Game Proposal: Ricochet Rage

CPSC 427 – Video Game Programming

Term: 2024W1

Instructor: Chris Kerslake

Team: Sigma Studios

Abhigyan Dabla (28266047)
Maysam Farahani (95167003)
Jason Liu (16585622)
William Cheng (86583648)
Sean Ford (42305409)

Table of Contents

Story

Background story

You have been abducted by aliens and need to find a way to escape their spaceship. You find a blaster to use against the enemies, but discover it's not very effective against the aliens when you aim at them. However, you notice that when the lasers you fire bounce off walls, these lasers seem more powerful (perhaps the spaceship's walls charge up the lasers?). You wonder what happens if the enemies are hit with these ricocheted lasers, and if you can use this to turn the tides against the aliens. You start pushing forward to the control room so that you can head back to Earth.

Gameplay elements

Ricochet mechanic: Our game's core mechanic is ricocheting lasers. Players are given a laser gun that deals minimal damage to enemies if aimed directly. However, if they allow the laser to ricochet off a wall first before hitting an enemy, the laser will do substantially more damage. A laser that has ricocheted off a wall will change color to indicate its higher-damage-dealing state. Lasers will also have a lifespan of 3 bounces before despawning off the screen.

Player goals: Players will be tasked with advancing through the alien ship by fighting their way through enemy alien rooms as well as clear any subsequent boss rooms to reach the final boss/level.The exact number of Enemy rooms in the game will be determined based on play testing and available time. Each room requires all enemies to be cleared before you can move on to the next level. If the player defeats every room in our game, they will be presented with a cool "You win!" screen with their completion time. Performance is also measured by how fast you complete each level, where players who are able to complete a level in record time are able to enter their names on a leaderboard on the home screen.

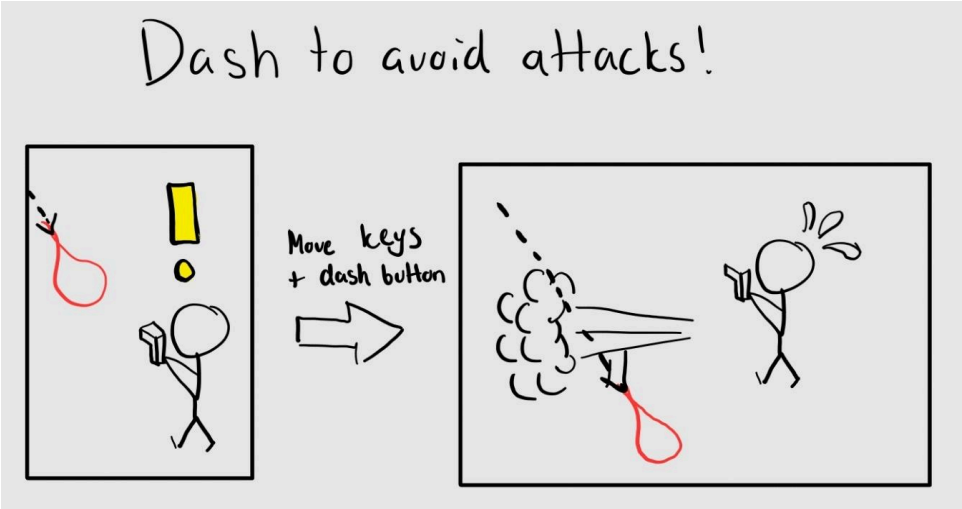
Level design: Each room will have an array of enemies and obstacles (such as square pillars and holes in the ground), all generated and positioned randomly around the room. Lasers will be able to ricochet off of pillars. If a player falls into a hole, they will lose 1/4th of their health and respawn adjacent to the hole. Once the player clears all of the enemies from the room, a red door will unlock at the top of the room and allow the player to walk through to the next room. Players will not be able to retreat to previous rooms once they've walked through these doors.

Health: The player will have a health bar. Collision with enemy projectiles or certain obstacles will decrease the player's health by some percentage. Once the player loses all of their health, they die and must try again from the beginning of the game. Enemies will also have health and will take multiple hits to kill.

Enemies: There will be enemies that attack with melee weapons and ones that attack with ranged weapons. Depending on the type of enemy, it will react differently to the player. Melee enemies will pathfind the player. Once they are within hitting range, they will swing at the player every second or so. Range enemies will pathfind up to a certain distance within the player (enemy is within a X-sized radius of the player), then begin shooting every second or so.

Bosses: The final room/level of our game will contain a boss that will have a much larger health pool than typical enemies, and a unique ability to alter the state of the room (e.g. walls rotating or translating), increasing the difficulty. If we have enough time, we plan on implementing additional bosses to fill out the game, and these bosses will occur every X levels, where X = (# of Enemy Rooms / # of Bosses).

Dashing Skill: To help the player with avoiding projectiles and repositioning, the player can use a dash ability that pushes them in the direction of their movement. To prevent overuse, this dash ability has a cooldown time that activates anytime the dash is used, and must elapse before the player can use the dash again.



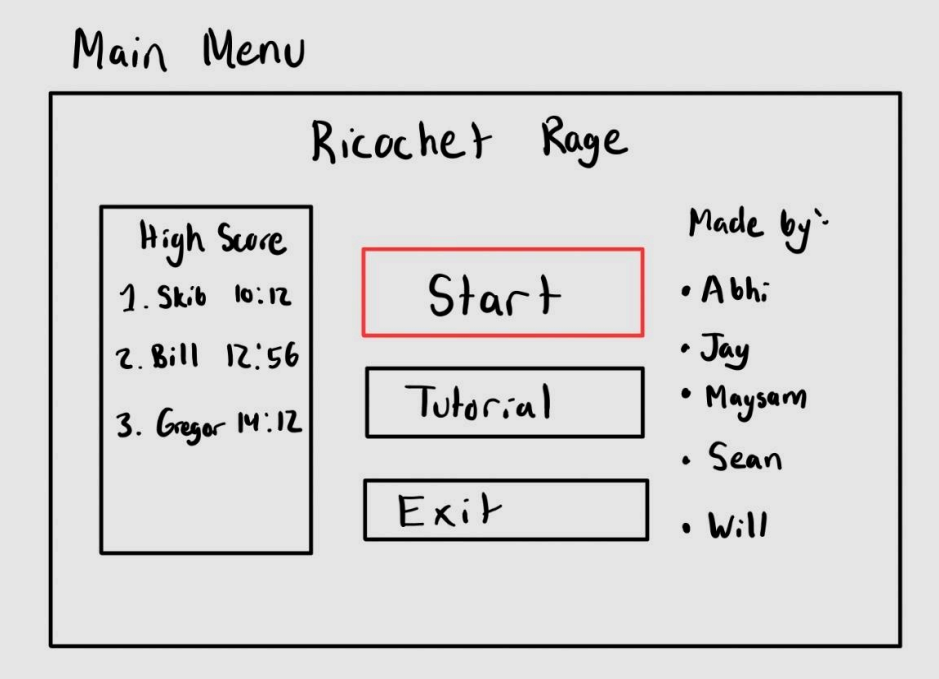
Power-ups: Every so often, a power-up will spawn in the room to aid the player in defeating enemies. Players can walk over the power-up to activate it. If a powerup is not picked up within a certain amount of time, it will despawn (a few seconds before it despawns, it will begin flashing).

- 50% heal potion (instant heal)
- Speed of light laser (instantly damages the first enemy in the laser's path)
- Invincibility to enemy attack (5 seconds)
- Rage mode - For 5 seconds, dashing through an enemy is an instant kill (except for the boss, so this powerup does not spawn in any boss rooms).

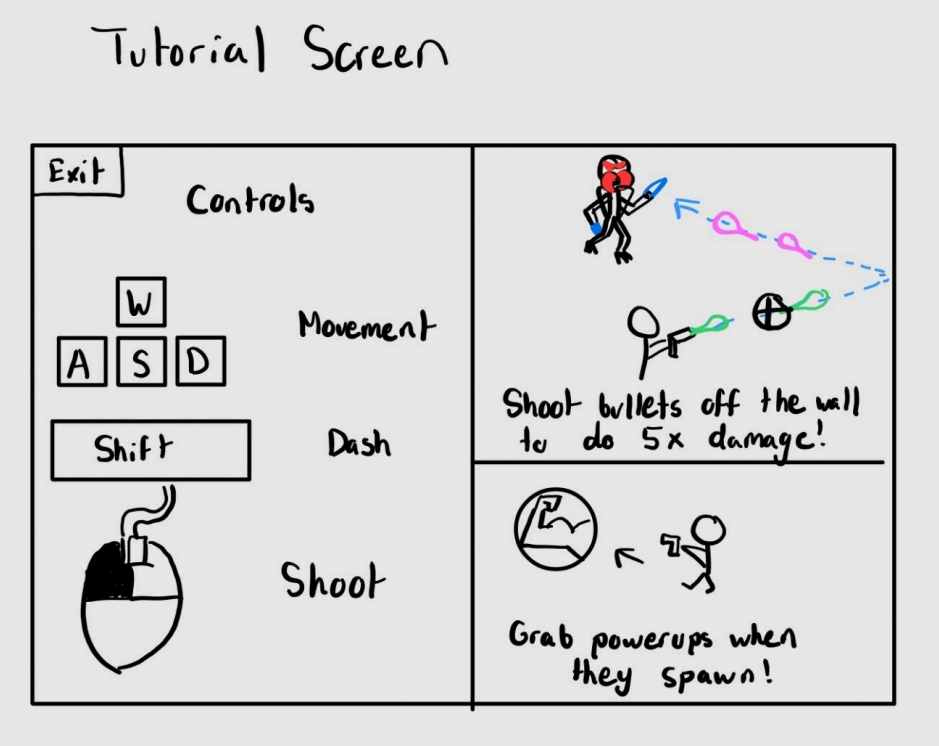
Permanent Upgrades: Every few rooms, the player will reach a screen where they will be able to select one of three random upgrades that are intended to permanently help the player with future enemy rooms. Under each choice there will be an upgrade name and a brief description of what each one does. The player gets to use these abilities as long as they remain alive, and therefore does not get to keep any upgrades upon death. Below are some sample powerup ideas:

- Double Dash (you can dash twice within X time)
- Bullet piercing (projectiles don't despawn on enemy hit)
- Dash invincibility (during dash, you are invincible)
- Flaming Dash (dashes do X damage)
- Vampire Gun (killing an enemy heals X health)

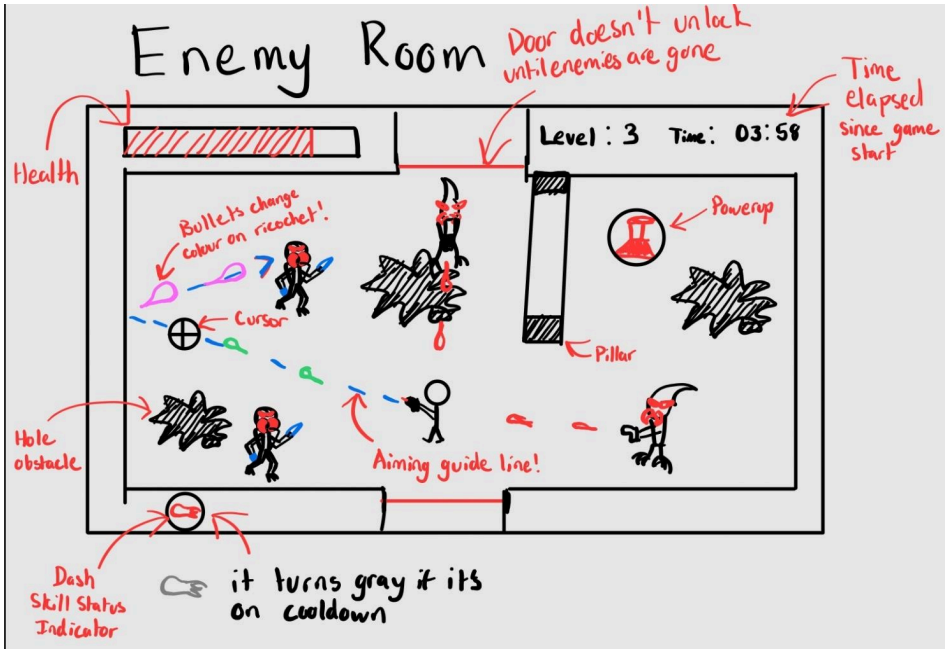
Scenes



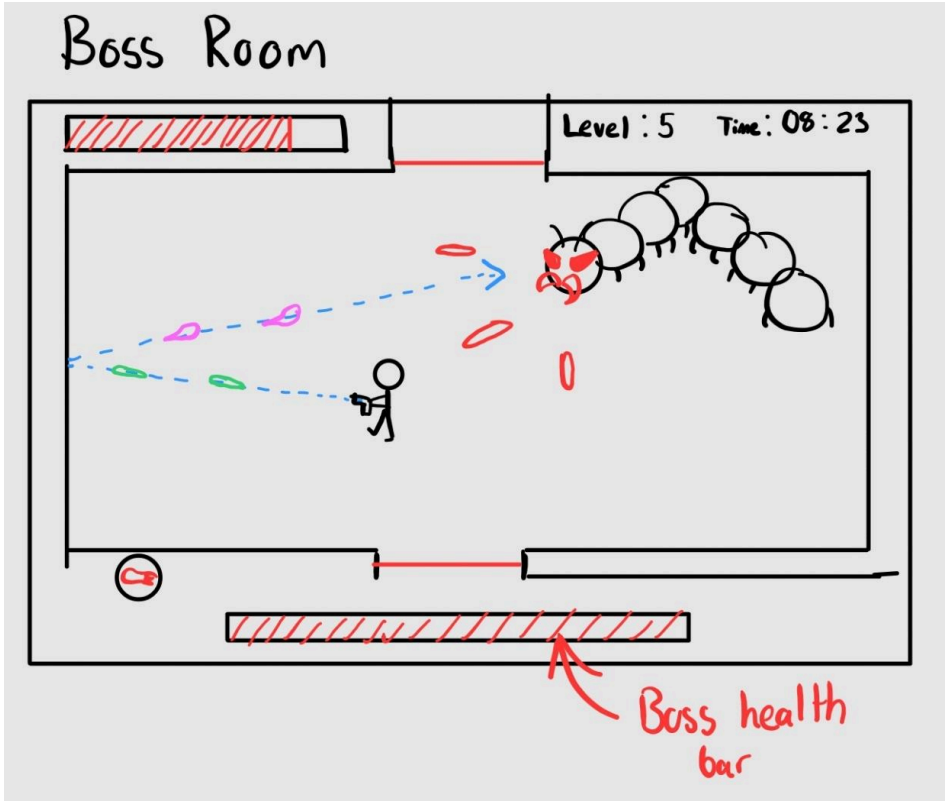
The screen a player sees when they first open the game.



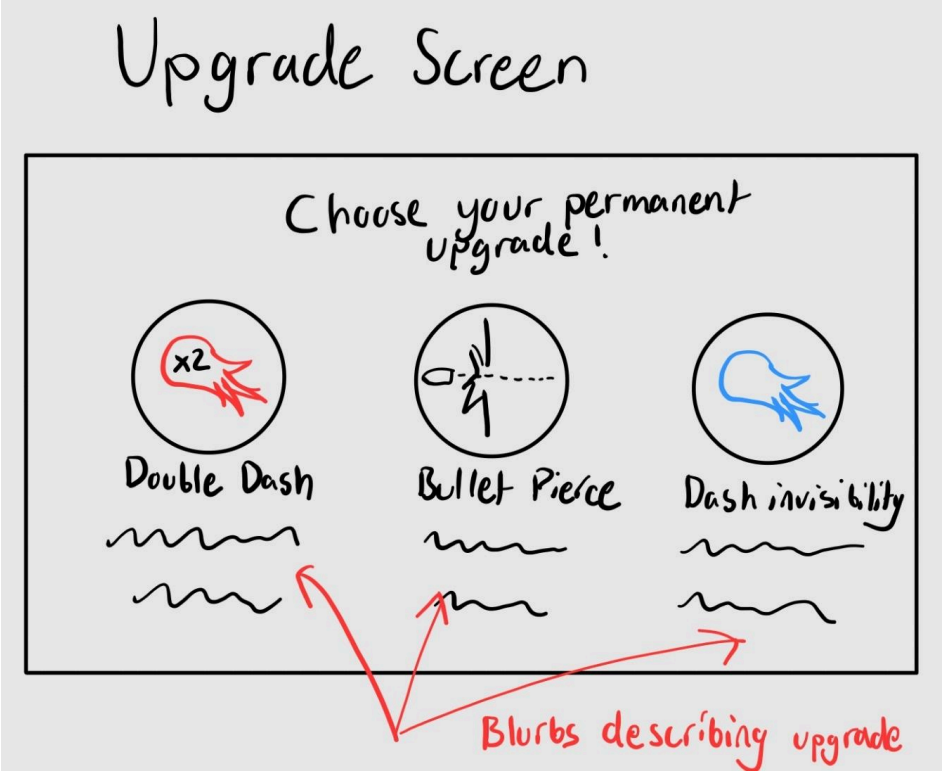
When the player clicks on 'Tutorial' from the Main Menu, they are presented with a screen explaining the controls. We will likely describe a background story right above the controls.



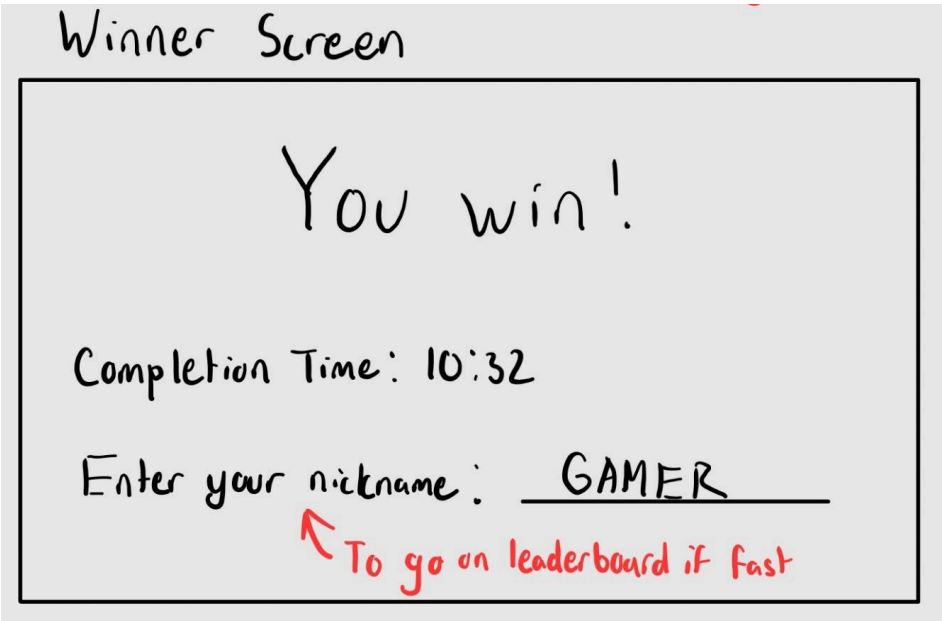
This is the game screen that the players will see the majority of the time. Once the player has cleared all the enemies, they can go through the door located on the top to reach the next level.



Once a player has clearly enough enemy rooms, they will reach a boss room like this.



Every few rooms, the player will get this screen where they choose from one of three random upgrades. Once they click on one of the upgrades, they will spawn in a new enemy room and be able to use the new ability instantly.



The screen the player sees if they clear all enemy and boss rooms. If they have a completion time that would put them on the leaderboard, they can enter their nickname to display.

The screen the player sees once they reach 0HP. They can choose to start the game from Level 0 with the ‘Try Again’ button, or go back to the main menu.

Technical Elements

Rendering: Top down orthographic camera view and sprites will be drawn according to z layers which will make sure that they are rendered properly.

Assets: Art will be primarily hand drawn 2D sprites and audio will be acquired from free asset sites. The assets will follow a pixel or minimalist art theme.

2D Geometry Manipulation: Enemies, players, and bullets will have sprites regularly moved by the physic system. Bullets can collide into characters/obstacles which will cause them to bounce or remove them from the scene. Simple animations are made through translations/transformation of sprites

Gameplay Logic: Player agency is available through keyboard movement (which translates into character movement) and attacking with mouse inputs. UI can be interacted with using keyboard and/or mouse inputs.

AI: Enemies (aliens, bosses) will have pathfinding to navigate the game world and avoid obstacles. Enemies will also use AI to target and attack the player through attack patterns and aiming (If they are ranged).

Physics: Collisions between obstacles, projectiles, characters, etc and updating of entity positions according to velocity and positions.

Sound: SFX for various actions such as shooting, being damaged, interacting with UI, etc as well as background game music.

Components: Physic components (position, velocity, collision box, etc), character attributes (health, damage, move speed, attack speed, etc), sprites (asset, transparency, size, etc), UI components (text, color, size, etc), sound (filename, volume, length, etc). If some components cannot be added we will most likely remove or hard code values to save on time.

Input system: Connects to input devices and updates components and systems accordingly upon input. Input will be entirely composed of mouse and keyboard inputs.

Advanced Technical Elements

Bullet bounce: Bullets will bounce upon hitting walls and some walls will have curved or complex geometry. This will mean we will have to calculate bounce based on normals. However, if this is too difficult we can instead have rectangular rooms and avoid using complex geometry. We can also instead approximate all complex shapes as just rectangles.

Collision detection: Used by obstacles, characters, and projectiles. Collisions between entities will trigger events. Collision detection will be done by checking for an area around the sprite in a predetermined shape. If this cannot be done we will instead calculate collisions using distances between the center of sprites.

Pathfinding: Allows enemy characters to navigate the game world without getting stuck on walls. A* will likely be used for efficient computation. If A* deemed too difficult to implement we can simply just have enemies shoot a ray at the player and if it doesn't hit any obstacles then they will walk in a straight path towards the player which will approximate simple pathfinding.

Rendering system: Runs every frame and allows sprites to be rendered in the correct order onto the screen and drawn in correct order by their z component value. Also renders UI onto the screen. We may also have a custom fragment shader to add post processing to our game. If this is too difficult we can instead do this through sprites.

Enemy system: Enemies will have AI for handling attacks and movement. Enemies will have randomized attack patterns. They will also have dodging mechanics such as moving away from players when they are too close or strafing side to side to dodge enemy projectiles. Enemies will also have a random offset to their attack to make them feel more “human”. If any of these features are too difficult to implement we will not include them in the game as these are mostly just additional quality of life changes to our other existing systems.

Sound system: Manages and plays different sound files. We will also have a menu to customize the volume of music and sound effects. However if we cannot feasible achieve this in the given time we will instead hardcode a volume value.

Event system: Manages events that can be subscribed to trigger a callback when triggered. (Ex: Collision system triggers the event onPlayerHit -> player gets updated, loses health, and player sprite flashes red). If not possible, we will just make public functions that will need to be called upon an action occurring.

Animations: Sprites will have different animations for various actions and events (attack, damage taken, death). Animations will be constructed using sprite sheets, keyframing, or procedural animation. We may scrap animations if we do not have enough time to complete them and may just opt for static poses that characters will take during actions.

Rigidbodies: Used to create more realistic and smooth collisions and physics. Will have a weight, velocity, and acceleration. Can have initial forces applied to the rigidbodies. If time does not permit we will not implement this and instead just stick with our simple physics system.

Design Choices

Devices

The input devices that we plan on supporting will be the keyboard and mouse. From the keyboard, the direction of the player's movement comes from the arrow key or WASD. The player can aim with their mouse, select the general direction, and shoot a laser with the left mouse click. The player can also dash with the "shift" button on the keyboard. The player will then dash towards the directional key selected on the keyboard.

All GUI elements will provide buttons for the player to click with their mouse. The player can pause the game by pressing the "ESC" button. We will make this clear to the player, so there is no need to use the mouse other than aiming in the actual gameplay.

Tools

Note: This proposal is subject to change over the development lifecycle as the requirements and expectations for each milestone are updated/discussed.

OpenGL is mostly a rendering API and does not support sound, inputs, windowing, animations, GUI and timing. Tools like GLFW are designed for windowing and inputs, providing support for keyboard and mouse inputs. We will be using SDL for our sound system.

Since our games rely heavily on physics for our player, enemies, bosses, and bullets, we could use physics libraries such as Box2D, which could help with the collisions and rigid bodies. It would be effective in representing the ricocheting lasers, especially against walls of differing shapes. However, we may not use it as the project requires us to write our own code.

There will be some level design tool we can use, such as Tiled.

We will use software for 2D animations, such as Blender or Pencil2D etc. This will be necessary for the player sprite, enemies, and lasers. It will make the game more engaging.

Other libraries will be suggested as the project grows.

Team Management

We have decided to assign tasks based on everyone's specialities, interests, and strengths. Abhigyan, Maysam, and William will serve as our backend development team and Sean and Jason will make up our frontend UI design and development team. In particular, Masyam and William will serve as our primary backend gameplay developers focusing on physics, gameplay logic, enemy AI, etc. Additionally, Abhigyan plans to take a more project manager role along with his contributions to the backend development by helping keep track of deadlines, documentation, and assisting in the coordination of both the backend and frontend teams. On the other hand, Sean and Jason will be involved in the art direction of the game as well as UI design, sound, character sprites, etc. for their involvement in the frontend development of the game. Lastly, we also designated a backend lead and a frontend lead role as well which belong to William and Sean respectively to ensure a stronger and more robust team structure.

Additionally, we plan to use Jira to track our tasks and to-do items to help organize our team tasks and distribute the work evenly and fairly. We have outlined our internal deadlines in our development plan on specific game requirements that we wish to accomplish each week, which we will strictly follow to ensure that development continues smoothly. For our policies, we plan to be transparent with one another in our work progress, following the *Teamwork Agreement* we signed earlier. We've also established a policy within the team to create PRs one day before the deadline for each item to enable teammates to thoroughly test and review the work completed, giving time to address bugs or other concerns.

Development Plan

This section outlines the development plan we will use as a roadmap throughout our project. Many key milestones and deadlines mentioned in this section must be adhered to ensure timely completion of this project. For each milestone, we will intentionally work on a larger number of features in week 1 of the sprint to ensure we have ample time to implement all required elements if delays arise.

Note: Keep in mind that this is a living document and subject to change during the development lifecycle. Therefore, further milestones of the project may not be as detailed as upcoming milestones. They will be updated with more precise deadlines as the project progresses.

Milestone 1: Skeletal Game - 10/06/2024

The aim of Milestone 1 will be to use our knowledge from A0 and A1 to create a strong foundation for our game, which will be built off of in future iterations.

Week	**Description**	**Deadline**
Week 1	Setup basic ECS system and implement player, enemy, and guns entities, along with their basic components.	09/27/2024
Week 1	Setup basic UI for the video game with walls and basic movement mechanics based on keyboard/mouse interactions. Create a window showing the main Enemy room screen. This screen will have a blue square (player) that is controllable by the WASD keys, inside a square room made up of 4 individual walls.	09/28/2024
Week 1	Implement basic enemies. Placeholder red squares (representing enemies) will randomly spawn around the room. These squares will respond to the player's movement by changing direction towards the player (using linear interpolation between their positions).	09/29/2024
Week 1	Test and optimize for frame rate after basic backend and UI are implemented to ensure stability with incoming user inputs	09/30/2024
Week 2	Design a basic collision system to stop the player at the walls boundaries and for bullets (if time permits).	10/02/2024
Week 2	Test out features, make improvements, fix bugs, and write out test plan/known bug list documents.	10/03/2024
Week 2	Plan and record demonstration video to showcase game features.	10/05/2024

Milestone 2: Minimal Playability - 10/27/2024

The aim of Milestone 2 will be to implement the more complex mechanics of our game. This includes a pathfinding algorithm for enemies and the “ricochet” mechanic in our game.

Week	**Description**	**Deadline**
Week 1	Setup home screen, user tutorial guide, and game screen, all connected via interactive buttons.	10/13/2024
Week 1	Improve our collision detection system for bullet ricochet and character movement.	10/15/2024
Week 1	Implement a pathfinding algorithm for enemies to attack the player more effectively.	10/18/2024
Week 2	Incorporate better animations for the player, enemies, room, laser guns, etc.	10/23/2024
Week 2	Test out features, make improvements, fix bugs, and update test plan/known bug list documents.	10/25/2024
Week 2	Plan and record demonstration video to showcase game features.	10/26/2024

Milestone 3: Playability - 11/17/2024

The aim of Milestone 3 will be to complete most game features, user abilities, and power-ups by this time, leaving behind only playtesting and bug fixing for our final weeks leading to the final release. We will also emphasize the robustness and stability of our video game.

Week	**Description**	**Deadline**
Week 1	Implement top performers leaderboard on the home page.	11/10/2024
Week 1	Design and create the final boss for the last room with different animations, sounds, and abilities.	11/11/2024
Week 1	Implement power-ups spawning in game and in-between rounds and player inventory for storing power-ups.	11/12/2024
Week 2	Design and create multiple game rooms and the functionality to move users to a new room when they complete a round.	11/15/2024
Week 2	Test out features, make improvements, fix bugs, and update test plan/known bug list documents.	11/16/2024
Week 2	Plan and record demonstration video to showcase game features.	11/16/2024

Milestone 4: Final Game - 12/01/2024

The aim of Milestone 4 will be to finish and release the final video game. This will require significant testing, bug fixes, implementation of final features, and improvements to the game art.

Week	**Description**	**Deadline**
Week 1	Test features thoroughly to gather a list of bugs and areas for improvement before final release.	11/21/2024
Week 1	Finish implementing remaining features from previous milestones.	11/23/2024
Week 2	Fix all bugs, make final adjustments to the game.	11/27/2024
Week 2	Test out features, make improvements, fix bugs, and update test plan/known bug list documents.	11/29/2024
Week 2	Plan and record demonstration video to showcase game features.	11/30/2024