# Assessment Task 4 – Retro Game Design Document

Asteroids go brr

A screenshot of a video game

Description automatically generated with medium confidence

Sorry its long ; x;

Contents

[Assessment Task 4 – Retro Game Design Document 1](#_Toc137220963)

[Design Concept 3](#_Toc137220964)

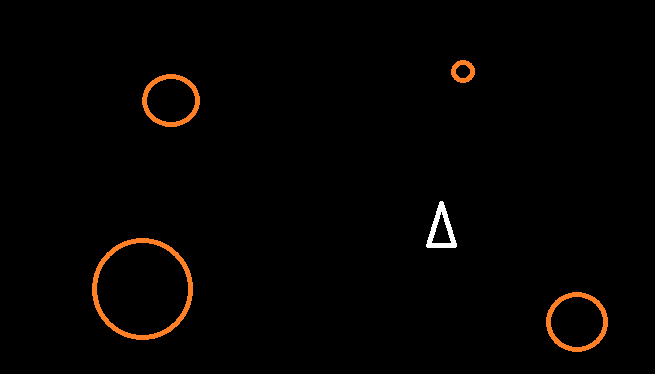
[Data structures and algorithms used. 4](#_Toc137220965)

[Design of the Game 5](#_Toc137220966)

[Version Control 7](#_Toc137220967)

## 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

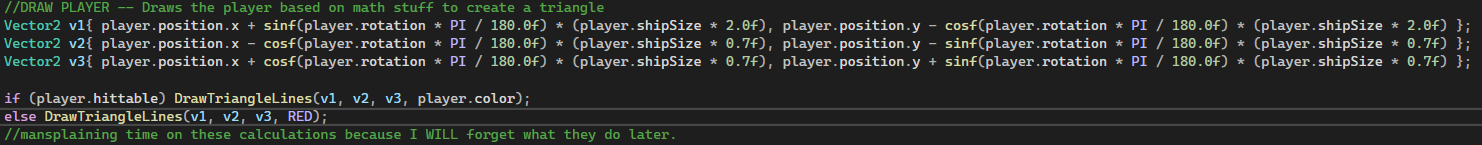
A picture containing text, screenshot, software, multimedia software

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 displayed as an outline.

Raylib has many draw functions for shapes, one of which being the triangle to display the player, but it takes in 3 vector2s, and needs to rotate correctly. Using some math that I adapted from [here](https://stackoverflow.com/questions/3837266/finding-the-points-of-a-triangle-after-rotation) (stack overflow), I can use the rotation variable to update each Vector2 that create the triangle, and then adding a slight offset to the calculations to shape the triangle by squeezing the sides, or extending the peak.  
  
The asteroids sizes and splitting are significantly less complex, and require me only to draw circles with different radius, and when they get hit to make deactivate them and spawn smaller asteroids if needed. To do this, I need to loop through all the bullets and see if they are active and colliding with an asteroid, and then split it or destroy it. After destroying 4 small asteroids, a big asteroid will spawn on the sides.  
A picture containing screenshot, text

Description automatically generated  
Splitting the asteroid is easy enough as I can it through a for loop that goes twice and spawns two different asteroids in different directions based on the bullets rotation when fired, and reversing the calculations to get the second asteroid going in the opposite direction, unless it’s the small asteroid, which just disappears.

The Lives and Score systems are super easy. When destroying an asteroid, I just need to increment the score and if getting hit by an asteroid, I need to remove a life.



Displaying these is even easier, as I just need to draw text for the score, and use the same calculations for displaying the player but with fixed rotations, and a position that moved over for each life you have, which is done using brackets and multiplying an offset value for its position.  
A screen shot of a computer

Description automatically generated with low confidence

The wrap around for the player and asteroids is done by detecting if they’re at the screen edge + / - their radius and an offset value to make sure you don’t see them pop out, then they get teleported to the opposite side, whereas the bullets just deactivate, and stop being drawn.

A picture containing text, screenshot, font

Description automatically generated

## 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