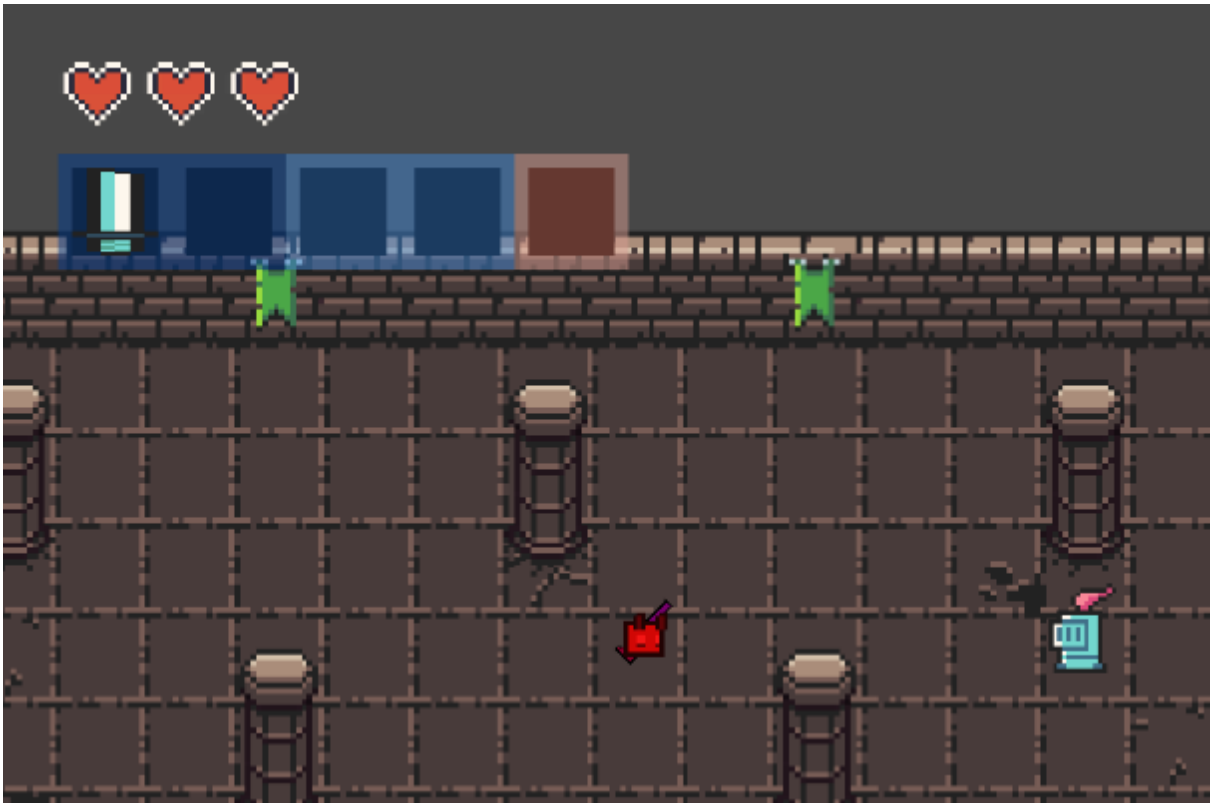
Functional Design

Samurai Shampoo

v 1.0



Coach: Martijn ter Schegget
Minor: HBO-ICT Games Programming

Authors:

Student	Student Number
Kylian de Vries	1125209
Nathan Snippe	1131636
Rixte de Wolff	1121381
Luke Tobin	1166180
Joey Einerhand	1167318
Koenraad Drost	1123524

Version control

Version	Changes	Date	Comments
1.0	First draft	14-4-2021	-
1.1	Finished certain chapters	15-4-2021	-

Distribution

Person/Organization	Date	Version
Martijn ter Schegget	00-00-2021	1.0

Index

1. Introduction

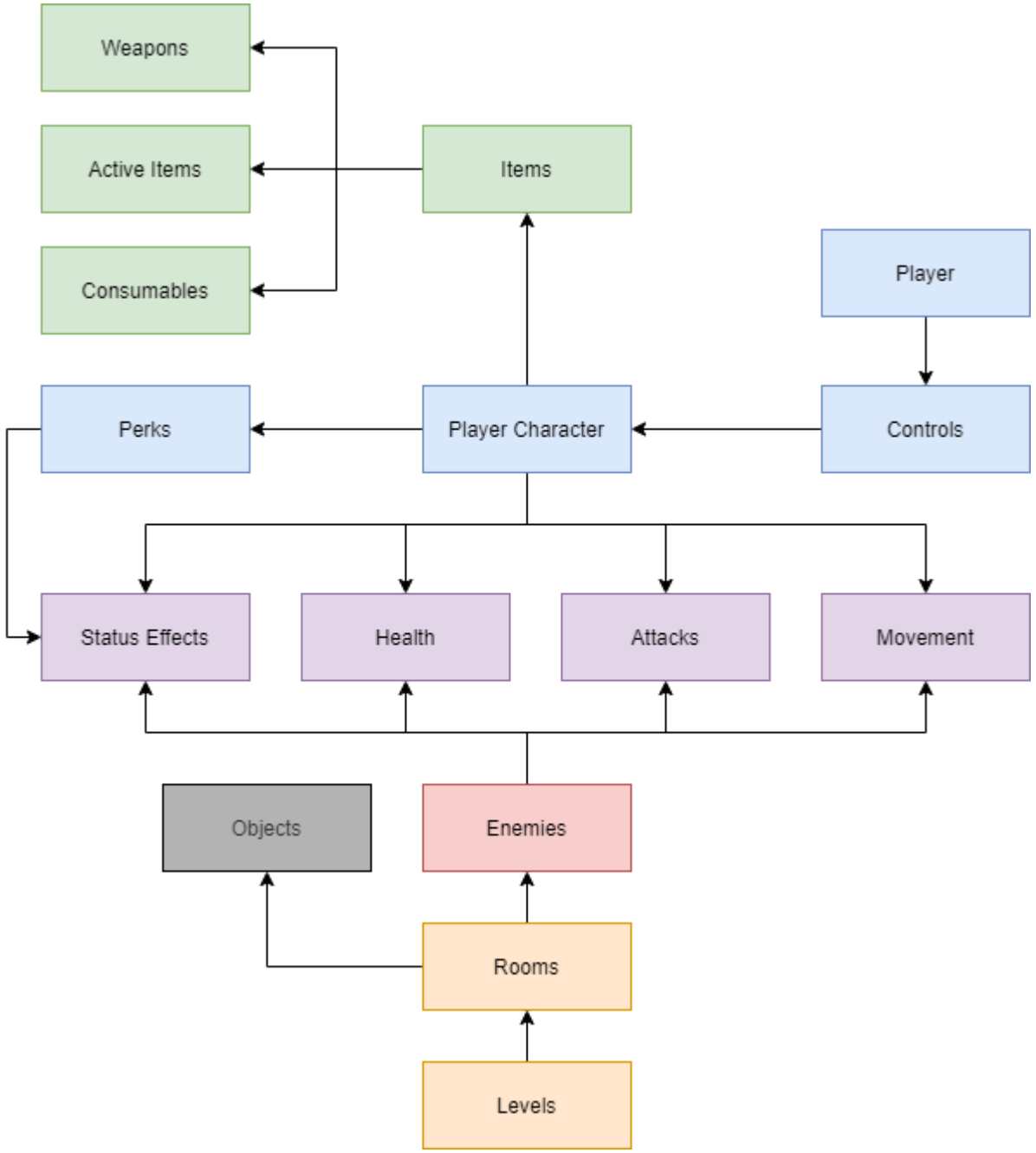
This document describes a baseline for the game design of the project without going into implementation details. This document serves as a reference for any non-technical design which this project is implementing.

The designs documented in the following chapters are based on three factors:

1. The game the project members would like to create,
2. What fits best with the genre of game that gets implemented in this project (2D Top down Roguelike/roguelite)
3. What the target audience of those genres would like to play.

The wants and needs of the target audience will be taken into account whilst designing the game, but will not be used as a major influence in making design decisions.

2. Component model



The model above shows the basic structure of our game. At the core we have the active entities in our game like the player character and the enemy characters. They both implement the movement and health systems, are able to execute attacks and be affected by status effects.
Then we have the levels, which consist of several rooms in which the player character can interact with other entities like the enemies. Rooms can also contain objects that are not affected by the systems most entities are. They might serve as a way to restrict movement, provide cover. Any object that can be interacted with by either the player or other entities will be counted as an entity itself as well. A static metal box would be an object, but an explosive barrel would count as an entity for example.

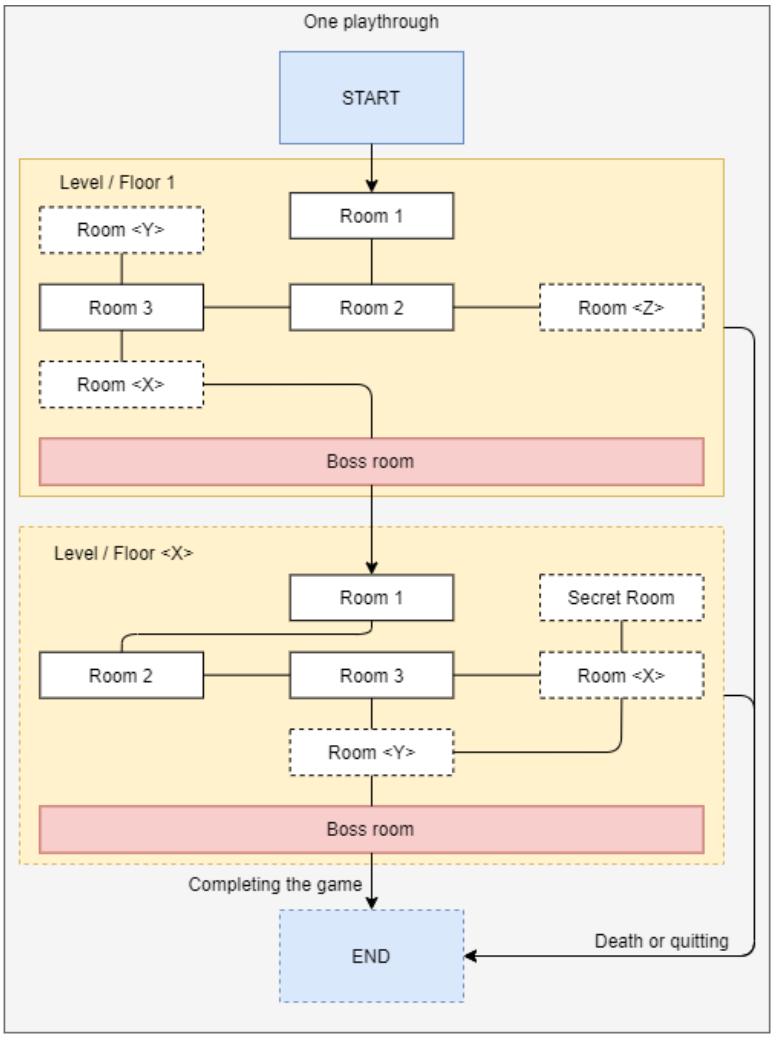
The player character is controlled by the player through the control interface and has access to certain items for use during gameplay and perks that provide certain modifiers to the player characters stats or behaviour.

3. Features

3.1. Environment

3.1.1. Levels

A single playthrough of Samurai Shampoo consists of starting the game and playing through multiple levels until either the player has reached the end of the game, the player has died, or the player has quit the game.



A level is a floor that consists of multiple rooms which the player can explore, including one boss room. While the rooms themselves are manually created, while which rooms are in the level and their interconnectivity is randomly generated (3.1.4).

When the player has defeated the boss of a level, the player can move to the next level, or choose to further explore the level they are on. It is not possible to go back to previous levels once they are left.

3.1.2. Rooms

A level consists of multiple rooms. These rooms will usually contain several enemies or traps. Rooms are premade and then randomly placed in the level, please read more about this in 3.1.4.

These premade rooms can fall under different types. The types of rooms in the initial version will be normal rooms, boss rooms and treasure rooms. Examples of other rooms are secret rooms and challenge rooms. Rooms are designed prior with the use of the unity tilemap system. This system allows for us to hand create a general layout for rooms, while still allowing for flexibility and procedural elements within each.

3.1.3. Objects

Objects are any non-entity placeable Unity GameObjects in a level.

3.1.4. Level generation

Levels are semi-randomly generated. There are manually created rooms that the generation algorithm will pick from. There are limits to certain rooms during generation, for example there is only a single boss room, and there should also be a limited amount of treasure rooms. These room type limits could differ per level, as well as the total number of rooms generated.

3.2. Entities

Entities are all placeable unity GameObjects in a level which have one or more of the following properties:

- It can move
- The player can interact with it
- It has HP
- It has an inventory
- It has AI

Examples of entities could be traps, goop, chests, the player, and weapons pickups.

3.2.1. Movement

With the type of game we're going for, precise controls for the player characters are important. The player should be able to make precise movements to dodge and weave through enemy attacks or handle obstacles. For this the controls need to be snappy and responsive and the player character should quickly respond to changes in direction.

3.2.2. Attacks

There are multiple different types of attacks. These differ in speed, range and behaviour. Attacks can be broadly defined as projectile attacks and swing attacks. Which attack is done, by players or by enemies, depends on the equipped weapon, where every weapon has a unique attack. If an entity has multiple weapons, that means they can use different attacks.

3.2.3. Health

All entities have health, but not all entities can be damaged. Some enemies are (temporarily) invulnerable and do not receive damage. If an entity reaches 0 health, the entity needs to react. For example, if a specific enemy reaches 0 health, it could die.

3.2.4. Inventory system

Each entity has access to their own unique inventory which resets after every run. The player has the ability to store a variety of items such as weapons, consumables and active items. Entities such as enemies typically have a preset or pseudo-random inventory containing items and on unique occasions additional items.

3.2.5. Status Effects

Effects provide entities with a temporary buff or debuff affecting a variety of their stats, items and behaviors. An example of a status effect being applied, is goo on the floor impacting an entity's movement speed while they remain in the area.

3.2.6. Perks

The player can pick a new perk from a (semi-)random selection after every floor boss. These can include simple but significant stat changes, or introduce interesting changes to gameplay. An example would be “Dodging through enemies slightly damages them”. Player perks last for the entire run.

The perk system also works for enemies to potentially be used as an extra difficulty slider, for example for certain bosses or challenge rooms.

3.2.7. Artificial intelligence

Some entities, for example enemies, use algorithms and artificial intelligence (but not machine learning!) to determine their behaviour. For example, an enemy might decide to move towards the player if it only has a melee weapon because otherwise the enemy won’t be able to attack the player.
Another example would be an enemy deciding where to move depending on obstacles, their goal, etc.

3.3. Items

Items can be obtained from many different sources. Each source can have different weightings for different item categories, for example a chest would be more likely to drop a weapon or active item than an enemy.

3.3.1. Weapons

An amalgamation of different weapon types are available within the game. Weapon types can vary from swords to hammers. A typical weapon has a set range. The primary set of weapons possibly available to the player are melee based. Weapons available to the enemies will in most scenarios be ranged, exceptions can occur in situations such as bosses or unique enemy encounters.

Player weapons only show when attacking,otherwise they stay in the Player’s ‘pocket’. Ranged enemies will always show their weapon to show their aiming direction. Melee enemies will either have their weapon animation tied to their idle and moving animation, or only show the weapon during the windup animation.

3.3.2. Consumables

The player has a single consumable slot. If a new consumable is picked up while the player already has one in their inventory, the already held item is dropped on the ground. The

consumables that fill this slot can be activated with a hotkey. A consumable has one or more charges, and when the charges are used up the item will disappear.

3.3.3. Instant Collectables

Instant Collectables are used immediately when picked up. This section of items includes health restoring drops and temporary buffs. These collectables are not stored within the entity's inventory but rather used instanously.

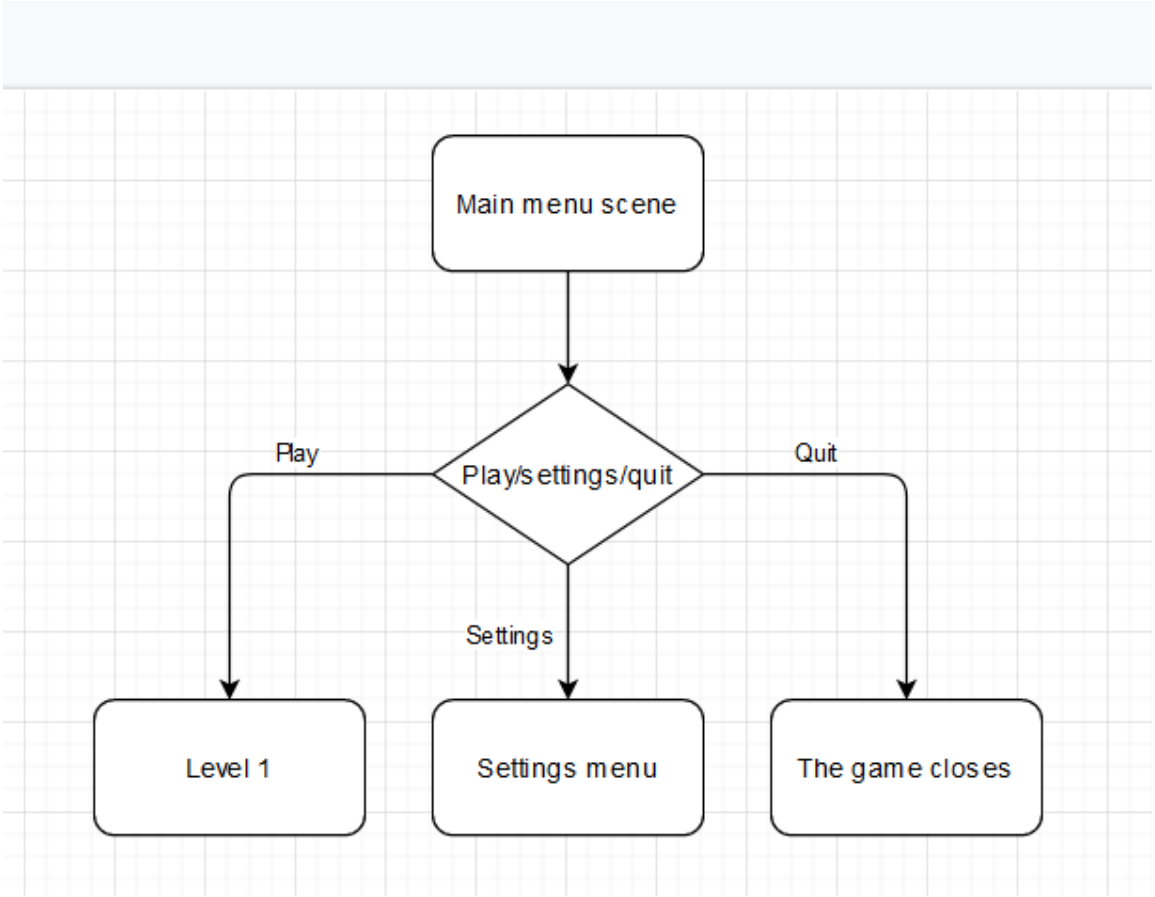
3.3.4. Active items

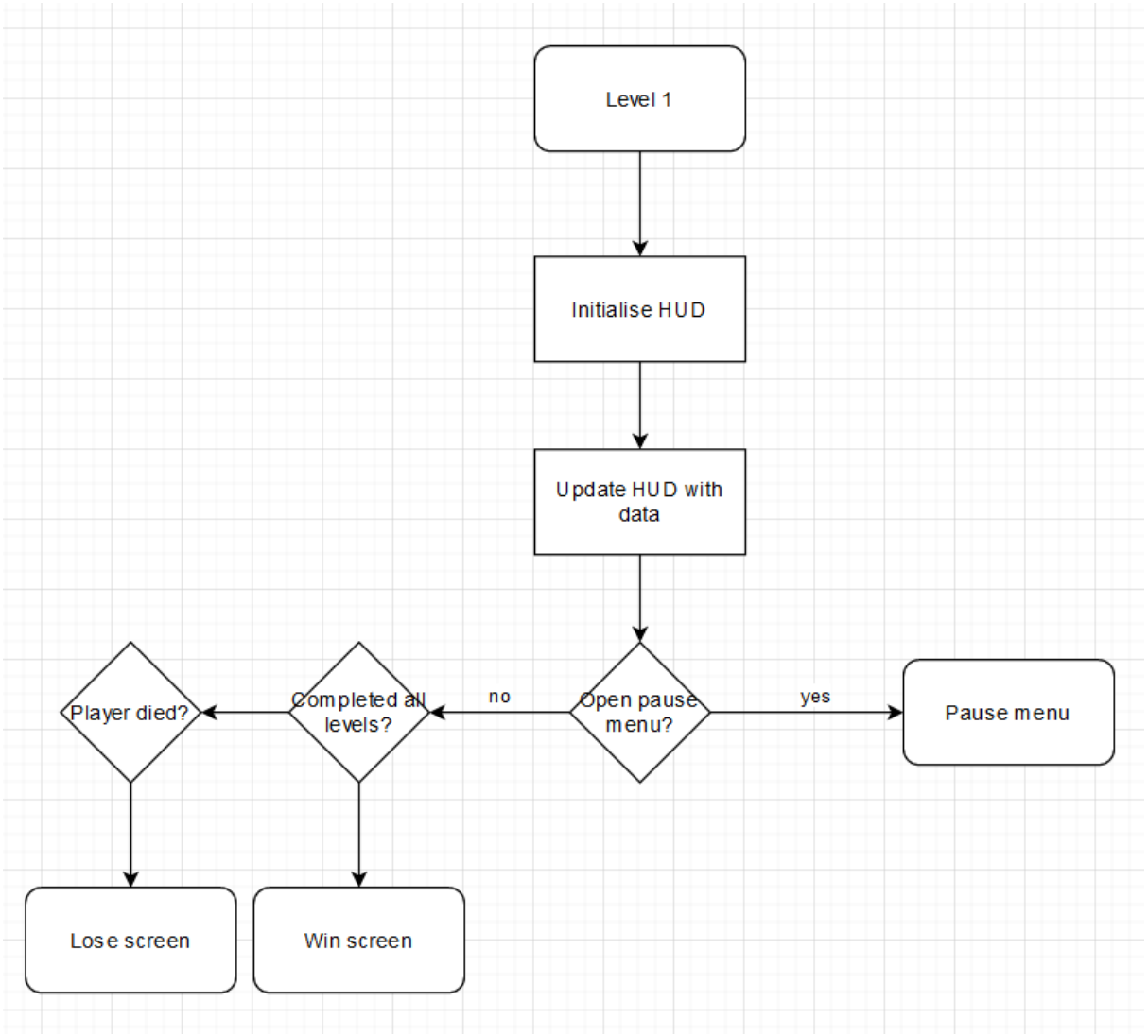
The player has a single active item slot. Each active item has different properties, effects and rules. With the use of a hotkey (when applicable) the ability will be activated and set on their respective cooldown if needed. The cooldown and rechargeability of items will be unique to each. Cooldown or recharging can be for example affected through passage of time, enemies killed or rooms cleared.

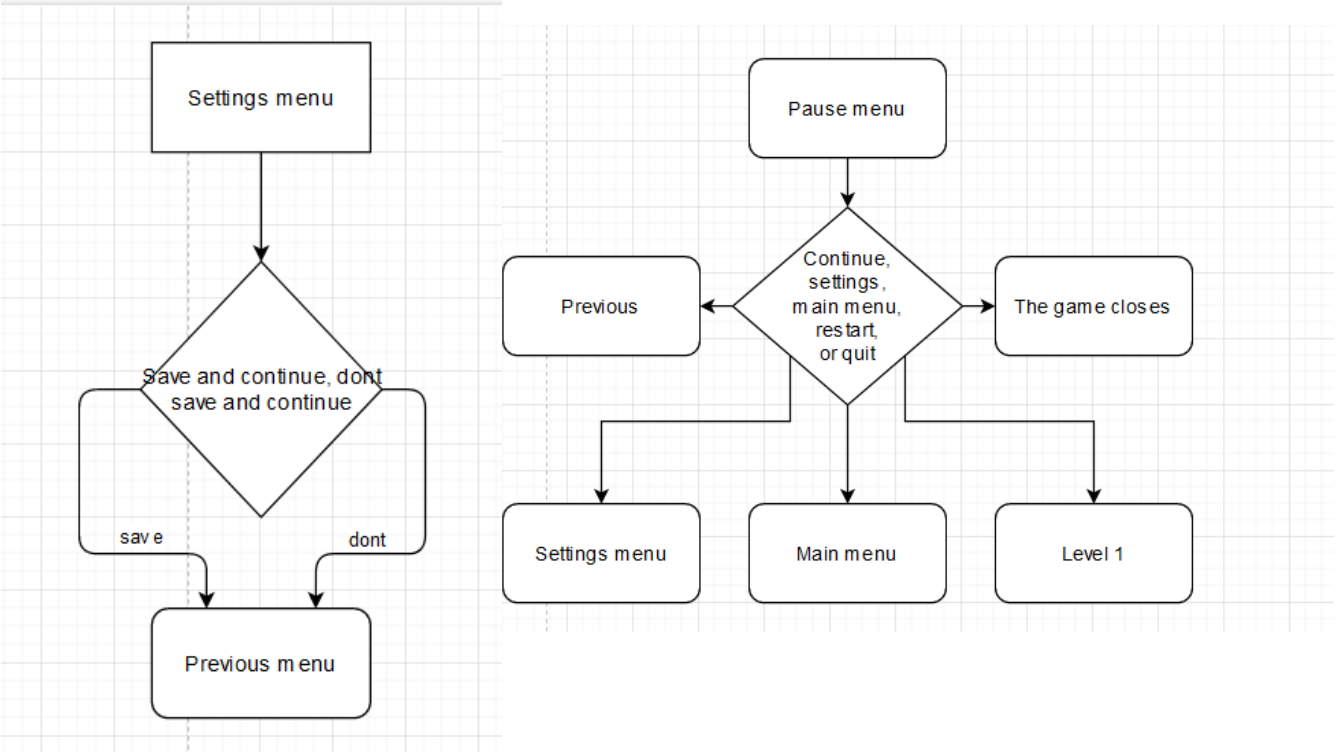
3.4. UI

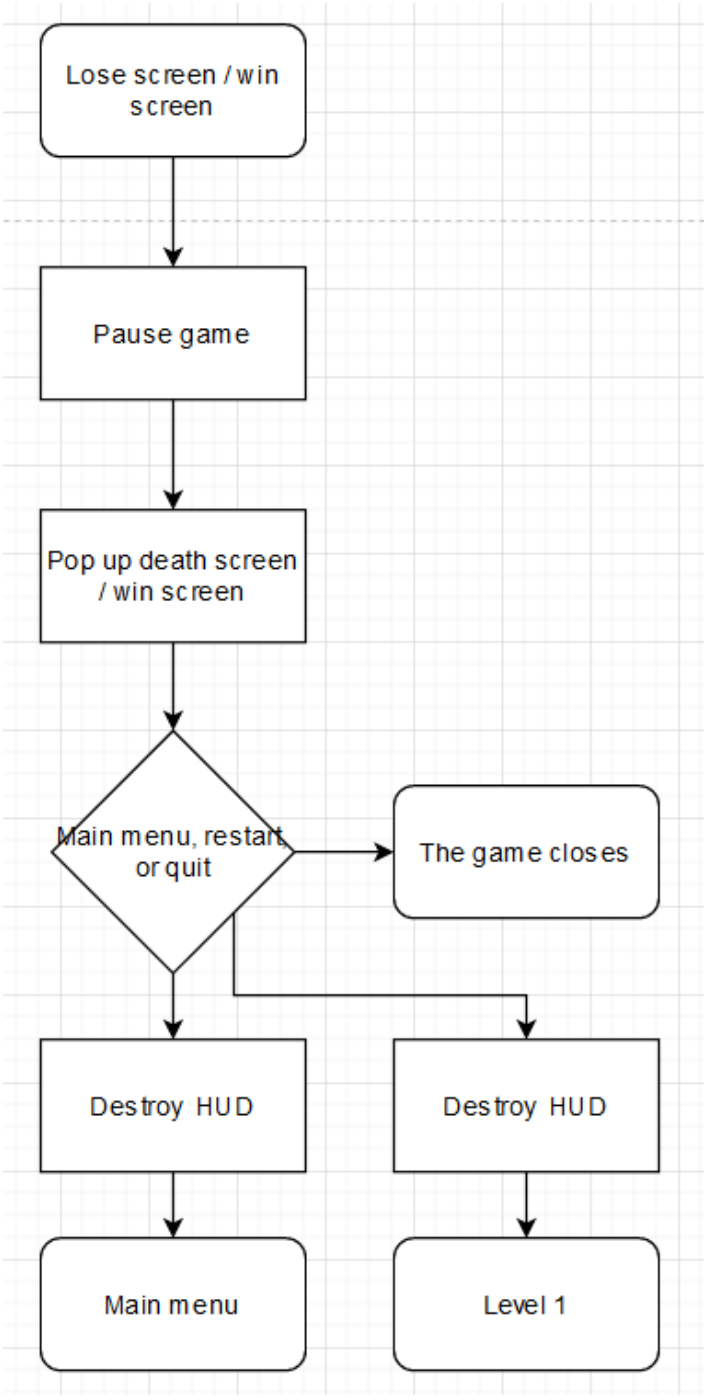
The UI is an important part of the user experience. If the UI in any way hinders the user, the user will be more likely to get frustrated, think negatively of the game, and quit. For this reason the general flow of the UI has been modelled, in which the hindering of the player has been minimalised.

UI Flow




Level 1
Initialise HUD
Update HUD with data
Open pause menu?
no
yes
Pause menu
Completed all levels?
Player died?
Lose screen
Win screen


Settings menu
Save and continue, dont save and continue
save
dont
Previous menu
Pause menu
Continue, settings, main menu, restart, or quit
Previous
The game closes
Settings menu
Main menu
Level 1

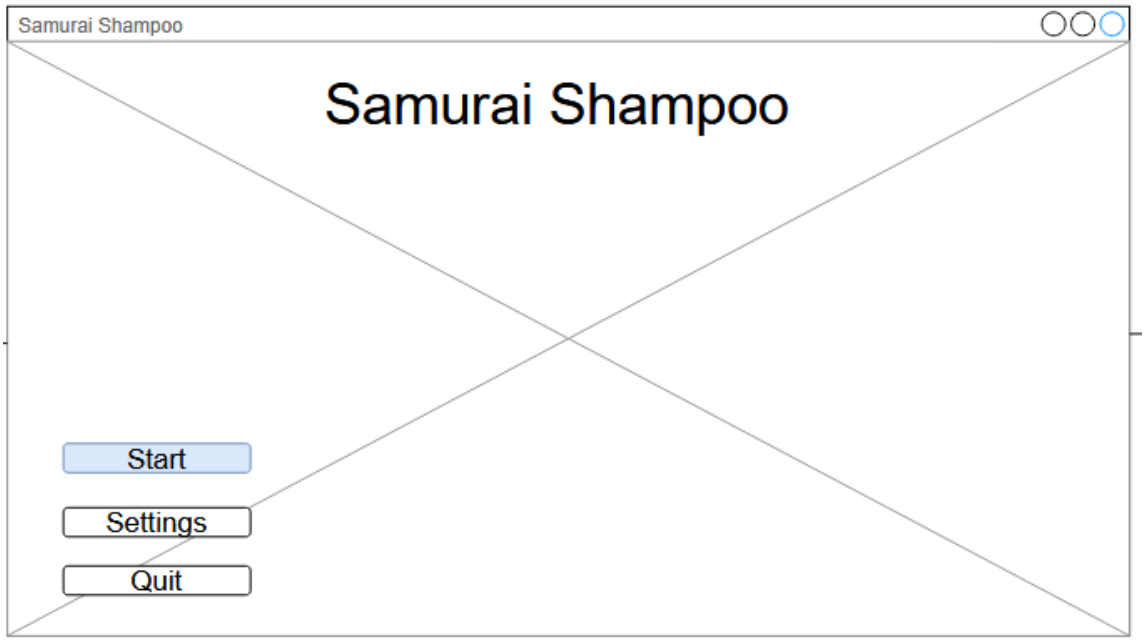

Lose screen / win screen
Pause game
Pop up death screen / win screen
Main menu, restart, or quit
The game closes
Destroy HUD
Destroy HUD
Main menu
Level 1

UI Design

Because Samurai Shampoo is a quick, reflex-based game, the UI should be clear of clutter and shouldn't take up too much screen space.

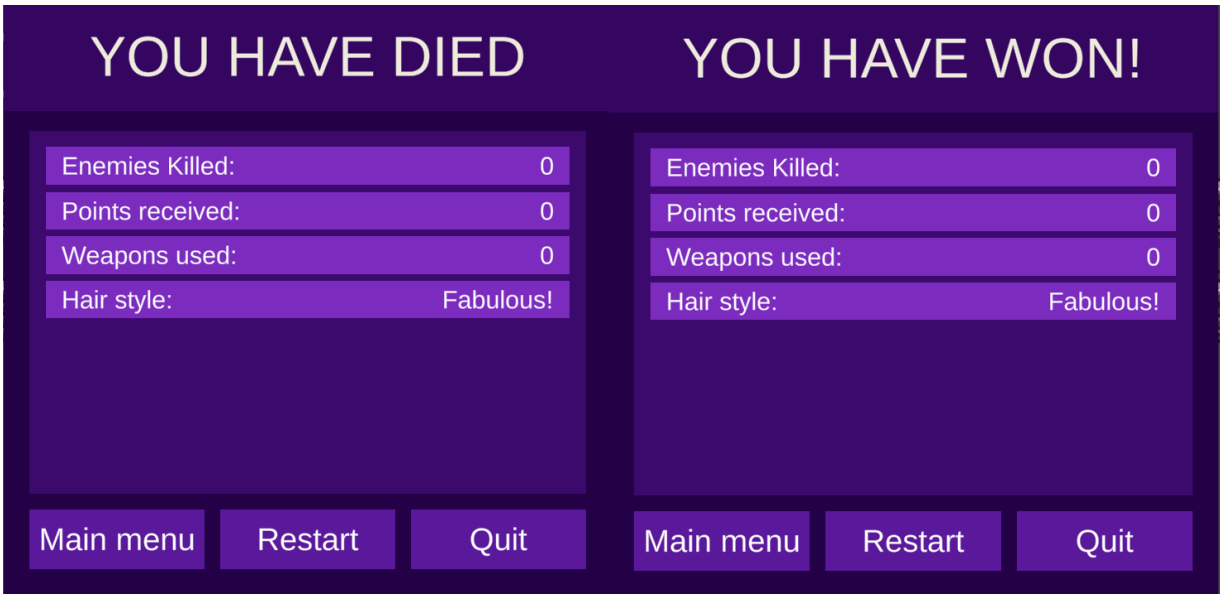
3.4.1. Menu Design

For the main menu and settings menu, we took inspiration from games like Portal and Battlefield. For a comparison and inspiration sheet, please see "MainMenuSettingsMenuWireframe.png" to see which menus we used as inspiration.
The following wireframes are the designs which we thought are the most pleasing to the eye, and are implemented in Samurai Shampoo



When the player dies or wins, they need to be shown a screen which shows their playthrough statistics, and options to go back to the main menu, restart the game, or quit.

Since this is a roguelike game in which the focus lies on the player's ability to play the game, the player might get frustrated if they lose. To minimise this frustration, the player must, at any given time, be max 2 clicks away from restarting the game. This reduces time between dying and restarting the game, reducing frustration.

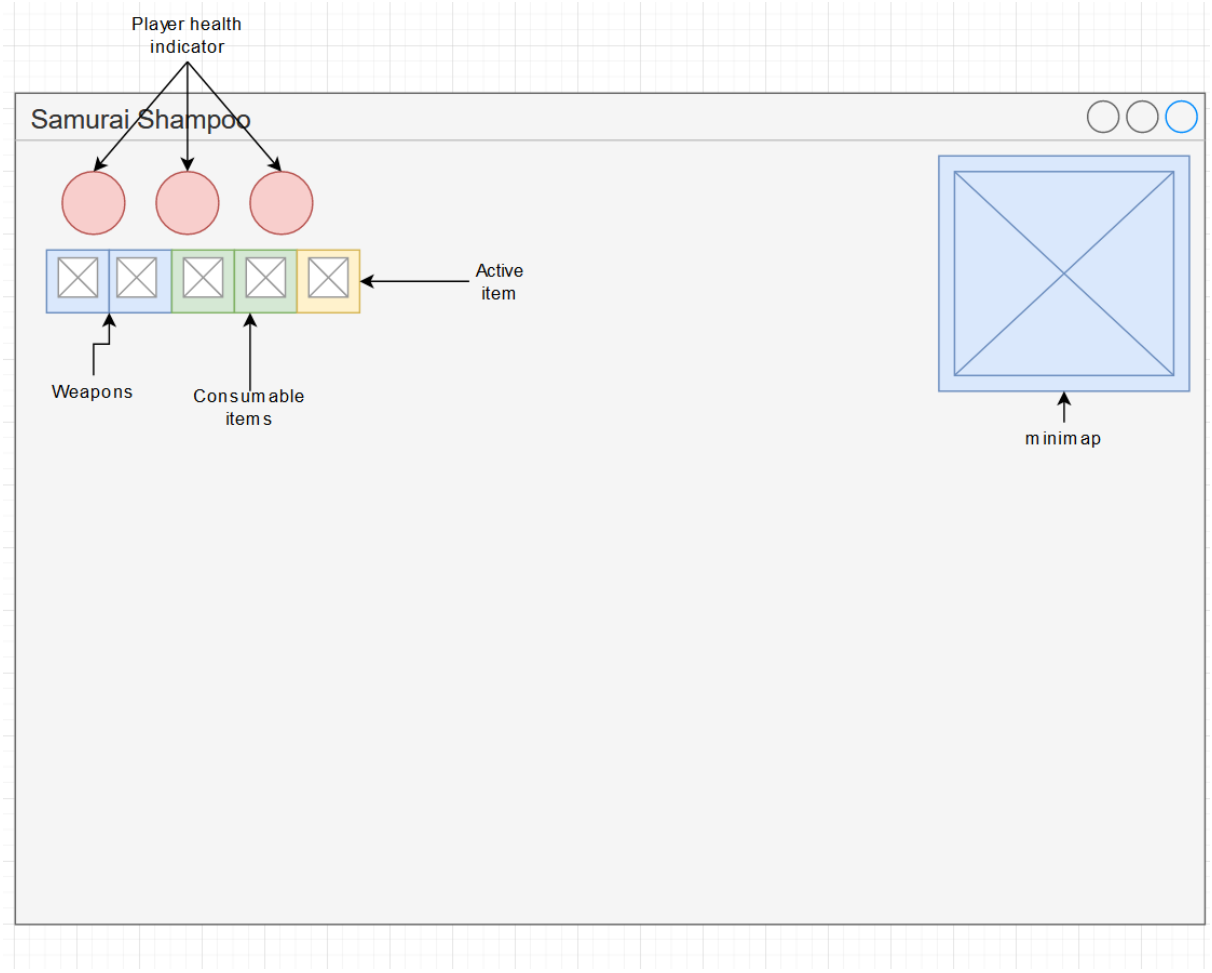


When the player presses the pause button, they must have access to a pause menu screen, in which they can unpause the game, go to the settings menu, restart, or quit the game.

If the player accidentally clicks the restart or quit button and loses their progress, they get frustrated. To counter accidentally losing progress, a confirmation popup is added when the player clicks on a button which would result in losing progress. In that popup, the player needs to click if they actually want to restart or quit, and therefore lose their progress.

3.4.2. UI Design

The HUD will contain a health indicator, two weapon slots, at least two consumable slots, status effect icons, an active item slot and a minimap that automatically hides itself during combat.



When the level is instantiated, the player object call the health an inventory UI elements and fills it with the items and health the player has. The amount of health indicators increase or decrease dynamically based on the amount of maxhealth and health the player has.
Every time the player gets damaged or gets healed, it calls the health UI and changes the amount of health indicators it should show.
Every time the player swaps weapon, or picks up a weapon or item, the player inventory feature calls the UI and asks it to update it to represent the current player inventory.

__MINIMAP_

4. Aesthetics

4.1. Style Guide

!TODO! Retro, pixel, shading, hue shifting, etc

4.2. Art Work

The sprite's we will be using for the game are based on an itch.io package.

4.3. Audio

The audio we will be using for the game is based on a unity package.