Blake Freer

Date	Duration (hours)	Description of Completed work
Week 1		
Tuesday May 26	4	Created the Vector2, Vector3 and Mathf classes. I had made a different Vector2 class (unused, renamed to Vector2_v1), but recreated it to resemble methods used in Unity. The Vector3 class will no longer be used, as I am planning on making a 2D engine instead of 3D.
Thursday May 28	6	Created the GameObject and Component parent classes, and set up the hierarchy structure of the game. Implemented a 60FPS game loop that recursively updates all of the GameObjects in the scene. Began to create specialized components for rendering circles. A separate object will be needed to order and execute the rendering.
Friday May 29	4	Made the renderer objects and implemented an insertion sort. Changed the positioning calculations to better center objects' renderers. Began to create a SpriteRenderer component.
Saturday May 30	4	Improved the Transform component to properly handle local positions. Finished the SpriteRenderer Component. Created a Camera GameObject to allow moving around the scene and zooming in. This will allow for the use of a more reasonable coordinate system (1 unit = 1 meter vs 100 units = 1 meter). Fixed the local position calculation again. Fixed issues in the hierarchical structure. Finished camera operations, allowing for camera movement and zooming. Tested player movement to ensure camera positioning worked properly.
Week 2		
Wednesday June 3	3.5	Researched ways to implement action listeners similar to C# delegates and events. Came across an article that describes the use of a custom EventListener interface and replicated the code on its own to test. Plan to use this to allow scripts to get key and mouse events for input. Removed the "Game" class as it added no functionality over GameData and only called GameData methods which could be called directly. Added an empty Start method to the Component class, and removed the "abstract" tag from Update so that components can choose to use them or not by overriding. Adapted the EventListener code for my game engine, using InputEvents to combine Key and Mouse events Created a basic input controller system for a player script
Friday June 5	5	Created in InputManager to replace the input events system. This will make it easier for scripts to access key states by requesting information, instead of being notified directly. Created an Axis system that combines input from two or four keys (ex WASD) and outputs a 1-Dimensional double or 2D Vector representing the input. Fixed the rendering issue where the positive y-axis points down by scaling the Graphics2D vertically by -1 and modifying some Renderer classes to reflect this change. Created the abstract Collider class and the child CircleCollider and RectangleCollider classes. Created the algorithms to calculate if 2 colliders of different types are colliding. Tested the collision detection with a moving rectangle

		and circle.
Week 3		
Wednesday June 10	4.5	Created a new class: TileMapCreator. This class takes a .tileset file to create a HashMap of Tile GameObjects that can be used to create a tiled world, similar to Super Mario Bros. or Pokemon. The tiles are based off of .png tilesets that I create in Aseprite. Upon construction, the object creates the map of tile gameobjects, complete with SpriteRenderers and RectangleColliders. CreateTileMapGameObject can be called using a .map file to use these tiles to create a game level using the tiles. A .map file looks like a grid with pipe bar (|) characters to separate 3-character tile codes, allowing for a semi-visual way of editing worlds. Created “Clone” methods for GameObject and all components. This allows for entities to be copied into a new object, as this is not regularly possible with Java. Finished the TileMap code, so levels can now be made using tiles. Created a basic world to test. Began to write Javadocs for some classes for easier usage. Fully Documented the classes: Vector2, Mathf, Component, Transform
Friday June 12	1	Added the Tag property to GameObject for grouping several objects. Documented more methods using Javadocs. Created a RaycastHit class for returning information from a ray collision. Researched methods to implement raycasting. Decided to use the Liang-Barsky algorithm for rectangle collisions.
Saturday June 13	4	Used a modified Liang-Barsky algorithm to calculate collisions between lines and rectangles. Created an ArrayList on the Scene to store all colliders in the scene, so Raycasts can check for them. Created the Raycast function, which returns a RaycastHit of the closest colliding object, if one is found. The Raycast method takes a “tagMask”, which is a list of tags that should be checked (allowing for objects to be ignored). With Raycasting done, I can begin working on a character controller. Began the character controller. Got a basic gravity and control system working, with ground collisions using my Raycasts.
Week 4		
Monday June 15	3	Translated 2D character controller code from C# to make a player controller that uses raycasting for collision detection. Modified the GetBounds method to properly return the Bounds of a collider, as there was some unwanted offset. Troubleshooted errors in calculating raycast positions. Learned the hard way that multiple references to a static variable from a separate class refer to the same object instead of creating a new one upon each reference.
Wednesday June 17	4.5	Expanded the TileMap1.map to make the world slightly larger. Created cannon and cannonball sprites in Aseprite and completed a basic Animator Component that changes the sprite of a sprite renderer using a ScheduledExecutorService. Began to create a better way to create animations using files, similar to the .tileset and .map files. Completed the Animator and AnimationClip classes, as well as the file reader to create animations. Created a few more movement sprites for the “Slime” character and made these react to the player’s movement.
Thursday June 18	3.5	Determined that a rendering error was occurring due to the frame rate being too high for the computer to process all renderings before the next frame. Reduced the framerate from 100 to 60 FPS, fixing this issue. Changed the static Vector2 default vectors from variables into methods that return

		new Vector2, as references were pointing to the same vector object. Created the scripts to make a cannon fire and have the cannonball fall down. Made a GameObject.Destroy method that removes a GameObject from the scene. Ran into a ConcurrentModificationException when trying to remove an object in the update loop, so I moved the Destroy code to run after update. Made a SlimeDeath animation in Aseprite and created the code that makes the player die, player the animation and return to the start of the scene. Used a scheduled executor service to make a delayed destroy method for GameObjects.
Friday June 19	3	Made a Coin Spawner object that creates coins. It searches through the world tiles for valid spawn tiles, then randomly places coins on them. A coin is a collectible item with a spinning animation and a “collection” animation when the player touches it. Made the world slightly larger and added a second, faster firing cannon. Greatly improved performance by only rendering sprites that are visible onscreen by checking if their bounds are outside of the window. This allows for a frame rate of 120 to be used instead of 60 (I would still recommend running at 60).
Friday June 19	1.5	Created a write up to explain how the game engine works.
Total: 51.5 hours over 4 weeks		