Un carrousel en EFL

Edje, Evas et Elementary sont dans un manège

Nicolas Aguirre

26 Janvier 2013

Notions que nous aborderons

Edje

Elementary

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje

Elementary

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary

Elementary

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary
- Création de groupes edje et utilisation en tant qu'objets Evas

Elementary

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary
- Création de groupes edje et utilisation en tant qu'objets Evas

Elementary

- Création d'une fenêtre

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary
- Création de groupes edje et utilisation en tant qu'objets Evas

Elementary

- Création d'une fenêtre
- Intégration d'un layout edje dans la fenêtre

Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary
- Création de groupes edje et utilisation en tant qu'objets Evas

Elementary

- Création d'une fenêtre
- Intégration d'un layout edje dans la fenêtre

Evas

- Création d'un Objet Evas

Notions que nous aborderons

Edje

- Création d'un fichier edje
- Utilisation d'un fichier edje dans elementary
- Création de groupes edje et utilisation en tant qu'objets Evas

Elementary

- Création d'une fenêtre
- Intégration d'un layout edje dans la fenêtre

Evas

- Création d'un Objet Evas
- Manipulation d'objets Evas

Show me the code !

Tout le Code de ce tutoriel est sous licence GPLv3 Il est disponible à cette adresse :

Utilisez git ou allez en enfer !

git clone https://github.com/naguirre/carrousel.git

Hello World

Cette étape permet la mise en place des autotools et du fichier main.c
Le fichier configure.ac contient en check sur elementary

```
PKG_CHECK_MODULES([CARROUSEL], [elementary >= 1.0.0])
```

Le fichier Makefile.am contient les fichier a compiler, uniquement main.c pour le moment.
Et voici le contenu du fichier main.c :

```
#include <Elementary.h>

int main(int argc, char **argv)
{
    printf("Hello E World\n");
    return 0;
}
```

Récupération du code et Compilation

Récupération du code :

git checkout step1

Compilation tres classique avec

```
./autogen.sh
./configure
make
make install
```

Step2 : ELementarization

```
1  #include <Elementary.h>

3  #ifndef ELM_LIB_QUICKLAUNCH

5  EAPI_MAIN int
   elm_main(int argc EINA_UNUSED, char **argv EINA_UNUSED
       )
7  {
       printf("Hello Elementary World\n");
9      return 0;
   }
11
   #endif
13
   ELM_MAIN()
```

Step3

Création de la fenêtre :

```
win = elm_win_add(NULL, "main", ELM_WIN_BASIC);
elm_win_title_set(win, "EFL Demo");
....
evas_object_resize(win, 800, 600);
evas_object_show(win);
```

Création d'un fond :

```
bg = elm_bg_add(win);
elm_win_resize_object_add(win, bg);
evas_object_show(bg);
```

Boucle de messages :

```
    elm_run();
```

Step4

Fermeture de la fenêtre :

```
1  evas_object_smart_callback_add(win, "delete,request",
       _win_del, NULL);
```

Callback de fermeture :

```
1  static void
   _win_del(void *data EINA_UNUSED, Evas_Object *obj
       EINA_UNUSED, void *event_info EINA_UNUSED)
3  {
       elm_exit();
5  }
```

Step4

Création d'un fichier edje contenant un group layout

```
group {
    name: "layout";
    parts {
        part {
            name: "bg";
            type: IMAGE;
            description { image.normal: "bg.png"; }
        }
        part {
            name: "caroussel.swallow";
            type: SWALLOW;
            mouse_events: 1;
            description { state: "default" 0.0; }
        }
    }
}
```

Intégration du layout

On créé un nouveau layout elementary. On charge le fichier edje précédement créé et on charge le groupe “layout”. On en profite également pour faire en sorte que la dimension de la fenêtre soit liée a celle de l'object layout. Et finalement on affiche le layout à l'écran.
Un layout elementary est un frontend a edje, permettant de charger des fichiers et des groupes Edje.
Au lieu de manipuler un object edje, on manipule un object elementary.
Dans les deux cas se sont des Evas_Objects.
Une préférence va a l'utilisation des objets elementary, car il héritent des propriétés globales de elm (theme, finger size ...)

code

```
   ly = elm_layout_add(win);
2  elm_layout_file_set(ly,
         PACKAGE_DATA_DIR"/themes/default/default.edj",
4        "layout");
   elm_win_resize_object_add(win, ly);
6  evas_object_show(ly);
```

Step5

Création d'un object carrousel basé sur un elm_grid.
(carrousel.[ch])

```
Evas_Object *
carrousel_add(Evas_Object *parent)
{
     Evas_Object *grid;

     grid =  elm_grid_add(parent);
     evas_object_grid_size_set(grid, 800, 600);

     return grid;
}
```

Intégration de l'objet dans l'interface

```
elm_object_part_content_set(ly, "caroussel.swallow",
    carrousel);
```

Elm_Grid

On spécifie une taille de grille de 800x600px.
Une elm grid permet de placer librement des Evas Objects a l'intérieur. Les objets insérés sont alors des enfants de la grille. Lorsque la grille est supprimée, les enfants sont supprimés a leur tour.
On pourrait utiliser evas_object_move et evas_object_resize en lieu et place de elm_grid.
Elm_Grid permet cependant de gérer l'arbre d'objets aisi que l'héritage des paramétres génériques ELM. Pour placer des objets dans la grille on utilise :

```
1 elm_grid_pack(grid, item->obj, 0, 0, 0, 0);
```

Step6-Step7

Les éléments du carrousel sont des elm_layouts basés sur un groupe edje : “caroussel/layout/item”. On créé dans un premier temps 8 Objets que l’on place a l’écran. L’affichage a l’écran se fait dans la fonction _anim(). Tous les objets créés sont ajoutés dans un Eina List.

la fonction _anim

```
1  static Eina_Bool
   _anim(void)
3  {
       Carrousel_Item *item;
5      Eina_List *l;
       Evas_Coord x, y, w, h;
7
       EINA_LIST_FOREACH(items, l, item)
9      {
           y = (HEIGHT / 2) - (ICON_SIZE_H / 2);
11         x = (WIDTH  / 2) - (ICON_SIZE_W / 2);
           w = ICON_SIZE_W;
13         h = ICON_SIZE_H;
           elm_grid_pack_set(item->obj, x, y, w, h);
15     }
       return ECORE_CALLBACK_RENEW;
17 }
```

avec du padding

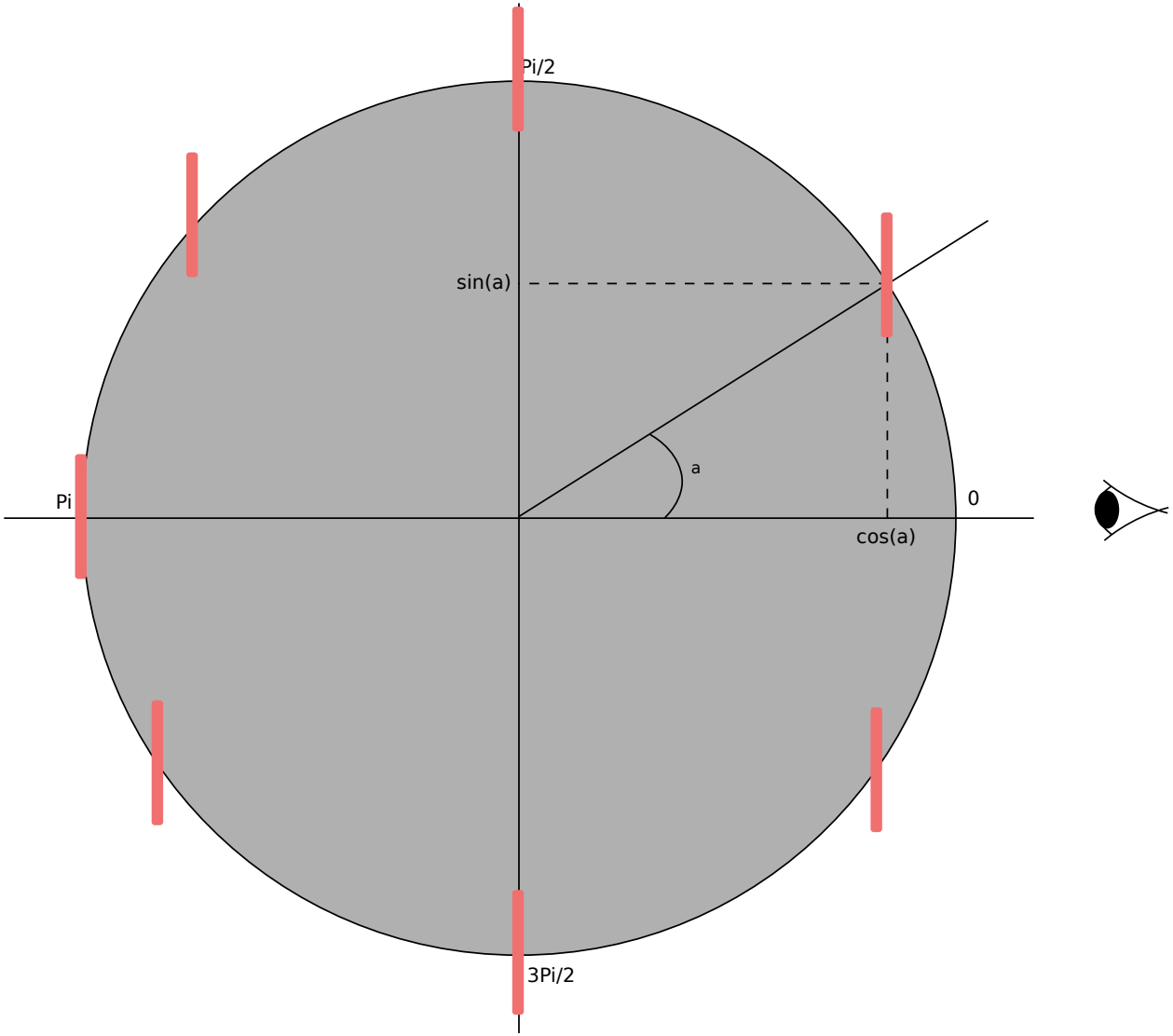
```
1  static Eina_Bool
   _anim(void)
3  {
       Carrousel_Item *item;
5      Eina_List *l;
       Evas_Coord x, y, w, h;
7      int i = 0;
       EINA_LIST_FOREACH(items, l, item)
9      {
           y = HEIGHT / 2 - ICON_SIZE_H / 2;
11         x = PADDING + i * (ICON_SIZE_W + PADDING);
           w = ICON_SIZE_W;
13         h = ICON_SIZE_H;
           elm_grid_pack_set(item->obj, x, y, w, h);
15         i++;
       }
17     return ECORE_CALLBACK_RENEW;
   }
```

Chargement des images sur le disque

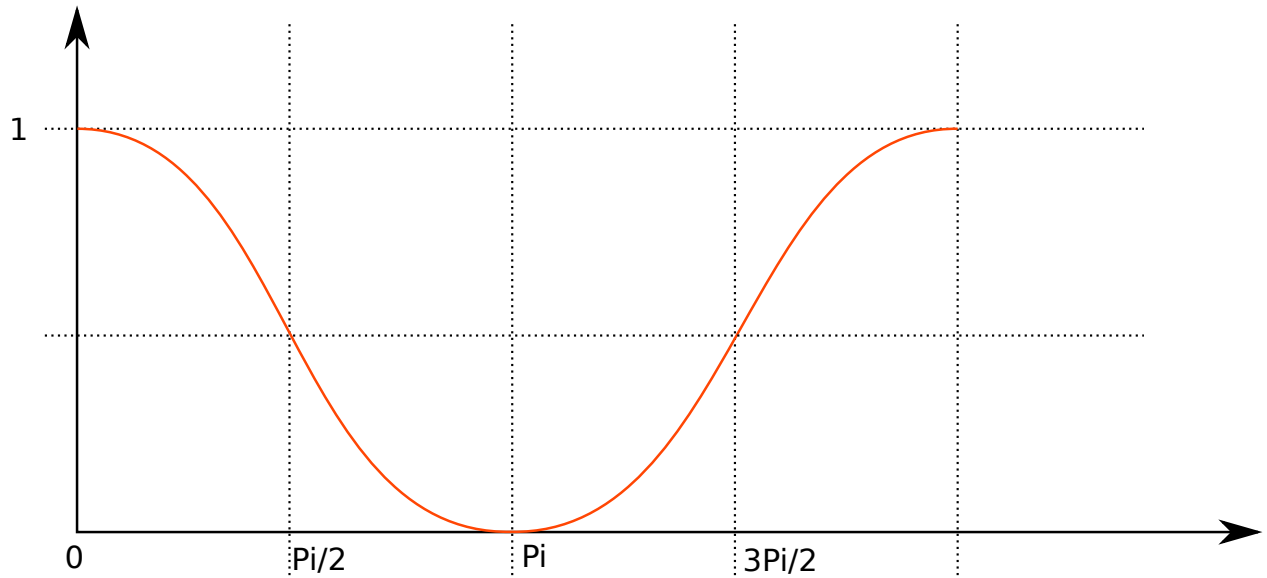
Ajoutons des images dans les layouts.
Utilisation de evas_object_image

```
snprintf(buf, sizeof(buf), PACKAGE_DATA_DIR"/images/%s
    ", files[i % 7]);
img = evas_object_image_filled_add(
    evas_object_evas_get(item->obj));
evas_object_image_file_set(img, buf, NULL);
elm_object_part_content_set(item->obj, "cover.swallow"
    , img);
```

Rappel trigo

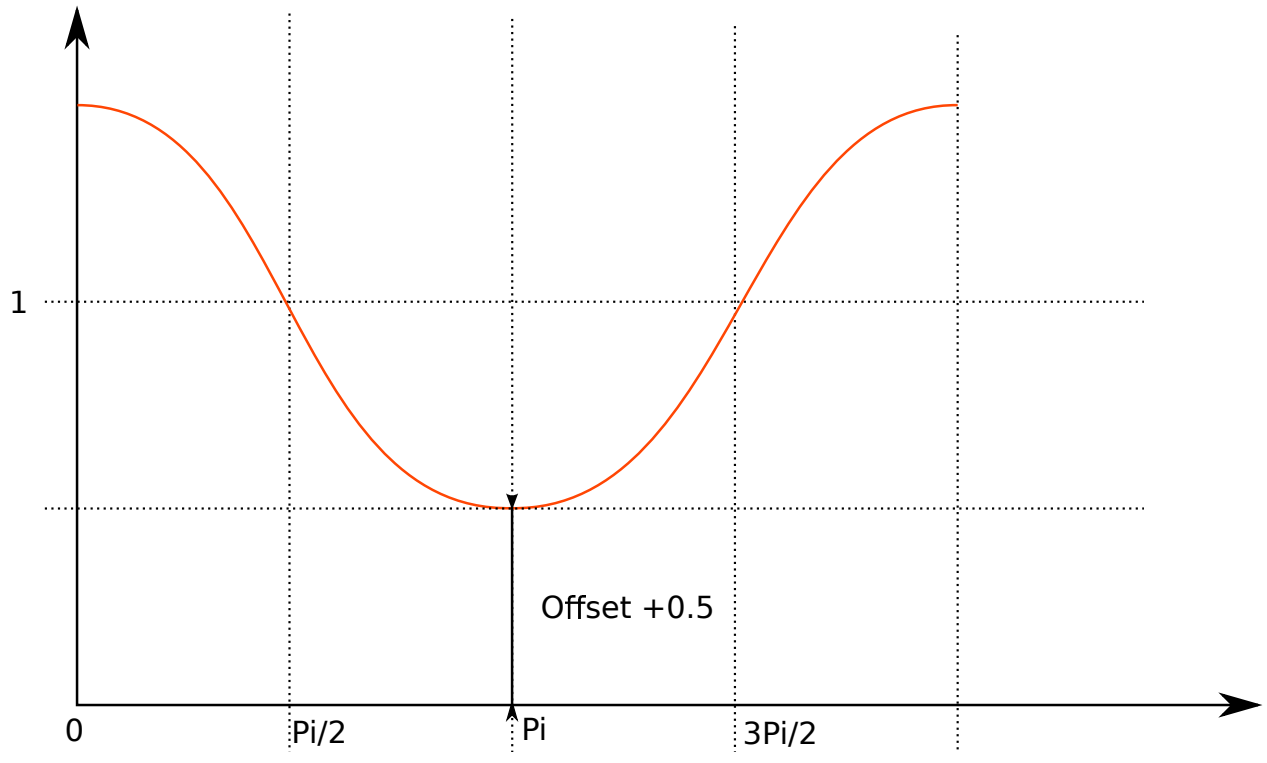


Rappel trigo



(1 + cos(a))/2

Rappel trigo



(1 + cos(a))/2

Facteur d'echelle des pochettes

Les pochettes sont soumis a un facteur multiplicateur. La taille des pochettes varie entre 0.5 x taille et 1.5 x taille.

```
scale = 0.5 + (1.0 + cos(item->angle + t)) / 2.0;
...
w = ICON_SIZE_W * scale;
h = ICON_SIZE_H * scale;
```

Position en X

La position des pochettes sur l'axe X est la projection des pochettes sur un plan. La position du centre des pochettes est donc fonction du sinus de l'angle.

```
x = WIDTH / 2.0 + (sin(item->angle + t) * (WIDTH /
    2.0)) - ICON_SIZE_W  / 2.0;
```

Position en Y

La position en y pourrait être calcullée en fonction du cosinus de l'angle. On choisis ici un calcul plus simple en réutilisant la valeur de scale calculée précédement.

```
1 y =  128 * scale;
```

Probléme de positionnement en z

Les pochettes sont maintenant bien positionnées, mais ne sont pas ordonnées correctement, visuellement certaines devraient se retrouver derriére certaines autres.
Pour positionner correctement les pochettes nous allons utiliser le scale de chaque pochette.
Plus le scale est faible, plus la pochette et loin.
Nous trions donc la liste des objets en fonction du scale a l'aide de la fonction eina_list_sorted_insert.
Puis on parcours la liste est on ordonne en utilisant evas_object_raise.

Fonction de tri

```
1  static int _compare_z(const void *data1, const void *
       data2)
   {
3      int d1 = evas_object_data_get(data1, "scale");
       int d2 = evas_object_data_get(data2, "scale");
5
       if (d1 > d2)
7          return 1;
       else if (d1 < d2)
9          return -1;
       else
11         return 0;
   }
13 ...
   z = eina_list_sorted_insert(z, _compare_z, item->obj);
15 ...
   EINA_LIST_FREE(z, obj)
17     evas_object_raise(obj);
```

Animons le tout

Crééons un ecore animator, qui va executer une callback au framerate défini dans ecore, par défaut à 30FPS.

```
ecore_animator_add(_anim_cb, NULL);
```

la fonction _anim prends maintenant un paramétre de type double : t. Ce parramétre est le temps courrant, ce qui permet d'animer notre carrousel en ajoutant cette valeur a l'angle des objets dans les calculs de cosinus et sinus.

```
static Eina_Bool
_anim_cb(void *data)
{
     _anim(ecore_loop_time_get());
     return EINA_TRUE;
}
```