# Assessment Task 4 – Retro Game Design Document

Asteroids go brr

A screenshot of a video game

Description automatically generated with medium confidence

Contents

[Assessment Task 4 – Retro Game Design Document 1](#_Toc136590683)

[Design Concept 3](#_Toc136590684)

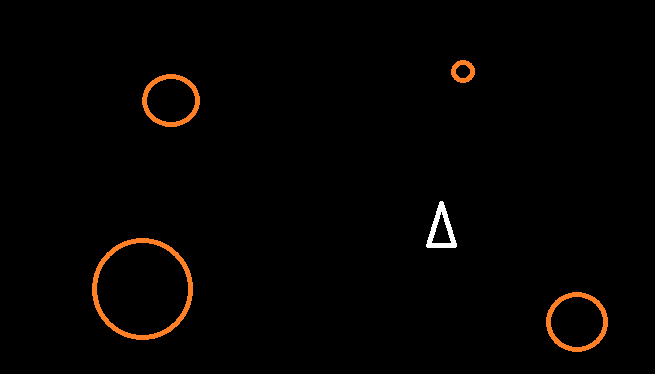
[Data structures and algorithms used. 4](#_Toc136590685)

[Design of Game 5](#_Toc136590686)

[Version Control 6](#_Toc136590687)

## Design Concept

The design of asteroids is simple. We just need a triangle representing the player that shoots at circular objects which break into smaller version of itself or gets destroyed if at its smallest.



In the retro game, the asteroids come in different shapes. This would be possible in Raylib using textures, which I didn’t use because I forgot, but also because I wanted to simplify the collision detections, so we just need to use some basic circle detections.

A picture containing black, text, font, sketch

Description automatically generated 

The ship is displayed using a triangle and shoots small little circles for bullets. The player starts with 3 extra lives, with extra lives being given every 10,000 points gained. The ship needs to move forward and keep that momentum whilst also being able to rotate freely so you can drift around and shoot asteroids. We also should add a brief period of immune time when getting hit in order to make things fairer to the player, otherwise getting hit would melt away all your lives in a snap.

## Data structures and algorithms used.

Raylib by default has a Vector2 struct built in, so I use that in order to track multiple aspects of the player, the bullets, and the asteroids. Specifically, we’ll be tracking their positions, and their directions using Vector2’s.  
Starting with includes, I needed to include raylib, and then all my headers that held my structs and variables, and finally, I create my structs.  
To initialize everything, I created a function called Initialize called only when loading the game that acts as default values when the game starts and resets.  
After initializing, the game is sent to the update and draw functions. Update handles all the inputs, collisions, and other calculations. Draw will draw everything onto the screen.  
A screen shot of a computer program

Description automatically generated with medium confidenceA screen shot of a computer program

Description automatically generated with low confidence  
Because we need to spawn more than 1 asteroid, when I define the Asteroids, I create them as arrays, and use many ‘for’ loops when interacting with them. I also do this with the players bullets.  
A screen shot of a computer code

Description automatically generated with low confidenceA screen shot of a computer code

Description automatically generated with low confidence

A picture containing text, screenshot, software, multimedia software

Description automatically generated

## Design of the Game

Asteroids is a simple game in terms of mechanics to code:

* A triangle that rotates, representing the player.
* 3 different sized circled to represent each of the different sized asteroids.
* The ability for the asteroids to be destroyed.
* Asteroids splitting upon being hit or being destroyed if small enough.
* A basic life and score counter, with extra lives being granted after gaining enough score.
* The player and asteroids wrapping around the screen, coming out on the opposite side, but the bullets don’t wrap around.
* Respawning asteroids so the game goes on.
* And finally, everything should be made of lines.

Raylib has functions that draw outlines of objects, so I can use that in order to draw everything.

## Version Control

Version control for this project was done within my repo for ‘Intro to CPP’

[History for AsteroidsGameTest - Sanzoz/cppRepo · GitHub](https://github.com/Sanzoz/cppRepo/commits/main/AsteroidsGameTest)

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated