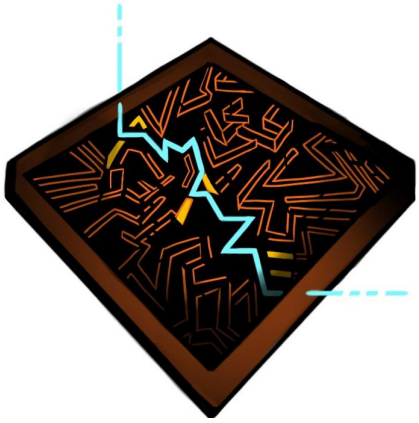
iAmaze



iAmUnknown *

March 2020

*Team Members:
Sergio Moubayed
Thuraya Shanbari
Carl Van Hoorebeke

Contents

1 Introduction **3**

2 The Project **4**

2.1 Thuraya Shanbari 4

2.2 Sergio Moubayed 12

2.3 Carl Van Hoorebeke 20

3 Conclusion **27**

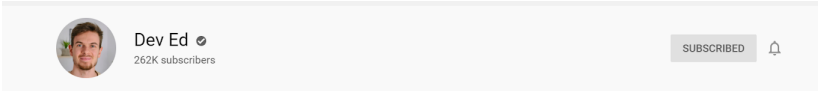
1 Introduction

In the beginning, iAmUnknown was supposed to be a group composed of four people but unfortunately, our fourth member discovered he couldn't join our team since he'd already validated the programming module (he's repeating his year). However, this didn't stop us from finishing the book of specification in time and having a plausible task division that would suit everybody.
We didn't detail the game and its rules in the book of specification because the game kept evolving. As we divided the tasks, we realized that some aspects we wanted to implement were far too complicated and were therefore not prioritized. But the game didn't only lose features, it also gained lots; when we started to work with unity, we discovered so many features that inspired us to develop our project.
In the book of specifications we provided, the task distribution was generalized for the whole project, but between a presentation and another, our goals change which means our tasks keep on changing during the whole semester. Each member was responsible enough to know what to work on. This is probably the first time any of us worked on a project where there wasn't "a group leader" in the sense that one person would be supervising everything. Our group leader, Carl, makes sure everything we do is coherent and guides us if we encounter any error with unity or have trouble understanding a concept. Since we are relying on each one's responsibility, trust was an important factor in our relationship. If we weren't as close as we are, we wouldn't be as happy and excited to work on the project.
Thuraya, who is more of an esthetic person has worked on the website (which looks awesome), the networking to implement the multiplayer function and the UI which is not yet finalized since we didn't complete the multiplayer function of the game. Carl has mostly worked on the mechanics of the game implementing the movements: walking, running, crouching, jumping, etc... Sergio has focused on generating a map that could then be edited and where enigmas could be added. To sum it up, our game started taking shape with functional movements and a potential map.

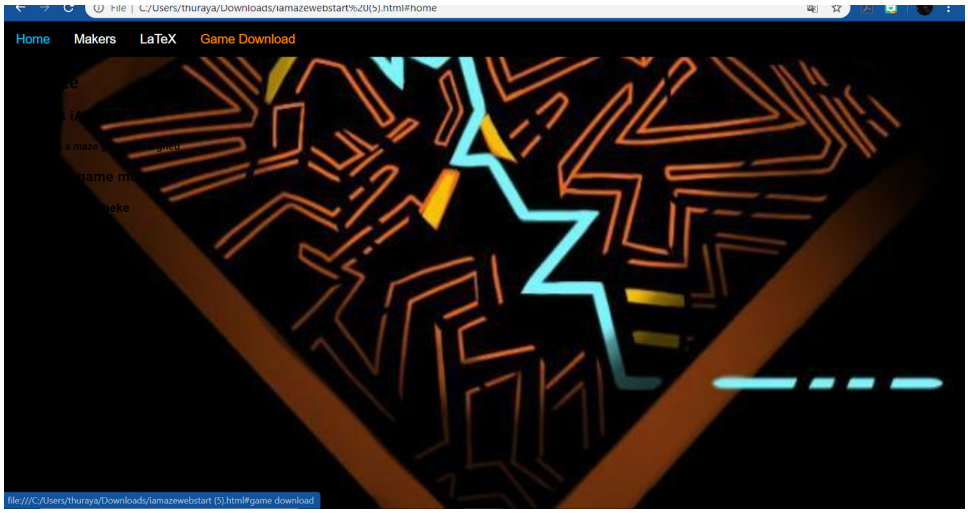
2 The Project

2.1 Thuraya Shanbari

Dividing the tasks was tedious work, it kept me weary that I wouldn't be able to finish my work on time and in the right format my teammates were expecting. This pushed me to work on everything I was assigned to do. I was very motivated. By the first presentation, I was given the task of creating our website as well as the creation of the UI (main menu, settings menu, gamemode option, and start on linking photon networking for multiplayer mode). I was very excited to make the website. It has been on my bucket list for a few years now and I'm glad I can finally say I've made a fully functional one. I'm not going to lie, I was extremely lost at first. I started using Codecademy to help me learn HTML but it didn't seem to suit my learning pace. I then put learning HTML on pause since I also had my studies to work on. Later, after midterms ended, I decided it was time to get back into learning what I started. Though this time I chose Youtube as a learning platform. I searched for a while and then stumbled upon a channel called "Dev Ed".



I watched about three of his videos and then started to understand that coding a website wasn't only HTML but also CSS and JavaScript. It then became clear to me that a website without CSS is extremely basic and very VERY ugly. It took me a total of one night and one morning to have the website functional. I was very proud of myself. Here is a before (how the website looked like before I learned HTML and CSS):



As you can tell, I was unable to resize the logo and the text was not clear.

The navigation bar was also not functional since I wasn't aware of how to go about linking several scripts with each other and so on... I was also unaware that I had incorporated CSS into my script.

Here is what my code looked like:

iamazewebstart - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  margin: 0;
  font-family: Arial, Helvetica, sans-serif;
}

.topnav {
  overflow: hidden;
  background-color: #000000;
}

.topnav a {
  float: left;
  color: #ffffff;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 17px;
}

.topnav a:hover {
  background-color: #000000;
  color: #FF8C00;
}

.topnav a.active {
  background-color: #000000;
  color: #00bfff;
}
</style>
</head>
<body>
<style>
body {
  background-image: url('iamaze-logo.jpg');
  background-repeat: no-repeat;
  background-position: center;
}
</style>
<div class="topnav">
```

iamazewebstart - Notepad

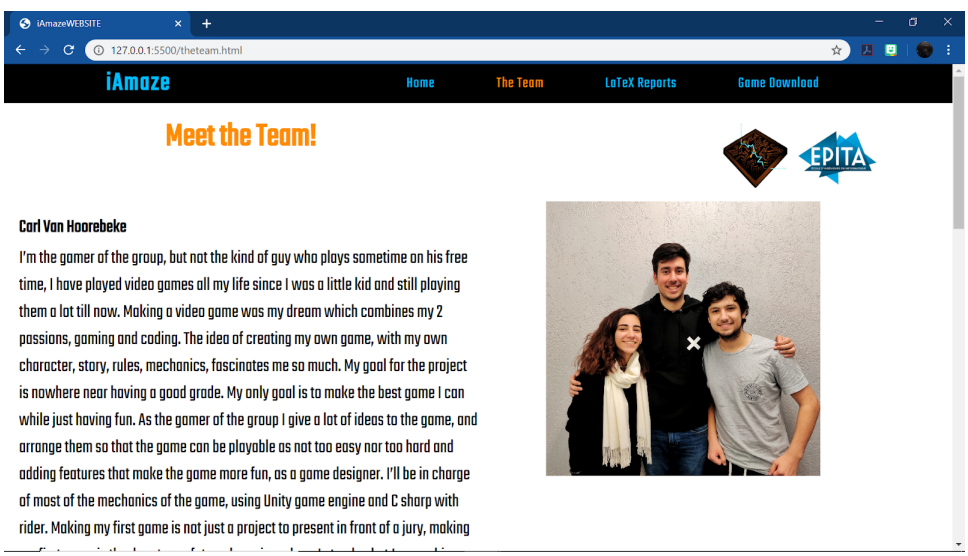
File Edit Format View Help

```
  background-position: center;
}
</style>
<div class="topnav">
  <a class="active" href="#home">Home</a>
  <a href="#makers">Makers</a>
  <a href="#latex">LaTeX</a>
  <a href="#game download">Game Download</a>
</div>

<div style="padding-left:16px">
  <h2>iAmaze</h2>
  <p>
  <h3>
        What is iAmaze?
</h3>
<h5>
 iAmaze is a maze game gneu gneu
</h5>
<h3>
 Meet the game makers
</h3>
<h4>
  Carl Van Hoorebeke
</h4>
<h5>
  carl write here
</h5>
<h4>
  Sergio Moubayed
</h4>
<h5>
  sergio write here
</h5>
<h4>
  Thuraya Shanbari
</h4>
<h5>
  thuraya write here
</h5></p>
</div>
</body>
</html>
```

I have never written code this short (not to mention this ugly) in my coding experience.

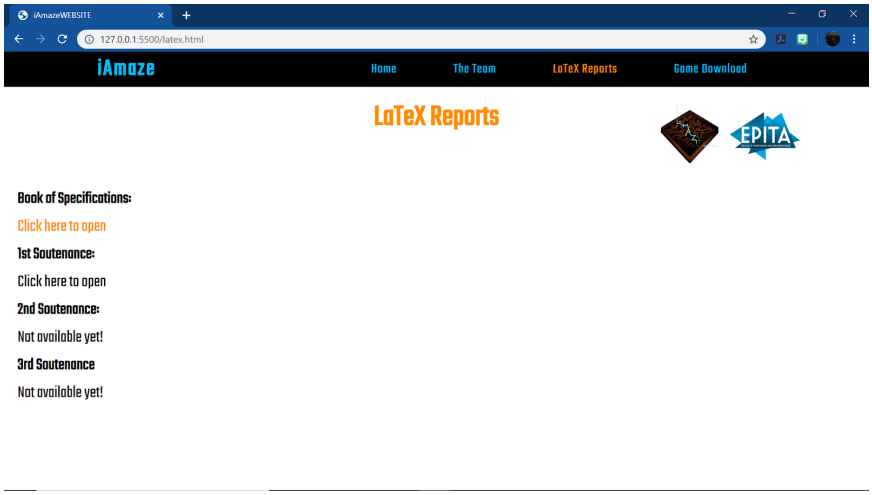
Here is the after:

For the Home section, it is still missing some text explaining the history and other requested items which will be up as soon as possible!

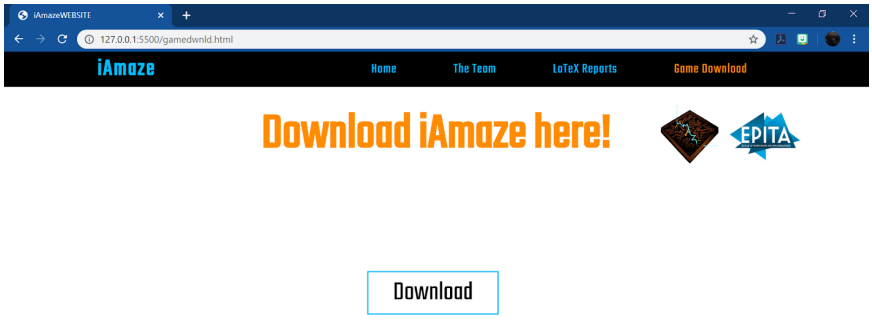


For this part, I have chosen to only include a picture of the top of the page.

As it can be noticed, there is a picture of the team. This picture not only has an animation I am rather proud to mention but also stays in place while scrolling. This effect was tricky to produce since I had changed the image position to absolute to be able to position it where I wanted to. To create an illusion, the position must have been set to sticky. This is not possible (or at least not to my knowledge after hours of searching for it) so, I decided to stick with a simpler illusion and make the image's position "fixed".



This section is pretty simple, I just made a hyperlink to open up the LaTeX reports (spoiler alert: this report will be available there too!)



Lastly, the download section. This part is not fully functional yet since we have not made a build for our game yet. I also plan on adding a second download button for the light version.

For the website, I used Visual Studio Code, I found it more convenient especially since I could run a preview on my browser.

```html
<body>
    <header>
        <nav>
            <div class="logo">
                <h4>iAmaze</h4>
            </div>
            <ul class="navbarlinks">
                <li>
                    <a class="current" href="index.html">Home</a>
                </li>
                <li>
                    <a href="theteam.html">The Team</a>
                </li>
                <li>
                    <a href="latex.html">LaTeX Reports</a>
                </li>
                <li>
                    <a href="gamedwnld.html">Game Download</a>
                </li>
            </ul>
        </nav>
    </header>
```

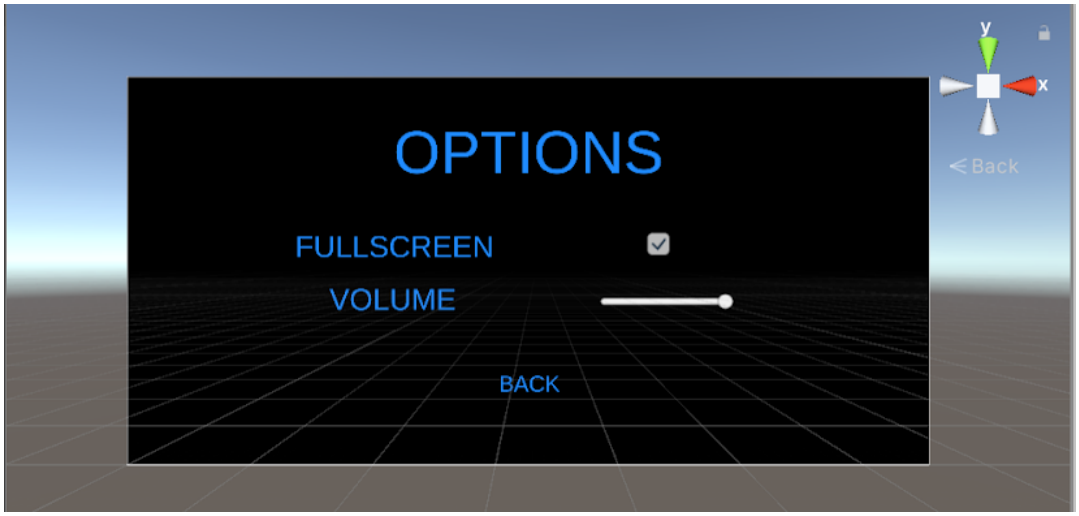
This is the code I wrote for the navigation bar.

```
.grouppic img{
    position: absolute;
    height: 95vh;
    bottom: 0%;
    left: 50%;
    padding: 150px 10px 100px 90px;
    animation: drop 1.5s ease;
    position: fixed;
}

@keyframes drop{
    0% {
        opacity: 0;
        transform: translateY(-80px);
    }
    100%{
        opacity: 1;
        transform: translateY(0px);
    }
}
```

This is the code I wrote for the animation of the group picture as well as showing a fixed position when scrolling.

Next, I was in charge of the UI. I took myself to Youtube once more and started binge watching tutorials until I decided i was ready to make it on my own. When i was ready, I launched unity and started dabbling with buttons and text mesh pro until i was happy with a design that my team approved as well. I then moved my work onto our unity collab platform and continued on there. As you already may know, there isn't much code when it comes to creating buttons and UI in general, except when their functionalities need to be implemented.

Let us take for instance the options menu:



To access these settings, small pieces of code are used:

```
1 asset usage   1 usage
public class OptionScript : MonoBehaviour
{
    public AudioMixer audioMixer;   Serializable

    Event handler   No asset usages
    public void SetVolume(float volume)
    {
        audioMixer.SetFloat( name: "volume", volume);
    }

    Event handler   No asset usages
    public void SetFullScreen(bool isFull)
    {
        Screen.fullScreen = isFull;
    }

}
```

Other than that, it was pretty simple to work on the menu. I also intend to search for a more appealing font and have it applied for the future presentations. Moving onto the more critical part of the menu: photon implementation. It was hard to start with and I'm still having difficulty grasping the whole idea of how the whole networking aspect works. Regardless, though, I believe that slowly but surely I will manage to get this to work. For now, I have started writing the code to connect to respective lobbies and rooms, but have not yet implemented it into the UI. Going into a small detail, we agreed with the team that it would be nice to show how many players are connected in a room. To do that I wrote this code:

```
1 usage
public void SetRoom(string nameInput, int countInput)
{
    roomName = nameInput;
    playerCount = countInput;
    nameText.text = roomName;
    sizeText.text = playerCount + "/4";
}
```

This will show the amount of players out of the total possible players, in our case is 4. The other variables are ones used in other scripts for Photon which I am still working on, so are susceptible to changes.

Overall, I think I managed to respect most of my deadline for this presentation, with the exception of multiplayer being at 70%. I had envisioned multiplayer functioning so, even though I have most of the code ready, it is a shame I could not have it implemented in time.

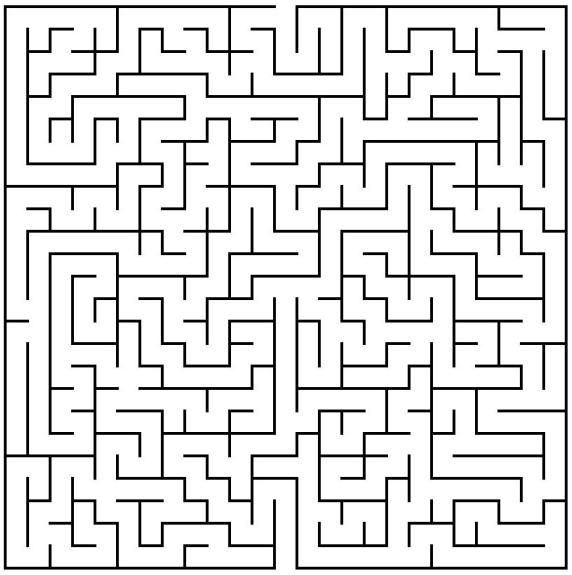
2.2 Sergio Moubayed

Thuraya and I never used unity before this project making it harder for us to work at the beginning, so Carl helped us by organizing several unity crash courses that really helped us achieve our goals.

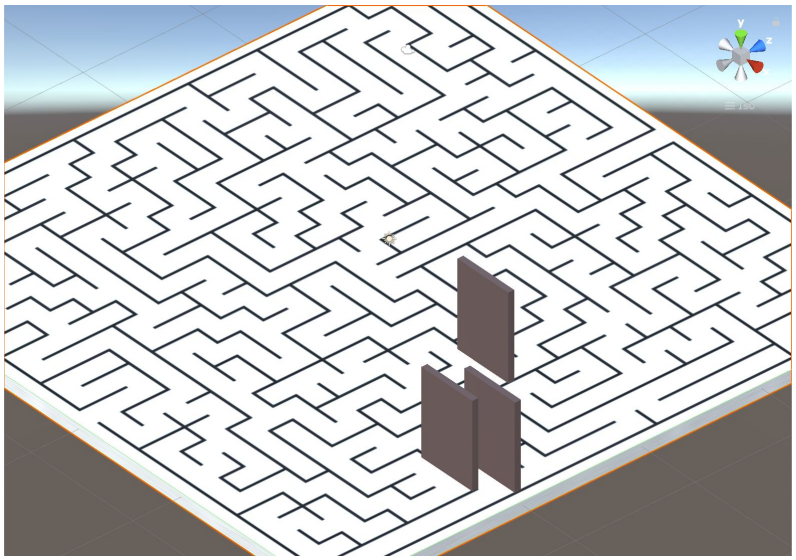
For this presentation, I needed to design and build a full maze. My task required lots and lots of research, so at the beginning, seeing myself just doing research while my colleagues were working and sending pictures and videos from their working achievements was very stressful for me. I am usually a productive person who is invested in the projects I'm in. My first plan was to draw by hand a maze on a paper and keep modifying it until the whole team is happy with the result. After finishing the design I would reproduce it on unity by just adding a prefab wall and placing it according to my design. Trying to mess around with the wall prefab I had created quickly made me realize there was a big problem with my plan: it is quite painstakingly hard to place the walls in the position we wanted keeping the angles correct and the narrowness of the maze the same.

After my first failure, I went back to research where I discovered several algorithms that would generate mazes; the most common one being by backtracking. This helped me come up with a design but still didn't solve my precision problem. After more and more research, I came up with my most efficient solution yet; generating a maze online, downloading the result as a picture, adding it as a material to the ground and finally all I'll have to do is try and align the wall prefab with the picture on the ground below it.

The following is one of my first attempts to generate a maze on http://www.mazegenerator.net/

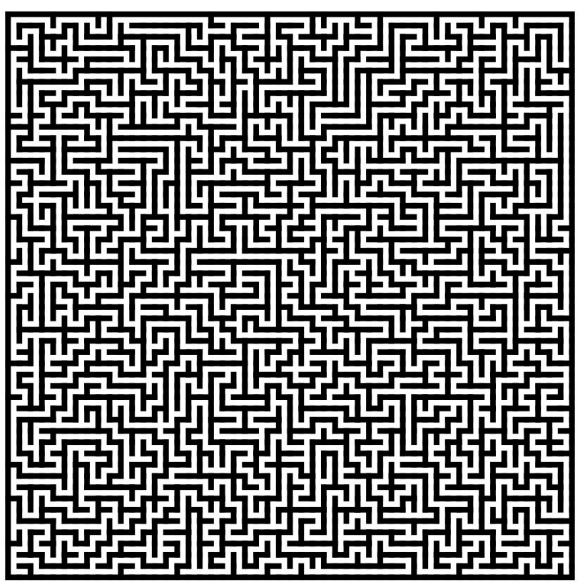


Here is how it was going to be made using this method:



After finding a reliable way to build the maze from a picture, I started researching ways to generate mazes myself to make it interesting. I chose Python as a language to do so because it is easier to prototype little projects and proof some concepts. I also found it easier to compile a maze into a picture and then use my method.

The following is an example of what a rendered maze on python looks like:



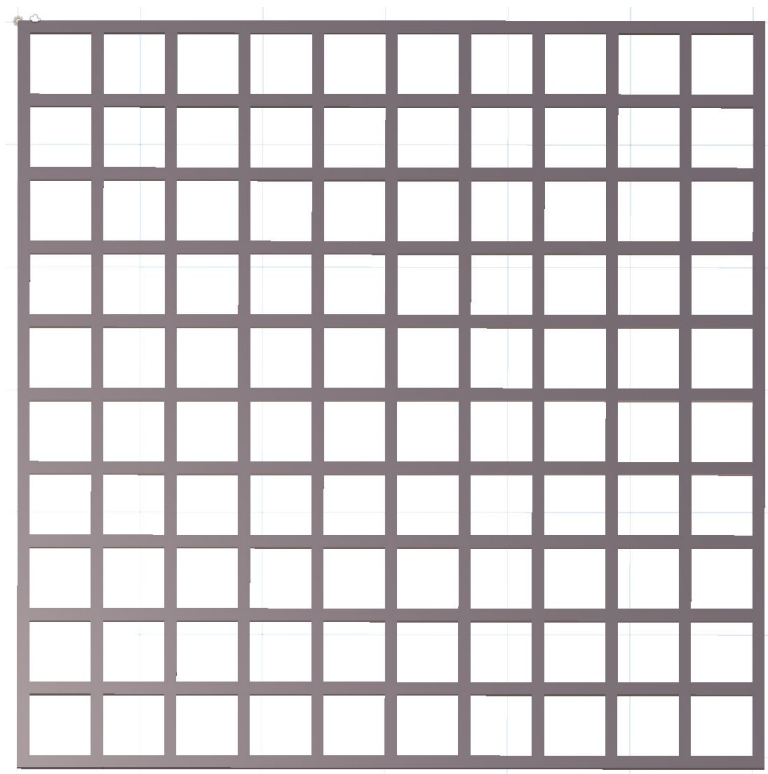
I was in the phase of researching the best algorithms to use and how to implement them in python when the magical practical had risen: Lost_in_Wonderland. The practical was all about generating mazes and solving them. Seeing how classes were used to build a maze gave me back the motivation to use the first idea I had to build the maze: let unity generate it from a matrix. Since now I knew how to represent a maze using a matrix, it seemed plausible to implement this idea and save lots of time. You may ask: why not just build it on unity? When I started working on the project and started building mazes it quickly became clear that this would take too much time to do. Having had this conversation with my teammates, we decided to stick with building it piece by piece. But now that we could implement the object-oriented version of the maze, the algorithm became clear in my mind.

Another reason is that by the third presentation we're planning to have more than one map, which means I had to repeat the painstaking process more than once. The only other problem that I had was with the computer power this whole process needed. As far as we knew it was not possible to save an instantiated game object into my assets (I didn't find anything about it on the internet). I decided to keep trying different things until I figured out how to save an instantiated object as a prefab.

I had a discussion with my teammates about my sudden change of plans. We all agreed that it would be a risky move. They told me to rely on my instincts and do as I see convenient. I finally decided to try and execute my final plan and I'm very happy with the result but I'm still planning to use another algorithm than Prim's algorithm (the one used in the practical).

I had first started by trying to build a complete maze (without any carved path) without using the maze array. It took a lot of tweaking and working on a paper for it to work. So at this point, the consisted of three double for-loop structures; I needed to use double loops to be able to build in two dimensions. One was used to instantiate the poles and the two others were used to instantiate the vertical and horizontal walls.

After doing that, I realized that it wouldn't be that easy to implement the array-maze technique if I used for-loops because more simultaneous variables were needed, so I decided to use while-loops. This gave much more flexibility and enabled me to build a maze from an array. There was again a lot of tweaking needed. After I was able to build a complete maze (without carved paths). Here is the result:

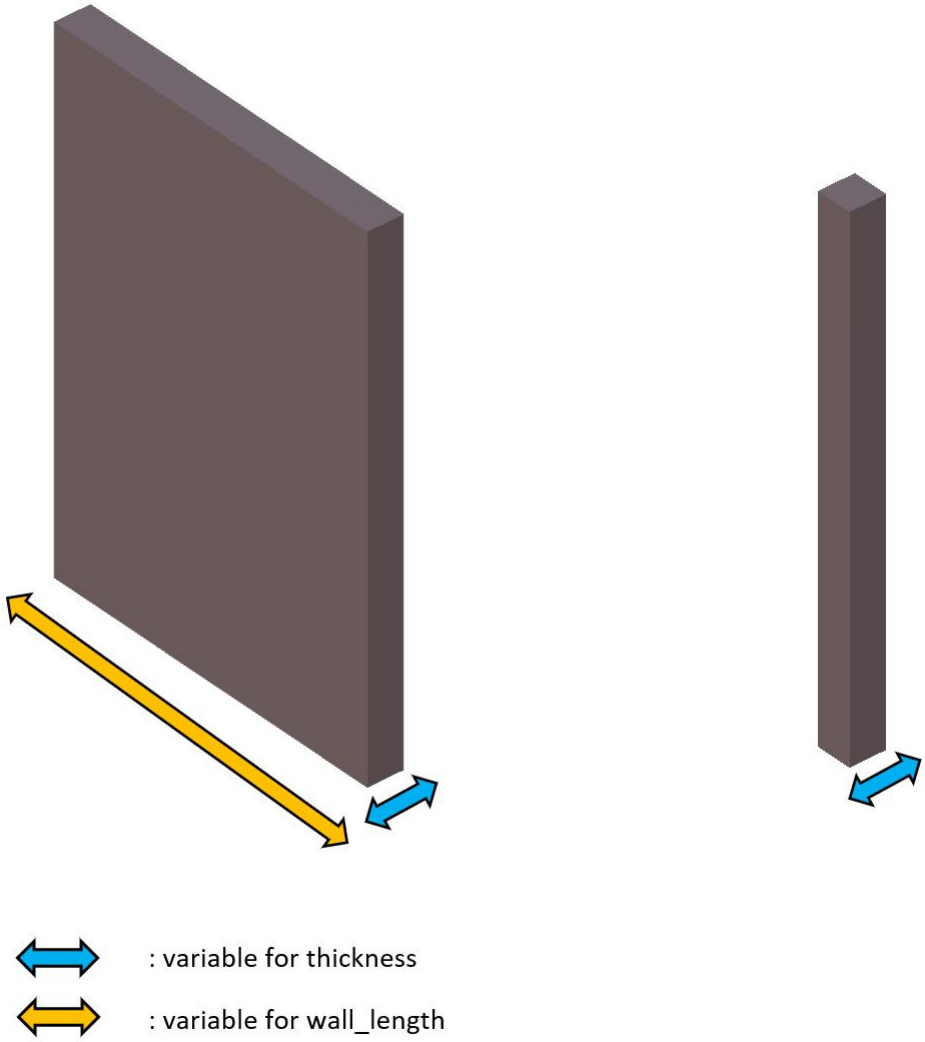


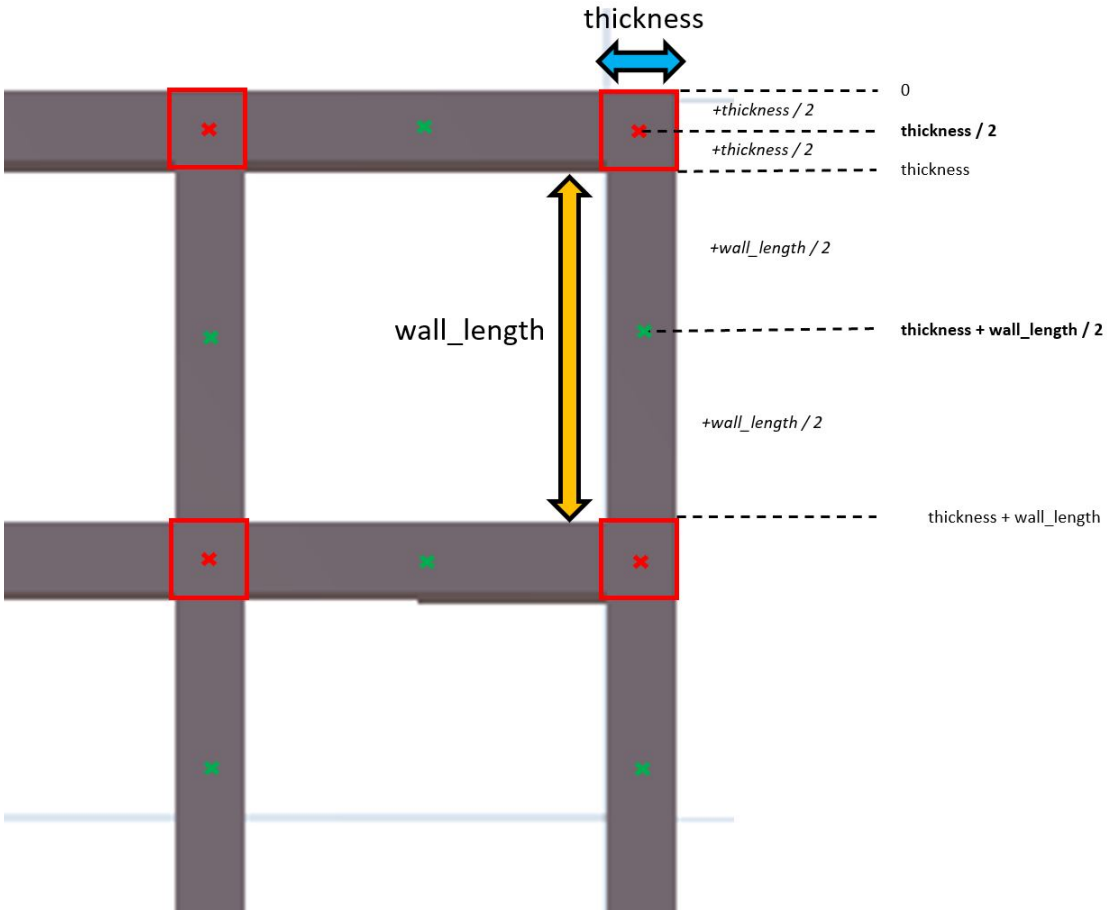
Here is the code (next page):

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Instantiator : MonoBehaviour
6 {
7     public GameObject pole;
8     public GameObject wall;
9     public static int size = 10;
10    public float thickness;
11    public float wall_length;
12    private Maze test = new Maze(size);
13
14
15    void Start()
16    {
17
18        float limit = size * (wall_length + thickness) + thickness;
19        int i = 0;
20        int j = 0;
21        float x = thickness / 2;
22        float z = thickness / 2;
23
24        while (x < limit)
25        {
26            z = thickness / 2 ;
27            while (z < limit)
28            {
29                Instantiate(pole, new Vector3(x, 2, z), Quaternion.Euler(0, 0
  , 0));
30                z += thickness + wall_length;
31            }
32            x += thickness + wall_length;
33        }
34
35        x = thickness + wall_length / 2;
36        z = thickness / 2;
37
38        while (x < limit && i < size)
39        {
40            z = thickness / 2;
41            j = 0;
42            while (z < limit && j < size)
43            {
44                if (test.Maze1[i, j].Right || z == thickness/2)
45                {
46                    Instantiate(wall, new Vector3(x, 2, z), Quaternion.Euler(
  0, 0, 0));
47                }
48                z += thickness + wall_length;
49                j++;
50            }
51            x += thickness + wall_length;
52            i++;
53        }
54
55
56        x = thickness + wall_length / 2;;
57        i = 0;
58
```

```
        while (x < limit && i < size)
        {
            Instantiate(wall, new Vector3(x, 2, z), Quaternion.Euler(0, 0, 0));
            x += thickness + wall_length;
            i++;
        }

        x = thickness / 2;
        i = 0;
        j = 0;

        while (x < limit && i < size)
        {
            z = thickness + wall_length/ 2;
            j = 0;
            while (z < limit && j < size)
            {
                if (test.Maze1[i, j].Down || x == thickness/2)
                {
                    Instantiate(wall, new Vector3(x, 2, z), Quaternion.AngleAxis(90, Vector3.up));
                }
                z += thickness + wall_length;
                j++;
            }
            x += thickness + wall_length;
            i++;
        }

        z = thickness + wall_length / 2;
        j = 0;

        while (z < limit && j < size)
        {
            Instantiate(wall, new Vector3(x, 2, z), Quaternion.AngleAxis(90, Vector3.up));
            z += thickness + wall_length;
            j++;
        }

    }
}
```

Here is an explanation of the algorithm:





Having used Prim's algorithm to create the maze (an algorithm usually used for graphs), I wasn't very satisfied with the result, there were some anomalies in the maze that required me to intervene manually. For next time, I will try to implement the backtracking technique which in theory should be better than prim's algorithm, and if some anomalies continue, I will write an algorithm to eliminate the anomalies from the maze.

Nb: I have not included the algorithms for generating the maze because they were all included in the practical "Lost_in_Wonderland".

For next time, I'll be more focused on the enigmas of the game and will this time, be using Blender to create props. I will also be working on the character's appearance along with his different animation (running, jumping, crawling). I will also be working on achieving our main goal for the next presentation, which is making the multiplayer game functional: having two different players in the same scene.

2.3 Carl Van Hoorebeke

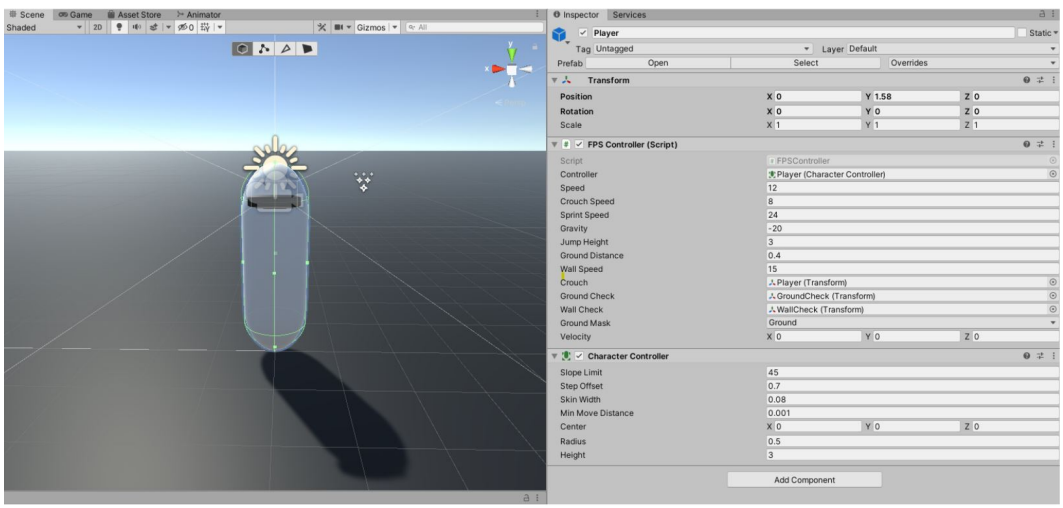
Being the only one that made Unity in the past, I was the first to start on the project. Since i had the most knowledge about video games, I knew a little about what we had to do. So I started dividing roles between us. My role was to do the mechanics of the game (mouvement, powers, etc..). I was really excited to start so, I first started a project just to learn how to do the basics. I then watched youtube videos and went on the documentation of Unity for everything that I wanted to do. I felt like I was learning a lot so I finally decided to create the project of the game. I started by creating a character to work with, which was a capsule to begin with because I thought it suited best the shape of a player. I then had two choices, either give him a "rigidbody" for the physics and code the movement or give him a "character controller" to help with the movement, but having to code the physics. I decided to give my player a "character controller" and started to code some physics.

```
void Update()
{
    Move();
    Jump();
    DoubleJump();
    Crouch();
    //create gravity
    velocity.y += gravity * Time.deltaTime;
    //apply gravity on the player (along with other forces like jumping)
    controller.Move( motion: velocity * Time.deltaTime);

}
Frequently called   1 usage
private void Move()
{

    isWalled = Physics.CheckSphere(wallCheck.position, radius: 0.6f,groundMask);


    float x = Input.GetAxis("Horizontal");
    float z = Input.GetAxis("Vertical");

    Vector3 move = transform.right * x + transform.forward * z;

    // Debug a problem when climbing up an edge
    if (!isGrounded && isWalled)
    {
        controller.stepOffset = 0.05f;
    }
    else
    {
        controller.stepOffset = 0.7f;
    }
    // Crouching
    if (isCrouching)
    {
        controller.Move( motion: move * crouchSpeed * Time.deltaTime);
    }
    // Sprinting
    else if (Input.GetKey(KeyCode.LeftShift) && !isCrouching)
    {
        controller.Move( motion: move * sprintSpeed * Time.deltaTime);
    }

    /*   // Wall Running
    else if (isWalled && !isGrounded)
    {

        controller.Move(move * wallSpeed * Time.deltaTime);
    }*/
    else
    {
        controller.Move( motion: move * speed * Time.deltaTime);
    }
```



```
}
```

```
    private void Jump()
    {
        isGrounded = Physics.CheckSphere(groundCheck.position,
groundDistance, groundMask);

        if (isGrounded && velocity.y < 0)
        {
            velocity.y = -2f;
        }

        if (Input.GetButtonDown("Jump") && isGrounded)
        {
            velocity.y = Mathf.Sqrt(jumpHeight * -2 * gravity);
        }
    }
```



With basics movement and physics working, I decided to try to go a little further and implement sprinting, crouching and double jump.

```
Frequently called   1 usage
private void DoubleJump()
{

    if (isGrounded )
    {
        doubleJump = true;
    }

    if (!isWalled)
    {
        wallJumped = true;
    }

    if (Input.GetButtonDown("Jump") && !isGrounded && doubleJump)
    {
        velocity.y = Mathf.Sqrt( f: jumpHeight * -2 * gravity);
        doubleJump = false;
    }

    if (Input.GetButtonDown("Jump") && isWalled && wallJumped)
    {
        velocity.y = Mathf.Sqrt( f: jumpHeight * -2 * gravity);
        wallJumped = false;
    }
}
```

```
Frequently called   1 usage
private void Crouch()
{
    if (Input.GetKey(KeyCode.LeftControl))
    {
        isCrouching = true;
        crouch.localScale = new Vector3( x: 1f, y: 0.5f, z: 1f);
    }
    else
    {
        isCrouching = false;
        crouch.localScale = new Vector3( x: 1f, y: 1f, z: 1f);
    }
}
```

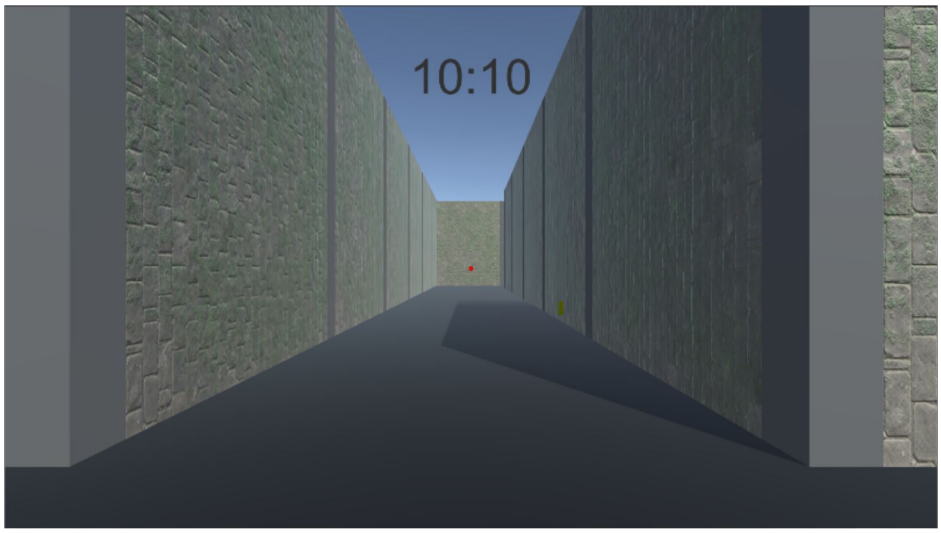
Then I started to work on the perspective of the “God” (2nd role available in the game) where the “God” will be seeing the whole map from above and move the camera like in a “Real Time Strategy” game. I made it so the “God” can move with either “WASD” or the arrow keys or with the mouse being on the edge of the screen. The camera also has some bounds that it can’t exceed, and it has the functionality to zoom with the mouse wheel.

```
private void Update()
{
    Vector3 pos = transform.position;

    if (Input.GetKey( name: "w") ||Input.GetKey(KeyCode.UpArrow) || Input.mousePosition.y >= Screen.height - panBorderThickness)
    {
        pos.z += panSpeed * Time.deltaTime * pos.y * 0.1f;
    }
    if (Input.GetKey( name: "s") ||Input.GetKey(KeyCode.DownArrow) || Input.mousePosition.y <= panBorderThickness)
    {
        pos.z -= panSpeed * Time.deltaTime * pos.y * 0.1f;
    }
    if (Input.GetKey( name: "d") ||Input.GetKey(KeyCode.RightArrow) || Input.mousePosition.x >= Screen.width - panBorderThickness)
    {
        pos.x += panSpeed * Time.deltaTime * pos.y * 0.1f;
    }
    if (Input.GetKey( name: "a") ||Input.GetKey(KeyCode.LeftArrow) || Input.mousePosition.x <= panBorderThickness)
    {
        pos.x -= panSpeed * Time.deltaTime * pos.y * 0.1f;
    }

    float scroll = Input.GetAxis("Mouse ScrollWheel");

    pos.y -= scroll * scrollSpeed * 100f * Time.deltaTime;

    pos.x = Mathf.Clamp( value: pos.x,  min: -panLimit.x,  max: panLimit.x);
    pos.y = Mathf.Clamp( value: pos.y, minY, maxY);
    pos.z = Mathf.Clamp( value: pos.z,  min: -panLimit.y,  max: panLimit.y);

    transform.position = pos;
}
```

Because of both the player and the "God" being in the same scene and having some of the same keybind I made that once one is active, the other scripts is deactivated.

```
void Update()
{
    posOfCamera = cam2.transform.position;

    if (Input.GetKeyDown(KeyCode.Tab))
    {
        if (cam1.depth == 0)
        {
            cam1.depth = 1;
            audio1.enabled = true;
            cam2.depth = 0;
            audio2.enabled = false;
            Cursor.lockState = CursorLockMode.Locked;
            movmentPlayer.enabled = true;
            mouseLook.enabled = true;
            canvas.SetActive(true);
            camMovement.enabled = false;
        }
        else
        {
            cam2.transform.position = posOfCamera;
            cam1.depth = 0;
            audio1.enabled = false;
            cam2.depth = 1;
            audio2.enabled = true;
            Cursor.lockState = CursorLockMode.None;
            movmentPlayer.enabled = false;
            mouseLook.enabled = false;
            canvas.SetActive(false);
            camMovement.enabled = true;
        }
    }
}
```

For the UI I implemented a crosshair on the center of the screen (red point) and added a timer for the player to know how much time he has left.



I tried to implement some advanced mechanics like the wall climbing/running, but it didn't work as expected so I stopped trying for the moment. For the second defense I'll be focusing on implementing more advanced mechanics like the wall climbing/running which will be part of the "power" sections, along with some "power" to the "God" as well. I'll also be implementing some of the riddles to the game.

3 Conclusion

Taking into consideration the fact that we had just started learning unity, we believe that what we achieved for this presentation is very good. It wasn't easy to try and learn completely new things online on our own. We couldn't also directly jump into the project without being sure that we knew what we were doing, so we had to start by experimenting with unity which gave us less time to work on the project.

What we have done may not seem like a lot but is quite significant for the game. First of all, we have a fully functional website in which the LaTeX reports, the book of specifications and the game (once we build it) can be found. The UI (user interface) is a very important part of the project since we will need to implement a multiplayer version, and for someone to be able to play with his friend, he will need to be able to navigate through a menu in order to choose a room to join. We didn't focus on the aesthetics of the menu but this part is easy since the menu is functional (except for the multiplayer part).
For any unity game, mechanics are essential and nothing can be done before they are programmed. We are pretty proud of how good they turned out to be and we have also implemented some potential powers into the mechanics, such as wall runs.
Another very important aspect of the game is the map. Being able to randomly generate it will make us save a lot of time in the future since we will be able to build maps much quicker and more flexibly.

For the next defense, we plan to work much more on the aesthetics of the game we will have: a character with all his animations, a definitive style for the maze (more graphics), work on the style of the UI, functional enigmas and their corresponding props and most importantly, we will have the multiplayer function working.

We believe we have respected most of the percentages mentioned in the book of specification except for the multiplayer; we underestimated how hard it would be, we are still troubleshooting errors in the scripts that let us create and join rooms. That is why we are prioritizing our work on the multiplayer section of the game for the next presentation.