Labyrinth of the Goblin King

Project Report

Labyrinth of the Goblin King is the First-Person Shooter I developed for my Game Design CA1. It is a dark fantasy dungeon-crawling game, where the player must fight their way through the Goblin King's labyrinth, killing goblins and grabbing treasure along the way. To escape from the maze, the player must leap into the glowing portal at the other side!

Concept

As a huge fan of tabletop games, roleplaying videogames, and general science fiction, I wanted to create a game in the conventional and popular dark medieval fantasy style. Drawing inspiration from games such as Skyrim and Dark Souls for my aesthetic, I wanted to recreate the feelings invoked by more fast-paced games such as DOOM. From the start, my concept had simple, clear objectives to win. To conquer the maze and earn score!

Narrative

You are a goblin vying for the Crown of the Goblin King. To prove yourself, you must escape the labyrinth, killing all who stand in your way. The Goblin King's treasure has been hidden in the tunnels for you to find and lay claim to… Rise up beyond his challenge and claim the throne for yourself!

Goals

- Maze: As a traditional directional puzzle, mazes show up consistently in games and stories as a fun, engaging obstacle. A glowing portal awaits you at the other end of the maze as your only hope of escape!
- Score: Score is tracked by defeating the goblins of the maze and collecting treasure chests! A classic feature across countless games, it is a simple way to identify how well a player did in comparison to other players.

FIGURE 1: THE GOBLIN OF LOTGK

Obstacles

- Time: Solving a maze puzzle is simple enough, given the time to do so. Adding time pressure offers an element of difficulty that makes the game more of a challenge to play.
- Goblins: The goblins of the maze are fearsome and vicious – if they get close enough to you, they will damage the player. Keep your health up if you can… When it runs out, it's game over!

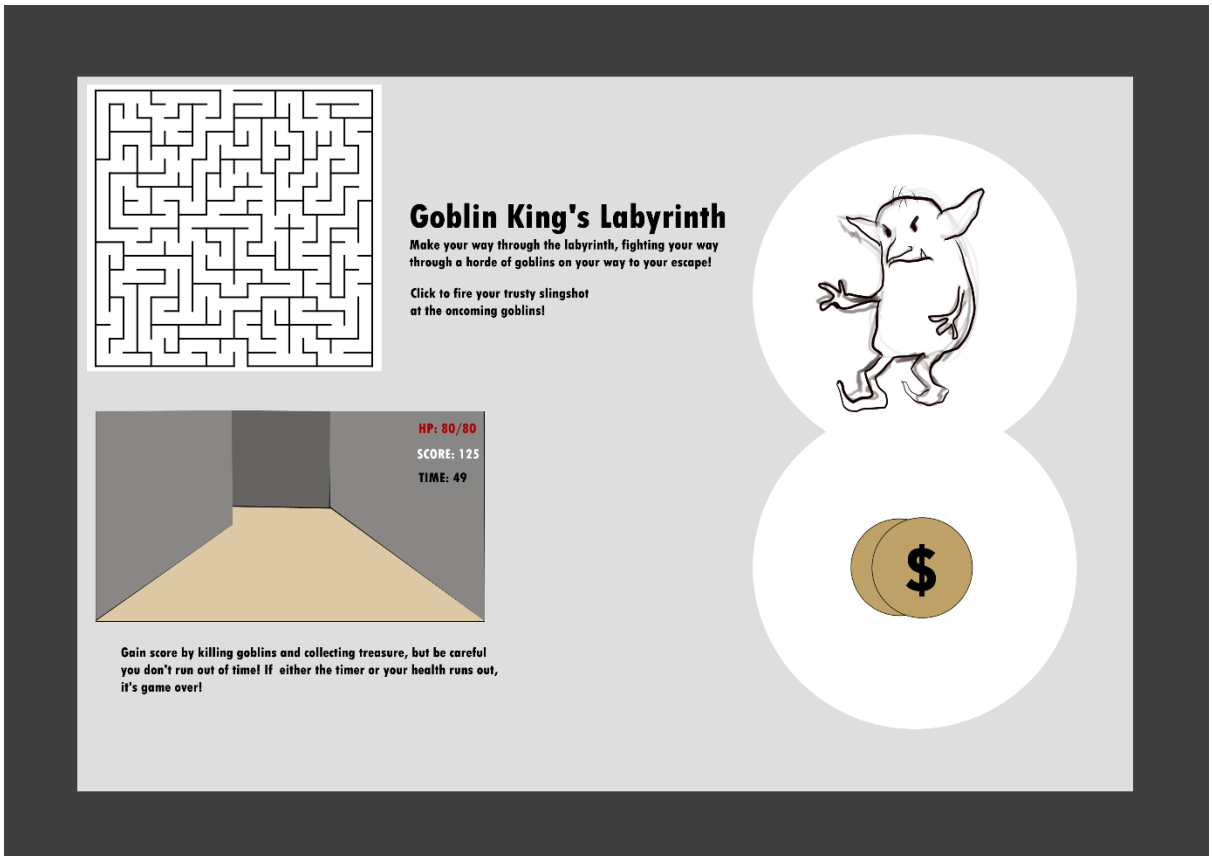


FIGURE 2: ORIGINAL GAME CONCEPT

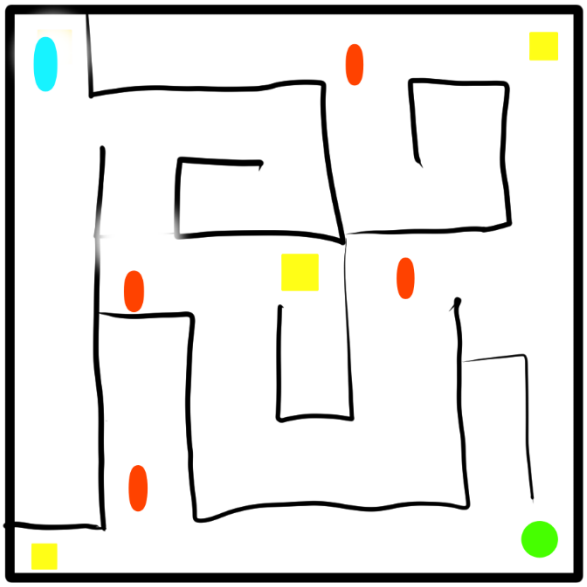
Concept Art

I wanted to give the game an ominous, subterranean look and feel, which is why I chose earthy stone colours and textures, as well as choosing goblins as the "enemy" obstacle of the game. The labyrinth needed to feel like the player was trapped in there with the goblins, so I used textures to mimic this. Pictured above is a simple conceptual graphic where I laid out my ideas for the game.

In addition to the visuals, I wanted to give the game an old-school soundscape, so I created my own sound files and music, except for the hit and death sounds for the enemies. This gave me great control over the overall audio engineering, giving the labyrinth the atmosphere I was going for.

Map Design

Originally, the map would have been built by hand-placed assets, but I elected to use random generation to offer the player more replayability within the game. Due to this, I was slightly limited when it came to placing permanent assets. The player always starts in the bottom right corner, and their goal is to make it to the top left. There are several treasure pickups dotted throughout the labyrinth, and an array of glowing crystals light the way through.



Level Design

Due to the nature of the game, despite only having a single level, it has a new layout every time. While this could make raising difficulty harder, there are plenty of other ways to tweak it; from raising the number of enemies to increasing their damage. The maze algorithm also accepts a height and width value, so it can be set to any size from a 2x2 grid upwards. The static assets (Player, End portal, lights & floor) are configured for a 10x10 cell grid.

FIGURE 3: LEVEL DESIGN

Pictured above is a top-down sketch of a sample level generated by the maze scripts. Throughout the labyrinth is a series of obstacles and collectibles:

- Blue: The player, who spawns in one corner of the map.
- Green: The exit, which appears in the opposite corner.
- Red: Enemies, which meander through the maze, chasing and damaging the player. They can be killed for more score points.
- Yellow: Treasure. These are pickups the player can find to add to their score.

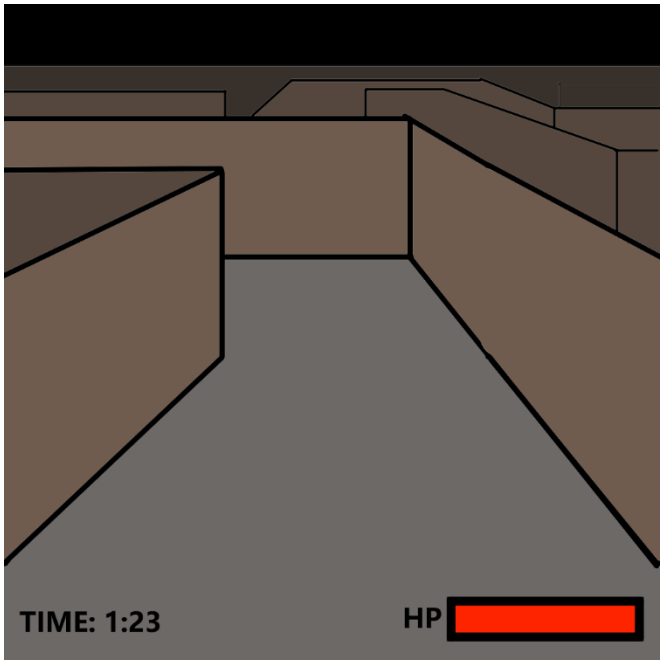


FIGURE 4: PLAYER PERSPECTIVE

Shown in Figure 4 is a sketch of how the player would see the game. Solving a maze from above is one matter but finding your way through in first person is another. From within the maze, it feels less like a traditional puzzle and more a winding mesh of tunnels. Due to the nature of the maze, it is easy to be surprised by a goblin rounding the corner, or sometimes even ambushed by a few them, depending on how they happened to spawn.

Code Process

The game began as a simple platform with several capsules and cubes dotted around the area. While most of these objects acted as obstacles, one of the capsules was scripted to act as a player. This was achieved by making the camera object a child of the player object, following as it moved. The camera was restricted to simulate a person's gaze, clamping the player's look range similarly to how the human neck is limited.

```
//clamping the look values, do not look up or down too far.
rotation.y = Mathf.Clamp(rotation.y, minClamp, maxClamp);
//rotating the character based on the Mouse x-pos
player.transform.RotateAround(transform.position, Vector3.up, Input.GetAxis("Mouse X")* lookSensitivity);
```

The next step was to add enemies; by applying a Nav Mesh Agent to the enemy and "baking" a Nav Mesh onto the plane used as a floor, I could tell the "enemy" objects to track and follow the player. With that completed, I added a "weapon" empty to fire "bullet" prefabs.

Using Unity's in-built collisions system, I gave the enemies HP, which would be removed when they collided with the bullets. I also gave the player HP, which would be removed when the enemies collided with the player. Finally, if the enemies ran out of HP, they would be removed from the game. If the player ran out, the game would end.

With that, the basic prototype was complete! I began implementing graphics, adding textures to the ground and obstacles and swapping out my enemy capsule for a goblin mesh. The next step was adding animations; In the end, I had added several simple animations such as fade-ins and fade-outs when changing scenes, as well as more complicated ones such as the spawn and movement animations for the goblins; which required scripted triggers.

After giving the game some of the aesthetics I wanted to implement in the final application, I added more in-depth features such as a Heads-Up display (HUD) and an enemy spawner. The HUD was a simple canvas element, upon which I

attached several text elements and images to. Various scripts in the project would send information to these elements such as remaining time and score, which would then be displayed to the screen. The player's health bar works similarly but converts the health "number" into a flat colour image, which is scaled to fill a "health bar" on the HUD (Brackeys, 2020).

The most important piece of my game was replacing the preset obstacles with a maze to navigate through. While I could have simply made a basic maze, I thought it would be more fun if the game featured randomly generating mazes. I managed to implement this with the addition of a recursive algorithm (Nambiar, 2020). Using this, the script sets a grid of walled cells, then draws a random path through them. After this first path, it backtracks through any unvisited cells, ensuring that every cell in the maze is theoretically reachable by the player. Using this algorithm, I then had the script apply a series of "wall" prefabs to build the maze in the scene.

```
var list = new List<NextCell>();

if (p.X > 0) {
    if (!maze[p.X - 1, p.Y].HasFlag(MazeWall.VISITED)) //If the left cell is unvisited, make it a neighbour of this cell
    {
        list.Add(new NextCell {
            Position = new Position
            {
                X = p.X - 1,
                Y = p.Y
            },
            CommonWall = MazeWall.LEFT
        });
    }
}
```

Above, the code checks if there is an unvisited cell to the left of the current cell. This code repeats for each direction and ensures that no cell is left unvisited, before flagging which walls will need to be filled in by the Maze Rendering script.

```
for (int j = 0; j < height; ++j)
{
    var cell = maze[i, j];
    var position = new Vector3(-width / 2 + i, 0, -height / 2 + j);

    if (cell.HasFlag(MazeWall.UP))  //If the "up" wall is flagged, instantiate the wall to reflect this.
    {
        var topWall = Instantiate(wallPrefab, transform) as Transform;
        topWall.position = position + new Vector3(0, 0, size / 2);
        topWall.localScale = new Vector3(size, topWall.localScale.y, topWall.localScale.z);
    }
```

This is a sample section of the Maze Renderer script; shown above the script checks if the cell is flagged for an upper wall.

Once the maze was fully assembled, I added the “exit portals” and “treasure” pickups. Both of these use a variant of the standard colliders, using the Trigger functionality instead of the standard collisions. This means that instead of causing a collision, the object detects when another enters its radius. The treasure object adds to the player’s score before deleting itself, and the exit portal (A shimmering green particle-effect orb, (Aguiar, 2017)) loads the victory scene.

Once finishing the important elements of the game I made some simple Menu, Win, and Game over scenes with some simple animations, then moved on to tweaks and minor fixes. I added sound effects, a typeface (Ink, 2019) and background music, and changed my skybox from a solid black to a glowing subterranean surface, to further reinforce the aesthetic of the game.

Most sound clips including the music and most effects were created by myself on the Beepbox site (Nesky, n.d.), but I had sourced a handful of them: the goblin hit (SFX, n.d.) and death (Koenig, 2009) sounds, as well as the portal’s warbling hum. (Ch0cci, 2006)

Conclusion

Overall, Labyrinth of the Goblin King is a simple game with clear objectives and obstacles. Its key features are its PlayStation One style and its random maze generation. Featuring a modular design, easily incremented difficulty and an immersive audio environment, I was pleased with the game overall. Future goals for Labyrinth of the Goblin King include advanced code for the Navigation Mesh for smarter enemy AI, as well as multiple levels that increment in difficulty as the player progresses, both in size, enemy count, and enemy strength.

Imported assets used in my game

In the creation of my game, I used a variety of assets from the Unity Asset Store to decorate the world and bring life to the level. From the player’s crossbow, to the lights, to the goblins, the asset store proved to be invaluable in the creation of my game.

Crossbow Asset (Noman)

Ground tiles (Indie)

Goblin (Kozhemyakin)

Wall texture (Studios)

Crystal Lights (SineVFX)

Treasure pickup (shop)

Skybox (Yughues)

References

Aguiar, G. (2017, May 20). *Unity 5 - Game Effects VFX - Glowing Orb*. Retrieved from Youtube: https://www.youtube.com/watch?v=ctmqr_8esT0&t=265s&ab_channel=GabrielAguiarProd.

Brackeys. (2020, February 9). *How to make a HEALTH BAR in Unity!* Retrieved from Youtube: https://www.youtube.com/watch?v=BLfNP4Sc_iA&ab_channel=Brackeys

Ch0cci. (2006, February 1). *Alien Ship Idle*. Retrieved from freesound.org: https://freesound.org/people/Ch0cchi/sounds/15347/

Indie, P. (n.d.). *Dungeon Ground Texture.* Unity Asset Store.

Ink, C. (2019, October 11). *Wayfarer's Toy Box Font*. Retrieved from fontspace.com: https://www.fontspace.com/wayfarers-toy-box-font-f40851

Koenig, M. (2009, August 13). *Gagging Sound*. Retrieved from Sound Bible: http://soundbible.com/844-Gagging.html

Kozhemyakin, A. (n.d.). *Goblin.* Unity Asset Store.

Nambiar, S. (2020, May 22). *Maze Generation Unity Tutorial*. Retrieved from Youtube: https://www.youtube.com/watch?v=ya1HyptE5uc&ab_channel=SandeepNambiar

Nesky, J. (n.d.). *Beepbox*. Retrieved from Beepbox: beepbox.co

Noman. (n.d.). *Crossbow [Medieval Weapons Pack].* Unity Asset Store.

SFX, F. (n.d.). *https://freesfx.co.uk/Category/Rubber/382*. Retrieved from Hard Rubber Impact.

shop, F. (n.d.). *Treasure Set - Free Chest.* Unity Asset Store.

SineVFX. (n.d.). *Translucent Crystals.* Unity Asset Store.

Studios, G.-R. (n.d.). *Tileable Bricks Wall.* Unity Asset Store.

Yughues, N. /. (n.d.). *Yughues Free Ground Materials.* Unity Asset Store.