

Projet C

Interaction Graphique



Amphi de présentation

F. Bérard 22 mai 2017

Introduction

Apprentissage du langage C

Par la pratique

Apprentissage de la gestion de projet

Par la pratique

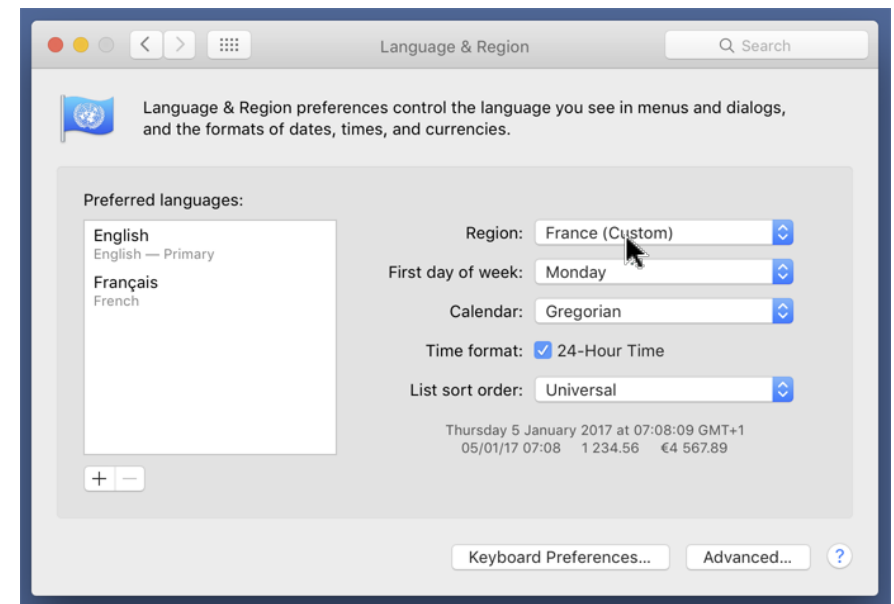
Travail en groupe

Travail d'envergure

Apprentissage de la programmation des interfaces graphiques

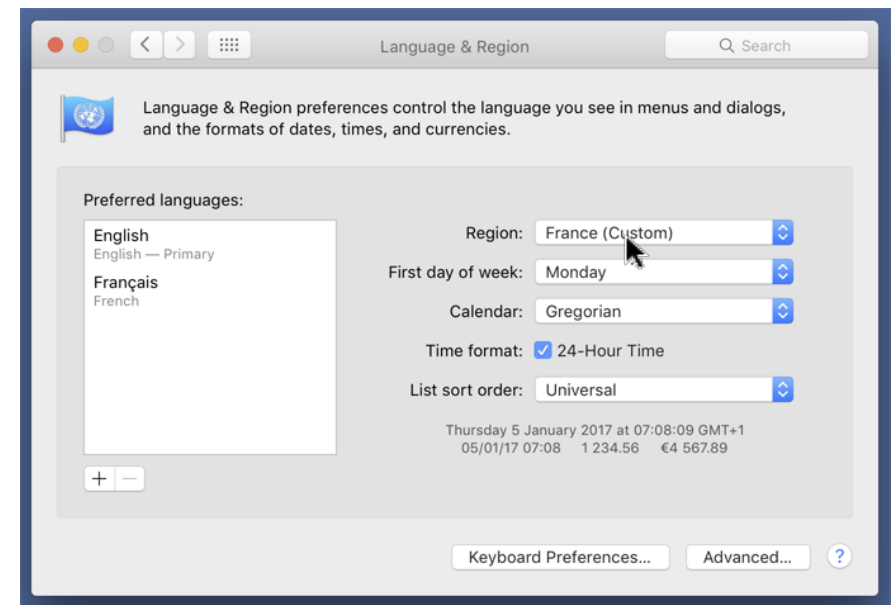
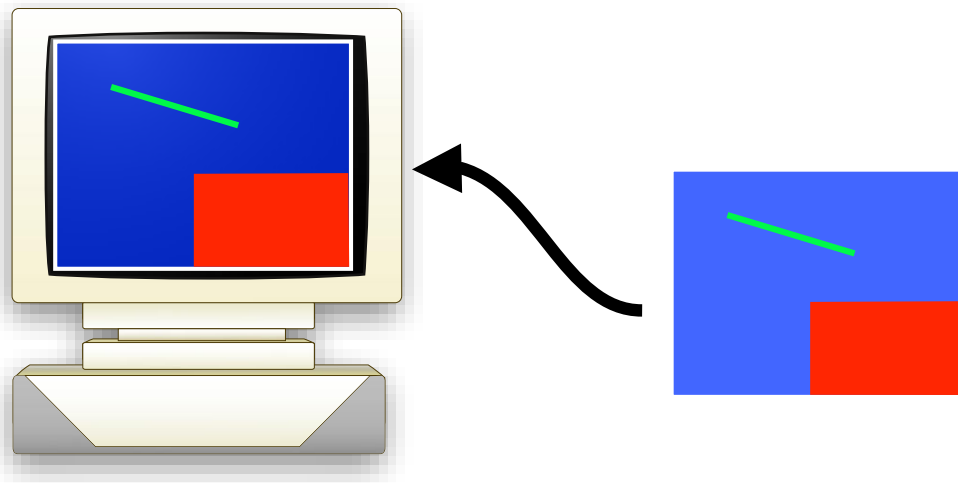
Par la réalisation d'une bibliothèque

Réalisation d'une bibliothèque de programmation d'Interfaces Graphiques



Réalisation d'une bibliothèque de programmation d'Interfaces Graphiques

En partant du *buffer graphique*



Réalisation d'une bibliothèque de programmation d'Interfaces Graphiques

En partant du *buffer graphique*,
des *événements utilisateur*

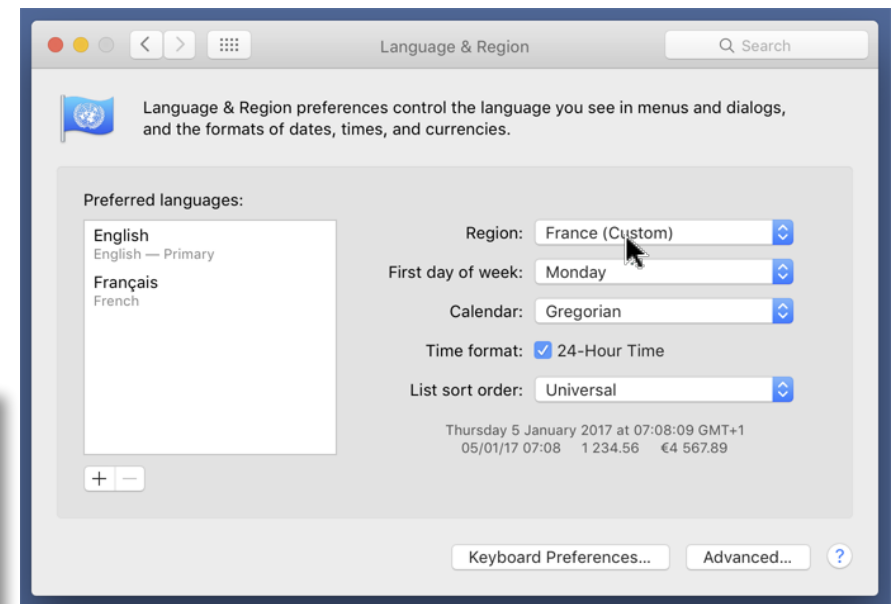
1876702	MouseMove	x=342 y=96
1876724	MouseMove	x=348 y=95
1877354	MouseButtonPress	n=1 1879265
	MouseMove	x=328 y=90
1879287	MouseMove	x=302 y=86
1879302	MouseMove	x=299 y=80
1881076	MouseButtonRelease	n=1
1892378	KeyPress	k="q"



Réalisation d'une bibliothèque de programmation d'Interfaces Graphiques

En partant du *buffer graphique*,
des *événements utilisateur*,
de la *structure* de la bibliothèque
(interface de programmation, ou API)
(fichiers .h fournis)

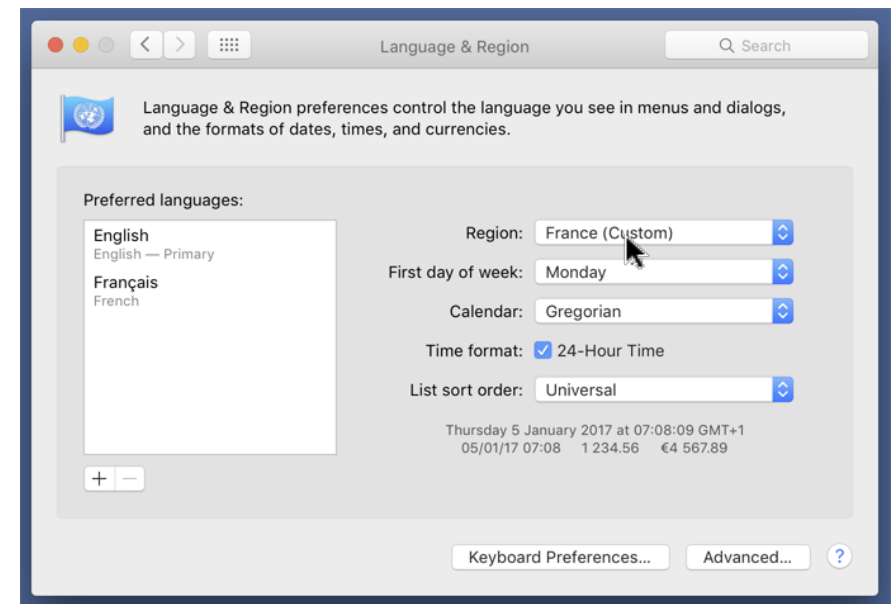
```
ei_widget_t* ei_widget_create  
             (ei_widgetclass_name_t class_name,  
             ei_widget_t* parent);
```



Réalisation d'une bibliothèque de programmation d'Interfaces Graphiques

En partant du *buffer graphique*,
des *événements utilisateur*,
de la *structure* de la bibliothèque,
et d'*exemples d'applications*.

```
ei_widget_t*   button;  
char          button_text[80] = "test";  
int           but_x = 10, but_y = 12;  
  
button = ei_widget_create("button",  
                          ei_app_root_widget());  
ei_button_configure(button, button_text);  
ei_place(button, &but_x, &but_y);
```



Les acteurs

Les **utilisateurs** de d'applications

Les **programmeurs d'applications**

Les programmeurs de la bibliothèque (**vous**)

Les **concepteurs** de la bibliothèque (nous, et vous en cas d'extensions)

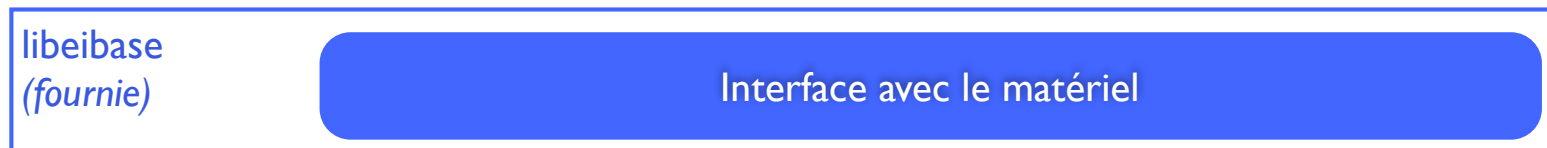
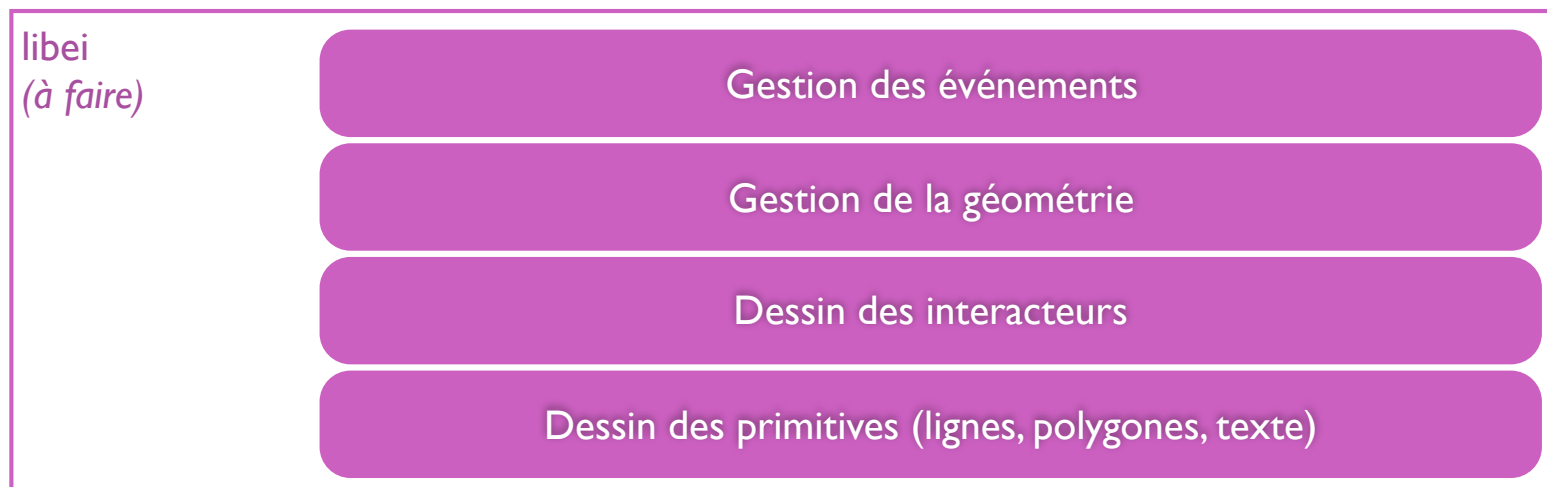
Survol

Dessin de **primitives graphiques** (lignes, polygones).

Création, configuration, et dessin des **interacteurs** (“**widgets**”)

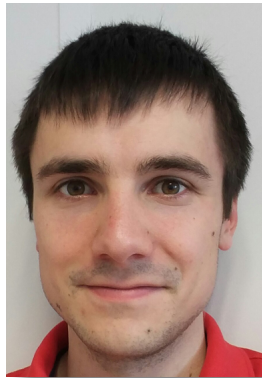
Placement à l'écran (position et taille) : gestion de la **géométrie**

Prise en compte des actions utilisateur : gestion des **événements**

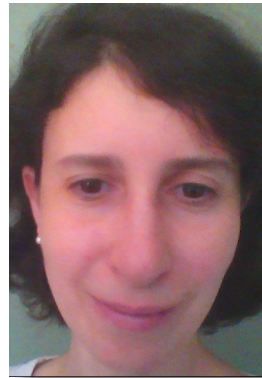


Organisation du projet

Les encadrants



Nicolas
Belleville
(NB)



Oum-El-Kheir
Aktouf
(OA)



Patrick
Reignier
(PR)



Yves
Panuel
(YP)



François
Bérard
(FB)

Les séances encadrées sont publiées sur ensiwiki.

Le site web du projet

Sur ensiwiki

http://ensiwiki.ensimag.fr/index.php/Nouvelles_du_projet_C_-_Interaction_Graphique

Les créneaux encadrés

8h45 => 12h45

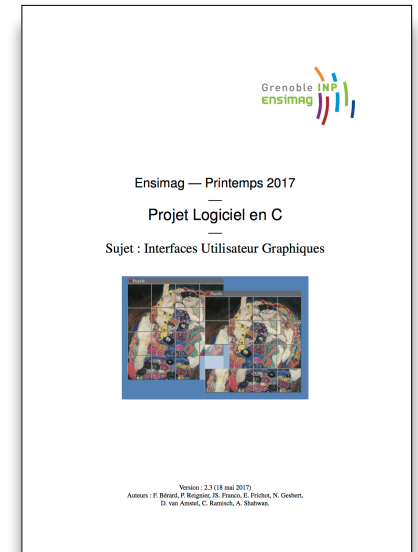
14h00 => 18h00

Salles **E200** & **E201** & **E212**

	Lun22	Mar23		Mer24		Lun29		Mar30		Mer31		Jeu1		Ven2		Mar6		Mer7
	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am
Francois		■		■	■			■					■				■	■
Patrick	■						■				■	■				■		
Oum-EI-Kheir		■	■							■				■			■	
Yves	■				■		■		■					■				
Nicolas			■			■				■			■				■	

Documentation

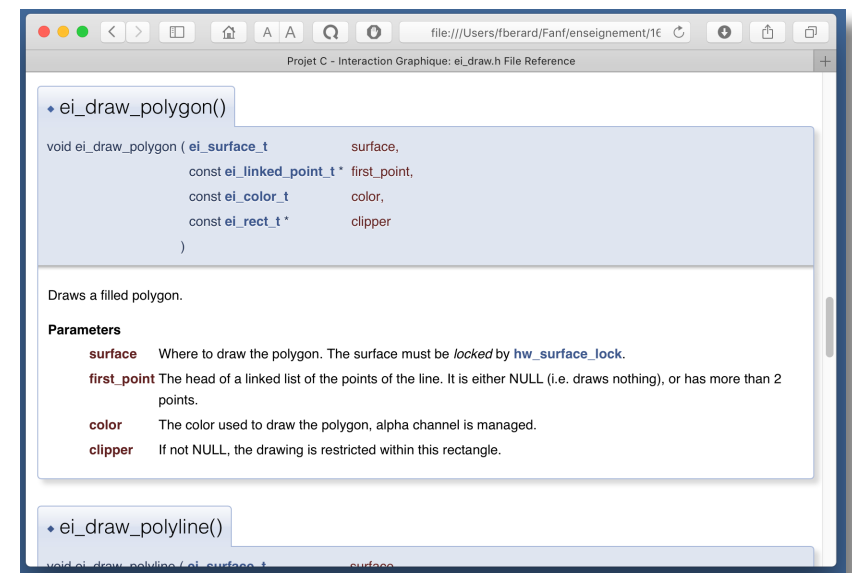
Toutes les notions nécessaires.
Dense.



Commentaires du code

Utilisation de Doxygen : un système de génération de documentation à partir des commentaires.

```
make doc  
open docs/html/index.html
```



L'archive des fichiers

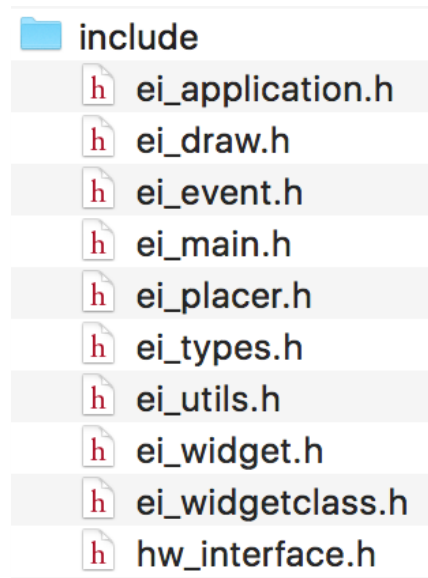
Téléchargeable depuis le site web du projet.

- Code compilé fourni (libeibase) →
 - Documentation doxygen →
 - Fichiers .h (API) →
 - Modèle de Makefile →
 - Fichiers annexes (fonte, image) →
 - Stockage des objets de compilation partielle (.o) →
 - Où mettre vos fichier source →
 - Exemples de code d'applications →
-
- | Component | Destination |
|---|-------------|
| Code compilé fourni (libeibase) | objs |
| Documentation doxygen | docs |
| Fichiers .h (API) | include |
| Modèle de Makefile | Makefile |
| Fichiers annexes (fonte, image) | misc |
| Stockage des objets de compilation partielle (.o) | objs |
| Où mettre vos fichier source | src |
| Exemples de code d'applications | tests |

L'archive des fichiers

Téléchargeable depuis le site web du projet.

Fichiers .h : interface de programmation de la bibliothèque

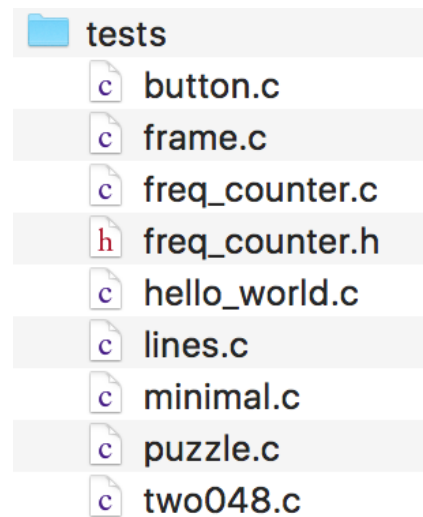


Interdiction absolue de modifier ces fichiers.
Placez vos déclarations dans d'autres fichiers.

L'archive des fichiers

Téléchargeable depuis le site web du projet.

Exemples de code d'applications.



Créez vos propres programmes de test !

Travail sur les machines personnelles

18

Possible et encouragé (libeibase.a fournie pour Mac OS, Linux), mais :

Les encadrants font du support uniquement pour le
Linux de l'Ensimag.

L'évaluation se fera uniquement sur le **Linux de l'Ensimag.**

Le projet nécessite la bibliothèque SDL et quelques dépendances.

http://ensiwiki.ensimag.fr/index.php/Projet_C_-_IG_-_Installation_de_SDL

Si vous développez sur vos machines personnelles, testez
très régulièrement que tout fonctionne à l'Ensimag.

Ne vous laissez pas déborder

Sollicitez de l'aide (de vos partenaires, des encadrants)

Demandez si votre travail correspond aux attentes

Travaillez sérieusement (Projet C = gros coefficient)

Développement

Avant de vous lancer dans le code :

- lire la documentation,
- acquérir une compréhension globale du projet,
- la partager avec les membres du groupe,
- se répartir les tâches.

L'annexe A du document vous suggère les premières étapes de développement.

Extensions

Si vous avez complètement réalisé l'API spécifiée dans le répertoire "include", alors vous pouvez développer des extensions.

La section 4.2 du document propose un ensemble d'extensions (clipping optimisé, nouvelles classes d'interacteur).

Ce ne sont que des suggestions, vous pouvez proposer vos propres idées d'extensions : parlez-en aux encadrants.

Chronologie

Mercredi (7/6) 19h

vous rendez les fichiers de votre projet sur TEIDE

Jeudi (8/6) matin

vous préparez votre soutenance

Jeudi (8/6) après-midi et vendredi (9/6) matin

vous faites une soutenance devant un encadrant
(40min, détail dans le document)

Critères d'évaluation

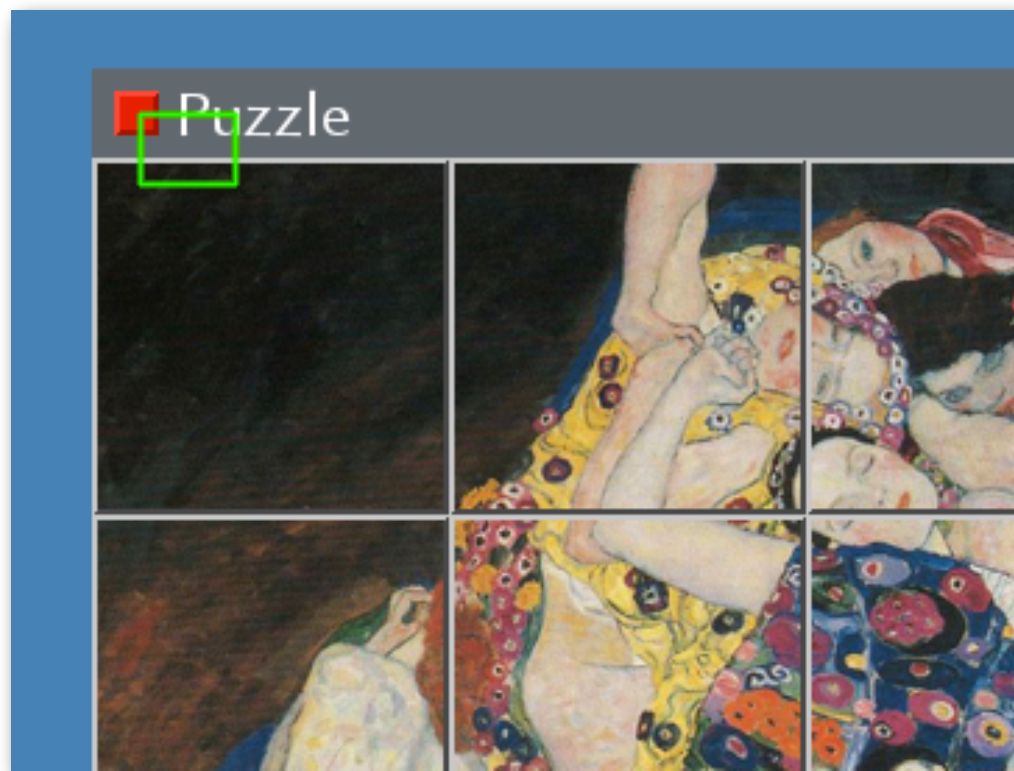
1. Exactitude : le projet fait ce qui est demandé.
2. Qualité de la structure de votre code (modules, fonctions).
3. Qualité de la forme du code (identificateurs, indentation, commentaires).
4. Extensions réalisées.

Rappel sur la fraude

- Il est interdit de copier ou de s'inspirer de fichiers concernant le projet, à l'exception des fichiers fournis par les encadrants.
Ceci inclus :
 - les fichiers des années précédentes
 - les fichiers d'étudiants n'appartenant pas au trinôme
- Il est interdit de transmettre des fichiers à des étudiants extérieurs au trinôme.
Il est de votre responsabilité de protéger vos fichiers en lecture (c.f. "Travailler à plusieurs" sur ensiwki).
- Des outils de détection automatique de triche sont utilisés.
- En cas de fraude avérée, la sanction est le 0 au projet, en plus des sanctions prévues dans le règlement de l'école.

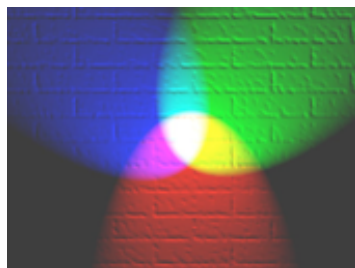
Génération d'Images Numériques

Représentation en mémoire

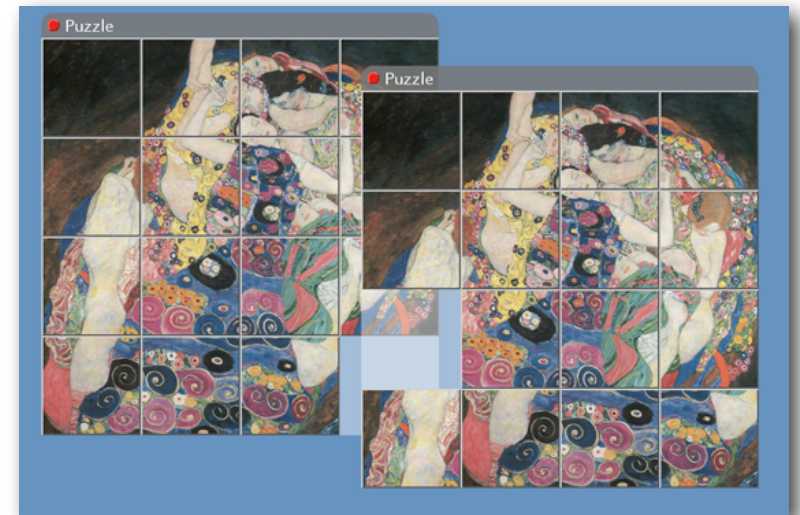
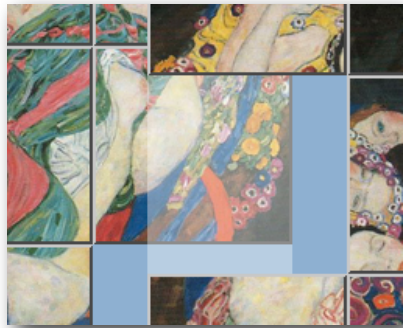


Représentation en mémoire

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
1	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
2	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
3	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
4	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
5	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
6	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
7	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
8	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
9	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
10	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
11	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
12	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
13	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB
14	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB	RVB



Effet de transparence



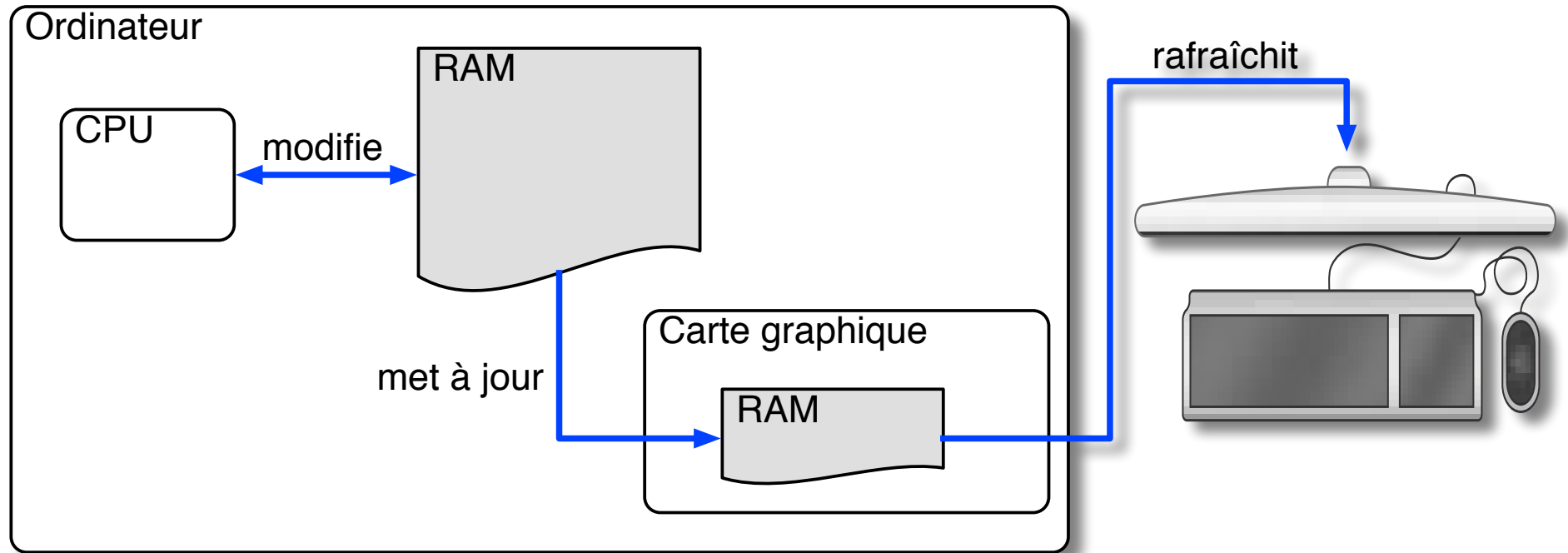
Dessin des “objets” du plus profond au plus proche

Le pixel résultat est une combinaison du pixel présent et du nouveau pixel à dessiner, en utilisant l' α du nouveau pixel :

$$D_R = (D_R \cdot (255 - S_\alpha) + S_R \cdot S_\alpha) / 255$$

$$D_G = \dots$$

$$D_B = \dots$$



Le programme n'agit pas directement sur la RAM de la **carte graphique** (i.e. pas sur l'écran).
Les mises à jour de l'image doivent être copiées sur la carte.

Cycle de mise à jour de l'écran

Une "surface" est bloquée

```
hw_surface_lock(my_surface);
```

Le programme modifie la surface

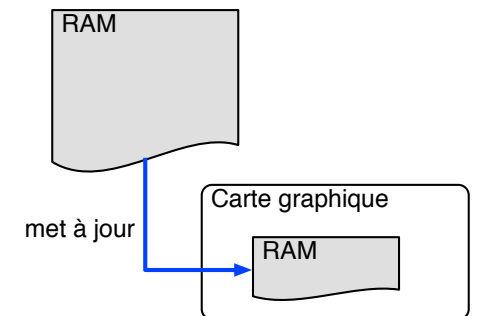
```
first_pixel = hw_surface_get_buffer(my_surface);  
*((uint32_t*)first_pixel) = some_pixel_value;  
ei_fill(my_surface, some_color, some_rectangle);
```

La surface est débloquée

```
hw_surface_unlock(my_surface);
```

Le programme demande la mise à jour de l'écran

```
hw_surface_update_rects(my_surface, the_rect_list);
```



Fonctions fournies dans “libeibase.a”

33

Fonctions déclarées dans “hw_interface.h” (2/2)

```
ei_surface_t hw_text_create_surface (const char*      text,  
                                   const ei_font_t   font,  
                                   const ei_color_t* color);  
  
ei_surface_t hw_image_load (const char*      filename,  
                           ei_surface_t     channels);  
  
void hw_event_wait_next(struct ei_event_t* event);  
double hw_now();
```

Fonctions déclarées dans “ei_draw.h” (1/2)

```
typedef struct {
    unsigned char    red;
    unsigned char    green;
    unsigned char    blue;
    unsigned char    alpha;
} ei_color_t;

uint32_t ei_map_rgba    (ei_surface_t surface, const ei_color_t* color);

void ei_draw_polyline  (ei_surface_t    surface,
                       const ei_linked_point_t* first_point,
                       const ei_color_t    color,
                       const ei_rect_t*    clipper);

void ei_draw_polygon   (...);
```

Fonctions déclarées dans “ei_draw.h” (2/2)

```
void ei_fill          (ei_surface_t      surface,  
                      const ei_color_t*  color,  
                      const ei_rect_t*    clipper);  
  
void ei_draw_text    (ei_surface_t      surface,  
                      const ei_point_t*  where,  
                      const char*        text,  
                      const ei_font_t    font,  
                      const ei_color_t*  color,  
                      const ei_rect_t*    clipper);  
  
int ei_copy_surface  (ei_surface_t      destination,  
                      const ei_rect_t*   dst_rect,  
                      const ei_surface_t source,  
                      const ei_rect_t*   src_rect,  
                      const ei_bool_t    alpha);
```

***Dessin des
Primitives Graphique***

Mise à jour de tout l'écran

1920 × 1080	= 2.07 M pixels
pixels RGBA (4 octets)	= 8.3 Mo
à 60Hz	→ 500 Mo/s

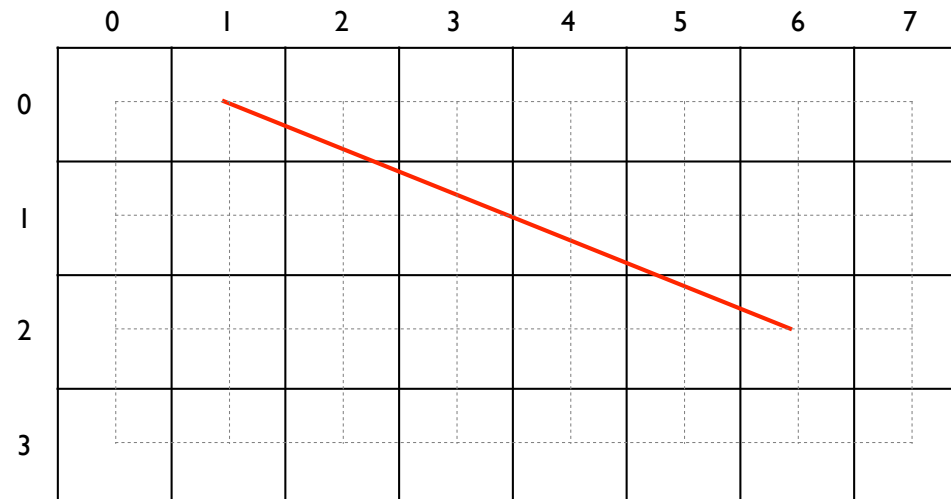
Animation (ex : bouger une grande fenêtre)

chaque opération sur un pixel × 2 million,
à faire 60 fois par secondes.

Les traitement de génération d'image doivent être **optimisés**.

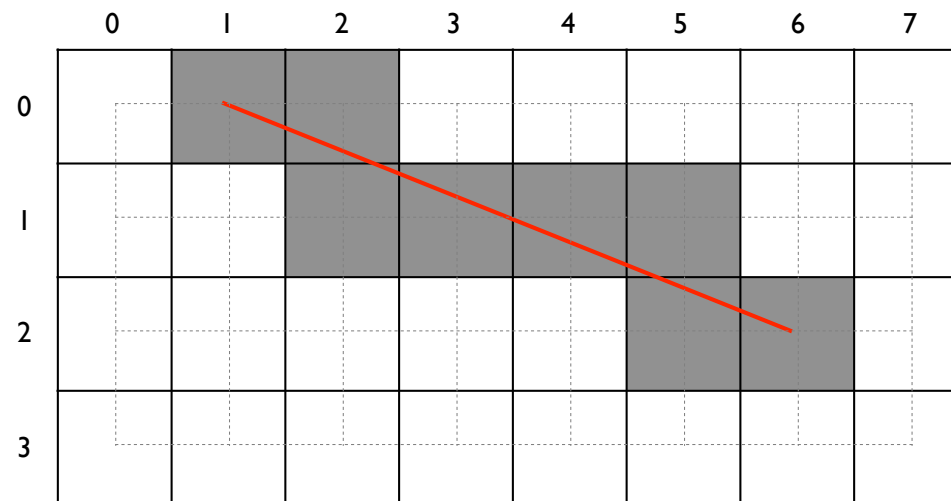
Dessin optimisé de lignes

ligne $((1,0),(6,2))$



Approche: tous les pixels qui touchent la ligne mathématique

ligne $((1,0),(6,2))$

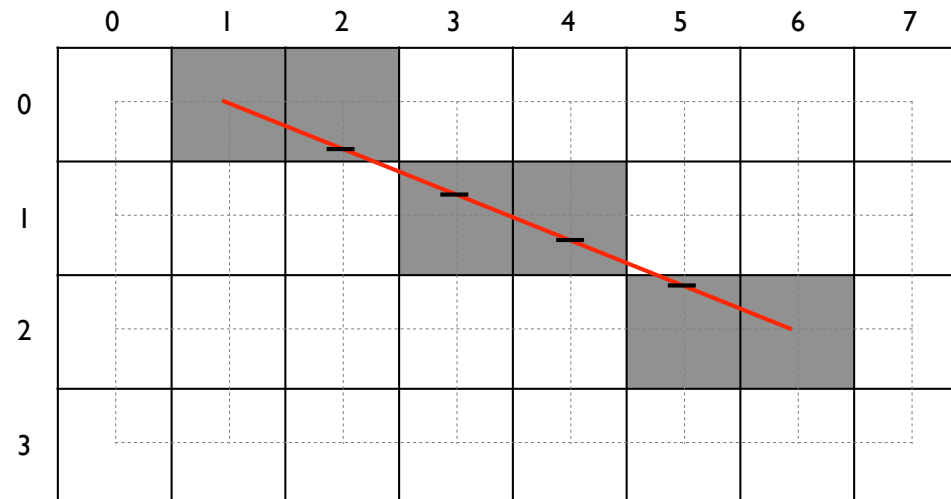


variations d'épaisseur visible, apparition d'angles

Dessin optimisé de lignes

Approche: un seul pixel par colonne : $-1 < \text{pente} < 1$
un seul pixel par ligne : $|\text{pente}| > 1$

ligne $((1,0),(6,2))$



Choix du pixel le plus proche.

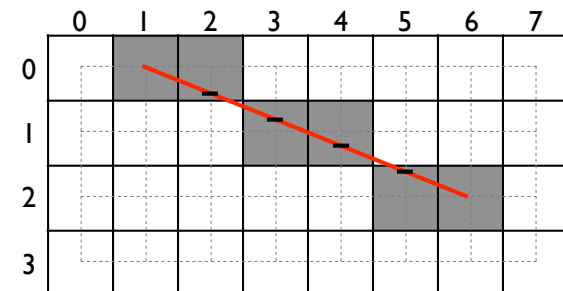
Arrondi de l'équation de droite

```
void draw_line(int x0, int y0, int x1, int y1)
{
    double    m    = (double)(y1 - y0) / (double)(x1 - x0);
    double    b    = (double)y0 - m * (double)x0;
    int       x_i  = x0;

    while (x_i < x1) {
        y      = m * (double)x_i + b;
        y_i    = (int)round(y);

        draw_pixel(x_i, y_i);
        x_i    += 1;
    }
}
```

ligne ((1,0),(6,2))



Arrondi de l'équation de droite

```
void draw_line(int x0, int y0, int x1, int y1)
{
    double    m    = (double)(y1 - y0) / (double)(x1 - x0);
    double    b    = (double)y0 - m * (double)x0;
    int       x_i  = x0;

    while (x_i < x1) {
        y      = m * (double)x_i + b;
        y_i    = (int)round(y);

        draw_pixel(x_i, y_i);
        x_i    += 1;
    }
}
```

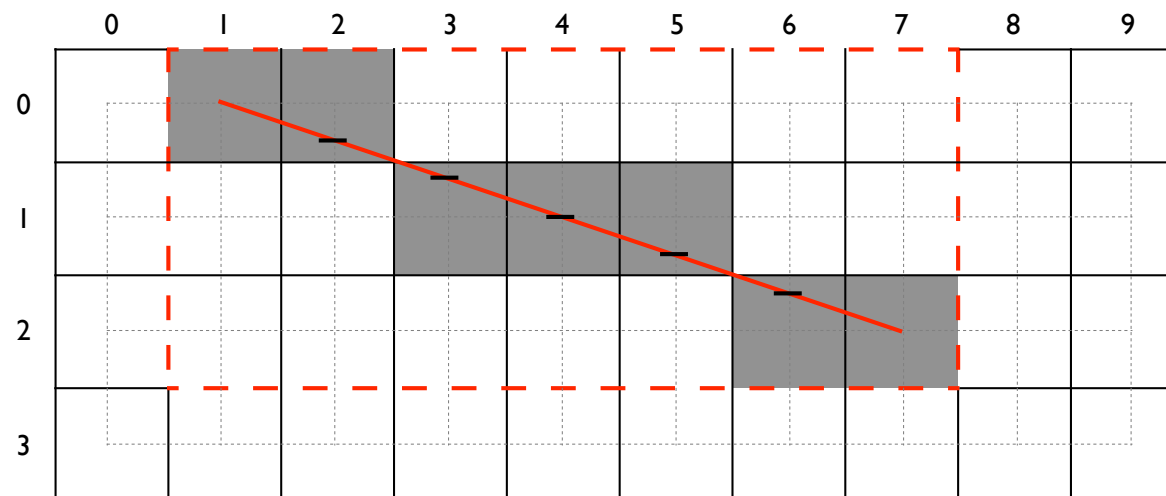
- Cast entier vers flottant
- Multiplication
- Arrondi
- Cast flottant vers entier

Version itérative

Quand x augmente de 1 , y augmente de $m = dy/dx$

Accumulation de l'erreur ε

y augmente de 1 quand $\varepsilon > 0,5$, alors $\varepsilon \leftarrow \varepsilon - 1$



Version itérative

```
void draw_line_iter(int x0, int y0, int x1, int y1)
{
    double      m      = (double)(y1 - y0) / (double)(x1 - x0);
    double      e      = 0.0;
    int         x_i     = x0;
    int         y_i     = y0;

    while (x_i < x1) {
        draw_pixel(x_i, y_i);

        x_i      += 1;
        e        += m;
        if (e > 0.5) {
            y_i   += 1;
            e     -= 1.0;
        }
    }
}
```

- Plus de cast !
- Plus de multiplication !
- Plus d'arrondi ! (mais un test)
- Opérations sur flottants

Dessin optimisé de lignes

Version itérative avec seulement des entiers

$E = \varepsilon \cdot dx$, soit $\varepsilon = E/dx$,

incrément de x , alors $E \leftarrow E + dy$

on teste $E > 0,5 \cdot dx$, ou $2E > dx$

quand $y \leftarrow y + 1$, alors $E \leftarrow E - dx$

Dessin optimisé de lignes

Version itérative avec seulement des entiers

```
void draw_line_iter_int(int x0, int y0, int x1, int y1)
{
    int          dx    = x1 - x0;
    int          dy    = y1 - y0;
    int          e     = 0;

    while (x0 < x1) {
        draw_pixel(x0, y0);

        x0          += 1;
        e           += dy;
        if (2*e > dx) {
            y0      += 1;
            e       -= dx;
        }
    }
}
```

[Bresenham 1965]

Généralisation à toutes les orientations

$dx = 0$ ou $dy = 0$, gérer le cas particulier

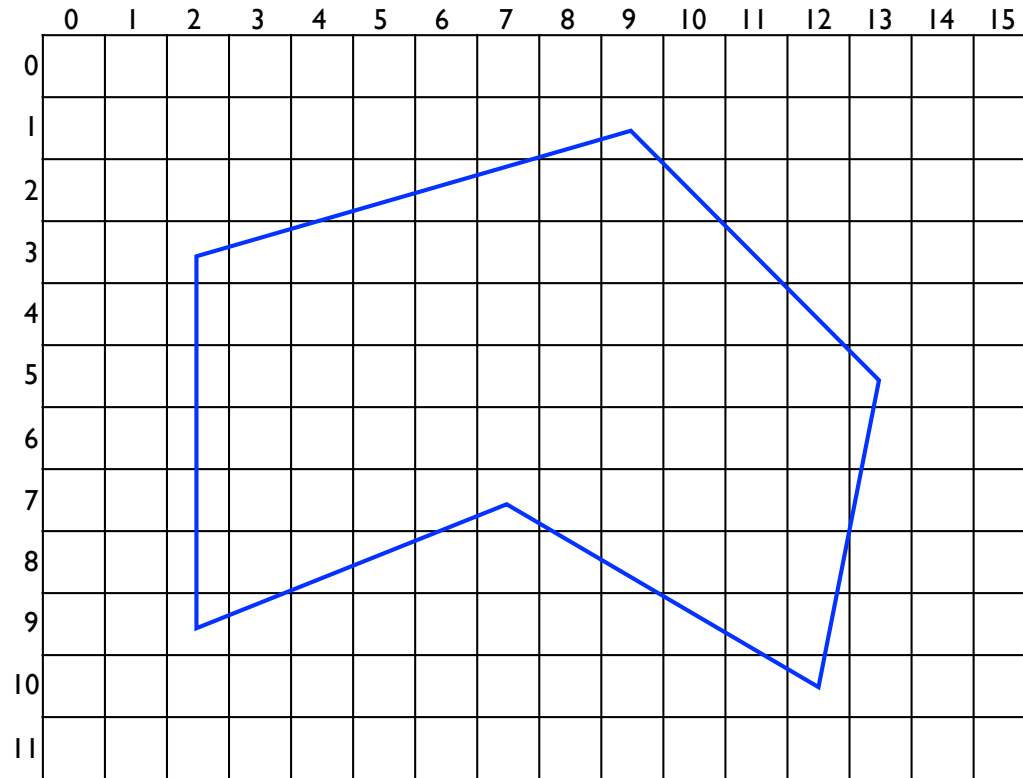
Pente négative, inverser les signes.

$x_0 > x_1$, inverser les signes

$|pente| > 1$, inverser les variables

Dessin optimisé de polygones (pleins)

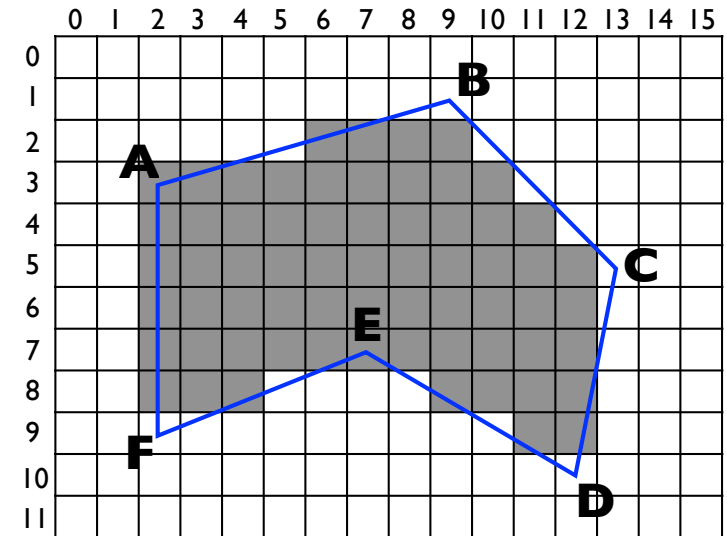
polygone ((2,3), (9,1), (13,5), (12,10), (7,7), (2,9))



Stratégie

Optimisation par exploitation des cohérences spatiales :

- Scanline : les pixels entre les intersection sont dans le même état (intérieur / extérieur)
- Côtés : si un côté intersecte la scanline, c'est le cas pour les scanlines suivantes jusqu'à y_{max}

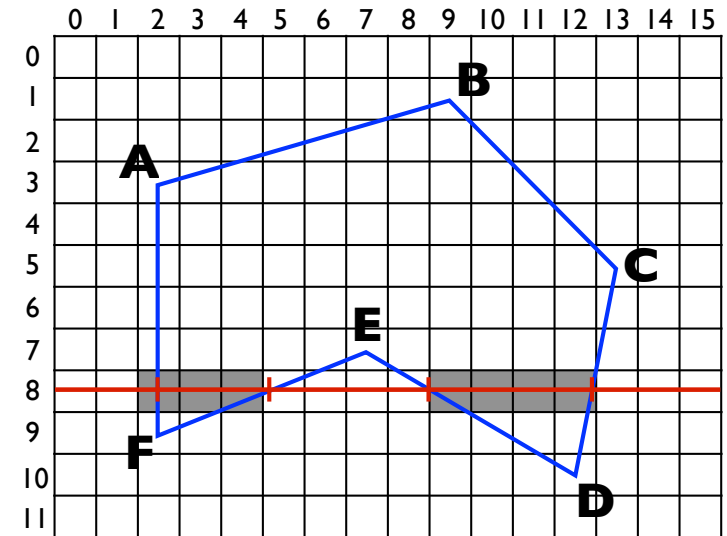


Dessin optimisé de polygones

50

Stratégie

- Localise les intersections entre côtés et scanline
- Règle de parité
 - Initialisation: pair
 - Chaque intersection inverse la parité
 - Les intervalles impairs sont intérieur
- Algorithme incrémental pour le calcul des intersections
- Stockage de l'état de chaque côté dans des tables



Dessin optimisé de polygones

- Algorithme incrémental pour le calcul des intersections

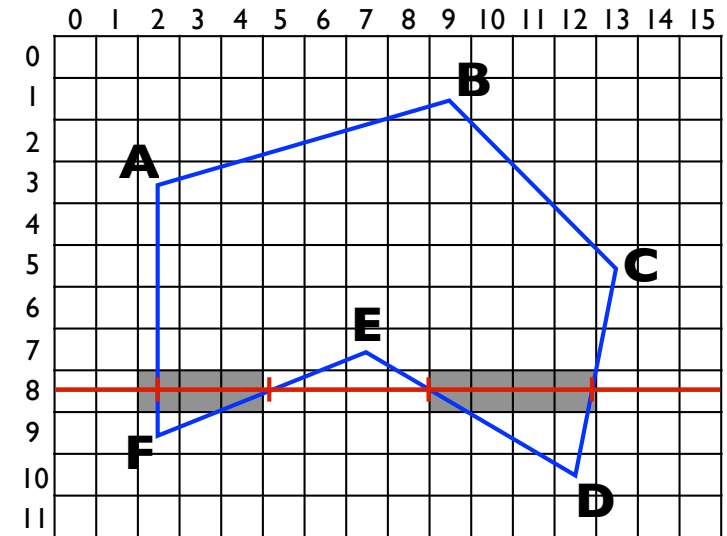
À chaque nouvelle scanline

$$y \leq y + 1,$$

augmente x de la réciproque de la pente

$$x \leq x + (x1 - x0) / (y1 - y0)$$

Version incrémentale, en entiers (avec cas *pente* > 1):
inspiré du tracé de segment de Bresenham.

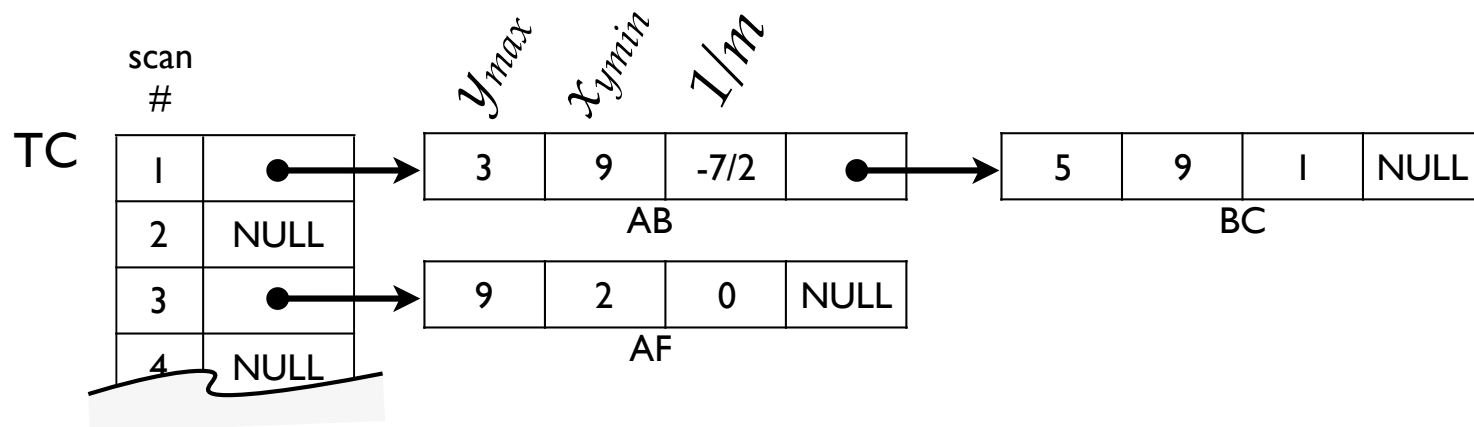
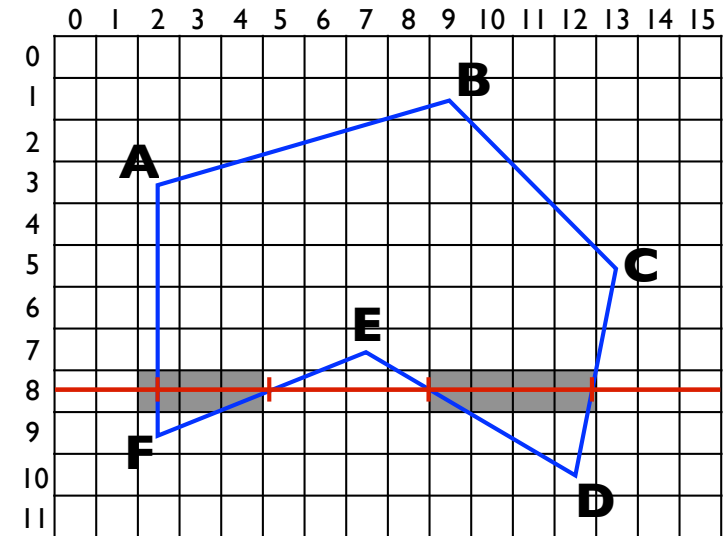


Dessin optimisé de polygones

- Stockage de l'état des côté dans des tables

Table des Côtés (TC)

- Une entrée par scanline
- Les entrées pointent vers la liste des sommets qui débutent sur la scanline
- La liste est triée par x

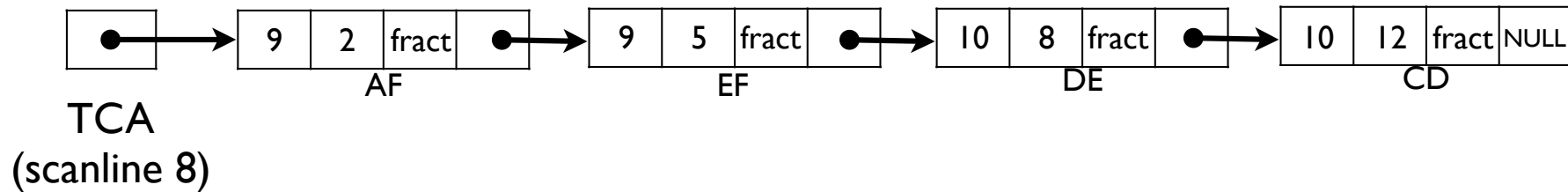
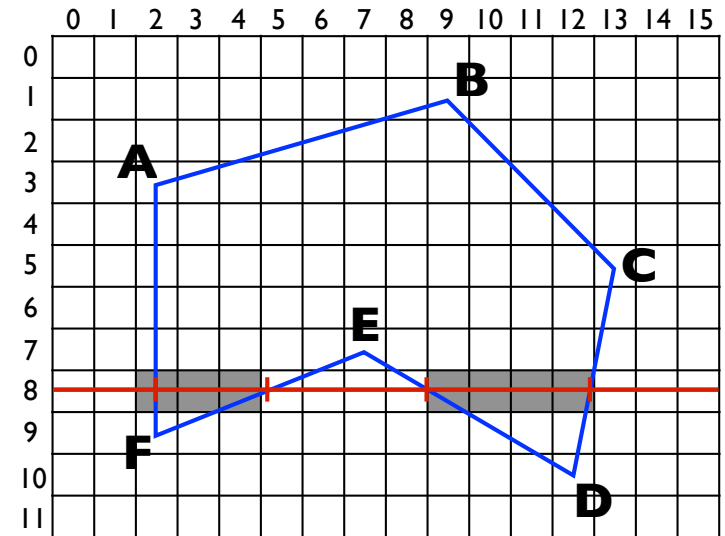


Dessin optimisé de polygones

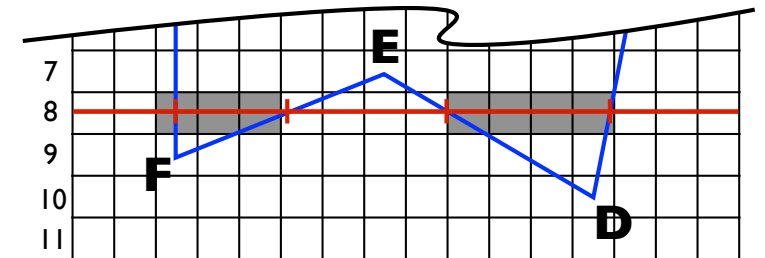
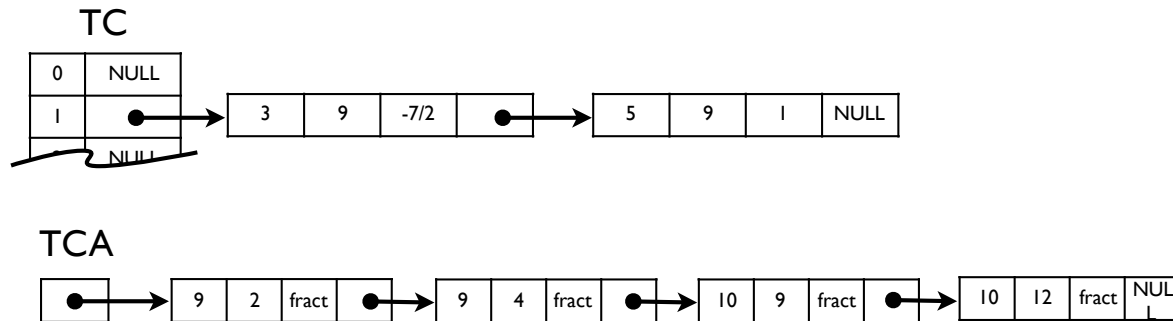
- Stockage de l'état des côté dans des tables

Table des Côtés Actifs (TCA)

- Triée par x croissants



Dessin optimisé de polygones



• Algorithme

- Initialise y à la première scanline, TCA à vide
- Répéter jusqu'à ce que TC et TCA soient vides
 - Déplacer les entrées de $TC(y)$ dans TCA
 - Supprimer de TCA les entrées $y_{max}=y$
 - Trier TCA sur x
 - Remplir les intervalles grâce à la règle de parité
 - Incrémente y
 - Mets à jour x dans les entrées de la TCA

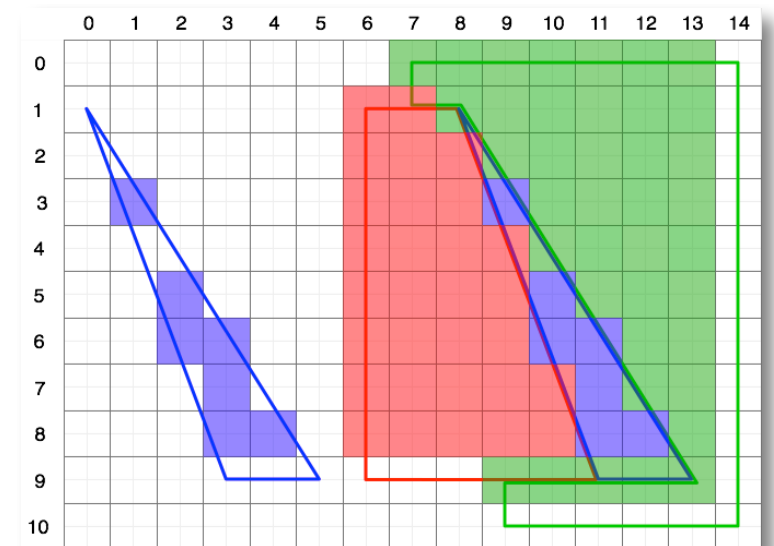
Dessin optimisé de polygones

55

- Localise les intersections entre côtés et scanline

Gestion des polygones adjacents

- Arrondi
 - entier supérieur pour le premier pixel de l'intervalle
 - entier inférieur pour le dernier
- Intersections sur coordonnées entières
 - seuls les pixels en entrée de l'intervalle en font partie
- Intersections partagées entre côtés
 - on compte uniquement celles définissant un y_{min}

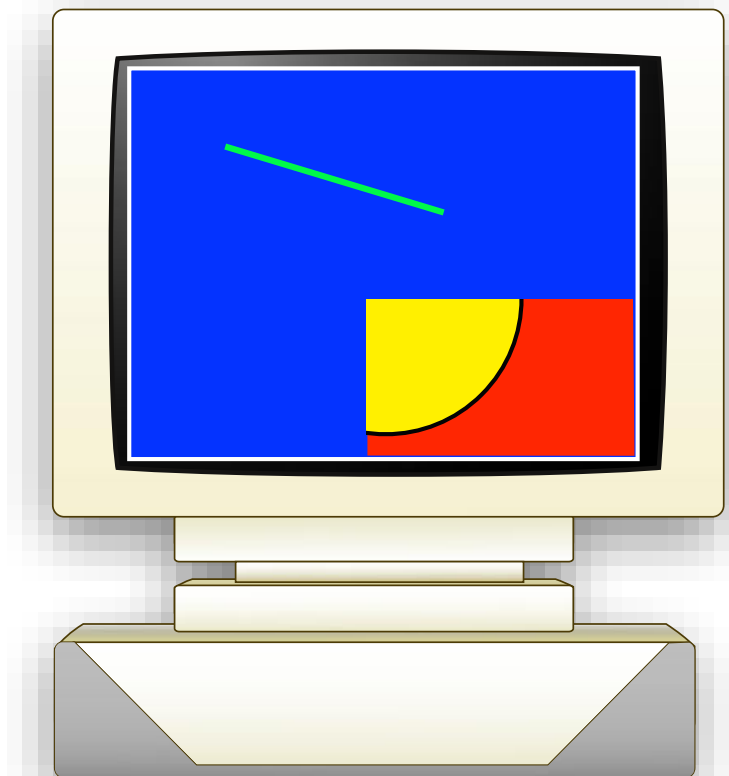


Clipping rectangulaire aligné à l'écran

Limiter le dessin d'une primitive à l'intérieur d'un rectangle aligné aux bords de l'écran

Utilité

- Ne pas sortir des limites de l'écran
- Ne pas sortir des limites du parent
- Limiter le re-dessin (optimisation)



Approches

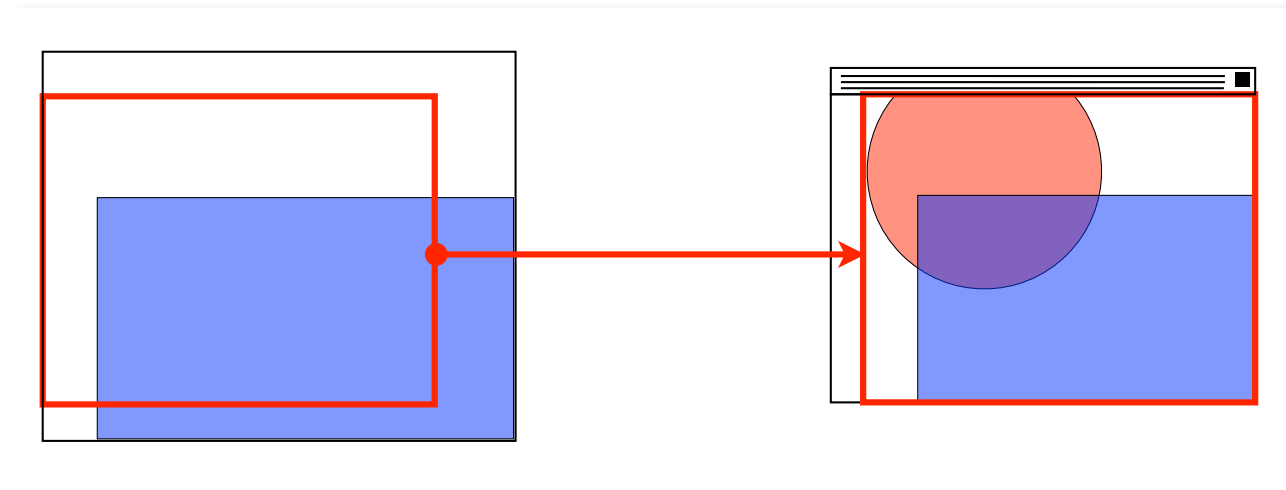
- Test avant écriture

```
if ((x >= xmin) && (x <= xmax) && (y >= ymin) && (y <= ymax))  
    draw_pixel(x, y);
```

- Ne nécessite pas de pré-calculs
- Coûteux en calculs (4 tests par pixel)
- N'exploite pas la cohérence spatiale (petit clipper sur grande forme)
- **À implémenter en premier !** (très simple à coder)

Approches

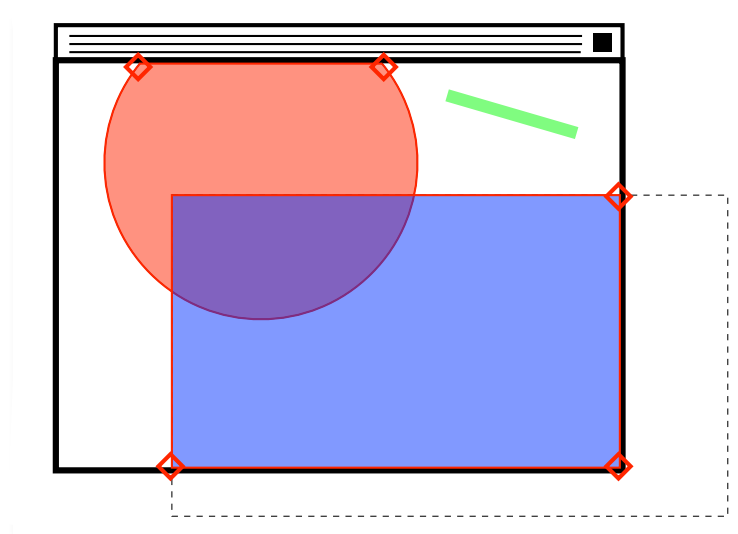
- Dessin offscreen, puis recopie



- Élimine les 4 tests dans la boucle interne
- Coûteux en mémoire
- N'exploite pas la cohérence spatiale (petit clipper sur grande forme)

Approches

- Calcul des intersections avec le clipper



- Pas de tests dans la boucle interne
- Optimise le nombre de pixels à traiter
- Complexe à réaliser, algorithme différent pour les différentes primitives
- Considéré comme **une extension du projet.**

Interface Utilisateur Graphique

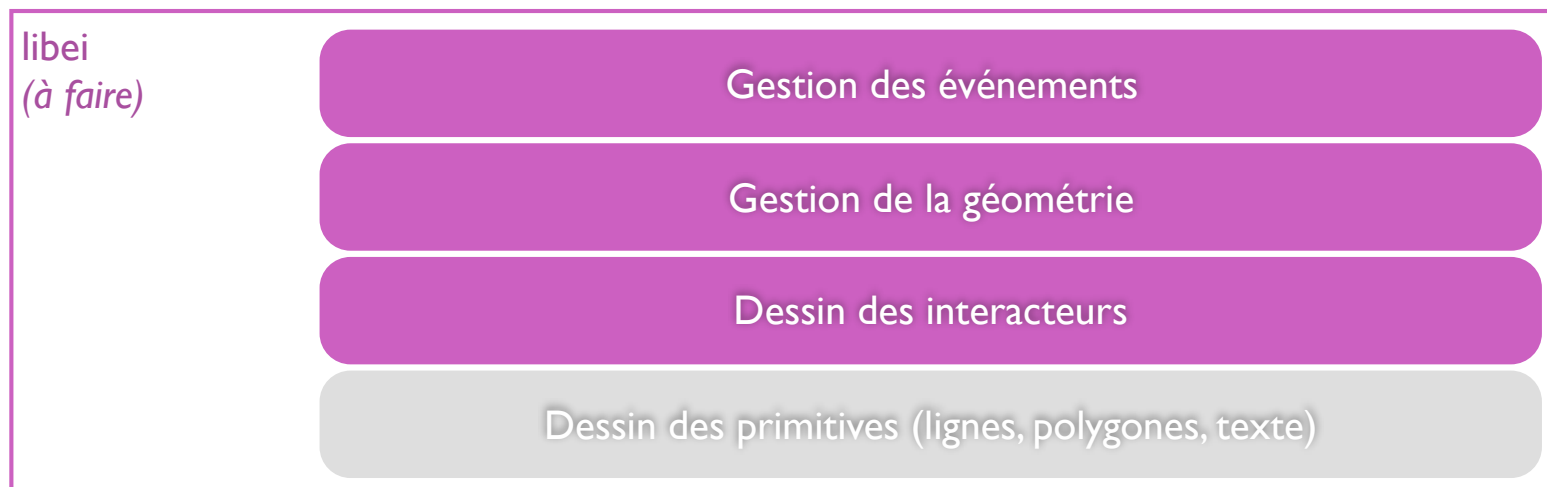
Survol

Dessin de **primitives graphiques** (lignes, polygones).

Création, configuration, et dessin des **interacteurs** (“**widgets**”)

Placement à l'écran (position et taille) : gestion de la **géométrie**

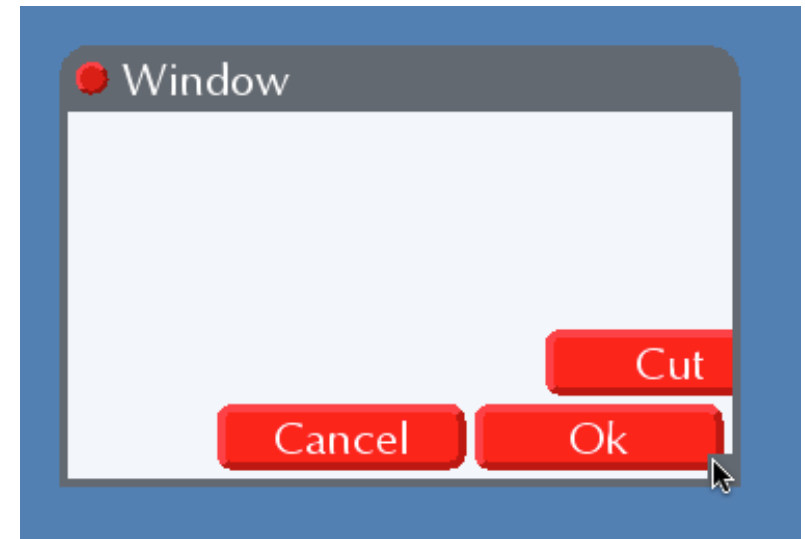
Prise en compte des actions utilisateur : gestion des **événements**



Organisation Hiérarchique

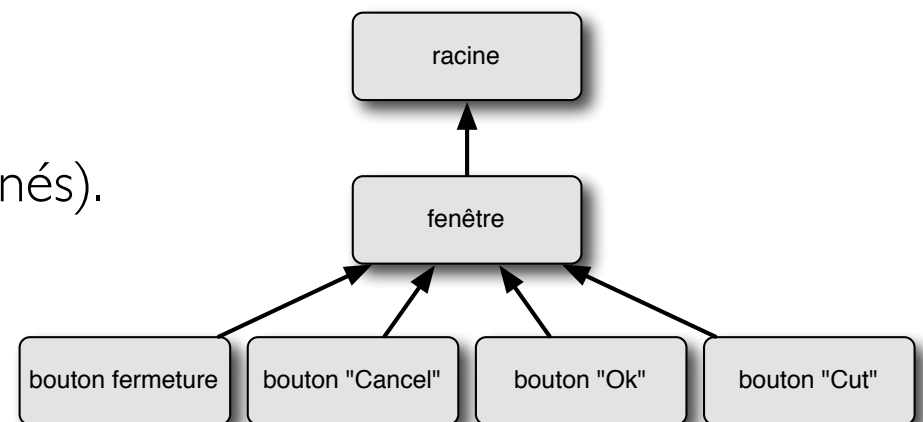
Tout interacteur :

- a un parent, hormis la **racine**,
- est **tronqué** (“clipped”) dans les limites de son parent,
- est positionné par rapport à son parent,
- est masqué avec son parent,
- est détruit avec son parent.



L'ordre de dessin est :

- en profondeur, puis,
- en largeur (les descendants sont ordonnés).



Classe d'interacteurs

Principe

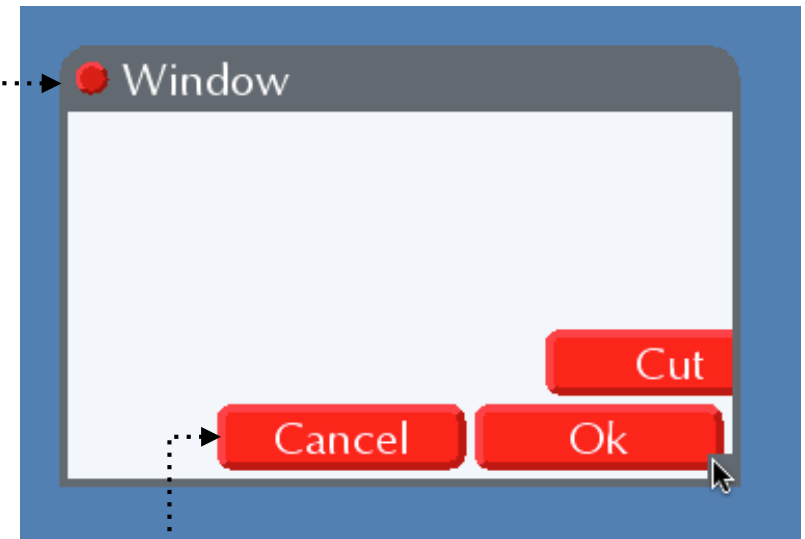
Tous les interacteurs partagent certaines caractéristiques *communes* (hiérarchie, géométrie, etc.)

Par contre, certaines caractéristiques sont *spécifiques* à une **classe d'interacteur**.

Exemples :

Fenêtre toplevel.....

Boutons



Mais aussi : champ de saisie, barre de défilement, case à cocher, etc.

Polymorphisme des interacteurs

Un **bouton**, par exemple, doit pouvoir être considéré :

- comme un **interacteur** pour les traitements communs à tout interacteur (hiérarchie, etc.).
- ou comme un **bouton** pour les traitements spécifiques aux boutons (dessin, etc.),

→ Nécessité d'un mécanisme de **polymorphisme**.

Représentation des interacteurs en mémoire

Attributs communs à tout interacteur.

```
typedef struct ei_widget_t {
    ei_widgetclass_t*    wclass;

    struct ei_widget_t*  parent;
    struct ei_widget_t*  children_head;
    struct ei_widget_t*  children_tail;
    struct ei_widget_t*  next_sibling;

    ei_size_t            requested_size;
    ei_rect_t            screen_location;
} ei_widget_t;
```

Représentation des interacteurs en mémoire

Ajout des attributs spécifiques à une classe donnée (ex: boutons).

```
typedef struct ei_widget_t {
    ei_widgetclass_t*    wclass;

    struct ei_widget_t*  parent;
    struct ei_widget_t*  children_head;
    struct ei_widget_t*  children_tail;
    struct ei_widget_t*  next_sibling;

    ei_size_t            requested_size;
    ei_rect_t            screen_location;
} ei_widget_t;
```

```
typedef struct {
    ei_widget_t          widget;

    int                  border_width;
    ei_relief_t          relief;
    char*                text;
} ei_button_widget_t;
```

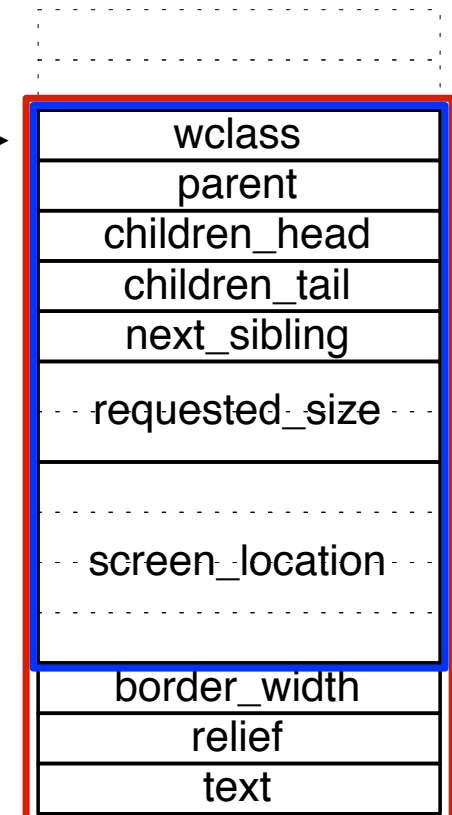
Représentation des interacteurs en mémoire

Ajout des attributs spécifiques à une classe donnée (ex: boutons).

```
typedef struct ei_widget_t {  
    ei_widgetclass_t*    wclass;  
  
    struct ei_widget_t*  parent;  
    struct ei_widget_t*  children_head;  
    struct ei_widget_t*  children_tail;  
    struct ei_widget_t*  next_sibling;  
  
    ei_size_t            requested_size;  
    ei_rect_t            screen_location;  
  
} ei_widget_t;
```

```
typedef struct {  
    ei_widget_t          widget;  
  
    int                  border_width;  
    ei_relief_t          relief;  
    char*                text;  
  
} ei_button_widget_t;
```

*ptr →

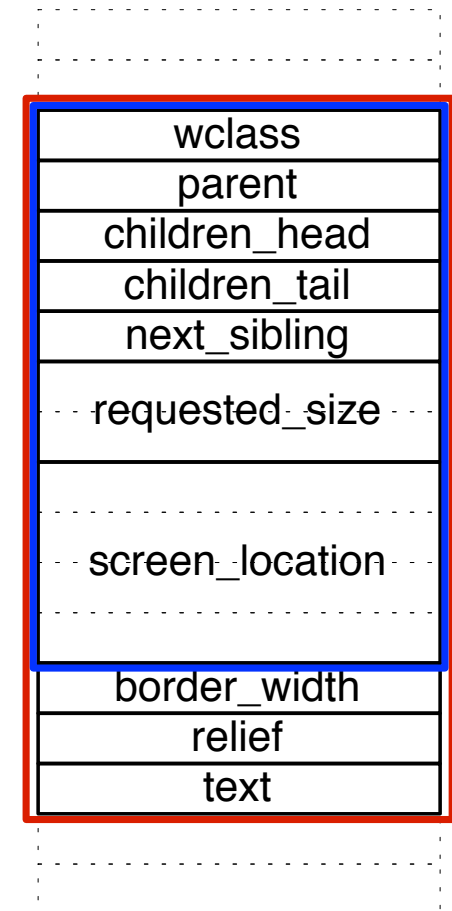


Polymorphisme des données

```
ei_button_widget_t*   button;
button = malloc(sizeof(ei_button_widget_t));

void init_button(ei_widget_button_t* button, ei_widget_t* parent)
{
    init_widget((ei_widget_t*)button, g_button_class, parent);
    button->border_width    = 1;
    button->relief          = ei_relief_raised;
    button->text            = (char*)NULL;
}

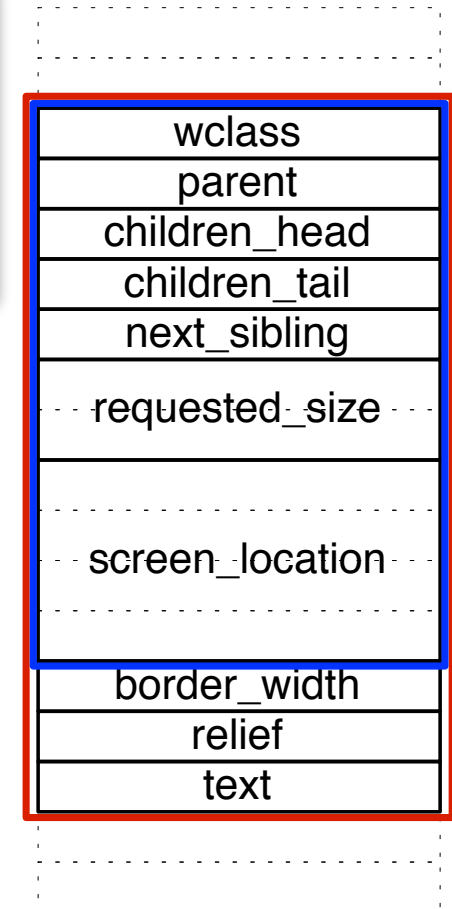
void init_widget(ei_widget_t* widget, ei_widgetclass_t* wclass,
                ei_widget_t* parent)
{
    widget->wclass          = wclass;
    widget->parent          = parent;
    widget_add_child(parent, widget);
    widget->children_head  = (ei_widget_t*)NULL;
    ...
}
```



Polymorphisme des traitements

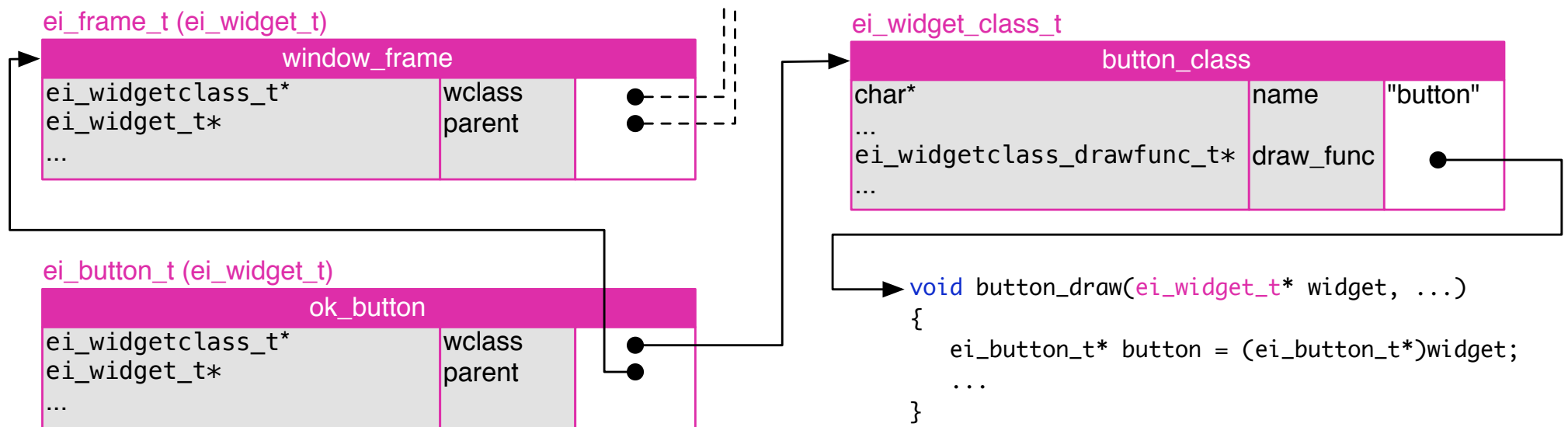
```
void widget_update_size(ei_widget_t* widget, ei_size_t* new_size)
{
    widget->requested_size.size = new_size;
    ...
    widget_draw(widget);
}
```

Comment appeler la fonction de dessin qui correspond à la classe de l'interacteur ?



Programmation des classes d'interacteurs

Polymorphisme des traitements



```
void widget_update_size(ei_widget_t* widget, ei_size_t* new_size)  
{  
    ...  
    widget_draw(widget);  
}
```

Polymorphisme des traitements

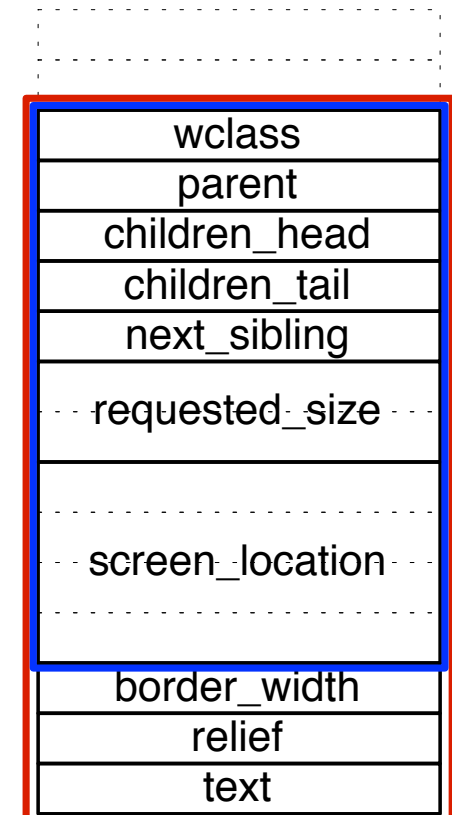
```
typedef void (*ei_widgetclass_drawfunc_t)
    (ei_widget_t* widget, ei_surface_t surface,
     ei_surface_t pick_surface, ei_rect_t* clipper);

typedef struct ei_widgetclass_t {
    ...
    ei_widgetclass_drawfunc_t    drawfunc;
    ...
} ei_widgetclass_t;

void button_draw (ei_widget_t* widget, ei_surface_t surface,
                 ei_surface_t pick_surface, ei_rect_t* clipper);

ei_widgetclass_t    button_class = { ..., button_draw, ...};
ei_button_widget_t button       = { &button_class, ... };

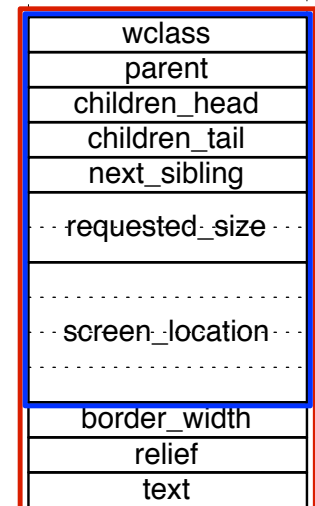
void widget_update_size(ei_widget_t* widget, ei_size_t* new_size)
{
    ...
    widget_draw(widget);
}
```



Ajout d'une classe d'interacteur dans la bibliothèque

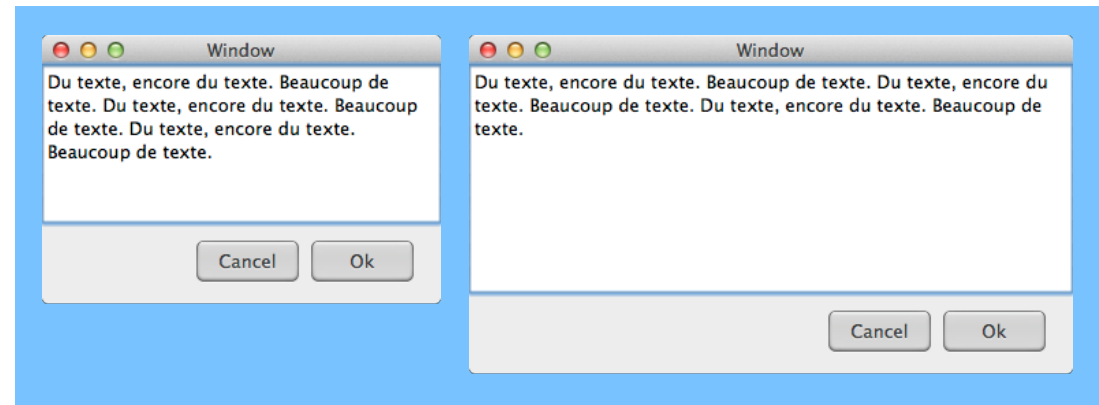
- Définition d'une structure qui étend `ei_widget_t` pour représenter les attributs spécifiques,
- définition de toutes les fonctions spécifiques,
- initialisation d'une instance de `ei_widgetclass_t`,
- enregistrement de la classe dans la bibliothèque par appel de `ei_widgetclass_register`.

```
typedef struct ei_widgetclass_t {  
    ei_widgetclass_name_t      name;  
    ei_widgetclass_allocfunc_t allocfunc;  
    ei_widgetclass_releasefunc_t releasefunc;  
    ei_widgetclass_drawfunc_t drawfunc;  
    ei_widgetclass_setdefaultfunc_t setdefaultfunc;  
    ei_widgetclass_geomnotifyfunc_t geomnotifyfunc;  
    ei_widgetclass_handlefunc_t handlefunc;  
    struct ei_widgetclass_t* next;  
} ei_widgetclass_t;  
  
void ei_widgetclass_register (ei_widgetclass_t* widgetclass);
```

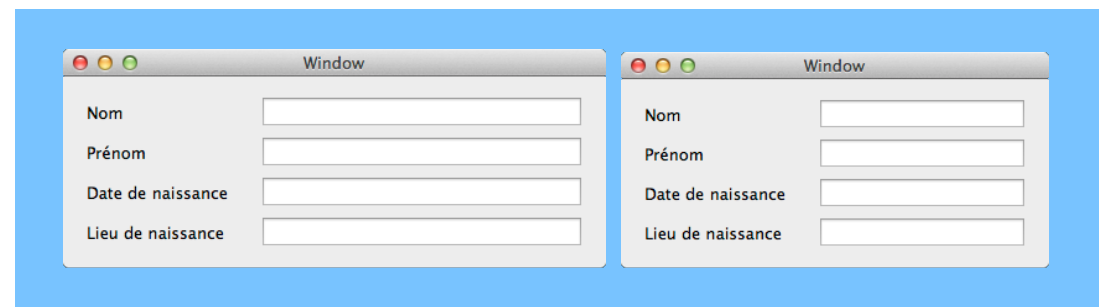


Le problème

Position relative au parent



Position et taille relative au parent et aux autres descendants.

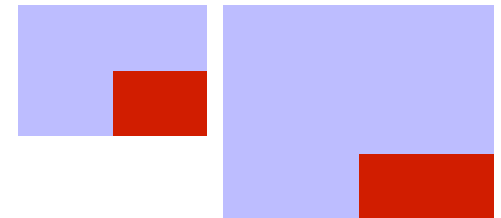


- ➔ Expression de contraintes pour la position et la taille des interacteurs, plutôt que des valeurs absolues.

Différentes stratégies

Placeur

Contraintes par rapport au parent uniquement.
“Place l’interacteur dans l’angle en bas à droite, avec une hauteur de 100 pixels et la moitié de la largeur du parent”.



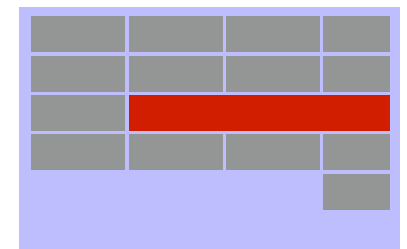
Packer

Contraintes par rapport au parent et aux descendants.
“Pack l’interacteur à droite des descendants déjà présents, en prenant toute la hauteur.”

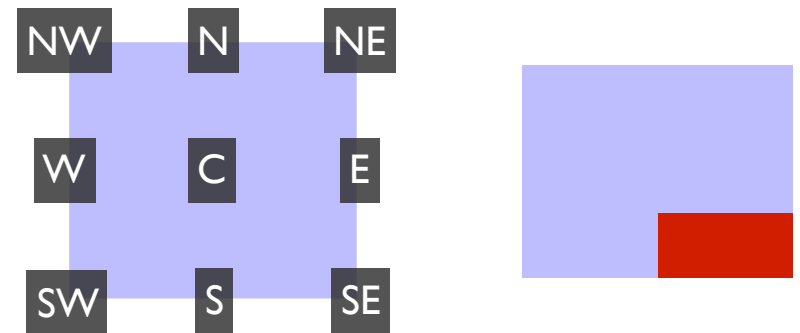


Gridder

Contraintes de grille.
“Grid l’interacteur en colonne 2, ligne 3, sur 3 colonnes.”



Interface de programmation du “placeur”

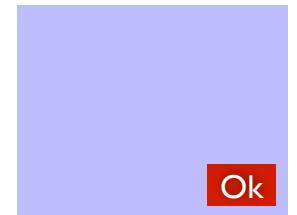


```
typedef enum {  
    ei_anc_none,  
    ei_anc_center, ei_anc_north, ei_anc_northeast, ei_anc_east, ei_anc_southeast,  
    ei_anc_south, ei_anc_southwest, ei_anc_west, ei_anc_northwest  
} ei_anchor_t;  
  
void ei_place (ei_widget_t* widget, ei_anchor_t* anchor,  
              int* x, int* y,  
              int* width, int* height,  
              float* rel_x, float* rel_y,  
              float* rel_width, float* rel_height);
```

Interface de programmation du “placeur”

```
ei_anchor_t    anchor = ei_anc_southeast;
int            x      = -10;
int            y      = -10;
float          rel_x  = 1.0;
float          rel_y  = 1.0;

ei_place(button, &anchor,  &x,      &y,      NULL, NULL,
           &rel_x,  &rel_y, NULL, NULL);
```



Interface de programmation du “placeur”

Mise en oeuvre des valeurs par défaut

```
ei_place(button, &anchor, &x, &y, NULL, NULL,  
        &rel_x, &rel_y, NULL, NULL);
```

Un paramètre NULL signifie “valeur par défaut” :

- Lors du premier appel, la valeur par défaut est donnée dans les spécifications.
- Quand la valeur a déjà été définie lors d'un appel précédent, elle est conservée.

Le principe de valeur par défaut s'applique à la configuration des widgets

```
ei_color_t background = { 0x00, 0x00, 0xff, 0x88 };  
void ei_button_configure(button, NULL, &background, NULL, ..., NULL);
```

Motivation

Programmation **séquentielle**

Séquences

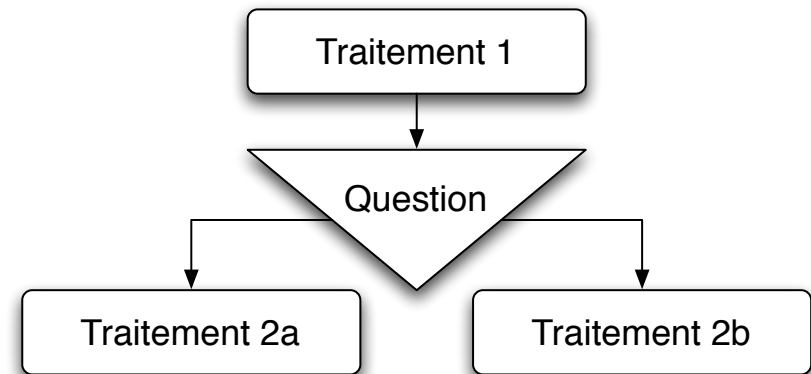
Répétitions

Branchements conditionnels

Le programme a le contrôle.

Le programme consulte les facteurs extérieurs à certains noeuds du graphe.

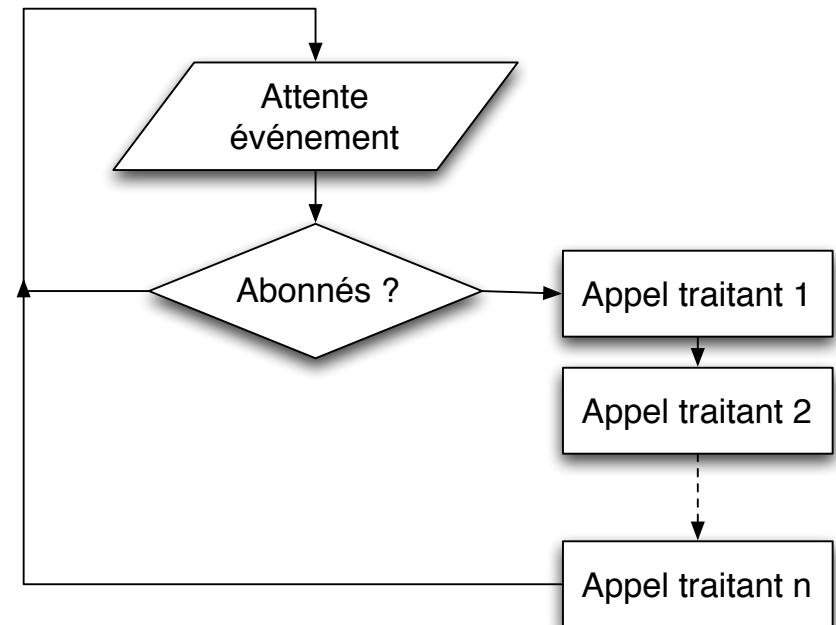
Cas des actions de l'utilisateur : à chaque étape, toute action est possible.



Motivation

Programmation **événementielle**

L'utilisateur a le contrôle.



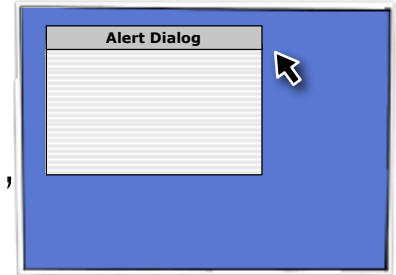
Le programmeur **abonne** des **traitants** à la réception d'événements.

Le programme principal se contente d'attendre un événement, puis d'appeler les traitants abonnés.

Exemple : glisser-déposer

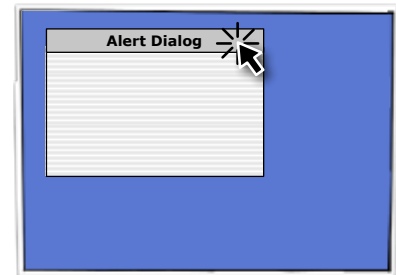
Initialisation

Définition de la fonction “handlefunc” de la classe “toplevel”



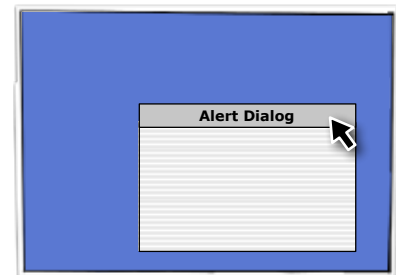
Réception de “ei_ev_mouse_buttondown”

Si le pointeur est sur la barre de titre de la fenêtre :
on s'intéresse maintenant aux événements “motion” et “buttonup”.



Réception de “ei_ev_mouse_move”

Déplacement de la fenêtre.



Réception de “ei_ev_mouse_buttonup”

On ne s'intéresse plus à “motion” et “buttonup”.

Interface de programmation

```
typedef enum { ei_ev_none, ei_ev_app,  
              ei_ev_keydown, ei_ev_keyup,  
              ei_ev_mouse_buttondown, ei_ev_mouse_buttonup, ei_ev_mouse_move,  
              ei_ev_last  
} ei_eventtype_t;  
  
typedef struct ei_event_t {  
    ei_eventtype_t    type;  
    union {  
        ei_key_event_t    key;  
        ei_mouse_event_t  mouse;  
        ei_app_event_t    application;  
    } param;  
} ei_event_t;  
  
typedef ei_bool_t (*ei_widgetclass_handlefunc_t)  
                (struct ei_widget_t*    widget,  
                 struct ei_event_t*     event);
```

Interface de programmation

Paramètres d'événement

```
typedef struct {
    SDLKey          key_sym;
    ei_modifier_mask_t modifier_mask;
} ei_key_event_t;

typedef struct {
    ei_point_t      where;
    int            button_number;
} ei_mouse_event_t;

typedef struct {
    void*          user_param;
} ei_app_event_t;
```

Interface de programmation

Pour le programmeur d'application.

```
typedef void (*ei_callback_t) (ei_widget_t* widget,  
                               struct ei_event_t* event,  
                               void* user_param);  
  
void ei_button_configure (ei_widget_t* widget,  
                        ...  
                        ei_callback_t* callback,  
                        void** user_param);
```

Programme principal

Le programmeur d'application :

- initialise l'interface graphique (création des widgets initiaux),
- enregistre ses traitants,
- lance la **boucle principale** (`ei_app_run()`).

Boucle principale

Le programmeur de la bibliothèque

- se met en attente d'un événement système (`hw_event_wait_next(&event)`),
- identifie le widget concerné,
- appelle les traitants concernés,
- met à jour l'écran,
- répète jusqu'à ce que le programmeur d'application appelle `ei_app_quit_request()`.

Identification du widget concerné

Cas des événements clavier (`ei_ev_keydown`, `ei_ev_keyup`).

La bibliothèque doit gérer l'interacteur qui a le *focus clavier*.

Ce n'est pas demandé dans le projet, mais peut être réalisé en extension.

Cas des événement souris

(`ei_ev_mouse_buttondown`, `ei_ev_mouse_buttonup`, `ei_ev_mouse_move`)

La bibliothèque doit pouvoir identifier l'interacteur *sous le pointeur* de la souris au moment de l'événement.

Ce service est appelé ***picking***.

Il y a différentes approches pour réaliser le picking.

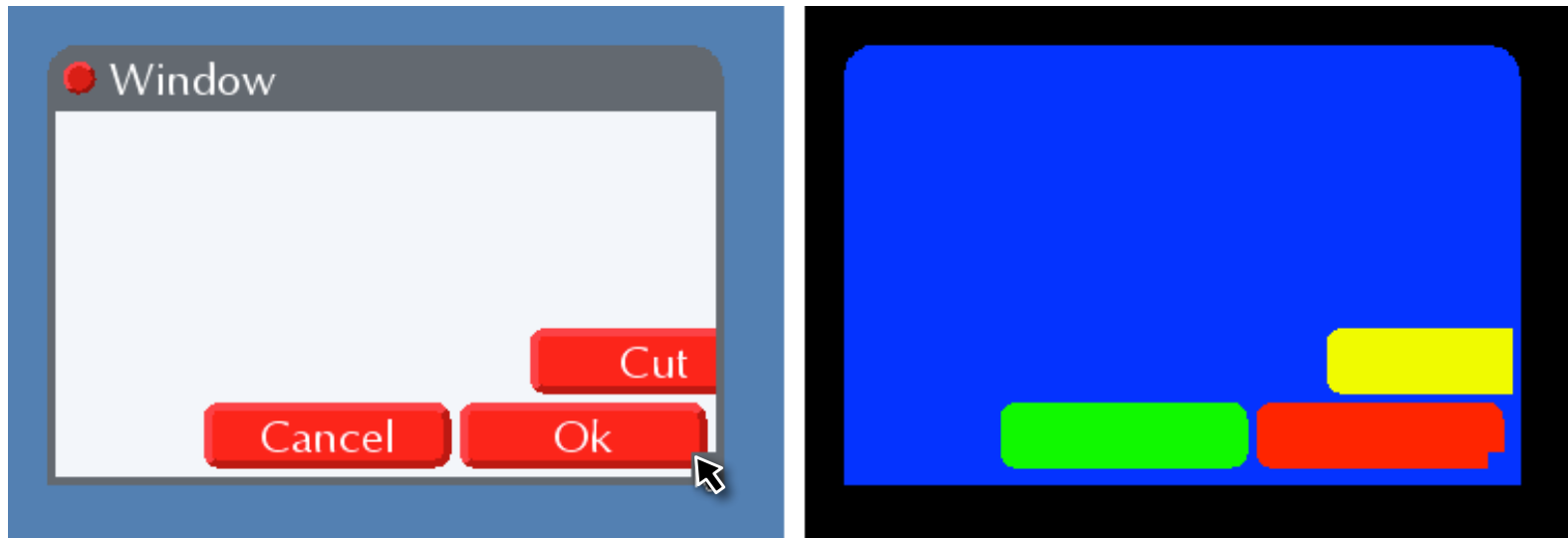
Vous réaliserez un ***offscreen de picking***.

Réalisation d'un offscreen de picking

Offscreen : surface de dessin qui n'est pas affichée.

Principe

Pour toute mise à jour de l'écran, l'offscreen de picking est mis à jour à l'identique, si ce n'est que la "couleur" utilisée est l'identifiant de l'interacteur.



Le picking consiste simplement à lire l'identifiant dans l'offscreen de picking à la position du curseur.

Réalisation d'un offscreen de picking

Attention à l'encodage des couleur, en particulier à la transparence.

```
typedef struct {
    unsigned char    red;
    unsigned char    green;
    unsigned char    blue;
    unsigned char    alpha;
} ei_color_t;

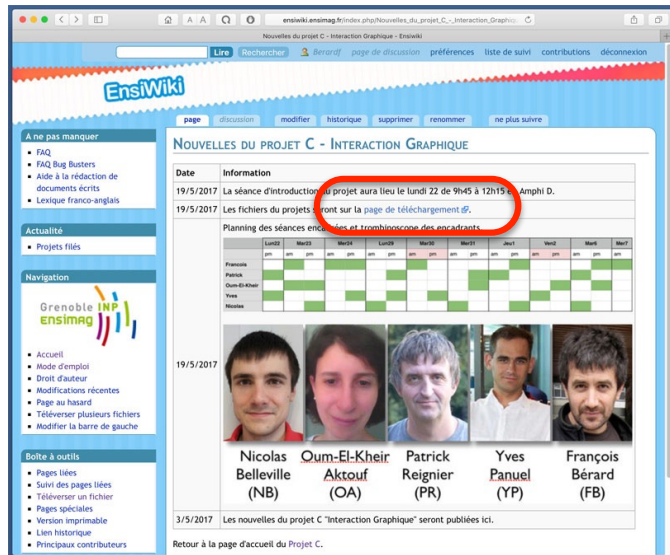
typedef struct ei_widget_t {
    ...
    uint32_t    pick_id;
    ei_color_t* pick_color;
    ...
}

uint32_t ei_map_rgba (ei_surface_t surface, const ei_color_t* color);

void ei_draw_polygone (ei_surface_t    surface,
                      const ei_linked_point_t* first_point,
                      const ei_color_t    color,
                      const ei_rect_t*    clipper);
```


Prochaine étape ?

89



projet aura lieu le lundi 22 de 9h45 à 12h15 en Amphi D.
ont sur la page de téléchargement [🔗](#).
trées et trombinoscope des encadrants.

```
[fberard@localhost ~/Downloads]$ l
total 468
-rw-r--r--. 1 fberard fberard 477479 May 22 07:28 projet_c_5620.tgz
[fberard@localhost ~/Downloads]$ tar -xzf projet_c_5620.tgz
[fberard@localhost ~/Downloads]$ cd projet_c_5620
[fberard@localhost projet_c_5620]$ make minimal
gcc -c -g -Wall -std=c99 -D_LINUX__ -m64 -I./include -I./tests -I/usr/include/SDL ./tests/minimal.c -o ./objs/minimal.o
gcc -o minimal -m64 -g ./objs/minimal.o _x11/libeibase64.a -lSDL -lSDL_ttf -lSDL_image -lm
[fberard@localhost projet_c_5620]$ ./minimal
```