# Concept

Genre: ‎Roguelike

## Core Gameplay

* First person camera
* Dispatch groups of enemies in melee combat with a sword and a combat skill
  + Combat skills act differently in the air vs on the ground
* Traverse medium-large maps using various mobility options
* Kill bosses to become stronger
* Complete level challenges to obtain special rewards

## Design Philosophy

* Gameplay should feel fast
* Player should have to focus to succeed in combat, requiring at least some concentration
* Choices should be given to the player outside of combat, giving them some respite
* Mobility should feel fun, fast, and natural
  + Player should retain control of their character while in the air

## Specific Design Elements

* Player will be equipped with standard mobility options
  + Ex. walk, jump, sprint
* Player will also have access to a double jump by default
* Player will have access to various skills for either combat or mobility
* Standard levels will be procedurally generated
  + Enemies will favor spawning apart
  + Enemies will be weak, but deal considerable damage
* Boss levels will be pre-made
  + Bosses will have much more health and be more punishing
  + Bosses will have a clear weakness
* I am a strong coder, but a weak artist
  + Most code will be made from scratch
  + Almost all assets will be sourced from the unity store
    - This includes art, music, SFX, and textures

## Combat Skill Concepts

Grounded Skills

* Charged Slash
  + General: holding the attack button charges a powerful slash. Release the attack button to fire a wide ranging slash.
  + Use case: area control
* Overhead Chop
  + General: raise your sword high and bring it down with all of your might; gather strength for a moment before dealing a % of the targets hp
  + Use case: taking down stronger targets like elites and bosses

Aerial skills

* Spinning Strike
  + General: spin your camera to increase the power of the slash
  + Use case: high risk high reward way of dispatching small groups of enemies
* Sky Cleave
  + General: power scales with height, instantly executes most enemies above a certain height
  + Use case: executing the strongest enemy in a pack before an engagement

## Mobility Skill Concepts

* Charged Leap
  + General: Hold the jump button while running to gather strength and release to jump extremely high.
  + Use case: gain air to make full use of aerial combat skills
* Grapple Hook
  + General: fire the hook at something in the environment and swing from it, release the hook at any time to carry momentum
  + Use case: scouting the level before committing to an engagement

## Imbuement Concepts

* Lightning
  + Intended feel: extremely fast - pure offense
* Fire
  + Intended feel: fast, but with a focus on raw power and area damage
* Ice
  + Intended feel: weighty combat with high single target damage - should reward good play

## Bare Minimum Requirements

* Replayability
  + After a ‘run’ ends, (regardless of death or victory) the player should be able to seamlessly start a new one
* At least one type of enemy to fight
* At least one boss to fight
  + Defeating the final boss will likely serve as the win condition
* A basic attack
* A ground based skill
* An aerial skill
* A mobility skill

## Personal goals for this project (on top of the bare minimum)

* An elemental imbuement
* At least 2 distinct enemy types
* Challenges that can be accepted or rejected
  + Potential challenges
    - Kill an elite enemy
    - Do not get hit
    - Do not use your mobility skill
  + Rewards for success
    - Elemental imbuement is the only viable reward unless more content is added
  + Punishment for failure
    - Lose some hp
* Have at least 2 non boss levels

## Stretch goals

* Randomly generate non-boss levels
* Have at least 2 bosses
* Have at least 4 types of enemy
* At least 2 of each skill
* A reward for winning a run

# Technical Writeup and Challenges

## 3D Environment

As of now, the 3D environment has not been fully designed. The majority of my development effort in this area has been spent on trying to prepare the procedural generation. Normally, procedural generation would be incredibly daunting for someone in my position. However, I had the fortune of acquiring an extremely useful tool in a cheap bundle a few months ago. Map Graph is a tool made by Insane Scatterbrain that allows you to procedurally generate 2D tilemaps. Here is the procedure that I created using their tool:

1. Generate a map made entirely of floor tiles
2. Randomly fill 45% of the map with wall tiles
3. Border the map with walls so the player cannot fall off the map
4. Smooth the random noise into a cave-like structure
5. If there are any ‘Islands’ made of floor, connect them to the main section of the cave
6. Place the player on a random floor tile
7. Create a safe zone around the player where enemies cannot spawn
8. Place enemies on 2% of the remaining floor tiles

Here is the procedure in the Map Graph GUI: <https://drive.google.com/file/d/1NZswmRfo4UJ-hHSkSQ4ky2OxhNFEBvwU/view?usp=share_link>

I also placed the tilemap in the testing playground with a button that allows you to generate a new map.

In this case, the red brick tiles are walls, the grey tiles are floor, the white ghosts are enemy spawn locations, and the stick figure is the hero’s spawn location. This demo is mostly meant to show that the generation can be done at runtime. My next steps will be to spawn objects at the tile locations to convert the tilemap to a playable 3D environment. Additionally, I’m planning to design a static boss stage on a mountainside or a plateau.

## UI

Currently, the UI consists of two things: a health bar (green, top-left), and a stamina meter (blue, bottom-middle). They are both fully functional and scale with the screen’s resolution.

While it’s technically not a part of the UI, it’s worth nothing that I also added a cursor to the middle of the screen for convenient interactions with things like buttons. This UI is very incomplete however, and my next steps will be to add abilities, ability cooldown indicators, and a minimap. The minimap will be in the bottom left corner of the screen, and the stamina bar will be moved up to make room for ability indicators. After these are done, the UI will be complete, save for the menus. The menus are not yet implemented at all, as I have placed a low priority on them.

## Character Controller and Scripting

The character controller has been built completely from scratch. Full source code can be found here: <https://drive.google.com/file/d/1eU2805D9rBUmWFVGu5_smHK2K-EuKuuT/view?usp=share_link>

These are the current functions provided by the character controller:

1. Looking around (mouse controlled)
2. Walking (WASD)
3. Consuming stamina to sprint (Shift)
4. Jumping (Space)
5. Multi-Jumping (Space)
6. Interacting with objects (E)
7. Tracking resources like health and stamina

These are some things I still want to implement:

1. Skills
2. Combat
3. Death

I chose to build my own character controller instead of using the Unity Essentials one because mobility is very important for my game, and I wanted complete control over the movement of my character. Broadly speaking, I also chose not to change my character’s transform component at all, instead preferring to use the Unity physics engine to move them. What this really means is that to walk or run, I calculate a total velocity vector using the user’s input and apply it to the player capsule, and to jump, I apply a vertical impulse.

void move(float x, float y)

{

rb.velocity =

((transform.right \* x + transform.forward \* y) \* moveSpeed)

+ (transform.up \* rb.velocity.y);

}

void OnJump(InputValue value)

{

if (allowedJumps > 0)

{

// Reset vertical velocity to 0 before launch

rb.velocity = new Vector3(rb.velocity.x, 0, rb.velocity.z);

rb.AddForce(Vector3.up \* jumpPower, ForceMode.Impulse);

allowedJumps--;

}

}

The hardest challenge I faced was implementing the interact feature. On the surface, it seems easy, I cast out a ray from the player camera, check if it collides with an interactable, and if it does, you interact. However, I quickly realized that I had no idea how to actually activate the interactable. The first thing I tried was adding a component to all interactables that would do something, but then all interactables would need the same component, which is rather problematic if you want two interactables to do two different things. It took me a while to figure it out, but eventually I realized that the best way is actually to mimic the new Unity input system. You see, when you press a key that corresponds to an action in Unity, it sends a signal. For example, pressing the ‘jump’ key, sends an OnJump() signal. When you try to interact with an object using my controller, it sends an Activate() signal. As long as all of my interactables have an Activate() method, everything works!

// Send an activate signal to interactables that the player is looking at

void OnInteract()

{

// Cast a ray from the player's camera and check if it is colliding with an interactable object

Physics.Raycast(cam.position, cam.forward, out RaycastHit hit, 2.5f);

if (hit.collider.gameObject != null)

{

Debug.Log(hit.collider.gameObject.name);

hit.collider.gameObject.SendMessage("Activate", null, SendMessageOptions.DontRequireReceiver);

}

}

// This method is from the button that generates a tilemap

private void Activate()

{

if (active)

graphRunner.Run();

}

# Citations

## Assets (Paid assets were obtained via Humble Bundle)

* <https://assetstore.unity.com/packages/3d/props/weapons/katana-sword-free-143768>
  + Used for sword model
* <https://assetstore.unity.com/packages/essentials/starter-assets-first-person-character-controller-196525>
  + Used for testing playground
* <https://assetstore.unity.com/packages/audio/sound-fx/creatures/monster-sounds-pack-176744>
  + Used for enemy SFX
* <https://assetstore.unity.com/packages/3d/characters/creatures/monsters-ultimate-pack-02-cute-series-179083>
  + Used for enemy models
* <https://assetstore.unity.com/packages/tools/utilities/map-graph-177023>
  + Used for procedurally generating level maps
* <https://assetstore.unity.com/packages/audio/music/orchestral/total-music-collection-89126>
  + Used for ambient and level music

## Resources

* <https://www.youtube.com/c/SebastianLague>
* <https://weeklyhow.com/how-to-make-a-health-bar-in-unity/>
* <https://mapgraph.insanescatterbrain.com/manual/creating_your_first_map_graph.html>
* <https://learn.unity.com/>

## Inspirations

* Attack on Titan (Anime)
* Prototype 1 & 2 (Activision Games)