



The Enlightenment Foundation Libraries

Prise en main de Edje

Nicolas Aguirre
aguirre.nicolas@gmail.com



This work is licensed under Creative Commons Attribution-Share Alike 3.0 License
(<http://creativecommons.org/licenses/by-sa/3.0/>)

April 30, 2011



Contents

1	Avant Propos	2
2	Introduction	3
3	Les Bases	5
4	Les Images	7
4.1	Les parts de type IMAGE	7
4.2	la directive images	7
4.3	edje_cc et les images	8
4.4	Les motifs	8



1 Avant Propos

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

J'espère 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 utilisateur de l'autre. Elle permet donc d'avoir deux équipes distinctes qui travaillent sur le projet, les graphistes, designers, ergonomes et les développeurs.

Parler de “Edje”, c'est employer un terme générique pour 3 concepts différents:

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

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

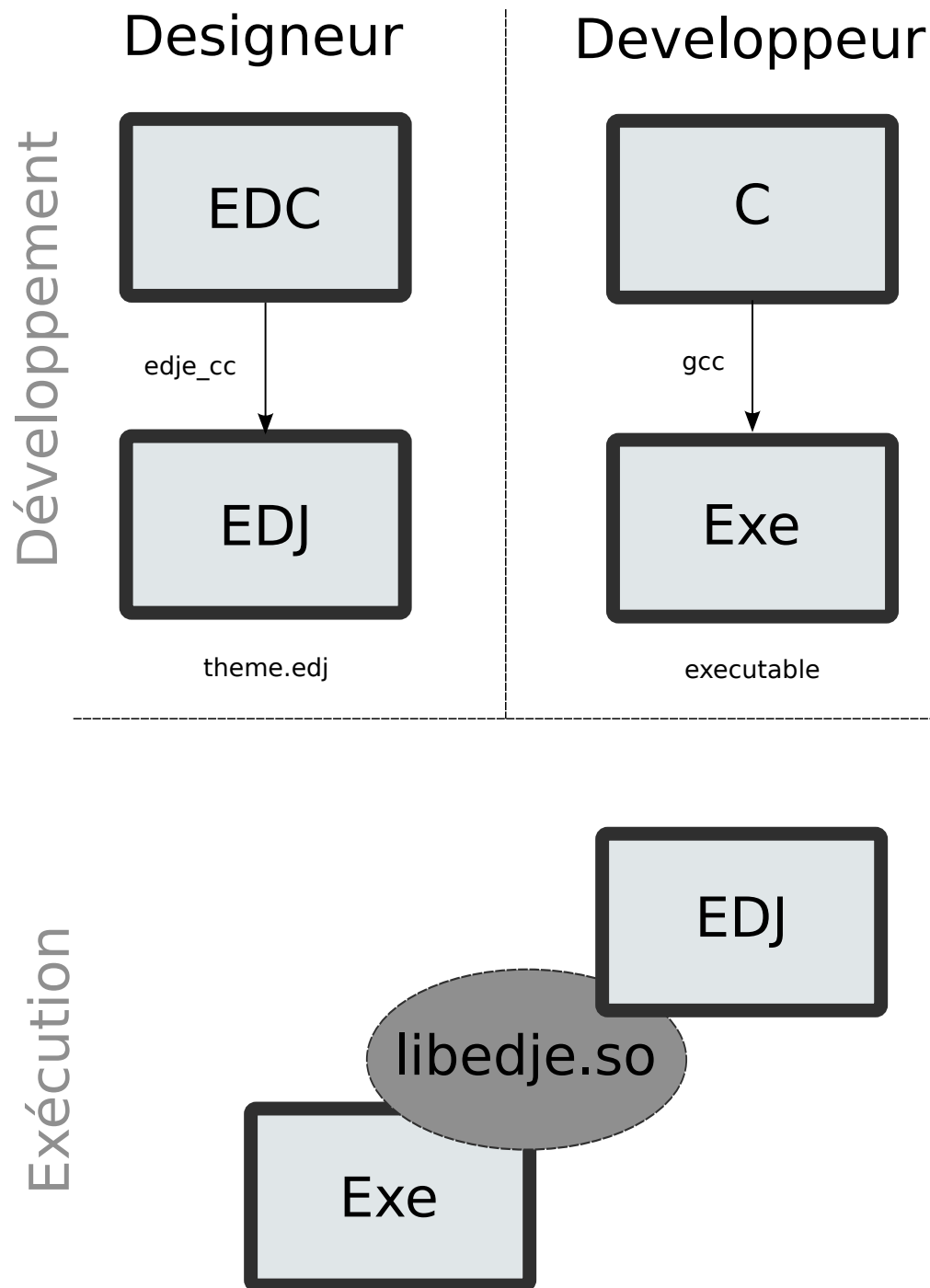


Figure 1: Edje Workflow



3 Les Bases

Dans ce chapitre, nous allons voir les bases du langage 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 cet exemple le mot clef “collection” qui comme son nom l’indique est un ensemble de “groupes”. Dans cet exemple nous avons un seul groupe, nommée “interface”. Un groupe est lui même 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 boîtes : BOX;
- Les tables : TABLE;
- Les objets 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 très pratique proposé par edje : `edje_player`.

```
edje_player tut01.edc
```

Et voici le résultat : Un Rectangle Rouge affiché à l'écran ! Emotionnement intense. Que ceux qui n'ont pas la cher de poule à 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'une interface (tactile) simple, qui nous permettra d'appréhender les différents concepts 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 précédemment, 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]: compression avec perte avec une qualité pouvant aller 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 : $800 \times 600 \times 4 = 1.8\text{MO}$!



4.3 edje_cc et les images

Le compilateur edje cherche les images dans les répertoires relativement au répertoire où il est exécuté. Nous avons la possibilité de donner l'emplacement relatif dans nos fichiers edc. Par exemple :

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

Ceci peut très vite devenir rebarbatif et long à 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'intégralité des exemples.  
.\build.sh #tut02 compile l'exemple contenu dans le repertoire tut02
```

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

4.4 Les motifs

Edje permet également l'affichage de motifs à l'écran en répétant une image. 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 répéter les motifs, je vous laisse les découvrir par vous-même, tout est décrit dans la [documentation de edje]