CE218 Report

# How to run the game:

Double click on the JAR file to launch the game.  
The JAR is located in asteroids/out/artifacts/Asteroids\_jar

# How to play the game:

* Use the arrow keys on the keyboard to move around.   
  Use left and right arrows to rotate the ship, use up arrow to move forward in the direction you are facing.
* Use spacebar to fire a bullet, holding space will fire multiple bullets, but you cannot turret fire while rotating.
* Acquire fairies that drop from enemy ships in order to gain another life and temporary invulnerability.
* Colliding with an enemy ship, enemy bullet or asteroid will cause you to lose a life. When you run out of lives, the game ends.
* Killing all enemies and asteroids will progress you to the next level, where the difficulty will increase.
* The high score is saved in the config folder. You can delete the save file to start your score at 0 again.
* Try to get the highest score possible and avoid dying.

# The game design:

This game is inspired the classical arcade game *Asteroids*. The main difference being that some powerups appear which grant the player an extra life and temporary invulnerability and there are different types of AI. Some additional artwork was also included (All artwork was created by student).

Several enemy AI are programmed into the game, with some being easy to avoid and quite slow, while others are very difficult to avoid.  
The goal of the game is avoid dying while killing enemies and asteroids to get a high score. The AI difficulty scales as the player score and level increases.

Software implementation  
The additions made to the code aim to follow correct OOP implementation with additional classes being added in the same way they were shown in the labs. A few additional classes were added in order to add more AI, and props to the background to allow a notion of depth. The structure of implementing these AI was also changed to allow them to store a target and reference the target location once attached to a GameObject. Best effort attempts were made to keep all fields in classes relevant to each class.

# Tuning

There were many factors to be tuned in the gameplay, a few will be listed below:

Player speed

I tuned the player speed to allow for faster dodging and evasiveness in response to making more difficult AI.   
  
Enemies  
**Enemy spawn rate**: I made the choice to have enemies spawn on a random timer. This was implemented by generating a random time since the game creation to spawn the first enemy. Once this enemy spawned, a new timer would begin some random n amount of seconds into the future which would decide when the next enemy spawns. This allows the game to be unpredictable between 1 playthrough and another.  
**Amount of enemies**: The above caused a problem then, where if the player did not kill an enemy before the next one spawned, they may start to become overwhelmed. Therefore, I added a limit to the number of enemies that can be on the screen at once. I did this by checking in the update loop, the amount of enemies, before deciding to add a new one to the list of GameObjects in the Game class.  
**Enemy AI/ difficulty**  
There were many options for creating AI for the enemy saucers. In the end I settled with some variety – the saucers appear in 2 sizes. Each size has a set speed and a radius/hitbox. Larger saucers tend to travel more slowly.   
For the AI, I tuned this so that any size saucer can have any AI, Some AI’s do not aim while others do.

The aiming feature was adjusted from the lab material. The initial aiming mechanism would check if the target was within a certain angle from the direction vector, then decide if it should shoot or not. The tighter this angle with the target, the more accurately the ship would aim at the target. It was also possible to have the saucer predict, based on the target’s velocity, where it would be in the next frame and aim toward where it thinks the player would be in future.  
Despite these features however, the enemy saucer still found it difficult to actually hit the player, especially when far away – this appeared to be down to the velocity vector of the bullet and not taking distance into account when firing at the player. To make the AI more accurate, I added a method to get a vector pointing at the target and applied this to the bullet’s velocity vector. This allowed the bullets to be generated with a velocity vector which was pointing directly at the player when they were generated.

In the end, the game has a variety of enemies with different sizes and AI, some are more difficult to avoid than others, and while the enemies do spawn with a random size and AI, they are scaled with some probability to ensure that most of the time, players will not encounter a difficult AI in the first 2 levels – this is mostly to ensure that anyone who is new can get used to the gameplay first and understand what to expect from the game.

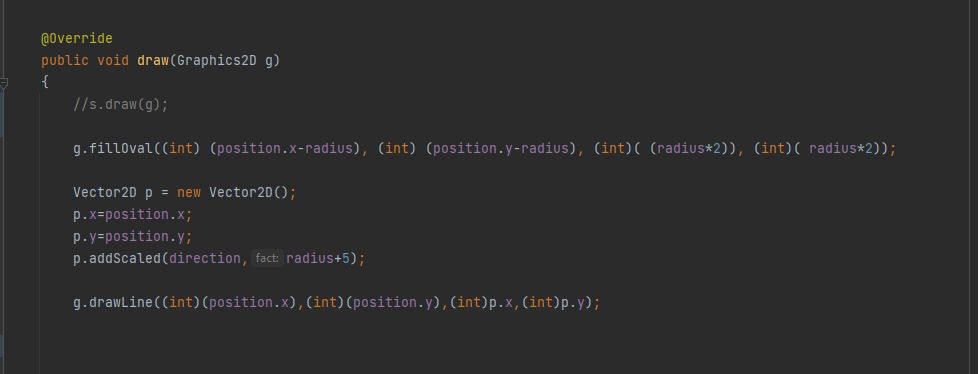
**Enemy and asteroid spawn position**  
Asteroids were programmed to only spawn on the edges of the screen – this avoids the scenario where the player is immediately killed on game start. Enemies also spawn in from the side of the screen and have a loud sound effect to warn the player of their arrival.

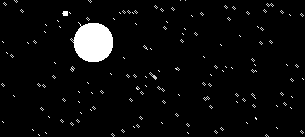
Score  
The score increments were adjusted with the following in mind: At every 10,000 points the player gains a new life. The player starts with 3 lives and will lose one if they are hit by an asteroid, enemy saucer or a bullet from an enemy.

The amount of points gained from killing more difficult enemies was increased, and the amount gained from hitting smaller asteroids was also increased. It was not made high enough that the player should be able to reach 10k points by the 2nd level however.   
  
The first 2 levels are designed to go fast (low number of asteroids, low chance to spawn hard enemy) so that new players may grasp controls without being interrupted or killed most of the time. However they will not get enough points to grant a new life in this time. A higher score is easier to achieve at higher levels as harder enemies/asteroids spawn but surviving becomes more difficult.  
This rewards players who manage to reach higher levels but still makes the game challenging so that it doesn’t feel repetitive and boring after the first 2 levels.  
  
The score is displayed on the edges of the screen in small text, along with other information. This was to ensure it does not cover any important game features.  
  
Procedural generation of background props and particle effects

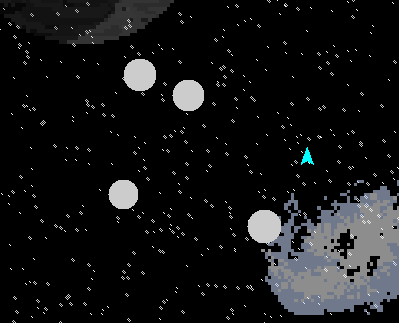
These features were tuned so that they are not overwhelming for the player to look at but also fit the general style/theme of the game. The palette generally fits the other artwork in the game and I dimmed the artwork in the background so it would not be too distracting to the player or cover any of the other important objects which were also drawn in white (asteroid/ship/fairy). This was done by setting the graphics object with a low composite value before drawing the scenery. To add depth, objects with a larger z value (distance from player) were drawn with a lower alpha value making them more transparent/dim.

Debugging  
Most of the debugging was done by repeatedly launching the application and observing changes. In some instances though, it was difficult to tell where issues were arising. Some sample code below shows one instance where changing the view of an object temporarily made debugging much easier:





ABOVE: Drawing the saucer as a circle with the size of its hitbox, and a line coming from the direction it is facing. This was very helpful for debugging enemy AI when targeting the player and shooting accurately at the player without jitter.

  
Above: Similar method used to debug asteroid hitbox issues, moving and increasing the size of the hitbox/scaling the sprite to align properly with the hitbox. As shown above, the hitbox properly covers the entire sprite.

Occasionally I did use debug mode to check the state/value of certain variables during the run as well. A lot of test code is commented in the final submission file, and I left prototypes/previous versions in comments as well (such as going from a static background image to an animated one).  
I worked with a personal Git repository in order to improve the game week by week, so was not afraid to make large changes at a time.

# Achievements

This project allowed me to learn about the mathematics used in video games and get a better understanding of vectors and angle rotations, which appear to be fundamental to video games.  
I found understanding the mathematics quite challenging at first, but with some support and time working with the project as well as seeing the outcomes of the changes I would make to the formulae, I started to get more comfortable using vector mathematics in the game and eventually managed to make some changes/features on my own.

I found adding animation/sprites was easier than expected and debugging issues with the target hitboxes and sprite sizes was not too difficult having built on basic shapes in java first and then simply applying an image over this radius. Reading the source code helped a lot with this.

I had hoped to add some networking functionality (allow players to compare scores online, or have a leaderboard), and I am aware of some small bugs/features that I would like to adjust given more time- such as a sound bug that occurs if 2 saucers are on the screen at once and 1 of them dies (the sound will stop completely). Also, despite making an effort to dim the artwork in the background, I still found that in some instances it may obscure the enemies or asteroids which makes the game a little more difficult to play. I wanted to add more props/maps and a menu with a selection screen but did not get time.  
Some small adjustments were left too late but would be easy to add if given more time, such as a small animation and sound effect when thrusting, and Fullscreen mode.

Overall, I am glad with the outcome and feel more comfortable working with and manipulating vector mathematics. I will in future take up more game development perhaps with an engine using the principles learned here as a basis.