Shooty Space Rocks

Technical Design Document

By Sam Ward

Contents

Game Overview

Game Concept

The player takes control of a UFO lost in an asteroid shield and must destroy all the asteroids to escape.

Platform

Shooty Space Rocks is being developed for Windows PC.

Development Environment

Game will be developed in c++ using the SDL Library

Third Party Tools

FMOD for audio

Software

Visual Studio Enterprise 2017

Game Play

Game flow

Game flow is achieved with the use of a state machine for the different game states.

Menus

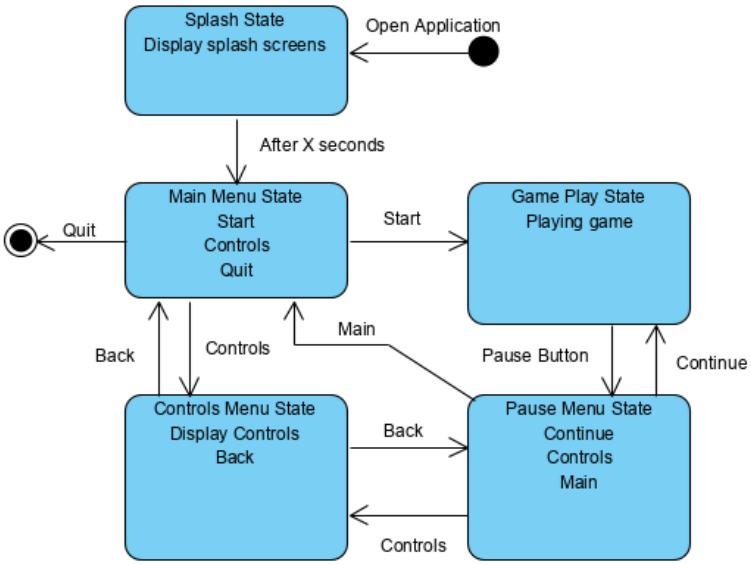
Main, Pause, Controls.

Playing

The game utilises continuous play progression. The game is played from start to finish in one motion with increasing difficulty.

Physics

The game does not require much physics so will use a basic physics model consisting of direction and acceleration vectors. The majority of the technical overhead will be in managing the collision detection.



State Machine Diagram for game flow

Asteroids

When an asteroid is split into two, the two new asteroids will continue a similar trajectory. Asteroids have velocity but not acceleration.

Player Ship

Has velocity, acceleration, and deceleration. E.g.:

- Player moves analog stick in direction, player ship accelerates in that direction until it reaches its max speed.
- Player releases analog stick, player ship continues in that direction for a time, slowing down to a stop.

Smooth rotation of the player ship is achieved by using the arctan function on the analog stick values

```
float angle = atan2((float)m_angleY, (float)m_angleX);
angle = RadiansToDegrees(angle);
```

Algorithm using current angle, target angle, and rotation speed in order to give the rotation a more realistic feel, rather than snapping to joystick position.

Enemy UFO
Share similar physics characteristics as the player.

Projectiles
Bullets and Rockets have fixed velocity.

Collisions
Makes use of vector Mathematics to determine accurate collision detection.

Collisions occur between the following objects:

- Player to Asteroid
- Player Bullet to Asteroid
- Player Bullet to Enemy
- Asteroid to Asteroid
- Enemy to Asteroid
- Enemy to Player
- Enemy Bullet to Player
- Enemy Bullet to Asteroid

Game Objects
Player Ship
Moves freely around x and y axis, rotates to face target direction.

Fires bullets and missiles.

Has health.

Asteroids
Follows a fixed line.

Has a size (Large, Medium, Small, Tiny) and corresponding level of damage (100%, 50%, 25%, 2%).

Has a speed. Bigger = slower, smaller = faster. Tiny asteroids will fade over time.

Background Objects
Non-interactive Stars, Planets, and Asteroids moving in the background to add to the atmosphere.

Bullets
Player primary weapon. These are fired at the asteroids to destroy them. Bullets fade over time.

Rockets
Player Secondary weapon. A rocket will destroy an entire asteroid in one shot. Has a different fire method, discussed under Player Actions.

Enemy UFO
Will sometimes spawn and follow a random path around the asteroid field. Will intermittently fire bullets at the players current position.

Player Actions

Player navigates the field using a combination of both analog sticks for direction and rotation.

Player fires bullets to destroy the asteroids or Enemy UFOs.

To fire a missile, the player must first load it by pressing a button, when a missile is loaded a crosshair appears on the screen and can be moved with the left analog stick. The missile is then fired with the fire button and travels toward the crosshair location. During this process the ability to rotate and fire the blaster will be disabled but can still move to avoid asteroids.

Victory Conditions

Once the player has destroyed a predetermined number of asteroids, the game is won. On Winning the player is returned to the main menu where they can choose to play again.

Artificial intelligence

Enemy UFOs

Enemy UFOs are aware of the players position and will fire bullets in that direction. The pathing will be random designed to mimic a player; however, the UFOs will not be aware of asteroids so will not move to avoid them.

User Interface

Play Screen

Game can be played in be fixed window size or full screen mode. The Game environment takes on the form of a borderless asteroid field in space through the use of wrap around edges.

Main Menu

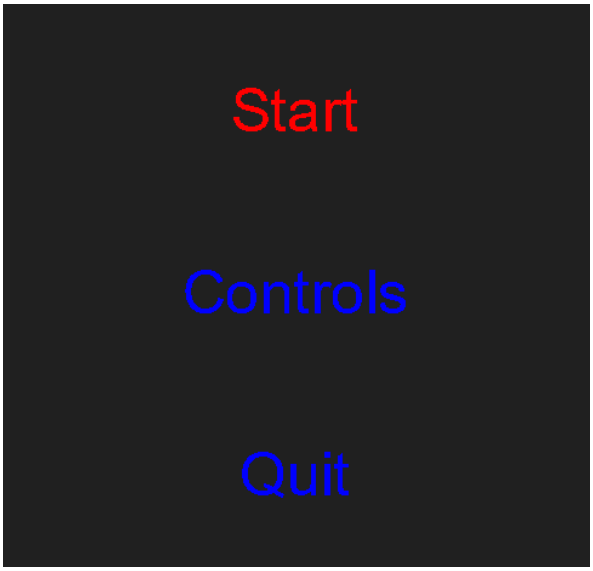
User scrolls through menu items with current option highlighted.

Pause Menu

When the player presses the pause button, the game is paused, and the pause menu overlays the game. Background animation is ceased but still visible.

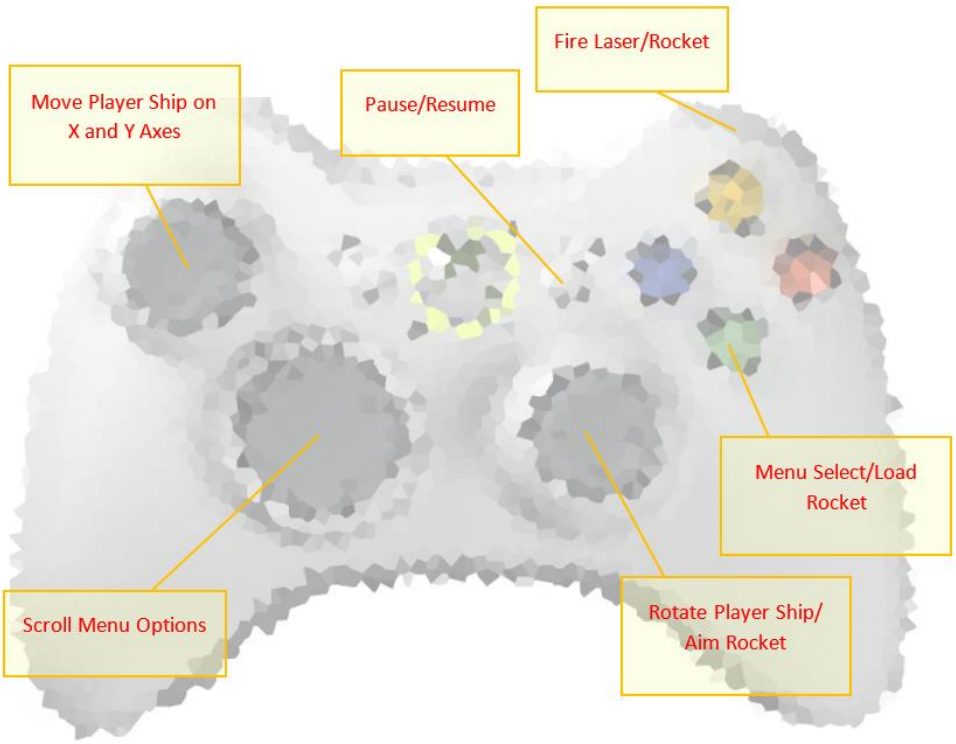
Controls Menu

Displays the controller keys



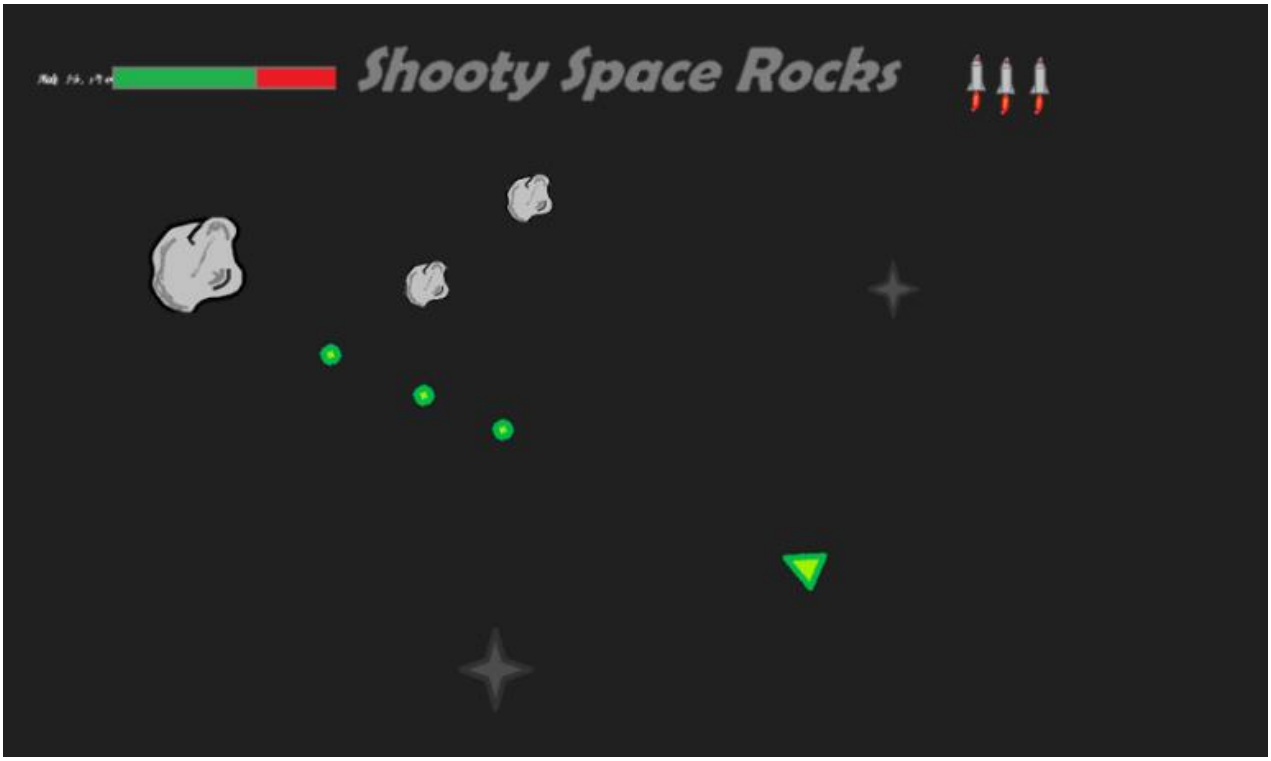
Game Controls

Shooty Space Rocks is played using a PC game controller with the following mapping.



In-Game HUD

Information display for ship health, number of missiles available, and number of asteroids destroyed (image example only).



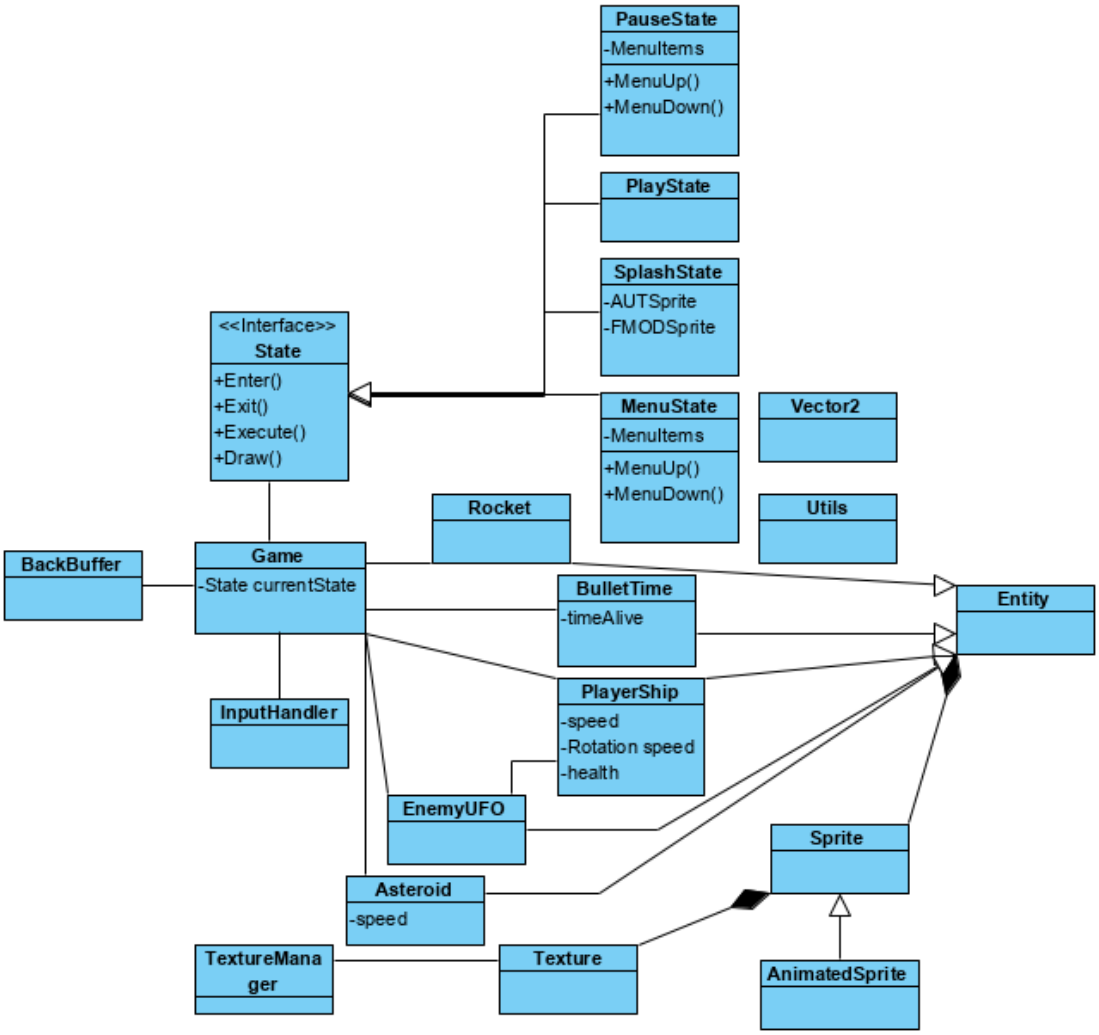
Win

When the player wins the game by destroying all the asteroids, the game will display a win message and then return to the main menu.

Code Overview

Object Structure

Game structure:



Coding Standards and Naming Schemes

Commenting

Functions and classes to be commented as necessary. Description above preferred but inline comments are acceptable.

Functions to be commented as necessary.

Code Craft

Large functions or classes to be broken down where appropriate.

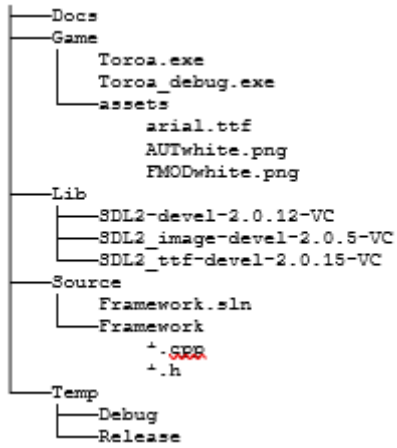
Promote code reuse, avoid replicating code by making modular functions/classes where possible.

Naming Conventions

Hungarian notation used where possible. P for pointer, m for member. Variable and function names should be self-explanatory.

Source Control

SVN to be used with regular commits, ensuring not to commit unnecessary files. Following file structure to be followed.



Debug features

Sprite axis can be toggled for checking directions

Functional testing

Difficulty level can be edited to avoid having to play through the initial stages of the game

Player invincibility

Graphics and Animation

Texture loading and Sprite creation using SDL

Game consists of animated and non-animated sprites as well as alpha blending

Game can be fixed window size or full screen mode

Formats

Sprite sheets and sprites in png format

Audio

FMOD audio for game sounds.

Formats

.wav format for audio

Acceptance Test Plan

Can the game be played?

Does it run on other machines?

Is it free of significant bugs?

Is the controller mapped correctly?

Can the player navigate all the menus?

Can the game be quit at any time?

Does the game have a goal?

Do the HUD objects react accordingly?

Can the game be lost?

Can the game be won?