Game Project: Spaceship vs. Asteroids

Name: Viraj Verma

Student ID: H00473469

Introduction

The objective of this project was to develop a game utilising concepts that were taught in the former half of this semester regarding the C programming language. The game titled “Spaceship vs. Asteroids” is the game I developed for this undertaking. The raylib library, which offers an easy-to-use framework for making 2D graphic-based games in C, was used in the creation of this game. The aim of the game is to avoid approaching asteroids. This is accomplished by moving the spaceship horizontally in accordance with the x-position of the cursor. There is also a system programmed that increases the difficulty as well as ones that keep track of the score and remaining lives. The game design and key implementation features are described in this report. It also contains comprehensive guidelines on how to play the game.

Game Design

The basic gameplay loop consists of avoiding approaching asteroids, the player's primary objective is to live as long as possible. At programmed score intervals, the player gains extra lives and receives points for each row of asteroids avoided. The core gameplay comprises of the controlling of movement, spawning of asteroids and collision detection. Movement is controlled using the horizontal position of the cursor. Asteroids are spawned at regular intervals but the increased speed over time gives the illusion that the intervals gradually decrease.

Key Implementation Features

1) Headers

```
#include "raylib.h"
#include <stdlib.h>
#include <time.h>
```

raylib.h is the Raylib header file for the graphics

stdlib.h is the standard library

time.h is a library that allows the manipulation of time

2) Constants

```
#define SCREEN_WIDTH 800
#define SCREEN_HEIGHT 600
#define PLAYER_SPEED 5
#define ASTEROID_SPEED 3
#define MAX_ASTEROIDS 5
#define STAR_COUNT 100
#define LIFE_MESSAGE_DURATION 2.0f // 2 seconds
```

SCREEN_WIDTH and SCREEN_HEIGHT define the window size.

PLAYER_SPEED defines how fast the player's ship moves.

ASTEROID_SPEED defines the speed of the approaching asteroids

MAX_ASTEROIDS defines the maximum number of asteroids

STAR_COUNT defines the number of stars in the background

LIFE_MESSAGE_DURATION defines the duration of the +life message

3) Structures

```
typedef struct {
    Rectangle rect;
    Vector2 speed;
    bool active;
} Asteroid;

typedef struct {
    Vector2 position;
} Star;
```

Asteroid represents the asteroids position as well as its speed and whether it is active.

Star represents the random position of the stars in the background

4) ResetGame Function

The next part of the code resets all variables to their predefined original state.

5) Main Game Function

It begins by initialising the window size and FPS. Next it loads the textures for the spaceship and asteroids. Then the spaceship is initialised placing it in the centre

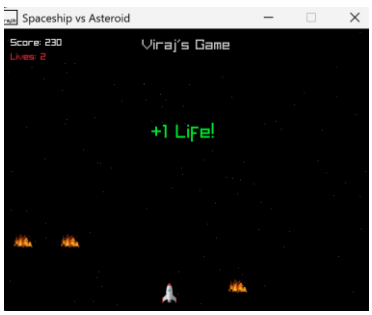
of the window and at the bottom. An array is created for asteroids which stores the asteroids' orientation. An array is also created for the stars to simulate outer space. The original variables for score, lives and difficulty are initialised.
This part of the code randomizes the placement of the asteroids and stars.

6) Core Game Loop

The main loop controls several operations. It checks the state of the game and offers the player to continue by pressing Enter. It also handles player movement which in this case is mouse controlled and is limited to the x-axis. The position of the asteroids is tracked in this part of the code including collisions with the player. It controls the speed at which asteroids are approaching the spaceship. The behaviour of the score and life counter is specified here. The next part of the code details the rendering of what is displayed on the screen. This includes the score, life counter, asteroids, spaceship, etc.

Instructions

1) Open the application file to load the game.
2) Press Enter to start playing.
3) Use the mouse to control the spaceship along the x-axis.
4) Guide the spaceship to avoid collisions with incoming asteroids.
5) 50 points will be gained for each row of asteroids dodged.
6) For every 200 points earned the life counter will increment by 1.
7) The percentage of score calculation will be decreased based on the challenge of increase speed of asteroids.
8) Colliding with an asteroid will reduce the life counter by one.
9) When the life counter hits 0 the game will end.
10) Press Enter to replay.



Conclusion

Spaceship vs. Asteroids is my initial foray into the world of 2d game design and programming. It is a graphics-based game developed using the Raylib graphics library and the C programming language. In theory the game is straightforward, but in execution it consists of many systems working in unison. The game is easy to learn but progressively gets more difficult, keeping the gameplay loop fresh. The game delivers an arcade-like experience with its short, fast-paced gameplays sessions that challenge the player's reflexes. The game lays out a basic yet solid foundation, one which can be built upon by adding more complex mechanics such as perhaps a power-up system. This project is an example of how competency in fundamental programming can evolve into exciting real-world applications.