), et être suivies de caractères alphanumériques. Si d'autres slashs suivent, alors l'implémentation dépend du système d'exploitation.
POSIX n'impose aucune règle sur le stockage des MQ, la plupart des *n*x (Linux, UNIX, UNIX-likes, etc.) les stockent directement sur le FS.

3. Spécificités sur certains OS

- Depuis le kernel 2.6 (environ), les POSIX MQ sont implémentées dans la plupart des Linux, et aucune manipulation spécifique n'est nécessaire pour les activer. L'option kernel Debian est : « **CONFIG_POSIX_MQUEUE** ». Debian 7.0 ne gère pas les threads lors d'un mq_notify, mais il le fait silencieusement.
- Depuis FreeBSD 7, les POSIX MQ sont partiellement disponibles [pas de gestion des THREAD sur mq_notify via crash de l'application]. FreeBSD 9.0 intègre les THREAD, mais les événements restent enregistrés en mémoire même après leur accomplissement… dans tous les cas, une manipulation est nécessaire pour activer les POSIX MQ dessus. Tout d'abord, activer le module noyau pour les gérer : « **kldload mqueuefs** » (ou recompiler avec l'option « **options P1003_1B_MQUEUE** »), puis les monter sur le FS :

```
su
mkdir /mqueue
mount -t mqueuefs null /mqueue
```

- Sur Cygwin 1.7, les POSIX MQ sont disponibles et activées de base, mais aucun événement ne sera remonté par mq_notify [juillet 2013].
- Sur MingW32, les POSIX MQ ne sont PAS disponibles.
- Visual Studio « sans extension » ne gère évidemment pas les POSIX MQ (certains Windows peuvent être POSIX ou partiellement POSIX sous certaines conditions). Il faut utiliser sur cette plateforme, soit Cygwin, soit des logiciels tiers payants, mais offrant beaucoup plus de fonctionnalités (messages intermachines et même intersystèmes).

4. Comparaison avec d'autres IPC

4.1. OpenMPI/Passage de Messages

Sur le forum, lors de la rédaction de cet article, une question très intéressante a été posée : « Quelle est la différence avec OpenMPI ? »
Cette question est très intéressante du fait que l'idée des MP/MPI est de partager des calculs par l'intermédiaire de « passage de messages ». Voici la réponse qui a été apportée :

- « Les POSIX MQ restent en « local » sur la machine (c'est l'inconvénient des System V MQ et POSIX MQ), mais elles sont « normalement » disponibles sur tous les systèmes POSIX ou SUS, et sont totalement gratuites (contrairement à beaucoup de produits travaillant sur les « MQ », mais qui eux travaillent sur plusieurs machines en parallèle comme MPI). En cherchant plus de détails, OpenMPI parait extrêmement plus

complet que les « MQ » classiques. Il y a UNE différence majeure qui change tout, et cela est lié à l'usage des MQ/Communication par rapport au MP/Parallélisme. »

Le MPI propose à l'envoi :

```
int MPI_Send (
        message,        /* actual information
sent */
        length,         /* length of the message
*/
        datatype,       /* MPI datatype of the
message */
        destination,    /* rank of the processor
getting the message */
        tag,            /* tag helps sort
messages - likely an int and the same as in
MPI_Recv */
        MPI_Comm        /* almost alway
MPI_Comm_World */
);
```

D'après ces slides (Introduction MPI Programming : Lien 29), on voit que le « rank » sert à identifier les processus qui vont exécuter une tâche, et que le « tag » sert à identifier une suite de messages liés entre eux… et SURTOUT (slide 25) :

Used to ensure messages are read in the right order
- standard says nothing about the order of message arrival

Ceci est la différence avec les MQ.
Les MQ assurent l'ordre d'envoi/arrivée ainsi que la gestion des priorités, tandis que les MP/MPI assurent la distribution et l'échange de structures au sein d'un système.
Le but du MP/MPI étant de calculer, et garder un « ensemble logique » de données entre elles, il faut surtout que tous les messages représentant la même donnée arrivent à un moment ou à un autre pour conserver l'intégrité de celle-ci, quelle que soit sa taille. À l'inverse, sur les MQ, on n'essaye pas d'envoyer des structures, mais simplement des messages (à but informatif), dont l'ordre d'arrivée est important.

Les points semblables entre MQ et MP/MPI :

- échange de données entre processus/machines : une MQ peut être lue/écrite par plusieurs processus (voire machines sur les produits commerciaux) ; plusieurs thread/processus/machines communiquent via plusieurs routines send/recv/sendrecv sur OpenMPI ;
- identification des communications : le nom d'une MQ sert à l'identifier sur l'ensemble du système, ensemble rank + tag permet de séparer les tâches et destinataires en « groupes ».

Les différences entre MQ et MP/MPI :

- les MQ travaillent sur l'ordre d'arrivée (paramètre priority) : on envoie avec une priorité au choix, on reçoit nécessairement le message le plus ancien avec la priorité la moins basse (donc priorité max sortira TOUJOURS en premier) ;
- le but d'une MQ est d'informer/transférer une petite donnée plutôt que de partager du calcul/des structures en mémoire ;
- les MQ (commerciales) se trouvent sur des systèmes d'informations « professionnels » pour coordonner l'ensemble des progiciels entre eux et remonter au plus vite une alerte (communications entre micro-ordinateurs, miniordinateurs et mainframes), tandis que les MP/MPI servent au calcul massivement parallèle sur des super calculateurs (grilles de calculs).

4.2. Pipes/Sockets/FIFO

Les pipes et les sockets permettent des échanges entre seulement 2 interlocuteurs, il est possible de faire des communications entre plus d'interlocuteurs, mais de nombreux autres pipes/sockets doivent être ouvert(e)s, toute leur gestion implique beaucoup d'appels système.
Les FIFO permettent des échanges entre N interlocuteurs, mais, il faut au moins qu'un interlocuteur soit en écriture et un en lecture avant de pouvoir envoyer des données, les priorités ne sont pas gérées, et la FIFO disparait après que tous les interlocuteurs ont fini leurs communications.
On peut donc retenir que les POSIX MQ sont créées et « existent » (ainsi que leur contenu) tant qu'elles ne sont pas détruites (impossible avec les FIFO, pipes et sockets).
Les priorités et l'ordre d'arrivée assuré permettent de conserver l'arrangement des messages en fonction de l'importance.

5. Utilisation Basique des POSIX MQ

Les POSIX MQ servent essentiellement à transférer des messages courts entre plusieurs processus.
Il est tout de même possible de lire ses propres messages.
La lecture d'une file est « destructrice » : une fois un message lu, il est retiré de la file.
La file extraira le message le plus prioritaire stocké en premier automatiquement. C'est à l'écriture que l'on indiquera la priorité du message.
POSIX impose au minimum 32 niveaux de priorités, et suggère que tous les messages « normaux » passent avec la priorité 0, et seuls les messages réellement « urgents » passent avec des priorités supérieures.
Tous les OS n'ont pas les mêmes priorités ! Par exemple, FreeBSD se contente de 64 niveaux de priorités, tandis que Linux propose plus de 32 000 d'entre eux.
Le maximum doit se trouver dans la macro **MQ_PRIO_MAX**, mais celle-ci n'est pas toujours disponible.

FreeBSD la met à disposition des développeurs tiers, mais Debian ne la conserve que dans les sources noyaux, nécessitant un appel à « **sysconf(_SC_MQ_PRIO_MAX)** ». Voici un exemple de code permettant d'accéder à la valeur sur plusieurs OS :

```
# ifndef MQ_PRIO_MAX
    g_mq_prio_max = sysconf(_SC_MQ_PRIO_MAX);
# else
    g_mq_prio_max = MQ_PRIO_MAX;
# endif
```

Les fonctions pour accéder aux POSIX MQ sont très similaires aux fonctions des autres IPC : ouverture/fermeture, lecture/écriture. La principale différence réside dans le fait que les MQ travaillent avec

des messages de longueurs fixes, ainsi qu'avec un numéro de priorité.
Les prototypes des 5 principales fonctions selon l'Open Group :

```
mqd_t mq_open(const char *, int, ...);
ssize_t mq_receive(mqd_t, char *, size_t,
unsigned *);
int mq_send(mqd_t, const char *, size_t,
unsigned);
int mq_close(mqd_t);
int mq_unlink(const char *);
```

5.1. mq_open

Avant de travailler sur une MQ, il faut évidemment l'ouvrir ou la créer avec la fonction mq_open(). La fonction mq_open() ne supprime AUCUN message, et n'en crée aucun. Cette fonction renvoie un identificateur de file qui est similaire à un file descriptor pour les fichiers : s'il est égal à -1, alors l'ouverture/création a échoué (voir errno pour les détails), sinon l'appel a réussi.

```
mqd_t mq_open(const char *, int, ...);
```

Le 1er argument est le nom de la MQ. Le nom DOIT commencer par un slash « / », et contenir une chaine alphanumérique. POSIX impose que l'ouverture d'un même nom commençant par un slash « / » ouvre la même MQ, quel que soit le processus.
Si d'autres slashs « / » sont présents dans le nom (ou aucun), POSIX ne donne aucune consigne, c'est à l'OS de décider comment réagir. Certains OS vont suivre ce chemin dans l'arborescence réelle, d'autres dans la partie dédiée aux MQ… à vous de vous renseigner sur la façon dont votre OS gère ces noms (s'il les gère !).
L'apparition dans le FS de la MQ n'est pas obligatoire. Mais si la gestion des MQ est faite avec des file descriptors classiques, alors le maximum de MQ ouvrables sera défini par la macro **OPEN_MAX**, en comptabilisant les autres file descriptors classiques.

Le 2e argument est un ensemble de flags similaires à ceux que l'on trouve pour les fichiers :

- **O_RDONLY :** la file est ouverte en **lecture** seulement, donc pour lire. On pourra donc utiliser la fonction mq_receive(). Un même processus peut ouvrir plusieurs fois la même MQ avec différents droits ! ;
- **O_WRONLY :** la file est ouverte en **écriture** seulement, on pourra donc utiliser mq_send() ;
- **O_RDWR :** ouvre la file en **lecture** et en **écriture**. Il est donc possible d'utiliser mq_send() et mq_receive() ;
- **O_CREAT :** permet de créer une MQ. Voir l'encadré plus bas pour les détails, car ce flag nécessite que mq_open() prenne 2 paramètres supplémentaires. Si la MQ existe déjà, et que le flag O_EXCL n'est pas activé, ce flag est ignoré ;
- **O_EXCL :** avec O_CREAT, il permet d'empêcher l'ouverture de la file si celle-ci existe. Le comportement de O_EXCL sans O_CREAT n'est pas décrit (à ne pas tester donc) ;
- **O_NONBLOCK :** permet de dire si mq_send() et mq_receive() doivent attendre que les ressources nécessaires soient disponibles ou non (mq_send() attendra que la file dispose d'une place supplémentaire pour le message, et mq_receive() attendra qu'un message arrive dans la file). Si les ressources ne sont pas disponibles, les deux fonctions échouent et placent errno à EAGAIN.

ATTENTION !
Avec le flag O_CREAT, la fonction prend 2 autres paramètres (soit 4 au total) OBLIGATOIREMENT !
mode_t mode
mq_attr *attr
Le **mode_t** sera le droit d'accès (en nombre chmod). Le **mq_attr** sera une structure spécifique à la file permettant de la configurer :

```
struct mq_attr
{
  long mq_flags;        /* Flags de la file */
  long mq_maxmsg;       /* Nombre maximum de
messages dans la file */
  long mq_msgsize;      /* Taille maximale de
chaque message */
  long mq_curmsgs;      /* Nombre de messages
actuellement dans la file */
};
```

Il est possible de laisser cette structure à NULL, dans le passage en paramètre à mq_open(), dans ce cas des droits par défaut seront appliqués et dépendant du système sur lequel le programme tournera. Si la structure n'est pas NULL, le processus appelant doit avoir les droits de créer la file, sinon mq_open() échouera.

5.2. mq_send

L'écriture dans la file se fait en envoyant un message. Il n'est pas nécessaire de préparer de forme spécifique au message, c'est mq_send() qui se charge de créer la structure.
La fonction renvoie 0 en cas de succès, sinon -1.

```
int mq_send(mqd_t, const char *, size_t,
unsigned);
```

Le 1er argument est le descripteur de la MQ (on l'a reçu lors du mq_open()).

Le 2e argument est le message que l'on souhaite envoyer. Il ne doit pas être plus grand que la taille déclarée à l'initialisation de la file !

Le 3e argument est la taille du message envoyé.

Le 4e argument est la priorité avec laquelle on souhaite envoyer le message. Par défaut, il est conseillé de mettre une priorité de 0 pour les messages classiques, et plus si l'information à transmettre n'est pas prévue dans le cadre du fonctionnement normal. Par exemple : plusieurs services émettent des demandes/traitements classiques, puis subitement une alerte doit être remontée. Les demandes seront de priorité 0, et l'alerte pourra être de priorité 1 à 31.

REMARQUE
Si O_NONBLOCK n'est pas activé, l'appel à mq_send() sera bloquant tant que la file est pleine, ou qu'aucun signal ne l'interrompt.
Si O_NONBLOCK est mis et que la file est pleine, une erreur est remontée.

5.3. mq_receive

La lecture (destructrice) d'un message se fait avec mq_received(). On est sûr que le message qui sera extrait par mq_receive() sera celui de plus haute priorité présent dans la file, et le plus ancien. La valeur de retour est le nombre de caractères lus, ou -1 en cas d'échec.

```
ssize_t mq_receive(mqd_t, char *, size_t,
unsigned *);
```

Le 1er argument est le descripteur de MQ (récupéré avec mq_open()).

Le 2e argument est un buffer de taille suffisante pour réceptionner le message. Étant donné que les MQ ont une taille maximale fixe, il est facile à la compilation d'inclure un buffer fixe (même chose à l'exécution avec malloc). Si le buffer est trop petit, le message ne sera pas lu ni détruit.

Le 3e argument est la taille du buffer. Si cette taille est inférieure à la taille donnée à la file à sa création (via mq_attr.mq_msgsize), alors l'appel échoue.

Le 4e argument est un pointeur vers un entier non signé. Précisément, si l'argument n'est pas NULL, mq_receive() va écrire dedans la priorité du message réceptionné. On peut donc créer une variable locale et la passer par référence à la fonction.

REMARQUE
Si O_NONBLOCK n'est pas activé, mq_receive() sera bloquant tant qu'un message n'arrive pas ou qu'un signal l'interrompt.
Si O_NONBLOCK est actif, et que la file est vide, aucune action n'est effectuée, et l'appel échoue.

5.4. mq_close

La fermeture d'une file ne sert qu'à retirer le descripteur de file associé au processus. La file n'est PAS détruite suite à mq_close() ! Le retour indique seulement si la fermeture a réussi (0) ou non (-1).

```
int mq_close(mqd_t);
```

Le seul paramètre est un descripteur de MQ. Il doit évidemment être celui d'une file précédemment ouverte.

5.5. mq_unlink

Une fois la MQ fermée, il est possible de la supprimer avec mq_unlink(). Si la MQ est encore référencée par d'autres processus ou descripteurs, elle sera supprimée après que la dernière référence a été fermée, et l'appel à mq_unlink() ne sera pas bloquant. La fonction renvoie 0 si tout a bien fonctionné, sinon -1.

```
int mq_unlink(const char *);
```

Le seul paramètre est le nom de la file.

5.6. Exemple Complet Simple

gcc -Wextra -Wall -Werror -c main.c
gcc -lrt main.o -o MQTestSimple

main.h

```
#ifndef MAIN_H_
# define MAIN_H_

# include <unistd.h>
# include <stdlib.h>
# include <stdio.h>
# include <fcntl.h>      /* For O_* constants */
# include <sys/stat.h>   /* For mode constants */
# include <mqueue.h>
# include <string.h>

#endif /* !MAIN_H_ */
```

main.c

```
# include "main.h"

# define MQ_NAME "/queuetest"
# define BUF_LEN 512

int main(void)
{
  struct mq_attr        attr;
  ssize_t               len_recv;
  mqd_t                 My_MQ;
  char                  *text;
  char                  recv[BUF_LEN];

  attr.mq_flags = 0;
  attr.mq_maxmsg = 10;
  attr.mq_msgsize = BUF_LEN;
  attr.mq_curmsgs = 0;

  My_MQ = mq_open(MQ_NAME, O_RDWR | O_CREAT,
0644, &attr);
  if ((int) My_MQ == -1)
  {
    perror("Error : mq_open failed !\n");
    return(-1);
  }

  printf("%s\n", "Envoi msg de priorite 42");
  text = strdup("Message test prio : 42 !");
  mq_send(My_MQ, text, strlen(text), 42);
  free(text);

  printf("%s\n", "Envoi msg de priorite 21");
  text = strdup("Message test prio : 21 !");
  mq_send(My_MQ, text, strlen(text), 21);
  free(text);

  memset(recv, 0, BUF_LEN);
  len_recv = mq_receive(My_MQ, recv, BUF_LEN,
NULL);
  printf("Premier msg recu (len : %u) : %s\n",
(unsigned int) len_recv, recv);

  memset(recv, 0, BUF_LEN);
  len_recv = mq_receive(My_MQ, recv, BUF_LEN,
```

```
NULL);
  printf("Deuxieme msg recu (len : %u) : %s\n",
(unsigned int) len_recv, recv);

  mq_close(My_MQ);
  mq_unlink(MQ_NAME);

  return (0);
}
```

6. Utilisation Avancée des POSIX MQ

La partie avancée se penchera sur les autres fonctions mises à disposition par POSIX pour ses MQ. Celles-ci servent à récupérer et modifier certaines propriétés de la file (en pratique uniquement le mode d'accès), être informé quand la file passe de l'état vide à l'état non vide, et envoyer/récupérer les messages avec un timeout maximum.

```
int mq_getattr(mqd_t, struct mq_attr *);
int mq_setattr(mqd_t, const struct mq_attr *
restrict, struct mq_attr * restrict);
ssize_t mq_timedreceive(mqd_t, char * restrict,
size_t, unsigned * restrict, const struct
timespec * restrict);
int mq_timedsend(mqd_t, const char *, size_t,
unsigned, const struct timespec *);
int mq_notify(mqd_t, const struct sigevent *);
```

ATTENTION !
Certaines fonctions et structures dans la partie avancée nécessitent des flags spéciaux à la compilation ! Sur Cygwin 1.7 et FreeBSD 7, ils n'étaient pas nécessaires, mais sur Debian 7 si.
Voici les flags à ajouter à la compilation :
-D_XOPEN_SOURCE=600
-D_POSIX_C_SOURCE=200112L

6.1. mq_getattr

Cette fonction sert à récupérer les propriétés de la file pour connaitre le nombre maximum de messages que l'on peut laisser en attente de traitement, leur longueur maximale autorisée au sein de la file, etc. La fonction renvoie un entier, 0 en cas de succès, sinon -1.

```
int mq_getattr(mqd_t, struct mq_attr *);
```

Le 1er paramètre est le descripteur de la file dont on souhaite récupérer les propriétés.

Le 2e paramètre est un pointeur sur une structure mq_attr, qui sera rempli avec les propriétés. Il parait approprié d'utiliser une référence sur une variable locale si un ou quelques tests uniques (non répétitifs) sont effectués.
Voici un rappel de la structure mq_attr :

```
struct mq_attr
{
  long mq_flags;        /* Flags de la file */
  long mq_maxmsg;       /* Nombre maximum de
messages dans la file */
  long mq_msgsize;      /* Taille maximale de
chaque message */
  long mq_curmsgs;      /* Nombre de messages
actuellement dans la file */
};
```

5.2. mq_setattr

La fonction permet de modifier **certains** paramètres de la file lors du fonctionnement du programme.
Un paramètre permet notamment de sauvegarder l'ancien état des propriétés. Les valeurs contenues dans mq_maxmsg, mq_msgsize et mq_curmsgs sont ignorées, on en déduit donc que seuls les flags d'accès sont modifiés (mq_flags). L'appel à cette fonction renvoie l'entier 0 en cas de succès, et -1 en cas d'échec.

```
int mq_setattr(mqd_t, const struct mq_attr *
restrict, struct mq_attr * restrict);
```

Le 1er argument est le descripteur de la file dont on souhaite modifier les propriétés.

Le 2e est un pointeur sur structure mq_attr, contenant les **nouvelles** valeurs. Une variable locale passée en référence est suffisante dans la plupart des cas.

Le 3e argument est aussi un pointeur sur structure mq_attr, et il peut être mis à NULL. Celui-ci sert à sauvegarder l'état précédent de la file ! Plutôt que de faire deux appels successifs à mq_getattr puis mq_setattr, il est possible de sauvegarder l'état précédent en même temps que l'on en écrit un nouveau. Il est vivement conseillé d'utiliser malloc dans ce cas, et de sauvegarder cet état quelque part à long terme.

5.3. mq_timedsend

La fonction agit de façon très similaire à mq_send(), mais diffère uniquement lorsque la file est pleine et que O_NONBLOCK est inactif : dans ce cas, la fonction patientera « au maximum » le temps déclaré dans le dernier argument (instant absolu en secondes au format time_t UNIX). La clock utilisée sera prioritairement celle utilisée par le module temps réel, mais si celle-ci n'existe pas, la clock système sera utilisée. Si la file n'est pas pleine, le message est obligatoirement posté, que le temps maximum soit révolu ou non. Si la fonction réussit, elle renvoie l'entier 0, sinon -1.

```
int mq_timedsend(mqd_t, const char *, size_t,
unsigned, const struct timespec *);
```

Le 1er argument, comme pour mq_send(), est le descripteur de la file que l'on souhaite utiliser.

Le 2e argument est un pointeur vers le message à envoyer.

Le 3e argument est la taille du message à envoyer (ne doit pas dépasser la taille maximum définie à la création de la file).

Le 4e argument est la priorité du message (ne doit pas dépasser le maximum).

Le 5e argument, unique à mq_timedsend(), est un pointeur sur structure timespec. Ce temps est un instant fixe à définir avec la structure. Si la file n'est pas pleine, l'argument est ignoré et le message posté. Voici la structure timespec :

```
struct timespec
{
```

```
  time_t tv_sec;        /* Temps en secondes
depuis le 1er janvier 1970 */
  long tv_nsec;         /* Temps en nanosecondes
*/
};
```

6.4. mq_timedreceive

Tout comme mq_receive(), la fonction effectue les mêmes opérations de la même façon, excepté lorsque la file est vide et que O_NONBLOCK est inactif. Dans ce cas précis, la fonction patientera jusqu'au moment indiqué par le dernier argument, et si celui-ci est dépassé, alors la fonction échoue. Si l'instant absolu déclaré est dépassé, mais que la file n'est pas vide, la fonction extrait malgré tout un message et ignore le dernier paramètre. La fonction renvoie la taille du message qu'elle a extrait, ou -1 en cas d'échec.

```
ssize_t mq_timedreceive(mqd_t, char * restrict,
size_t, unsigned * restrict, const struct
timespec * restrict);
```

Le 1er argument est le descripteur de file.

Le 2e argument est un buffer où sera copié le message. Il doit être suffisamment long et peut être retrouvé grâce à mq_getattr().

Le 3e argument est la taille du buffer. Si la taille est inférieure à mq_msgsize, définie à la création de la file et disponible avec mq_getattr(), alors l'appel échoue.

Le 4e argument est un pointeur sur unsigned qui sera rempli avec la priorité du message récupéré. Il vaut mieux déclarer localement une variable et la passer en référence.

Le 5e argument est un pointeur sur structure timespec. Cette structure définit l'instant maximum que la fonction attendra si la file est vide et que O_NONBLOCK est inactif. Si la file n'est pas vide, et que le maximum est déjà dépassé, la fonction ignore ce paramètre et extrait un message. Voir l'explication avec mq_timedsend() pour les détails.

6.5. mq_notify

Cette fonction permet d'être informé quand une file vide ne l'est plus (quand une file vide commence à être alimentée). Attention, si un thread est en attente sur un mq_receive(), alors le thread sera débloqué, et aucune notification ne sera remontée via le mq_notify(). Le mécanisme de notification est lié aux signaux, il faudra donc penser à inclure <signal.h>. Un seul et unique processus peut s'enregistrer pour scruter une file. Une fois la notification remontée le processus est automatiquement désabonné. Si un processus s'enregistre, il peut se désinscrire et permettre qu'un autre s'enregistre.
La fonction renvoie l'entier 0 si elle a réussi, sinon -1. Les causes d'échecs peuvent être que la file désignée n'existe pas (errno à EBADF), ou qu'un autre processus est déjà inscrit (errno à EBUSY).

```
int mq_notify(mqd_t, const struct sigevent *);
```

Le 1er argument est le descripteur de la file que l'on souhaite scruter.

Le 2e argument est un pointeur sur structure sigevent. Il est possible de mettre ce paramètre à NULL afin de se désinscrire de la notification. Pour s'inscrire voici le contenu d'un sigevent :

```
struct sigevent
{
  int sigev_notify;
/* Type de notification */
  int sigev_signo;
/* Numero du signal */
  union sigval sigev_value;
/* Valeur du signal */
  void (*) (union sigval) sigev_notify_function;
/* Fonction de notification */
  (pthread_attr_t *) sigev_notify_attributes;
/* Attributs de la notification */
};
```

La structure sigevent doit contenir au moins le membre sigev_notify avec une des 4 valeurs suivantes :
SIGEV_NONE, **SIGEV_SIGNAL**, **SIGEV_THREAD**, [SIGEV_THREAD_ID - spécifique à Linux].

- **SIGEV_NONE :** rien ne se passe quand l'événement se produit. La seule utilité dans le cadre des MQ pourrait être d'empêcher tous les autres processus d'effectuer un notify.
- **SIGEV_SIGNAL :** envoie un signal lorsque l'événement se produit. Le numéro du signal se trouve dans le 2e membre de la structure sigevent : sigev_signo. Le signal envoyé portera quelques informations transmises depuis la structure sigevent (que vous remplissez lors de l'appel à mq_notify()). Les informations se trouvent dans la structure siginfo_t en lecture lorsque le signal est émis :
  - siginfo_t->si_code : valeur que l'API appelante remplit ;
  - siginfo_t->si_signo : numéro du signal (contenu dans sigevent->sigev_value) ;
  - siginfo_t->si_value : valeur que l'utilisateur (vous) remplit des informations qu'il souhaite depuis le champ sigevent->sigev_value (de type union sigval)

```
union sigval
{
  int sival_int;
  void *sival_ptr;
};
```

On comprend donc qu'il faut également ajouter un sigaction du côté de l'application pour récupérer ce signal et effectuer une action « utile ». L'union passée dans la structure permet de passer un entier 32 bits ou un pointeur 32/64 bits, ce qui peut être très utile.

- **SIGEV_THREAD :** crée un thread (si sigev_notify_attributes n'est pas NULL et contient une structure pthread_attr_t correcte permettant la création d'un thread) ou simule un thread (appel forcé de la fonction, comme un handler de signal) avec comme fonction d'appel celle définie dans sigev_notify_function et en paramètre le contenu de sigev_value (voir l'union dans le paragraphe précédent avec l'entier 32 bits

ou le pointeur 32/64 bits).

- SIGEV_THREAD_ID : spécifique à Linux. Je ne me suis pas renseigné dessus, à tester.

7. Projet Exemple : Utilisation Complète des POSIX MQ

Dans le projet inclus avec l'article (Lien 30), vous trouverez plusieurs exemples :

- **./MQTest ex1 :** permet de lancer l'exemple simple de communication entre deux processus. Un processus père écrit plusieurs messages avec des priorités différentes, et un processus fils les lit. Deux fichiers sont créés, parent.log qui contient les messages envoyés par le père, et child.log contient les messages que le fils lit dans la MQ. L'utilité est de voir [sur FreeBSD] la limite maximale du niveau de priorité, et comment les messages de priorité élevée passent « devant » les messages de priorité faible.
- **./MQTest ex2 :** permet de lancer l'exemple plus complexe utilisant les mq_timedreceive et mq_timedsend, ainsi que mq_notify avec les 3 méthodes de gestion d'événements. On écrit 11 messages dans une MQ de taille 10, les mq_timedsend échouent au bout de 3 secondes, et des événements sont déclenchés lorsque la file passe de l'état vide à l'état non vide. Sur FreeBSD 9.0, on peut voir que l'événement est conservé en mémoire tant qu'aucun gestionnaire n'est mis en place, tandis que sur Debian 7.0, uniquement le « moment » de l'événement permet de déclencher le mécanisme associé. Debian 7.0 ne gère pas non plus les threads. Cygwin ne remonte aucun événement [juillet 2013]. Dans les sources, ex2/notify_* correspondent aux handlers (préparation de la récupération du signal, et de la fonction à envoyer dans le thread), tout le fonctionnement se trouve dans ex2/simulator_ex2.c.

Pour utiliser le projet, il suffit d'extraire, configurer, compiler, puis le lancer :

```
tar xvf boissi_f-MQAdvancedExample.tar.bz2
./configure
make
./MQTest
```

8. Autres MQ, produits commerciaux et usages professionnels

Dans l'industrie, on entendra parler de « **MOM** » (**Message-Oriented Middleware**) pour désigner les logiciels permettant la communication intermachines via des MQ au sein de bus **SOA** (**Service Oriented Architecture**), ou dans le cadre d'**ERP** (**Enterprise Ressource Planning**).
Le but des MQ, en pratique, est de remonter des informations indifféremment des plateformes matérielles.
En effet, en entreprise, de nombreux OS sont présents sur diverses architectures matérielles, il existe TCP/IP pour transporter des paquets d'informations, mais la gestion de l'ordre et des priorités n'est pas assurée. Avec les MQ, ces propriétés sont assurées.
On peut ainsi remonter de petites informations ou des alertes de dysfonctionnement logiciel via ce mécanisme (les dysfonctionnements réseau seront gérés par SNMP de préférence).

Chez **IBM** on trouvera le très célèbre **MQSeries/WebSphere MQ** implanté depuis 1992 au sein des entreprises souhaitant faire communiquer leurs Mainframes, Minis et UNIX pour permettre à plusieurs applications « métier » de fonctionner ensemble. Il permet aujourd'hui de communiquer avec quasiment tous les autres services de files de messages.
Depuis 1997, **Microsoft** a mis à disposition des développeurs le service **MSMQ (Microsoft Message Queuing)** disponible à l'époque sur Windows 95 et NT4.0. Ce service permet à plusieurs machines d'échanger des messages, plus récemment les versions embarquées de Windows l'incluent.
JMS (**Java Message Service**) est l'équivalent disponible sur Java depuis 2001, maintenu aujourd'hui par **Oracle**. Il s'agit d'une API implémentée et utilisée par d'autres services.
Amazon SQS est un service payant à l'usage proposé par Amazon permettant l'échange de messages via MSMQ ou JMS.
Apache ActiveMQ et **RabbitMQ** sont aussi des exemples appartenant au monde du Libre.
Beaucoup d'autres sont disponibles, ceci est une liste non exhaustive.

Un projet intéressant visant à réduire la taille des communications tout en offrant un service de MQ existe et est à l'étude : MQTT - MQ Telemetry Transport (Lien 31).
Le but du projet serait de permettre des communications entre la plupart des devices actuels (smartphones, lecteurs Blu-ray, appareils médicaux connectés, etc.) en échangeant plus rapidement avec un cout plus faible en bande passante.

9. Conclusion

POSIX a défini une IPC intéressante nommée POSIX MQ afin d'apporter sa propre implémentation « locale » similaire aux System V MQ.
L'intérêt de ces files de messages est d'assurer l'ordre et la priorité lors d'échange de messages entre plusieurs interlocuteurs.
Tout système d'exploitation certifié POSIX ou SUS les implémentera, ce qui assure une certaine compatibilité au sein des sources.
Il existe depuis plusieurs années des équivalents payants (et plus récemment certains gratuits) permettant d'échanger des messages entre machines totalement différentes d'un point de vue logiciel et matériel.
Les POSIX MQ sont donc parfaites pour découvrir le système de passage de messages, et faire des maquettes locales à un système.
Certains systèmes libres et gratuits ne les implémentent pas parfaitement, mais il est probable que cela s'améliorera avec le temps du fait de l'importance que les MQ « en général » commencent à acquérir dans le monde professionnel.

Retrouvez l'article de Fabrice Boissier en ligne : Lien 32

Sortie de PyQt 5.1, le support de nouveaux modules rend le binding plus complet.

Riverbank a annoncé récemment la sortie de la nouvelle version de son binding Qt pour Python. Cette dernière version, disponible pour Python 3.3 en version 32 bits et 64 bits, se base dorénavant sur la version 5.1.1 du framework de Digia et pour laquelle nous avons déjà largement communiqué lors de sa sortie : Lien 33.

Dans les nouveautés on notera la prise en charge des nouveaux modules Qt Sensors et Qt SerialPort et le support bien avancé des bindings OpenGL 2.0 et OpenGL ES/2.

Qt Sensors, apparu à l'origine dans Qt Mobility pour Qt 4, gère des capteurs tels qu'un accéléromètre, un gyroscope ou une boussole ; Qt SerialPort, quant à lui, est un dérivé de QSerialDevice, une bibliothèque pour la gestion des ports séries.

Concernant les bindings OpenGL, ceux-ci sont destinés à simplifier l'utilisation de l'API notamment dans la réalisation de scènes tridimensionnelles complexes.

Pour finir, la compilation croisée est dorénavant supportée et une configuration pour Raspberry Pi est maintenant possible.

Commentez la news de Charlie Gentil en ligne : Lien 34

Quel binding Qt pour Python utilisez-vous pour vos programmes en 2013 ? Votez et participez au débat en donnant les raisons de votre choix.

Bonjour,

PyQt et PySide sont deux bindings phares de Qt pour Python. Cela dit, il existe pour chacun d'eux des versions différentes.
Pour des raisons de version de Qt et/ou Python supportées ou pour des raisons de licences le choix du développeur va s'orienter sur un binding précis.

Et vous ?

Quel binding Qt utilisez-vous pour vos applications ?
Quelles sont les raisons qui ont motivé votre choix ?
Seriez-vous prêts à changer de binding ? Si oui, pour lequel ? Et pourquoi ?

Participez au débat, confrontez vos idées avec les autres membres et tentez peut-être de les faire changer d'avis.

Pour alimenter la discussion, n'oubliez pas de préciser les versions précises de Python et Qt supportées par le binding que vous avez choisi.

Bon débat à tous.

Commentez la news de Charlie Gentil et votez en ligne : Lien 35

Nouveau graphe de scène dans Qt 5.2

Qt 5.2 introduit un nouveau moteur de rendu pour le graphe de scène de Qt Quick.

Lors de la création du graphe de scène il y a trois ans, l'objectif était de tirer parti d'OpenGL et qu'il ait les performances d'un jeu vidéo. Le moteur de rendu livré avec Qt 5.0 est sur la bonne voie pour satisfaire les objectifs initiaux. Pour le contenu opaque, il trie le contenu selon l'état et affiche d'avant en arrière afin de minimiser le surcout de communication avec le GPU. Dans le dépôt playground/scenegraph, différents moteurs de rendu traitent les primitives similaires par lot afin de réduire le nombre d'appels aux fonctions de rendu (ce qui implicitement réduit le nombre de changements de contexte).

Le nouveau moteur de rendu de Qt 5.2 combine ces deux techniques et essaye de détecter les zones qui ne changent pas dans la scène et les garde dans la mémoire du GPU. Qt 5.2 ajoute aussi le support des textures atlas ce qui était précédemment situé dans playground/scenegraph/customcontext. Cela participe grandement au traitement par lot du contenu texturé. Avec ce nouveau moteur de rendu, Qt 5.2 satisfait aux objectifs initiaux plus que jamais.

Une nouvelle section de la documentation (Lien 36) explique les détails du moteur de rendu pour les plus curieux. Une connaissance approfondie des moteurs de rendu n'est pas nécessaire, mais le document sera néanmoins intéressant pour beaucoup d'entre vous.

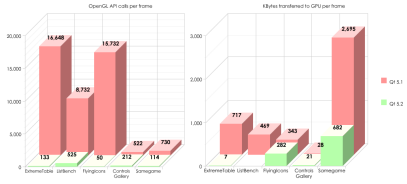
Quelques chiffres pour continuer :

Trois tests de performance sont disponibles dans les dépôts (Lien 37) :

- « Extreme Table » contient une grande table avec un peu de contenu animé au-dessus. Cet exemple met en avant les bénéfices de la rétention des données sur le GPU ;
- « List Bench » affiche un grand nombre de listes déroulantes avec icônes, en alternant la couleur de

fond et avec deux textes par cellule ;
- « Flying Icons » contient plus de 3 000 images animées, chacune possédant sa propre animation.

Sont aussi inclus la première page de la galerie d'exemples des contrôles Qt Quick ainsi que Samegame. Samegame est joué en mode « Zen » et les clics sont effectués sur le rythme de « Where Eagles Dare » d'Iron Maiden.

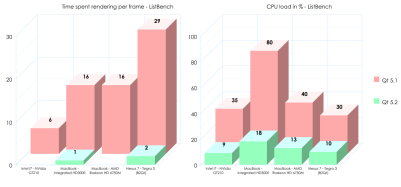


La mesure du nombre d'appels à OpenGL pour le rendu ainsi que du trafic a été réalisée grâce à apitrace, un outil incroyable pour déboguer avec OpenGL. Comme vous pouvez le voir au niveau des nombres en vert, le nombre d'appels aux fonction d'OpenGL a été drastiquement réduit, notamment dans les cas où il y avait de nombreuses occurrences du même contenu, c'est-à-dire des listes, des tables ou des grilles.

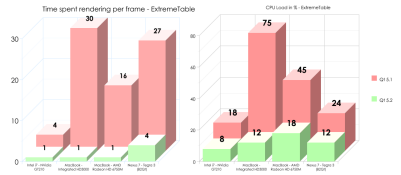
Le trafic par image a lui aussi été grandement amélioré. Les meilleurs exemples sont « ExtremeTable » et « ListBench », qui ont été échantillonnés avec une image sur laquelle aucun nouveau délégué n'était ajouté ou supprimé. Remarquez ici que l'exemple « ListBench » apparait avec un trafic de 0 par image. Il y a bien sûr un peu de trafic, les appels aux fonctions de rendu elles-mêmes et d'autres appels fonctionnels, mais aucune donnée contenant des sommets ou des textures, tout ce que l'outil est capable de mesurer. « FlyingIcons » change le contenu entier de la scène à chaque nouvelle image, il est ainsi impossible d'effectuer la moindre rétention sur le GPU. La galerie des contrôles est principalement statique, mais contient des barres de progression animées qui nécessitent une nouvelle texture à chaque image. Cela constitue la majorité du trafic d'environ vingt kilooctets. Samegame apparait comme gourmand, ce qui est principalement dû à son usage intensif des particules. La leçon du jour est que, si vous êtes limité au niveau de la mémoire, vous devriez limiter le nombre de particules de la scène.

Ce sont des nombres théoriques mais ils donnent une idée des capacités du nouveau moteur de rendu. Regardons maintenant les nombres provenant des cas réels. Les tableaux qui suivent montrent les cas où le nouveau moteur de rendu a le plus d'impact. Le tableau complet des mesures se trouve à la fin de l'article.

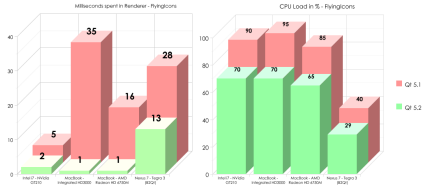
Notez que ces tests de performance sont exécutés sans utiliser le nouveau moteur QML basé sur V4. Ce nouveau moteur V4 affecte lui aussi les performances et il était plus pertinent de s'intéresser uniquement aux améliorations uniquement dues au nouveau moteur de rendu.



Le temps de rendu est mesuré avec la synchronisation verticale activée, mais sans compter le swap. Quand le MacBook passe 16ms lors de la phase de rendu, il est en fait ralenti lors des commandes OpenGL. En regardant les indicateurs d'OS X, on s'aperçoit que le pilote passe beaucoup de temps à attendre la synchronisation, ce qui signifie que le pipeline est obstrué. Avec le moteur de rendu de Qt 5.2, cette attente lors de la synchronisation n'existe plus. C'est une très bonne nouvelle étant donné que les contenus de type liste et assimilés sont très courants dans les interfaces graphiques.



Au niveau du MacBook avec le processeur graphique intégré et la Nexus, le nouveau moteur de rendu réduit drastiquement le temps passé à exécuter les appels à OpenGL. Un bogue connu (Lien 38) empêche de réduire le temps de rendu de l'exemple « Extreme Table » à zéro.



L'exemple « FlyingIcons » met le CPU à rude épreuve, à cause des 3 000 animations tournant en parallèle. Vous voyez néanmoins une nette amélioration quant à la durée passée dans le moteur de rendu.

Voici les autres résultats :

	Rendering time		CPU Load	
	Qt 5.1	Qt 5.2	Qt 5.1	Qt 5.2
ExtremeTable				
Intel i7 – Nvidia GT210	4 ms	1 ms	18 %	8 %
MacBookPro – Integrated	30 ms	1 ms	75 %	12 %
MacBookPro – AMD	16 ms	1 ms	45 %	18 %
Nexus 7 – Tegra3	27 ms	4 ms	24 %	12 %
FlyingIcons				
Intel i7 – Nvidia GT210	5 ms	2 ms	90 %	70 %
MacBookPro – Integrated	35 ms	1 ms	85 %	70 %
MacBookPro – AMD	16 ms	1 ms	85 %	65 %
Nexus 7 – Tegra3	28 ms	13 ms	40 %	29 %
ListBench				
Intel i7 – Nvidia GT210	6 ms	2 ms	35 %	9 %
MacBookPro – Integrated	16 ms	1 ms	80 %	18 %
MacBookPro – AMD	16 ms	1 ms	40 %	13 %
Nexus 7 – Tegra3	29 ms	13 ms	30 %	10 %
Controls Gallery				
Intel i7 – Nvidia GT210	0 ms	0 ms	7 %	7 %
MacBookPro – Integrated	4 ms	1 ms	20 %	20 %
MacBookPro – AMD	1 ms	0 ms	13 %	10 %
Nexus 7 – Tegra3	1.5 ms	0.5 ms	4 %	2 %
Samegame (Zen)				
Intel i7 – Nvidia GT210	1 ms	1 ms	25 %	20 %
MacBookPro – Integrated	18 ms	1 ms	25 %	17 %
MacBookPro – AMD	16 ms	4 ms	45 %	40 %
Nexus 7 – Tegra3	3-30 ms	16 ms	20 %	15 %

Les résultats obtenus concordent avec les estimations théoriques du départ. Lors de l'affichage de contenu récurrent, comme les listes ou les grilles, Qt 5.2 est une vraie amélioration vis-à-vis de Qt 5.1. Pour les autres cas typiques, les améliorations ne sont pas fulgurantes, mais au moins ne les font pas empirer.

Commentez la news d'Arnold Dumas en ligne : Lien 39

Compte-rendu du week-end de création de jeux vidéo du 05 au 07 juillet 2013

Du 05 au 07 juillet 2013 s'est tenu le troisième week-end de création de jeux vidéo sur le chat de Developpez.com (Lien 40). Comme pour la seconde édition (Lien 41), ce fut l'occasion pour les participants de créer ou d'améliorer un jeu dans un laps de temps très limité.

1. Introduction

Un week-end, c'est seulement 48 heures pour faire un jeu vidéo. Voici le défi qu'ont relevé une dizaine de membres du forum !
Le principe est très simple. Il suffisait de programmer. Le choix de la technologie était libre, tout comme le choix du jeu. Ainsi, nous pouvons énumérer différentes technologies et langages pour la réalisation des jeux, tels que : C++, Java, Flash, HTML5… L'événement avait pour lieu de rencontre le chat de Developpez.com (Lien 40), où les participants racontaient l'avancement de leur projet ou demandaient de l'aide. Il n'y avait donc aucune contrainte, le but principal était de s'amuser.
Durant le week-end, nous avons vu plus de trente personnes sur le salon spécialement créé pour l'événement (ce qui est incroyable pour un week-end).
Retrouvez cet événement sur le forum : Lien 42.

2. Participations

Vous pouvez télécharger le pack des participations (certains jeux ne sont consultables que sur Internet et donc non présents dans le pack) : Lien 43.

Liste des projets :

- Ubiquity de Balaz (Lien 44) ;
- UN fps d'eclesia (Lien 45) ;
- Le jeu du carré d'EpiTouille (Lien 46) :
  Vous dirigez le carré rouge à la souris, votre but est d'éviter les autres formes qui vous foncent dessus. Des carrés verts peuvent également apparaitre produisant différents effets ;
- TradeOnYT d'Etanne (Lien 47) :
  Pariez sur les vidéos les plus vues de YouTube ;
- KonKassor de Mikaël Guillemot (forthx) (Lien 48) :
  Jeu de casse-brique où chaque brique peut cacher une nouvelle balle ! (Seul le premier niveau est jouable en raison des bogues présents en fin d'événement.)
  Un éditeur de niveau est disponible, mais les sources doivent être modifiés pour intégrer de nouveaux niveaux (pas de système de chargement).
  Le code source est un très bon exemple de ce qui ne doit pas être fait ! L'instabilité du jeu en témoigne ;
- Jeu du carré rouge et tic-tac-toe de Guillaume Belz (Lien 49) :
  Le but était de montrer l'utilisation de Qt et en particulier de Qt Quick pour la création de jeux vidéo simples en 2D. Donc aucun jeu original, aucun gameplay original ou graphisme original, juste reproduire des jeux existants.
  Finalement, j'ai pu travailler le vendredi soir (faut éviter de faire ce genre de week-end en été quand il fait super beau dehors…) et je n'ai donc fait que deux jeux. Le premier (dont j'ai honteusement volé l'idée à EpiTouille) est le jeu du carré rouge. Il consiste à déplacer le carré rouge en évitant les carrés bleus et de sortir de la zone blanche.
  Le second jeu est un Tic-tac-toe, il faut simplement aligner trois croix ou trois ronds ;
- Get7 de germinolegrand (Lien 50) :
  Un jeu de cartes basé sur de courts 1vs1 dont le but est d'être le premier à sept points pour avoir ensuite le choix entre gagner immédiatement et tenter plus pour gagner plus ou pour tout perdre si l'adversaire atteint lui aussi sept points. Pour gagner des points, on rassemble une armée en faisant des paires avec les cartes du plateau et en réalisant des combos. Le projet est encore en travaux et en cours de développement ;
- RPG 3D Mystic Tower de Kannagi (Lien 51) :
  C'est un petit RPG 3D avec un système de combat original ;
- Petit jeu de casse brique en HTML5 d'imikado (Lien 52) :
  Alignez quatre briques de la même couleur horizontalement ;
- Atomix de LittleWhite (Lien 53) :
  Atomix est un jeu de réflexion ayant pour principe de faire glisser des atomes sur le plateau de jeu afin de rassembler des molécules. Les atomes ne s'arrêtent de glisser que s'ils touchent un mur ou un autre atome ;
- Last Dungeon de Neckara, Sentinelle, William Ground, Léo Jean et Mystogan98 (Lien 54) :
  Last Dungeon est un projet similaire à un jeu de go en ligne dont le but est de poser des « pierres » sur des « socles » afin d'entourer les « pierres » adverses. Mais Last Dungeon utilisant un système de script Lua, il est possible à partir de celui-ci de créer ses propres règles, voire son propre jeu.
  Last Dungeon se démarque du jeu de go de par son graphisme original, mais aussi grâce à ses propres règles, en effet quelques modifications ont été ajoutées :
  - des « plateaux » 3D (ex. cube) au lieu de plateau carré en 9x9, 13x13 ou 19x19 ;
  - la possibilité de faire des parties en équipe et de pouvoir jouer contre plusieurs

adversaires ;
    - …
- Formipachi de Shaga-Rha (Lien 55) ;
- YAMS en HTML5/CSS3 de Sylvain Pollet-Villard (SylvainPV) (Lien 56) :
  Le jeu du Yam's fait en HTML5/CSS3 ;
- ChestDefense de Timothée (yetimothee) et Théo (avec une assistance ponctuelle de Logan) (Lien 57) :
  Défendez votre coffre contre une horde de monstres voleurs !

3. Conclusion

J'ai demandé à chaque participant un petit retour sur l'événement :

- EpiTouille
  Je le referai avec plaisir l'année prochaine, je n'ai pas pu m'investir à fond ce week-end, car j'avais autre chose à faire. Ambiance sympa, des gens disponibles pour aider et donner leur avis sur nos projets/codes ;
- Mikaël GUILLEMOT (forthx)
  Comme le précédent, la bonne ambiance règne, j'ai eu le « malheur » d'avoir mes soirées déjà prises ce qui a grandement réduit le temps que j'y ai consacré. Cela m'a permis de tester un framework que je développe suite à mes expériences de création de jeux. Cela est très positif, car j'ai pu mettre le doigt sur un certain nombre de problèmes en peu de temps. Par contre je suis un peu déçu du résultat (jeu incomplet et truffé de bogues) qui ne correspond pas à ce que j'ai l'habitude de partager ;
- Guillaume Belz (gbdivers)
  Malheureusement, le salon Dev Jeux est devenu rapidement un remplaçant du salon Dev App. Le Mumble était plus sympa finalement (bien que pas trop de monde). Ça manquait de game designers, graphistes, etc., mais on n'y peut pas grand-chose.
  J'essaierai de refaire le prochain, en entier cette fois-ci. Et peut-être passer à la 3D (avec Qt bien sûr) ;
- germinolegrand
  Extrêmement sympathique ! C'est très motivant de développer un jeu tout en échangeant avec d'autres développeurs sur d'autres projets ! On ne regrette pas son week-end ;
- Kannagi
  J'ai bien aimé, j'ai hâte du prochain ;
- imikado
  Rien à redire, j'ai trouvé ça sympa, juste un peu perdu au moment de soumettre mon petit jeu, je ne savais pas où poster…
  Mais, oui, si je suis disponible pour le prochain je testerai de nouveau.
  Un truc qui pourrait être pas mal c'est par exemple d'imposer une bibliothèque Web, genre Quintus HTML5, Turbulenz ou autres.
  Ça permettrait de faire un événement autour d'une bibliothèque en montrant des démos faites en un week-end ;
- Neckara, Sentinelle, William Ground, Léo Jean, Mystogan98
  Ce week-end a été très intéressant et nous a permis de bien avancer sur notre projet.
  Mystogan98, William Ground et Neckara en ont profité pour se réunir samedi, cette journée n'a pas été très productive et a été ponctuée de quelques parties de LAN.
  Malgré tout, nous avons pu, dès dimanche, avoir une version « jouable ».
  Ce week-end a surtout été l'occasion pour nous de passer du bon temps en équipe ;
- Sylvain Pollet-Villard (SylvainPV)
  une bonne ambiance, mais quelques lacunes en matière d'organisation, par exemple pas d'endroit où s'inscrire pour recenser les projets, toutes les technos confondues dans le chat ce qui fait qu'il y avait plein de discussions croisées, etc. et le fait qu'il ait fallu attendre jusqu'à maintenant pour avoir la liste des projets réalisés ;
- Timothée (yetimothee) et Théo (avec une assistance ponctuelle de Logan)
  Très bon événement, comme à chaque fois (sur deux fois pour moi). Variétés de projets et de compétences, cela m'a permis de mettre à l'épreuve une bibliothèque destinée au développement de jeux vidéo, et de me confronter à la limite de temps, ce qui joue beaucoup (par exemple, je n'ai pas pu fignoler le gameplay du jeu en rajoutant des monstres…).
  Merci encore à(aux) l'organisateur(s) !

Au vu de la réussite de l'événement, il est très probable que nous fassions une quatrième édition, avec peut-être quelques modifications dans le déroulement de l'événement.

Retrouvez l'article d'Alexandre Laurent en ligne : Lien 58

Créer sa première bibliothèque Android pour Unity

Aujourd'hui, Unity permet gratuitement l'exportation de vos projets sur les plateformes Android et iOS. Unity comprend déjà un large éventail d'outils permettant de gérer et d'accéder directement à certaines fonctions de votre smartphone (caméra, gyroscope…). Cependant, il est probable que dans votre projet, vous avez besoin d'utiliser un composant qui n'est pas géré nativement par Unity (par exemple, utiliser le système de notification sur Android). Pour cela, vous aurez recours à la création de plugins.

Dans cet article, nous allons voir comment créer, ajouter et utiliser une bibliothèque JAR pour la plateforme Android à un projet Unity.

1. Prérequis

Avant de commencer sur ce tutoriel, je suppose que vous êtes déjà en possession de ces différents éléments sur votre ordinateur :

- Unity ;
- Eclipse + ADT.

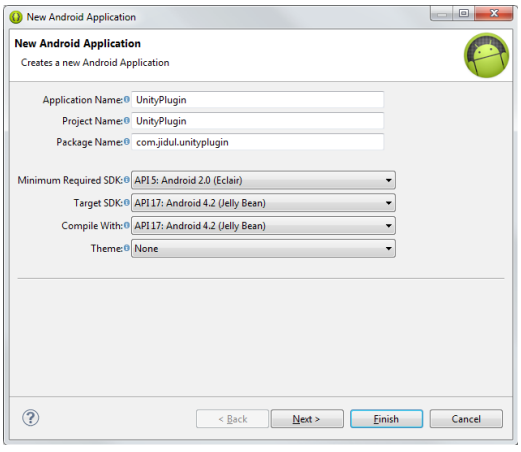
Pour rappel, afin qu'un projet Unity fonctionne sur votre terminal Android, ce dernier doit posséder la configuration minimale suivante :

- Android OS 2.0 ou plus ;
- ARMv7 (Cortex family) ;
- CPU GPU avec support OpenGLES 2.0 est recommandé.

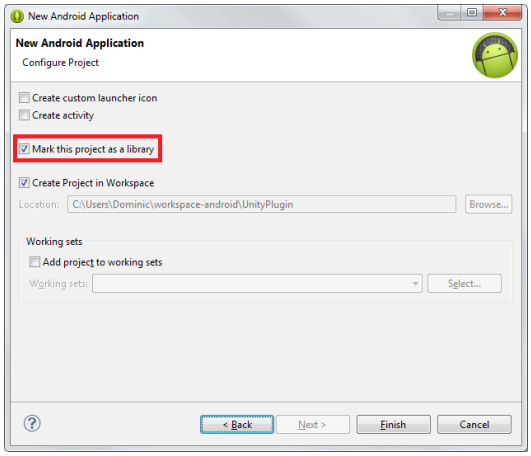
Plus d'informations : Lien 59.
Dans notre exemple, nous allons voir comment créer une bibliothèque Android et l'utiliser à l'intérieur d'un projet Unity. Nous allons simplement créer une classe Java contenant une fonction statique écrivant dans les logs du smartphone. Ensuite, nous verrons comment appeler cette méthode depuis un projet Unity.

2. Projet Android

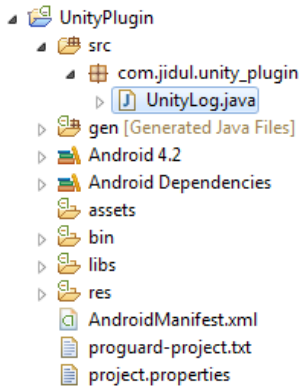
La première chose à faire est de tout d'abord créer un projet Android dans Eclipse.
Utilisez le même SDK que vous comptez utiliser lors de la génération de votre projet Unity par la suite.



Dans la suite, pas besoin de créer une icône, ni même de générer une Activity pour votre application : Unity le générera automatiquement pour vous. Cependant, très important, pensez à marquer le projet comme étant une bibliothèque.



Ensuite, commençons par créer un package que, pour l'exemple, je nomme « com.jidul.unity_plugin », et enfin créons à l'intérieur de ce dernier une classe appelée « UnityLog ».

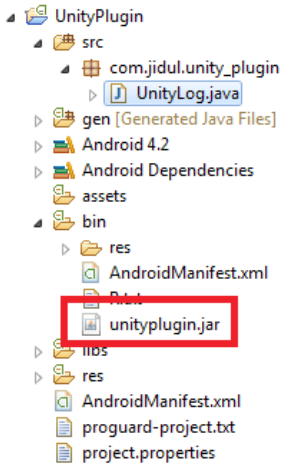


Dans cette classe, nous allons ajouter une simple méthode qui prend en paramètre une variable de type String correspondant au message que l'on souhaite afficher dans les logs.

Voici le code :

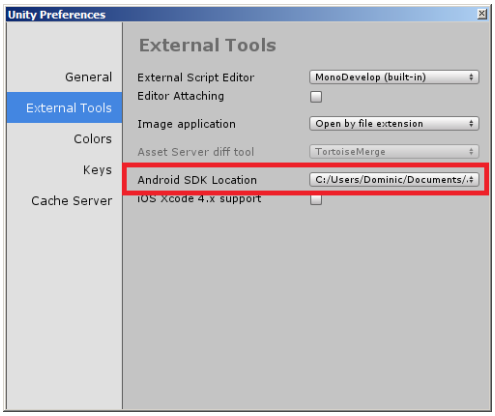
```
package com.jidul.unity_plugin;

import android.util.Log;

public class UnityLog {
    public static void trace(String message) {
        Log.d("UnityAndroidPlugin", message);
    }
}
```

Notre simple bibliothèque Android est d'ores et déjà prête. Pour la récupérer, il suffit de récupérer le fichier *.jar généré dans le dossier bin du projet.

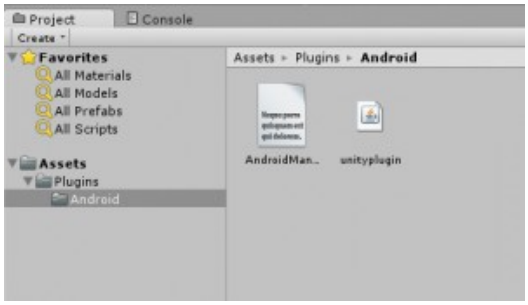


3. Projet Unity

Il est maintenant temps de passer à la partie Unity. Avant de continuer, un projet doit être auparavant créé.
Pensez bien d'abord à vérifier que vous avez bien indiqué à Unity la location du SDK Android. Pour cela, allez dans « Edit », « Preferences… » puis « External Tool ».



Ensuite, ajoutons la bibliothèque Android au projet. Créez tout d'abord le dossier« Plugins » dans les Assets, puis un dossier « Android » à l'intérieur de celui-ci. Enfin, copiez-y le fichier jar généré auparavant dans Eclipse.
Concernant le fichier AndroidManifest, Unity vous en crée un lors de la génération de votre application. Cependant, il peut arriver que vous ayez besoin d'utiliser votre AndroidManifest personnel (par exemple pour la définition de certains paramètres ou permissions). Pour cela, trouvez le fichier AndroidManifest par défaut de Unity dans ce répertoire (valable pour Windows) : C:\Program Files\Unity\Editor\Data\PlaybackEngines\androiddevelopmentplayer ; ensuite, copiez-le dans le même répertoire Android du projet Unity. À partir de là, vous pouvez éditer à votre guise le fichier Manifest du projet.
Une fois copié dans le dossier du plugin, vous pouvez éditer ce fichier Manifest qui sera automatiquement utilisé lors de la génération de votre projet Unity sous Android.



Maintenant, nous allons créer un simple bouton sur la scène qui fait que lorsqu'on cliquera dessus, cela appellera notre fonction de notre plugin Android.
Créez un dossier nommé « Scripts » dans les Assets et créez un nouveau « Script C# » nommé Button.cs. Ouvrez ensuite ce fichier script.

```
using UnityEngine;
using System.Collections;

public class GUITest : MonoBehaviour {

    void OnGUI () {
        GUI.Box(new Rect(10,10,100,90), "JiDuL
Android Plugin");

        if(GUI.Button(new Rect(20,40,80,20),
"Log")) {
            AndroidJavaClass jc = new
AndroidJavaClass("com.jidul.unity_plugin.UnityLog
");
            String message = "It looks like a
bird, but it's not a bird.";
            jc.CallStatic("trace", message);
        }
    }
}
```

Explication :

```
AndroidJavaClass jc = new
AndroidJavaClass("com.jidul.unity_plugin.UnityLog
");
```

On récupère une instance de notre classe en passant le nom du package puis le nom de la classe en paramètre.

```
jc.CallStatic("trace", "It looks like a bird, but
it's not a bird.");
```

On appelle ensuite la méthode static de cette classe en passant en paramètre le nom de la méthode, et son argument.
Cet exemple décrit un cas d'utilisation très simple, mais vous pouvez bien entendu récupérer la valeur de retour d'une méthode, instancier des objets et utiliser des méthodes non statiques, ainsi que de manipuler des objets

complexes Java depuis un script Unity. Vous trouverez plus d'informations et de détails sur cette page : Lien 60.
Pour finir, on crée un simple GameObject sur la scène, et on attache ce script dessus. Rien de plus simple. Il n'a plus qu'à générer l'APK et tester sur un votre appareil Android.
Voici le résultat attendu :


```
536     608     InputDisp...  Delivering touch to current input target: action: 0x1
536     608     InputDisp...  Delivering touch to current input target: action: 0x1
536     567                   inv_get_mag_rate
536     567     Sensors       accelHandler -0.116158 0.219144 9.750125
16976   16991   UnityAndr...  It looks like a bird, but it's not a bird.
536     567                   inv_get_mag_rate
```

4. Allez plus loin

4.1. Surcharger la classe Activity par défaut

Lors de la génération de votre projet Android sous Unity, ce dernier utilisera une Activity (UnityPlayerActivity) qui instanciera le player Unity dans votre application.
Vous pouvez trouver les sources de cette classe dans le répertoire :

- /Applications/Unity/Unity.app/Contents/PlaybackEngines/AndroidPlayer/src/com/unity3d/player sous MAC ;
- C:\Program Files\Unity\Editor\Data\PlaybackEngines\androidplayer\src\com\unity3d\player sous Window.

Pas besoin d'importer les sources, mais simplement la bibliothèque JAR « classes.jar » dans votre bibliothèque Android créée précédemment.
Vous pouvez trouver cette bibliothèque dans le répertoire :

- /Applications/Unity/Unity.app/Contents/PlaybackEngines/AndroidPlayer/bin sous MAC ;
- C:\Program Files\Unity\Editor\Data\PlaybackEngines\androidplayer\bin sous Window.

Ensuite, dans votre projet Android, créez une nouvelle classe et faites-la hériter de la classe UnityPlayerActivity. À partir de là, vous pouvez ensuite ajouter des composants ou de nouvelles fonctions à votre Activity. Lorsque tout est prêt, exportez votre Activity en bibliothèque JAR comme au début de ce tutoriel et ajoutez-la à votre projet Unity. Maintenant, il ne vous reste plus qu'à surcharger et éditer le fichier AndroidManifest.xml pour mettre votre Activity comme Activity par défaut de lancement.
Une autre solution est de créer une Activity vierge et d'y ajouter le composant UnityPlayer présent dans la bibliothèque « classes.jar ». Vous pouvez jeter un œil aux sources de la classe UnityPlayerActivity pour comprendre comment cette chose est réalisée.
Enfin, n'hésitez pas à visiter la documentation officielle pour plus d'informations : Lien 60.

Retrouvez l'article de Jonathan Odul en ligne : Lien 61

Liens

Lien 01 : http://www.w3.org/TR/CSS2/zindex.html
Lien 02 : http://nicolasgallagher.com/css-drop-shadows-without-images/demo/
Lien 03 : http://philipwalton.com/articles/what-no-one-told-you-about-z-index/
Lien 04 : http://philipwalton.com/
Lien 05 : http://philipwalton.developpez.com/tutoriels/css/connaitre-z-index/
Lien 06 : http://lesscss.org/
Lien 07 : http://twitter.github.io/bootstrap/index.html
Lien 08 : https://github.com/twitter/bootstrap/blob/master/less/variables.less
Lien 09 : http://lesscss.org/#usage
Lien 10 : http://samuelrossille.developpez.com/tutoriels/css/introduction-less/
Lien 11 : http://www.paulund.co.uk/create-animated-css-box-menu
Lien 12 : http://paulund.developpez.com/tutoriels/css/menu-boites-animees/
Lien 13 : https://github.com/synbioz/rubymotion_timezones
Lien 14 : http://synbioz.developpez.com/tutoriels/ruby/rubymotion-entrees-utilisateur/
Lien 15 : https://github.com/emmanuelremy/blog_amd
Lien 16 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/fichiers/amd-master.zip
Lien 17 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/fichiers/05_chargement_js_pas_amd.html
Lien 18 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/fichiers/03_chargement_css.html
Lien 19 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/fichiers/07_chargement_js_tpl.html
Lien 20 : http://ippon.developpez.com/tutoriels/javascript/mustachejs-icanhazjs/
Lien 21 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/fichiers/testTwitter.html
Lien 22 : http://dojofoundation.org/packages/dgrid/
Lien 23 : http://ippon.developpez.com/tutoriels/javascript/amd-loader-code-javascript-organise-performant/
Lien 24 : http://mac.developpez.com/actu/56613/Compte-rendu-de-la-keynote-WWDC-d-Apple-iOS-7-OS-X-10-9-Mavericks-Nouveau-Mac-Pro-et-MacBook-Air/
Lien 25 : http://www.apple.com/fr/ios/whats-new/
Lien 26 : http://www.apple.com/fr/iphone-5c/
Lien 27 : http://www.apple.com/fr/iphone-5s/
Lien 28 : http://www.developpez.net/forums/d1377201/systemes/mac/apple-presente-l-iphone-5s-iphone-5c-sortie-d-ios-7-a/
Lien 29 : http://www.ecmwf.int/services/computing/training/material/hpcf/Intro_MPI_Programming.pdf
Lien 30 : http://c.developpez.com/telecharger/detail/id/3537/Exemple-d-utilisation-POSIX-MQ
Lien 31 : http://mqtt.org/
Lien 32 : http://fabrice-boissier.developpez.com/articles/introduction-posix-mq/
Lien 33 : http://qt.developpez.com/actu/60546/Sortie-de-Qt-5-1-1-qui-confirme-la-maturite-des-Qt-Quick-Controls-et-la-stabilite-des-ports-pour-Android-et-iOS/
Lien 34 : http://www.developpez.net/forums/d1383871/autres-langages/python-zope/gui/pyside-pyqt/sortie-pyqt-5-1-binding-riverbank-supporte-nouveautes-qt-5-1-a/
Lien 35 : http://www.developpez.net/forums/d1374859/autres-langages/python-zope/gui/pyside-pyqt/binding-qt-python-utilisez-vos-programmes-2013-a/
Lien 36 : http://doc-snapshot.qt-project.org/qt5-dev/qtquick/qtquick-visualcanvas-scenegraph-renderer.html
Lien 37 : https://github.com/qtproject/playground-scenegraph/tree/master/benchmarks
Lien 38 : https://bugreports.qt-project.org/browse/QTBUG-32997
Lien 39 : http://www.developpez.net/forums/d1375362/c-cpp/bibliotheques/qt/qt-quick/nouveau-graphe-scene-qt-5-2-a/
Lien 40 : http://chat.developpez.com/
Lien 41 : http://jeux.developpez.com/evenements/we-jeux/2/
Lien 42 : http://www.developpez.net/forums/d1347041/applications/developpement-2d-3d-jeux/evenement-05-07-juillet-developper-jeu-en-week-end-chat-developpezcom/
Lien 43 : http://jeux.developpez.com/evenements/we-jeux/3/fichiers/pack-we3.zip
Lien 44 : http://www.developpez.net/forums/d1360573/applications/developpement-2d-3d-jeux/projets/we-jeux-3-ubiquity/
Lien 45 : http://www.developpez.net/forums/d1360679/applications/developpement-2d-3d-jeux/projets/we-jeux-3-fps/
Lien 46 : http://www.developpez.net/forums/d1360578/applications/developpement-2d-3d-jeux/projets/we-jeux-3-jeux-du-carre/
Lien 47 : http://www.developpez.net/forums/d1360569/applications/developpement-2d-3d-jeux/projets/we-jeux-3csharpapplication-tradeonyt-pariez-videos-plus-vues-youtube/
Lien 48 : http://www.developpez.net/forums/d1360512/applications/developpement-2d-3d-jeux/projets/we-jeux-3-konkassor/
Lien 49 : http://www.developpez.net/forums/d1360264/applications/developpement-2d-3d-jeux/projets/we-jeux-3cppqt-participation-au-week-end-jeux-du-5-juillet-2013/
Lien 50 : http://www.developpez.net/forums/d1360335/applications/developpement-2d-3d-jeux/projets/we-jeux-3-get7/
Lien 51 : http://www.developpez.net/forums/d1360384/applications/developpement-2d-3d-jeux/projets/we-jeux-3rpg-3d-mystic-tower/
Lien 52 : http://www.developpez.net/forums/d1360676/applications/developpement-2d-3d-jeux/projets/we-jeux-3-petit-jeu-casse-brique-en-html5/
Lien 53 : http://www.developpez.net/forums/d1360495/applications/developpement-2d-3d-jeux/projets/we-jeux-3-atomix/
Lien 54 : http://www.developpez.net/forums/d1360316/applications/developpement-2d-3d-jeux/projets/we-jeux-3last-dungeon/
Lien 55 : http://www.developpez.net/forums/d1360429/applications/developpement-2d-3d-jeux/projets/we-jeux-3cpp-formipachi/
Lien 56 : http://www.developpez.net/forums/d1360795/applications/developpement-2d-3d-jeux/projets/we-jeux-3yams-en-html5css3/
Lien 57 : http://www.developpez.net/forums/d1360498/applications/developpement-2d-3d-jeux/projets/we-jeux-3-chestdefense/
Lien 58 : http://jeux.developpez.com/evenements/we-jeux/3/
Lien 59 : http://unity3d.com/unity/system-requirements
Lien 60 : http://docs.unity3d.com/Documentation/Manual/PluginsForAndroid.html
Lien 61 : http://jodul.developpez.com/tutoriels/android/creer-sa-premiere-bibliotheque-android-pour-unity/