

Projet S102

**Comparaison d'approches algorithmiques
par la programmation d'un (petit) jeu vidéo**

Simon Giraudot & Hélène Bonneau

Jeu vidéo

Un descendant du jeu de société ?



Jeu vidéo

Un descendant du dessin animé / film d'animation



1958

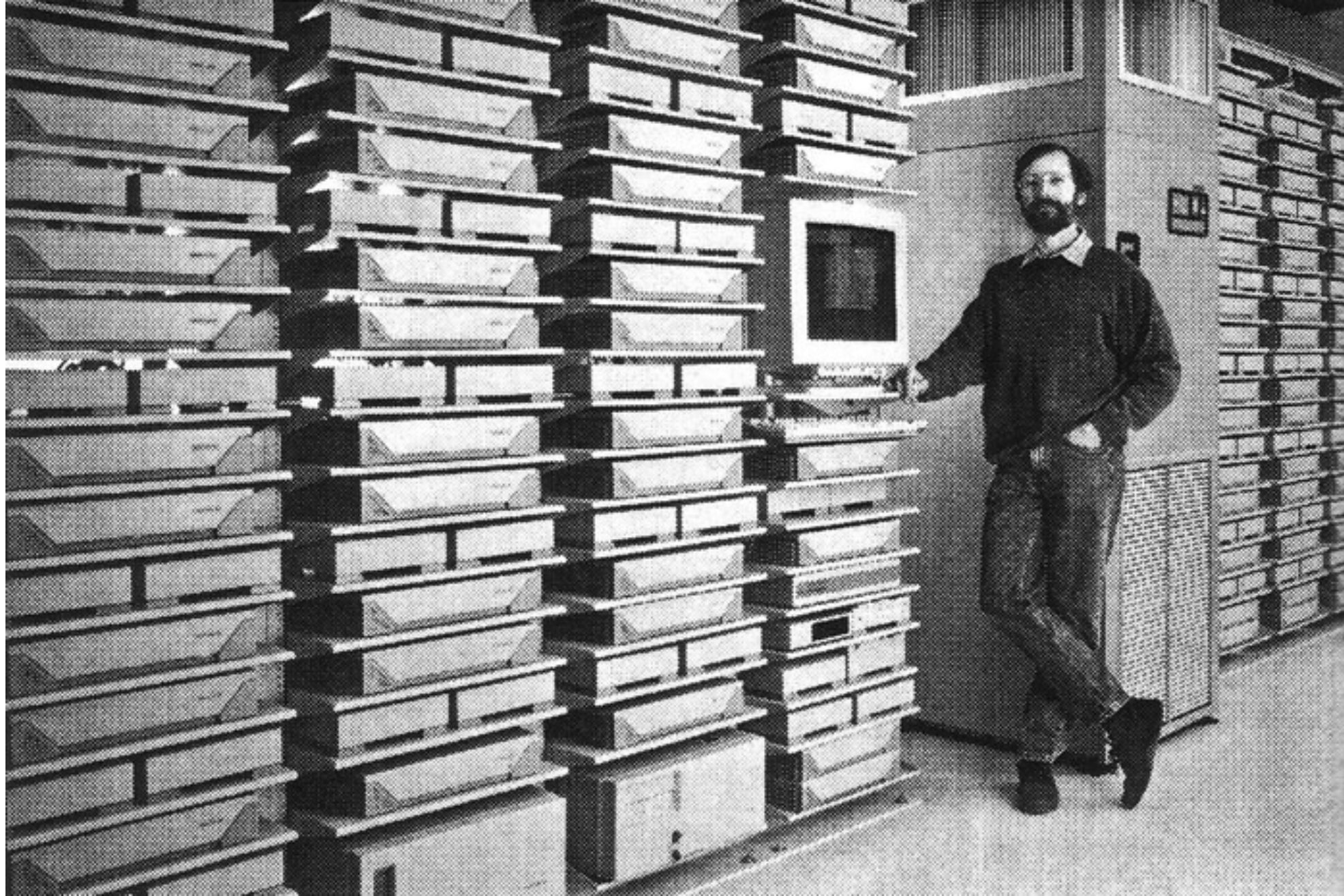
1995

2001



Jeu vidéo

Une différence majeure !



Comment ça marche ?

La boucle de jeu

```
while (!fin)
{
    gererEvenements();
    calculerEtatJeu();
    afficherImage();
}
```

→ tout ça 60 fois par seconde !

Comment on programme ça ?

Différents niveaux

Moteurs de jeu



Bibliothèques



Langages (C++...)

Assembleur



Bibliothèque SDL 2.0

Simple DirectMedia Layer



- ▶ Gérer une fenêtre 2D
- ▶ Charger des images
- ▶ Afficher des images
- ▶ Traiter des événements

(Extensions : SDL Mixer, SDL TTF, SDL Network..)

Bibliothèque SDL 2.0

Simple DirectMedia Layer



Bibliothèque SDL 2.0

Une bibliothèque C

```
// On lance le moteur
if (SDL_Init(SDL_INIT_VIDEO) != 0)
    std::cerr << "SDL_Init failed: " << SDL_GetError() << std::endl;

// Sous-bibliotheque SDL_image qui permet de charger du PNG
if (IMG_Init(IMG_INIT_PNG) == 0)
    std::cerr << "IMG_Init failed: " << SDL_GetError() << std::endl;

// Creation de la fenetre
SDL_Window* window = SDL_CreateWindow
("Mon jeu", // Le titre de la fenetre
 SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
 800, 600, // Taille
 SDL_WINDOW_RESIZABLE); // On peut changer la taille de la fenetre

if (!window)
    std::cerr << "SDL_CreateWindow failed: " << SDL_GetError() << std::endl;
```

```
// Creation du renderer (qui gere l'affichage)
SDL_Renderer* renderer = SDL_CreateRenderer (window, -1,
                                             SDL_RENDERER_ACCELERATED);

if (!renderer)
    std::cerr << "SDL_CreateRenderer failed: " << SDL_GetError() << std::endl;

// Chargement d'image
SDL_Surface* surface = IMG_Load(nomDuFichier.c_str());
SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, surface);

// Rendu de l'image
SDL_Rect target;
target.x = 0;
target.y = 0;
target.w = surface.w;
target.h = surface.h;
SDL_RenderCopy (renderer, texture, NULL, &target);
```


Classes Moteur et Image

Des « wrappers » C++ autour de la SDL

```
class Moteur
{
    // Structures internes de la SDL
    SDL_Window* _window;
    SDL_Renderer* _renderer;
    int _temps;

public:

    // Constructeur qui initialise le moteur, la fenetre, etc.
    // Par default la fenetre fait 160 x 128 pixels (gros 5 fois pour l'affichage)
    // -> l'univers de jeu se fait sur 160 x 128 pixels (c'est arbitraire),
    // soit une grille de 10x8 cases de 16 pixels de côté
    Moteur(const std::string& nomDuJeu);

    // Destructeur qui ferme le moteur et libere la memoire
    ~Moteur();

    // Initialise l'image par un ecran noir
    void initialiserRendu();

    // Finalise l'image et l'envoie a la carte graphique pour affichage sur l'ecran
    void finaliserRendu();

    // Renvoie le dernier evenement reçu (AUCUN si rien n'est reçu)
    Evenement evenementReçu() const;

    // Renvoie true s'il faut mettre a jour les animations
    bool animationsAMettreAJour();

    // Met le jeu en pause pendant un certain nombre de secondes
    // (ATTENTION : le jeu ne répond plus lorsqu'il est en pause,
    // les evenements ne seront traités qu'après la pause)
    void attendre(double secondes) const;

    // (Pour un usage interne, vous n'aurez pas jamais besoin de cette methode.)
    SDL_Renderer* getRenderer();
};
```

```
class Image
{
    Moteur* _moteur;

    // Structures internes de la SDL
    std::shared_ptr<SDL_Texture> _texture;
    SDL_Rect _rectangle;

public:

    Image();

    // Charge l'image (PNG par exemple) contenue dans le fichier
    Image(Moteur& moteur, const std::string& nomDuFichier);

    // Selectionne quelle partie de l'image sera dessinee
    void selectionnerRectangle(int x, int y, int largeur, int hauteur);

    // Dessine l'image (ou la partie selectionnee) aux coordonnees voulues
    void dessiner(int x, int y) const;
};
```

→ dans votre code :

```
Moteur moteur("Nom de mon jeu");
Image image(moteur, "assets/image.png");
```

```
moteur.initialiserRendu();
image.dessiner(42, 64);
moteur.finaliserRendu();
```


Gestion des événements

Entrées au clavier, à la souris, etc.

```
// Boucle de jeu, appelee a chaque fois que l'ecran doit etre mis a jour
// (en general, 60 fois par seconde)
while (!quitter)
{
    // I. Gestion des evenements
    Evenement evenement = moteur.evenementRecu();
    while (evenement != AUCUN)
    {
        switch (evenement)
        {
            // QUITTER = croix de la fenetre ou Echap
            case QUITTER_APPUYE:
                quitter = true;
                break;
            // TODO: gerer les autres evenements
            default:
                break;
        }

        evenement = moteur.evenementRecu();
    }

    // II. Mise à jour de l'état du jeu

    // TODO: faire bouger vos personnages, etc.
```


Gestion des événements

Un point technique : enum

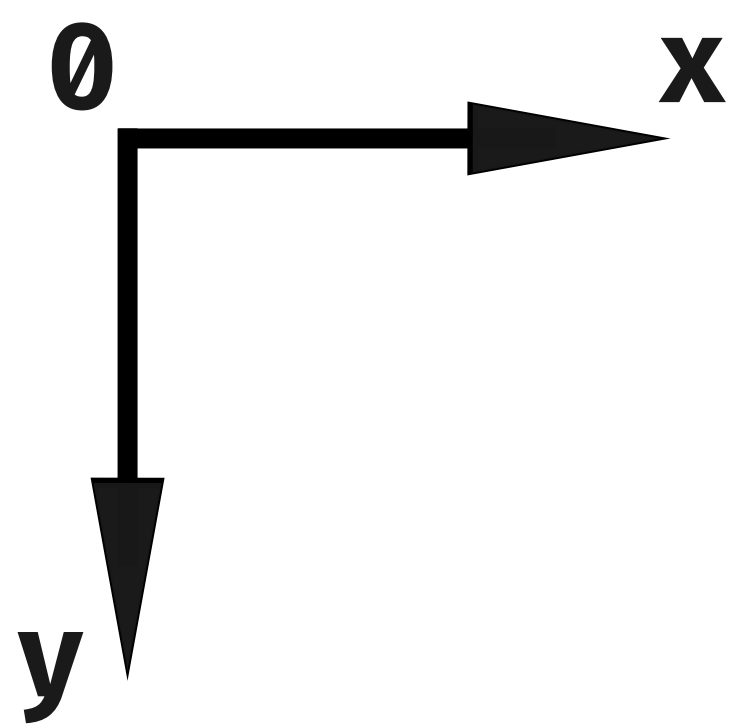
```
// Les différents types d'événements disponibles
enum Evenement {
    AUCUN,
    GAUCHE_APPUYE,
    GAUCHE_RELACHE,
    DROITE_APPUYE,
    DROITE_RELACHE,
    HAUT_APPUYE,
    HAUT_RELACHE,
    BAS_APPUYE,
    BAS_RELACHE,
    ESPACE_APPUYE,
    ESPACE_RELACHE,
    QUITTER_APPUYE,
    QUITTER_RELACHE
};
```



```
// Les différents types d'événements disponibles
const int AUCUN = 0;
const int GAUCHE_APPUYE = 1;
const int GAUCHE_RELACHE = 2;
const int DROITE_APPUYE = 3;
const int DROITE_RELACHE = 4;
const int HAUT_APPUYE = 5;
const int HAUT_RELACHE = 6;
const int BAS_APPUYE = 7;
const int BAS_RELACHE = 8;
const int ESPACE_APPUYE = 9;
const int ESPACE_RELACHE = 10;
const int QUITTER_APPUYE = 11;
const int QUITTER_RELACHE = 12;
```


Gestion de l'affichage

Assets, sprites, tuiles..

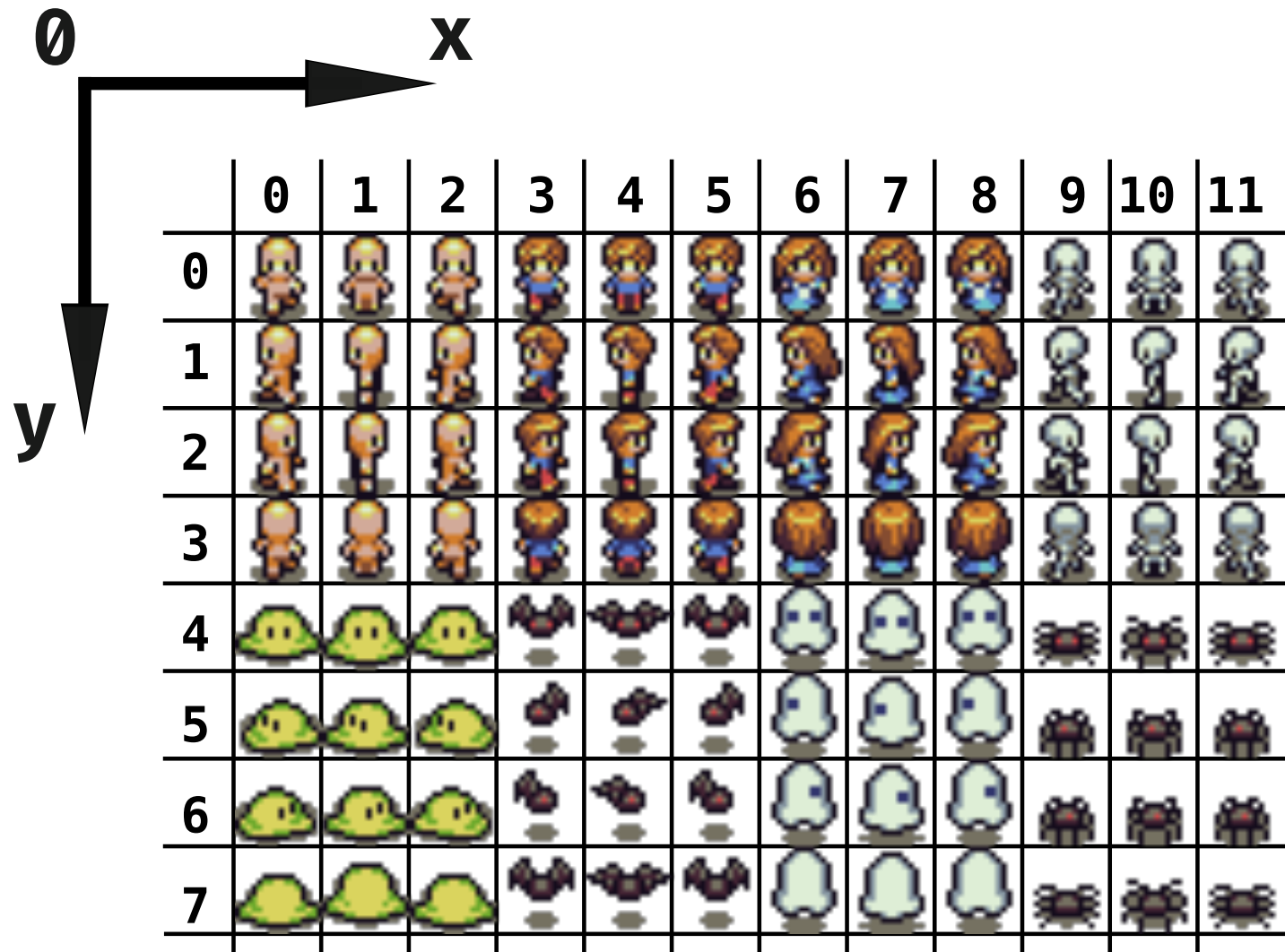


	0	1	2	3	4	5	6	7	8	9	10	11
0												
1												
2												
3												
4												
5												
6												
7												



Gestion de l'affichage

Assets, sprites, tuiles..



```
class Image
{
    Moteur* _moteur;

    // Structures internes de la SDL
    std::shared_ptr<SDL_Texture> _texture;
    SDL_Rect _rectangle;

public:

    // Cree un objet image vide
    Image();

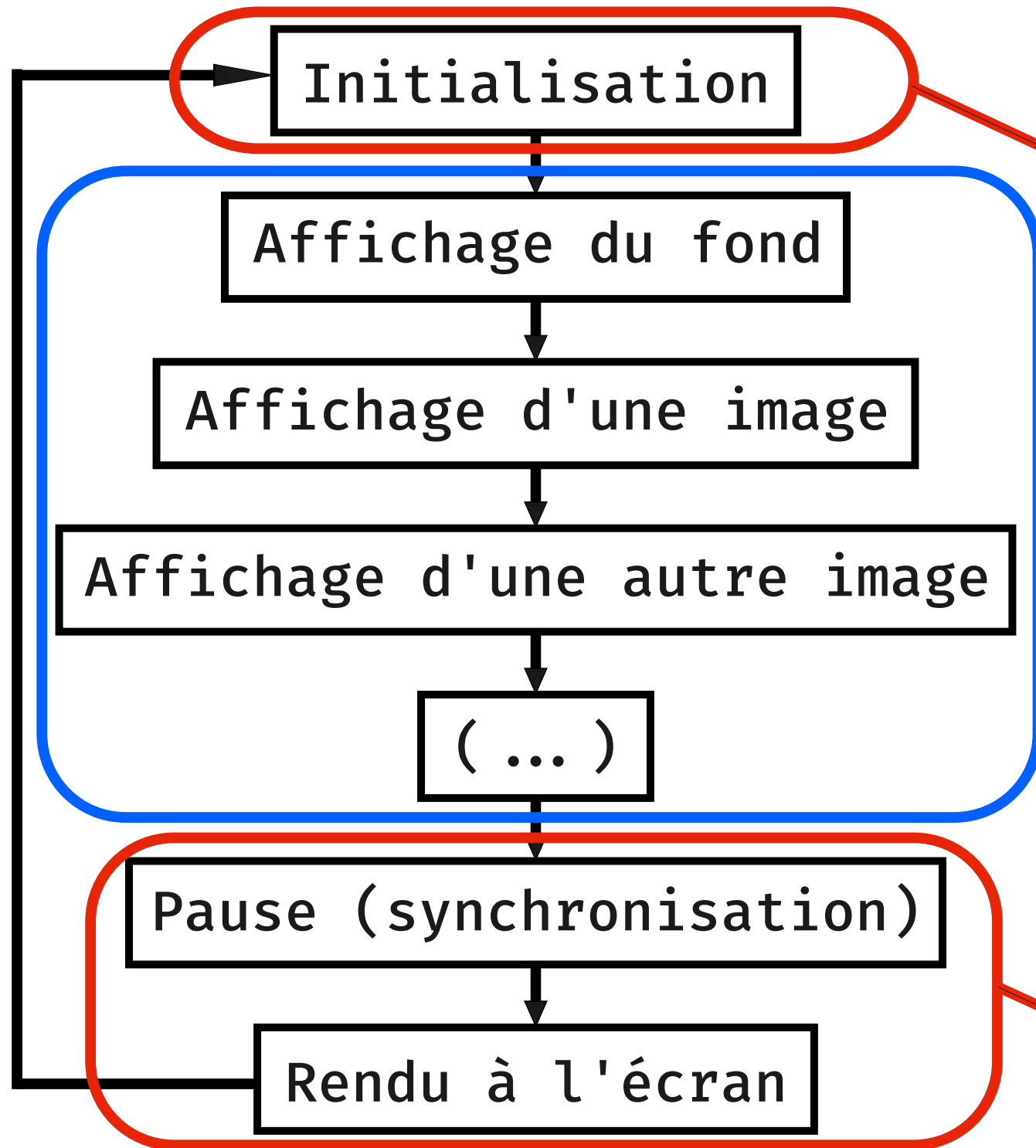
    // Charge l'image (PNG par exemple) contenue dans le fichier
    Image(Moteur& moteur, const std::string& nomDuFichier);

    // Selectionne quelle partie de l'image sera dessinee
    void selectionnerRectangle(int x, int y, int largeur, int hauteur);

    // Dessine l'image (ou la partie selectionnee) aux coordonnees voulues
    void dessiner(int x, int y) const;
};
```


Gestion de l'affichage

Le principe du rendu



// III. Generation de l'image à afficher

*/**

*Efface ce qui avait ete affiche precedemment
et reinitialise en ecran noir*

**/*

`moteur.initialiserRendu();`

// TODO: afficher vos personnages, objets, etc.

*/**

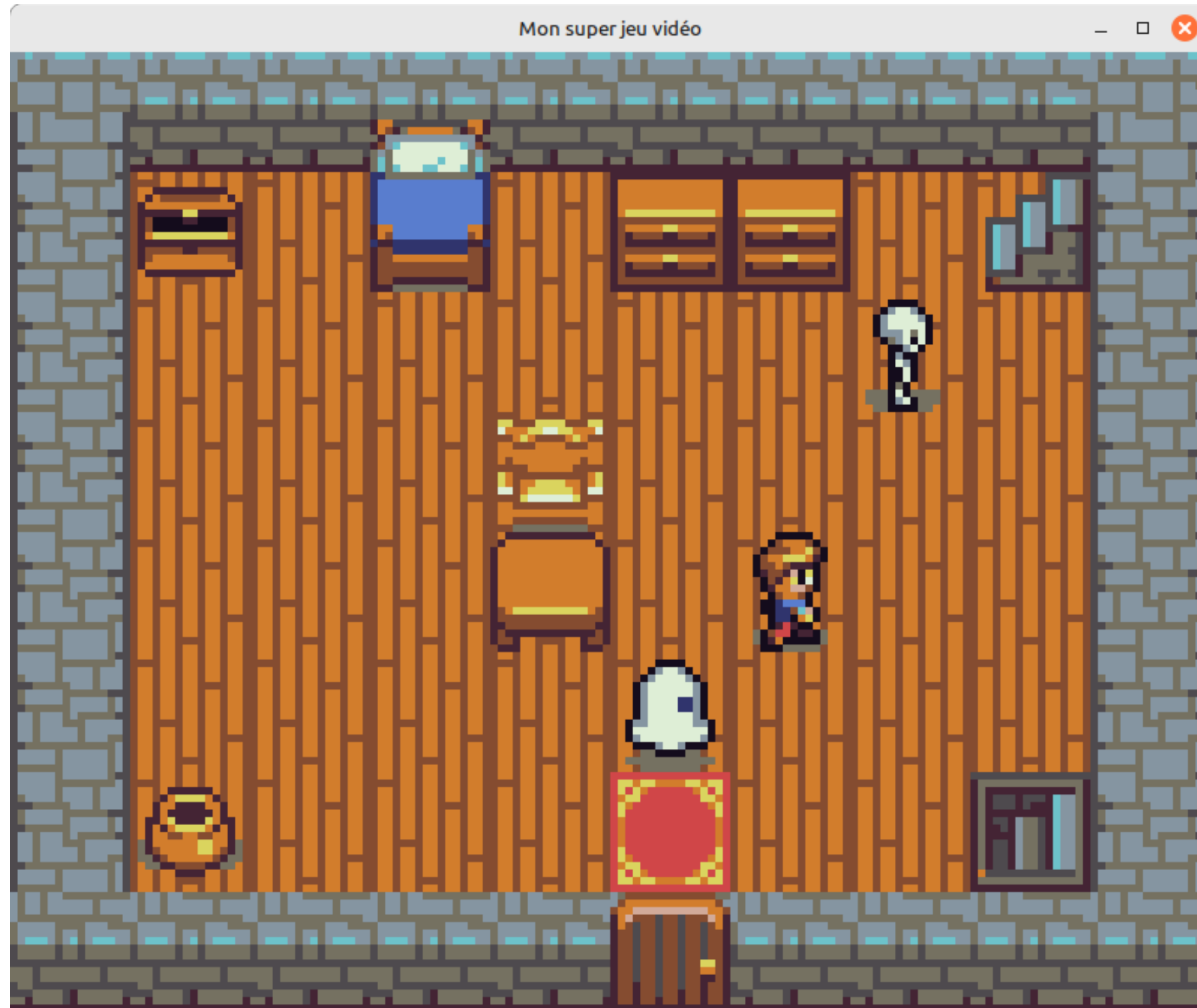
*Affiche l'image en se cadencant sur la
frequence de rafraichissement de l'ecran
(donc va en general mettre le programme en
pause jusqu'a ce que l'ecran soit rafraichi).
En general, 60 images fois par seconde, mais
ca peut dependre du materiel*

**/*

`moteur.finaliserRendu();`

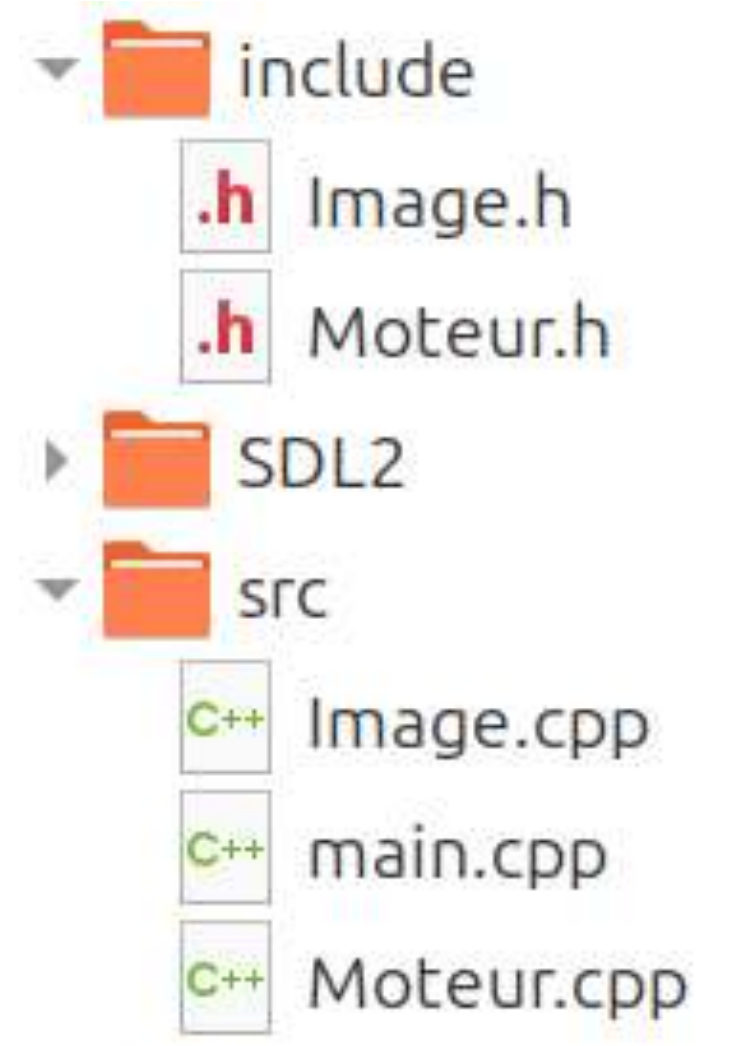
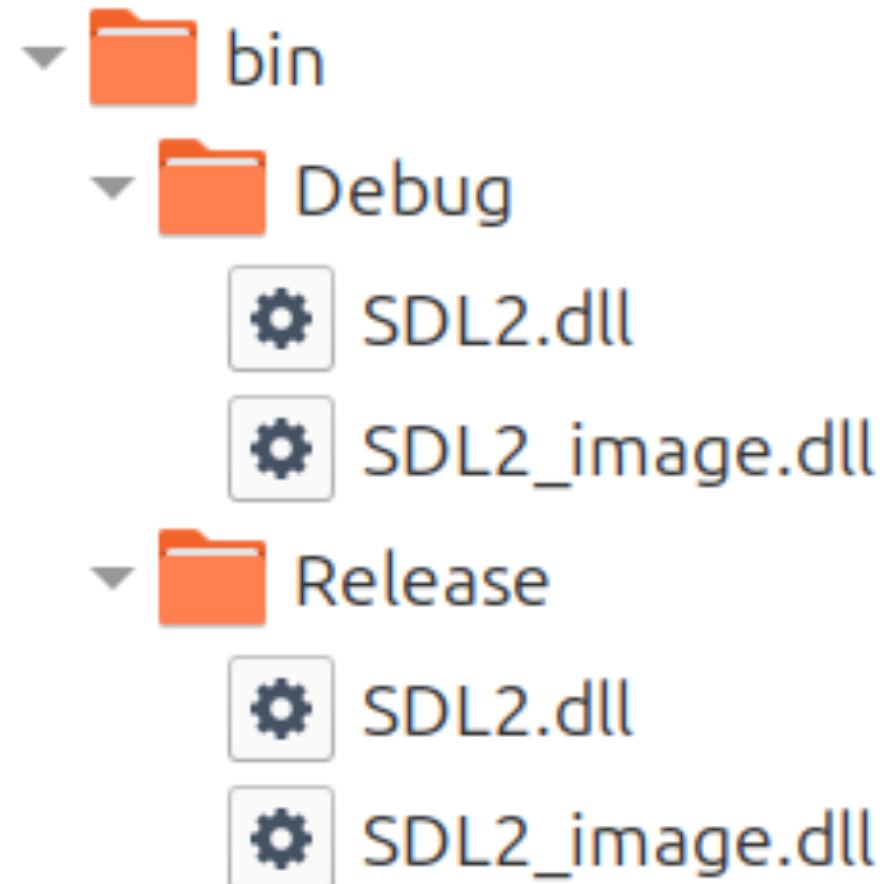
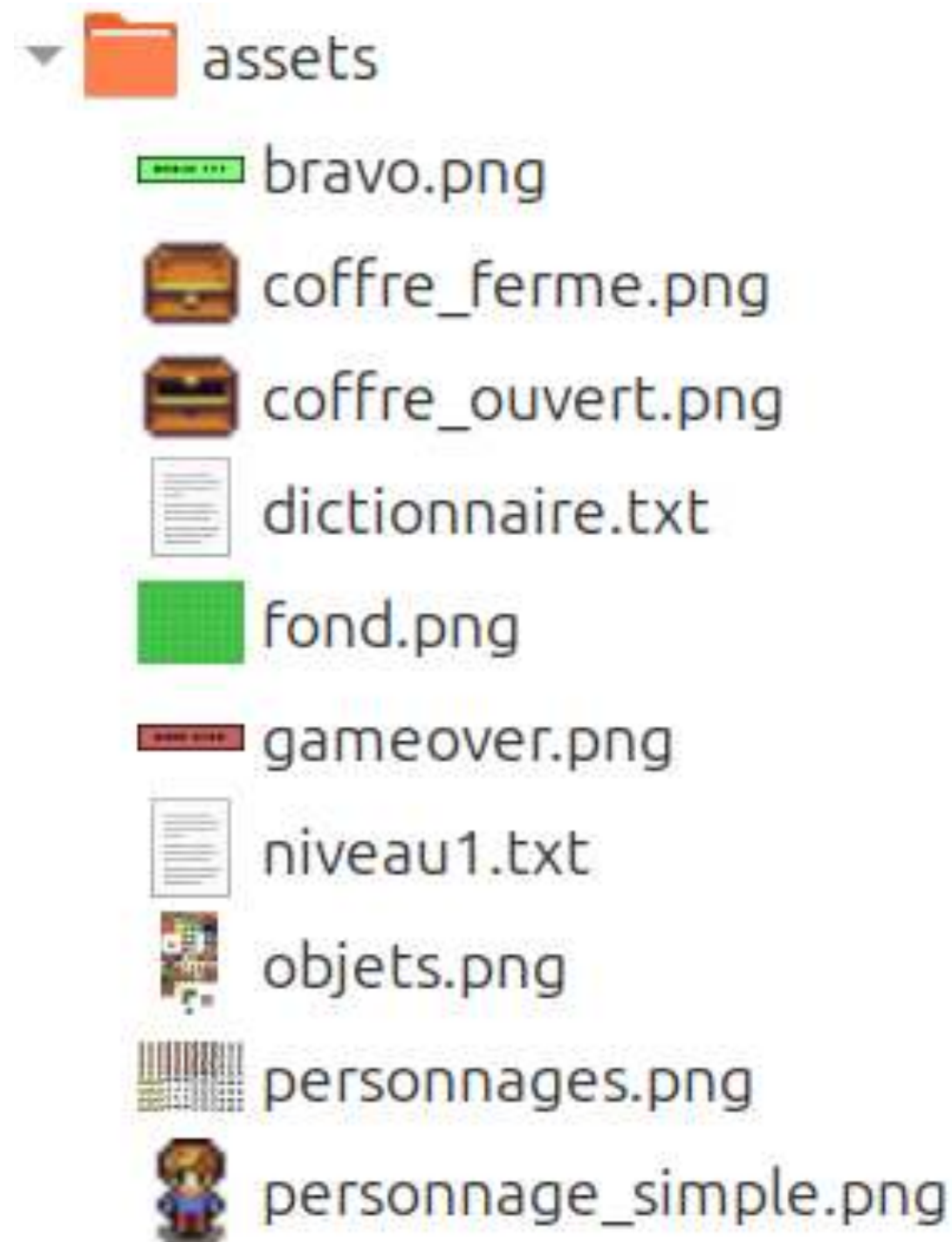
Le jeu que vous allez programmer

2D, pixel art, vue de dessus, jeu au clavier



Projet CodeBlocks fourni

Il suffit de le charger !



**à ouvrir dans
CodeBlocks**

Partage de code

Par dossier partagé

- ▶ Créer un dossier dans le cloud
- ▶ Partage en lecture/écriture
- ▶ Le dossier apparaîtra à la connexion


Évaluation

Sur plusieurs points

- ▶ Rendu (note commune)
- ▶ Quiz (note individuelle)
- ▶ TP noté : ajout de fonctionnalité (note individuelle)

Les graphismes

Open Game Art

**OPENGAMEART.ORG**

OpenID

Username or

Password

LOG IN

Register

HomeBrowseSubmit ArtCollectForumsFAQLeaderboardsDonate

CHAT WITH US!

Discord: OpenGameArt
<https://discord.gg/yDaQ4NcCux>

IRC: #OpenGameArt on
<https://freegamedev.net/irc/#opengameart>

ACTIVE FORUM TOPICS - (VIEW MORE)

- Videogame - The past-creeps into the darkness 15 hours 6 min ago by DREAM_SEARCH_REPEAT
- Sex care products, can i upload them? 1 day 19 hours ago by DREAM_SEARCH_REPEAT
- Change Username Requests 1 day 23 hours ago by LaurentO
- Sharing My Music and Sound FX - Over 2000 Tracks 2 days 12 hours ago by Eric Matyas
- Building a Library of Images for Everyone 5 days 28 min ago by Eric Matyas
- Collaborate with Tibia Harry Potter Origin. 5 days 18 hours ago by L3K0T
- [AFFORDABLE] 2D/3D Game & Animation Artists AVAILABLE for work @ indie price 1

LEGAL NOTICE REGARDING NFTs:

WARNING: Taking art from OpenGameArt.org to be sold as NFTs? You may be committing FRAUD. Visit this link for legal details: <https://opengameart.org/content/warning-taking-art-from-opengameartorg-t...>

Note of caution to NFT purchasers or those interested in trading NFTs: You could be getting scammed! Please visit this link for more information: <https://opengameart.org/content/note-of-caution-to-nft-purchasers-or-tho...>

POPULAR THIS WEEK - (VIEW MORE)

CHIROPTERA

SUNKEN SHIP ...

COWBOY HAT

GENERATOR

PIXEL ART FLA...

WALL SCENCE ...

COMMANDO

FANTASY CLAS...

LATEST ART - (VIEW MORE)

CHIROPTERA, ...

DRONE 53


RPG CHARACT...

WALL SCENCE ...

COMMANDO

FANTASY CLAS...



Tiny 16: Basic, licence Creative Commons By 4.0



 creative commons




MOST OPEN




LEAST OPEN





Licenses






  ZERO






  BY

   BY SA


   BY NC


    BY ND

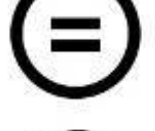
     BY NC SA


     BY NC ND


Icons

 **Public Domain Dedication (CC0)**
This is considered a dedication to the public domain, and thus the creator(s) associated with this item have waived all their rights to the work worldwide under copyright law.

 **Attribution (BY)**
Others can copy, distribute, display, perform and remix the work if they credit/cite the creator/author.

 **Derivative Works (ND)**
Others can only copy, distribute, display or perform verbatim copies of the work. (No modifications allowed.)

 **Share Alike (SA)**
Others can distribute the work only under a license identical to the one attached to the original work.

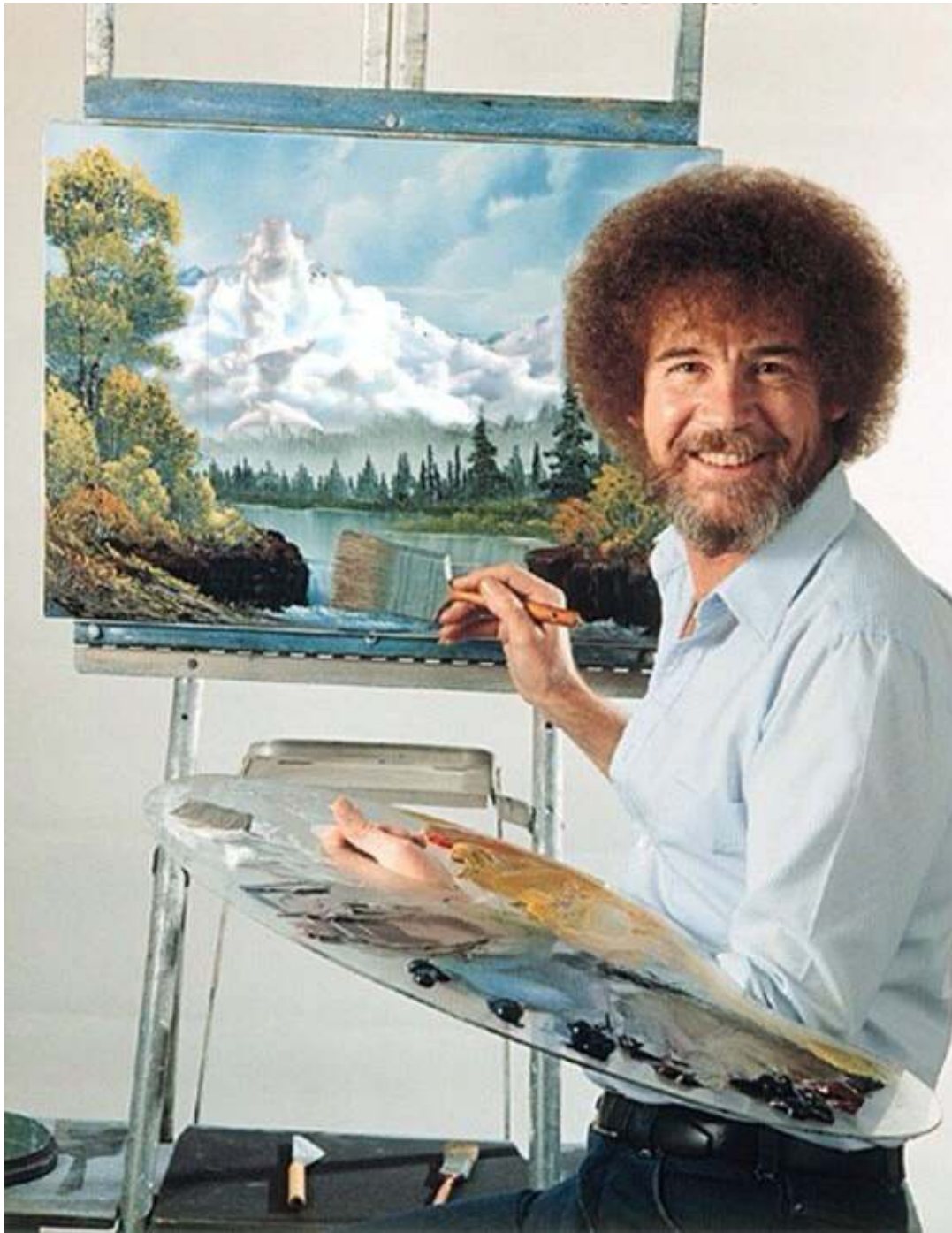
 **Non-Commercial (NC)**
Others can copy, distribute, display, perform or remix the work but only for non-commercial purposes.

Terms of the Licenses

This work is a CC0 Public Domain Dedication work.

Oui mais...

J'ai une âme d'artiste



- ▶ Vous êtes en IUT d'informatique
- ▶ Nous n'évaluerons pas ça
- ▶ Nous ne vous aiderons pas
- ▶ Ne perdez pas de temps là-dessus
- ▶ ... mais à part ça,
faites-vous plaisir !

Oui mais...
J'ai tout fini en avance

- ▶ ... vraiment ?
- ▶ Questions optionnelles
- ▶ Ajout de fonctionnalités

Merci de votre attention !

Des questions ?