IN2026 Games Technology Coursework

-Asteroids Game Development Documentation-

Author: Gabi Prefit

# Github repo

<https://github.com/gabi3737/games-tech-asteroids>

# Part 1

For Part 1 of this coursework, I decided to go for b): Implementing asteroids that split into smaller asteroids when hit.

In order to do this, we first have to ensure that there is a way for asteroids to not split apart endlessly. To this end, I introduced a “splitHealth” value within the Asteroid class in order to differentiate the asteroids that have been split apart once damaged from the ones that weren’t. I then initialised the value of splitHealth with a random integer between 0 and 2 using rand(), in order to ensure that not all asteroids are the same and that some may split apart more than others. Pictured below are setters and getters as well as the initialiser for the splitHealth value.

Text

Description automatically generatedGraphical user interface, website

Description automatically generated

Then, using splitHealth, I move over to the Asteroids class in order to have access to what happens when an asteroid leaves the scene. I first copy the code that populates the game with Asteroids in order to only replicate it when an asteroid leaves the scene. This obviously leads to an endless loop of asteroids spawning in after one has been destroyed, leaving no room for anyone to complete the game. Pictured below is the first snippet of code, before being altered to only split apart asteroids on death if they have a large enough splitHealth value.

Text

Description automatically generated

Here, there have been quite a few changes made. First, we get a temporary pointer to the destroyed asteroid. We use this pointer to get the position of the old asteroid and paste a new one right where it died. We only do this if the asteroid that was destroyed had a splitHealth of 1 or more. That way, there is a max limit to how many times an asteroid can split.

Text

Description automatically generated

In the image below, more changes have been made to how the new asteroids are spawned in. First, we make sure that their splitHealth value decreases by one. Then, we scale their size and hitbox accordingly based on how many times they’ve split. We make this change to the function that populates the scene with asteroids as well in order to ensure that asteroids that will not split apart are smaller than the ones that will. Then, we repeat this process once in order to make the destroyed asteroids split apart into two smaller versions of itself. Text

Description automatically generated

Graphical user interface, application

Description automatically generated

Different sized asteroids moving around after having defeated the player.

# Part 2

For Part 2 of this coursework, I decided to go for b): Implementing a power-up system.

In order to do this, first off, we had to implement a way to flag when the spaceship would be powered up. We do this by adding in a method to set an amount of time for the spaceship to be “powered up” and a way for us to track when the spaceship is powered up (through a Boolean). Picture below shows this in the Spaceship.h file:

Text

Description automatically generated

Then, we move on to creating the logic that supports this. In order to activate the power up, I’ve decided to use a score-based reward system that awards the played with the brief power up once they gain 500 points. Once they gain 500 more points, they regain the power up:

Text

Description automatically generated

Last, but definitely not least, I created the code that sets apart being “powered up” from being the regular old spaceship.

In the image below, I’ve set a condition so that once you are powered up, you shoot two more bullets. I’ve changed their trajectory slightly by adding or subtracting 1 from the spaceship\_heading calculation in order to make them shoot slightly angled from the original shot. I then spawn them in using this modified trajectory, and this leads to a three-way shot from the tip of the spaceship.

Text

Description automatically generated

A picture containing indoor, star, outdoor object, dark

Description automatically generated

In the image above you can see the three different trajectories that the bullets are taking once the ship is powered up.

# Part 3

For Part 3 of this coursework, I decided to go for b): Implementing an enemy spaceship

I did not have time to implement this feature, however, the way I would’ve implemented an enemy spaceship would have followed the following procedure:

1. Figure out what the player spaceship needs in order to traverse and shoot and make note of these methods/logic.
2. Create an extension of the player spaceship class called AISpaceship.
3. Give it information about the position of the player. Using the GameObject’s angle properties, I would figure out what direction the spaceship would need to point towards in order to be facing the player.
4. Make the AI spaceship shoot at the player sporadically when it is facing them.
5. If the spaceship is NOT facing the player at the moment, check (by creating a few “phantom” game objects ahead of the spaceship that act like triggers) if there are any asteroids in front of it. If yes, then also shoot towards them. If not, then move forward and try to locate the player.
6. If the AI spaceship collides with the player or with an asteroid, it dies.
7. The AI spaceship’s bullets would also have to be extended from their original class in order to have different collision checks.
8. If the AI spaceship’s bullets would collide with the player, destroy the player.
9. If all the asteroids in the scene are cleared but the AI spaceship isn’t, don’t progress to the next level. In order to progress, the player must also eliminate the AI spaceship.

Important notes:

It is extremely likely that using this sort of implementation would lead to the AI spaceship either accelerating into an asteroid before it is able to destroy it, it would get hit by an asteroid coming at it from behind (as it does not check behind itself), or it would start accelerating at an infinite pace once there are no more asteroids in the scene, leading for it to be extremely hard to be hit by the player with bullets. In order to change some of these things, it would be worth considering implementing some sort of limit on the AI spaceship’s speed, and a way for it to detect objects behind or around it that is also cost efficient. The AI spaceship should also have a different sprite (preferably at least a different colour) and so should its bullets. This is in order for it to be easily differentiated from the player spaceship.