Name: Bonny Chen

Daily Work Log

Date/Time	Things done	Things to do (Before Sprint 1)
June 6th, 5 hours	- Write up - Research - Set up the folders where I'll be coding my game - Learnt about LibGDX library (the glx lib and the config.set lib) - Created the application/game panel (blank black screen)	- Add keyboard keyboard input, test keyboard input - Create GameKeys, GameInput, GameScreenManager, PlayScreen, GameScreen classes
June 7th 4 hours	- Added keyboard input, worked on GameKeys and GameInput class - Added GameScreenManager, GameScreen, and PlayScreen class - Was able to get screen to update - Created Player and SpaceObject class	- Work on Player and SpaceObject class - Add the attributes position, vector, direction, speed, rotation speed, size, shape - Need to find the angles of the ship to draw the player, you find it with radians - Calculations for speed so ship can move
June 8th 4.5 hours	- Worked on the player class - Worked on creating the outline of the ship with ShapeRenderer and calculated the points for it to be drawn - Worked on the movement and speed methods - Created screen wrap method in SpaceObject (superclass) which will wrap every SpaceObject around the screen - Ship can now move	- Centre the ship - Add ship rockets - Add ship shooting mechanic - Create asteroids - Fix camera

Date/Time	Things done	Things to do (Before Sprint 2)
June 9th 1 hour	- Played around with different speeds, velocity, acceleration, deceleration, rotation speed - Centred the ship - Resized the game to fix camera	- Add the acceleration rocket - Add ship shooting mechanic - Create asteroids
June 10th 2 hours	- Added the acceleration rocket, now when you press the up key, the rockets will come out of the ship - Added the draw rocket variables to the methods so when the ship is drawn, the rocket is drawn too	- The rocket leaves a trail on the screen, need to fix that - Add ship shooting mechanic - Create asteroids

Name: Bonny Chen

	- Played around with different rocket sizes, ship sizes	
June 11th 2.5 hours	- Created the Bullet class - Added ship shooting method - Put the bullets in an array list - Connected the Bullet class to the Player class so the ship can shoot - When you press space, 4 bullets should shoot out at a time - Encountered an error in the PlaySceen that I could not figure out so I couldn't run my code?!?!?!	- Create asteroids - Fix the rocket error and the PlayScreen error
June 12th 2.5 hours	- Moved on to create asteroids because the error doesn't effect that - Tested asteroids and it spawns	- Still could not fix the errors from before - Need desperate help and I want to cry - Add particles - Work on collisions - Add levels
June 13th 5 hours	- Mr. Lee fixed the PlayerScreen error, the shooting now works - I fixed the rocket trail error, turns out I forgot an if statement in front of a loop - Worked on collisions, which is added to PlayScreen - The collisions are Bullet-Asteroid, and Player-Asteroid - Created the explosion effect after player hit asteroid - Created the Particles class so particles can spawn when you shoot an asteroid - Added levels, once you shoot all the asteroids, it will move onto the next level automatically;	- Add scoring - Add extra lives
June 14th 5 hours	- Tried to add font ?? - Created the scoring methods - Spent a long time figuring out how to add score font, but it says it can't find the file path?? - Added extra lives, there are 3 lives but there's no restart game yet	- Need to figure out how to put the font on there - Fix score increments

Name: Bonny Chen

Date/Time	Things done	Things to do (Before Final Code is due)
June 15th 1.5 hours	- Font now appears - Created MenuScreen - When the game launches, the MenuScreen is the first thing you see - Fixed score increment - Aligned score and lives	- Add MenuScreen options - Add saucers
June 16th 2 hours	- When a player loses all 3 lives, it will return to menu - Added menu options as play or quit	- Create the FlyingSaucers class
June 17th 3 hours	- Created the FlyingSaucers class, implemented all the methods needed	- Implement FlyingSaucers into the actual game - Make the saucers shoot back at the player - Add collision between the saucers and other objects
June 18th 4.5 hours	- Implemented the FlyingSaucers in PlayerScreen - Added collision between the saucer and asteroids, saucer and player, saucer bullets and player etc. - The special feature of my game is that the saucers can shoot back at the player, and there are different sized saucers - Flying saucers are now a part of the game	- Create GameOverScreen - Add it to the end screen when the player loses all it's lives
June 19th 3 hours	- Create GameOverScreen - Now when a player dies, it will go to game over and will be given the option to play again or quit - The text is randomly placed and not centred on the screen	- Fix the layout of the MenuScreen and GameoverScreen
June 20th 3 hours	- Worked on MenuScreen and GameOverScreen layout, learned about GlyphLayout to centre the text	- Play the game a few times and see what could be fixed
June 21st 3 hours	- Debug and fix issues with the game, screens not switching properly, player can still shoot after dying, minor drawing errors were all fixed	

	- Tried to add animations to the MenuScreen but I could not get it to work so I removed it :(	
June 22nd	- Code Submission	- Complete the reflection and presentation!

REFLECTION

Through this project, I learned more about game development and what it's like to partially code a game from scratch. I learned more about how the classes work together to create the game and the logic behind game development and design. I also learned the basics in coding a game where instead of using double variables, you use float. You use float variables in games because it uses less memory than double, which increases game optimization and speed. I had to learn how things worked in the LibGDX library and learning a new library has been fun and challenging. I learned mostly from the LibGDX wiki page because everything about the libraries is on there. I also watched YouTube for tutorials to learn how to set up LibGDX. The most challenging part working with the LibGDX library was probably learning sprites, fonts, and graphics. A lot of the other things in the library were easier to understand and were similar to the normal basic java library. The graphics were confusing because the graphics libraries in LibGDX have a lot of different features, and I need to learn what they all do in order to create my own. Throughout coding my game, I also learned that a lot of methods in the classes are repetitive. Every class has update(), initialize() cause that's what gets the game to work. Each class under entities will be similar to each other, each class under screens will be similar to each other, because the subclasses extend from the superclasses. Every game also needs delta time variables to get the time every frame per second that holds the time between now and last call in milliseconds. I experienced problems with getting the Player and Bullet class to connect, getting the font to display on the screen, errors with the player, draw() method errors, shapes rendering wrong, loops looping wrong and other minor errors. One of the drawing errors I encountered was the thruster not drawing in the right spot and it left a trail on the screen. It took a long time of looking over my code to figure out the error was because my if statement was in the wrong place. It was also really annoying when the font file path could not be found and when the Player and Bullet classes could not connect. The solution to such a problem was so simple but during the moment I had the error, I felt panicked because I could not understand what was wrong with my code. Doing the maths to render the shapes was also pretty irritating. I could've just used sprites for graphics instead of doing maths to draw the shapes. I played with different numbers so I could get the shapes to look how I wanted them to. It was really hard to keep track of so many classes and methods. When I create a new class, that class has to be implemented into other classes. Making changes in one class needs you to carry over the change into all the classes connected to that class. Sometimes I forget to carry over and implement the changes I made in one class to all the other classes, and that creates error and

confusion for me. Then after the error message pops up, I have to retrace my steps and figure out which class I missed to add code into. This was frustrating because I had to keep undoing the code to figure out what I missed. I would undo and redo the code I added to see what is wrong and run the game with small portions of added code at a time to ensure it will keep working. A lot of problems I encountered throughout this project were solvable through trial and error, retracing my steps, reading about the issue online, watching videos and asking the teacher or friends for help. I think my time management for this project was pretty decent as I was able to produce a working functional game at the end. It may have not been fully complete but it is playable and fun. If I had more time for this project, I think I would've made a different and less generic game. I picked such a generic game to make because with the limited time I had. I'd rather learn how to do everything first before creating a new original game. I feel like I really learned a lot about game development in these past weeks and my next steps in the future is to hopefully create a game on Unity and learn how to do animations and graphics. This is the first game I've ever made and I will definitely take what I learned throughout this course and project into the future.