Computer Science Project

Zakariyya Kurmally (2315839), Ihsaan Ramjanee (2315007), Nirvesh Bhagirath (2313628)

August 18, 2023

Contents

1 Rules of Cartesia

Cartesia is a 2D platformer. The aim to the game is to collect gold enough gold coins to unlock the chest at the bottom of the map. You will need to avoid traps, monsters and falling into the void.

- WASD or arrow keys can be used to move the character
- Player starts with 3 lives
- Falling into a trap or off the map results in death

2 Function description

To understand the working of the following functions, it will be important to know the contents of the struct GameState. The latter contains information needed for these functions to work properly. Here are its components and their description:

Texture2D tileset[NUM_OF_TEXTURES];	Array of textures
Texture2D bg1;	Background image
Particle particles[PARTICLES_COUNT];	Array of particles
int card[MAP_SIZE][MAP_SIZE];	Card array representing entities
int collisionMap[MAP _SIZE][MAP_SIZE];	Array indicating where collisions should occur
int playerPos[2];	Contains the coordinates of the player
int coinsCollected;	...
int remainingLife;	...
Sound sounds[2];	Array of sounds used
Direction direction;	Direction the player is facing
int playerMoving;	Indicates if the player is moving
Texture2D scarfy;	Spritesheet of player character
int framesCounter; int framesSpeed; int currentFrame; Rectangle frameRec;	Used to handle sprite animation

2.1 utils.c

Contains utility functions mostly for handling graphics and in calculations.

- _initGraphics (): Starts the window and audio devices
- _processImage (const char* filepath, Rectangle src Vector2 size): Loads image files from disk, resizes them, converts them to the proper data type and returns them
- _loadResources(GameState* currGameState): loads all the images and sounds from disk into memory. Makes use of _processImage
- _unloadResources(GameState* currGameState): unloads all the loaded images from memory
- _updateFrameInfo(GameState* currGameState): used to keep track of frame times for character animation

- _initParticles(GameState* currGameState): creates particles with random values for position and speed
- _displayParticles(GameState* currGameState): outputs the particles screen
- _updateParticles(GameState* currGameState): updates the position of particles based on the speed ans re-positions them if need be

2.2 core.c

Contains the main functions of the game. Used for displaying graphics and initialising the map.

- _init _card (GameState* currGameState): fills the map with entities like platforms, traps, trees. It is called before the main game loop
- _display _card (GameState* currGameState): loops through the array card, located inside currGameState, and displays the appropriate textures
- _handle _player (GameState* currGameState): handles player movement, collisions and player respawns

3 Problems encountered and how they were dealt with

3.1 VCS

When working on such a project, we would often need to switch back to previous versions of the code after a certain implementation went south. It was becoming a pain to remember what was changed and how to revert back. This was solved using git which allowed changes to be commited and easy reversion.

3.2 Unscalable function parameters

Many functions of Cartesia needed access to distinct information about the game state and textures. Early on, we noticed that passing them one by one would have been an absolute headache, especially when unloading all the resources since we make use of more than 10 of them. As such, we created the GameState struct which holds all the necessary information for the proper functioning of our functions.

4 Limitations

4.1 Not cross-platform (Yet)

C does not statically link with Raylib by default. As such, the executable is not distributable. However, shipping the DLL (for Windows) or the SO (for Linux) along with the compiled binary should solve this problem. Furthermore, Raylib has build scripts for Android.

4.2 No save points

Apart from loading in necessary game resources, Cartesia does not yet have functionality to save or load a game from a file on disk.

5 Individual Contributions

5.1 Bhagirath Nirvesh (25%)

Handled the particles system, sound and the display of textures.

5.2 Ihsaan Ramjanee (35%)

Map initialisation, collision detection and texture placement.

5.3 Zakariyya Kurmally (40%)

Graphics processing, player movement, library setup as well as structuring the project.