The Enlightenment Foundation Libraries

Prise en main de Edje

Nicolas Aguirre
aguirre.nicolas@gmail.com

May 3, 2011

Contents

1 Avant Propos

Le but de ce tutorial est de survoler au travers d'un exemple pratique toutes les fonctionnalités de Edje.

J'espére qu'il vous permettra également de vous faire comprendre comment Edje peut vous aider dans le développement de vos interfaces graphiques.

De considerer cette technologie comme l'un des outils les plus puissant des EFL plutôt que comme votre plus grand cauchemar.

Comment, en séparant la logique et le code d'une part et l'interface d'une autre, vos interfaces graphiques peuvent gagner en flexibilité.

Comme exemple concret, permettant d'illustrer cette présentation, j'ai choisis le développement d'un interface (tactile) tres simple. Voici a quoi ressemblera l'interface a la fin de ce tutoriel :

2 Introduction

Edje est une des briques de base des EFL. Elle vous permet de décrire une interface graphique sans écrire une seule ligne de C. Ce qui permet,de facto, de réaliser une des choses les plus complexes lors du développement d'un programme avec une interface utilisateur : la séparation de l'interface et du code. Cette séparation est importante à plus d'un titre. Elle permet d'une part d'avoir la logique du programme et la gestion des données d'un côté et l'interface utilsateur de l'autre. Elle permet donc d'avoir deux équipes distincte qui travaille sur le projet, les graphistes, designer, ergonomes et les développeurs.

Parler de "Edje", c'est employer un therme générique pour 3 concepts différents:

- Le format de description, le format EDC, pour Edje Data Collection;
- Le fichier binaire EDJ, résultante compilée de toutes les ressources décrites dans le fichier EDC;
- La bibliotheque de fonctions libedje.so, permettant de manipuler les objets décrits dans le EDC au niveau de evas.

Le schémas ci dessus montre a quel moment ces trois concepts sont utilisés lors de la création d'une application utilisant Edje :

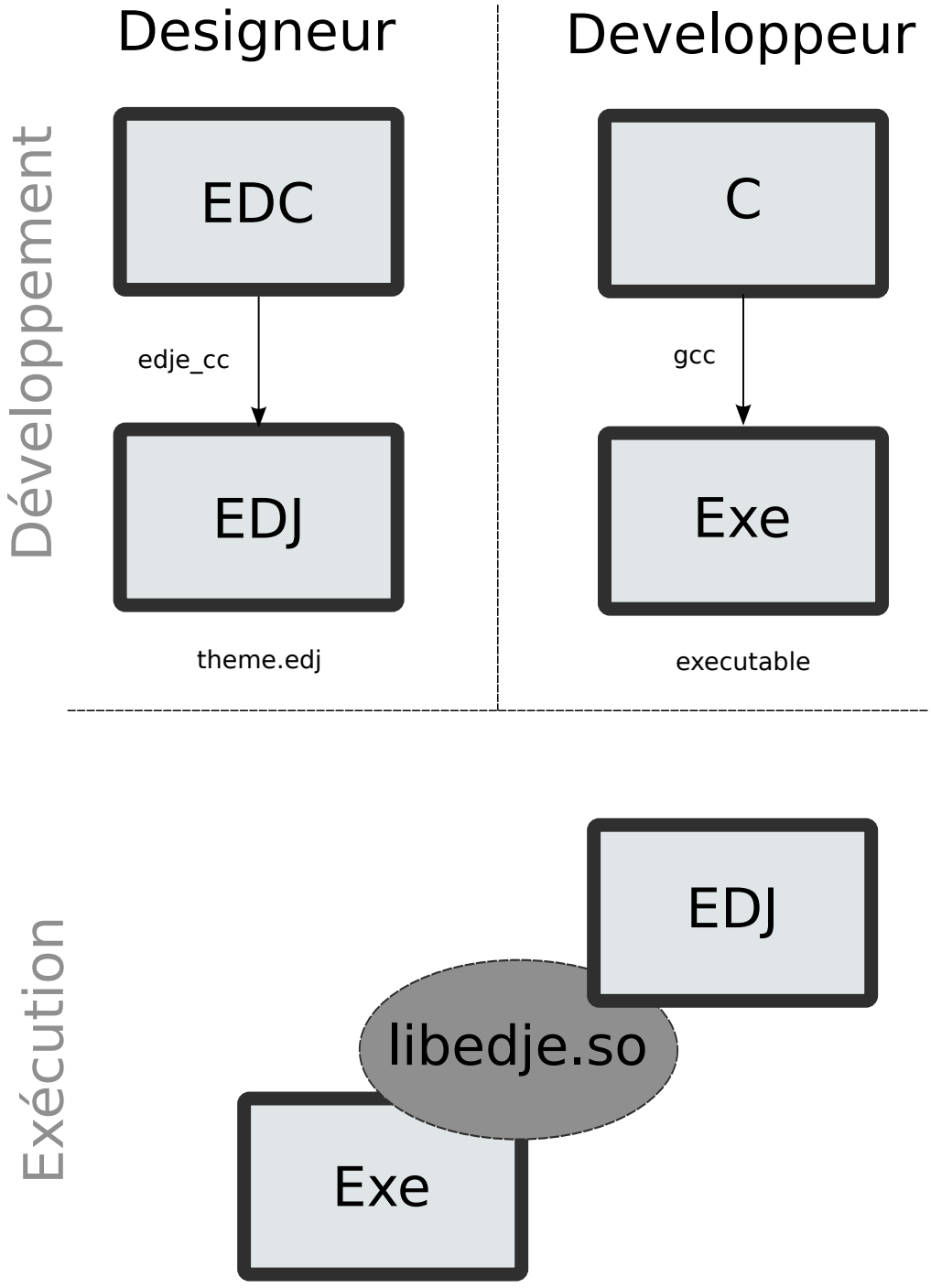

Figure 1: Edje Workflow

3 Les Bases

Dans ce chapitres, nous allons voir les bases du language de description EDC.

Une fichier EDC (Edje Data Collection) minimal ressemble a ceci : (fichier tut01/tut01.edc)

```
collections {
  group {
     name: "interface";
     parts {
        /* Rectangle Rouge */
        part {
           name: "Rectangle";
           type: RECT;
           description {
              state: "default" 0.0;
              color: 255 0 0 255;
           }
        }
     }
  }
}
```

Nous pouvons voir dans cette exemple le mot clef “collection” qui comme son nom l’indique est un ensemble de “groupes”. Dans cette exemple nous avons un seul groupe, nommée “interface”. Un groupe est lui meme un ensemble, et représente un objet, qui pourra être manipulé sur le canvas graphique plus tard dans notre programme ou réutilisé dans le fichier edc. Un Group contient des “parts” qui sont les primitives que sais manipuler Evas. Voici une liste exhaustive des “parts” que nous pouvons utiliser :

- Les rectangles : RECT;
- Les images : IMAGE;
- Les textes : TEXT;
- Les blocs de texte : TEXTBLOCK;
- Les containers : SWALLOW;
- Les groupes : GROUP;
- Les boites : BOX;
- Les tables : TABLE;
- Les objects externes : EXTERNAL;

Chaque type fera l’objet d’une étude plus approfondie dans la suite de ce tutoriel.

Figure 2: Interface edje à la fin de ce tutoriel

Dans notre exemple nous décrivons donc un Rectangle rouge, rien de bien original. Nous allons maintenant compiler ce fichier edc en un fichier binaire EDJ. :

```
edje_cc tut01.edc
```

Si tout c'est bien passé, nous devrions trouver un fichier tut01.edj dans notre répertoire. Comme nous l'avons vu un peu plus haut. Ce fichier edj doit être chargé par notre programme pour pouvoir être affiché. Dans un premier temps nous allons donc utiliser un outil tres pratique proposé par edje : edje_player.

```
edje_player tut01.edc
```

Et voici le résultat : Un Rectangle Rouge affiché a l'écran ! Emotionnelement intense. Que ceux qui n'ont pas la cher de poule a ce moment précis arrêtent tout de suite la lecture. Quand aux autres, vous pouvez trouver ci-dessous une capture d'écran de l'interface que nous allons développer dans la suite de ce tutoriel. j'ai choisis le développement d'un interface (tactile) simple, qui nous permettra d'apprehender les différents concept de Edje par la pratique.

Voici le résultat final:

4 Les Images

4.1 Les parts de type IMAGE

Les explication de cette section portent sur le fichier tut02/tut02.edc.

En partant du premier exemple, nous allons ajouter un fichier qui sera le fond de notre interface.

Edje supporte un large type d'images, celles supportées par Evas, PNG, JPEG, TIFF, BMP, ... Pour décrire une image, il faut créer un part de type "IMAGE". Et dire a edje quelle image insérer :

```
part {
  name: "Fond";
  type: IMAGE;
  description {
    state: "default" 0.0;
    image.normal: "bg.jpg";
  }
}
```

4.2 la directive images

Comme nous avons vu précédement, le fichier EDJ généré contient toutes les ressources de notre interface, images incluses. Si nous compilons avec edje_cc ce fichier, "bg.jpg" ne sera pas trouvé dans nos ressources. Il faut ajouter cette images dans la collection. Ceci est réalisé par l'ajout de cette directive :

```
  images {
     image: "bg.jpg" COMP;
  }
```

A quoi correspond "COMP". Edje ajoute les images dans le fichier binaire EDJ, et nous pouvons lui dire de compresser ou non cette image. Plusieurs type de compressions sont supportés :

- RAW: Uncompressed.
- COMP: compression sans perte, comparable au format PNG.
- LOSSY [0-100]: comression avec perte avec une qualité pouvant allé de 0 à 100, comparable au format JPEG
- USER: l'image n'est pas intégrée au fichier edje, mais est lue depuis le disque.

Attention cependant, si vous utilisez la balise RAW, a la taille finale de votre fichier binaire. Pour une image de 800x600 en 32bits de couleurs (RGBA), la taille embarqué dans le fichier binaire sera : 800x600x4 = 1.8MO !

4.3 edje_cc et les images

Le compilateur edje cherche les images dans les repertoires relativement au repertoire ou il est executé. Nous avons la possibilité de donner l'emplacement relative dans nos fichiers edc. Par exemple :

```
   images {
      image: "images\bg.jpg" COMP;
   }
```

Ceci peut tres vite devenir rebarbatif et long a écrire, nous pouvons donc ajouter les répertoires qui contiennent nos images en utilisant l'argument "-id" (image directory) de edje_cc

```
edje_cc -id images -id images\icons file.edc
```

Pour faciliter la compilation, vous trouverez dans le répertoire de ce tutorial, un script shell qui permet de compiler tous les fichiers edc (et C dans la suite) nommé build.sh

```
.\build.sh #compile l'intégralite des exemples.
.\build.sh #tut02 compile l'exemple contenu dans le repertoire tut02
```

Les fichiers binaires sont quand a eux générés dans le répertoire build.

4.4 Les motifs

Edje permet également l'affichage de motifs a l'écran en répétant une images. Le fichier tut03/tut03.edc montre comment utiliser une image motif avec la balise fill

```
 size {
   relative: 0.0 0.0;
   offset: 20 20;
 }
```

Dans notre cas l'image a une taille de 20x20px nous voulons qu'elle soit répétée sur l'axe des X et des Y.

Il y a plusieurs autres options permettant de repeter les motifs, je vous laisse les découvrir par vous même, tout est décris dans la [documentation de edje]

5 Le Texte

5.1 description d'une icone

Regardons le code du fichier tut04/tut04.edc plus en détails. Un nouveau groupe a été ajouté nommée "icon". Il contient une image "icon.png" et un nouveau part de type "TEXT".

```
part {
  name: "text";
  type: TEXT;
  description {
    state: "default" 0.0;
    color: 0 0 0 255;
    text {
      font: "Sans";
      size: 12;
      text: "Description";
    }
  }
}
```

Les différentes options parlent d'elle méme :

- Couleur du texte : Noir;
- Fonte utilisée "Sans";
- Taille de la fonte : 12;
- Texte a afficher "Description";

Regardons a quoi ressemble notre exemple avec edje_player. Attention dans cet exemple nous avons deux groupes. Nous devons donc spécifier a ejde_player quel groupe nous voulons visualiser, et nous allons également lui dire de changer la couleur de fond.

```
edje_player -c=255,255,255,255 -g icon tut04.edj
```

Nous sommes loins du résultat escompté ! Par défaut tous les parts sont centrés au centre de l'écran et occupe tout le taille du groupe. Nous devons décrire comment les objets sont placés les uns par rapport aux autres. C'est l'objet des tutoriaux 5 à 8.

6 Placement des objets

6.1 Proportions

L'icône se doit d'être carré, c'est le cas de toutes les icônes en informatique non ? Pour cela Edje propose la balise aspect et aspect preference. Ces deux balises sont liés. Regardons a quoi ca ressemble pour un définir un part carré:

```
  aspect: 1.0 1.0;
  aspect_preference: BOTH;
```

aspect prends deux flottants comme paramétres, min et max. Dans un cas normal, les dimensions de l'objet ne sont pas liés, en utilisant le paramétre aspect, on force edje a garder un ration entre la largeur et la hauteur de notre part. Dans le cas 1.0 1.0 l'icône aura donc meme hauteur et largeur. aspect_preference, lui donne la direction on veux que ce ration s'applique. Rien de mieux pour comprendre qu'un exemple. regardez les fichier tut05.1.edj, tut05.2.edj et tut05.3.edj pour visualiser les effets de aspect et aspect_preference

6.2 Positions relatives et offset

Edje permet de positionner les objets les uns par rapport aux autres de deux façons. Positionnement relatifs et positionnement absolu.

Le positionnement relatif est en pourcent (valeur ramenée a un floatant entre 0.0 et 1.0) alors que les positions absolu sont en pixels.

Les balises rel1 et rel2 permettent de donner la position d'un objet par rapport à un autre. Par défaut la position est celle par rapport au groupe. Le positionnement relatif est donnée avec la balise rel1.relative et rel2.relative alors que le positionnement absolu est donnée par rel1.offset et rel2.offset.

Dans cet exemple le positionnement est relatif au groupes, mais nous pouvons spécifier un positonnement par rapport à un autre part en utilisant la balise :

```
rel1.to: ``autre_part1'';
rel1.to: ``autre_part2'';
```

Ou encore relativement à un autre part mais uniquement sur un l'axe X ou Y :

```
rel1.to_x: ``autre_part1'';
rel1.to_y: ``autre_part2'';
```

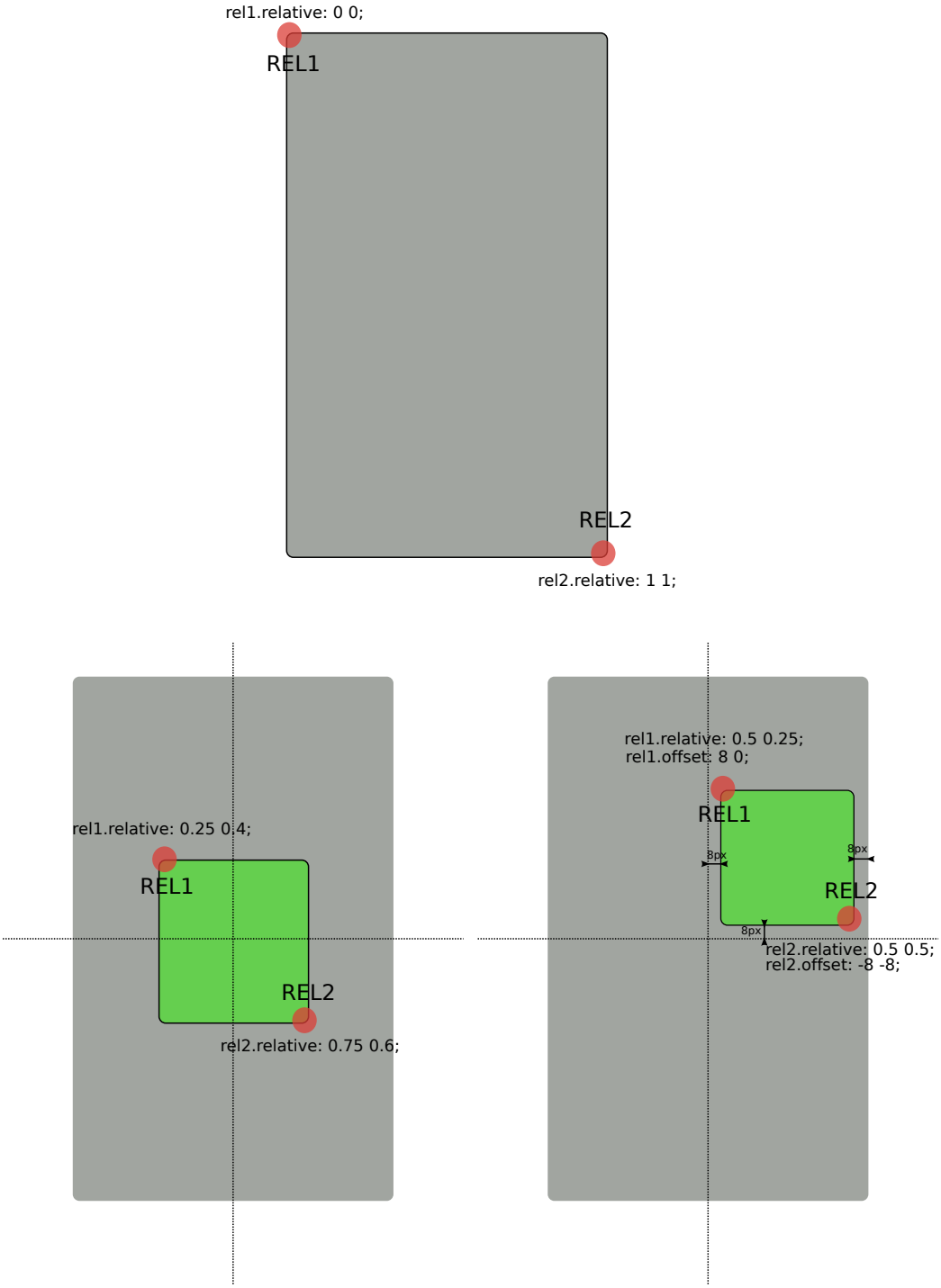

Figure 3: Positionnement relatif et absolu