Guardian of Earth

Jacob Mahoney

Owner

Vision Statement

For Android smartphone users of all ages who want a fun, quick-performing, and re-playable game, Guardian of Earth is an Android application that takes the formula of a basic spaceship shooter and adds depth to the objective of the game. Unlike many other spaceship shooters, Guardian of Earth possesses a unique simplistic art style making it an application that is not only visually satisfying, but is quick to pick and play.

Requirements

Actors

Users – Motivated users of the app with sufficient enough knowledge to get the app open, and learn about the game through the tutorial to the main menu.

Actor-Goal List

Actor	Goal
User	Navigates easily through main menu
	Play the game
	Control spaceship rotation with buttons in lower portion of screen
	Control spaceship laser firing with button in lower portion of screen
	Pause the game
	Press the pause button in upper portion of screen
	Resume the game
	Press the resume button that appears after user has paused game
	Change layout and size of user controls
	Go into settings portion of app to modify settings to user's liking

Product Backlog

Story ID	Story	Story Points	Priority	Status
S1	Allow user to navigate easily through main menu	3	2	Completed
S2	Allow user to visualize a loading screen that appears momentarily before the game is loaded	2	1	Completed
S3	Allow user to pause the game while in the middle of playing it by selecting the pause button on the game HUD	2	10	Not completed
S4	Allow user to resume the game after they have paused by selecting the play button	2	11	Not completed
S5	Allow user to modify settings like user controls to their liking in a separate settings screen navigated to from the main menu	3	13	Not completed

S6	Allow user save high scores	4	14	Not completed
S7	Allow user to, while playing game, to interact with the spaceship’s rotation by clicking buttons	7	3	Completed
S8	Allow user to, while playing game, to shoot lasers out of the spaceship by clicking another button on the screen	8	5	Completed
S9	Allow user to, while playing the game, visualize a HUD (heads-up display) including, for example: scores, lives left, pause/play, etc.	5	9	Not completed
S10	Allow user to, while playing the game, for each wave, to have a set, random number of meteors following towards that they have to destroy to not lose the game	12	4	Completed
S11	Allow user to, while playing the game, to also be able to visualize non-interactive game elements like the earth which lies at the bottom of the screen	6	7	Not completed
S12	Allow user to, while playing the game, to progress through a set amount of waves and each wave gets progressively harder	8	6	Not completed
S13	Allow user to, when they play the game for the first time, to walk through a tutorial which shows them how to operate the spaceship in rotation and firing, and how to interact with certain HUD elements	4	12	Not completed
S14	Allow user to, while playing the game, have it so when they destroy meteors, they earn points that goes towards their total score which is visible on the HUD	4	8	Not completed

Sprint #1

Sprint Backlog

Story ID	Story / Task	Estimated Hours	Actual Hours
S2	Setup android studio project and startup activities	1	3
	Determine if game needs loading screen to show during the loading of game assets	1	1
S1	Design main menu ui	1	0.5
	Code main menu ui	2	2
	Test main menu ui and app startup	2	1
S7	Determine best algorithm for game loop	1	2
	Code game loop	2	3
	Test game loop with placeholder animations	2	5

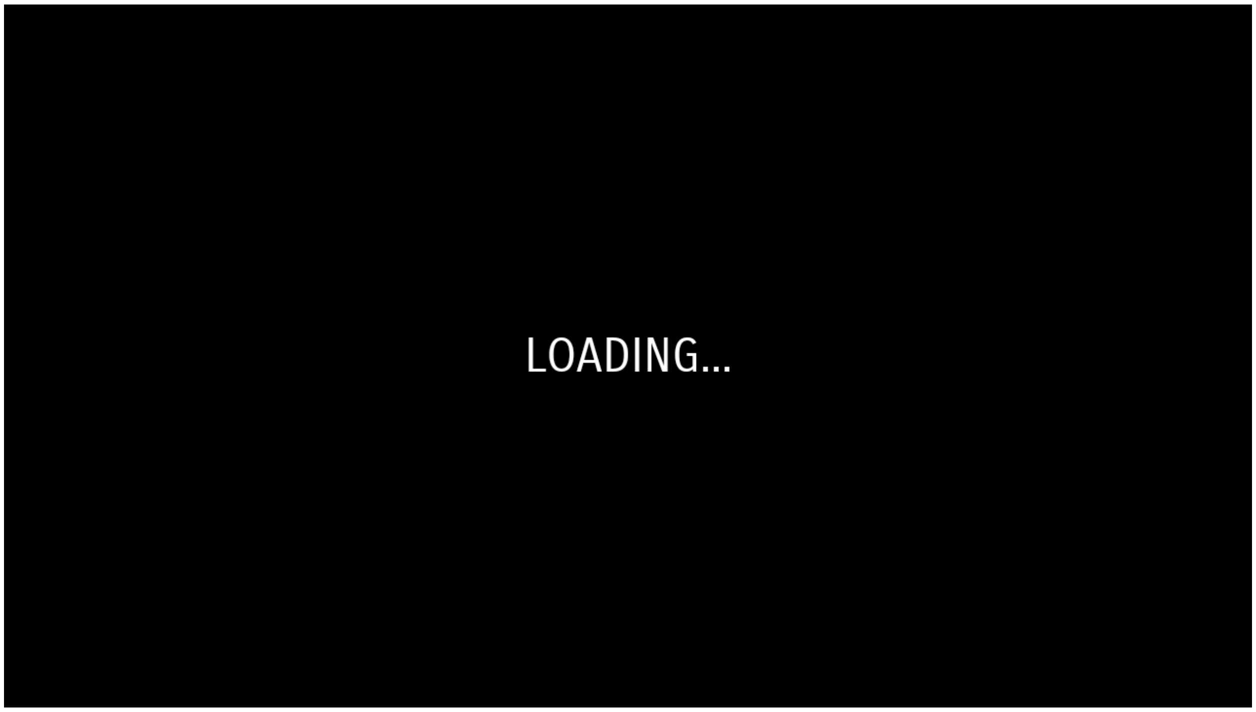
Retrospective

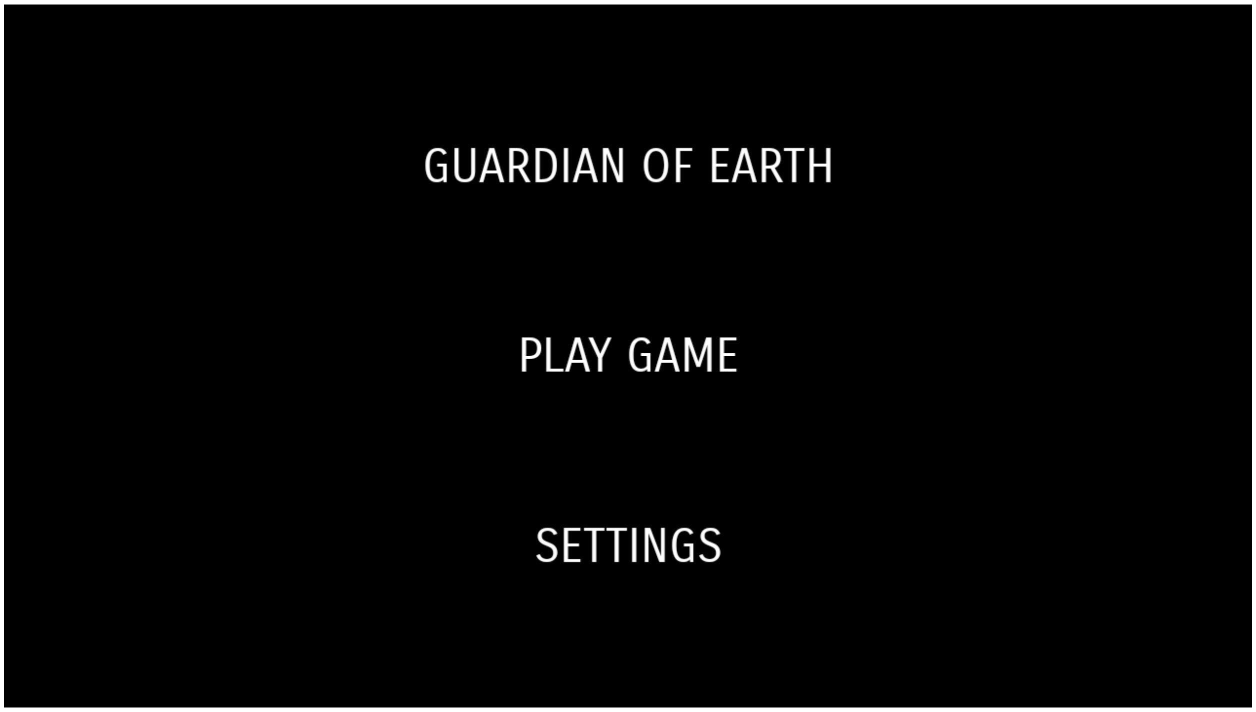
I thought my first iteration under the agile methodology went pretty well. I planned to do several tasks for story 1 and story 2. What could have gone better though, was in the beginning of the iteration, I definitely underestimated the learning curve of developing an android app, and simply using Android Studio. So I plan to, in the future, take that into account when planning out my tasks for an upcoming sprint. I also experienced some difficulty in testing game loop, which resulted in an essential complexity rather than an accidental one. That is because, testing the game loop is inherent to the problem, because I don't have all the game elements in place to test the game loop. So I guess, what I didn't realize off the bat, is that I will need to test the game loop as I continue to add game elements to the scene to maintain a good performance for the game. So in the future, I will try to plan out and notice those essential complexities beforehand so I don't waste time on trying to solve something that can't be solved yet. Overall, I feel the sprint went pretty well.

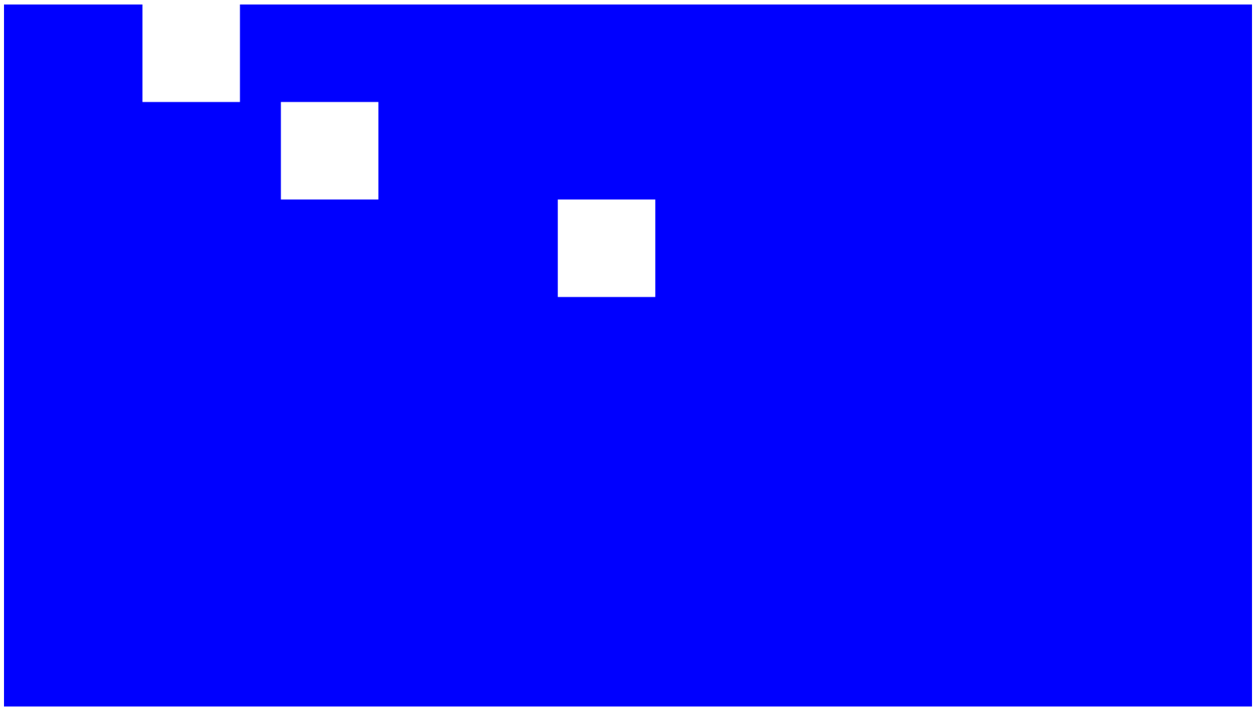
Project Velocity

The project velocity after this sprint is 5 because I completed story 1 and story 2.

Review


LOADING...





Sprint #2

Sprint Backlog

Story ID	Story / Task	Estimated Hours	Actual Hours
S7	Put placeholder objects into game scene: spaceship, earth, buttons for rotating, button for firing	4	2
	Add more definition to game object interface	2	1
	Capture button touches for rotating spaceship	3	4
	Rig up buttons to rotate the spaceship when they are tapped	4	5

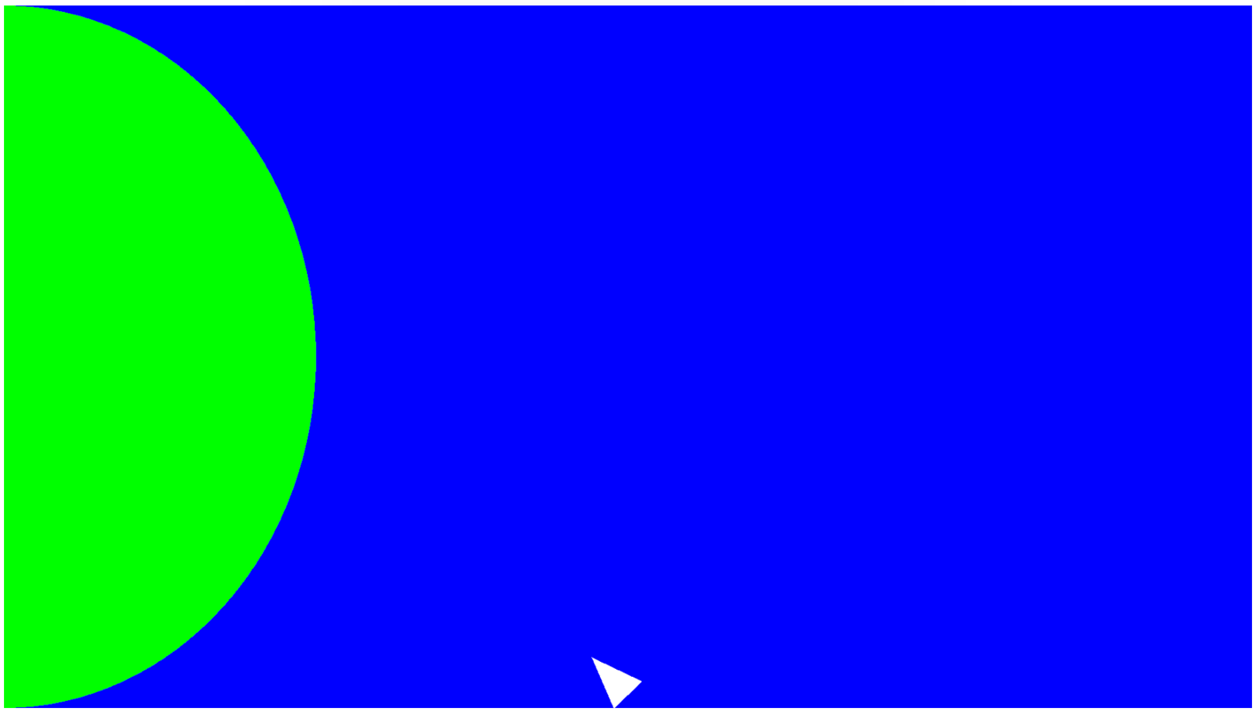
Retrospective

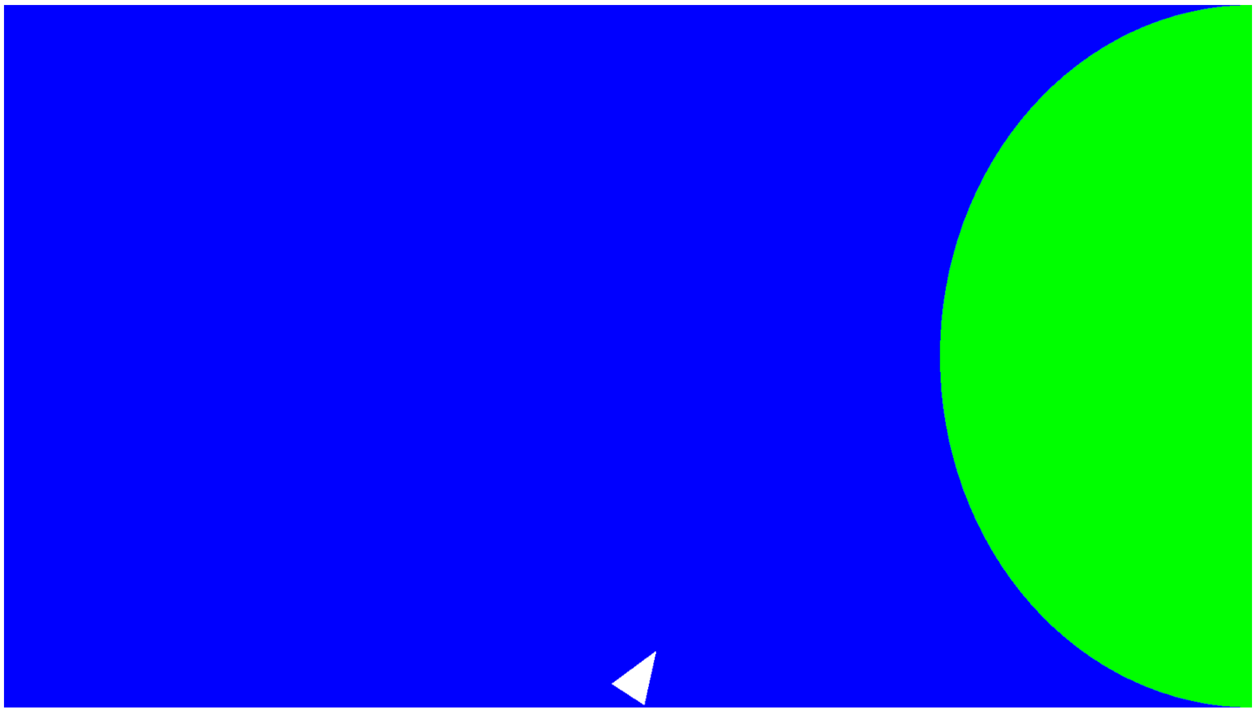
My second iteration, went pretty well. I did encounter some bugs though, which did slow me down a bit in the beginning. Specifically, dealing with the game loop itself, I spotted a bug that I should've noticed sooner, but I didn't. The bug was that my game loop drastically slowed down as time went on. After many, many hours of searching online for solutions, and eventually asking for help from an online programming community, I found the source of my bug myself. It was a few lines missing from each GameObject's update function. The main problem with this past iteration was that, even with that main bug that slowed me down a bunch, I feel I still didn't accomplish much from this past iteration. Because after I squashed the bug that caused extreme lag with my game, the development for rigging user presses to rotate the spaceship turned out to be quite easy. I guess it's just because I haven't had much experience at all developing Android apps, and specifically developing a game before, that I don't know exactly how much to plan to code for each iteration.

Project Velocity

The project velocity after this sprint is 7 because I completed story 7.

Review





Sprint #3

Sprint Backlog

Story ID	Story / Task	Estimated Hours	Actual Hours
S8	Add button for firing lasers from spaceship	2	3
	Capture button press for firing lasers out of spaceship	4	2
	Create a laser emitter class which will handle spawning of multiple lasers	8	12
	Rig the fire button up to the laser emitter class which will spawn lasers from the end of the spaceship	4	4
S10	Create structure class for handling the meteors for each wave	2	2

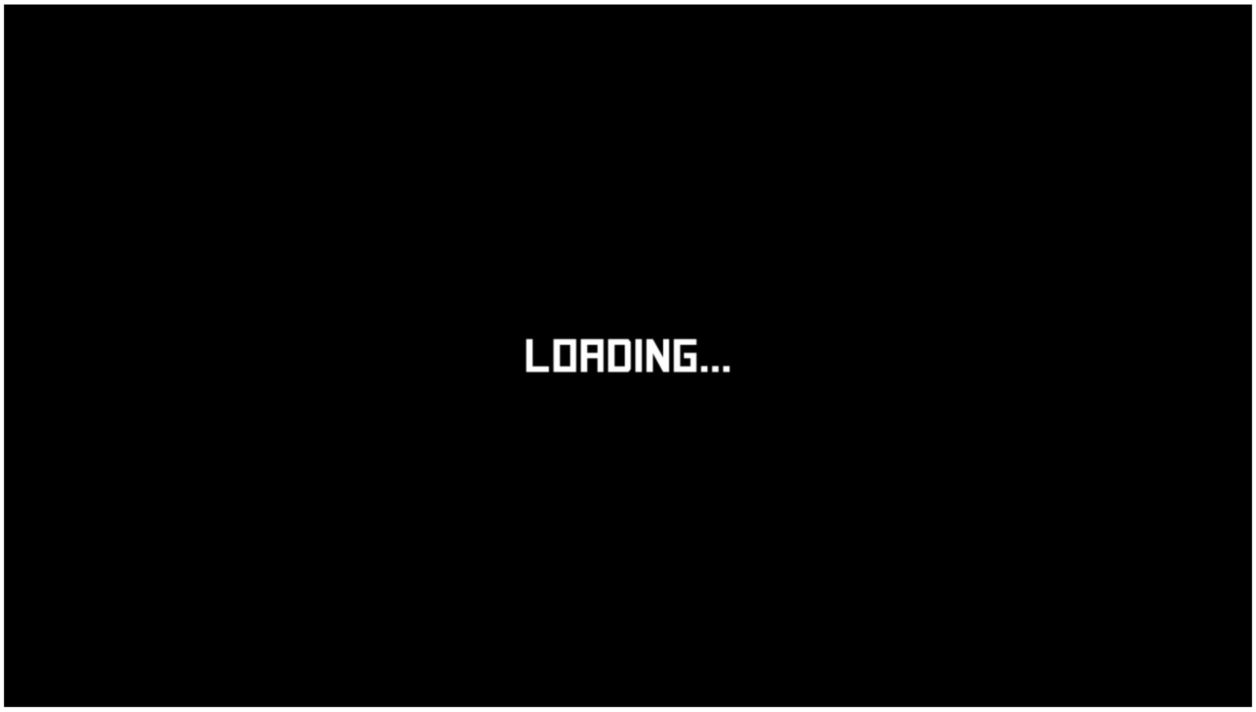
Retrospective

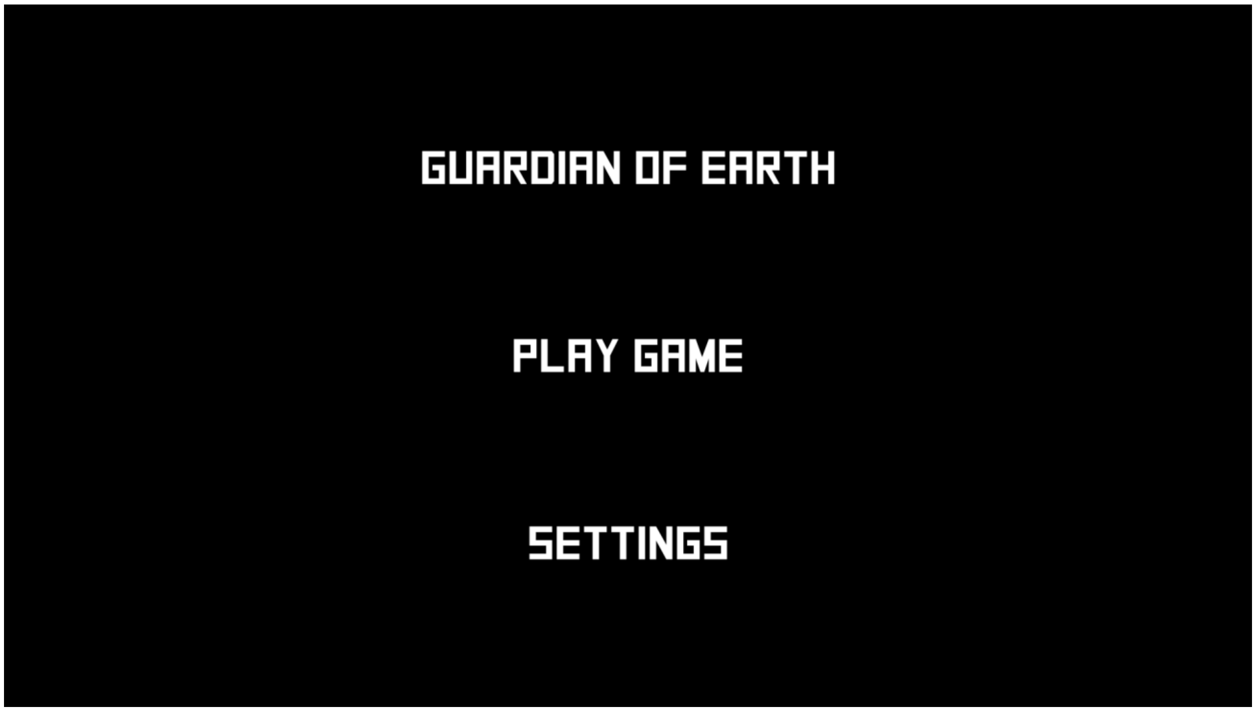
This sprint was definitely a roller coaster. I went from thinking I wasn't going to complete the things I had planned out in the spring backlog, to even adding another task to do from a different story because I had a little extra time. The specific task I got held up on was making the function to determine the new nose position of the spaceship after it had been rotated a bit on the canvas. I had to figure out this new position, because that was where the lasers of the spaceship would spawn when the user wished to fire. As I finally finished up, and figured out how to do that I began thinking about the overall structure, design, and layout of the app and how I could use abstract classes to perform similar things. This is what led me to create a ParticleEmitter class which handles the movement of all particles on the screen, whether they are spaceship lasers, or in the near future, meteors. It also led me to create a whole new wrapper class called, GameHolder, name pending, which will handle all the game objects; functions the game performs, such as firing spaceship lasers; and game constants, such as the different values I will need stored for each wave of the game.

Project Velocity

The project velocity after this sprint is 8 because I completed story 8.

Review





Sprint #4

Sprint Backlog

Story ID	Story / Task	Estimated Hours	Actual Hours
S10	Finish class for handling the meteors that will take in parameters to specify the rate, speed, and number of meteors	8	9
	Somehow pass a signal or event back to the parent class from the MeteorContainer class which signals the end of a wave	10	8
	Make function for determining collisions between spaceship lasers and the meteors	6	3
	Test MeteorContainer using different parameters acting as the different waves the game will progress through	8	5
S12	Start design of Wave class which holds a MeteorShower, which also prints out the wave name at the beginning of the wave	6	6
	Make Wave class observed by GameHolder	2	1

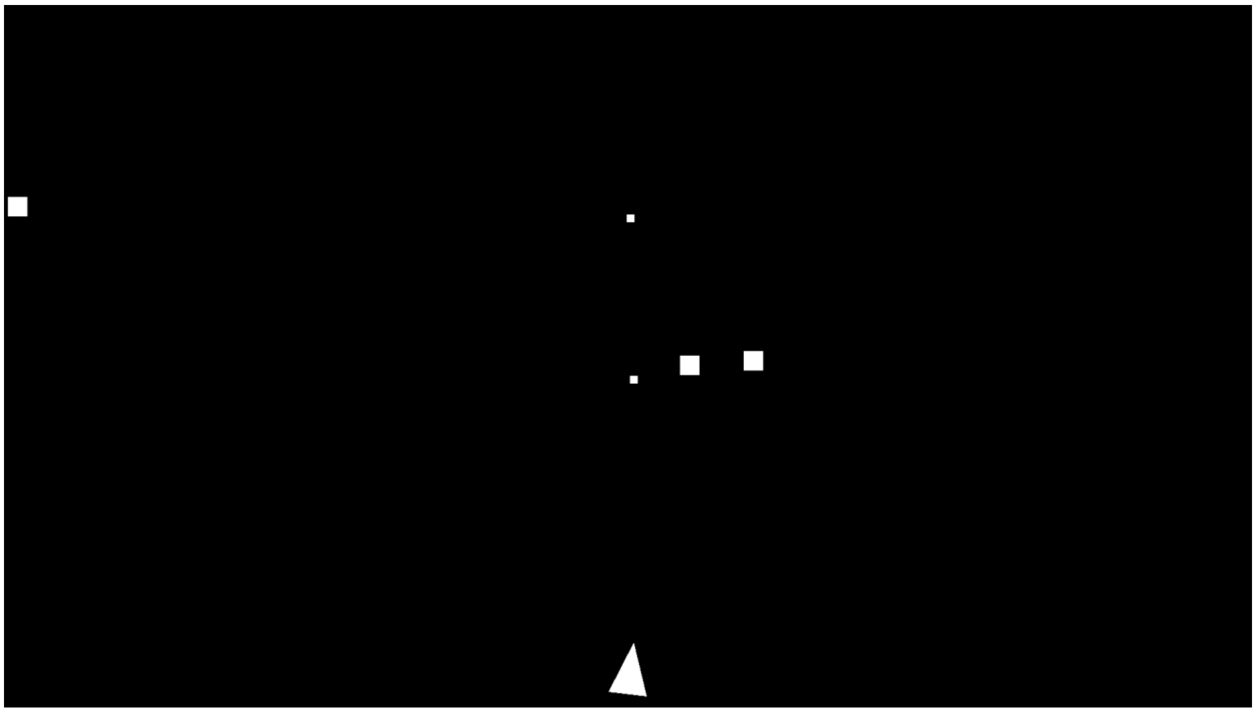
Retrospective

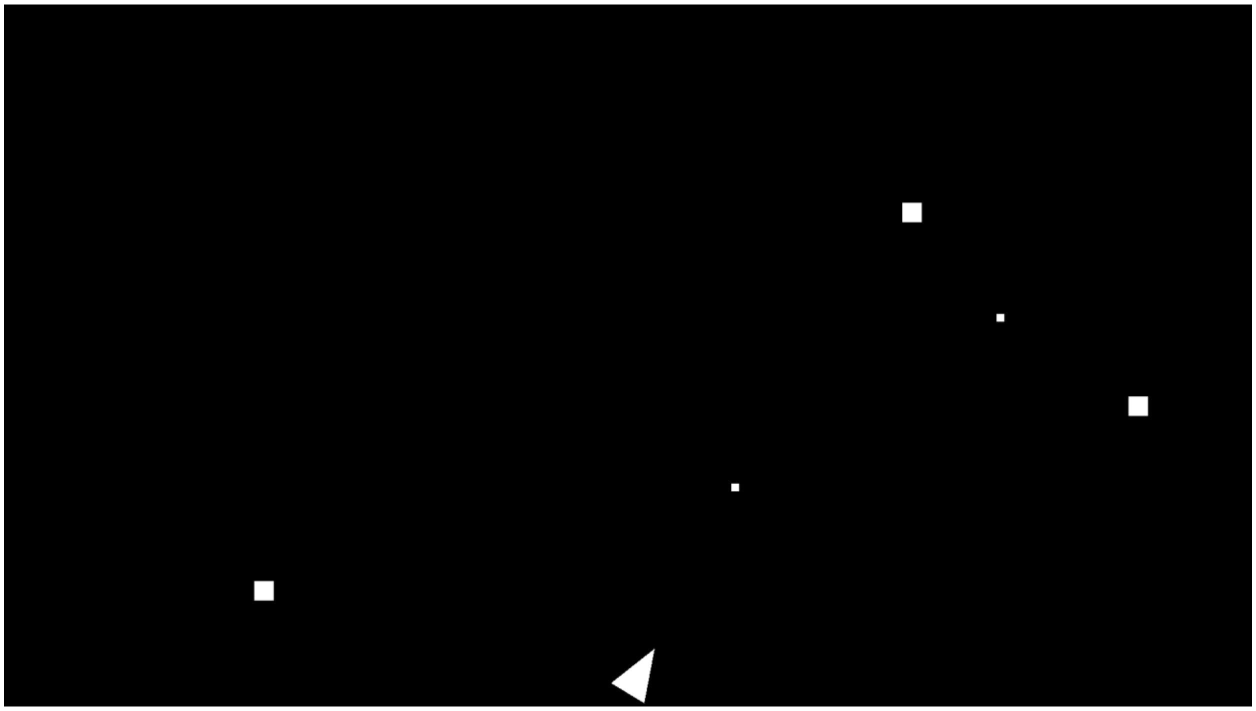
This sprint, for the most part went decently well. It started off kind of slow because this whole process of developing an android app, and a game at that, is still really new to me, and I was simply unsure on how I was going to implement all the ideas I had in my head in a design that made sense, was modular, and followed all the rules of an object-orientated design philosophy. I then started to pick it up about halfway through the sprint when I began to get a grasp on what I needed to code and how to implement a lot of the stuff I wanted the game to have. I began to seek out solutions in the form of design patterns, and this time it came in the form of the Observer design pattern. Since I knew the ParticleEmitter class was fit to determine collisions between lasers and meteors, I had to be able to signal or alert the GameHolder class of things like those collisions, and also when meteors have actually reached the bottom of the screen. It was a no-brainer for me, I had to use an Observer design pattern, making GameHolder observe ParticleEmitter. Also I realized that GameHolder could also observe each Wave class so that each Wave object can signal GameHolder when all of its meteors in the MeteorShower have been emitted.

Project Velocity

The project velocity after this sprint is 8 because I completed story 12.

Review





Sprint #5

Sprint Backlog

Story ID	Story / Task	Estimated Hours	Actual Hours
S12	Finish up and test Wave class throwing in different parameters	2	
	Initialize all waves in GameHolder class	3	
	Create GameStatus enum in GameHolder class and determine effective way of monitoring the game status at all times based on observer notices from ParticleEmitter and Wave	12	
S11	Make GameObject for Earth lying at the bottom of the screen using either placeholder object or actual image drawn on canvas	4	