

Projet informatique CPI2 2023/2024

Thème : création d'un jeu en 3D Ray casting avec la SDL en C

Professeur : M. Calcado

Etudiants :

Diakité Ousmane
Djerlil Salah
Lacroze Maxime

Lien pour visionner le projet :

https://github.com/Ousmane001/jeu_sdl_3d_en_c_raycasting

Cahier de charge du projet d'informatique :

Créer un jeu avec du graphisme en utilisant essentiellement la bibliothèque SDL en langage C.

Objectifs :

- Savoir comment télécharger et manipuler des bibliothèque externe au langage
- Apprendre des outils nécessaire pour un projet informatique (git, make,...)
- Créer un jeu FPS en 3D en utilisant la technique de Ray casting

Sommaire

1. Recherche et apprentissage de tous les outils nécessaire a la réalisation du projet
2. Initialisation à la SDL
3. Création d'une branche pour les fonctions gérant la vision et l'affichage 3D
4. Création d'une branche pour l'interaction avec le joueur via le menu
5. Création d'une branche pour la musique et tout ce qui arrière – plan
6. Création d'une branche pour les PNJ et de toutes les interactions dans le jeu
7. Chargement des ressources graphiques pour les fonctionnalités du programme
8. Assemblage et fusion des différentes branches
9. Test du logiciel et débogage des potentielles erreurs
10. Soutenance.

1- Recherche et apprentissage de tous les outils nécessaire a la réalisation du projet



A -) Git & GitHub :

introduction :

Git est un système de gestion de versions distribué qui permet de suivre les modifications apportées à des fichiers et des dossiers au fil du temps.

Il est extrêmement utilisé dans la collaboration dans un projet comportant plusieurs personnes.

En effet, dès les premières séances du projet, nous nous sommes rendu compte de la difficulté de collaborer tous dans un même projet en un même instant avec actualisation automatique. Les sites web tels que replit ne permettant pas d'afficher du graphisme via des fenêtres, nous étions obligé de chercher un moyen de partager nos modifications sur le projet sans avoir à tout compresser et à partager ce fichier sur discord à chaque fois.

C'est ainsi que nous avons découvert git. En plus du partage et d'hébergement de code gratuit, git nous a permis de versionner notre projet et de pouvoir revenir sur une version précédente facilement sans avoir à copier à chaque fois l'intégralité du projet comme sauvegarde.

Mais ce qui est de loin l'avantage ultime de git, c'est le fait qu'il nous a permis à chacun de travailler en même temps sur des parties différentes du projet sans que cela n'affecte en aucun cas la partie d'un autre collaborateur avec le système de branches qu'il offre. Et à présent, notre projet compte 4 branches au total, une pour chacun d'entre nous et une dernière qui n'intervient qu'une fois que les fonctions d'une branche sont jugées à l'unanimité correctes.

Conclusion :

Sans git, on aura eu du mal à partager le code de notre projet et à pouvoir le versionner et l'utiliser comme on le veut. Et surtout que ce outil va nous servir énormément dans notre carrière informatique !

B- Logiciel Make :

Introduction :



Make est un outil puissant qui permet de générer des fichiers à partir d'autres fichiers, en fonction de règles définies par l'utilisateur. Il est utilisé par des développeurs de logiciels, des administrateurs système et des personnes de tous horizons pour automatiser la construction de projets.

Make fonctionne en lisant un fichier Makefile, qui contient une liste de règles. Chaque règle définit comment un fichier est généré à partir d'autres fichiers. Les règles sont composées d'une cible, d'une liste de dépendances et d'une commande.

En effet, l'augmentation constante du nombre de répertoires et de fichiers sources contenus dans ceux-ci nous rendait la tâche de compilation de plus en plus difficile, et surtout qu'à chaque nouveau fichier source, il faut obligatoirement l'inclure dans la commande de compilation du gcc. Et le pire c'est que ça prenait du temps pour chaque compilation qui était totale pour tous les fichiers sources.

Face à cela, nous nous sommes posé une question, n'est-t-il pas possible d'automatiser cela rapidement de façon générique de telle manière à ce que même si nous rajoutons des fichiers sources, eh bien nous aurons pas à les rajouter dans cette commande de compilation et que ceci se fasse automatiquement ?

La réponse de cette question a été « l'outil make ». En plus d'atteindre nos exigences, l'outil make ne compile que les fichiers qui ont été modifiés depuis la dernière compilation. Avec ça, la compilation était devenue très rapide car il n'y avait que peu de fichiers à compiler !

Conclusion :

L'outil make est de loin celui qui nous a permis de gagner en productivité et surtout que ce dernier nous servira plus tard dans le monde professionnel et le fait qu'il soit compatible avec tous les autres langages que nous verrons !

2- Initialisation à la SDL

Introduction :

La SDL (Simple Direct Media Layer) est une bibliothèque puissante et flexible qui peut être utilisée pour créer une variété d'application allant de la gestion du graphisme au multimédia etc..

C'est une bibliothèque externe du langage C, ce qui veut dire donc qu'il faut le télécharger l'installer directement (il automatiquement inclus dans le gcc !).

Dans le cadre de notre projet, nous avons utilisé le système d'exploitation linux pour développer le programme. Et comme beaucoup d'autres bibliothèques, linux contient déjà le package SDL qu'il suffit juste de télécharger avec les commandes suivantes :

Commande pour mettre a jour les packages de linux :

sudo apt update

sudo apt upgrade

sudo apt install build-essential

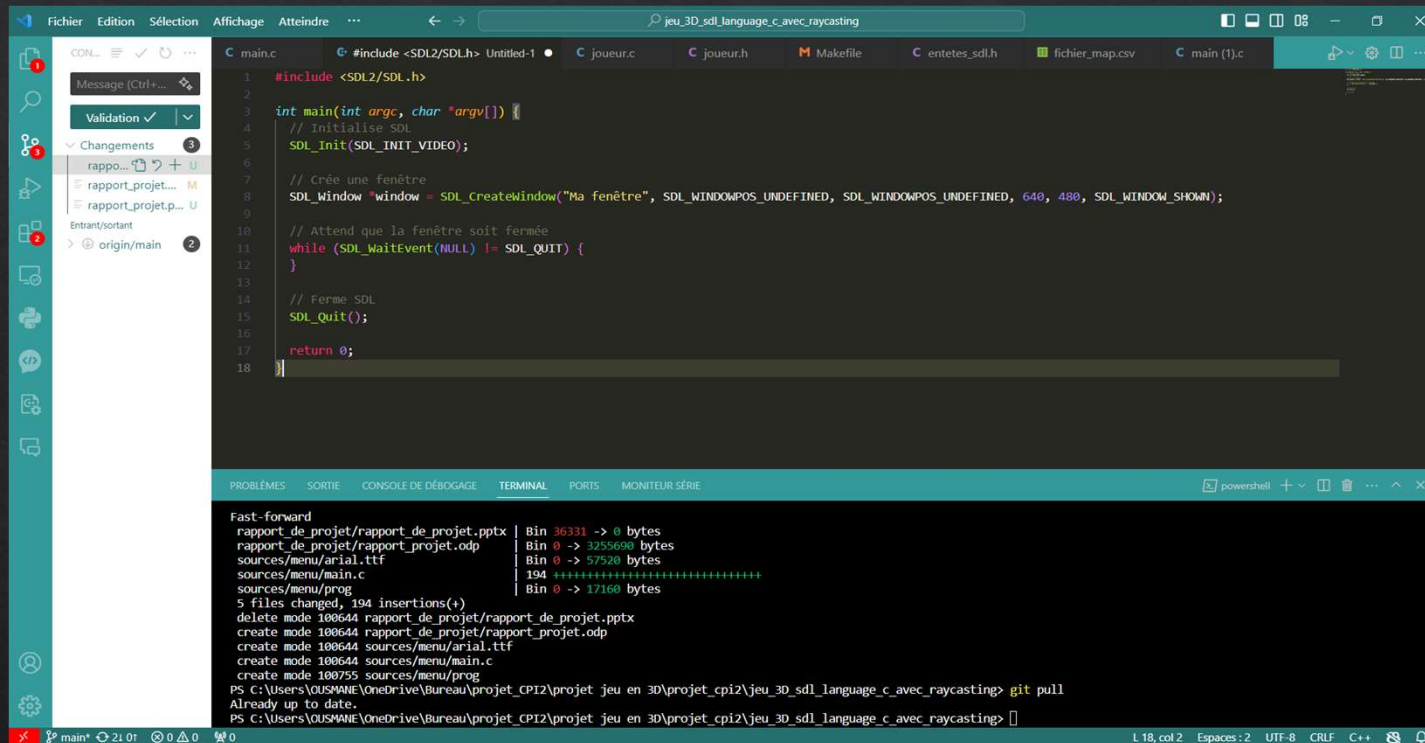
Commande pour installer les fichiers SDL qui nous intéressent :

sudo apt Install libsdl2-dev libsdl2-ttf-dev

Et voila donc que nous avons installé très facilement sur linux la bibliothèque externe SDL dans la suite nous verrons comment compiler un premier programme.

Premier programme :

Voici un exemple de code que nous avons utilisé pour créer et afficher une fenêtre de taille 640 * 480 pixels. Le processus attendra que l'utilisateur clique sur l'icone « quitter » de la fenêtre ou sur Alt+F4 avant de s'arrêter.



```
1 #include <SDL2/SDL.h>
2
3 int main(int argc, char *argv[]) {
4     // Initialise SDL
5     SDL_Init(SDL_INIT_VIDEO);
6
7     // Crée une fenêtre
8     SDL_Window *window = SDL_CreateWindow("Ma fenêtre", SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 640, 480, SDL_WINDOW_SHOWN);
9
10    // Attend que la fenêtre soit fermée
11    while (SDL_WaitEvent(NULL) != SDL_QUIT) {
12    }
13
14    // Ferme SDL
15    SDL_Quit();
16
17    return 0;
18 }
```

```
Fast-forward
rapport de projet/rapport_de_projet.pptx | Bin 36331 -> 0 bytes
rapport de projet/rapport_projet.odp | Bin 0 -> 3255690 bytes
sources/menu/arial.ttf | Bin 0 -> 57520 bytes
sources/menu/main.c | 194 ++++++
sources/menu/prog | Bin 0 -> 17160 bytes
5 files changed, 194 insertions(+)
delete mode 100644 rapport_de_projet/rapport_de_projet.pptx
create mode 100644 rapport_de_projet/rapport_projet.odp
create mode 100644 sources/menu/arial.ttf
create mode 100644 sources/menu/main.c
create mode 100755 sources/menu/prog
PS C:\Users\OUSMANE\OneDrive\Bureau\projet_CPI2\projet jeu en 3D\projet_cpi2\jeu_3d_sdl_language_c_avec_raycasting> git pull
Already up to date.
PS C:\Users\OUSMANE\OneDrive\Bureau\projet_CPI2\projet jeu en 3D\projet_cpi2\jeu_3d_sdl_language_c_avec_raycasting>
```

Si l'installation de la SDL fut très facile, la compilation n'en a pas été autant !

Car d'après nos recherches sur le wiki de la SDL, après avoir tapez la commande de compilation (comme d'habitude) il faut aussi comme toutes les bibliothèque externes tapez son Link.

Pour la SDL2 en système linux, il suffit d'ajouter à cette commande de compilation « \$(sdl2-config --cflags --libs) » Heureusement nous savons l'automatiser avec l'outil make vu précédemment !

3- Branche des fonctions gérant la vision et l'affichage 3D

A- gestion de la map :

Pour la gestion de la map, nous avons optez de sauvegarder statiquement celle-ci dans un fichier CSV.

Tout un répertoire portant sur le nom de gestion_du_map à été créée pour bien la gérer. Cette map sera dans le cadre de notre projet, une matrice dont les coefficient déterminerons l'état du lieu courant (ex : le coefficient 0 indiquera au joueur que cette partie du jeu contient que du vide, il pourra donc se déplacer etc. ...)

Dans le fichier .CSV contenant les informations de la map, nous avons mis tout au début le nombre de ligne de la matrice_map, puis dans la ligne suivante, le nombre de colonne de celle-ci.

Ensuite, tout les autres nombres détermineront l'état de la map.

Ainsi lors du lancement du jeu, le processus chargera cette map et l'affichera dans la fenêtre prévu a cet effet.

Il se peut que dans certains cas, pour une raison X inconnue, que le fichier_map contenant la map soit corrompu (erreur sur les coordonnées ou manque de coefficients ou manque de lignes ou de colonnes etc. ...), dans ce cas le processus du jeu doit s'interrompre immédiatement en affichant l'erreur concerné.

C'est pourquoi nous avons codé avant tout une fonction « scanner_une_map » qui permet de vérifier que le fichier map n'est pas corrompu !

Suite de quoi un tableau 2D d'entiers représentatif de la matrice_map est créée via la fonction allouer_map en fonction des informations lues dans le fichier map.

Puis le processus termine par charger cette map dans la matrice via la fonction « charger_une_map » et ainsi le tour est joué ! (à noter que toute ces fonctions ne retournent que des codes d'erreurs ou de succès qui seront ensuite utilisés pour interrompre ou non le processus principal .)

D'autres fonctions comme `afficher_une_map` ou `afficher un mur 2D` ont ensuite été créées pour afficher une miniature du plan de la map pour permettre au joueur de s'orienter.

3- Branche des fonctions gérant la vision et l'affichage 3D

B- gestion du joueur :

C'est sans doute la partie la plus importante et la plus délicate du projet qui est d'ailleurs toujours en cours de programmation. Tout d'abord, nous avons créé une structure « struct joueur » représentant un joueur, un bandit ou un PNJ du jeu. Pour l'instant, les attributs du joueur ne sont que sa position dans la map, et le début de son angle de vision. D'autres attributs ou « champs » comme sa durée de vie, l'état de ses munitions etc. ... s'ajouteront au fur et à mesure de l'avancement du projet.

Nous avons défini des fonctions comme `afficher_joueur` pour afficher le joueur en carré couleur rouge dans la miniature de la map. Nous avons aussi défini une autre qui permet de supprimer son ancienne position une fois qu'il aura bougé sans avoir à nettoyer tout le rendu (pour éviter des effets désagréables aux yeux).

L'une des fonctions les plus importantes est « `deplacer_joueur` » qui prend plein d'arguments dont l'événement qui conduit au déplacement du joueur (touche clavier haut, bas etc. ... ou changement de vision avec la touche Q, D etc....) après avoir si possible, en fonction du map, incrémenter la position du joueur, la fonction se charge aussi de sauvegarder son ancienne position pour que celle-ci soit supprimée.

Et bien entendu, toutes ces fonctions citées depuis le début du rapport ne retournent que des codes (erreur ou succès) à travers des macros constantes qui nous permettront de savoir l'état de déroulement de la fonction.

Ainsi vient la fonction mère de notre projet, celle qui permet de lancer un rayon et de récupérer la distance séparant le joueur de l'obstacle. Pour cela nous sommes libres d'utiliser l'algorithme de Bresenham ou imaginer notre propre algorithme. Cette partie est toujours en cours de développement, nous fourniront plus d'informations d'ici la fin du projet !

To be continued ...

4 - Branche des fonctions gérant l'interaction avec le joueur via le menu

4 - Branche des fonctions gérant l'interaction avec le joueur

via le menu :

Présentation des Fonctions :

◆ 1. `initSDL()` : Cette fonction initialise la bibliothèque SDL ainsi que `SDL_ttf`. Elle crée une fenêtre, un « render » pour effectuer le dessin, et charge une police TTF. En cas d'échec, elle affiche des messages d'erreur et termine le programme.

```
57 // Initialisation de la SDL et de SDL_ttf
58 int initSDL() {
59     if (SDL_Init(SDL_INIT_VIDEO) != 0) {
60         SDL_Log("Erreur SDL_Init : %s", SDL_GetError());
61         return -1;
62     }
63
64     if (TTF_Init() != 0) {
65         SDL_Log("Erreur TTF_Init : %s", TTF_GetError());
66         SDL_Quit();
67         return -1;
68     }
69
70     window = SDL_CreateWindow("Menu Principal", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED, WINDOW_WIDTH, WINDOW_HEIGHT, 0);
71     if (!window) {
72         SDL_Log("Erreur SDL_CreateWindow : %s", SDL_GetError());
73         SDL_Quit();
74         return -1;
75     }
76
77     renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
78     if (!renderer) {
79         SDL_Log("Erreur SDL_CreateRenderer : %s", SDL_GetError());
80         SDL_DestroyWindow(window);
81         SDL_Quit();
82         return -1;
83     }
84
85     font = TTF_OpenFont(FONT_PATH, 24);
86     if (!font) {
87         SDL_Log("Erreur TTF_OpenFont : %s", TTF_GetError());
88         SDL_DestroyRenderer(renderer);
89         SDL_DestroyWindow(window);
90         SDL_Quit();
91         return -1;
92     }
93
94     return 0;
95 }
```


◆2. `initButton(Button *button, int x, int y, int w, int h, SDL_Color color, const char *text)` : C'est une fonction pour initialiser un bouton avec ses propriétés telles que les coordonnées, dimensions, couleur, et texte. Elle prend en paramètre un pointeur vers la structure `Button` et les différentes propriétés du bouton. Cette fonction peut être modifiée pour créer d'autres menus en ajustant les propriétés des boutons.

```
97 // Initialisation des boutons
98 void initButton(Button *button, int x, int y, int w, int h, SDL_Color color, const char *text) {
99     button->x = x;
100    button->y = y;
101    button->w = w;
102    button->h = h;
103    button->color = color;
104    button->text = text;
105 }
```

◆3. `renderButton(const Button *button)` : Cette fonction « affiche » un bouton sur l'écran. Elle remplit un rectangle avec la couleur du bouton, puis ajoute le texte au centre du bouton. Le texte est rendu en utilisant la police chargée. Le rendu sera mis à jour à chaque itération de la boucle.

```
107 // Afficher un bouton
108 void renderButton(const Button *button) {
109     SDL_SetRenderDrawColor(renderer, button->color.r, button->color.g, button->color.b, button->color.a);
110     SDL_RenderFillRect(renderer, &(SDL_Rect){button->x, button->y, button->w, button->h});
111
112     SDL_Surface *textSurface = TTF_RenderText_Solid(font, button->text, BACKGROUND_COLOR);
113     SDL_Texture *textTexture = SDL_CreateTextureFromSurface(renderer, textSurface);
114
115     int textW, textH;
116     SDL_QueryTexture(textTexture, NULL, NULL, &textW, &textH);
117     SDL_Rect textRect = {
118         button->x + (button->w - textW) / 2,
119         button->y + (button->h - textH) / 2,
120         textW,
121         textH
122     };
123     SDL_RenderCopy(renderer, textTexture, NULL, &textRect);
124
125     SDL_DestroyTexture(textTexture);
126     SDL_FreeSurface(textSurface);
127 }
128
```


5 - Branche des fonctions gérant la musique et tout ce qui arrière – plan

Introduction :

Comme beaucoup de jeu introduire de la musique et des effets sonores est un point important. En effet que serait Mario sans sons. C'est pourquoi il était indispensable de créer des fonctions capables de gérer des musiques et des effets sonores. Pour cela un premier problème c'est posé :

installer la bibliothèque SDL_mixer.

Pour cela même manière de faire qu'avec d'autre composant de la SDL. Il faut utiliser les commandes :

apt-get install libsdl2-mixer-2.0-0

apt-get install libsdl2-mixer-dev

```
Fichier  Édition  Affichage  Recherche  Terminal  Aide
lacroze@LennsJourney:~$ su -
Mot de passe :
root@LennsJourney:~# apt-get install libsdl2-mixer-2.0-0
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfluidsynth3
The following NEW packages will be installed:
  libfluidsynth3 libsdl2-mixer-2.0-0
0 upgraded, 2 newly installed, 0 to remove and 16 not upgraded.
Need to get 0 B/312 kB of archives.
After this operation, 782 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Selecting previously unselected package libfluidsynth3:amd64.
(Reading database ... 242888 files and directories currently installed.)
Preparing to unpack .../libfluidsynth3_2.2.5-1_amd64.deb ...
Unpacking libfluidsynth3:amd64 (2.2.5-1) ...
Selecting previously unselected package libsdl2-mixer-2.0-0:amd64.
Preparing to unpack .../libsdl2-mixer-2.0-0_2.0.4+dfsg1-4build1_amd64.deb ...
Unpacking libsdl2-mixer-2.0-0:amd64 (2.0.4+dfsg1-4build1) ...
Setting up libfluidsynth3:amd64 (2.2.5-1) ...
Setting up libsdl2-mixer-2.0-0:amd64 (2.0.4+dfsg1-4build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
root@LennsJourney:~# apt-get install libsdl2-mixer-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  libsdl2-mixer-dev
0 upgraded, 1 newly installed, 0 to remove and 16 not upgraded.
Need to get 0 B/83,3 kB of archives.
After this operation, 333 kB of additional disk space will be used.
Selecting previously unselected package libsdl2-mixer-dev:amd64.
(Reading database ... 242899 files and directories currently installed.)
Preparing to unpack .../libsdl2-mixer-dev_2.0.4+dfsg1-4build1_amd64.deb ...
Unpacking libsdl2-mixer-dev:amd64 (2.0.4+dfsg1-4build1) ...
Setting up libsdl2-mixer-dev:amd64 (2.0.4+dfsg1-4build1) ...
root@LennsJourney:~#
```


Comment ça marche :

La musique et les sons sous SDL2 peuvent être réparties en deux catégories :

- Les pointeurs Mix_Music*
- Les pointeurs Mix_Chunk*

Pour pouvoir jouer une musique de fond il suffira de créer un pointeur Mix_Music de load le fichier de la musique et de la jouer.

Les chunk eux sons tous les effets sonores qu'il est possible d'entendre par dessus la music.

Pour cela il nécessaire d'avoir plusieurs canaux de sons.

```
//p est le chemin vers un fichier .mp3
Mix_Music* loadM(char* p){

    Mix_Music* musique=NULL;
    musique = Mix_LoadMUS(p);
    if(musique==NULL){
        erreur_sdlm("Impossible de charger la musique",musique);
    }
    return musique;
}

//p est le chemin vers un fichier .wav
Mix_Chunk * loadChunk(char * p){
    chunk=Mix_LoadWAV(p);
    if(chunk==NULL){
        erreur_sdlm("Impossible de charger l effet sonore", NULL,chunk);
    }
}

void initialisationCanal(int canal){
    if(Mix_AllocateChannels(canal)!=canal){
        erreur_sdlm("Impossible de d initialiser le canal", musique,NULL);
    }
    Mix_Volume(canal,MIX_MAX_VOLUME/2);
    return;
}
```

To be continued ...