Cours d'Interfaces Graphiques n°5

Edouard THIEL

Faculté des Sciences
Université d'Aix-Marseille (AMU)

Janvier 2016

Les transparents de ce cours sont téléchargeables ici :
http://pageperso.lif.univ-mrs.fr/~edouard.thiel/ens/igra/

Lien court : http://j.mp/optigra

Plan du cours n°5

1. Affichage de texte avec Pango
2. Coordonnées du texte
3. Fonctions variadiques
4. Mini-langage de balises
5. Timeout et animations

1 - Affichage de texte avec Pango

Pango = Pan (all) + Go (languages)

Librairie pour afficher du texte

- Internationalisation : UTF-8, orientations
- Composition : caractères latin, japonais, chinois, persan, arabe, thai, hébreu, braille, mathématiques... > 115
- Rendu : FreeType Fonts, Win32 fonts (Windows), CoreText (MacOS)
- Intégration avec Cairo

Gestion complète du texte

- découpage du texte intégral en paragraphes
- détermination du sens de lecture
- application de la police de caractère
- découpage en segments insécables
- calcul du nombre de lignes à afficher
- positionnement des segments sur chaque ligne

Le PangoLayout

Objet dérivant de `GObject`

Contient tous les paramètres d'affichage du texte

Le créer puis le détruire

```
gboolean on_area_draw (GtkWidget *area, cairo_t *cr,
                       gpointer data)
{
  PangoLayout *layout = pango_cairo_create_layout (cr);

  // Ici on affiche du texte en utilisant layout

  g_object_unref (layout);
  return TRUE;   // événement traité
}
```

Affichage simple

```
// On attache le texte au layout
pango_layout_set_text (layout,
    "Voici du texte\nsur deux lignes", // en UTF-8
    -1);                               // strlen

// On affiche le layout
pango_cairo_show_layout (cr, layout);
```

Changer la police

Décrire la police à partir d'une chaîne :

```
PangoFontDescription *desc;

desc = pango_font_description_from_string ("Serif, bold 12");
pango_layout_set_font_description (layout, desc);
pango_font_description_free (desc);
```

Liste de polices séparées par des virgules,
suivi des options et de la taille :

```
"Century Schoolbook L, Courier 10 Pitch, Ubuntu,
 DejaVu Sans, bold italic 14"
```

Nom des polices : voir "Visionneur de police"

Taille de la police

On peut changer la taille de la police courante :

```
PangoFontDescription *d1, *d2;

d1 = pango_layout_get_font_description (layout);
d2 = pango_font_description_copy (d1);
pango_font_description_set_absolute_size (d2,
    size * PANGO_SCALE);
pango_layout_set_font_description (layout, d2);
pango_font_description_free (d2);
```

`size` en points ; `PANGO_SCALE` pour conversion

Paragraphes

Un paragraphe = texte terminé par `\n` ou `\0`

Par défaut, Pango affiche un paragraphe par ligne

Afficher les paragraphes sur plusieurs lignes
de largeur max `width` en pixels :

```
pango_layout_set_width (layout, width*PANGO_SCALE);
```

Revenir au comportement par défaut :

```
pango_layout_set_width (layout, -1);
```

Exemple

```
pango_layout_set_text (layout,
  "Ceci est un paragraphe d'une certaine longueur.\n"
  "Et voici un autre paragraphe.", -1);
pango_cairo_show_layout (cr, layout);
```

Affiche :

Ceci est un paragraphe d'une certaine longueur.
Et voici un autre paragraphe.

```
pango_layout_set_width (layout, 200*PANGO_SCALE);
pango_cairo_show_layout (cr, layout);
```

Affiche :

Ceci est un
paragraphe d'une
certaine longueur.
Et voici un autre
paragraphe.

Alignement

Par défaut, le texte est aligné à gauche. Pour modifier :

```
pango_layout_set_alignment (layout, alignement);
```

où `alignement` est `PANGO_ALIGN_LEFT`
`PANGO_ALIGN_CENTER`
`PANGO_ALIGN_RIGHT`

Ceci est un paragraphe d'une certaine longueur.
Et voici un autre paragraphe.

Ceci est un paragraphe d'une certaine longueur.
Et voici un autre paragraphe.

Ceci est un paragraphe d'une certaine longueur.
Et voici un autre paragraphe.

Justification

On peut également justifier le texte avec :

```
pango_layout_set_justify (layout, gboolean justify);
```

```
pango_layout_set_width (layout, 200*PANGO_SCALE);
pango_layout_set_justify (layout, TRUE);
pango_cairo_show_layout (cr, layout);
```

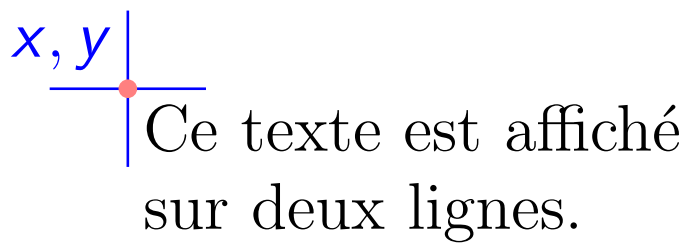
Affiche :

Ceci est un paragraphe d'une certaine longueur. Et voici un autre paragraphe.

2 - Coordonnées du texte

Point de référence : le texte est affiché par rapport au coin en haut à gauche

```
pango_layout_set_text (layout,
  "Ce texte est affiché\nsur deux lignes.", -1);
cairo_move_to (cr, x, y);
pango_cairo_show_layout (cr, layout);
```



Taille du texte à afficher

On peut obtenir la taille du texte avant l'affichage, qui dépend

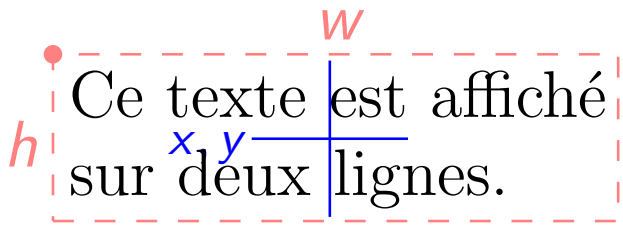
- des paragraphes
- de la largeur disponible $\rightarrow$ lignes
- de la police

```
int w, h;
pango_layout_get_pixel_size (layout, &w, &h);
```

Décalage du point de référence

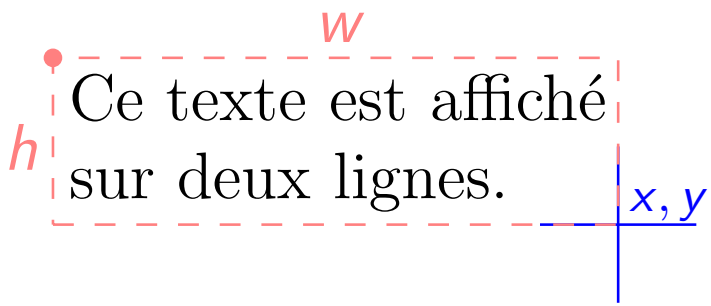
Affichage par rapport au centre :

```
int w, h;
pango_layout_get_pixel_size (layout, &w, &h);
cairo_move_to (cr, x - w/2., y - h/2.);
```



Affichage par rapport au coin droit en bas :

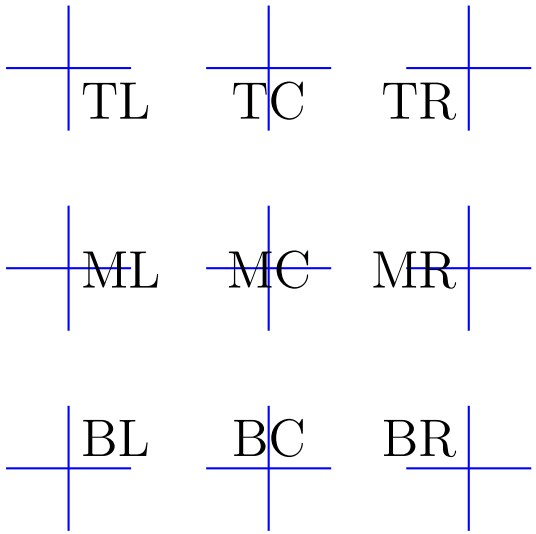
```
int w, h;
pango_layout_get_pixel_size (layout, &w, &h);
cairo_move_to (cr, x - w, y - h);
```



Fonction utilitaire de positionnement (1/3)

Positions relatives : Top, Middle, Bottom ; Left, Center, Right

```
typedef enum {
  FONT_BEGIN,
  FONT_TL,  FONT_TC,  FONT_TR,
  FONT_ML,  FONT_MC,  FONT_MR,
  FONT_BL,  FONT_BC,  FONT_BR,
  FONT_END
} Font_align;
```



Décodage d'un paramètre `Font_align align` :

```
align -= FONT_BEGIN + 1;
int halign = align % 3;
int valign = align / 3;
```

Fonction utilitaire de positionnement (2/3)

Décalage du point de référence :

```
int text_w, text_h;
pango_layout_get_pixel_size (layout, &text_w, &text_h);
cairo_move_to (cr,
    x - text_w * halign/2.,
    y - text_h * valign/2.);
```

Alignement horizontal du texte :

```
pango_layout_set_alignment (layout,
    halign == 0 ? PANGO_ALIGN_LEFT :
    halign == 1 ? PANGO_ALIGN_CENTER :
                  PANGO_ALIGN_RIGHT);
```

Fonction utilitaire de positionnement (3/3)

Module `font.c` à récupérer en TP

Usage :

```
int w = gtk_widget_get_allocated_width (area),
    h = gtk_widget_get_allocated_height (area);

font_draw_text (cr, layout, FONT_TL,   2,   2, "Top\nLeft");
font_draw_text (cr, layout, FONT_TC, w/2,   2, "Top\nCenter");
font_draw_text (cr, layout, FONT_TR, w-2,   2, "Top\nRight");
font_draw_text (cr, layout, FONT_ML,   2, h/2, "Middle\nLeft");
font_draw_text (cr, layout, FONT_MR, w-2, h/2, "Middle\nRight");
font_draw_text (cr, layout, FONT_BL,   2, h-2, "Bottom\nLeft");
font_draw_text (cr, layout, FONT_BC, w/2, h-2, "Bottom\nCenter");
font_draw_text (cr, layout, FONT_BR, w-2, h-2, "Bottom\nRight");
```

Utilisation comme `printf`

On voudrait pouvoir utiliser `font_draw_text` comme un `printf` :

```
font_draw_text (cr, layout, FONT_TL, x, y,
    "La taille du area est %d * %d pixels", w, h);
```

Solution : déclarer une fonction variadique

3 - Fonctions variadiques

```
#include <stdio.h>
#include <stdarg.h>

void somme (int x, int y, ...)
{
    int s = x + y, t;
    va_list ap;
    va_start (ap, y);
    while ((t = va_arg (ap, int)) != 0) s += t;
    va_end (ap);
    printf ("somme = %d\n", s);
}

int main () {
  somme (2, 5, 0);
  somme (2, 5, 8, 6, 0);
  somme (2, 5, 8, 6, 12, 7, 0);
}
```

Variantes de printf

```
#include <stdio.h>

int printf (const char *format, ...);
int fprintf (FILE *stream, const char *format, ...);
int sprintf (char *str, const char *format, ...);
int snprintf (char *str, size_t size, const char *format, ...);

#include <stdarg.h>

int vprintf (const char *format, va_list ap);
int vfprintf (FILE *stream, const char *format, va_list ap);
int vsprintf (char *str, const char *format, va_list ap);
int vsnprintf (char *str, size_t size, const char *format,
               va_list ap);
```

Utilisation de vsnprintf

Dans le module font.c en TP :

```
void font_draw_text (
        cairo_t *cr, PangoLayout *layout,
        Font_align align, double x, double y,
        const char *format, ...)
{
  char buf[10000];

  va_start (ap, format);
  vsnprintf (buf, sizeof(buf)-1, format, ap);
  va_end (ap);
  buf[sizeof(buf)-1] = 0;
  if (buf[0] == 0) return;

  ....
}
```

Autre exemple

```
void set_status (GtkWidget *status, const char *format, ...)
{
    char buf[1000];
    va_list ap;

    va_start (ap, format);
    vsnprintf (buf, sizeof(buf)-1, format, ap);
    va_end (ap);
    buf[sizeof(buf)-1] = 0;

    gtk_statusbar_pop  (GTK_STATUSBAR(status), 0);
    gtk_statusbar_push (GTK_STATUSBAR(status), 0, buf);
}
```

4 - Mini-langage de balises

Pango propose le langage de balises GMarkup, sous-ensemble de XML :

`& < > " '`	`& < > " '`
`
`	`\n`
`<b>Ga</b> <i>bu</i> <u>zo</u>`	**Ga** *bu* zo
`<span color="red">meu</span>`	meu

La balise accepte de nombreuses options :

```
<span font="Sans, italic 12" size="20"
      stretch="extraexpanded"
      color="red" bgcolor="#102030"
      alpha="50%" underline="error"
>Le texte</span>
```

Exemple

```
Le <span foreground="blue" size="x-large">texte bleu</span>
est <i>cool</i>&#10;
Le <span font="Verdana, bold 20">cœur</span> est
<u>souligné</u>&#10;
C'est une <span underline="error" underline_color="red">
erreur</span> qui est <s>barrée</s>&#10;
Le 1<sup>er</sup> de Z<sub>n</sub>
&amp; &lt; &gt; &quot; &apos;
```

Le **texte bleu** est *cool*
Le **cœur** est souligné
C'est une erreur qui est ~~barrée~~
Le 1er de Z_n & < > " '

Analyse du texte balisé

Méthode simple :

```
char *markup_text = "Hello <b>World</b>";

pango_layout_set_markup (layout, markup_text, -1);
pango_cairo_show_layout (cr, layout);
```

Remplace le texte actuel et les attributs

Inconvénient : pas de gestion des erreurs de syntaxe

Gestion des erreurs

```
char *markup_text = "Hello <b>World</b>";
PangoAttrList *attr_list;

int res = pango_parse_markup (markup_text, -1, 0,
  &attr_list, &markup_text, NULL, NULL);

if (!res)
  pango_layout_set_text (layout, "Syntax error.", -1);
else {
  pango_layout_set_text (layout, markup_text, -1);
  pango_layout_set_attributes (layout, attr_list);
  g_free (markup_text);
  pango_attr_list_unref (attr_list);
}

pango_cairo_show_layout (cr, layout);
```

5 - Timeout et animations

Pour réaliser une animation il faut un timeout, pour

- faire avancer l'animation :
  - calcul de nouvelles coordonnées ;
  - collisions, rebonds, ...
  - changement d'état éventuel ;
- provoquer un réaffichage du Drawing Area, avec
  - les coordonnées courantes ;
  - l'état courant.

Ne pas utiliser `sleep()` ou `usleep()`

Créer un timeout

```
guint g_timeout_add (guint interval,          // en ms
                     GSourceFunc func,
                     gpointer data);
```

La fonction `func` est appelée toutes les `interval` µs, jusqu'à ce qu'elle renvoie `FALSE`.

```
gboolean func (gpointer data)
{
  // Nouvelles coordonnées, collisions, etat...

  // Provoque le réaffichage du Drawing Area
  refresh_area (data->area);

  return TRUE;
}
```

Supprimer un timeout avant échéance

Stocker le numéro de timeout (> 0) renvoyé par `g_timeout_add`, puis le supprimer avec `g_source_remove` :

```
guint timeout1 = g_timeout_add (10, on_timeout1, data);
...
g_source_remove (timeout1);
timeout1 = 0;
```

Exemple d'animation (1/2)

```
typedef struct {                    |  void mydata_init (Mydata *my)
    GtkWidget *area1;               |  {
    int count;                      |    my->count = 0;
} Mydata;                           |  }

gboolean on_timeout1 (gpointer data)
{
  Mydata *my = data;
  my->count++;
  refresh_area (my->area1);
  return TRUE;
}

void on_app_activate (GtkApplication* app, gpointer data)
{
  Mydata *my = data;
  ...
  g_timeout_add (20, on_timeout1, my);
}
```

Exemple d'animation (2/2)

```
gboolean on_area_draw (GtkWidget *area, cairo_t *cr,
                       gpointer data) {
  Mydata *my = data;
  PangoLayout *layout = pango_cairo_create_layout (cr);
  pango_layout_set_text (layout, "Hypnose", -1);
  font_set_name (layout, "Sans Bold 24");

  int w = gtk_widget_get_allocated_width (area), h = ...;
  cairo_move_to (cr, w/2, h/2);
  cairo_rotate (cr, my->count * G_PI / 180.);

  cairo_set_source_rgb (cr, abs(my->count % 200 -100) / 100.,
                            abs(my->count % 300 -150) / 150.,
                            abs(my->count % 450 -225) / 225.);
  pango_cairo_show_layout (cr, layout);
  g_object_unref (layout);
  return TRUE;   // événement traité
}
```