Coursework 1 – Weekly questions + tutorial

304CR – Games and AI

Joao Pedro Palma

A computer and a cup of coffee on a table

Description automatically generated with low confidence

Contents

[Week 1 Exercises 3](#_Toc96066750)

[Week 2 Exercises 5](#_Toc96066751)

[Week 3 Exercises 10](#_Toc96066752)

[Week 4 Exercises 11](#_Toc96066753)

[Zombie game tutorial 12](#_Toc96066754)

# **Week 1 Exercises**

**Find and briefly describe a bug related to the AI in a game of your choice.**

The bug I’m choosing for this component question is one of many in a pretty recent game, which I have played not too long ago, this game is “Uncharted 4: A thief’s End”.

As a kid I wouldn’t notice many of the bugs from games I would play, but now since I started this course I am much more perceptive to them.

The bug in cause is none other than mission progress being stopped due to AI issues, for example, I would enter a fighting zone against many enemies, which wasn’t easy by itself since it was hard mode, and after completing this region the AI I would be with wouldn’t progress to the next session, meaning I couldn’t progress the story, making me having to reset the game to the previous checkpoint and do the same area again to potentially fix the bug.

I searched online and found many other people complaining about the same issues, I found this person saying the following “My girlfriend's been playing U4 and she's currently at Chapter 6, the moment where Sam is supposed to give you the grappling hook, however she's telling me that never happens.” (Adzey, 2017). [His girlfriend got stuck in chapter 6 due to AI bug]. GameFAQs.

**What do you think causes this bug?**

**I think this issue is caused due to fast progression, meaning, I would enter the fighting arena, kill all the enemies quickly and progress to the next stage, but since I’d do it so quickly the friendly AI must’ve get bugged somehow, making it stay still in a certain spot of the level not allowing me to progress.**

**The major issue must’ve been my lack of patience, the enemy AI and interactions and the friendly NPC AI that must’ve been causing conflict to each other creating these kinds of funny glitches.**

﻿**Give your answer in terms of how you think the underlying algorithms are working.**

In my understanding the friendly AI should advance to the next session/open the next area after all the enemies are dead and the puzzle (if there is one) are completed, so I know for a fact that all these criteria must’ve been met progress in the story.

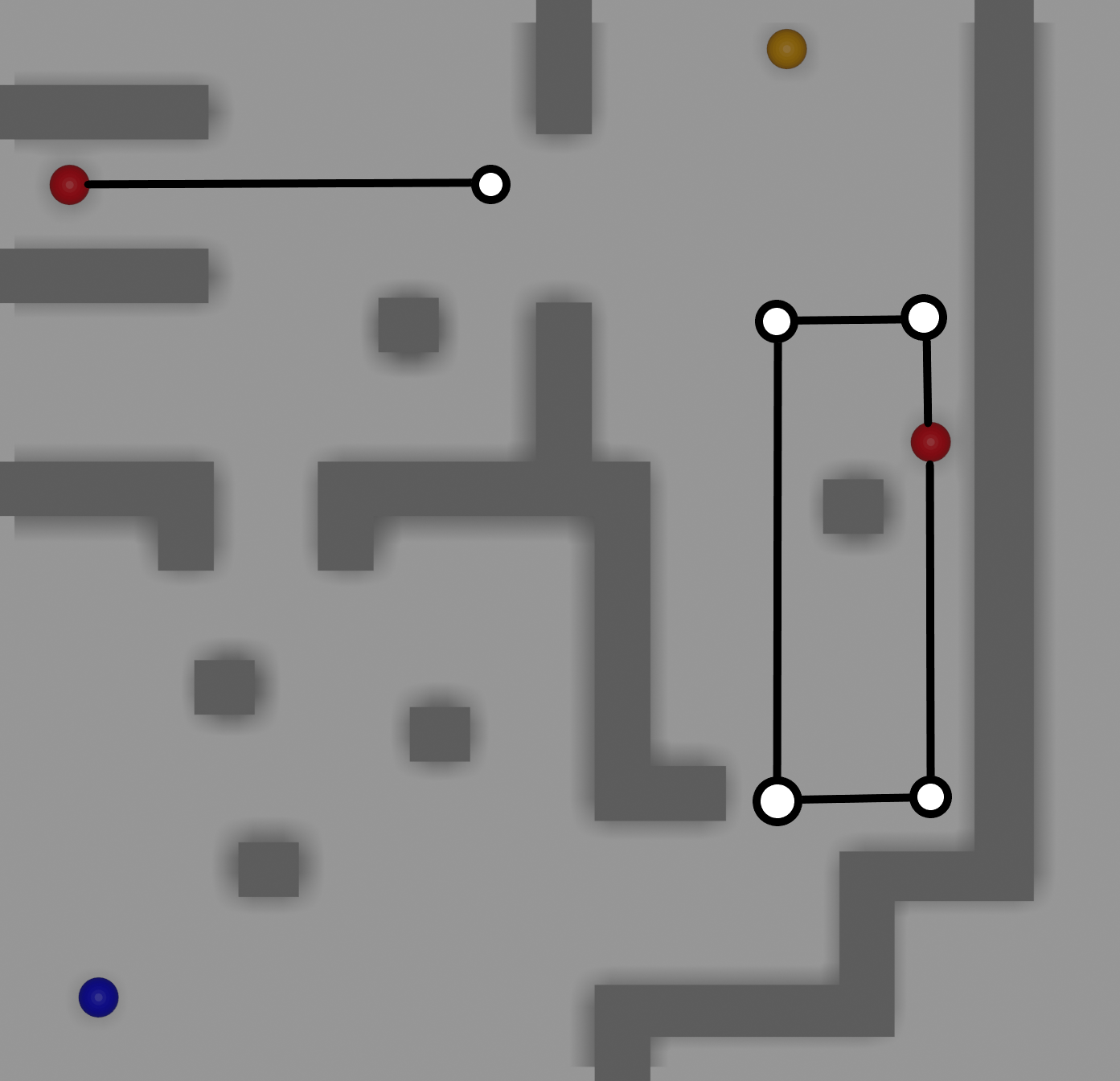
Maybe some pathing issue or a bugged enemy in the level (i.e. inside a wall) wasn’t allowing me to continue further into the game.

Normally the friendly AI would help you kill a few enemies, give you some tips on how to progress and maybe help you interact (with a long jump, or lifting something heavy), I noticed that it was mostly on the section where I needed the actual NPC help to advance further.

Going back to the example from another player in the 1st question, I tried to look for a solution in the same thread and found this “All I can say is that you need to look at the beam to work. Sam gives it to Nate in a scene. Try pressing L3 while looking towards it to see if that works. Also, try waiting for Sam…” (megaman623, 2017) [Explaining other player how he could possibly progress]. GameFAQs, but if you follow the thread you will find that this did not fix it, making me believe in this case that Sam’s AI was glitched stopping the player from progressing.

# **Week 2 Exercises**

**Where would you position the waypoints for each guard to create the best gameplay?**

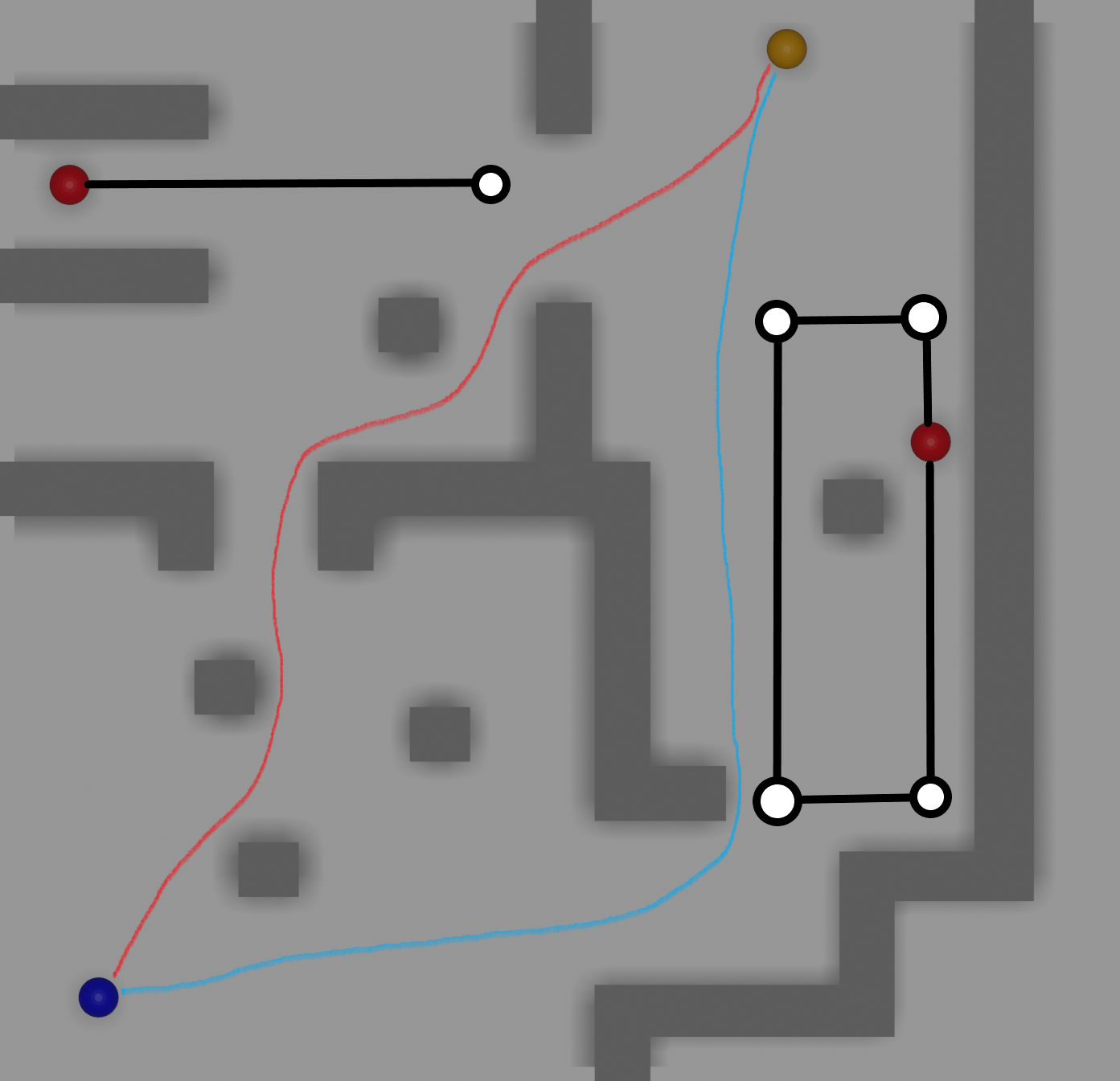


These are the paths that I would personally pick. I took the routes into consideration, having the longer one being the easiest and the top left the hardest.

I wanted to make the level somewhat difficult but rewarding in some areas and not so much in others, thus creating some balance in terms of gameplay.

When I thought of these paths I took speed running into consideration as well and that is why the paths are so funny looking but simple.

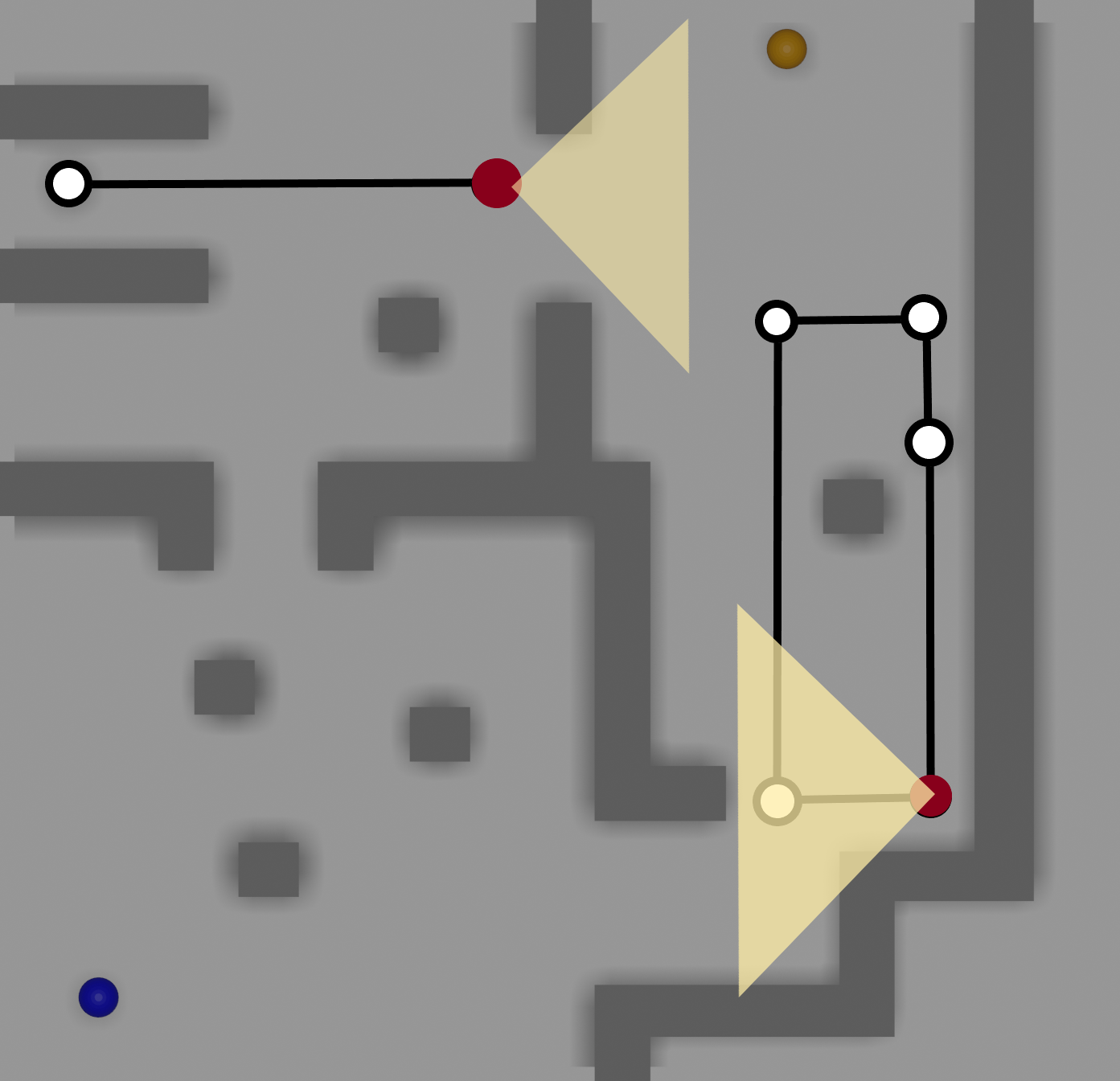
To explain how this pathing would work here are some examples:



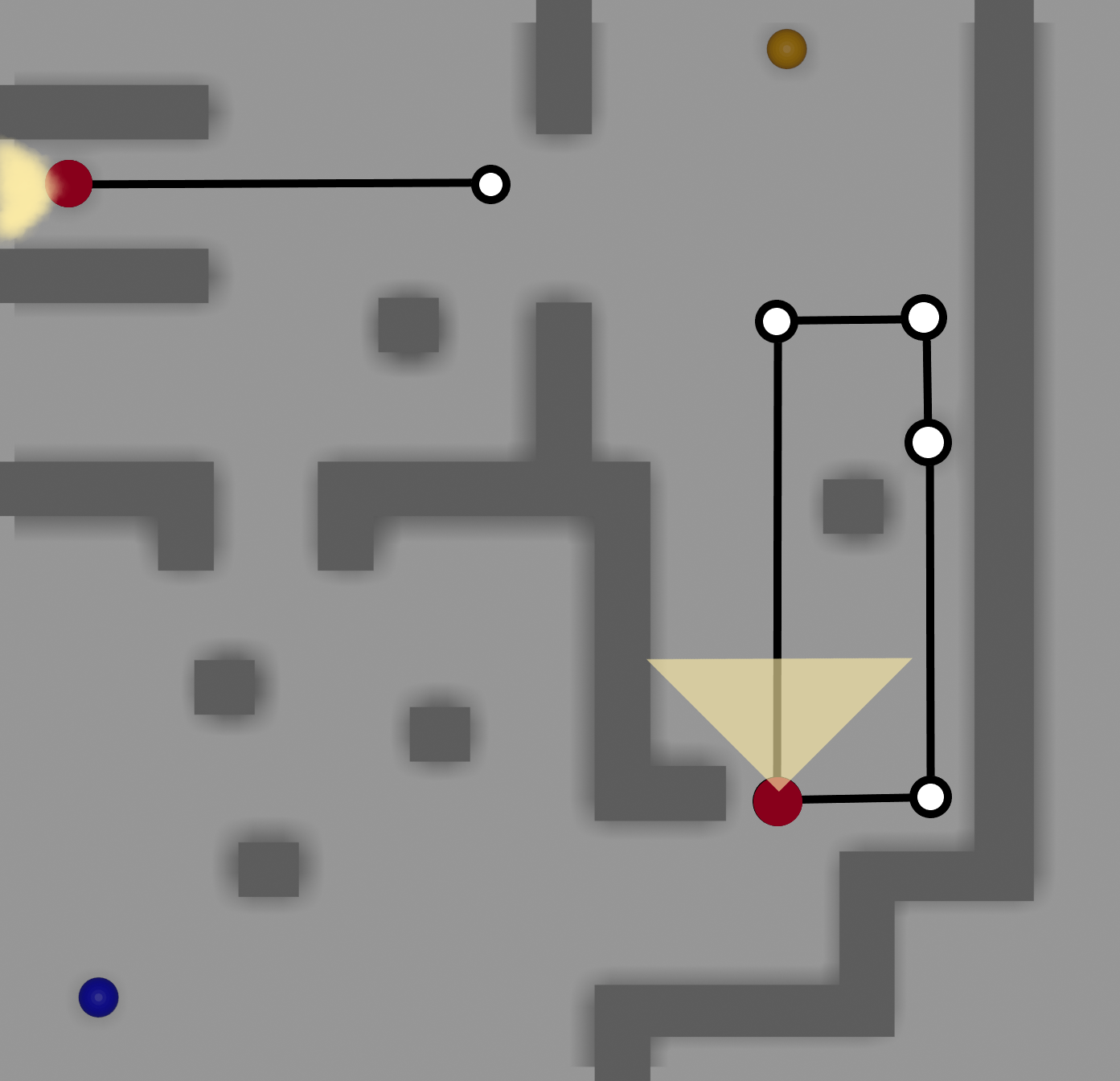
The top left being the shortest and quickest path to the treasure it would only make sense to increase its difficulty, having the guard making a shorter thus quicker path helps achieve the wanted challenge. This also allows the guard to cover for the other one when their routes don’t coincide.

The bottom right has the longest way around for maybe newer players, being the easiest path, this way does shove off a lot more time, where in a speed run would not be beneficial.

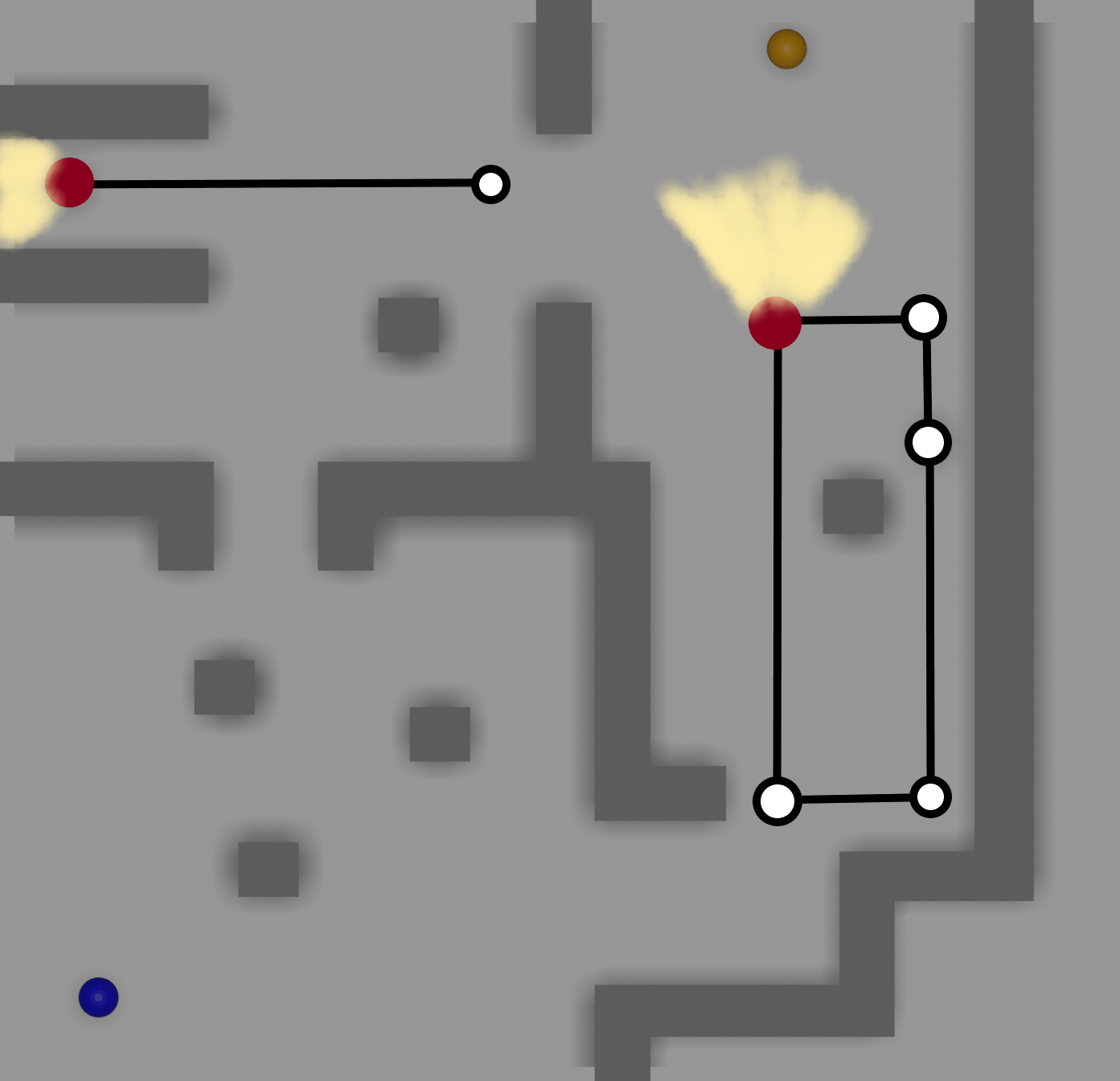
With all this said I think these would be optimal path for a level of this scale and difficulty and over doing it would make it too hard.



To give some context on the guards routes and their vision range I added visual aids, and as you can see, these cover a good portion of where the player would go through, what I really liked about the paths I chose is the fact that both guards at one time will be covering for their blind spots, onec again not making the level too easy and having the player thinking better and time his route to achieve the goal.



As the top left guard goes back, the top path becomes “open” to traverse, but as you can see, as soon as the player is reaching the final entry point to the treasure, the other guard is coming straight up to potentially catch the player.



The same applies for when the player chooses to go the long way around, while the guard on the right would lose vision from the player he would be free to go straight to the treasure, but since the top left guard path is much smaller, making faster path rotations, the treasure is always somewhat guarded.

# **Week 3 Exercises**

Identify an **open problem** in AI are explain if/how it relates to Game AI

On my research about AI and some problems that are still to be solved until today, I have found some interesting subjects which I could branch out, but I decided to focus on one which intrigued me the most, this being their inability or lack of exhibit a common sense.

What do I mean by this? As we already know AI is nothing but a bunch of instructions that the programmer gives it, essentially creating it’s “personality” or behaviour, now AI unfortunately or fortunately (I’ll leave it to your criteria) still hasn’t evolved enough to gain a “brain” or a way of being self-sentient, this of course does not stop AI from evolving from its current stage and possibly in the future gaining this sentience I’m talking about.

What it does do is, not having this essence can make AI a bit clumsy when in decision making, when in contrast a human player has a common sense so it know what to do by itself and how to do it in the most effective way, AI does not achieve the level of common sense we humans have, which can affect gameplay in some levels.

All this does not mean that AI are weak or underdeveloped at this stage, it simply means that this is an area that AI could evolve more into, and eventually will, that creates a different type of connection and playability in games, the way I see it is, imagine you’re playing against an AI which doesn’t run on just programming, an AI that actually knows what it’s doing and has some sort of personal knowledge of how the game (or world, if we’re talking about AI like Sophia) works.

In my opinion and final words, AI will evolve to a point where it does allow for this type of behaviour, until then us players have to content ourselves with this “basic” AI format. In simpler words, AI with a common sense to it would make it less predictable and more challenging for players, it would make the game genre even bigger than what it has become, not knowing what the AI’s next play would be or having completely no idea of what the AI could potentially do would allow for some unique interactions, for example some crazy outplays just because the AI has some sort free will so to speak.

# **Week 4 Exercises**

**What are the most common AI approaches, and problems, in a game of your selected genre?**

For the game I’m making for my coursework 2, I decided to go for a stealth game, I really like stealth games as they challenge your wits and your reflexes and timings to reach a certain goal.

AI in these games is pretty simple and straight forward, usually in these types of games, the enemies tend to have a predefined path which they always follow, and if the player make any noise or is seen the guard or enemy starts chasing them or catches the players having repeat the level.

Basically this is how an AI of this scale would work:

* Walks in a predefined route;
* Has a conical vision for detecting the player;
* If the player makes any sound or the guard catches a glimpse of him, he will start the chase;

I can see where some problems may start to arise, probably the guard’s detection with sound which makes them investigate where they heard the sound comes from may be hard to get working correctly, as it depends on many factors, such as knowing where the noise came from and making him approach it with care.

Some problem I see with some modern stealth games is that AI in them are always clunky, meaning, they lack some sort of intelligence in terms of detection and following noises, where the AI gets all confused and starts shooting or running towards a spot where the player is not.

Another thing to look for is the guard’s behaviour, as in how they react once they see the player, will they run towards him, will they shoot at him? What happens if the player gets away, will they keep searching or give up? These are all good question on how to approach different issues.

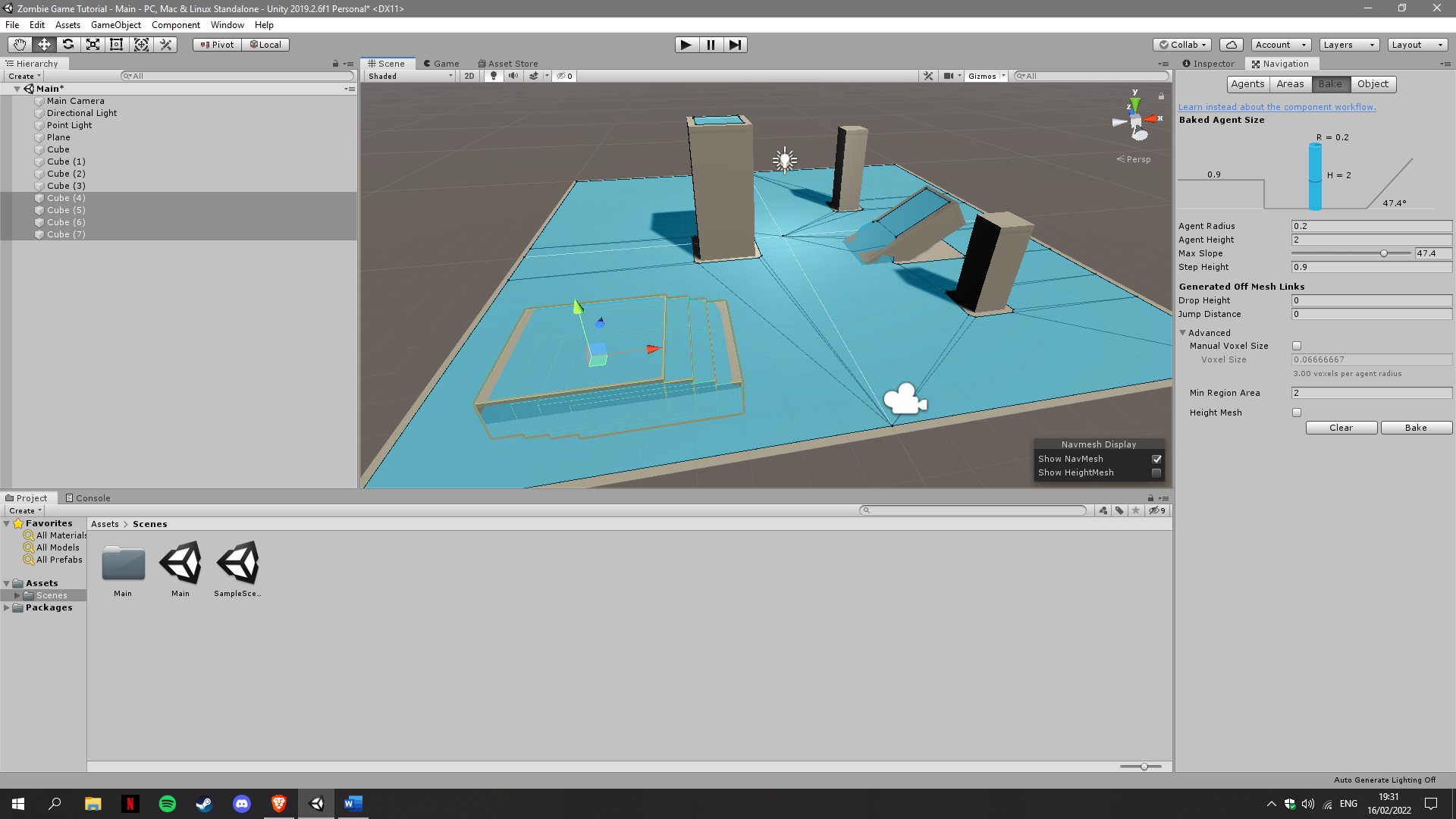
# Zombie game tutorial

For this section of the coursework, I followed the tutorial which is given to us in the module page on aula, the first tutorial video from what I understood is where we use the unity’s navigation to define the areas the AI can walk through or on to.

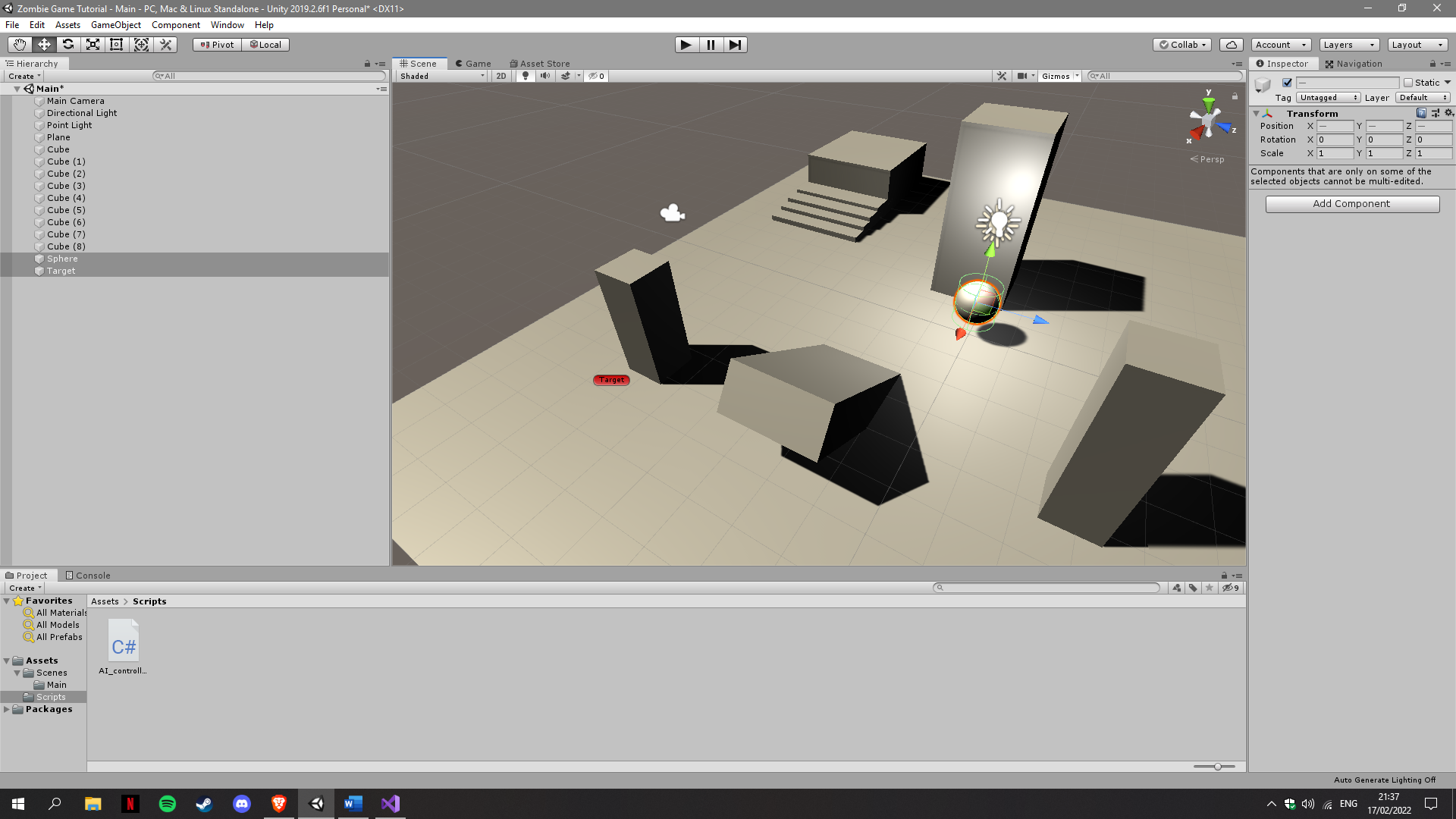
To start off with, I created a plane and a few cubes to create some objects to better understand how baking these objects create the pathing areas the AI can walk on. Below is an image of the first few minutes of the video where we are taught how to bake and create the pathing, messing with the angle of slopes, steps and such, the blue area you can see is where the AI will be able to access during gameplay.

To achieve this stage, you must firstly insert some objects, select them and make them static (on the upper right tick box, on an object inspector), then you go to Window > AI > Navigation, which will pop up a window called navigation on the right-hand side of the program.

Once you have navigation open, you can start baking objects, take their heights and rotations into consideration and make sure you leave some room on the sides of the objects to make sure baking is done perfectly.



After setting up all the surfaces using the baking method of unity, I continued to follow the tutorial, I started by creating a sphere to be my zombie placeholder and created a waypoint as a reference for the AI’s movement.



I proceeded to create a script and attached it to the sphere, in this script I created a public variable called ‘player’ to get the target position for the AI to follow. This is currently a placeholder and will be changed in the future of course.

Now all I have to do in order for the sphere to move is add the following line:

agent.SetDestination(player.position);

I also messed around with the baking setting and these were the optimal I found after a few playtests (taking the map I created into consideration).

A picture containing chart

Description automatically generated

Now I had to create the prefab for the sphere, but I decided to include a spawn point for the spheres so that they would automatically spawn and chase the player.

Using a empty game object, I used it as a spawn point and attached a script to it, firstly I defined the prefab I wanted to spawn in, using a public variable:

public GameObject agentPrefab;

And in the start function I include a for loop to spawn 4 iterations of the same prefab:

for (int i = 0; i < 4; i++)

{

Instantiate(agentPrefab, new Vector3(0, 2, 0), Quaternion.identity);

}

Now I had to find a way to make them chase the player without using a public variable to receive the GameObject, I did so by using this command:

player = GameObject.FindGameObjectWithTag("Player");

And of course, I attached the “Player” tag to the target.

To help distinguish the player from the “zombies”, I created a material and added a random colour.

At this point in the coursework, the game looks like this:

Graphical user interface, application

Description automatically generated

As you can see I have a player controller script is already present, and I got basic movement done in 2 different ways, I did a basic input scheme, where I got the horizontal and vertical movement of the player.

This is how it looks (having predefined position as current object position):

position = transform.localPosition;

position.y += Input.GetAxis("Jump") \* Time.deltaTime \* jumpHeight;

position.x += Input.GetAxis("Horizontal") \* Time.deltaTime \* moveSpeed;

position.z += Input.GetAxis("Vertical") \* Time.deltaTime \* moveSpeed;

transform.localPosition = position;

I also did the tutorial version, which is this (having predefined agent as NavMesh):

if (Input.GetMouseButtonDown(0))

{

RaycastHit hit;

if (Physics.Raycast(Camera.main.ScreenPointToRay(Input.mousePosition), out hit))

{

agent.SetDestination(hit.point);

}

}

For this next part, I downloaded some animations and a character model from mixamo, I imported them into unity and created an animation controller, this is how it looks:

A screenshot of a computer

Description automatically generated with medium confidence

And these are the modifications I did on the script:

if (Vector3.Distance(player.transform.position, transform.position) < 1.5f)

{

anim.SetBool("isWalking", false);

anim.SetBool("isAttacking", true);

anim.SetBool("isIdle", false);

anim.speed = 1f;

}

else if (Vector3.Distance(player.transform.position, transform.position) > 1f)

{

anim.SetBool("isWalking", true);

anim.SetBool("isAttacking", false);

anim.SetBool("isIdle", false);

anim.speed = 2.5f;

}

else

{

anim.SetBool("isAttacking", false);

anim.SetBool("isIdle", true);

anim.SetBool("isWalking", false);

anim.speed = agent.velocity.magnitude / 2f;

}

Of course don’t forget to add the controller in the inspector, for the zombies.

For the next portion I implemented the player model and animations to go along with it. I started off by getting both from mixamo and insert them into the game, once again I created an animation controller and messed around with the animations.

The only different thing in this process was the variable that is used for controlling the animations, which looked like this:

Graphical user interface

Description automatically generated

I also had to add a bit of code, which also looks like this (in the player controller):

anim.SetFloat("toward", agent.velocity.magnitude);

For the last part I’m going to add ragdoll and a dying system for the player, to add a bit more realism, just for reference, the game looks like this at this point:

Graphical user interface, application

Description automatically generated

For the ragdoll section, I selected the player and hit GameObject > 3D > Ragdoll and this money popped up, where I then proceeded to add the corresponding model parts:

Graphical user interface

Description automatically generated with low confidence

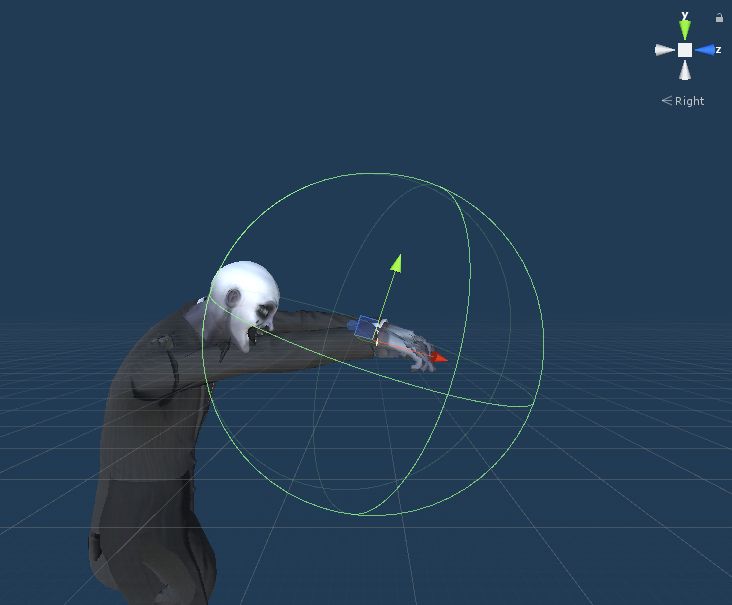
This is how my player ended up looking like, you can see all the collision boxes done for the ragdoll, now when the animator is deactivated the ragdoll enters in effect.

Graphical user interface

Description automatically generated

Now all that’s left to do is add a way for the player to die, to do so I added a collider into the zombie’s right hand (due to the swing animation), and when this hand touches any of the players collider, it activates the ragdoll.

This is how the zombie’s hand looks with the collider:



Finally, I added a new script to this hand called Hitbox, which activates the ragdoll for the player, this is the code:

private void OnTriggerEnter(Collider other)

{

if(other.GetComponentInParent<Player\_controller>() != null)

{

Debug.Log("Ere");

other.GetComponentInParent<Player\_controller>().Die();

}

}

And on the player, to make all the colliders be touchable, I created a list of rigidbodies for each part of the model:

Text

Description automatically generated

For the killing the player function:

public void Die()

{

foreach (Rigidbody rb in allRbs)

{

rb.isKinematic = false;

}

anim.enabled = false;

agent.isStopped = true;

}