Interim Report

Soulbound Escape

Lukas Liu

Mariia Iurtaeva

Arda Karaman

Overview

In this milestone, we show our progress with respect to our target criteria set for this milestone. All our implementations were done on the latest stable Unity version. Our main goal was to create a first playable demo and implement some core mechanics from our idea. Since we consider mechanics more important at this stage, we spent more time on it, rather than working on complex level design. But we still implemented some crucial mechanics that will influence our levels (such as isometric camera view).

As we are a small team of 3 people, managing our team and working with tasks is simple and we are able to work on our own tasks and help each other with implementation problems.

Progress

Currently, we have implemented all the goals from our functional minimum and some from our low target. Thus, we have most of the core mechanics, which will serve as fundamentals for future development. We also added animations for the implemented models, so we won't need to start that from scratch for the desired target.

The only important big mechanics left to implement is map generation. But since it is not necessary to see other basic functionality and is more needed for player's enjoyment and replayability, it wasn't our priority in this milestone.

Overall, we are up to schedule and the progress feels stable for us, so we are going to continue working further in the next milestone in the same manner. Below you can see our progress in the to-do list.

Functional Minimum

- ☑ ~~Simple Map~~
- ☑ ~~Isometric camera view~~
- ☑ ~~Player controls~~
- ☑ ~~Basic combat system~~
- ☑ ~~Basic enemies~~

Low Target

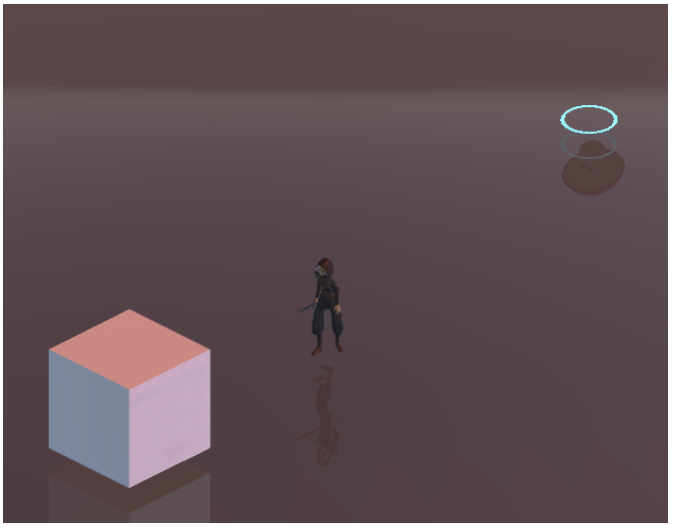
- ☐ World flip mechanic
- ☐ Simple map generation

Desired Target

- [ ] Procedural map generation
- [ ] Interactables
- [ ] Animations
- [ ] SFX

Isometric camera

We tweaked the camera settings so it follows the player character's movement on a 30° angle around the x-axis and rotated 45° along the y-axis to give it the isometric camera view look. We also included a second camera which follows the secondary player character located on the underworld side of the map which is going to play a vital role in the world flip mechanic. It is exactly the opposite of the overworld camera following the player in the same manner.



In tandem with the world flip mechanic, it posed some difficulties getting the player controller right as swapping to the secondary camera flipped the player's perspective in terms of directions which needed to be addressed.

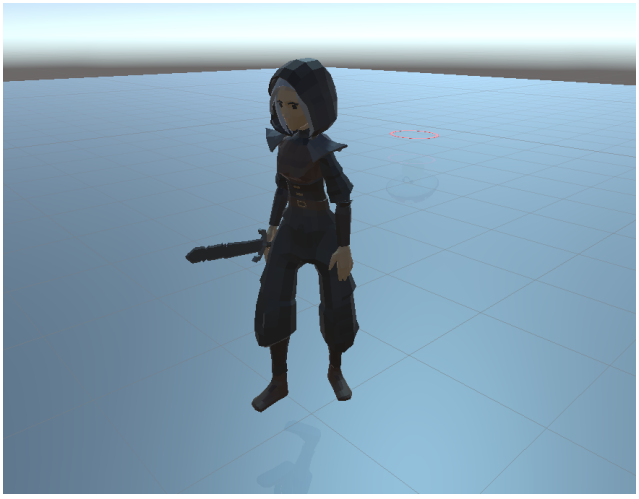
Player controller

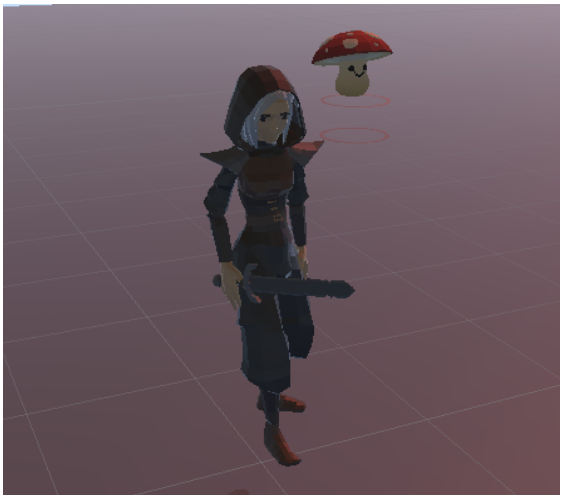
As we opt to make an isometric action-based hack-and-slay game, the player controls needed to feel responsive, fluid, and snappy. We use simple WASD keyboard input for the player movement and the left mouse click for a simple attack. As we cannot translate WASD input to immediate directional input as the player would only move diagonally through the camera because of the camera rotation, the input is rotated 45° to fix this issue.

The player moves both character models simultaneously and is only able to perform actions like attacking or combat rolling on the character they are currently controlling. For the movement, we use a parent object to move both characters around at the same time. This object also keeps track of which character we are currently controlling to accordingly execute the previously mentioned actions and animations.

Pressing the ‘F’ key will flip the current camera to the camera on the other side and switch the camera view to the other player's character. Due to perspective changes, the input needed to be adjusted as well which we elaborate on further in the next chapter.
To make the gameplay feel more dynamic, we also added a combat roll to give the player more freedom in strategy and movement by adding invulnerable frames for dodging.

We also opted to focus on animations for the player character a lot more to make the controls feel more responsive and fun to press as that is the most important aspect of the game due to its focus on gameplay. Running, slashing, and rolling all are animated including animation events and morphing to fluidly transition between animation states.





World flip mechanic

As this is our main mechanic tied to the theme, this needed to feel fluid and intuitive as well. Originally, we thought about flipping the main camera itself to make the player feel like they are actually flipping the world perspective but that proved to be a difficult approach due to the player still being able to move around while the flip is happening. As mentioned before, we instead added a secondary character that is constantly following the underworld player character, and both cameras are activated/deactivated according to whichever character the player is currently controlling.

However, since the camera on the underworld side has a different perspective, the controls need to be adjusted. For example, moving forward in the game means moving away from the camera. Since we only flip the perspective and do not change the axes, the movement vectors need to be negated in order for the forward vector to still “move away from the camera” regardless of which world we are in.

It is currently missing an animation for the UI so it feels more like an actual flip for the player rather than just changing between two cameras.

Player and Enemies

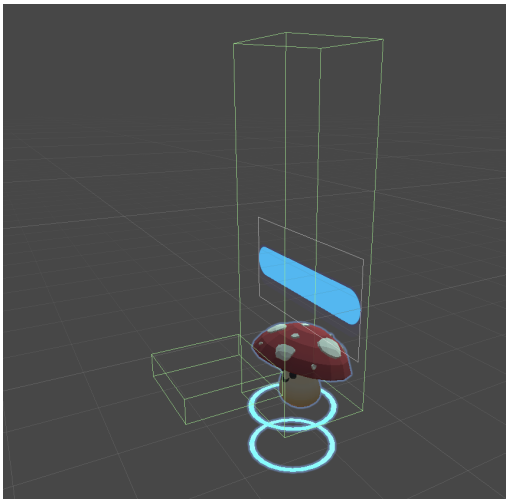
Currently, there is one enemy type in our demo. The “mushroom enemy” exists on one of the sides of the world(in the future it might be only assigned to one world or have different characteristics depending on the world). It has some simple animations for standing, moving, and attacking. This enemy constantly follows the player on the side, where it is spawned. If the enemy catches the player up and they are located in the attack range, the mushroom attacks the player. The next attack only happens after a cooldown.

Even though the enemy only exists in one world, the player exists in both worlds simultaneously, so they are getting followed on both sides. If the player currently controls the opposite side from a mushroom, its position is shown through a circle: blue for the overworld or red for the underworld.

Enemies also have a health bar, which appears over them, and several adjustable attributes like speed, cooldown time, attack range, maximum health, and damage, making the code adjustable for new types of enemies in the future.

Combat System

The simple combat system currently relies on colliders and triggers. For the enemy attacking system, the enemy checks if the player is inside a box collider area and hits the character if the player is inside. The player's combat system is even more simple with just one mesh collider only activates when the player leftclicks and then the mesh collider checks for the trigger entries, damaging every enemy it collides with.

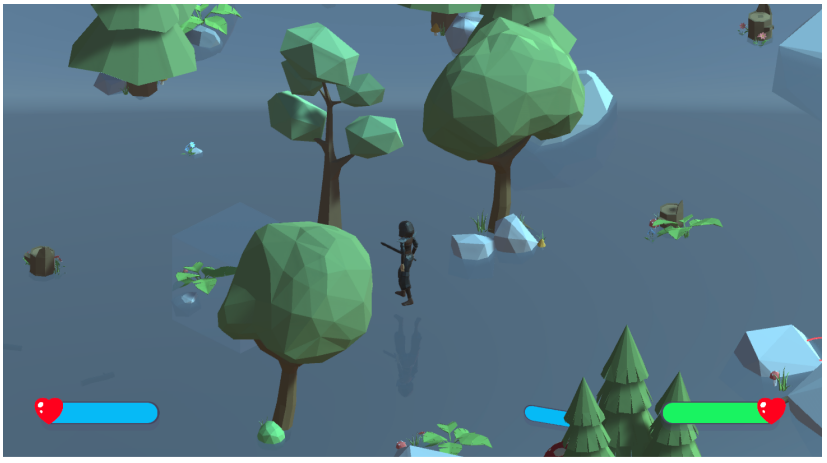


Models, Environment, and Level Design

As we mentioned before, our goal is to make a low-poly style game. Thus, we collected 3D models from several free-to-use sources. Since those sources often offer animations for the models, we also adjusted them for our needs in the game. We might also adjust some colours and/or details in the future.

At the moment, we have one main scene that shows the core implemented mechanics. The scene still doesn't have a complex environment and complex level design, which was mentioned earlier. It still aligns with our plans for this milestone, as one of the main goals was implementing the double world mechanics with an isometric camera view and making this part of the level design work. Currently the floor is semi-transparent mainly for testing purposes and will later be opaque in favour of visual clarity.

Even though the environment is not complex, we tried to work with the low-poly style and our worlds' colouring schemes.



Evaluation and Takeaways

Our main goals for this milestone were reached, so we will continue working according to our schedule. We have implemented solid core mechanics, so we can add more variety to the game. Adding new enemies, obstacles and smaller mechanics are in our future plans.

We will also work more on the level design, which will be crucial for the map generation later. In this area, we will definitely be adding interactable objects to the map, assets, and other mechanics. It would be also necessary to playtest the levels in the later development stages, so we can adjust some design decisions and understand how they work best with our camera view and double-world mechanics.

The next big step would be adding map generation and some smaller things that make the game more joyful and alive: UI, SFX, and so on. Those will make the game feel like a real replayable game and not just a demo for implementing core mechanics.

One problem that was definitely time-consuming and annoying for our development process was the bad compatibility of Unity with git. Since Unity has some unsolved problems for development with git, we stumbled upon merge conflicts and lost some time trying to solve them. In the end, we just learned once again to not work on the same scene at the same time and to use extra testing scenes.