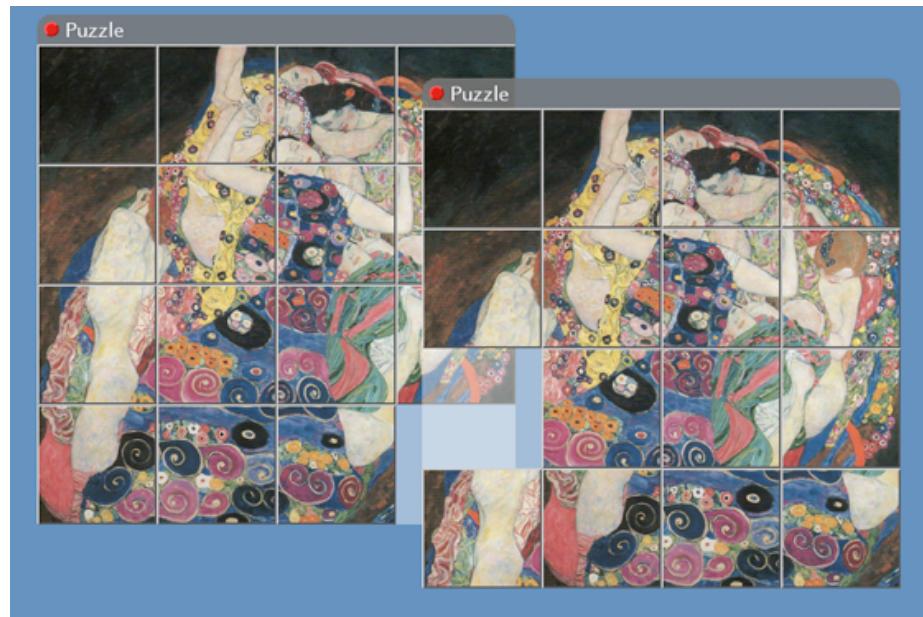
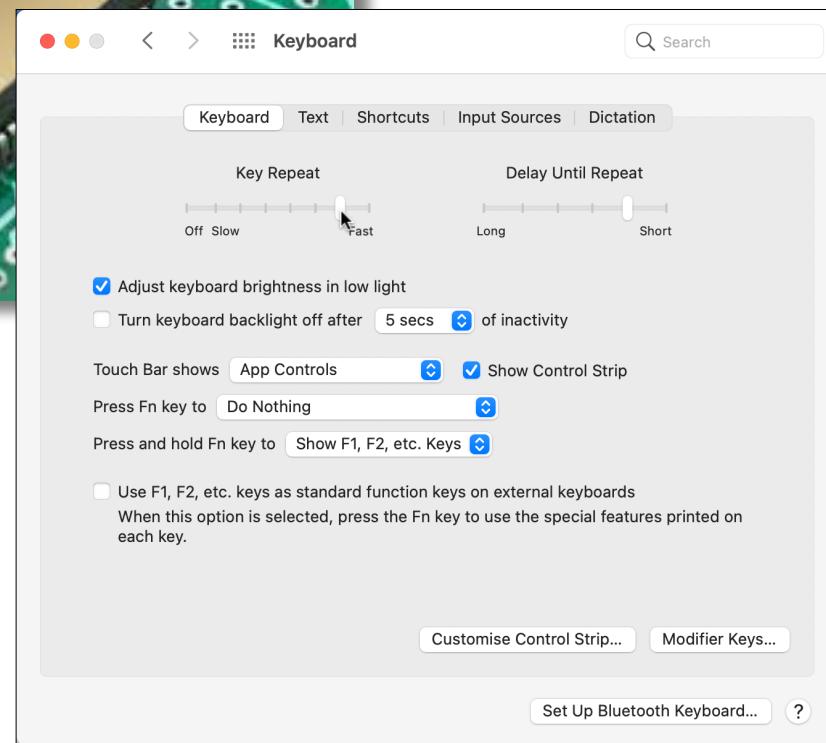
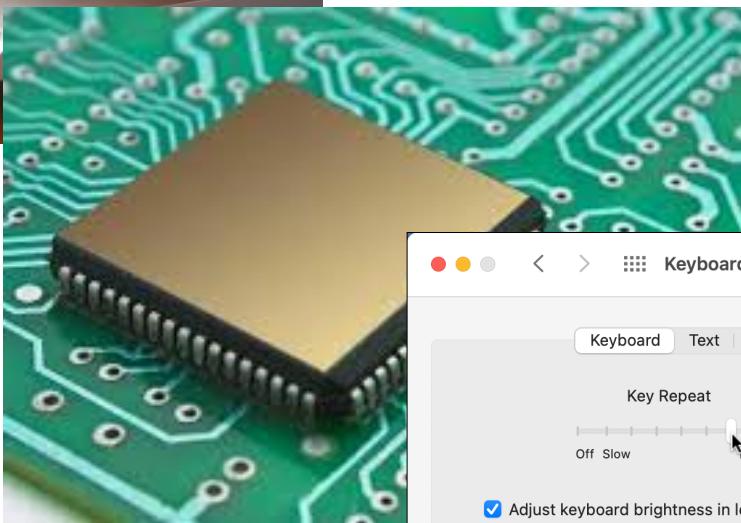


Projet C

Interaction Graphique



Amphi de présentation
F. Bérard 5 mai 2021





Le sujet

Les acteurs

Les **utilisateurs** de d'applications

Les **programmeurs** d'applications

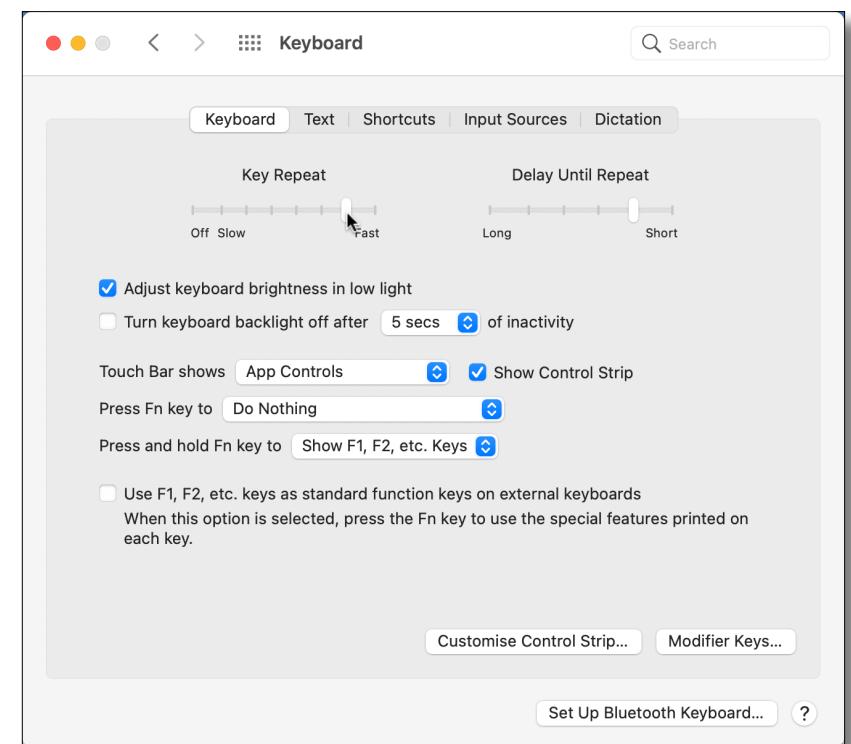
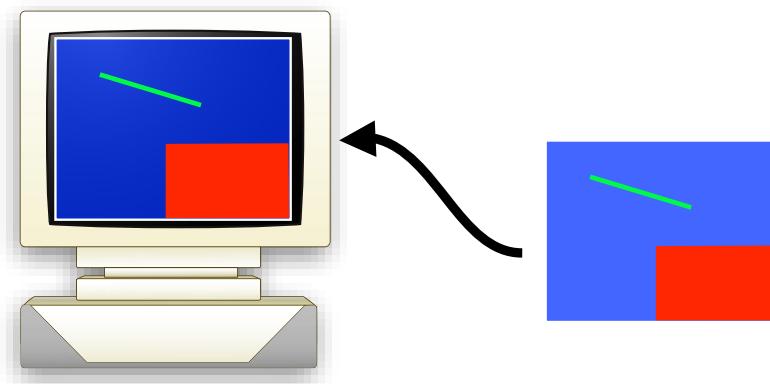
Les programmeurs de la bibliothèque (**vous**)

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

Le sujet

Réalisation d'un bibliothèque de programmation d'Interfaces Graphiques

En partant du buffer graphique



Le sujet

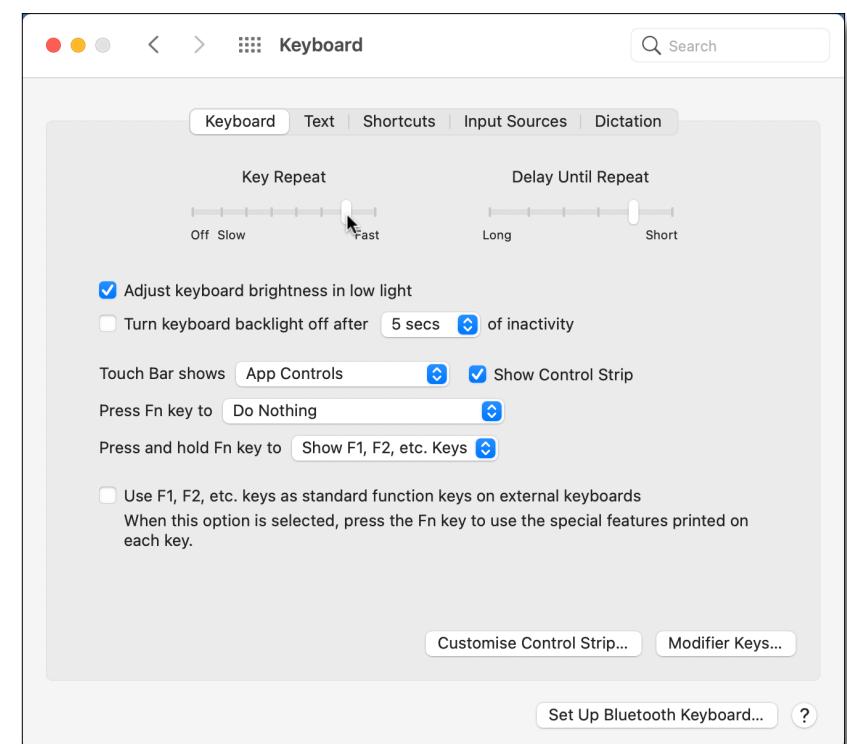
Réalisation d'un bibliothèque de programmation d'Interfaces Graphiques

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"

```

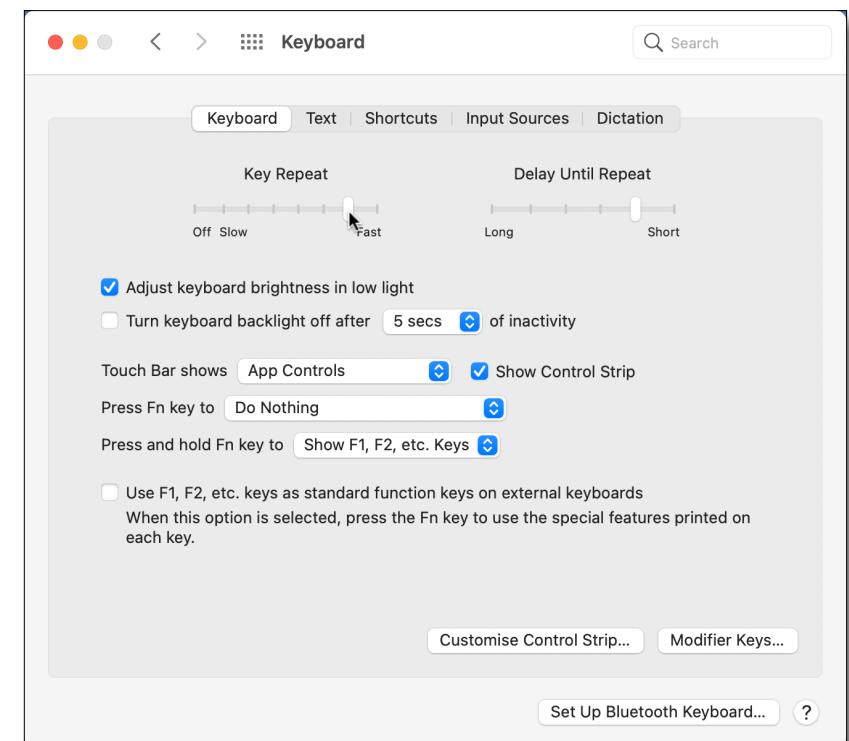


Le sujet

Réalisation d'un 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);
```



Le sujet

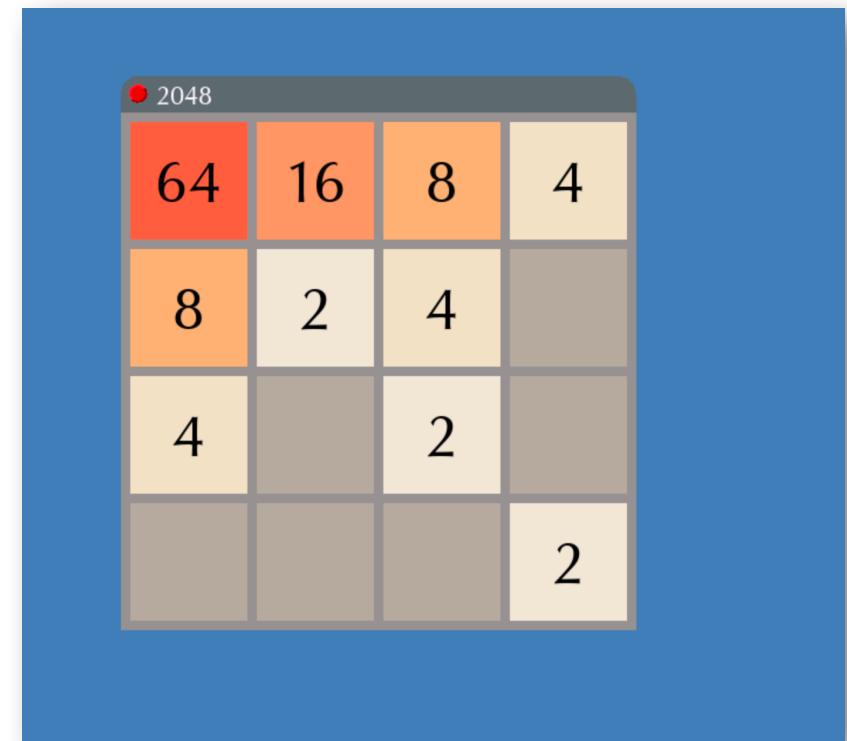
Réalisation d'un bibliothèque de programmation d'Interfaces Graphiques

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);
```



Services de la Bibliothèque

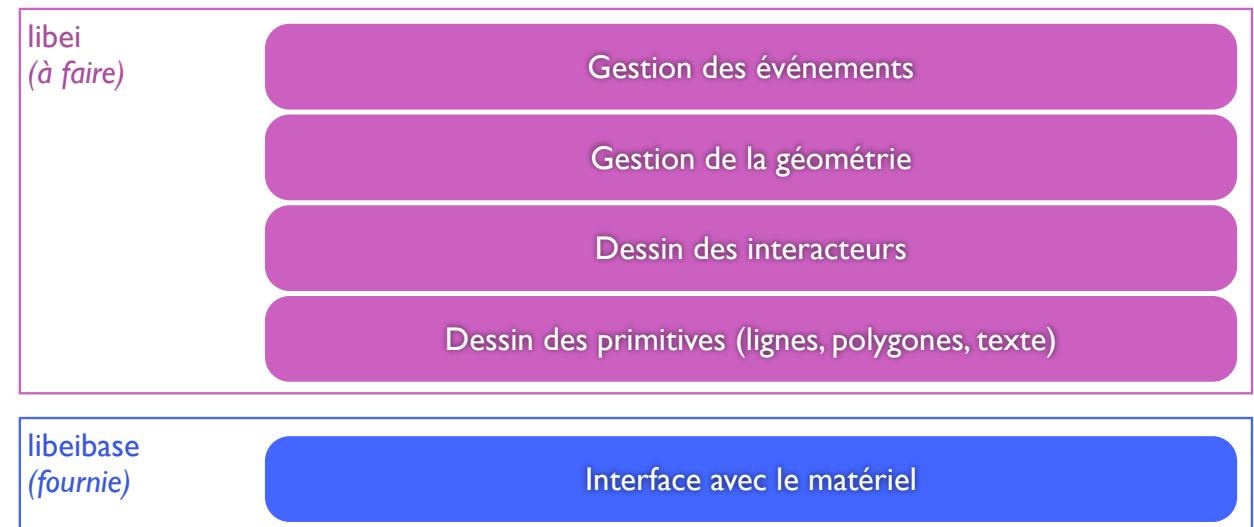
Survol

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

Création, configuration, et dessin des **interacteurs** ("widgets")

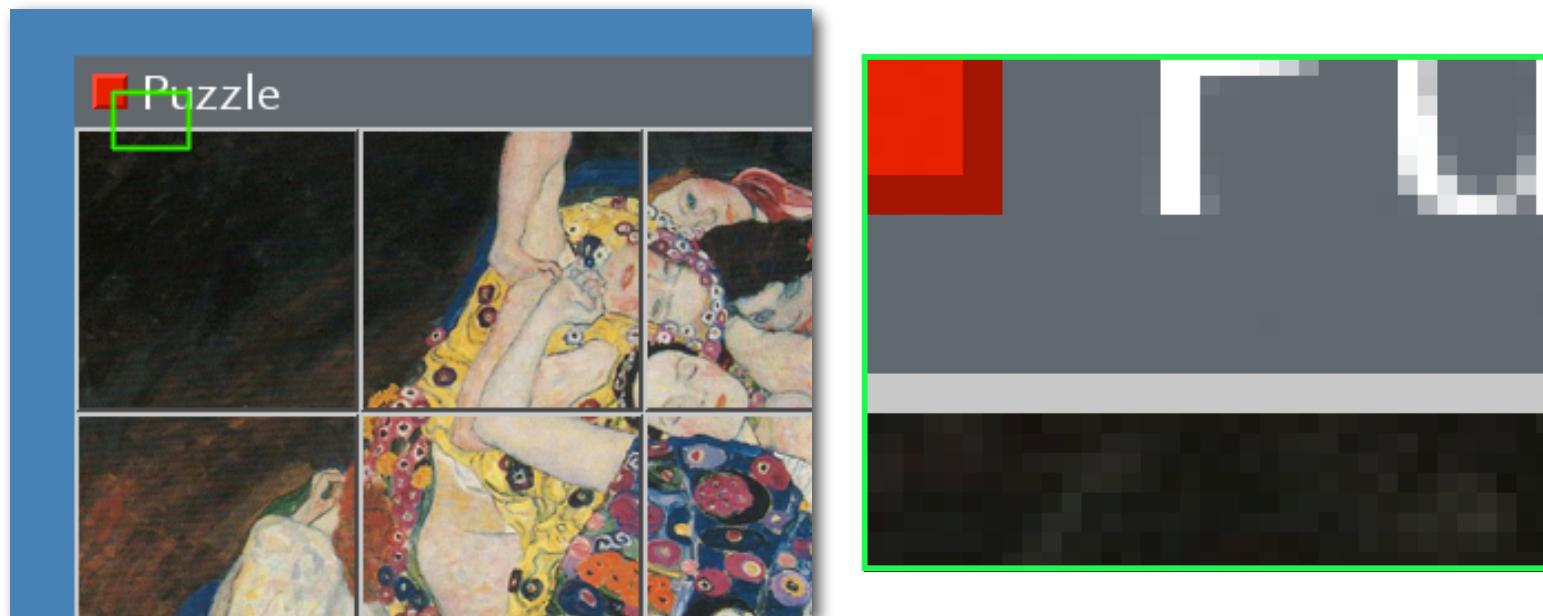
Placement à l'écran (position et taille) : gestion de la **géométrie**

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



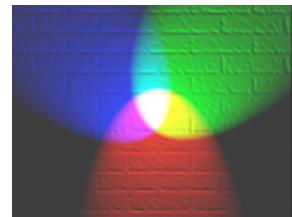
Génération d'Images Numériques

Représentation en mémoire



Représentation en mémoire

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



Représentation en mémoire

Différents formats de pixels

Image noir & blanc

1 pixel = 1 bit. 8 pixels dans un octet.

Image en niveaux de gris

256 niveaux de gris: 1 pixel = 1 octet

Image en couleur

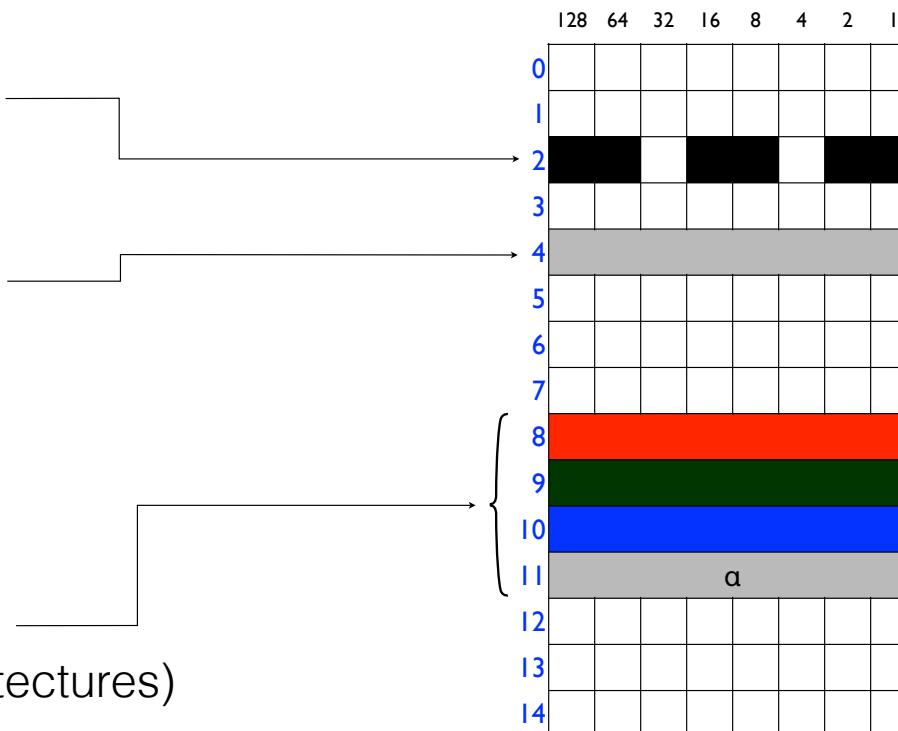
+ transparency (alpha)

256 niveaux par composante R,V,B,A

(> 16 000 000 couleurs):

1 pixel = 4 octets (32 bits)

(l'ordre des canaux varie suivant les architectures)



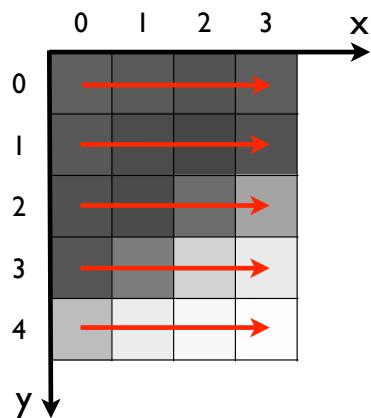
Représentation en mémoire

Ordre des pixels

de haut en bas de l'image,

de gauche à droite d'une ligne.

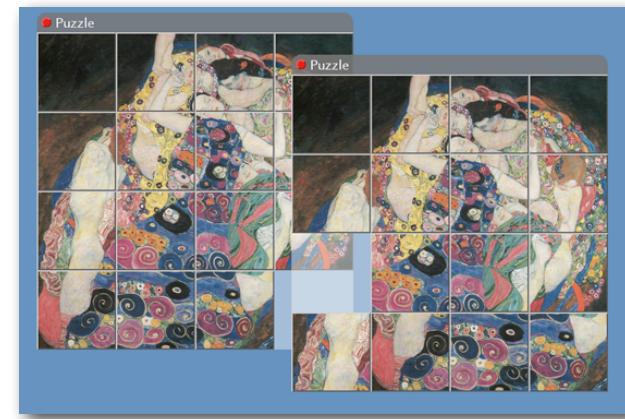
Autorise un “parcours scanline”



	128	64	32	16	8	4	2	1
0								
1								
2	x = 0, y = 0							
3	x = 1, y = 0							
4	x = 2, y = 0							
5	x = 3, y = 0							
6	x = 0, y = 1							
7	x = 1, y = 1							
8	x = 2, y = 1							
9	x = 3, y = 1							
10	x = 0, y = 2							
11	x = 1, y = 2							
12	x = 2, y = 2							
13	x = 3, y = 2							
14	x = 0, y = 3							
15	x = 1, y = 3							
16	x = 2, y = 3							
17	x = 3, y = 3							
18	x = 0, y = 4							
19	x = 1, y = 4							
20	x = 2, y = 4							
21	x = 3, y = 4							
22								
23								

```
for (i = nb_pixels; i != 0; i--)
    *pixel_ptr++ = 0xFFFFFFFF;
```

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 :

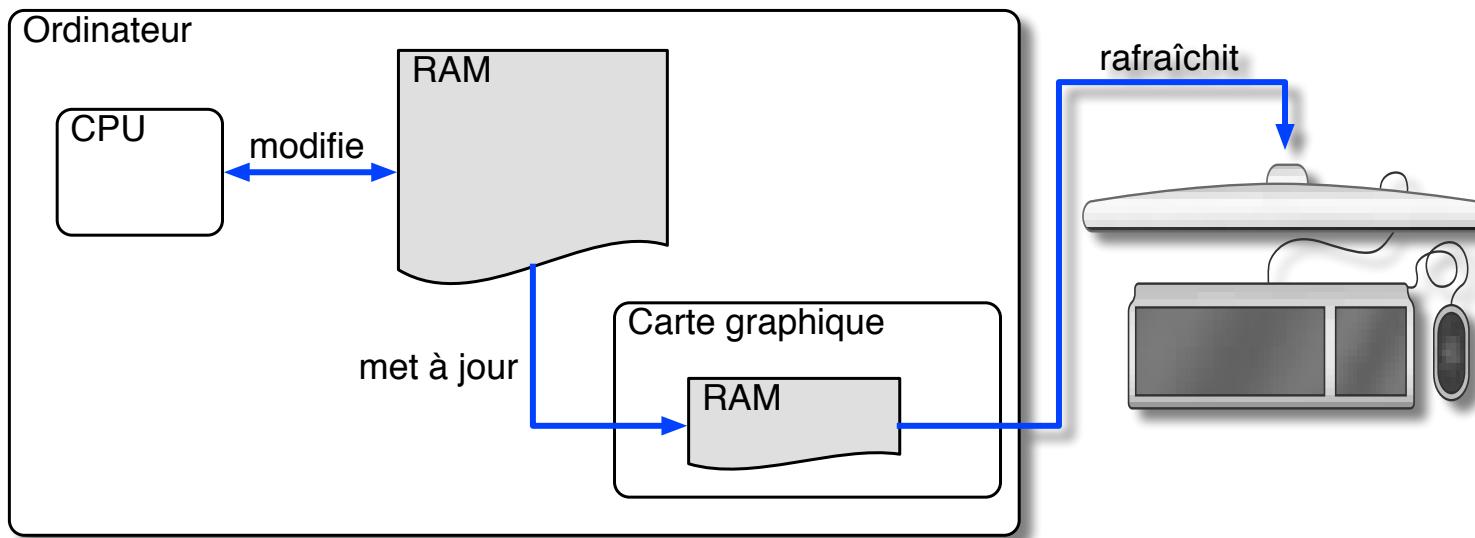
$$D_R = (D_R \cdot (255 - S_\alpha) + S_R \cdot S_\alpha) / 255$$

$$D_G = \dots$$

$$D_B = \dots$$

Architecture Matérielle

16



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

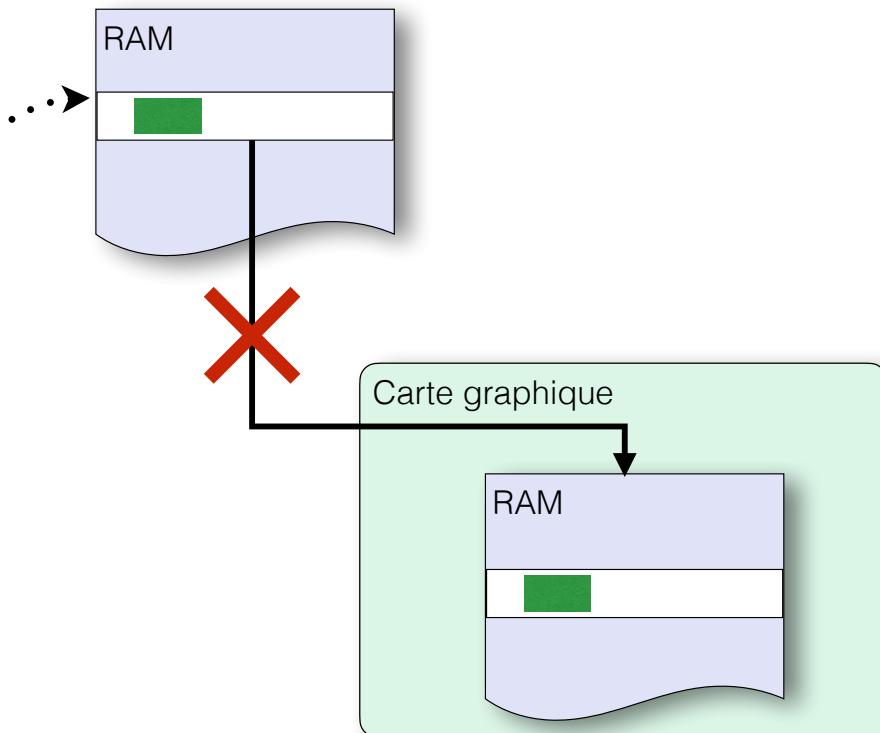
```
first_pixel      = hw_surface_get_buffer(my_surface);
*((uint32_t*)first_pixel)      = some_pixel_value;
ei_fill(my_surface, some_color, some_rectangle);
```

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);
```



Dessin des Primitives Graphique

Performance

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)

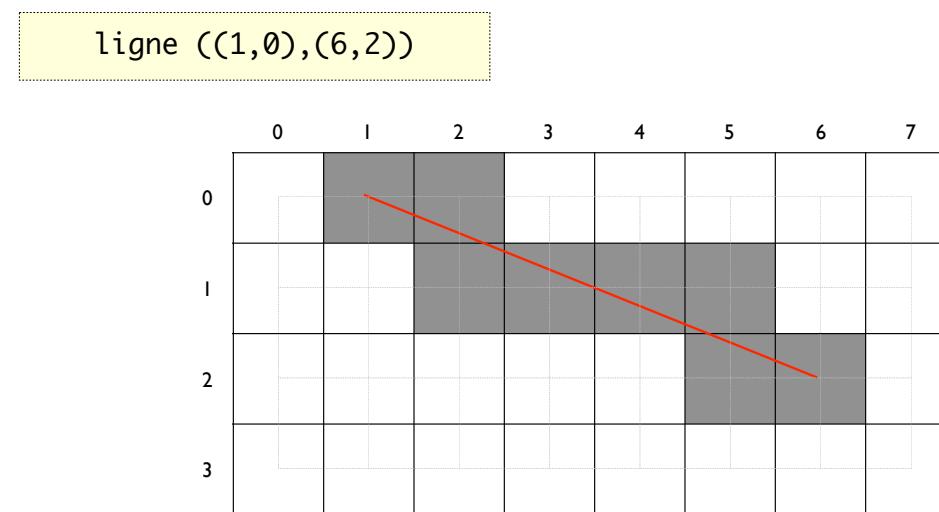
chaque opération sur un pixel × 2 million,

à faire 60 fois par secondes.

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

Dessin optimisé de lignes

Approche : tous les pixels qui touchent la ligne mathématique

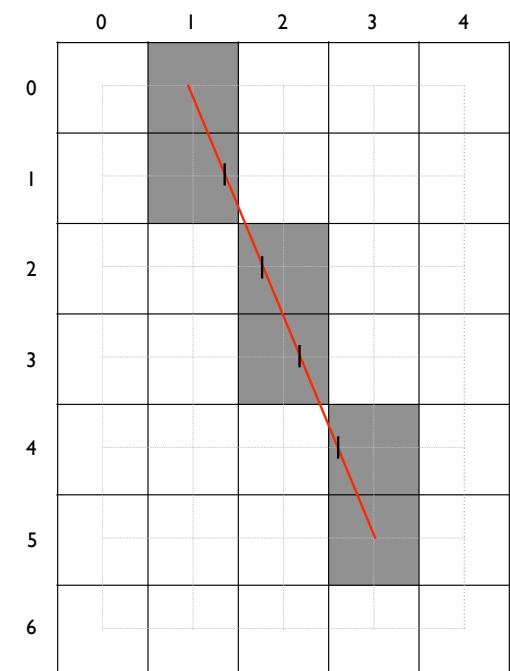
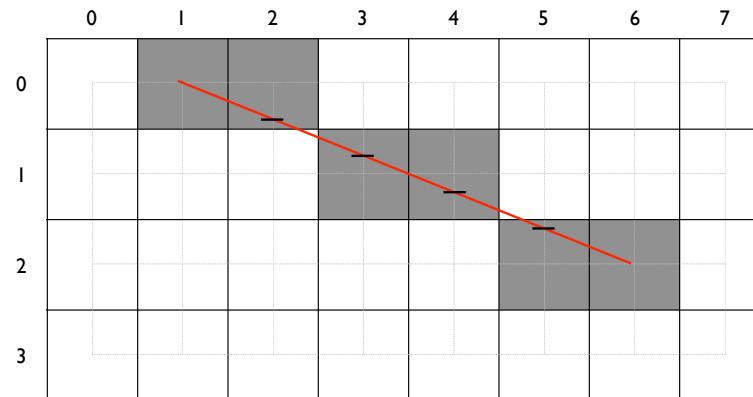


vibrations d'épaisseur visible, apparition d'angles

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))



Dessin optimisé de lignes

Arrondi de l'équation de droite

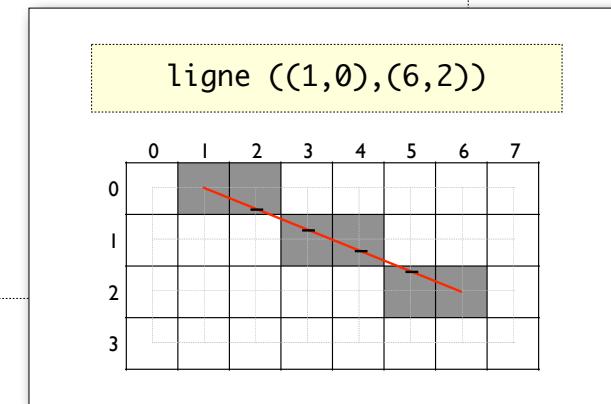
```

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;
    }
}

```



Dessin optimisé de lignes

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

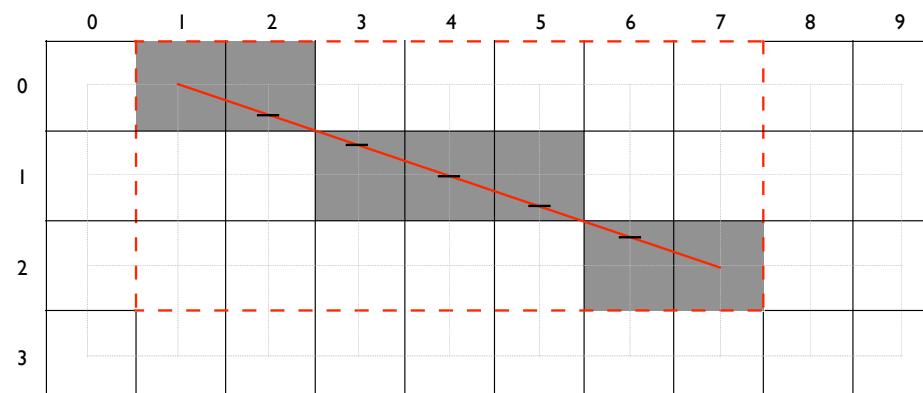
Dessin optimisé de lignes

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 $\varepsilon \leftarrow \varepsilon - 1$



Dessin optimisé de lignes

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 $E \leq E + dy$

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

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

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]

Dessin optimisé de lignes

Généralisation à toutes les orientations

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

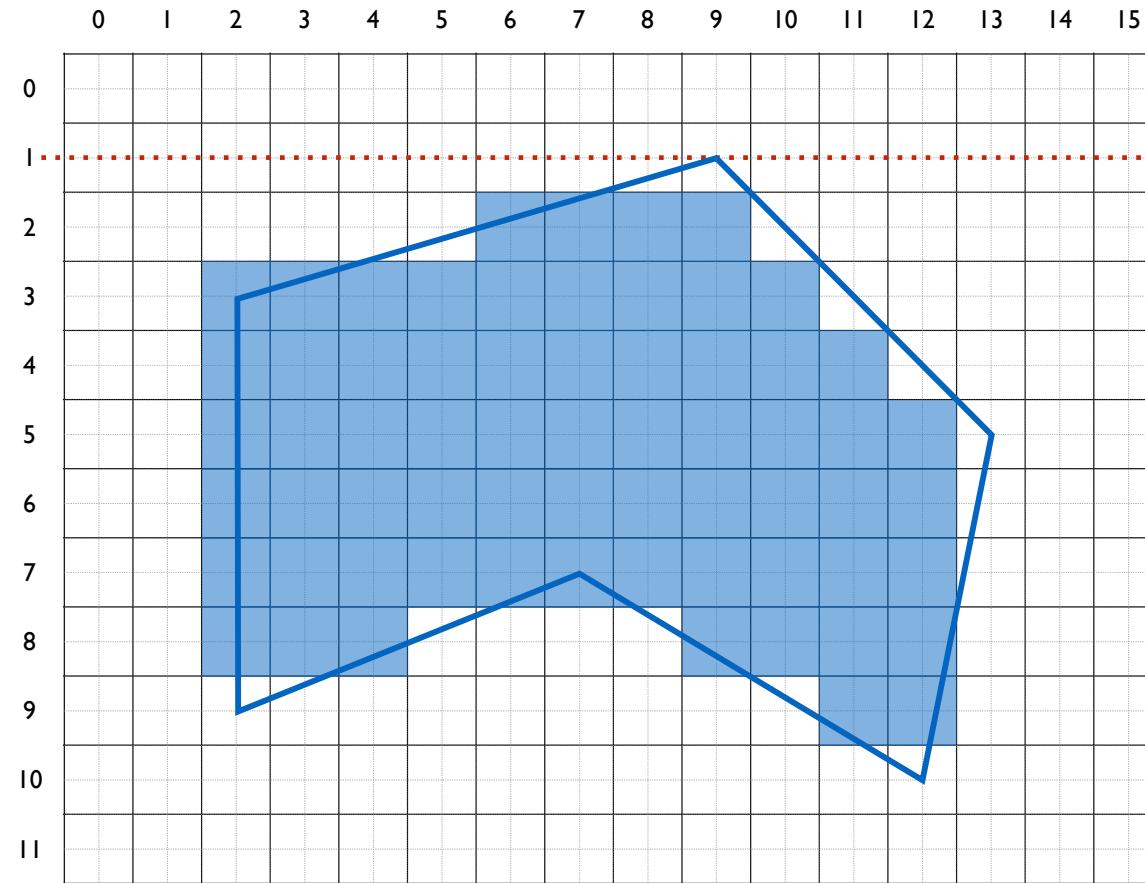
Pente négative, inverser les signes.

$x0 > x1$, 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))
```



Dessin optimisé de polygones (pleins)

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

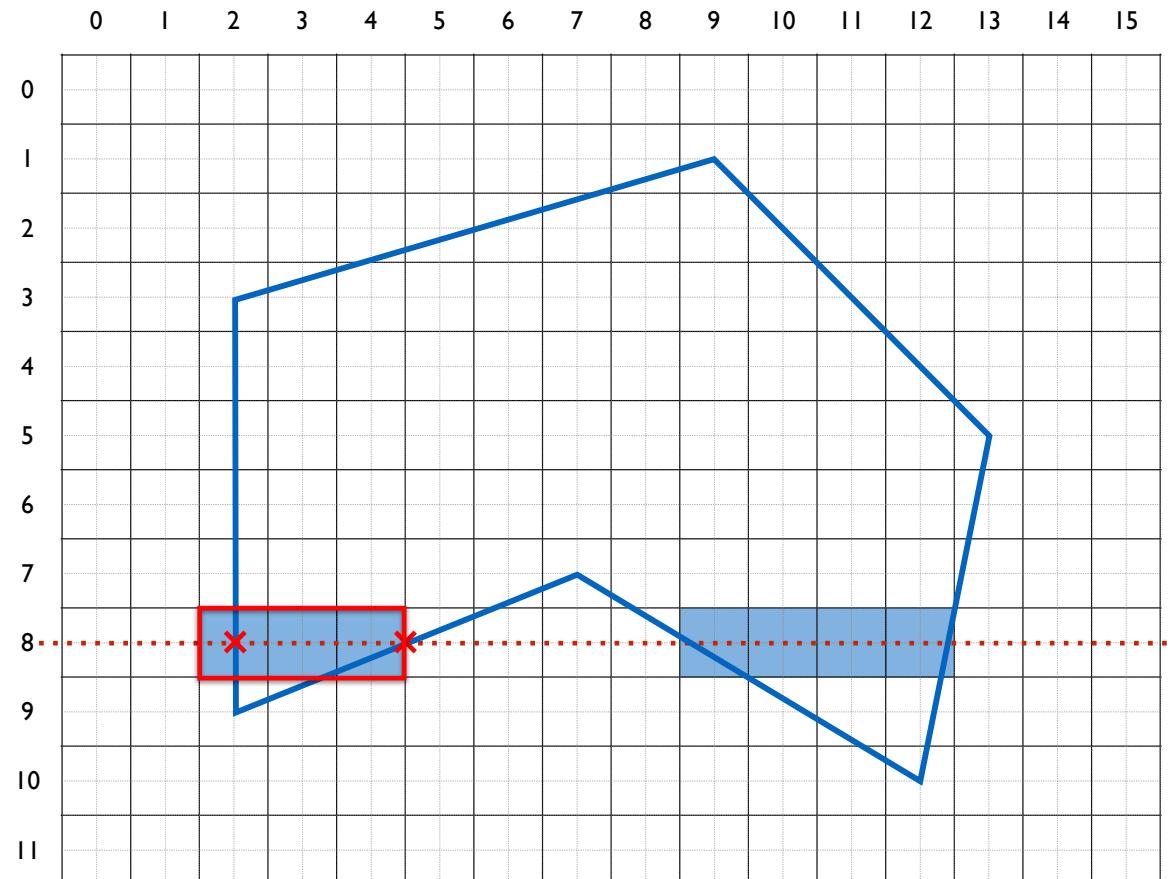
Stratégie

Cohérences spatiales :

Scanline : les pixels entre les intersections 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}

Calcul itératif optimisé des intersections (Bresenham)



Dessin optimisé de polygones (pleins)

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

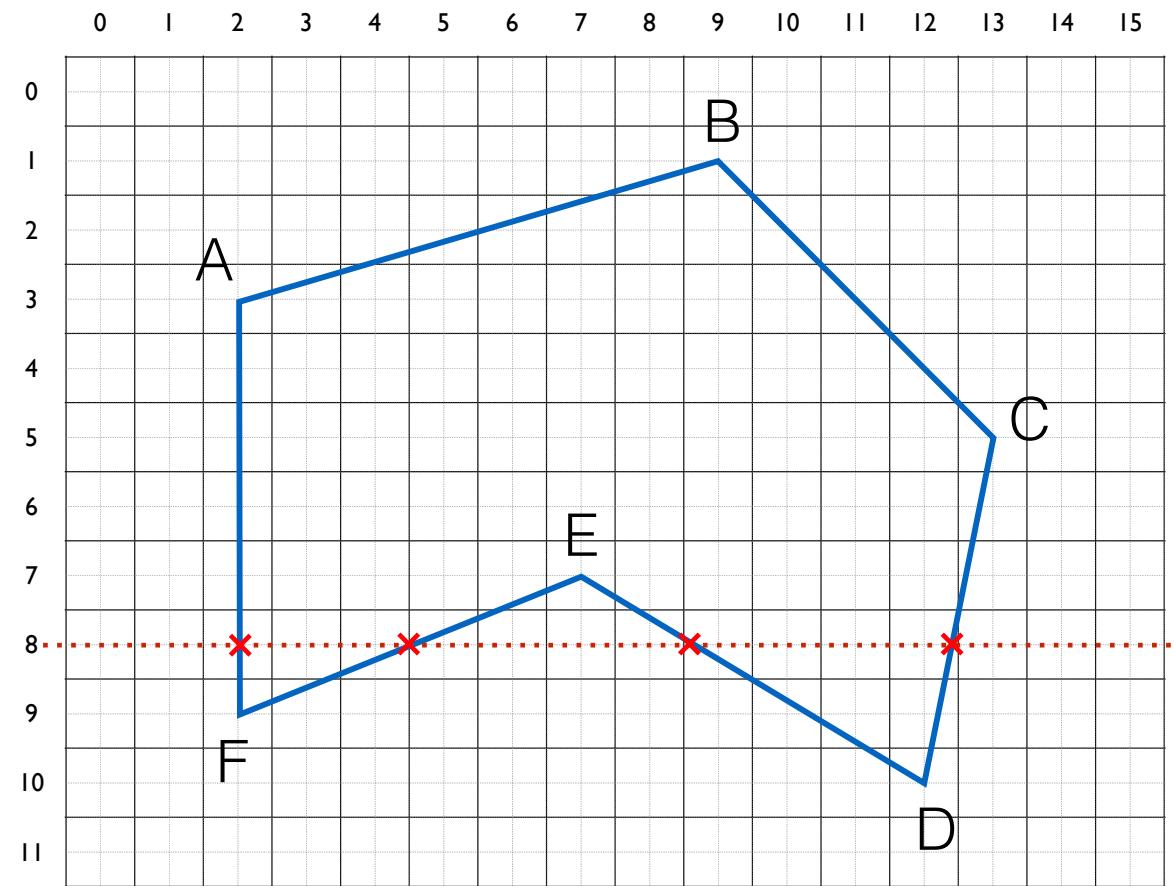
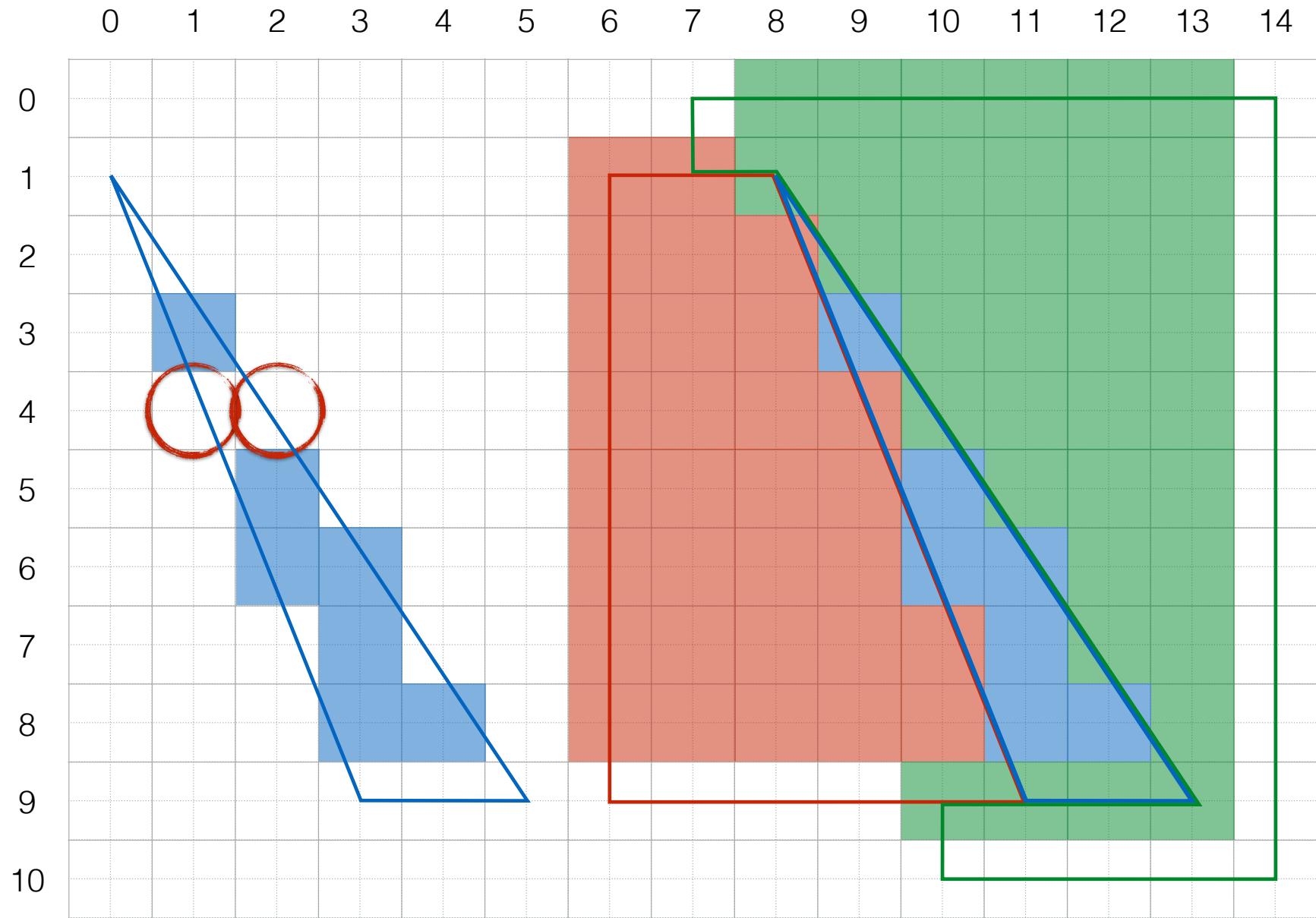


Table des Côtés Actifs (TCA)





Clipping

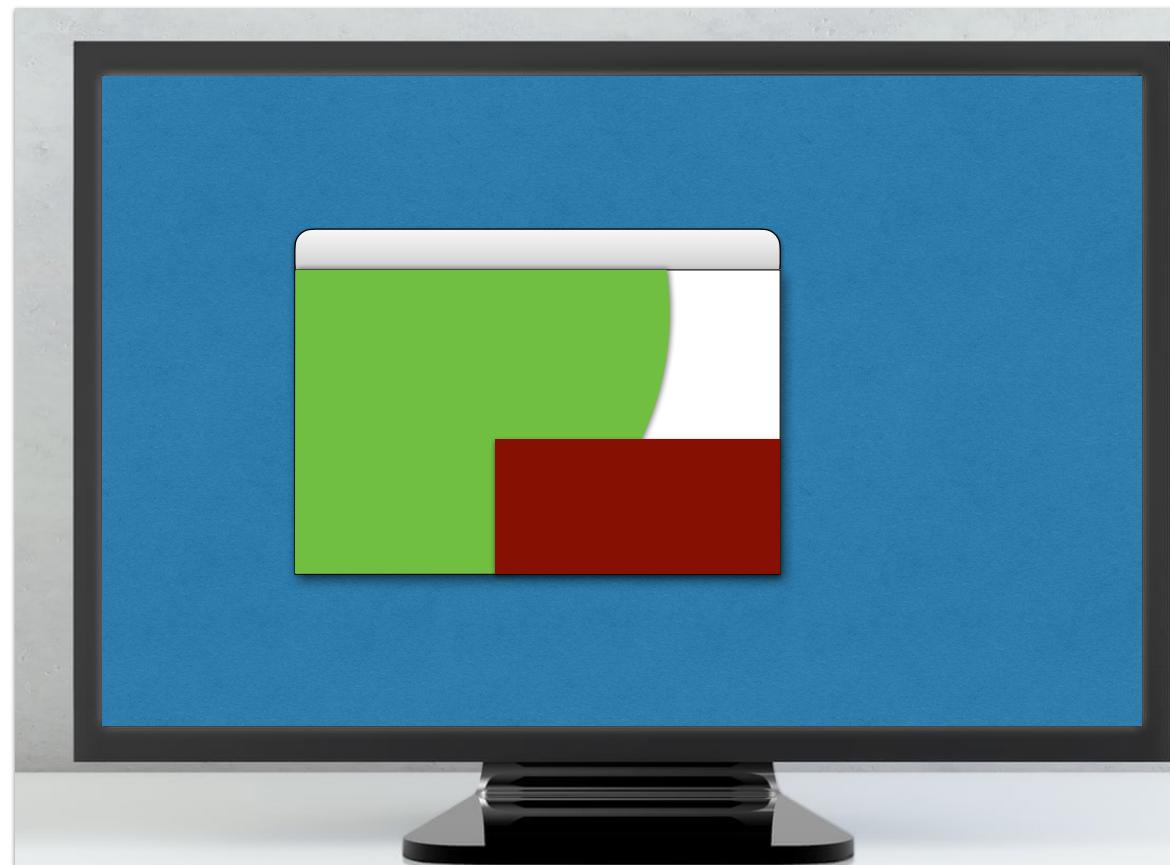
33

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)



Clipping

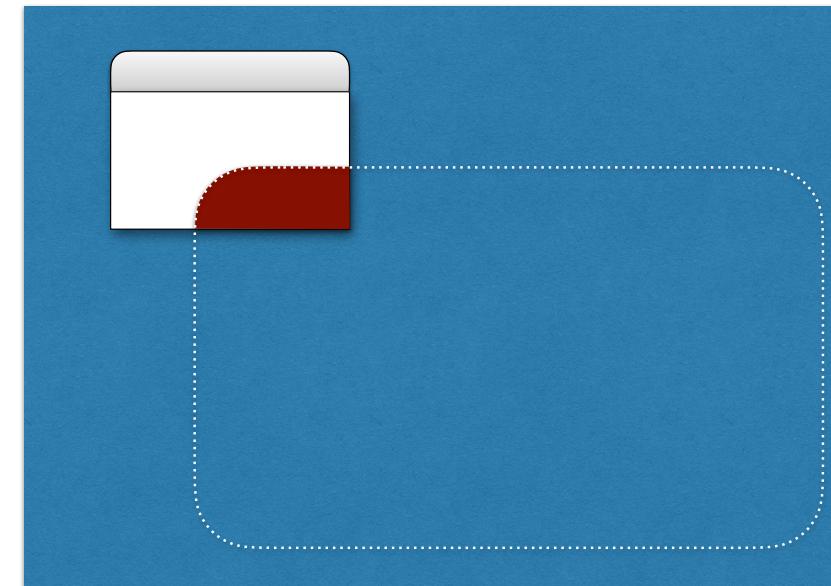
Test avant écriture

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

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)

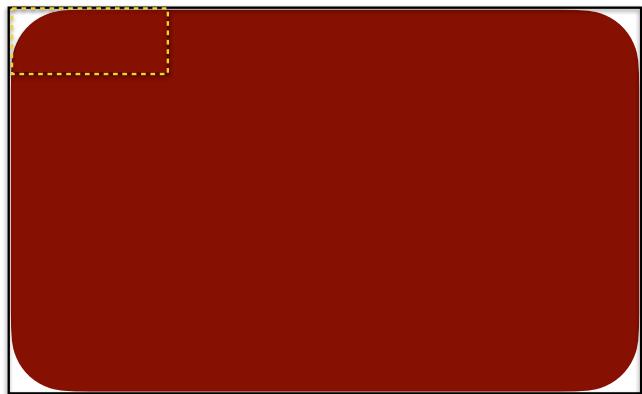


Clipping

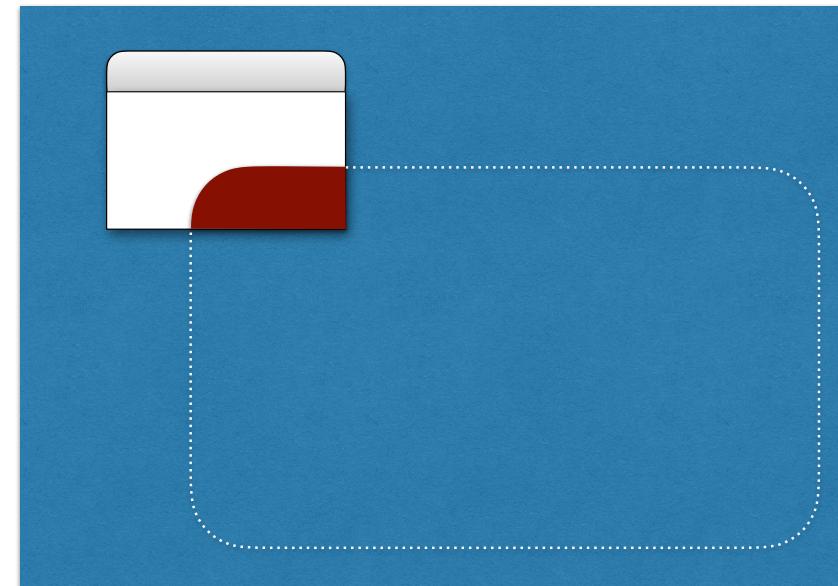
35

Offscreen

Dessin dans l'offscreen, puis copie



Élimine les 4 tests dans la boucle interne



Clipping

Analytique

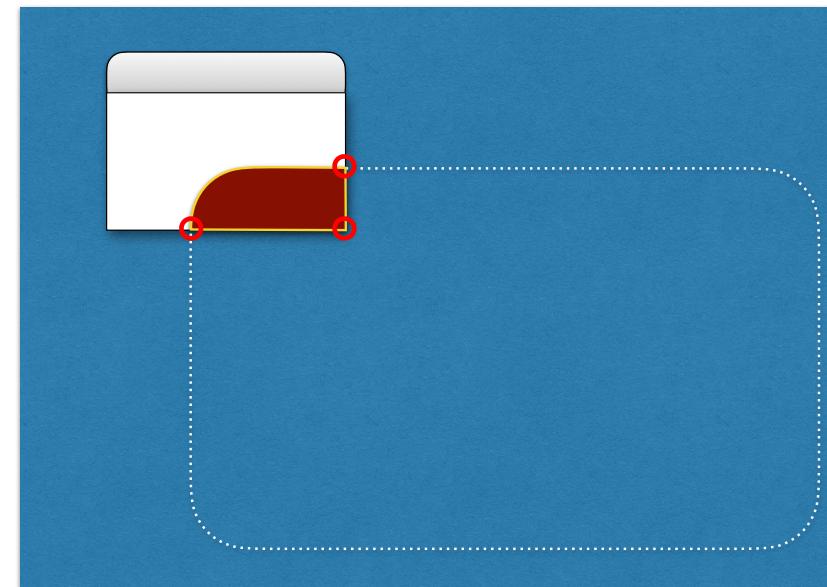
Calcul des intersections avec le clipper, dessin de la forme tronquée

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

Gestion des interacteurs

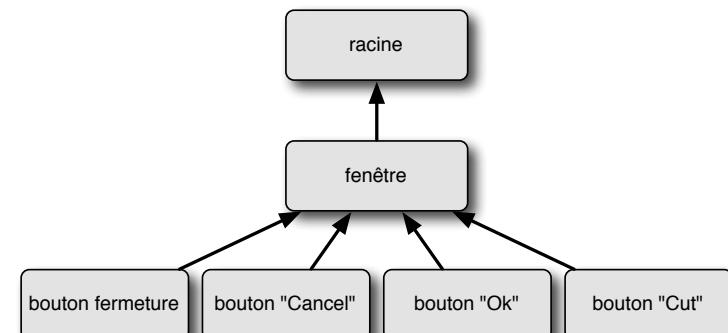
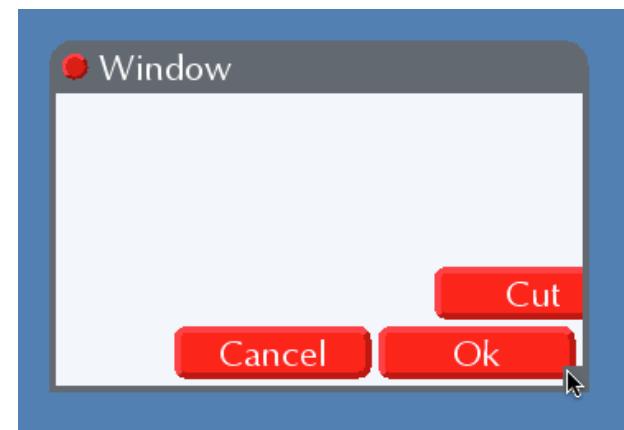
Organisation Hiérarchique

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

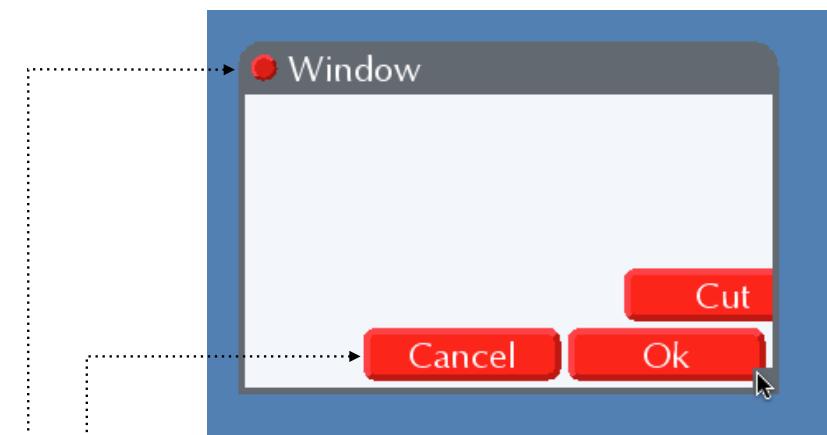
Caractéristiques communes :

Parent, liste de descendant, zone rectangulaire à l'écran, ...

Caractéristiques spécifiques d'une **classe d'interacteur** :

Fenêtre toplevel : bouton fermeture ?

Boutons : label, relief



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

Classe d'interacteurs

40

Polymorphisme des interacteurs

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

comme un **interacteur** pour les traitements communs à tout interacteur (définition du parent, etc.).

ou comme un **bouton** pour les traitements spécifiques aux boutons (dessin, etc.),

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

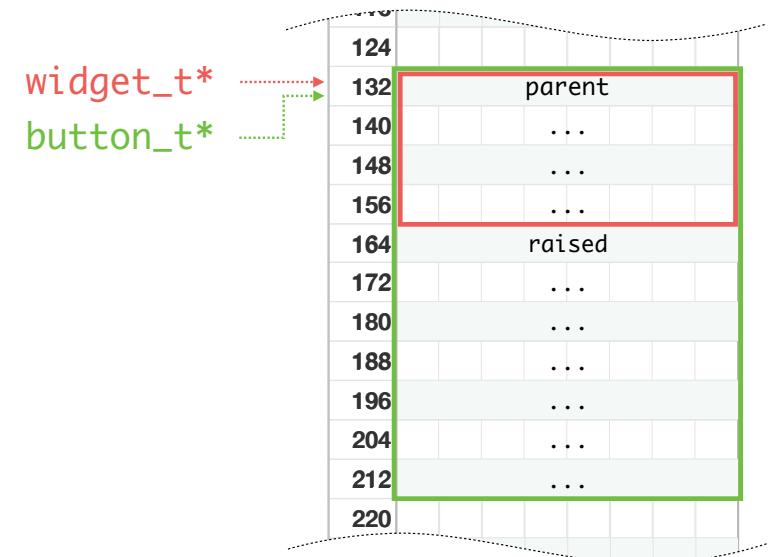
Classe d'interacteurs

41

Polymorphisme des données

```
button_t* create_button(widget_t* parent, bool_t raised)
{
    button_t* b = (button_t*)malloc(sizeof(button_t));
    b->raised = raised;
    init_widget((widget_t*)b, parent);
    return b;
}

void init_widget(widget_t* widget, widget_t* parent)
{
    widget->parent = parent;
}
```



```
typedef struct {
    struct widget_t* parent;
    ...
} widget_t;

typedef struct {
    widget_t
    bool_t raised;
    ...
} button_t;
```

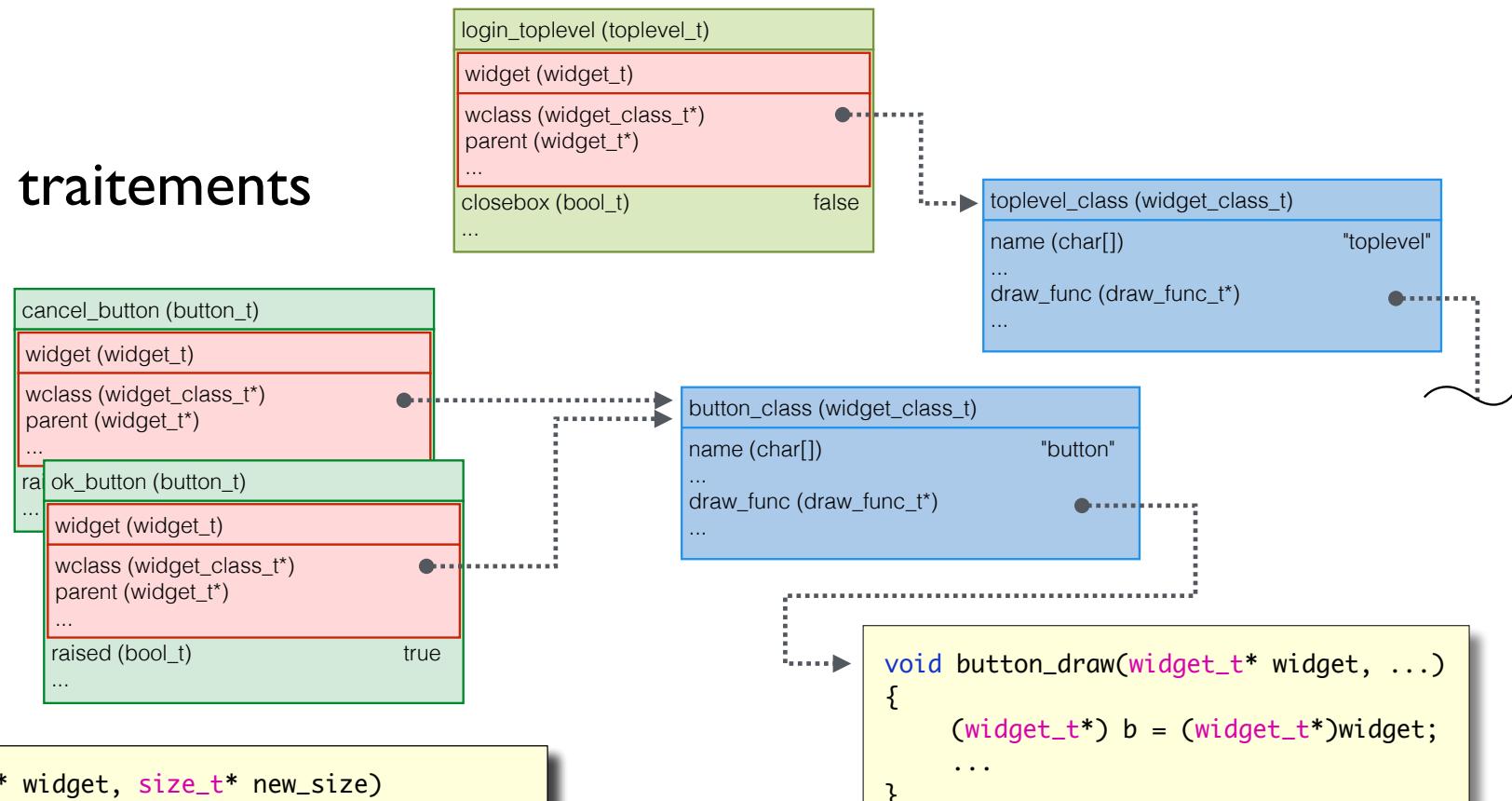
Classe d'interacteurs

Polymorphisme des traitements

```
void widget_update_size(widget_t* widget, size_t* new_size)
{
    widget->requested_size.size      = new_size;
    ...
    widget_draw(widget);           draw_button ? draw_toplevel ?
```

Programmation des classes d'interacteurs

Polymorphisme des traitements



```

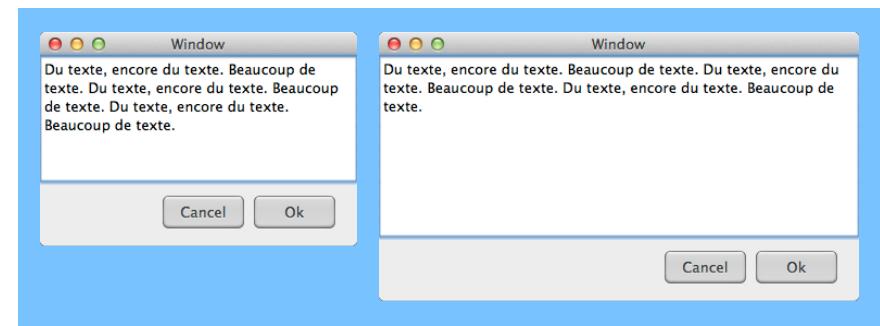
void widget_update_size(widget_t* widget, size_t* new_size)
{
    widget->requested_size.size = new_size;
    ...
    (*(widget->widget.wclass->draw_func))(widget, ...);
}

```

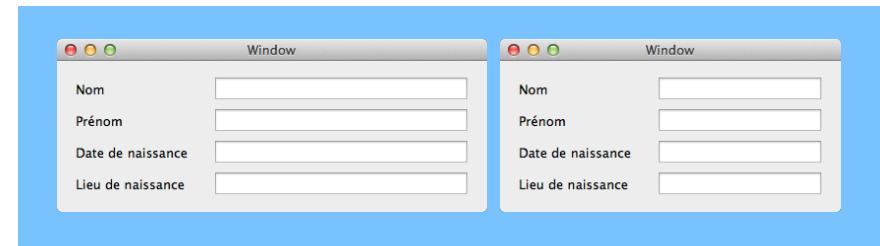
Gestion de la géométrie

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.

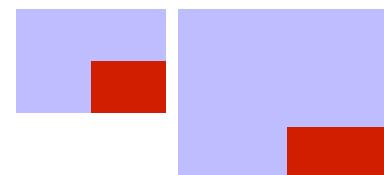
Gestion de la géométrie

Différentes stratégies

Placeur

Contraintes par rapport au parent uniquement.

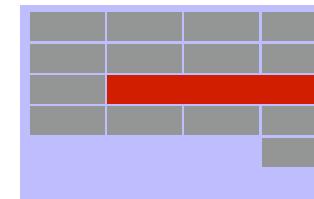
“Place l’interacteur dans l’angle en bas à droite, avec une hauteur de 100 pixels et la moitié de la largeur du parent”.



Gridder

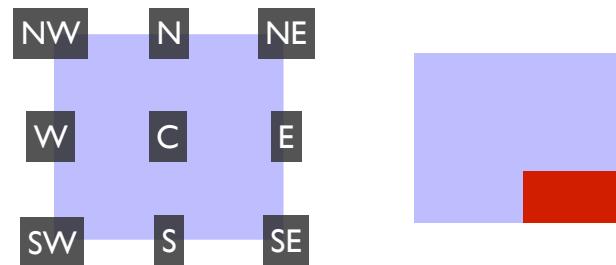
Contraintes de grille.

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



Gestion de la géométrie

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);
```

Gestion de la géométrie

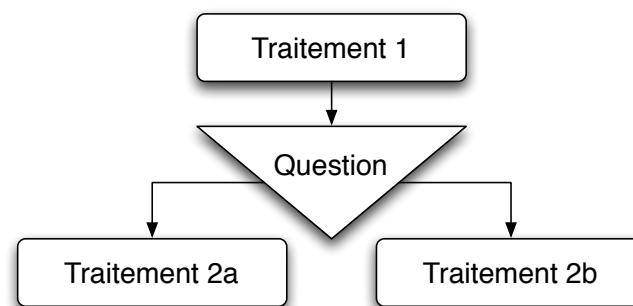
Interface de programmation du “placeur”

```
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);
```

Ok

Gestion des événements

Programmation séquentielle



Le programme a le contrôle.

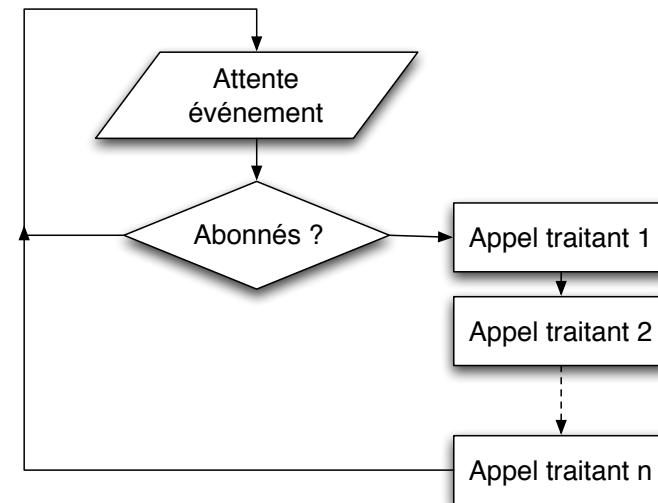
Le programme consulte les facteurs extérieurs à certains noeuds du graphe.

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

Gestion des événements

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.

Gestion des événements

Exemple : glisser-déposer

Initialisation

La fonction "traitant_top" est associée à la classe "toplevel"
 "traitant_top" s'intéresse aux événements "**button_down**"

Réception de "**button_down**"

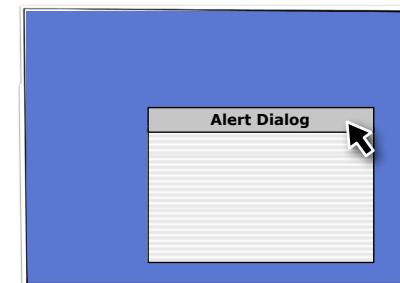
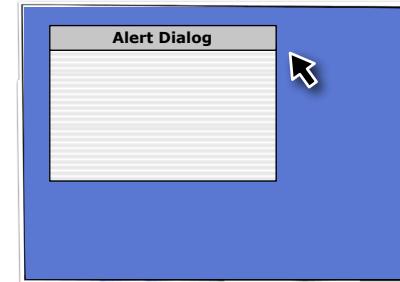
Si le pointeur est sur la barre de titre de la fenêtre :
 "traitant_top" s'intéresse à "**mouse_move**" et "**button_up**".

Réception de "**mouse_move**" (plusieurs fois)

Déplacement de la fenêtre.

Réception de "**button_up**"

"traitant_top" ne s'intéresse plus à "**mouse_move**" et "**button_up**".



Programmation de la gestion des événements

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()`.

Programmation de la gestion des événements

52

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**.

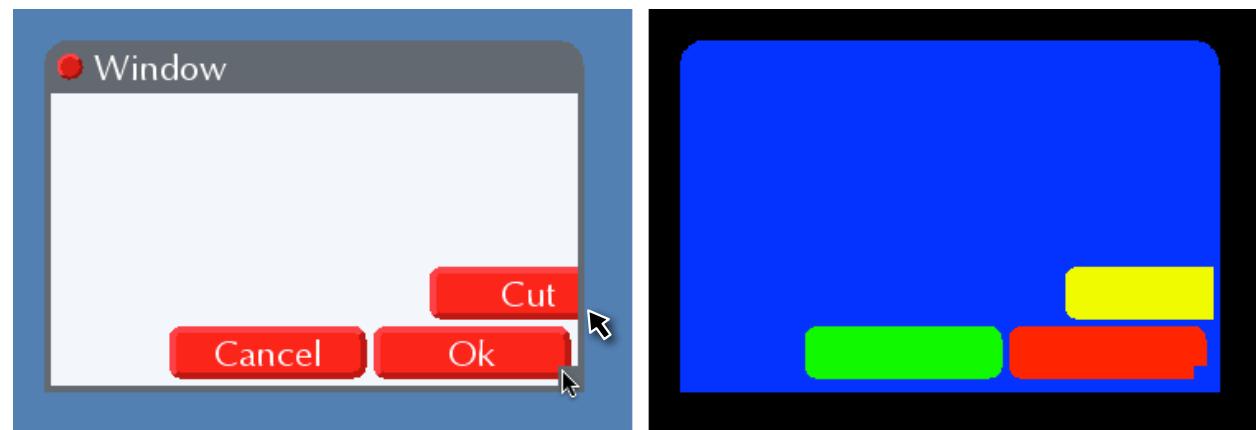
Programmation de la gestion des événements

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.

Organisation du projet

Où trouver l'info

Sur le site web du projet

<http://brouet.imag.fr/fberard/ProjetCLL/ProjetC>

Les enseignants

Salles E200, E201 en libre service avec support des enseignants

François Bérard, Gaëtan Lepage, Olivier Lim, Patrick Reignier, Sébastien Valat.

	mer5	jeu6		ven7		lun10		mar11		mer12		lun17		mar18		mer19		jeu20		ven21		mar25		mer26
	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am	pm	am
François																								
Patrick																								
Gaëtan																								
Sébastien																								
Olivier																								

Sur discord : appelez-nous avec @prof

Où trouver l'info

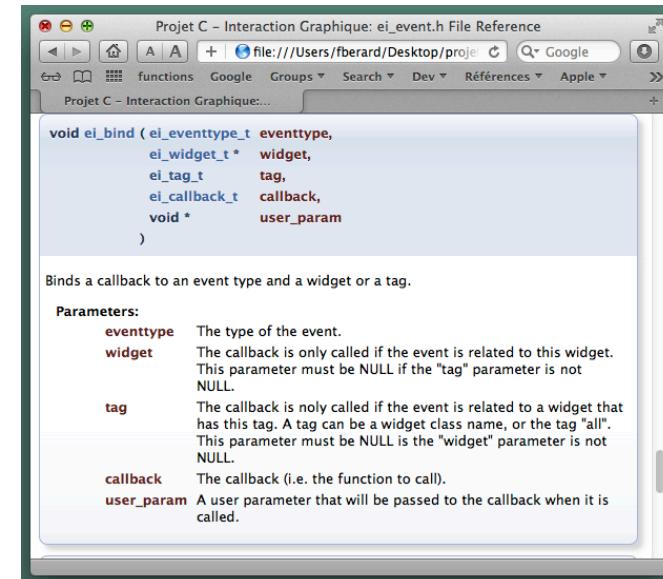
Le sujet en version pdf



Commentaires du code

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

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

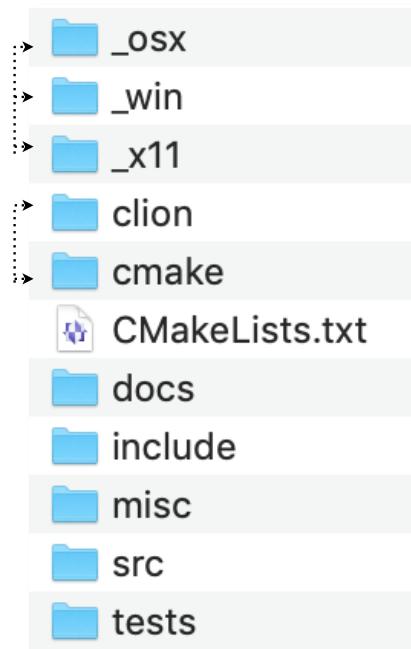


Fichiers fournis

57

L'archive des fichiers

- Code compilé fourni (libeibase)
- Où construire votre projet
- Définition du projet avec CMake
- Documentation doxygen
- Fichiers .h (API)
- Fichiers annexes (fonte, image)
- Où mettre vos fichier source
- Exemples de code d'applications

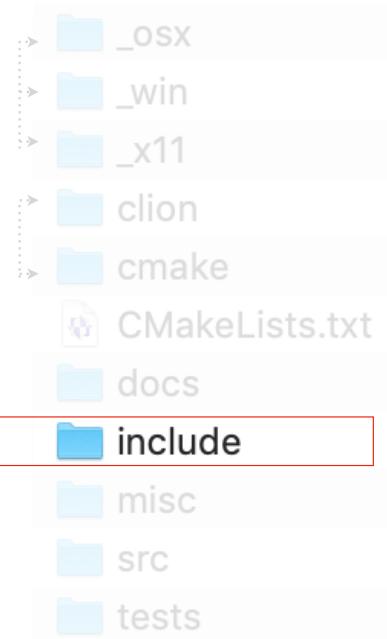


Fichiers fournis

58

L'archive des fichiers

- Code compilé fourni (libeibase)
- Où construire votre projet
- Définition du projet avec CMake
- Documentation doxygen
- Fichiers .h (API)
 - Fichiers annexes (fonte, image)
 - Où mettre vos fichier source
 - Exemples de code d'applications



**Interdiction absolue de modifier les fichiers .h fournis !
(vous pouvez ajouter vos propres fichiers .h)**

Travail sur les machines personnelles

Développement sous Linux, Mac OS, Windows

Le projet nécessite la bibliothèque SDL et quelques dépendances.

<http://brouet.imag.fr/fberard/ProjetC/InstallSDL>

Vous pouvez compiler en utilisant CMake en ligne de commande

```
cd cmake  
cmake ..  
make minimal  
cd ..  
./cmake/minimal
```

Mais il est recommandé d'utiliser l'environnement de développement CLion

<http://brouet.imag.fr/fberard/ProjetC/CLion>

ProjetC_IG – minimal.c

project_c_v6.0 copy > tests > minimal.c

minimal.c x hw_interface.h

Project Files

Structure

~/Fanf/enseignement/2021_ENSI1_ProjetC/projet_c_v6.0

- _macos
- _win
- x11
- clion
- cmake
- docs
- include
 - ei_application.h
 - ei_draw.h
 - ei_event.h
 - ei_placer.h
 - ei_types.h
 - ei_utils.h
 - ei_widget.h
 - ei_widgetclass.h
 - hw_interface.h
- misc
- src
- tests
 - button.c
 - frame.c
 - freq_counter.c
 - freq_counter.h
 - hello_world.c
 - lines.c
 - minimal.c
 - puzzle.c
 - two048.c
- CMakeLists.txt
- Scratches

minimal.c

```
#include <stdlib.h>
#include "ei_types.h"
#include "ei_event.h"
#include "ei_utils.h"
#include "hw_interface.h"

int main(int argc, char* argv[])
{
    ei_surface_t           main_window = NULL;    main_window: NULL
    ei_size_t               main_window_size = ei_size(640, 480);    main_window_size: ei_size_t
    ei_event_t              event;    event: ei_event_t
    uint32_t                white = 0xffffffff;    white: 4294967295 [0xffffffff]
    uint32_t*               pixel_ptr;    pixel_ptr: 0x00007ffee3d50958
    int                     i;    i: 32766 [0x7ffe]

    // Init acces to hardware.
    hw_init();

    // Create the main window.
    main_window = hw_create_window(main_window_size, EI_FALSE);    main_window_size: ei_size_t    main_window: NULL

    // Lock the surface for drawing, fill in white, unlock, update screen.
    hw_surface_lock(main_window);

    pixel_ptr = (uint32_t*)hw_surface_get_buffer(main_window);
    for (i = 0; i < (main_window_size.width * main_window_size.height); i++)
        *pixel_ptr++ = white;

    hw_surface_unlock(main_window);
    hw_surface_update_rects(main_window, NULL);

    // Wait for a key press.
    event.type = ei_ev_none;
    while (event.type != ei_ev_keydown)
        hw_event_wait(next(&event));
}
```

Debug: minimal

Debugger Console

Frames

Thread-1<-com.apple.main-thread>

main minimal.c:21

start 0x00007fff2036c621

Variables LLDB Memory View

- argc = {int} 1 [0x1]
- argv = {char **} 0x00007ffee3d50970 0x00007ffee3d50970
- main_window = {ei_surface_t} 0x0 {NULL}
- main_window_size = {ei_size_t}
- width = {int} 640 [0x280]
- height = {int} 480 [0x1e0]
- event = {ei_event_t}
- white = {uint32_t} 4294967295 [0xffffffff]
- pixel_ptr = {uint32_t *} 0x00007ffee3d50958 0x00007ffee3d50958
- i = {int} 32766 [0x7ffe]

Problems Debug Terminal Python Packages R Console R Jobs CMake Messages TODO Event Log

Build finished in 202 ms (3 minutes ago)

Gestion de projet

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 mais se **synchroniser** souvent.

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

Ne pas essayer de coder toutes les fonctions des .h systématiquement.

Implémentez les quand vous en avez besoin = vous avez compris à quoi elle serve.

Gestion de projet

Plagiat (copie)

Ne **partager aucun code** avec un autre groupe, même des années précédentes.

Pas de dépôt de code public.

Nous faisons tourner un programme de détection de plagiat.

Plagiat = 0/20 au module.

Déroulement

Chronologie

Aujourd'hui

vous commencez le projet à plein temps

Mercredi 26 mai à 19h

vous **rendez les fichiers** de votre projet sur TEIDE

Jeudi 27 matin

vous **préparez** votre soutenance

Jeudi 27 après-midi et vendredi 28 matin

vous faites une **soutenance** devant un encadrant

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.

Prochaine étape

Grenoble INP
ensimag

Courses at Ensimag

Projet C IG (1A)

IHM (2A)
Cours
Projet
Moderation des sujets
Groupes
Soutenances
FAQ

VR/AR/Post-WIMP (3A)

Projet image (2A)

HCI (MoSIG 2)

Projects Docs

logout fberard

Projet C - Interaction Graphique

Nouvelles

4/5/21 L'encadrement en présence se fait en E200 et E201. Les enseignants sont présents pour vous aider. Ces salles vous sont réservées tout le temps jusqu'au 26/5.
Pour des raisons de traçage des cas de COVID, vous devez [inscrire votre nom](#) à chaque fois que vous venez à l'Ensimag.

4/5/21 L'encadrement à distance se fait sur discord. Consultez la [page discord](#).

16/4/21 C'est ici qu'apparaîtront les nouvelles pour le projet 2020-2021

Fichiers à télécharger

N'imprimez pas le sujet !! (sauvez les arbres !).

- [20200420 SDL2_windows.zip](#) ... 7,441,440 bytes ... April 30, 2021, at 04:44 AM

Autres pages sur le projet

- [Questions fréquentes](#).
- Installation de [SDL](#) sur les machines personnelles.
- Utilisation de [CLion](#).
- Tutoriels en [vidéos](#).

[Edit](#) - [History](#) - [Upload](#) - [Print](#) - [Recent Changes](#) - [Search](#)
Page last modified on May 04, 2021, at 02:01 PM

```
fberard@grostet-ubuntu: ~/projet_c_v6.0 — ttys001
[fberard@grostet-ubuntu:~/projet_c_v6.0/cmake$ cmake ..
-- The C compiler identification is GNU 9.3.0
-- The CXX compiler identification is GNU 9.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Building for Linux with eibase: /home/fberard/projet_c_v6.0/_x11/libeibase64.a
-- Configuring done
-- Generating done
-- Build files have been written to: /home/fberard/projet_c_v6.0/cmake
[fberard@grostet-ubuntu:~/projet_c_v6.0/cmake$ make minimal
Scanning dependencies of target minimal
[ 50%] Building C object CMakeFiles/minimal.dir/tests/minimal.c.o
[100%] Linking C executable minimal
[100%] Built target minimal
[fberard@grostet-ubuntu:~/projet_c_v6.0/cmake$ cd ..
[fberard@grostet-ubuntu:~/projet_c_v6.0$ ./cmake/minimal
```