Game Proposal: Laurel Nest

CPSC 427 – Video Game Programming

Team: Burnt Bird

Kuter Bora - 83592808

Mandy - 92389279

Anikait - 83243055

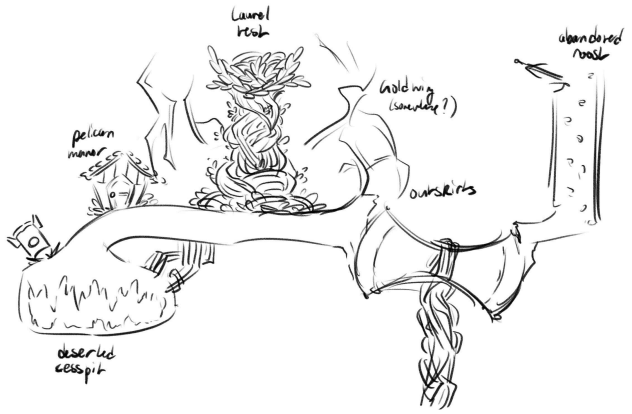
Jett - 23956840

Dizhe (Sandy) - 47565742

Brian - 27539022

To view larger images, view them in the appendix by clicking on the provided links.

Story:



The game takes place in a sci-fi/fantasy world deep in space. The Lord of the Birds, the Great Bird, tried to conquer the known universe but failed, retreating to his nest to lick his wounds. The news of his misfortune spread far and wide, and many have come to the kingdom of the birds, looking for loot, power, or revenge.

The player is a human (though wearing a bird mask), among many, who has come from outer space to seek treasures, power, and, most importantly, the Crown of Claws worn by the Lord of the Birds, an ancient artifact which grants the bearer untold power. The Lord of the Birds (or the Great Bird) rests in his palace, the Laurel Nest, where the player initially aims to steal the crown. However, their spaceship malfunctions and crashes too soon [29], and they land far from the Laurel Nest.

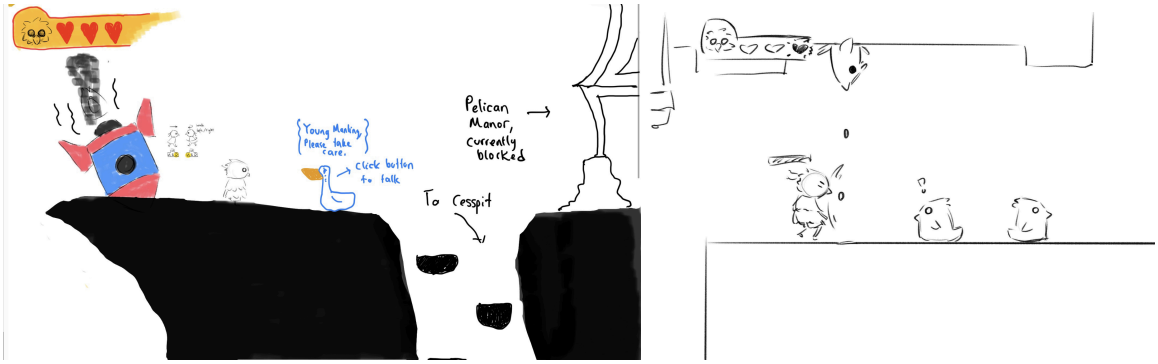
The player will have to walk and jump [4] to traverse through up to three areas of the game: the Deserted Cesspit [2], Town Outskirts(if time allows) [32] and Birdmen Town [3]. The areas will feature various platforming challenges [9], as well as difficult enemies that can set back your progress to previous checkpoints by killing you.

At the end of each area, there will be a tough and challenging boss. The bosses will have attack patterns the player must recognize to dodge successfully. The first boss is the Blazing Chicken [19], which drops its flame-throwing beak that the player can use to break down obstacles [12]. If time allows, the bosses “Birdman Elder” [21] and “Goldwing”[20] will be implemented too, each dropping a tool (wings [14] and dash [13]) to allow the player more versatile movement.

The final boss is the Great Bird [22]. Successfully defeating The Great Bird grants the player a choice of one of two endings [31], to take the Crown of Claws and abandon the kingdom or to stay behind and attempt to restore the bird kingdom to its former glory.

Along the way, the player will encounter up to four NPCs [7] (at least two). The player who takes the time to interact with the NPCs will learn about them and their troubles. These interactions may or may not influence the player's choice of ending.

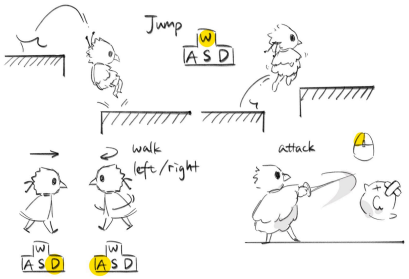
Scenes



Player Character

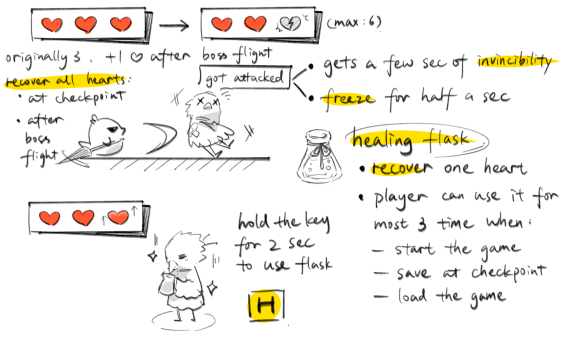
Player Initial Movement [4] [15]

Initially, the main character can walk right or left, jump, and attack with their sword. The jumping ability enables the player to move between different platforms. Enemies can damage the player through contact, and the player can damage enemies through contact with their swords and other tools (see Tools section for additional details). The sword is used by pressing the left button of the mouse, and has a small cooldown before it can be used again to keep the visuals clear. The sword's hitbox covers the immediate area the player is facing.



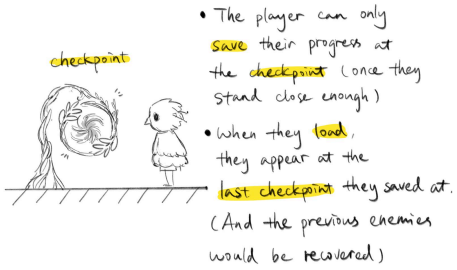
Player Health [5]

The player initially has three hearts. Every time the player gets hit, they lose a heart. When damaged, the player gets a few seconds of invincibility and freezes for half a second. The player also has access to a healing flask that can restore their lost hearts. The player starts the game with this flask and may use it at most 3 times before reaching a checkpoint [6]. The player must press the key “H” for 2 seconds to use the flask. The enemies also have a set number of hearts, which are hidden from the player, requiring them to keep hitting the enemies until they are eliminated. However, they could also try to jump over and run past, with the exception of bosses.



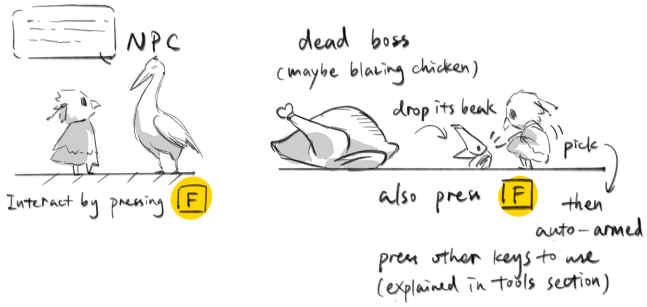
Checkpoints [6]

Once the player loses all their hearts, they start again from a checkpoint. Each area has a single checkpoint (The locations will be decided later in development, with a better idea of the game's difficulty). The progress will be auto-saved when the player is close enough to a checkpoint. When they load the game, they will start from the last checkpoint at which they saved it. They can recover all the hearts and healing flasks by approaching the checkpoint. Further implementation details are discussed in the "save/load" section of technical elements.



Picking up items/ character interaction [7]

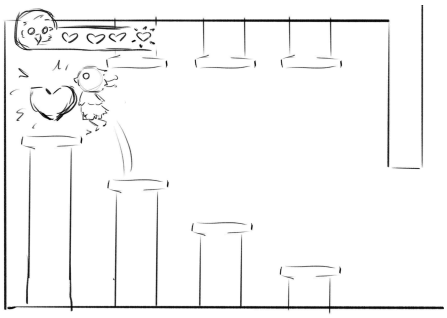
After defeating a boss or interacting with NPCs, the player can obtain new tools by picking up items dropped by these characters. The player can initiate conversations with NPCs by approaching them and pressing the "F" key. Similarly, they can also pick up tools by pressing the "F" key.



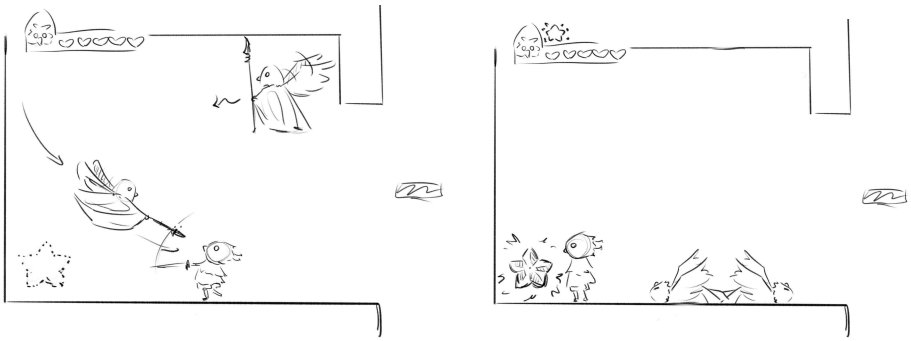
Player Progression System

1. Platforming Challenges [8] [9]
   Completing platforming challenges can earn the character more hearts. Each area has one challenge, and completing each challenge gives the player an additional heart. Completing them

all will leave the player with six hearts. The challenges include jumping on/over obstacles and avoiding hazards.



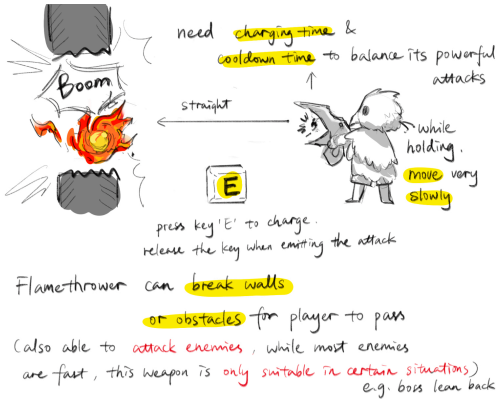
2. Combat Challenges [10] [11]
   Each area also has a combat challenge the player can complete to increase the damage of their sword. The challenges will include small rooms full of enemies where the player enters and cannot leave until all the enemies are defeated. Once successful, they obtain an upgrade to their damage output.



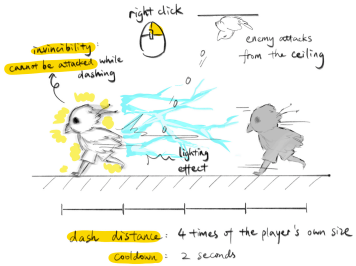
Player Tools

By defeating bosses or communicating with NPCs, the player unlocks new tools. These tools are auto-equipped and bound to a key on the keyboard that can be customized from the options menu. There are up to three tools (at least two), each with a different benefit.

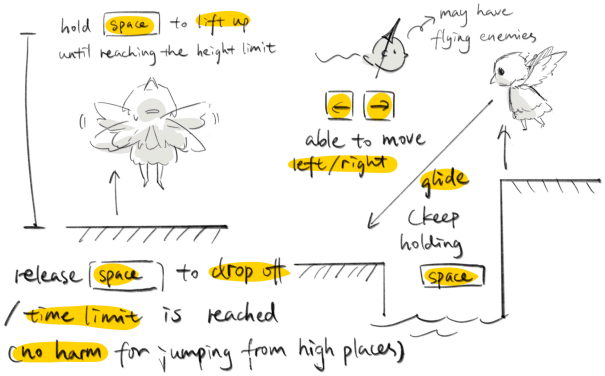
1. Flamethrower [12] [19] [7]
   After defeating the first boss, ‘Blazing Chicken’, it drops the ‘Flamethrower Beak’, which the player can use to deal a lot of damage to enemies or to break walls to access previously blocked areas. The beak must be charged for a few seconds by holding its associated key before use and has a long cooldown period. It also requires the player to be stationary, leaving them vulnerable to enemies.



2. Dash (if time allows) [13]
   If the boss Goldwing [20] can be implemented, the player's reward for defeating it will be the dash ability, which the player can use to move away from enemies quickly. In Birdmen Town [3], some enemies will attack fast enough that the player will need to use this tool if they do not want to get hit. This tool will have a cooldown after being used for a few seconds.



3. Wings [14]
   If the boss, Birdmen Elder [21], can be implemented, the wings will be rewarded for defeating him. However, if we can't implement Birdmen Elder, then an NPC character will drop the "Wings" which the player can use to fly for a duration of time by holding the space button. The player can elevate by flapping the wings either until they stop holding the jump button or until a set maximum duration is reached. Once the flight is over, the player can glide down slowly while holding the space button, or they can fall quickly by not holding. When gliding and flying the player can move to the left or right with the keys they normally use to walk. This tool will be helpful when the player tries to avoid attacks from the final boss [22].

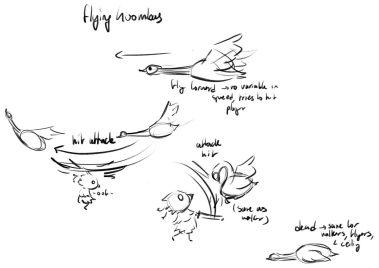


Enemies

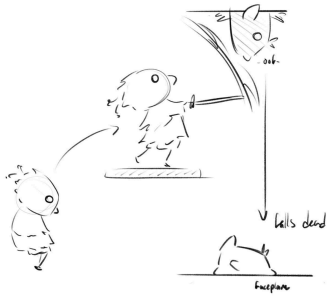
"Goomba" Bird - Walker [15] [30]

A very simple enemy that patrols left and right in a given range. When a player enters their range, they speed up and try to crash into the player. If successful, they take a step back from the player and then attack again. They have animations for walking and dying, as well as sound effects for walking, attacking and dying. They die after a few hits of the player's sword.

<u>"Goomba" Bird - Flyer (if time allows)</u> [16]
Similar in design to the "Walker", this enemy flies left and right in a given range. Once the player is in range they fly towards them slowly and try to have contact. If successful, they fly away for a few seconds and attack again. They have animations and sound as the Walker with the exception of walking animation and sound effects; instead they have a flying animation and sound effect. They die after a few hits of the player's sword.

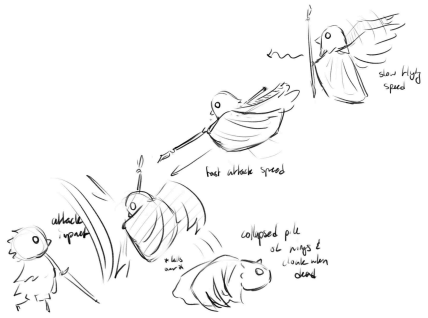


<u>"Goomba" Bird - Ceiling</u> [17]
A static enemy that is stuck to the ceiling and spits projectiles downwards every couple of seconds. They have animations for spitting and dying, and mostly stand still. They require a single sound effect which is for spitting. The user must make use of the nearby platforms to kill the Ceiling Goomba Bird; however, they must be careful as a mistimed jump will result in the player landing directly in the part of the Ceiling Goomba Bird's projectiles.



<u>Birdman</u> [18] [10]
Birdman is a more complex enemy that is found in Birdmen Town. They dash very quickly towards the player with their lance raised. After attacking, they slowly fly back for a few seconds which provides the

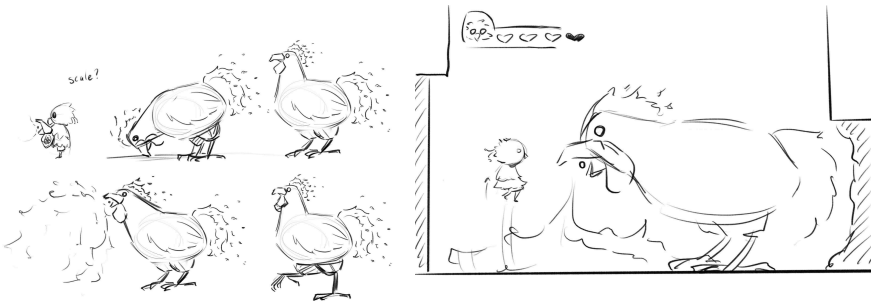
player a moment to attack or heal. They have animations and sound effects for flying, attacking, and dying. It takes many hits of the player's sword to defeat them.



Bosses

Blazing Chicken [19]

The Blazing Chicken is the boss of the Deserted Cesspit, found in a large rectangular arena. It attacks the player by pecking the ground if they are nearby, forcing them to stay out of range. The Blazing Chicken also attacks by spraying flames that move left and right on the ground, forcing the player to jump. The Blazing Chicken can also walk back and forth to change the arena's size dynamically as it is too large for the player to jump over. It has animations and sound effects for walking, pecking, breathing fire, and dying. This encounter will take a few minutes for a player to beat if they have a grasp of the game' mechanics. Music will be included during the fight, as well as a large text reading "Great Bird Vanquished" upon its defeat.



Goldwing (If time allows) [20]

Goldwing or the Lightning Bird will be an "environmental boss" that shoots lightning off screen towards the player when they try to traverse the Town Outskirts [32]. The lightning will deal a lot of damage to the player, making it impossible to climb to Birdmen Town [20] (The lightning in that region will be scripted to hit the player). The player will have to traverse all the way to the rightmost region of the map and climb the tower there to find and fire a giant ballista. A loud scream will be heard upon firing, and when the player ventures back to the Town Outskirts, they will find the corpse of Goldwing. The design for Goldwing requires a single asset for its corpse, as well as his lightning. Sound effects are needed for his death scream and lightning attacks.

Birdman Elder (if time allows) [21]

Birdman Elder is a static boss at the fourth level of Birdmen Town who attacks the player by summoning flying lances that track the player. The boss flies in a static spot in the middle of the room and summons

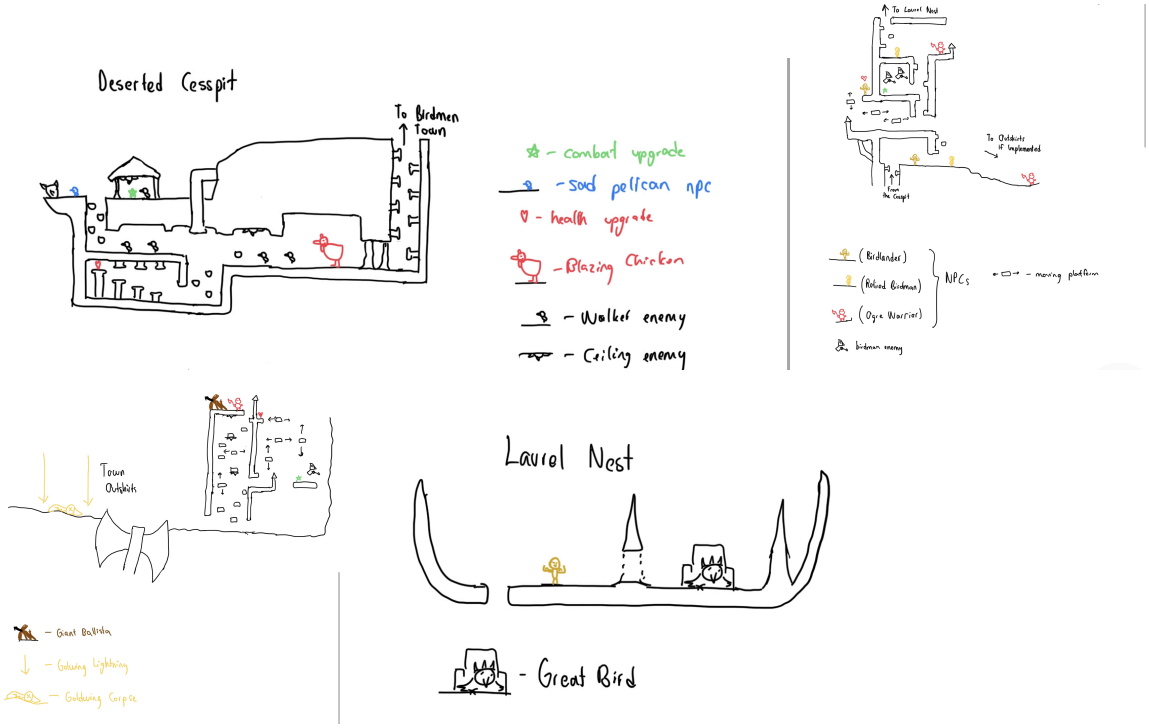
the lances into existence from the two sides of the room. The player must avoid the lances and attack the boss at any safe interval they can find. As the boss gets damaged, the lances will get faster, becoming very hard to avoid without dashing. It requires animations and sound effects for flying, dying and the lances.

Great Bird [22]
The final boss is the Great Bird, who has grave wounds, missing two out of its three legs and one of its wings. The Great Bird is static and has two attacks: slamming his head to the ground to create a golden shockwave, forcing the player to jump, and summoning golden spears that rise from the ground (golden dots will appear first to indicate where they will appear). This specification for the Great Bird requires assets and sounds for when it is idle, slamming its head, opening its beak, and dying as well as the golden shockwave and the golden spears. This design is subject to simplification and expansion, depending on time.

NPC Quests
All of the NPCs will function similarly - they will stand still and speak to the player with text boxes. If time allows, they could be made to move/interact with the world as they speak. An example NPC is provided below:

Poor Old Pelican [7] [29]
The first NPC the player encounters is an old pelican right at the very beginning. The pelican provides the player with some initial context and sends them off to the Deserted Cesspit [2]. If the player insists on speaking with the pelican, it complains that its house's entrance has been blocked by a pile of waste, mentioning its anger for the chicken clan who have deserted their duties at the cesspit, causing it to be overflown. The pelican's house is easy to find, being right next to the entrance of the Deserted Cesspit. After defeating the Blazing Chicken, the player gets the ability to break down waste walls with their flamethrower, including the door to the house of the pelican. The player will find the house empty initially, but if they go back and speak to the pelican, it will move back to its house. If the player visits the house after the pelican has moved in, the door will be locked behind them and many enemies will be spawned inside, beginning the optional combat challenge of the Deserted Cesspit. If the player defeats all the enemies, they will gain an upgrade to their damage output. They will also find the poor pelican's corpse in the house, having been killed by the same enemies.

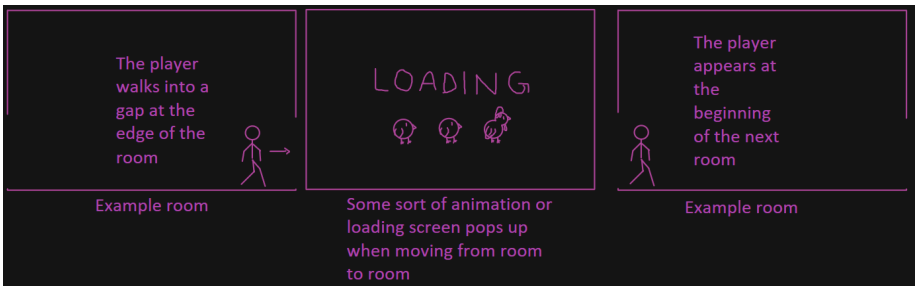
Areas
There will be up to three areas in the game. Deserted Cesspit [2] and Birdmen Town [3] are the core areas, and Town Outskirts [32] will be implemented if time allows.



Technical Elements:

Graphics/Rendering

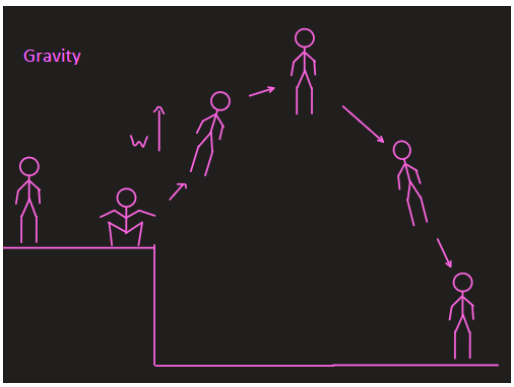
- Simple rendering effects:
    - Using OpenGL, sprites will be texture-mapped quads (rectangles) rendered in a 2D space. Load textures (like .png files) representing characters, enemies, and objects, then map them onto the quads for rendering.
    - Add fragment shaders to wrap rocks/obstacles/ground with textures. The fragment shaders also simulate lighting or shadow effects similar to Hollow Knight to enhance the visual atmosphere.
    - Also, we will change the color of a sprite over time. The color should change based on a uniform input (e.g., change the uniform based on time for the part the flamethrower is charging or cooling down).
    - Sprite Handling: sprites will be stored as texture data and mapped to entities in the ECS structure. Each entity (e.g., player, enemy) can have a component that stores the texture and coordinates for its appearance.
- Backgrounds
    - We decided to divide a big map into different parts (called “rooms”). While the room background is fixed, when the player reaches the boundary of a map, there will be a hole in the ground or road sideways, and we will add some scrolling effects to make an animation for transferring to a new room.



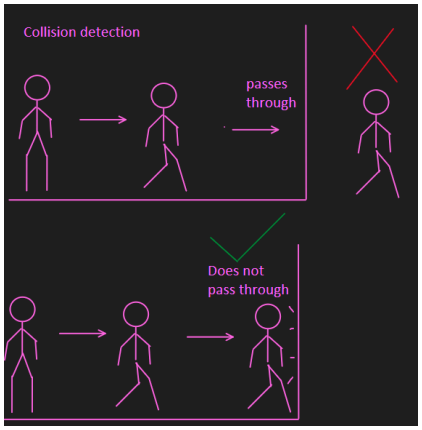
- Skinned motion
    - For the player's 2d image, we may simplify the implementation by separating the parts of its arm and swords to make the attack animation. If we have enough time, we can upgrade it to have bones to make the animation smoother using transformation.

Physics & Simulation

- Basic Physics:
    - Implement simple physical phenomena: when the player jumps or glides using the wings, we need to update the y-position by gravity. For gliding, we will temporarily reduce the gravity applied to the player, allowing for slower descent while maintaining horizontal movement. Each entity can have components for velocity and acceleration, with a system applying gravity to the player and enemies.



    - Also, the game has inelastic collisions between enemies and the player.
- 2D geometry manipulation:
    - Transformations: In OpenGL, handle transformations (scaling, rotation, translation) using matrices. In the ECS structure, each entity could have a transformation component for position, rotation, and scale.
    - Collision Detection: For collision detection, use bounding boxes or bounding circles depending on assets. We will implement a simple collision detection system where entities in ECS have collider components, and a collision system checks for overlaps (important for player-enemy and player-platform interactions).



Gameplay Logic/AI

- Player Movement: The player's movement (jumping, gliding, dashing) will be handled by systems within ECS. For example, the player's entity could have a movement component updated every frame based on input.
- AI for Enemies: Implement basic AI for enemies, like patrolling behaviour or attacks triggered when the player is within a specific range. The AI logic can be a system that checks for conditions and updates enemy behaviour accordingly (e.g., the birds shooting down from the ceiling, change in speed for dashing goombas).
- Boss Fight Mechanics: Simplify boss fights by creating a state machine that controls different attack phases triggered by hidden health boundaries, similar to Hollow Knight bosses, with intervals allowing the player to attack.

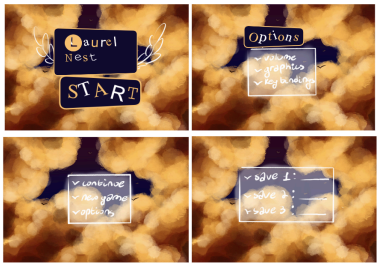
Software Engineering

- Save and Load (re-loadability):
    - Press "Esc" to pop up the menu, and the player can auto-save their progress when they are at the checkpoint. When the player loads the archive, restart at the last checkpoint they reached before exiting. The player will keep all their possessions; however, all enemies respawn.
    - To ensure proper save functionality, we will use the JSON format for serializing entity and component data. This will allow us to preserve the player's progress (e.g., currency, items) while resetting variables that are necessary for gameplay balance, like enemy positions.
- External integration:
    - We will integrate one or more external libraries to enhance game functionality, such as physics simulation (e.g., Box2D) or the EnTT ECS system for managing entities and components efficiently. These libraries will streamline complex features like collision detection and entity management, ensuring smoother development.
    - **EnTT** is a fast and lightweight Entity Component System (ECS) library that simplifies the management of game objects and their behavior. And **Box2D** is a 2D physics engine

that simulates real-world physics in games, handling things like gravity, collisions, and object interactions.
    - **Notes**: Before merging any integration into the main branch, we will ensure that all team members can install and run the libraries on their development environments. This includes compatibility checks across different operating systems (Windows, macOS) to avoid issues that have arisen in past projects where teammates were unable to contribute or test the program due to system differences.
    - Brian has written a stylized text generator browser app that can be used for UI elements. It currently lives in a private Github repo.

User Interface (UI) & Input/Output (IO)

- Art and Assets: the assets and art for most of the sound effects, all of the backgrounds and objects (checkpoints, health/combat upgrades, platforms), most of the visual effects (like flames and lightning), will be taken from websites providing free assets.
- UI text design by Brian, for music, Kuter's friend can help, but if he can't, there's always royalty-free from incompetech.com (Kuter's friend is 99% confirmed. He is already done with Blazing Chicken and has started the theme for Deserted Cesspit).
- Audio feedback: Sounds needed for any movement (player movement, enemy movement) and action (Sword swing, Flamethrower) will also be taken from websites providing free assets. Sounds that are needed for characters, such as the player sounds, the major bosses, and NPCs, will be voiced by the members of the team.
  List of all sounds that should be included, as much as time allows:
    - Player sounds: walk (can be different depending on type of ground), heal, get damaged, die, jump, land, sword attack, sword attack that hits an enemy, flamethrower charge (if implemented), flamethrower attack, dash (if implemented), wing flap, receive upgrade.
    - NPCs: dialogue
    - UI: game saved, button click - could be different for different buttons
    - Enemies: Goomba walk, fly, notice player, attack player, goomba spit, get hit, die. Birdmen fly, attack, get damaged, die.
    - Blazing Chicken: Chicken step, beak slam, breathe fire, at least two chicken screams for the two different attacks, death.
    - Great Bird: general crow sound, sound for the spear, sound for head slam, sound for death.
    - Ambient sounds: water dripping in the Deserted Cesspit, wind sound at any room above ground, gears/industrial sounds at Birdmen Town, whispering in Laurel Nest.
- The main menu screen and the sprites for the player character, enemies and the bosses will be hand drawn by two of the team members.
- Screen wipe effects to transition between scenes
- UI - Start Menu/Options: [26] UI



Quality & User Experience (UX)

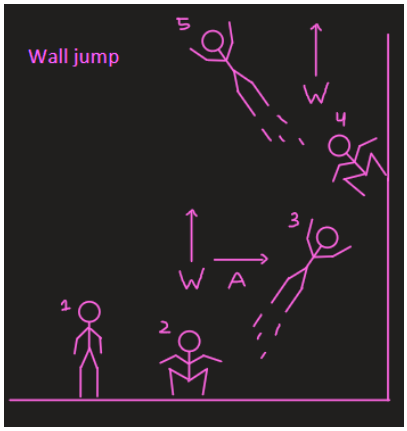
- Game balance(Do this only for the last milestone)
    - Design progression system and healing flasks to make the player have more hearts and attack damage, and be able to recover after being attacked. It would be implemented by changing the health and attack components of the player's entity. and heart recovery mechanism.
    - Add checkpoints so that the player can restart from the latest checkpoints they saved before, which was explained in the save/load function.
- Story elements:,
    - Change dialogue of NPCs to add interactiveness. For example, after the player defeats the first boss, the first NPC's state would be changed so the next time the player talks to this NPC, its dialogue would be changed.
    - Tutorial: At the beginning of the game, after the user gets a new tool/weapon, or the player faces a new challenge, there will be a tutorial description with a cut screen to show the player how to use skills or pass the challenges.

Advanced Technical Elements:

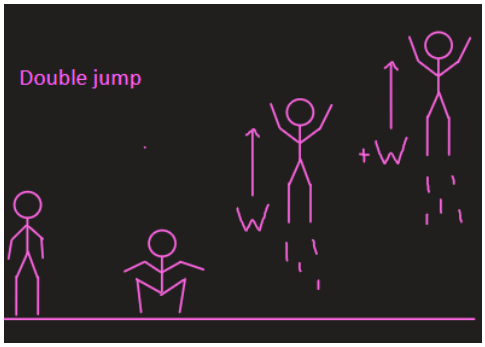
Plan B

- Given the feedback that our original design scope was too broad, we have adjusted our project to include a more focused and feasible set of core gameplay features, as well as reducing the number of bosses and main maps. Our revised design prioritizes implementing critical elements necessary for the core gameplay experience while allowing room for additional enhancements if time permits. Below is a detailed breakdown of the Plan B scope:
- **Wall Jump:**
    - **Description:** The player will be able to perform wall jumps to navigate between platforms or escape enemies. When the player comes into contact with a wall, they can press "<key binding for jump>" to jump off the wall in a diagonal upward direction. This mechanic will only include a simple wall jump (no wall grabbing).

- **Technical Requirements:** Proper collision detection between the player and wall entities will be implemented, along with checks for the player's position component to ensure accurate jumping behavior.
- **Fallback Plan:** If time constraints arise, we will simplify the wall jump to a basic one-way jump, reducing the complexity of the collision detection and physics interaction.



- **Double Jump:**
  - **Description:** The player can perform a second jump while in mid-air, but this jump will have less height compared to a regular jump. This ability can only be used once until the player touches the ground again.
  - **Technical Requirements:** Implement state management to track whether the player is on the ground or in mid-air. Create logic to handle double-jump resets when the player lands on a surface.
  - **Fallback Plan:** If we are unable to implement double jump physics as planned, we will scale back to a single jump mechanic and adjust platform heights to maintain balance.



- **Look/Attack Up and Down:**
  - **Description:** Enable camera control by using the viewport to allow players to look up or down, giving them a better view of the environment and targeting enemies. The viewport will follow the player to keep them in focus and will shift as needed based on player movement.

    - **Technical Requirements:** Integrate viewport manipulation using OpenGL transformations. Implement key bindings to control the camera's vertical movement.
    - **Fallback Plan:** If full camera controls cannot be achieved, we will implement a simple static camera view that follows the player horizontally, with no additional vertical movement.
- **Projectile Physics:**
    - **Description:** Implement projectile motion for the flamethrower and other projectile-based attacks. Projectiles will have linear or arcing trajectories depending on the type of attack.
    - **Technical Requirements:** Create a projectile component and system that calculates motion using physics equations. Implement collision detection for projectile interactions with the environment and enemies.
    - **Fallback Plan:** If advanced physics calculations are too time-consuming, we will implement basic straight-line projectile paths and refine collision detection to ensure core functionality.
- **Particle System for Fire-Based Attacks:**
    - **Description:** Use a particle system to display fire and smoke effects for attacks, such as those from the flamethrower or the "Blazing Chicken" boss. This will visually enhance the impact of the attacks and provide visual feedback for the player.
    - **Technical Requirements:** Use a particle system library or implement custom particle rendering using OpenGL. Create components for particle emitters and renderers in the ECS to handle fire and smoke animations.
    - **Fallback Plan:** If the particle system cannot be fully developed, we will use static sprites or simpler visual effects to represent fire-based attacks.

Devices:

The game will support keyboard and mouse inputs, which will be discovered more in detail during milestone 3 - for example, an option to customize the keys could be added:

- Spacebar: jump and hold to fly when unlocked
- A: walk left
- D: walk right
- Left click: attack with sword
- Right click: dash
- R: flamethrower attack
- Esc: pause/see options
- H: heal

Tools:

- Brian has written a HTML/CSS/JS app that can generate the UI text features.
- We will also use artwork, music and audio from external sources and credit the artist wherever their work is used.

- **EnTT -** a fast and lightweight Entity Component System (ECS) library
- **Box2D** - a 2D physics engine that simulates real-world physics in games, handling things like gravity, collisions, and object interactions.

Team management:

We have written out a list of features that we want to design and implement into our game. From that list of features, we will create user-stories to help break them down into manageable sizes that can be done within a week. Having planned out the list of features we want to implement, team members can pick and choose from the list of user stories that they would want to do. The user stories that we will have created will be based on the list of tasks that is outlined in our development plan. Based on the weekly tasks that are listed on our development plan, as well as the user-story feature on github, we can accurately track tasks that have been completed on time, and those that have not.

If a team member fails to complete a task by our internal deadlines, we will hear them out on their reasoning. If they fail to provide sufficient reasoning, that will be a strike out of two. If a team member reaches two strikes, then the rest of the group will complain to the TA and have a discussion.

Team Responsibilities - **General division**:

- Kuter Bora:
    - Story Setting - Story background design/bosses battle design
    - Level Design - Player progression system//map design
    - Backend - Weapons/tools function
    - Enemies AI - normal enemies attack mechanism (e.g. detect the player, dash, patrol)
- Mandy Deng:
    - Graphics/Assets - Focusing on enemies/map sprite, openGL rendering
    - UI - Starting screen/option Menu
    - Backend - characters/enemies entities/components
- Jett Stephen Limin:
    - Backend - characters/enemies entities/components
    - Backend - Player health and healing function
    - Enemies AI - normal enemies attack mechanism (e.g. detect the player, dash, patrol)
- Dizhe(Sandy) Xiang:
    - Graphics/Assets - Focusing on the player/map sprite, openGL rendering
    - UI - Exiting screen/option Menu
    - Backend - characters/enemies entities/components
- Anikait Kapur:
    - Physics - Collision Detection
    - Enemies AI - normal enemies attack mechanism (e.g. detect the player, dash, patrol)
- Brian Chu: UI/Backend
    - UI - Font design/Starting screen/option Menu/exiting screen
    - Backend - Player movement/attack system
    - Physics - gravity

Development Plan:

Milestone 1: Skeletal Game

Week 1:

- Entities and Components - **Kuter, Anikait <- Will be done by Monday 11:59pm**
    - We must finish building the entities and components required in the game - this included the Component container and the Entity Manager
- Character Movement System, including walking and jumping as well as Key Mapping for it - **Brian** (Gameplay - Random/coded action)
    - Must finish the system that controls the movement of the character as well as the key mappings for the movement (walking and jumping)
- Collision Detection System for enemies and character - **Anikait** (Gameplay - Simple collision detection & resolution)
    - Must finish the system that deals with detecting collisions between two entities
- Sword functionality - sword key mappings and damage functionality - **Kuter** (Gameplay - Random/coded action)
    - Must finish the system that deals with doing damage to the enemies every time the sword collides with an enemy
- Gravity - **Brian** (Gameplay - Well-defined game-space boundaries)
    - Must finish the system that deals with gravity physics in the game - the system must account for gravity for the player, ground enemies as well as flying enemies who should not be affected by gravity
- Dash functionality - **Kuter** (Gameplay - Random/coded action)
    - Must finish the system that will allow the player to teleport x distance from their current position - x will be a set amount, and it will be -x or +x depending on the direction key the player is holding while clicking the dash button.
    - Must finish the key mapping for dashing.
- Sprites for initial player motions (walk/jump/dash/attack/get damaged/heal/die) - OpenGL - **Sandy** (Rendering - Textured geometry)
- Sprites for goomba enemies (walk, notice, dash, fly (+attack), idle, spit, die) - OpenGL - **Mandy**
- Player getting damaged and healing - **Jett** (Gameplay - Random/coded action)
    - Must finish the system that involves the player getting damaged by enemies and the player healing themselves using the healing flasks

Week 2:

- Start screen - UI Start Screen and Function - **Mandy**, **Brian** (UI, not listed in categories)
    - Must implement a system that allows the user to start the game from the start screen and its corresponding UI.
- Exit game - UI and Function - **Sandy**, **Brian** (UI, not listed in categories)
    - Must implement a system that allows the user to exit the game and its corresponding UI.
- Enemy AI for the “Goombas”: Patrol - **Anikait** (Gameplay - Random/coded action)

    - Must create the system that houses the AI for Goombas' patrol movement
- Maps - Find and implement assets for the background - **Kuter, Mandy**
    - Must find/create assets for the background of the various maps and code them in.
    - Abandoned Roost map - Find and implement assets for the maps and background - **Kuter, Mandy**
    - Deserted cesspit map - Find and implement assets for the background - **Kuter, Mandy**
    - Birdmen town map - **Kuter, Mandy**
- Testing - We will leave the last three days for integration testing - putting the system together and making sure all the individual parts work well together
    - For each feature, we will use the user stories to test it, making sure that we can do everything specified in the definition of done.
    - The User Stories can be found in the Appendix.

Milestone 2: Minimal Playability

Week 1

- Sprites for tools - **Mandy**
    - The sprites for the various tools(wings and flamethrower) in the game
- Enemy AI for the "Goombas": notice player - **Kuter**
    - The system that will allow the Goombas to notice the player when they get within range.
- Tutorial Screen UI and Functionality - **Kuter**
    - The system for displaying a tutorial screen that will show the user the basic controls (Walking, Jumping and Attacking).
- NPC Quests: - **Jett and Anikait**
    - Adding NPC quests to the game
    - NPC quest: birdlander - first location
    - NPC quest: birdlander - second location
    - NPC quest: birlander - attack and death
- Enemy AI for the "Goombas": pathfind towards player - **Anikait**
    - The system that will allow the Goombas to go towards the player once the player has been noticed
- Options (Menu- function and UI) and FPS Counter - **Brian**
    - The UI and functionality for the options menu - will involve exiting the game and changing its volume.
    - System for calculating and displaying an FPS counter
- Wings functionality - **Sandy**
    - The system will allow the player to fly for a given period of time—this will include the key mappings, flying, and glide functionality, which will allow the player to move left or right as they slowly fall down as long as they hold the jump button.
- Birdmen AI - **Jett**
    - The system for controlling the Birdman enemy type - the system will have a similar code to the Goombas AI with the addition of code for dashing and flying back once the enemy reaches its intended destination - for example, if the character starts dashing towards

Point A, it will fly back to its original position once it reaches point A. The pathfinding for this will differ here as the AI should be pathfinding when it's not dashing.

- Breakable poo doors functionality and UI- **Brian**
    - Create a system for the functionality of the poo doors as well as render them in

Week 2

- Sprites for bosses - **Sandy and Mandy**
    - The sprites for all the bosses in the game - idle, walking, attacking and dying
    - Chicken Boss Sprite
    - Birdman Elder Sprite
    - Goldwing dead body sprite
    - Great Bird Sprite
- NPC birdlander sprites - **Mandy**
    - The sprites for the Birdlander NPC - idle and walking/running
- Goldwing Lightning functionality and UI - **Brian and Sandy**
    - The system for lighting damages the player and for throwing lighting at random intervals.
    - The UI for the lighting and its impact on the ground.
- Flamethrower functionality - **Brain**
    - System for the flamethrower
- Platforming Challenges: - **Kuter**
    - Create a system for rendering all the platforming challenges
    - Platforming challenge 1: pipes **-**
    - Platforming challenge 2: over the roost
    - Platforming challenge 3: moving platforms in birdmen town
- Integrate Mesh Based Collision System - **Anikait**
    - Improve the existing collision system with a mesh based system
- Testing - We will leave the last three days for integration testing - putting the system together and making sure all the individual parts work well together

Milestone 3: Playability

Week 1:

- Making the game more memory efficient - **Anikait, Mandy and Sandy**
- Player character dying and returning to checkpoint - **Kuter and Brain**
    - The system for loading and saving at checkpoints involves saving once the player reaches a checkpoint, respawning at the last checkpoint when the player dies, and resetting all the enemies in the level.
    - Saving
    - Loading
    - Death Animation
- Giant Ballista Rendering and Functionality - **Jett**
    - Rendering the great ballista and the system that allows the player to interact with it.
    - The system must also render the

- Render in Goldwing corpse when required - **Jett**
- Testing the game to ensure that it is stable - **Anikait, Kuter and Brain**

Week 2

- Combat Challenge:
    - Create the systems required for the combat challenges (spawning enemies, locking room, damage boost reward)
    - Combat challenge 1: pelican manor
    - Combat challenge 2: Roost ambush
    - Combat Challenge 3: Birdmen Ambush

- Upgrades:
    - Create a System for acquiring combat and health upgrades, rendering them in as well as rendering in the extra heart once the player acquires a health upgrade
    - Acquire combat upgrade
    - Acquire health upgrade
- Testing the game to ensure that it is stable - **Anikait, Kuter and Brain**
- Integration testing- **Everyone**

Milestone 4: Final Game

Week 1:

- Sound effects and Music
    - Add the sound effects for all the relevant areas
- Balancing - Damage dealt and amount/number of time the player can heal so that game is not too difficult
- Balancing - Ensure that the enemy's speed is not too slow or fast.
- Balancing—Ensure that the time it takes for the player to perform actions and use tools (for example, cooldowns) is not too long or short but fits the general speed of the game.
- Implement multiple endings
    - Add a system that allows all the players to choose the ending they like best - they will have two options.
- NPC Sprite:
    - Ogre
    - Retired Birdman
    - Pelican - corpse
    - Pelican - dialogue
    - Pelican - move to manor
- NPC quests: Ogre and Retired birdmen
    - Add systems to implement all the NPC quests as well as render their relevant assets

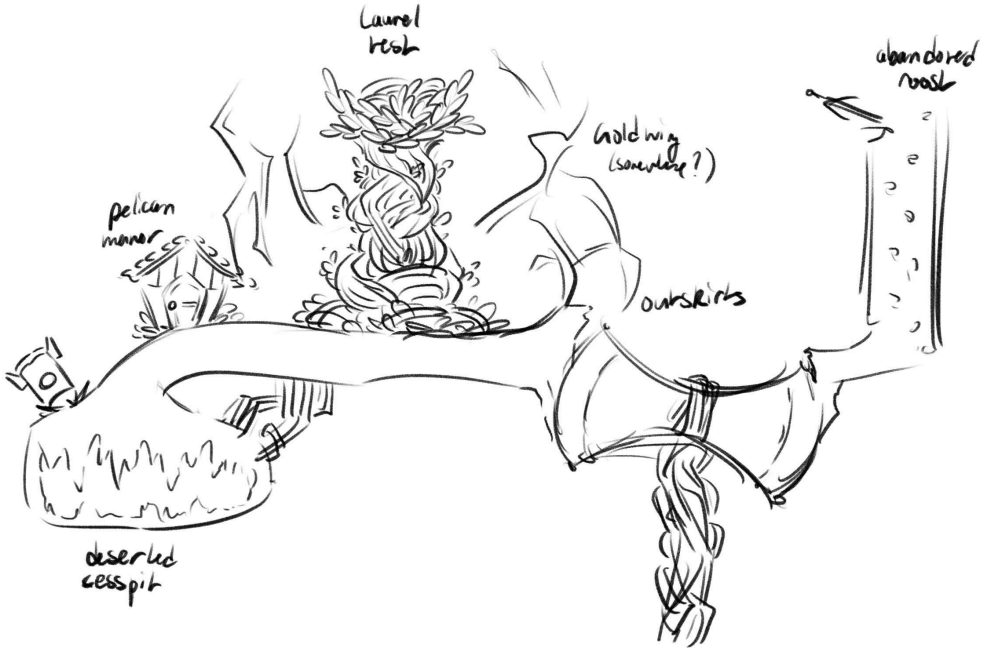
Week 2

- Integration testing for all the new features
- Test stability and Robustness of the game
    - Making sure there is no lag
    - Make sure unexpected input doesn't cause any bugs or unexpected behavior.
    - Memory Management
- Go over the bugs list and ensure that none of them remain

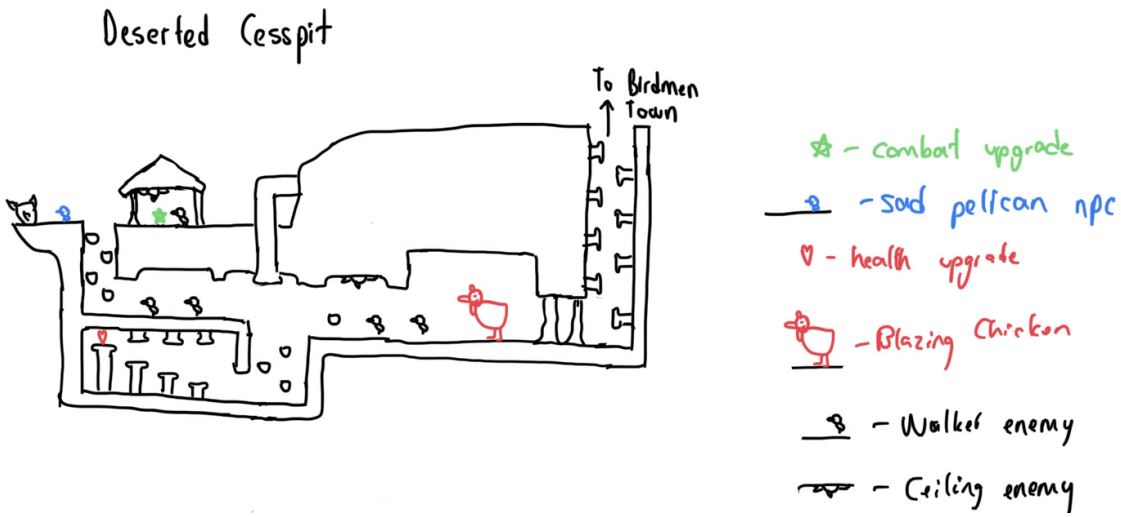
Appendix

Images, Art, and Scenes

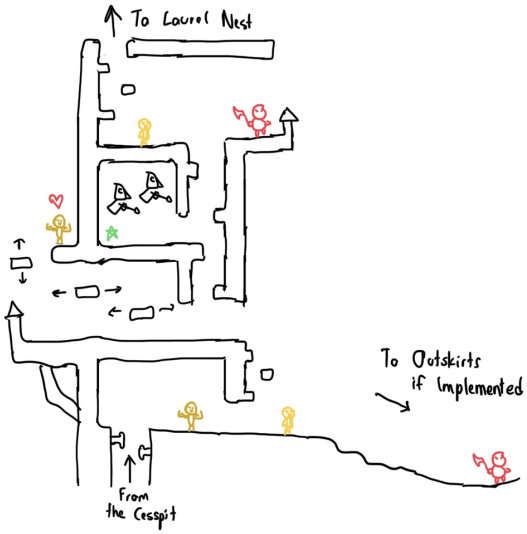
[1] Overall Game Map

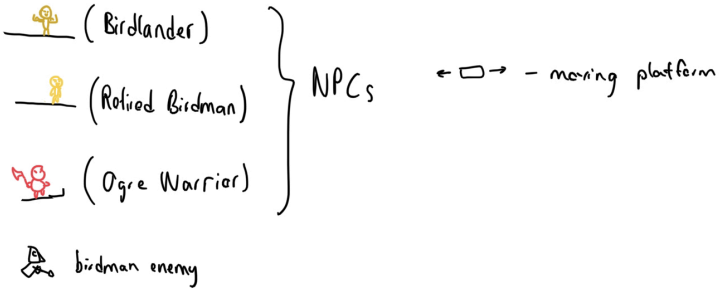


[2] Deserted Cesspit Map

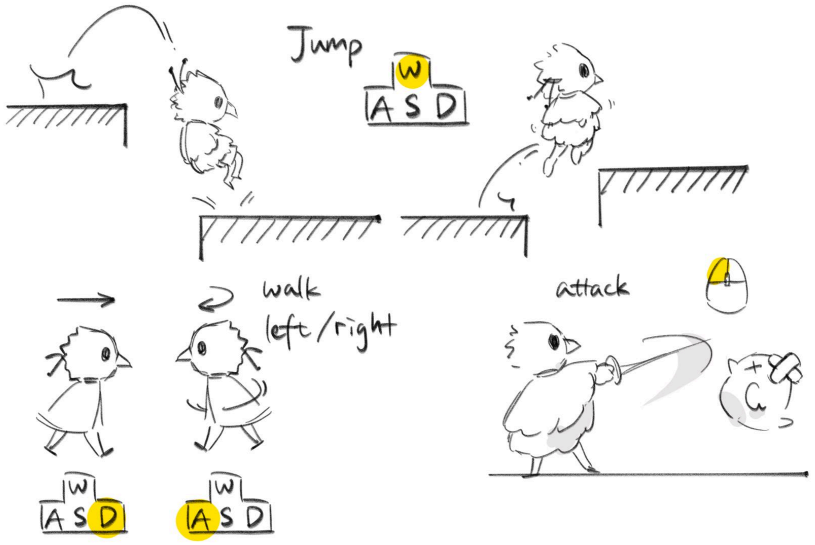


[3] Birdmen Town Map

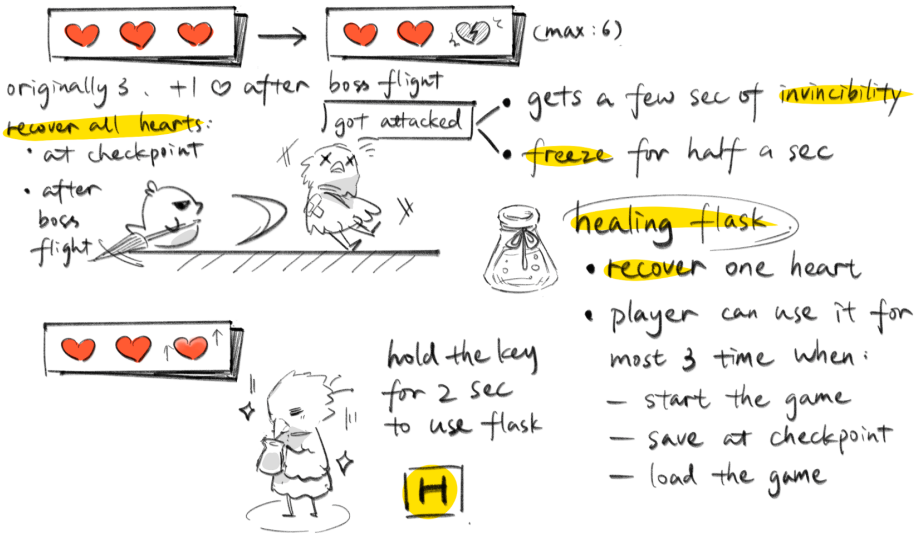




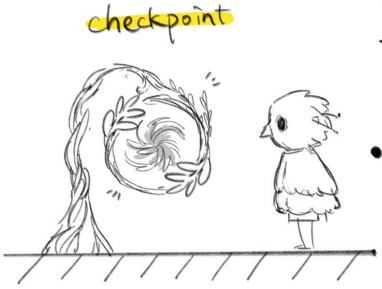
[4] General Player Movement and Character



[5] Heal Functionality

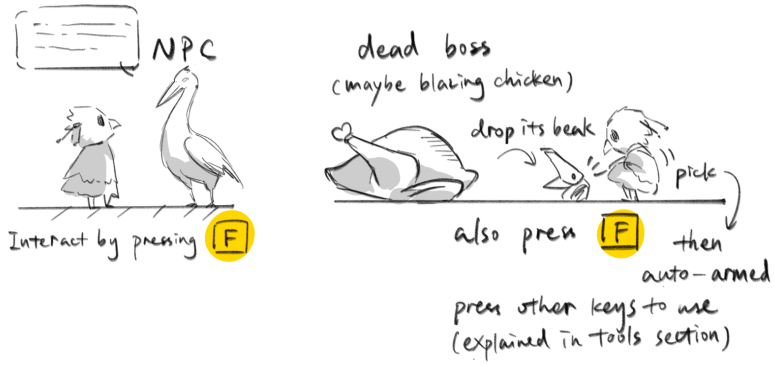


[6] Checkpoints

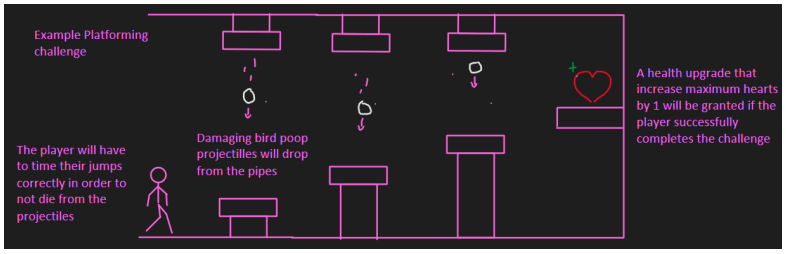


- The player can only save their progress at the checkpoint (once they stand close enough)
- When they load, they appear at the last checkpoint they saved at. (And the previous enemies would be recovered)

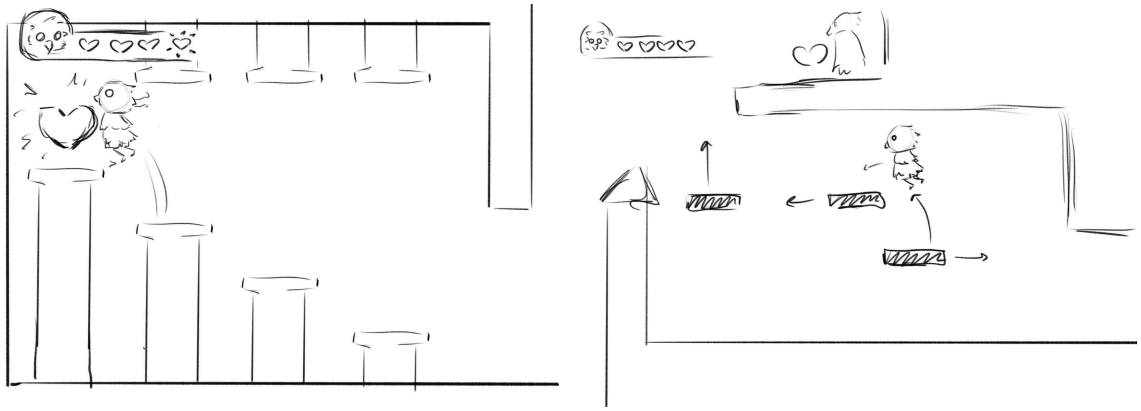
[7] Interact with NPC/pick up item



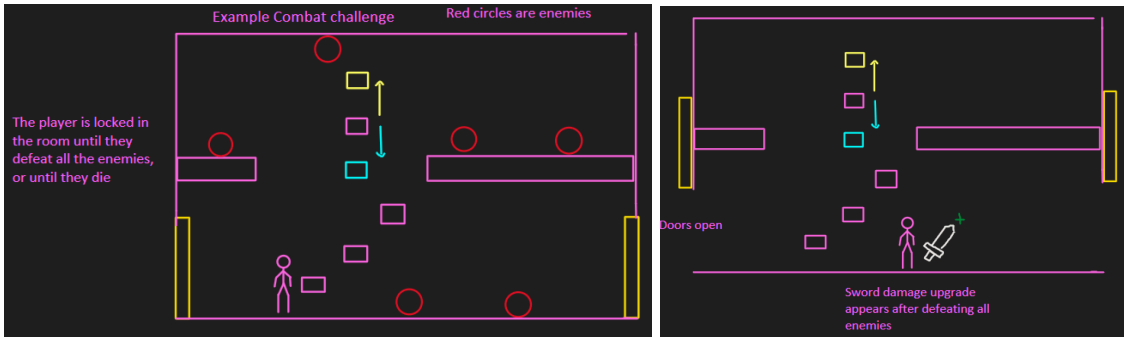
[8] Platforming Challenge (Abstract)



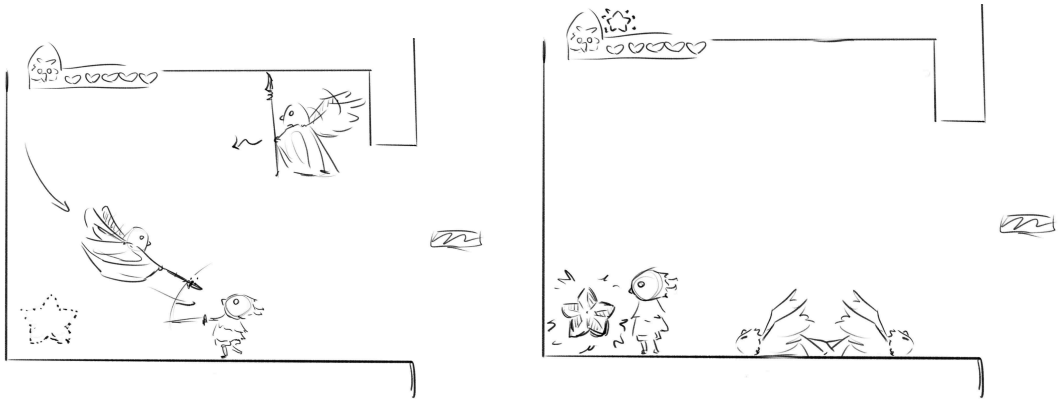
[9] Platforming Challenge (Scene)



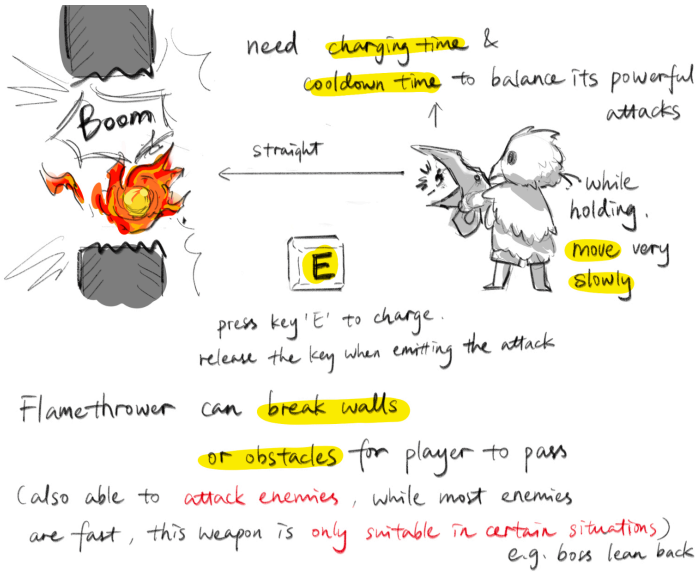
[10] Combat Challenge (Abstract)



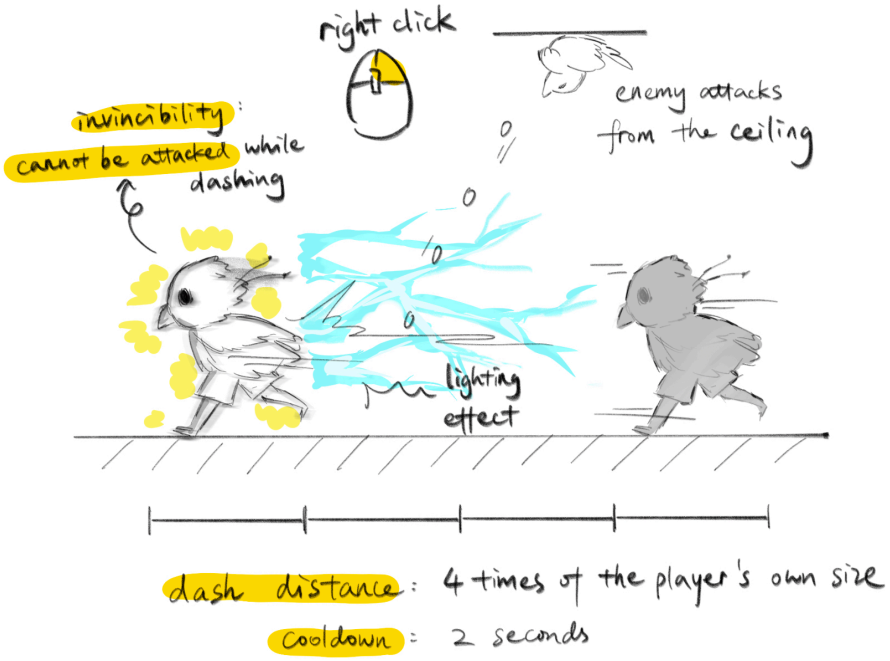
[11] Combat Challenge (Scene)



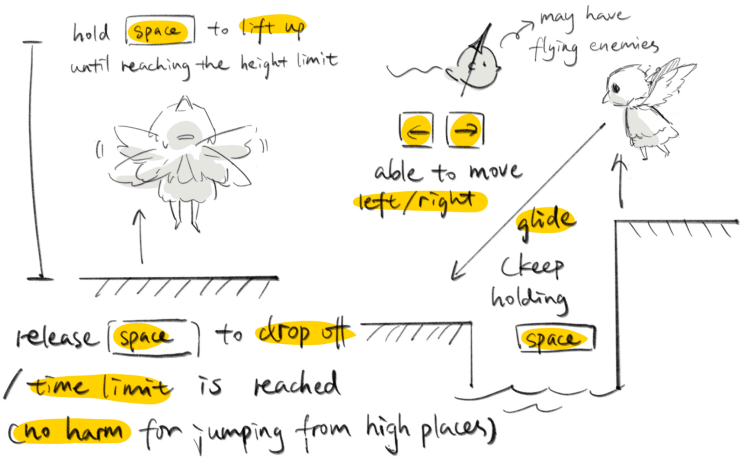
[12] Flamethrower Beak



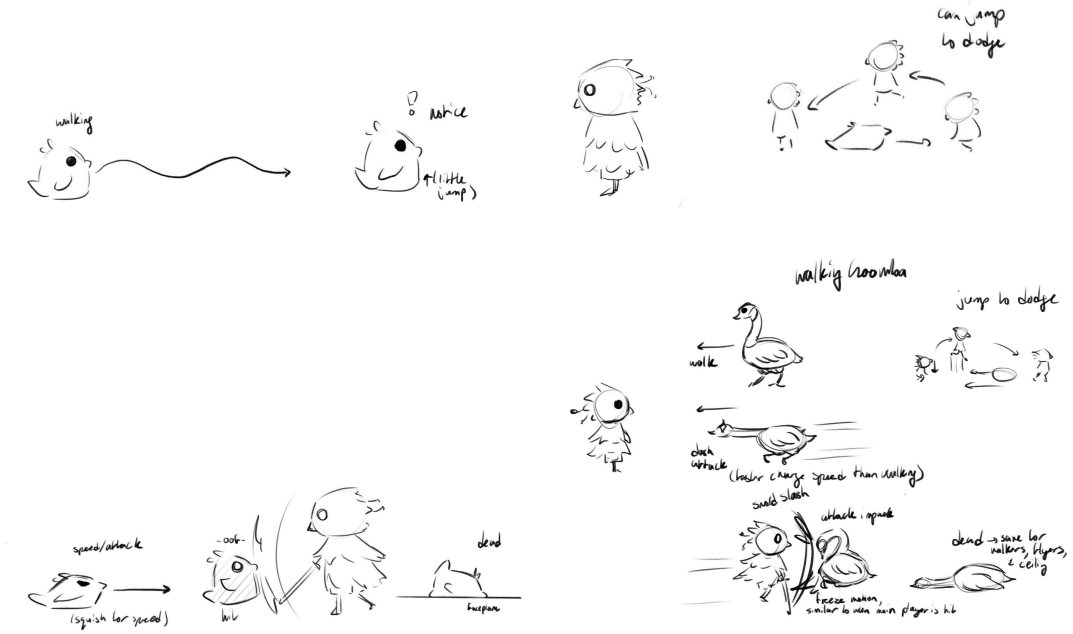
[13] Dash



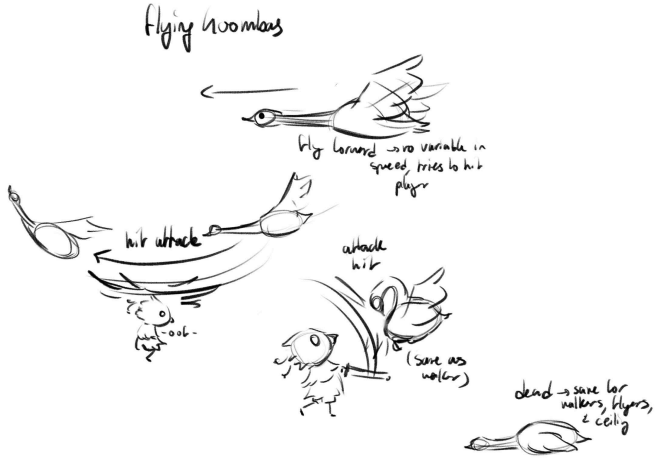
[14] Wings



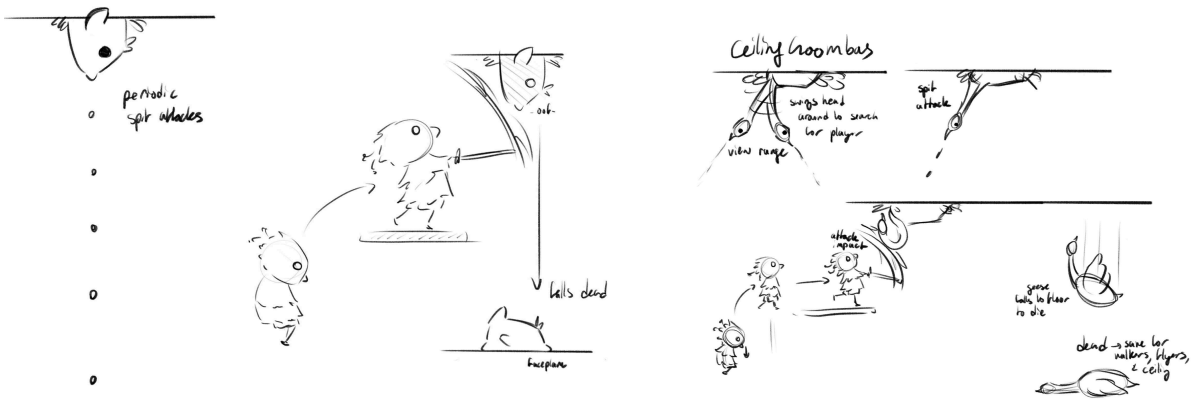
[15] Walking Goomba Enemies



[16] Flying Goomba Enemies

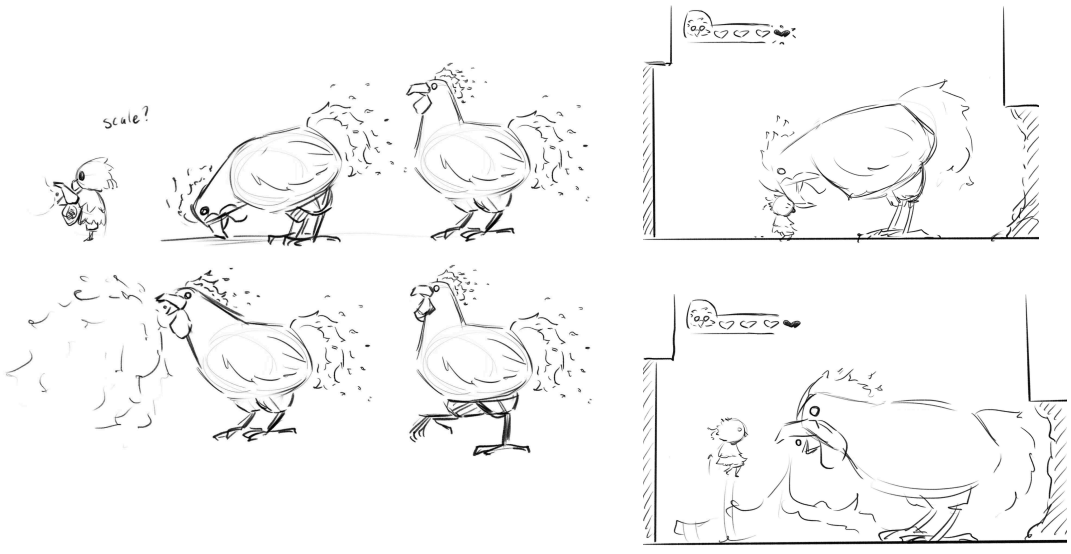


[17] Ceiling Enemies



[18] Birdmen Enemies

[19] Blazing Chicken

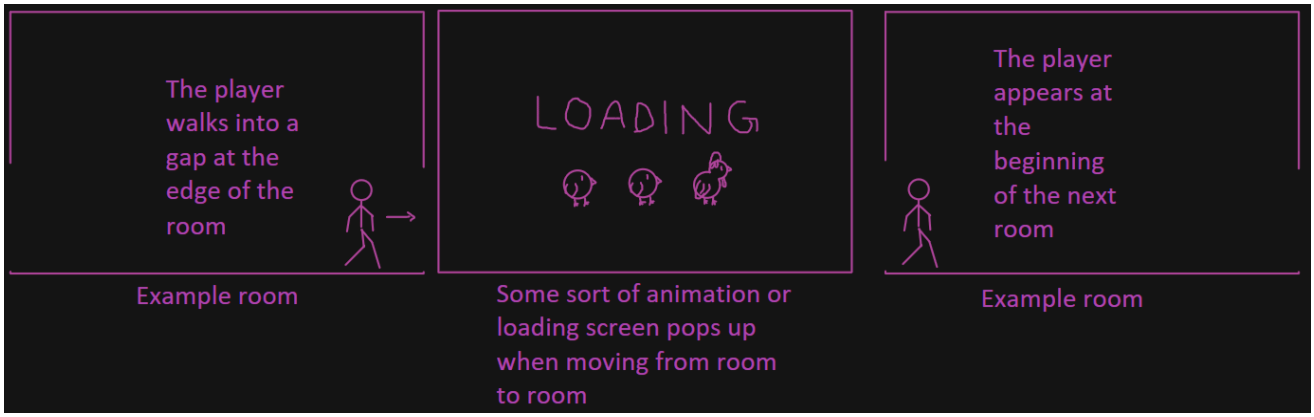


[20] Goldwing

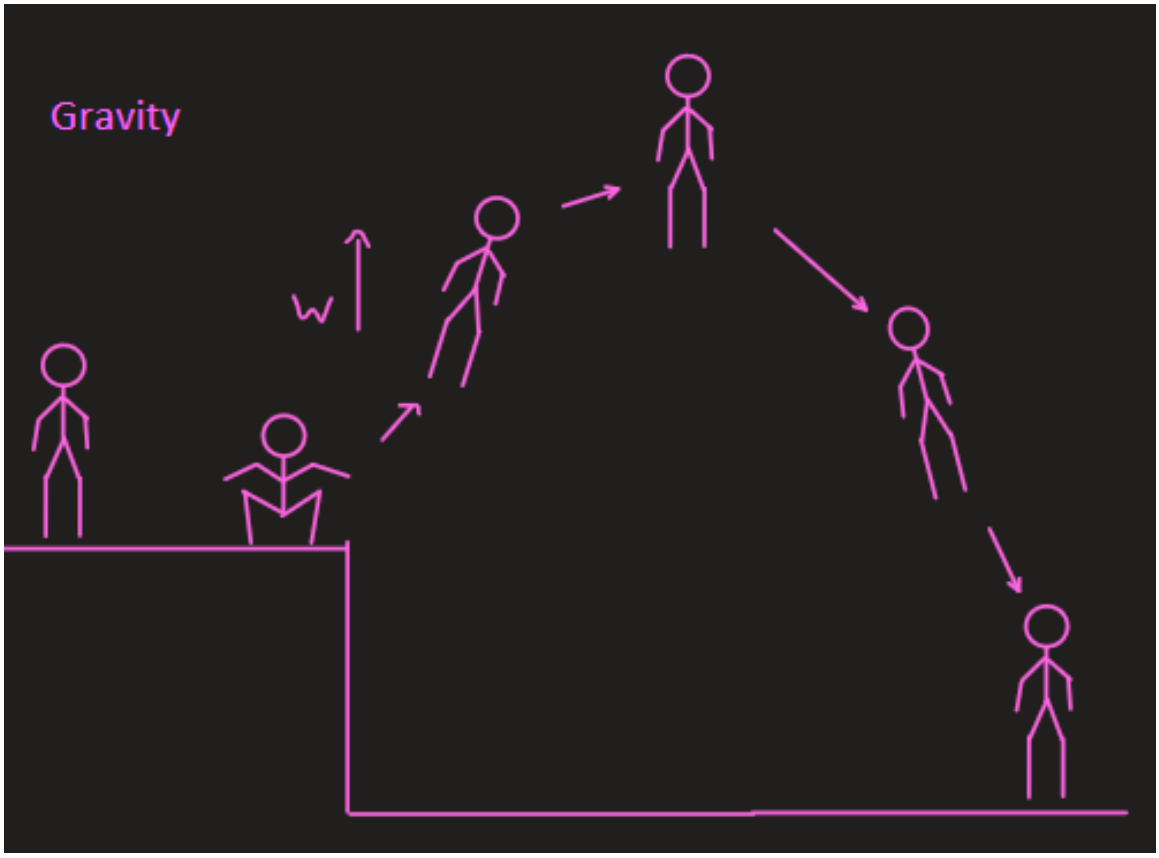
[21] Birdman Elder

[22] Great Bird

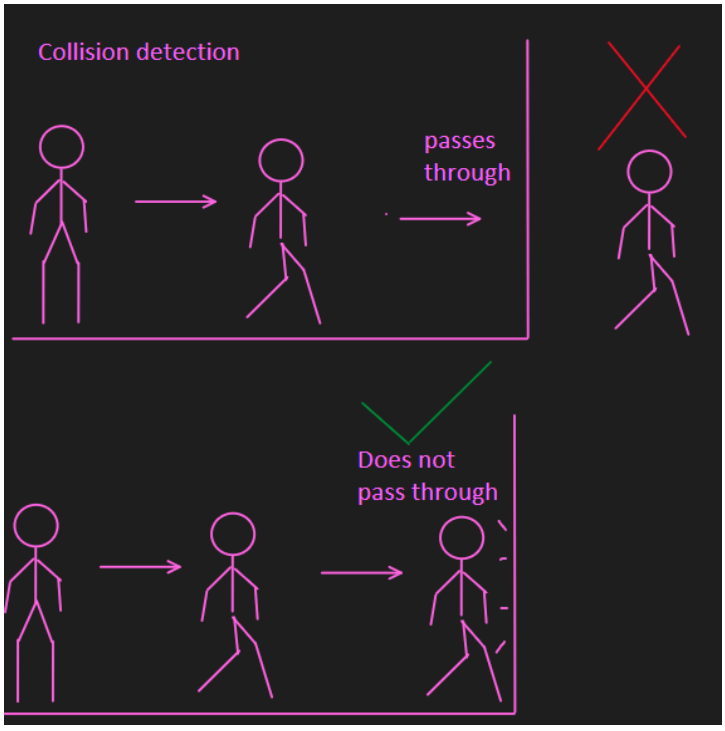
[23] Loading



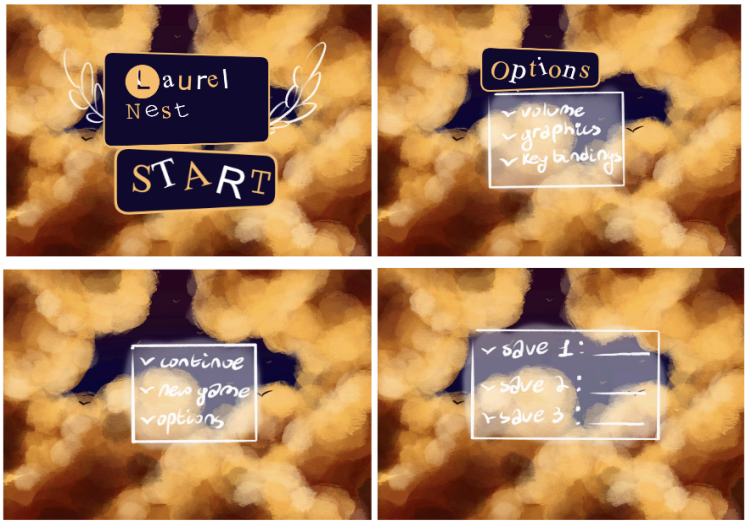
[24] Gravity



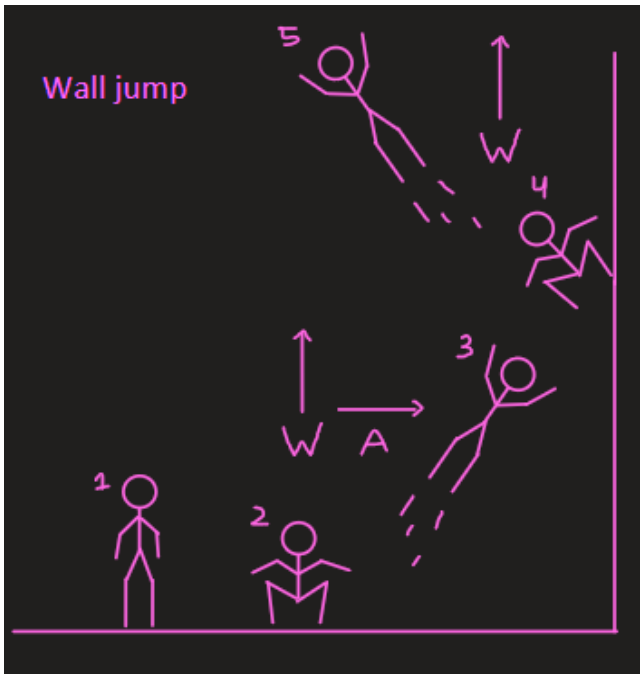
[25] Collision



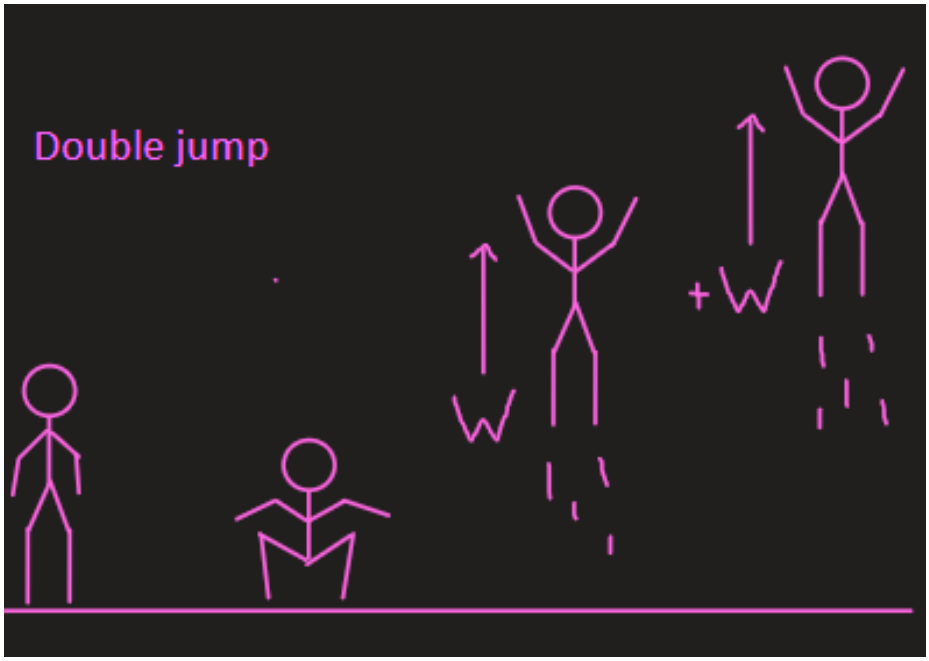
[26] UI



[27] Wall Jump

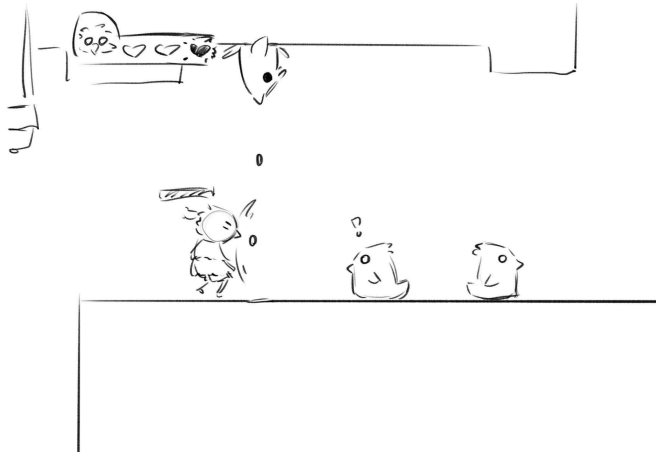


[28] Double Jump

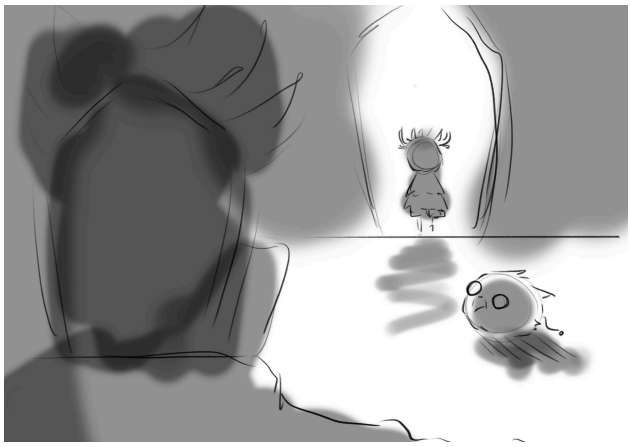


[29] Game Begins Scene

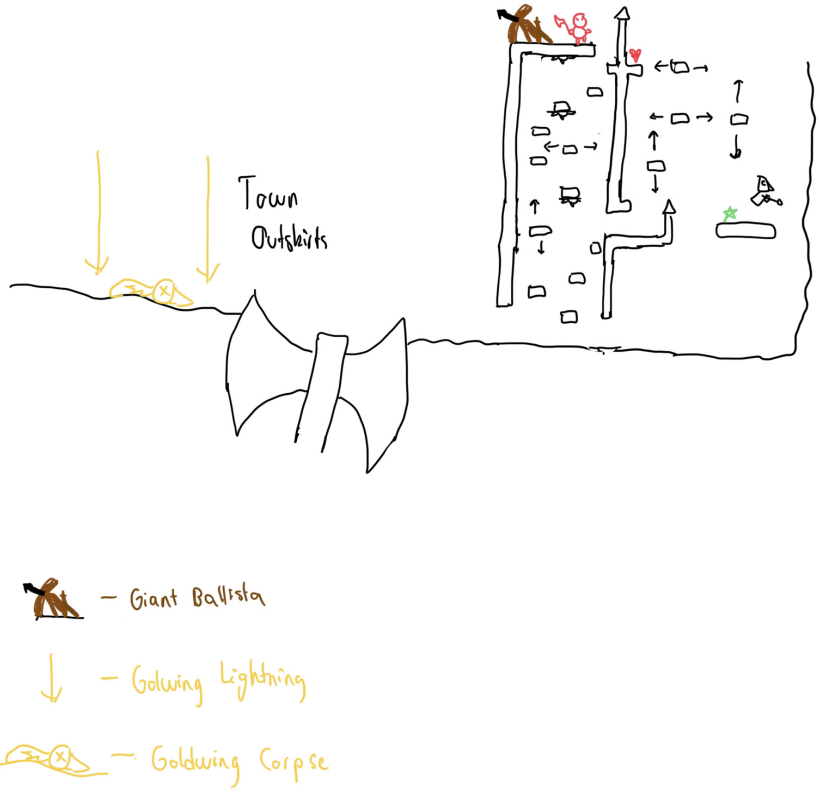
[30] Cesspit Scene



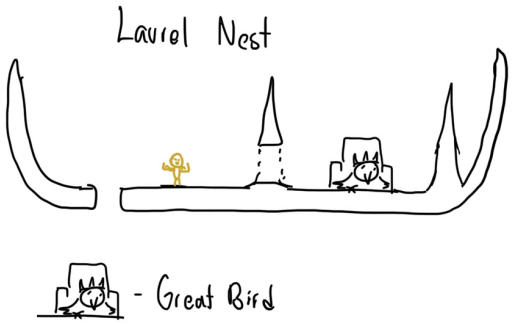
[31] Endings



[32] Outskirts Map



[33] Laurel Nest Map



User Stories

Gameplay as user stories:

- **Initial Character movement:** As a player, I want to be able to move left and right so that my player character can get from point A to point B - image
    - Definition of done:
        - Moves when you press the buttons.
        - Movement feels smooth.
        - Movement respects environmental boundaries, such as walls
        - Movement is properly animated
- **Initial Character movement:** As a player, I want to be able to jump so that my player character can get onto higher areas. -land on platforms - image
    - DOD:
        - Jumps when appropriate button pressed
        - Jumping and falling obey Newton's laws
        - Cannot jump in midair (without appropriate item)
        - Can turn around in midair
        - Jumping is animated and appears smooth
- **Sword functionality:** As a player, I want to be able swing my sword/attack so that my player character can defeat enemies -
    - DOD:
        - Can attack by pressing appropriate button
        - Attack is animated and appears smooth
        - Feedback for successful and unsuccessful hit
        - Enemy is damaged by hit
        - Attack has appropriate cooldown animation
- **Enemy AI for the "Goombas": patrol:** As a player, I want the enemies to walk around aimlessly so that my player character has to avoid them not to get hit
    - DOD:
        - Enemy walks back and forth.
        - Enemy turns around when it encounters a wall.
        - Enemy damages the player upon collision
        - Enemy's movement is smooth and animated
        - Player can damage the enemy
        - Enemy dies when it has no health left
- **Enemy AI for the "Goombas": notice player:** As a player, I want the enemies to be able to "spot" the player, so that there is some level of difficulty
    - DOD:
        - Enemy can see the player at a certain distance
        - Visual or audio cue that the enemy has seen the player
        - Enemy's behavior changes after seeing a player, and begins to chase or attack
        - Enemy loses visual of player and goes back to patrol
- **Enemy AI for the "Goombas": pathfind towards player:** As a player, I want the enemies to be able to walk towards the player so that my player character has to either kill them or run away.

    - DOD:
        - Uses pathfinding algorithm to find the player
        - It won't get lost
- **Player getting damaged and healing:** As a player, I want the enemies to be able to damage me so that I have to play well in order to not die
    - DOD:
        - The player loses a heart when hit by an enemy
        - Player becomes invulnerable for a short amount of time after being hit
        - Player's health is in a UI
        - Player respawns at a checkpoint when they lose all their hearts
- **Breakable poo doors:** As a player, I want there to be breakable obstacles so that I have an incentive to defeat the boss that drops the item that breaks the obstacles
    - DOD:
        - The Player is able to use the Flamethrower to break the doors
        - Once the doors are broken, the player has access to the areas blocked by the door
        - The Player should not be able to access the area if the door is not broken.
- **Flamethrower functionality:** As a player, I want to be able to use the flamethrower/Blazing Chicken's beak for a brief period to break down obstacles and to use as a weapon
    - DOD:
        - The Player is able to charge the flamethrower and use it to damage enemies
        - The enemies affected by the flamethrower must be damaged heavily
        - The Player should not be able to use the flamethrower during the cool-down period.
- **Dash functionality:** As a player, I want to be able to quickly move/teleport a short distance in front of me, so that I can traverse the map faster
    - DOD:
        - The player can press right-click to move quickly in one direction
        - The player can dash left and right
        - The player can dash while in the air
        - The player is given invincibility frames while in the action
        - The dash functionality has a short cooldown
- **Dash functionality platforming:** As a player, I want to be able to dash in order to get to point C from point A that isn't accessible from regular walking and jumping
    - DOD:
        - The player is able to use the Dash functionality to access an area of the platforming challenge.
        - The area must be inaccessible through any other means.
- **Flying functionality:** As a player, I want to be able to fly for a brief period.
    - DOD:
        - The player is able to hold the jump button to fly
        - The player is able to glide after flying by holding the jump button
        - The player is able to move to either side while gliding
        - The player is not able to fly once the maximum duration is reached

- **Flying Functionality Platforming:** As a player, I want to be able to fly to get to point D from point A, which isn't accessible from regular walking, jumping and dashing
    - DOD:
        - The player can not finish the platforming challenge without flying.

- **Goldwing Lightning:** As a player, in the Town Outskirts, I want there to be lightning that periodically shoots from the sky so that I have to move in a calculated way that isn't boring.
    - DOD:
        - Visual and audio cue when lightning strikes
        - Lightning damages the player
        - Lightning is fair so the player can still progress with skill
- **Giant Ballista interaction:** As a player, I want to be able to interact with the giant ballista at the top of the Abandoned Roost so that I can kill the Lightning Bird and reach the Birdman Tower
    - DOD:
        - The player can interact with the ballista by pressing the appropriate key when in range.
        - When the ballista is fired, there is a clear visual animation of the ballista launching the projectile.
        - A sound effect plays when the ballista is fired.
        - Upon firing, there is a cue indicating the Lightning Bird has been hit.
        - The Lightning Bird is defeated, and its corpse appears in the Town Outskirts upon the player's return.
        - The player gains access to the path to Birdman Tower, which is now open and traversable.
        - The ballista cannot be fired again after the Lightning Bird is defeated.
- **Player getting damaged:** As a player, I want to be able to lose hearts when an enemy attacks me so that I have to use some skill in order to not die and restart at the beginning of the area
    - DOD:
        - When hit by an enemy attack, the player loses health.
        - The player is frozen in place for half a second and invincible while they are frozen.
- **Player healing:** As a player, I want to be able to use an item that restores my health back to full so that I do not have to worry as much about dying
    - DOD:
        - The item has three uses total throughout the entire game
        - The healing takes 3 seconds to complete before receiving the health
        - The player gets all their health back when they use the healing flask.
- **Player character dying and returning to checkpoint:** As a player, I want to be able to set a checkpoint so that when I die, I can return to that checkpoint and not the very beginning
    - DOD:
        - When a player dies, they respawn at the last checkpoint.
        - The player has the option to set set the checkpoint that they are standing on to their current checkpoint

- **Birdmen AI - all:** As a player, I want there to be flying enemies that try to attack me so that I have to use more skill in order to kill them
    - DOD:
        - A flying enemy attacks the player when the player is in range.
        - The enemy is killed once they take enough damage.
- **Acquire health upgrade:** As a player, I want to be able to receive upgrades to increase my health so that I can be a bit more lenient in how I play.
    - DOD:
        - The player gets an additional heart when they finish a platforming challenge.
- **Acquire combat upgrade:** As a player, I want to receive upgrades to increase my damages and receive a satisfying reward after completing the combat challenges.
    - DOD:
        - The player's sword damage is increased by a set amount once they finish a combat challenge.
- **Collision Detection - Use just a Collision System:** As a player, I want to be able to stand on top of platforms and not be able to go through walls, so that I can properly move in the game and not fall through the map.
    - DOD:
        - The system must be able to detect when an enemy collides with the player.
        - The system must also be able to detect when the player collides with the boundary of the level/room.
        - The system must be able to detect when the player collides with a solid object within the game.
- **Platforming Challenge 1:** As a player, I want to be able to complete the platforming challenge in the deserted cesspit which involves jumping over pipes while avoiding ceiling enemies. Once finished I want to be able to pick up the health upgrade (even if it does not work just yet).
    - DOD:
        - The pipes should be spaced apart in such a way that the player must execute a combination of short and long jumps.
        - There should be ceiling enemies present that shoot or drop projectiles, which the player must avoid or time jumps to dodge.
        - There should be a health upgrade at the end of the challenge that the player can pick up.
- **Platforming Challenge 3:** As a player, I want to be able to complete the platforming challenge in birdmen town which involves a long room and moving platforms that require precise timing. Once finished I want to be able to pick up the health upgrade (even if it does not work just yet).
    - DOD:
        - There should be moving platforms in the room, moving horizontally and requiring the player to time their jumps.
        - There should be one platform moving vertically that the player must stand on to reach the health upgrade.
        - There should be a health upgrade at the end of the challenge that the player can pick up.

- **Combat challenge 1: pelican manor**: As a player, I want to be able to break the doors of the pelican manor and enter it. Inside, if I have done the required NPC quest, I want the combat challenge of the area to begin by locking the doors and spawning in enemies. Upon defeating the enemies, I want to be able to acquire the combat upgrade. (even if it does not work just yet).
    - DOD:
        - There should be a check to make sure the player has done the NPC quest first. This requirement should be communicated to the player.
        - There should be a visual or audio cue that the combat challenge is beginning.
        - The challenge should spawn one or more waves of enemies that increase in difficulty.
        - Once the challenge is completed it should spawn the combat upgrade which the player can then pick up.
- **Combat challenge 3: Birdmen Ambush:** As a player, I want to be able to enter the ambush room in the Birdmen Town. Inside I want the combat challenge of the area to begin by locking the doors and spawning two Birdman enemies. Upon defeating the enemies, I want to be able to acquire the combat upgrade. (even if it does not work just yet).
    - DOD:
    - There should be a check to ensure the player has done the NPC quest first. This requirement should be communicated to the player.
    - There should be a visual or audio cue that the combat challenge is beginning.
    - The challenge should spawn one or more waves of enemies that increase in difficulty.
    - Once the challenge is completed, it should spawn the combat upgrade, which the player can then pick up.
- **NPC quest: Pelican - dialogue:** As a player, I would like to talk to the Pelican NPC at the beginning of the game and get some context about the story. When I talk to him after defeating the Blazing Chicken, I want him to mention how his house has been blocked.
    - DOD:
        - The player can initiate the dialogue with the pelican when prompted by pressing a key.
        - The pelican should speak in an appropriate manner.
        - Dialogue boxes are displayed, and they will transition to the next dialogue box when prompted, and the user can skip the dialogue when prompted.
- **NPC quest: Pelican - move to manor:** As a player, once I break down the door to the Pelican Manor, I want the pelican NPC to have new dialogue that thanks me. Next time I visit him, he should have moved away.
    - DOD:
        - There should be a different set of dialogue when the side quest has been completed.
        - When I leave the area and come back, he will have left.
- **NPC quest: Pelican - corpse:** As a player, once I defeat the combat challenge in Pelican Manor, I should be able to see the Pelican NPC’s corpse in the manor’s back room.
    - DOD:
        - After the player defeats the enemies in the Pelican Manor combat challenge, the Pelican NPC's corpse appears in the back room of the Manor.

        - The corpse sprite should be visually distinct and fit the setting.
        - It should still be obvious they're the same character.
- **Render in Goldwing corpse when required:** As a player, after I interact with the giant ballista, I want to be able to see the giant corpse of Goldwing on the Town Outskirts.
    - DOD:
        - When the player returns to the Town Outskirts, the giant corpse of Goldwing should be visible, lying on the ground in a dramatic pose.
- **Entities and Components:** As a player, I want there to be separate entities with their own characteristics so that I can interact with other things in the game.
    - DOD:
        - All entities and components have been implemented
- **Physics (Gravity):** As a player, I want to be able to fall as if there was gravity in the game so that I am not just floating in the air.
    - DOD:
        - When the player jumps, and there is no platform immediately underneath their feet, they will fall to the ground until they land/collide on a platform
- **Start Screen function:** As a player, I want there to be an intermediary screen when I load up the game so that I am not unexpectedly thrown straight into the game
    - DOD:
        - There is a "Play" button that brings you to another selection: "Continue", "New game", "Options"
- **Start Screen function (Continue)**: As a player, when I select "Continue", I want to be able to see my save files that I am currently playing, so that I can choose which run I can pick.
    - DOD:
        - There is a "Continue" button that when selected, will bring up a 3 save files
        - When clicking on a save file that is currently being played, will load up that run through
- **Start Screen function (New Game):** As a player, when I select "New Game", I want to be able to play start an entirely fresh game so that I can play an entirely new game
    - DOD:
        - There is a "New Game" button that when selected, will create an entirely new run though with zero progress
- **Options functionality (Volume):** As a player, when I select "Volume" from the Options screen, I want to be able to adjust the music and sound fx so that I won't wake up my family when I am playing this game late into the night
    - DOD:
        - There is a "Volume" button that when selected, will bring up two adjustable bars for music and sound fx
        - Sliding the bars to the right will increase volume for music/sound fx
        - Sliding the bars to the left will decrease volume for music/sound fx
        - Can't slide beyond limits.