Charlotte Was Alone

Tristan Debeaune, Wendy Gervais, Quentin Huet

DÉROULÉ DU JEU

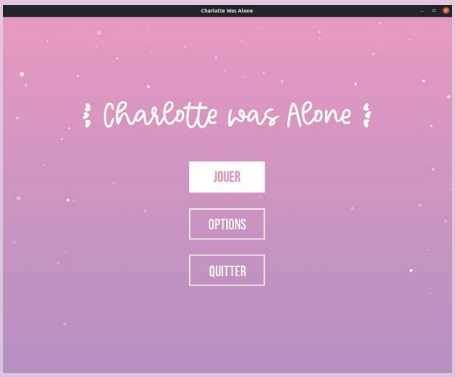
LANCEMENT DU JEU SUR LINUX

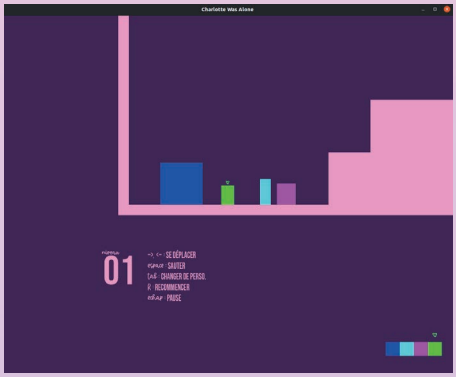
dans un terminal (à la racine de imac_thomas2) :
make jeu
bin/jeu.out

(si erreur, **make clean** puis répéter les deux commandes du dessus)

Note : il faut les librairies lSDL2, lGLU, lGL

JEU





Le jeu présente un menu principal permettant d'accéder au jeu, aux options, et de quitter avec les touches du clavier.

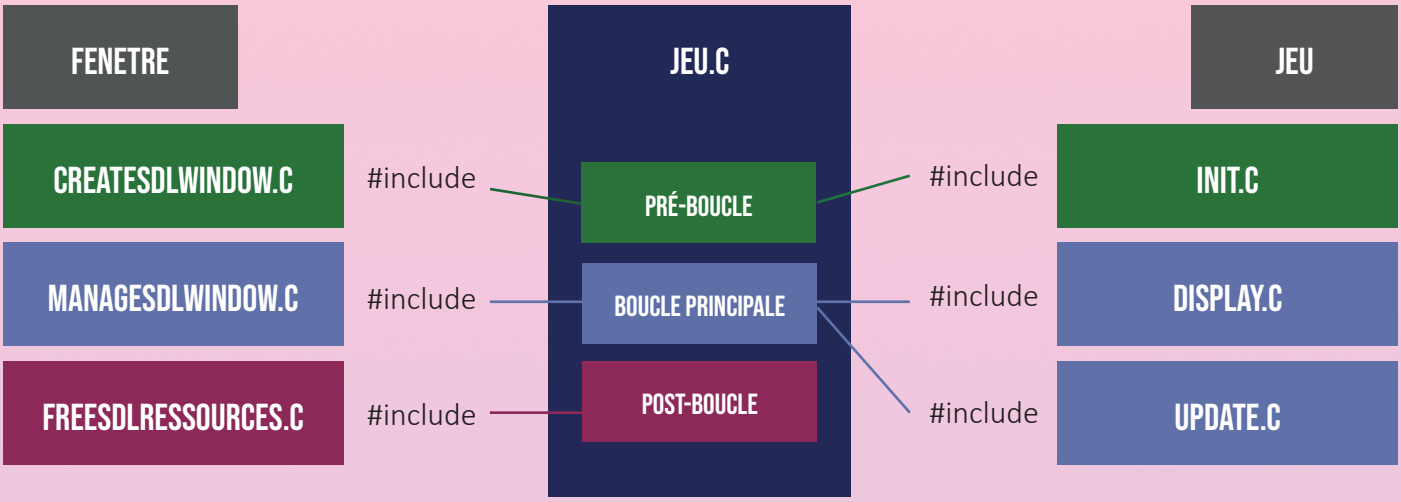
Les différents états sont gérés via une variable Gamestate (0 pour le menu, 1 pour le jeu, 2 pour l'écran de fin- le jeu est fini lorsqu'on a fini tous les niveaux et on peut alors revenir au menu).

En plus des déplacements de base, indiquées dans les premiers niveaux on peut mettre le jeu en pause avec Échap et recommencer le niveau avec R.

Nous sommes restés fidèles au gameplay d'origine, avec notre univers graphique et nos personnages !

RETOUR AU MENU

STRUCTURE DU CODE



ANIMATION.C

struct Animation, ListeAnimation
animations des plateformes

CAMERA.C

struct Camera
scrolling et recentrage de la caméra

COLLISIONS.C

fonctions de maths utiles pour les collisions

DESIGN.C

fonctions de dessins de rectangles, de rectangles texturés, numéro niveau et personnage actif

GESTIONSDL.C

redimensionnement de la fenêtre, fonction de chargement des textures

MAP.C

struct RectDecor/RectDecorFin, Map
création et dessin d'une map entière

MENUS.C

affichage et inputs liés aux menus (menu principal, pause, fin)

NIVEAU.C

struct Niveau, Niveaux
création et dessin des niveaux du jeu, gestion du passage entre niveaux

PERSO.C

struct Perso
création de personnage, gestion des déplacements, de la physique et des collisions, dessin personnage

PLAYER.C

struct Player
création du joueur (= équipe de personnages), gestion des inputs et changements, dessin personnages et overlay équipe en bas à droite

QUADTREE.C

struct QuadTree
création, parcours et positionnement d'un rectangle perso dans un quadtree

<SDL2/SDL.H>
<GL/GL.H>
<GL/GLU.H>
<STDLIB.H>
<STDIO.H>
<MATH.H>
<STRING.H>
<TIME.H>

LODEPNG.C

FAKESDLIMAGE.C

permettent de charger les textures

RÉPARTITION DU TRAVAIL

Commun

- JEU.C
- FENETRE
- JEU
- MAKEFILE
- DESIGN.C
- GESTIONSDL.C
- NIVEAU.C

Quentin

- PERSO.C
- PLAYER.C
- CAMERA.C

Tristan

- ANIMATION.C
- CAMERA.C
- QUADTREE.C
- COLLISIONS.C

Wendy

- MAP.C
- MENUS.C
- QUADTREE.C
- COLLISIONS.C

+ création des textures

Nous avons fait quelques sessions de code en présentiel, mais nous avons utilisé Git pour merge nos versions et travailler ensemble à distance.
Lien du repo : https://gitlab.com/rubykiara1712/imac_thomas2

Nous avons tous les trois travaillé sur Linux (Debian et Ubuntu).

DIFFICULTÉS MAJEURES RENCONTRÉES

NIVEAU.C

Il fallait faire l'allocation des champs du niveau, à faire avec malloc (sans cela on écrasait nos niveaux à chaque fois)

ANIMATION.C

Pour gérer les animations des plateformes, on a du créer des structures pour les combiner et les boucler. Cela a aussi créé des problèmes d'allocation mais au final la gestion est optimisée.

COLLISIONS.C

On a du séparer les collisions avec le décor fixe, le décor animé et les autres personnages, cela a été long pour obtenir des collisions cohérentes.

PERSO.C

La physique a été compliquée à modéliser mais une fois le modèle trouvé, avec la friction/gravité/accélération cela allait.
De même, la gestion des inputs du perso prenant en compte l'état actuel du perso a été longue à mettre en place mais le modèle est satisfaisant.

QUADTREE.C

On a aussi rencontré des problèmes d'allocation du quadtree quand bien même le principe et la mise en place de la structure étaient acquises.
On a encore des problèmes avec notre mode «debug» mais le jeu fonctionne avec les collisions du quadtree.

PISTES D'AMÉLIORATION

- Physique plus réaliste
- Collision des persos sur les rectangles animés (on n'a pas réussi à concilier les deux)
- De la musique/son
- Collectibles, objets interactifs (clés, eau, portes...)
- + de niveaux et personnages
- + de jeux sur les mécaniques

- une version Windows

- transporter le code en C++ pour utiliser classes et new
- mieux gérer l'allocation
- ré-séparer encore le code même s'il est déjà bien séparé
- QuadTree dynamique comprenant les personnages et les plateformes animées ?