

# EXPLORATION OF THE CORE COMPONENTS OF THE NAVMESH TOOL WITHIN UNITY.

OneDrive link to prototype:

[https://livereadingac-my.sharepoint.com/:f:/g/personal/ip821912\\_student\\_reading\\_ac\\_uk/Eph7u2ReqsxHn7TsYyxoWTYBE43Jh\\_bUzEqTR5Ivk48QoA?e=Zbfwa3](https://livereadingac-my.sharepoint.com/:f:/g/personal/ip821912_student_reading_ac_uk/Eph7u2ReqsxHn7TsYyxoWTYBE43Jh_bUzEqTR5Ivk48QoA?e=Zbfwa3)

Written by: Luke Collinson

Unit: CEM242 Advanced visualisation and interactive technologies

Assignment: Individual Project (report + prototype)

Wordcount (not including references, captions, or appendices): 1489

## CONTENTS

AIMS/ RESEARCH QUESTIONS	I
RESEARCH ACTIVITES	
- INTRO	I
- ACTIVITY 1 - NAVMESH SURFACES	2-6
- ACTIVITY 2 - NAVMESH AGENT	6-8
- ACTIVITY 3 - NAVMESH OBSTACLE	9-10
- ACTIVITY 4 - NAVMESH LINK	11-12
REFLECTION, PROTOTYPE AND FUTURE USES	13
BIBLOGRAPHY AND REFRENCES	14

## Exploration of the core components of the NavMesh tool within unity.

NavMesh (short for Navigation Mesh) is a data structure which describes the walkable surfaces of the game world and allows to find path from one walkable location to another in the game world. (Unity Documentation, 2020)

### **AIMS/ RESEARCH QUESTIONS**

This independent research project will look to evaluate the functionality, ease of use, and explore the limits of the tool through the 4 core components that make up NavMesh. These being:

- NavMesh Surface                    – Defines a traversable surface
- NavMesh Agent                   – Moving AI sprite
- NavMesh Obstacle               – Block portions of the traversable surface
- NavMesh Link                     – Allows traversing disconnected surfaces

To help tackle this issue, the independent research project has been divided into 4 activities. Each activity will explore one of the four key components and the tools within the components, eventually producing a prototype that showcases all four activities and some potential future uses of the tool.

Activity 1 – NavMesh Surfaces. This task will involve setting up a traversable surface to enable the other 3 activities. I shall be specifically looking at the NavMeshSurface component and its tools within the component.

Activity 2 – NavMesh Agent. This task will involve creating a script that controls an agent that navigates its own path around the NavMesh Surface following a user’s directional input.

Activity 3 – NavMesh Obstacle. This task will involve creating obstacles that aim to block the agent’s path, these shall be both static and dynamic.

Activity 4 – NavMesh Link. This task will involve connecting 2 disconnected NavMesh surfaces together.

The independent project activities shall be completed on unity build 2020.3.24f1 using a window 64bit PC. The NavMesh components tools used in some of the activities are not in the standard unity install and require you to visit: <https://docs.unity3d.com/Manual/NavMesh-BuildingComponents.html> to download the required assets.

### **REASEARCH ACTIVITIES**

I have initially set up a basic testing environment consisting of 3 objects. A ground cube (green) walls (grey) and a cylinder/ sprite (white) this environment will be our starting point to explore the NavMesh tool and its capabilities. Refer to fig 1.

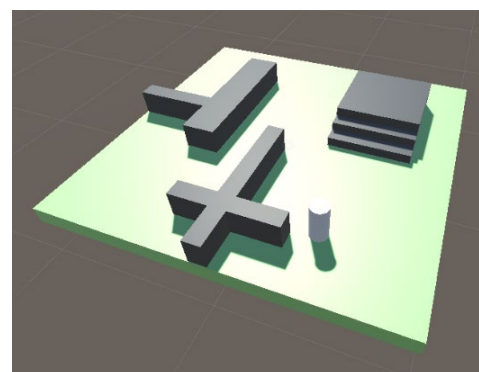


Figure 1 - test environment

## Activity 1 – NavMesh Surfaces.

I have created an empty under the hierarchy tab and renamed it NavMesh and applied the component 'NavMeshSurface'. Upon clicking bake unity detects surfaces and applies a traversable area. The traversable area showcased in a tinted blue area. Note, only the top plane of our surfaces has had a NavMesh applied. Refer to fig 2.

The component 'NavMeshSurface' itself has a wide range of customisable features, within this section we shall investigate a few of these further whilst developing our final prototype to demonstrate the features of these tool.

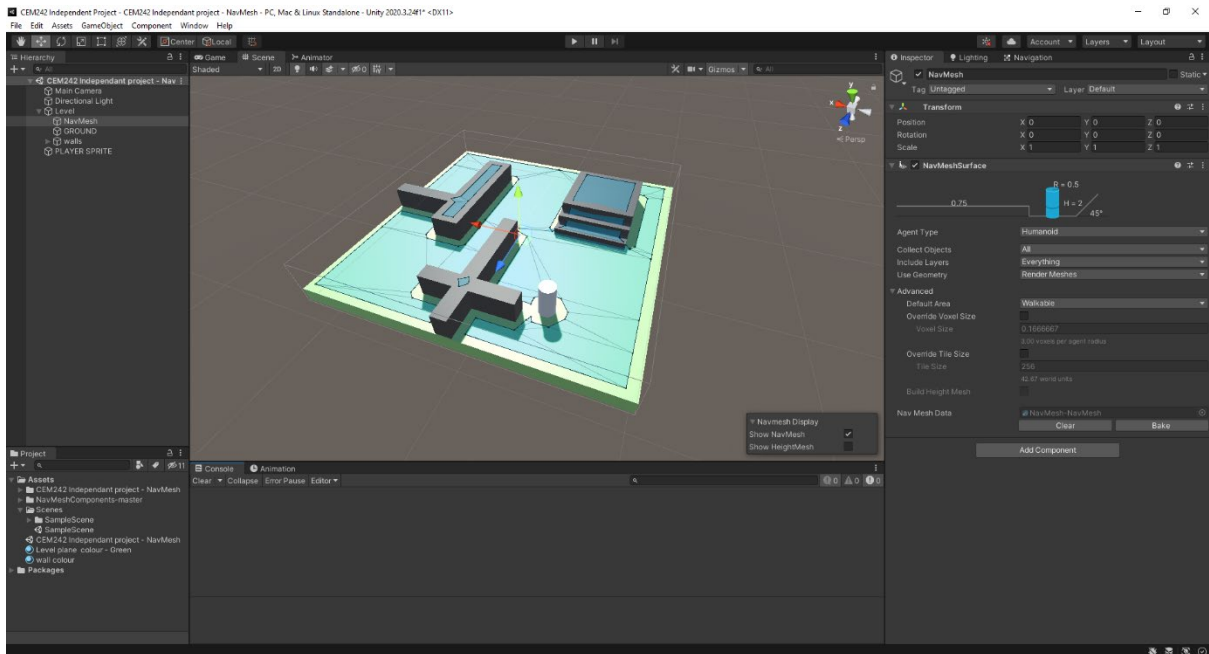


Figure 2 – Test environment

The NavMeshSurface component also works on externally imported models, refer to figure 3. Here you can clearly see what is detected as walkable space with the light blue tint but some areas, possibly due to level changes within the model have been miss detected.

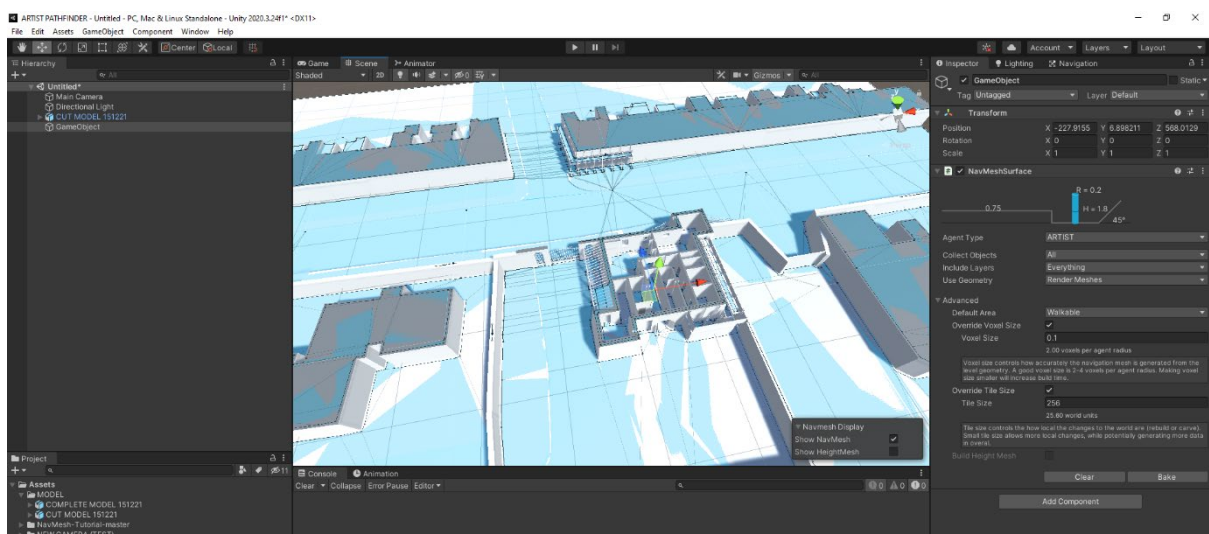
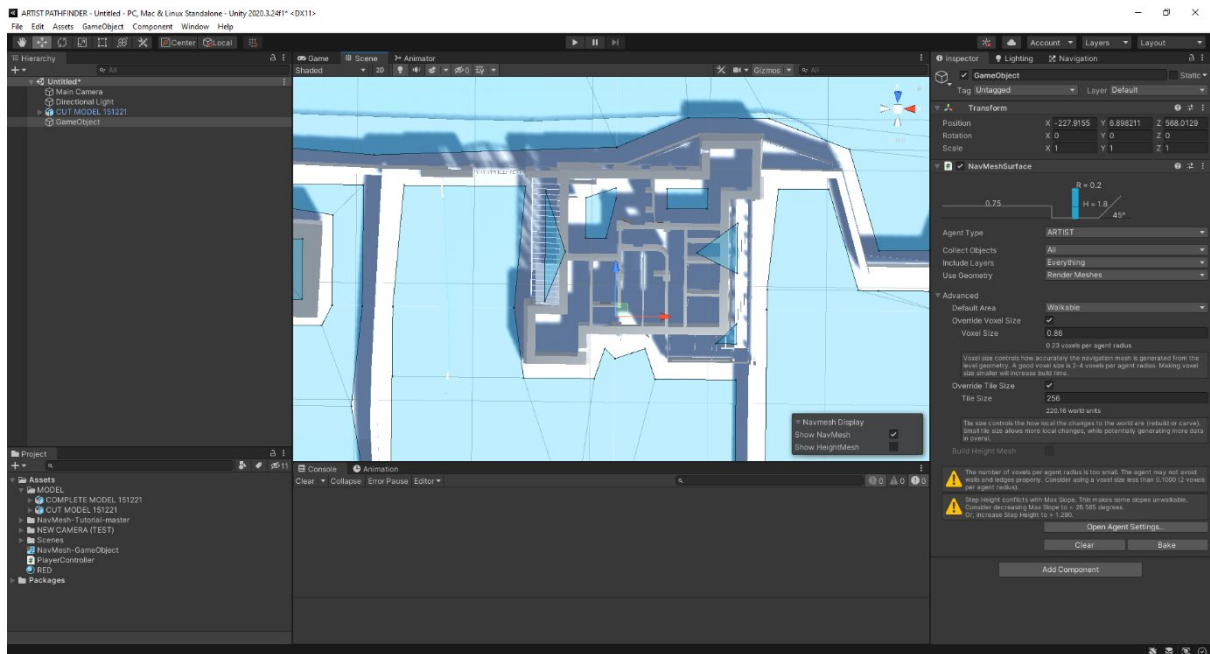


Figure 3 - Applying NavMeshSurface component to an externally imported model

Through changing the voxel size, we can create a more defined ground, but with the lack of accuracy you lose the recognition of the internal spaces. Refer to fig 4.



The NavMesh Surface component also handles custom painted terrain. Interesting due to only the top surfaces of objects becoming traversable spheres are limited. Refer to fig 5.

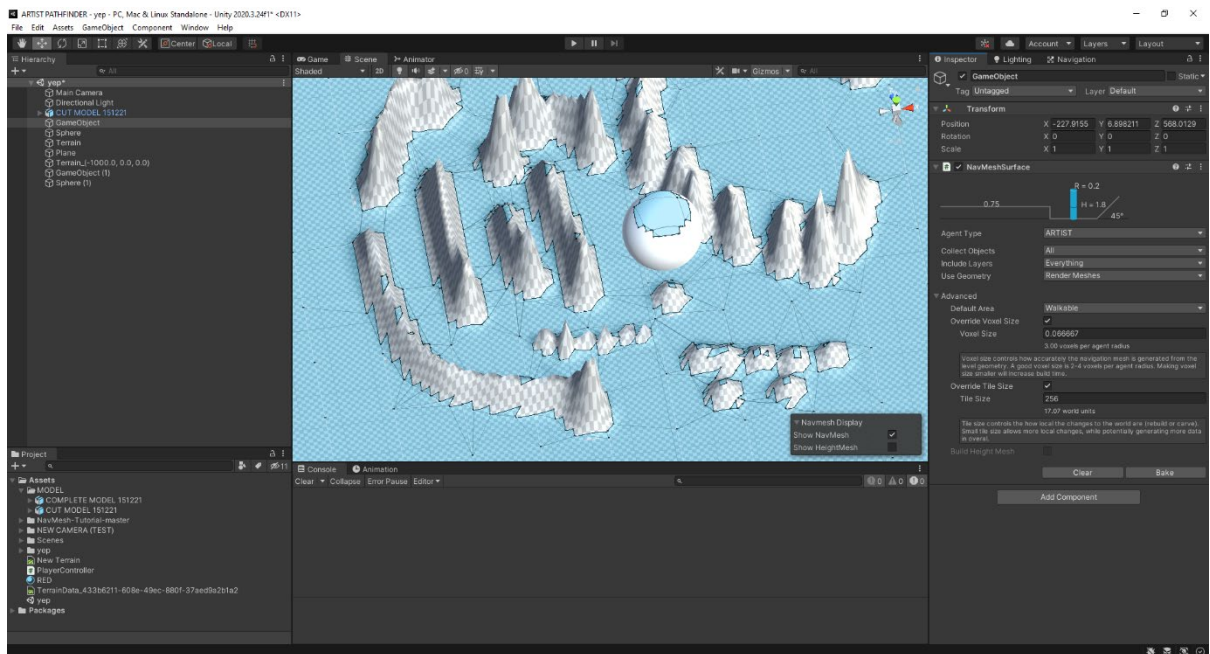


Figure 5 - exploring different shapes with NavMesh

The cylinder shall become our agent but first we need to make sure that the NavMesh ignores the cylinder when rendering the traversable area, so we need to put the cylinder on to new layer. Refer to fig 6.

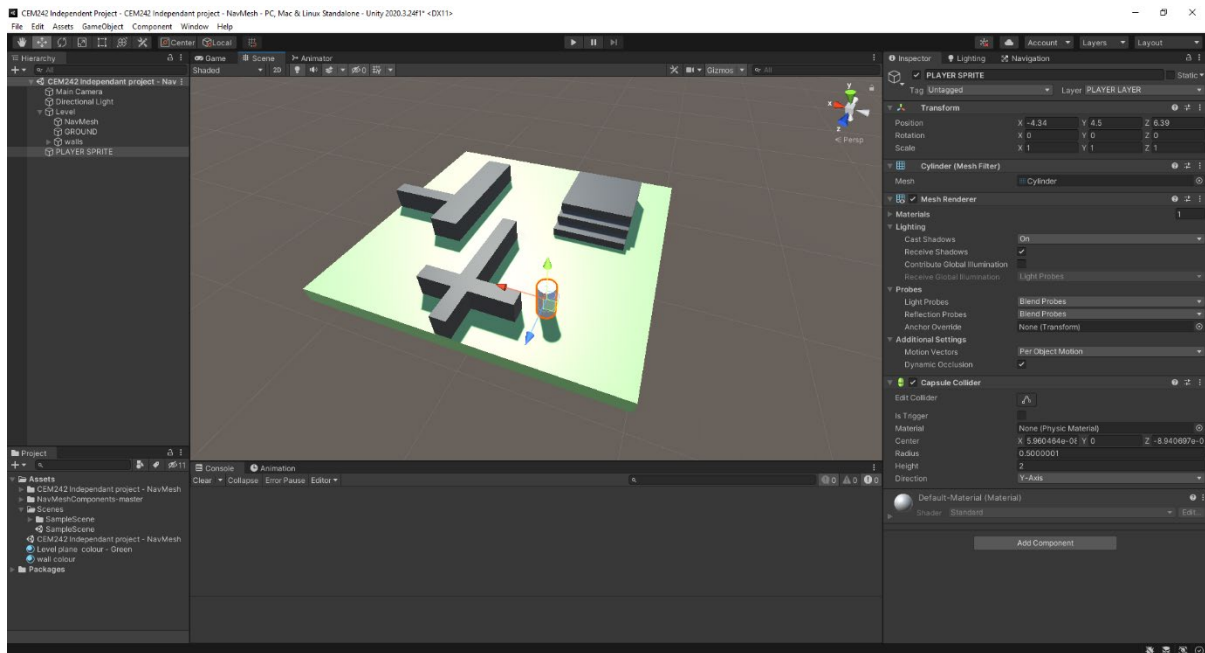


Figure 6 - assigning the cylinder its own layer

Once the cylinder has been placed onto the player layer you can manage what layers are detected by the NavMesh using the NavMeshSurface component. Under 'include layers' once the newly created player layer has been unticked and upon re-baking, the NavMesh no longer detects the cylinder. This tool under NavMeshSurface allows us to control which parts of our created environment we want to bake. Refer to fig 7.

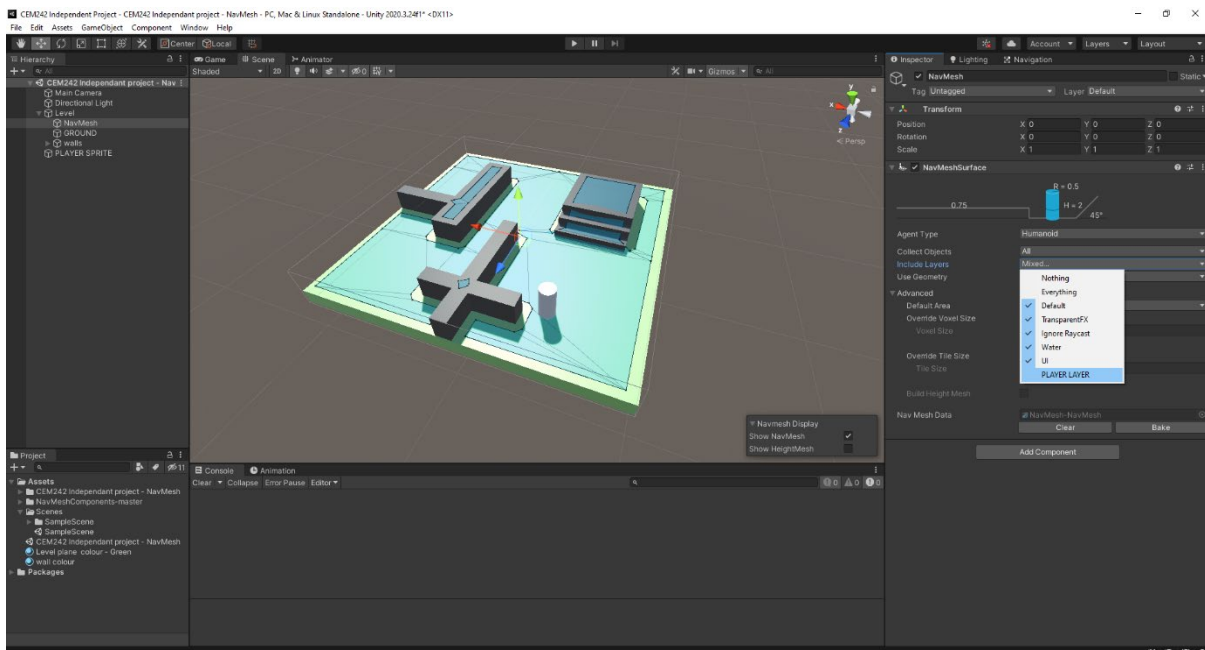
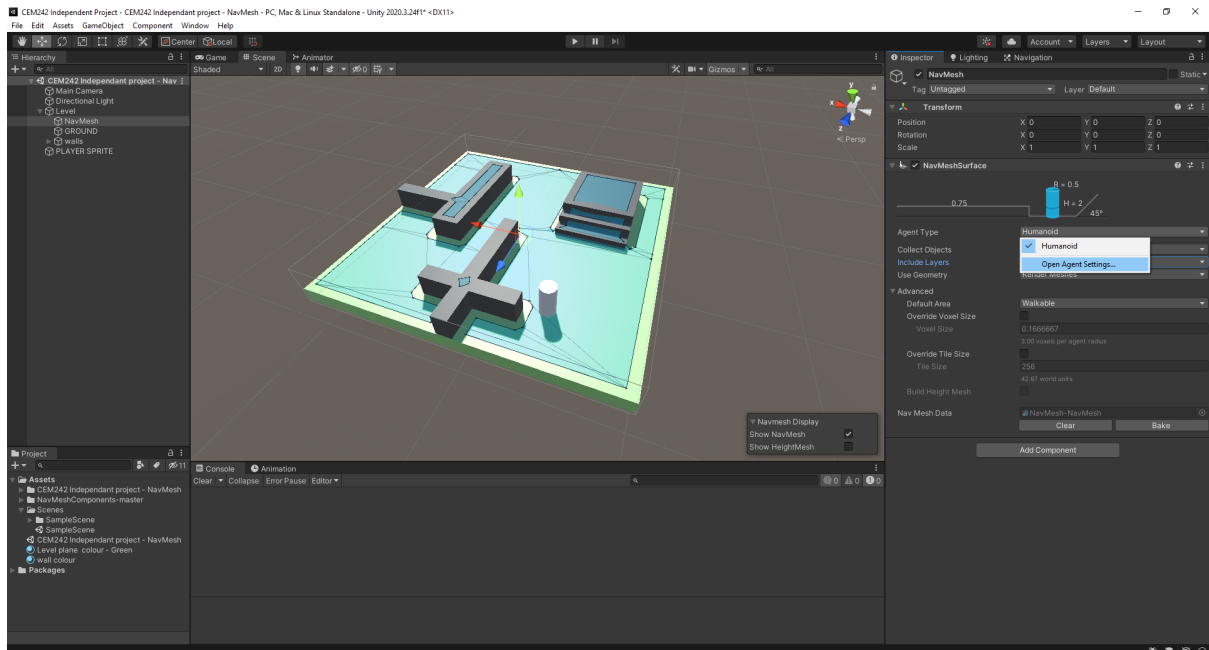


Figure 7- unticking the player layer and re-baking makes sure the NavMesh bakes the complete ground floor

Within the NavMeshSurface component we can also control how agents and NavMeshes interact. We can change these settings under the agent type drop box and opening the agent settings, this settings tab can also be found at that top right under navigation. Refer to fig 8.



Currently the default settings for the agent are set to human, within the settings we can change the current parameters and created new agents, I have created a giant agent. Refer to fig 9.

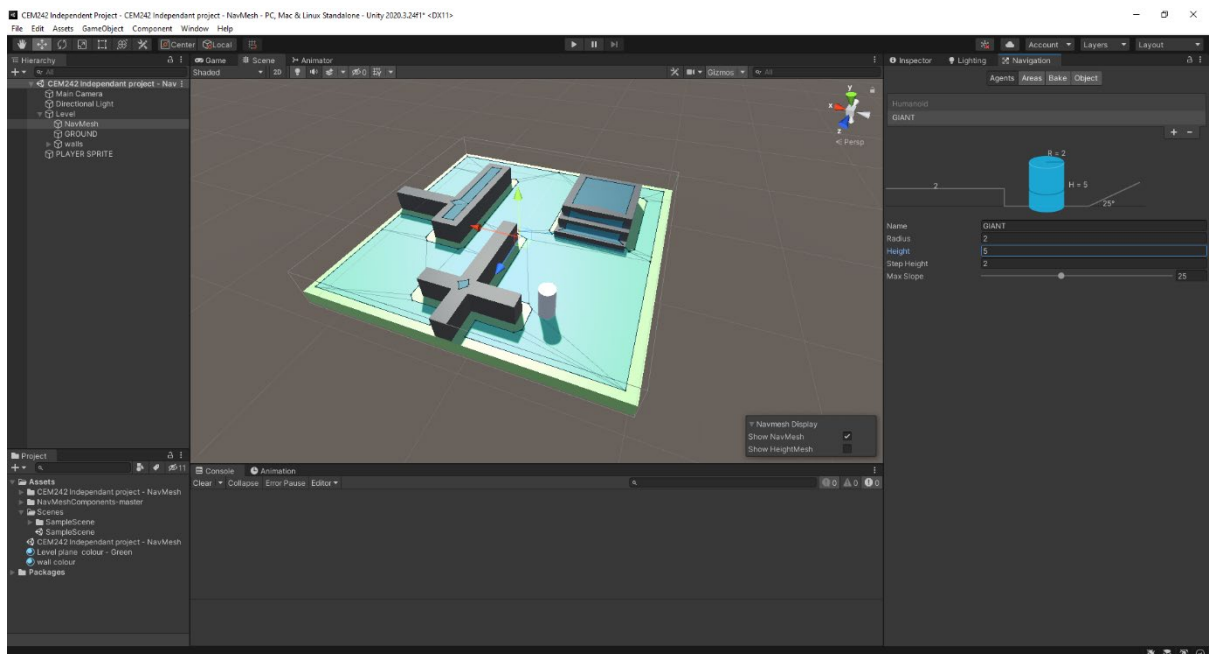


Figure 9 - custom agent created

Returning to the inspector tab for the NavMesh we can change our agent to our newly created giant and upon re-baking the NavMesh you can see that the newly traversable area has been baked for the giant agent. Refer to fig 10.

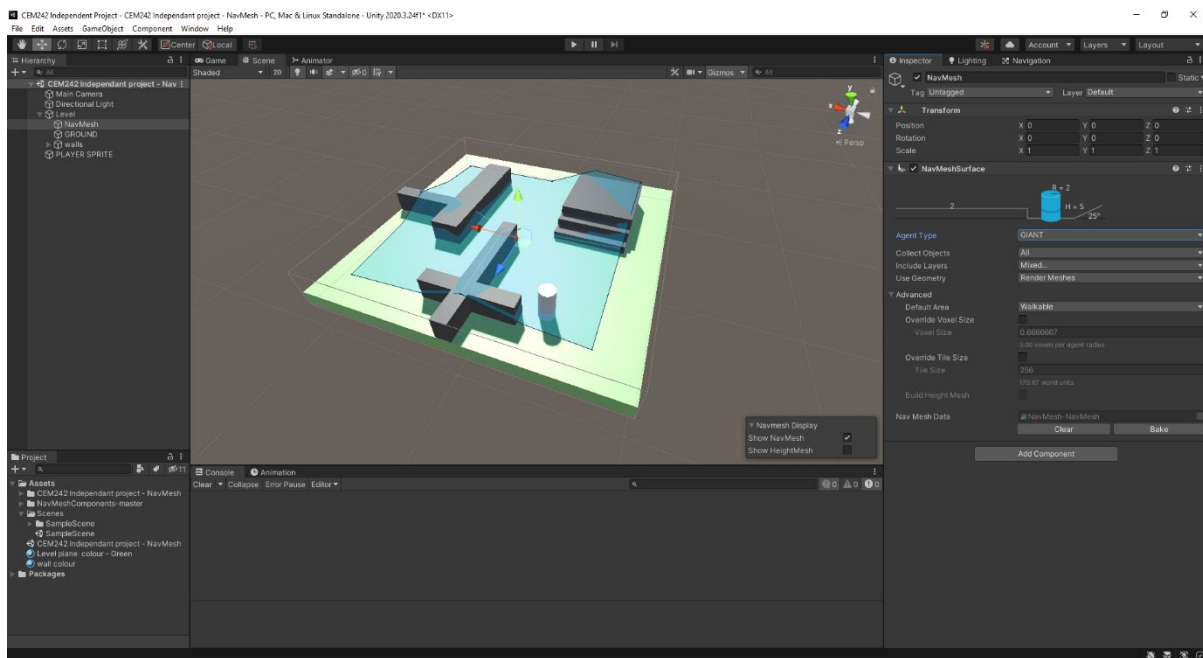


Figure 10 - baked NavMesh for custom agent

## Activity 2 – NavMesh Agent.

The NavMesh agent is agent that can navigate around the mesh. To explore this component first we must click on our agent (cylinder) and add the component 'NavMesh Agent'. Within NavMesh agent you are able to change parameters that will affect the character. Refer to fig 11.

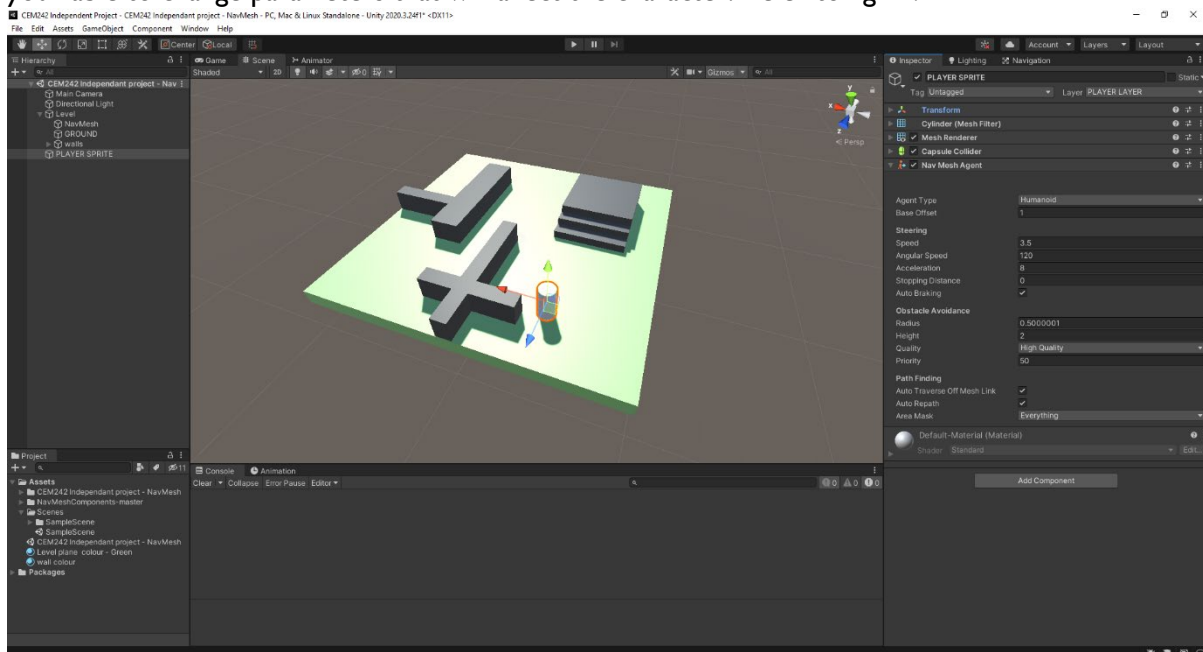
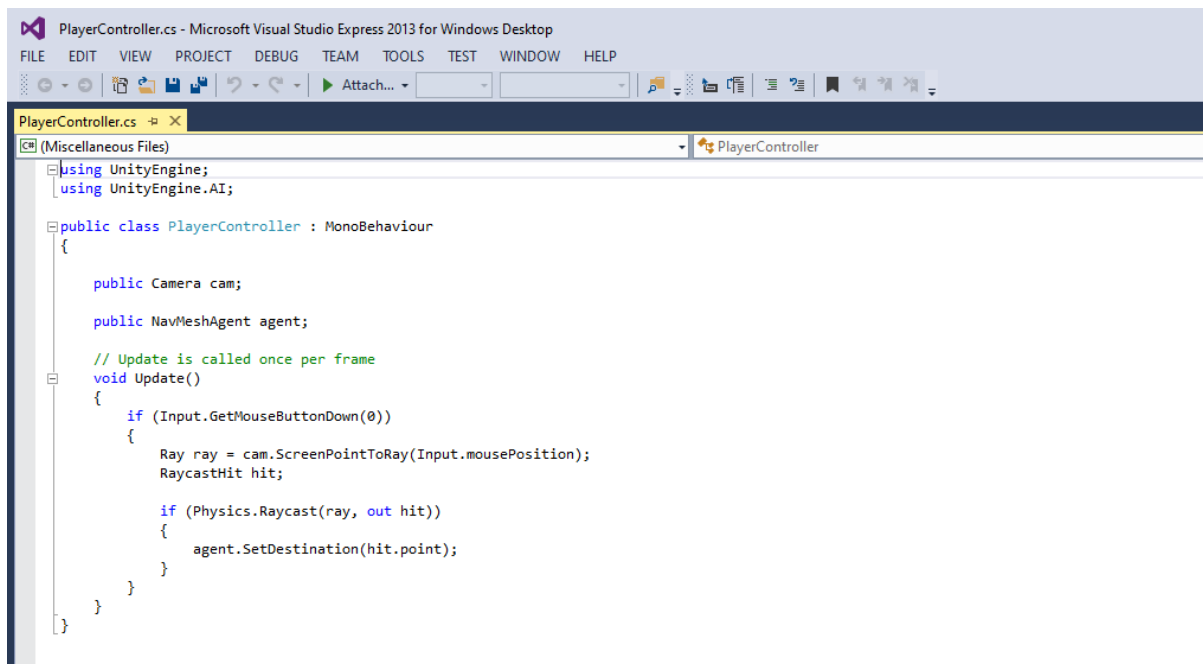


Figure 11 - NavMesh Agent added to the cylinder

Firstly, before we start editing the parameters of our agent, we need them to move. Moving agents around on a NavMesh is easy. The basis of the script shall revolve around `SetDestination()`; this will



allow us to assign a position our agent shall move to. To add the script, I add component and click new script. The code below takes the point you left mouse click and tells the agent to move there. For code refer to Fig 12.



```
PlayerController.cs - Microsoft Visual Studio Express 2013 for Windows Desktop
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST WINDOW HELP

PlayerController.cs (Miscellaneous Files) PlayerController

using UnityEngine;
using UnityEngine.AI;

public class PlayerController : MonoBehaviour
{
    public Camera cam;

    public NavMeshAgent agent;

    // Update is called once per frame
    void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {
            Ray ray = cam.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;

            if (Physics.Raycast(ray, out hit))
            {
                agent.SetDestination(hit.point);
            }
        }
    }
}
```

Figure 12- code for controlling agent movement

Upon saving the code and reopening unity the script shall update and create two new boxes under the script. One for the camera (used for the user input) and agent (moveable sprite). Refer to fig 13.

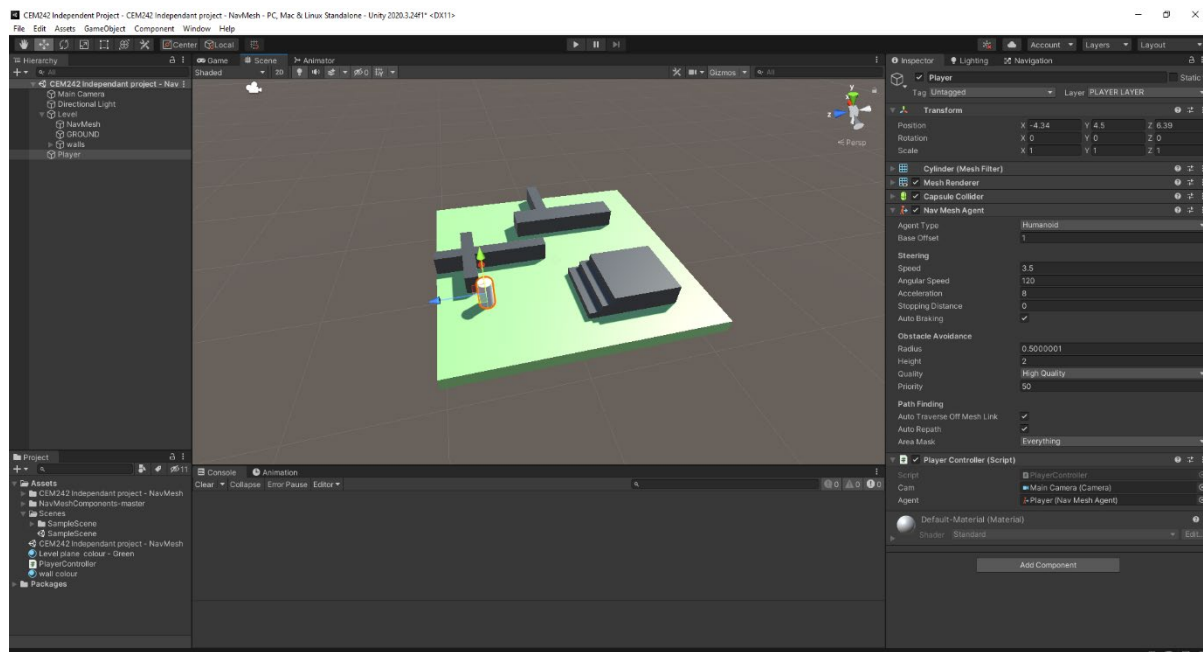


Figure 13 - added camera and agent into script

Upon loading the game, the script works and moves the agents to the location the user left clicks. A short illustration is shown below refer to fig 14.

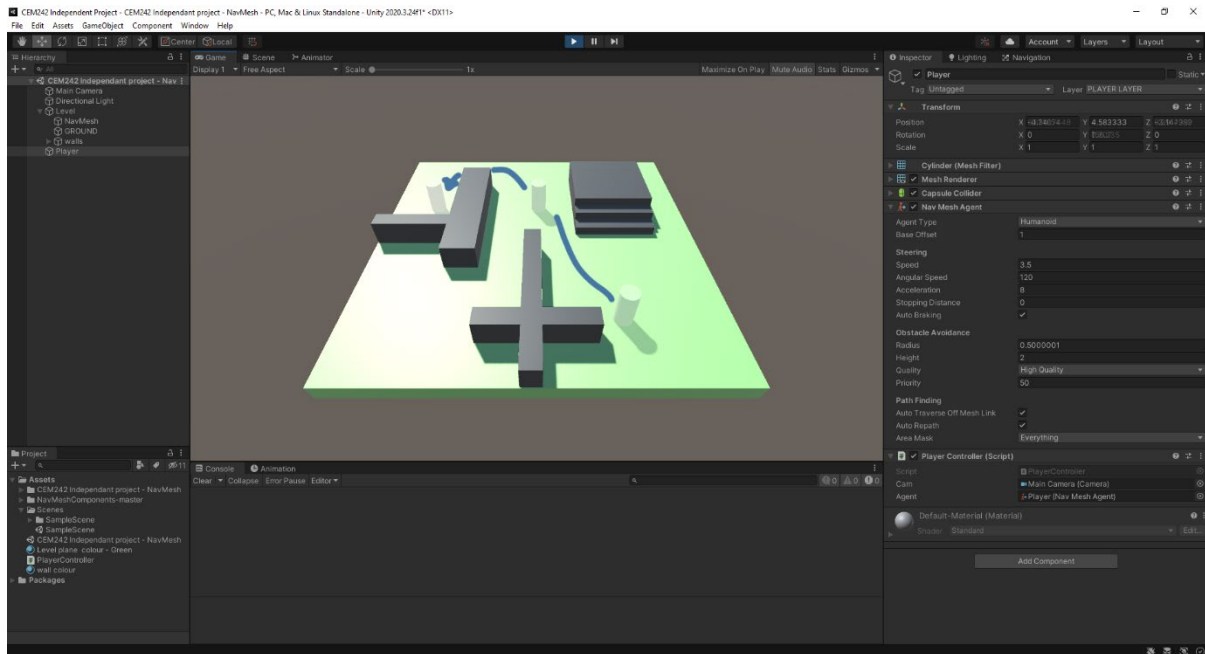


Figure 15- in game illustration of movement

Upon duplicating our set up agent can start to explore potential crowd simulations and see how agents react with other agents, in our build environment, a common issue is noticed was that they tended to get stuck upon tight corners. Refer to fig 15.

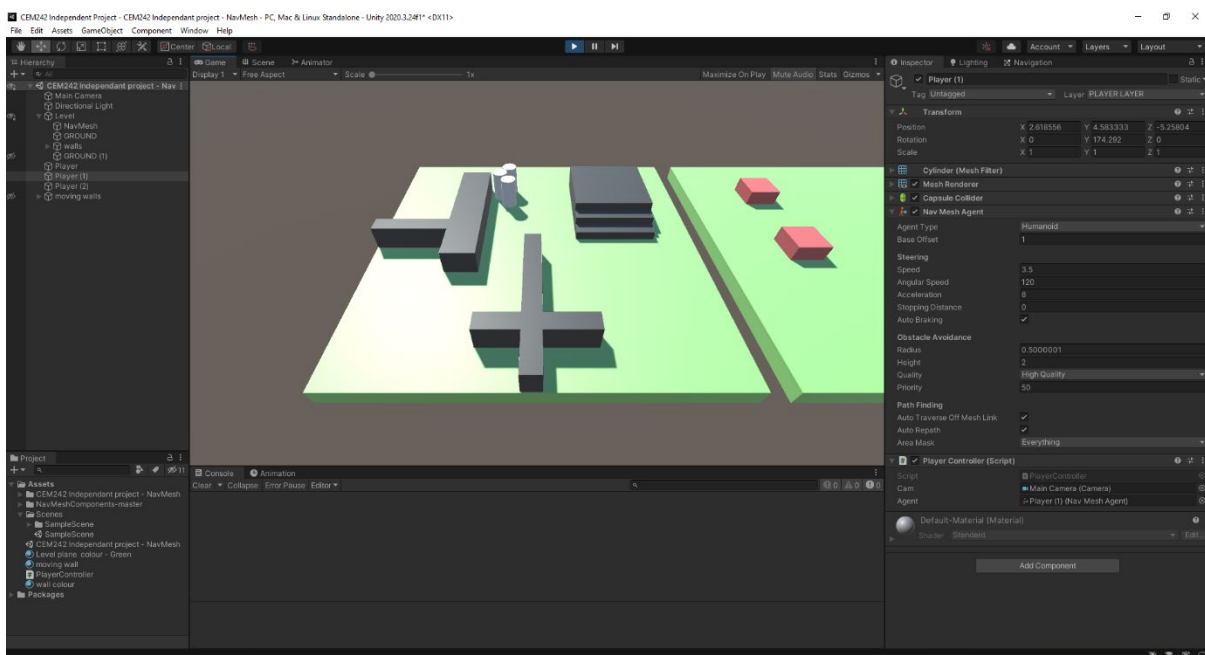


Figure 14 - Multiple agents added

### Activity 3 – NavMesh Obstacle.

The NavMesh Obstacle component allows you to describe moving obstacles that Nav Mesh Agents should avoid while navigating the world. (Unity Documentation, 2020) Here I have set up 3 addition cubes (identified in red) and applied the NavMesh Obstacle component to them. Ref to fig 16.

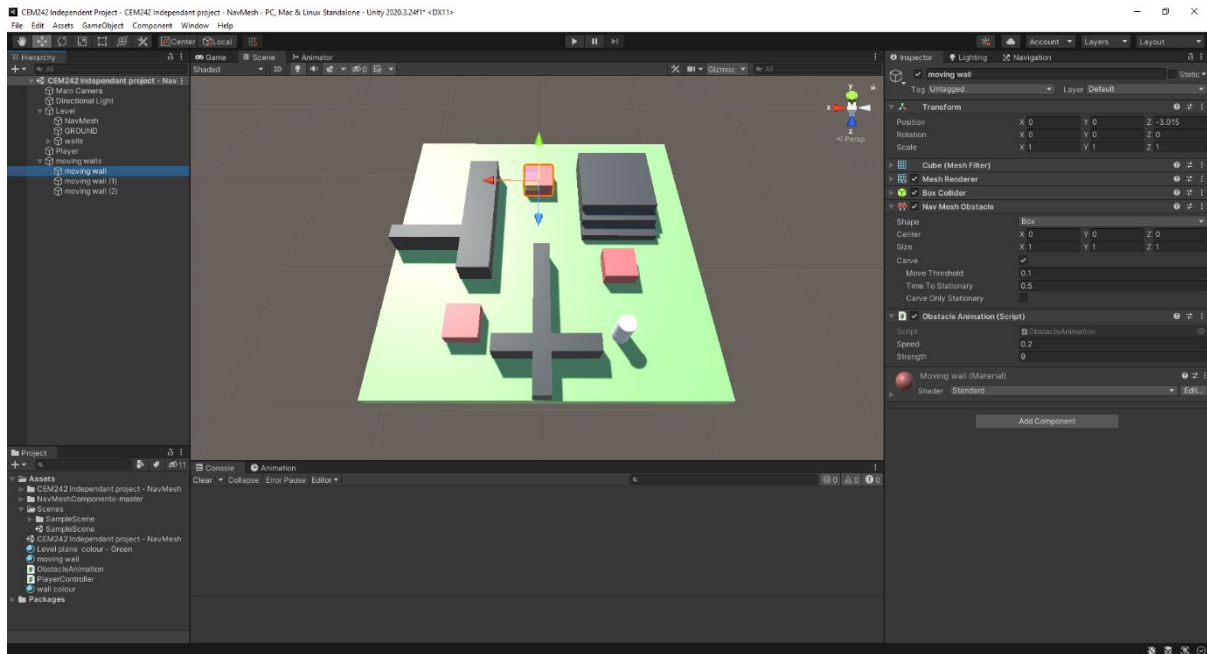


Figure 16 - addition ground and walls created

Upon running the prototype, the agent recognises the newly created cubes as obstacles and actively avoids them, cleverly pathing around them. Refer to fig 17.

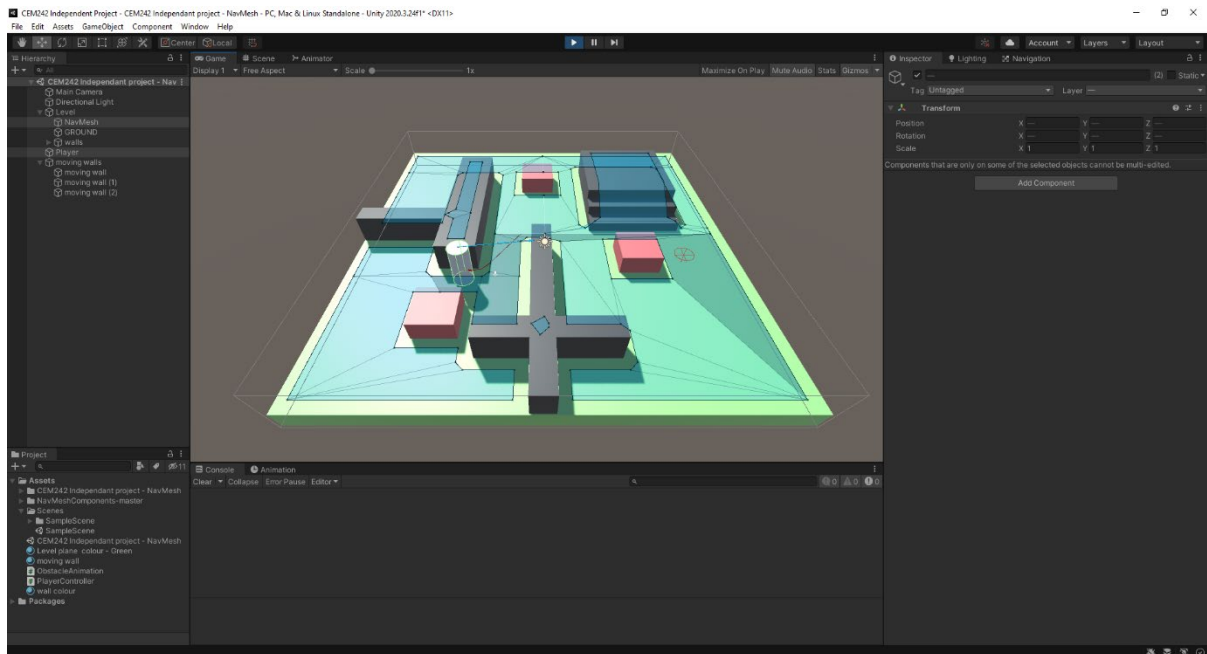


Figure 17 - agent actively avoiding and recalculating pathing to dynamic walls

To push this further I used a small script that moves the red obstacles to the left and right out of interest to see if the agent could actively adjust its movements. Refer to fig 18

Interesting the agent actively on the fly reacted to the moving walls and readjusted its path to always find the shortest route.

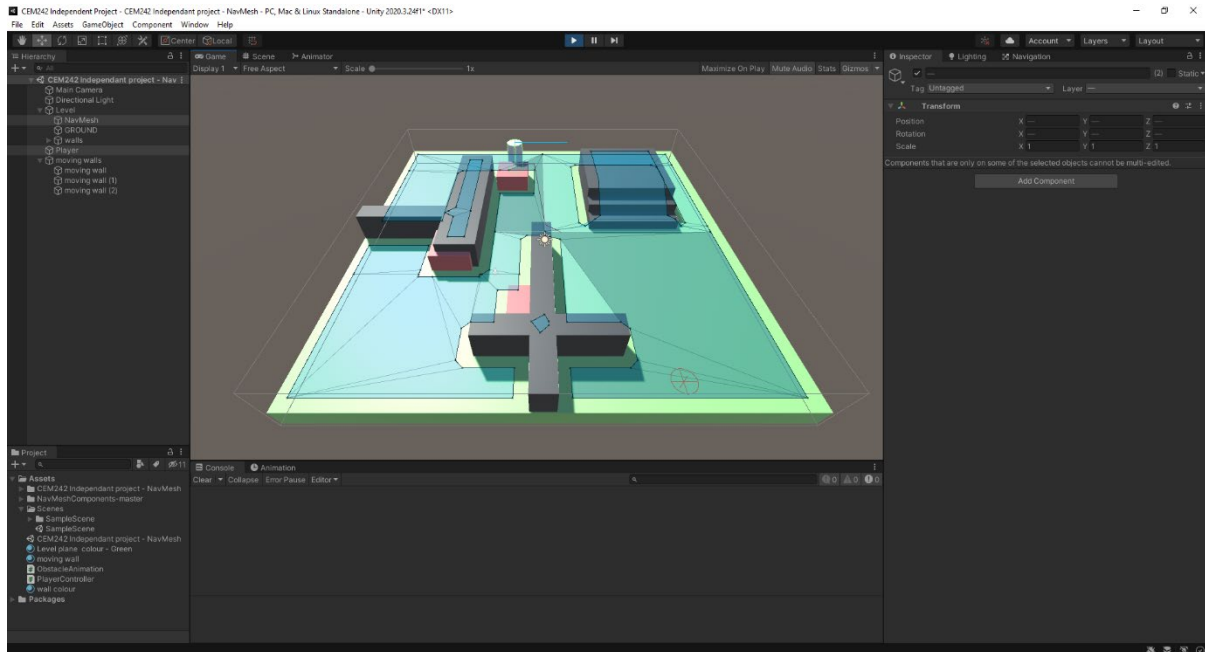


Figure 18 - running prototype, agent pathing adjusting

## Activity 4 – NavMesh Link.

NavMesh Link creates a navigable link between two locations that use NavMeshes. (Unity documentation, 2020) In the image below I have set up an addition ground with a separated NavMesh, currently the agent can't get to the other ground. Refer to fig 19.

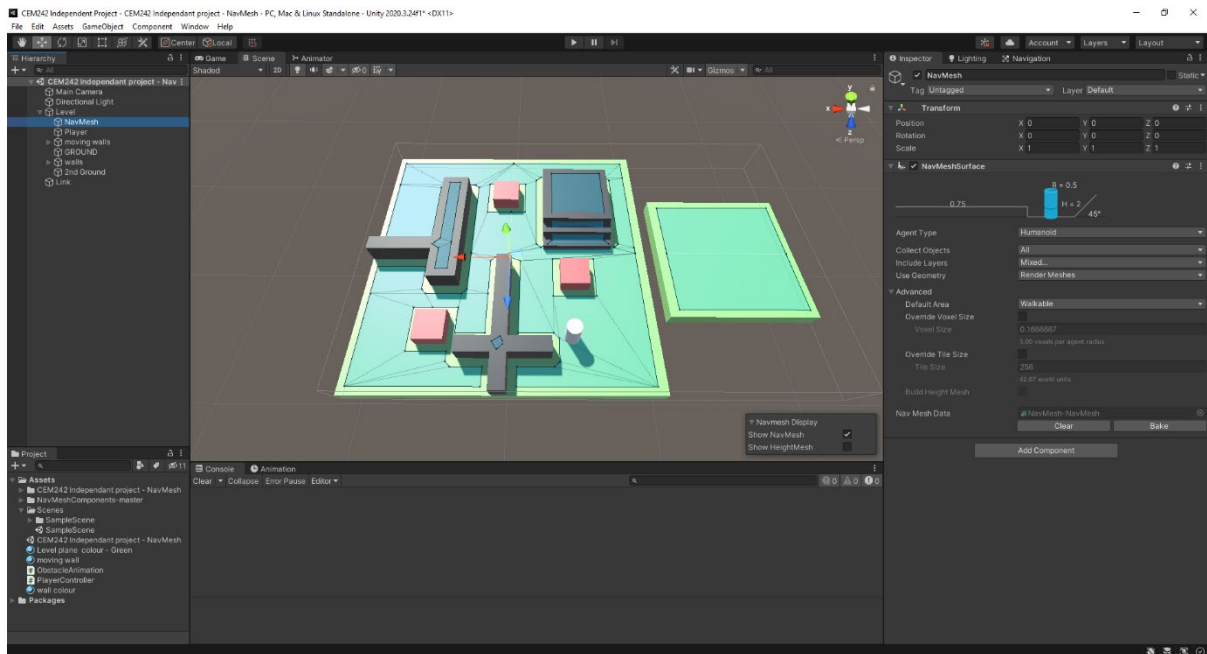


Figure 19 - addition ground added with NavMeshSurface applied

To allow the agent to travers between NavMeshes we must use the NavMeshLink component, I have created an empty call 'link' and added the NavMeshLink component. An on-screen indication appears in the form of an arching arrow. Refer to fig 20.

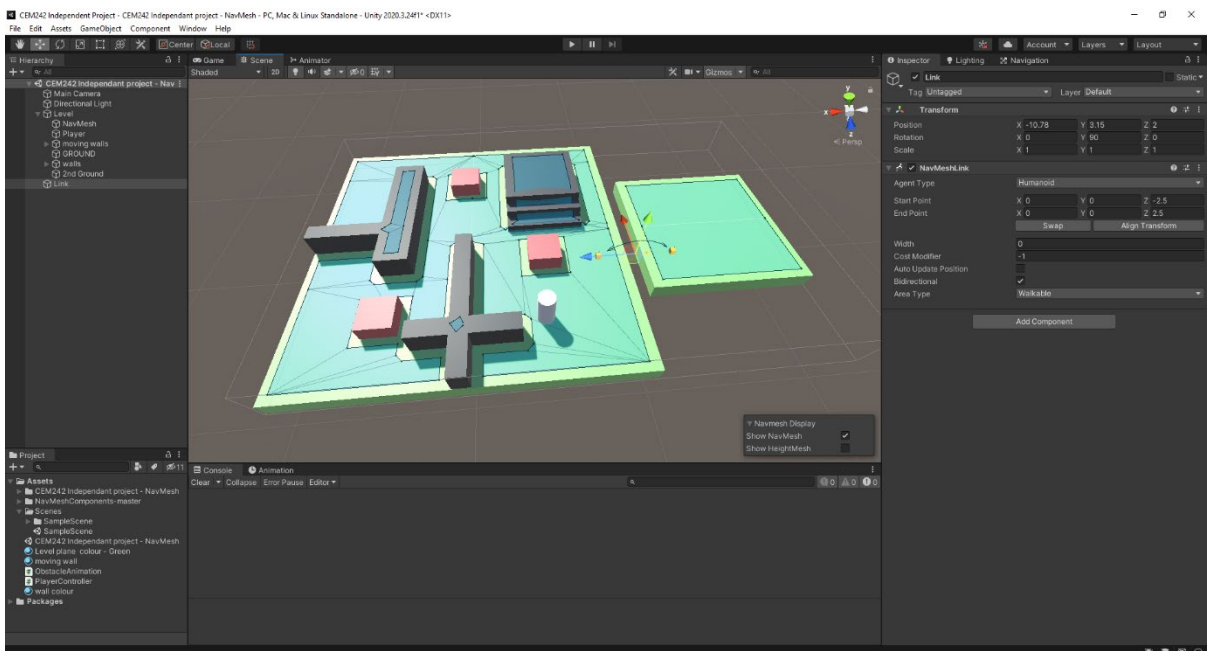


Figure 20 - NavMesh Link applied and an on-screen indication appears

Upon running the prototype our agent will approach the link and traverse over the link to the other NavMesh. Refer to fig 21. Also, within the prototype you will notice the agent speeds up when traversing the link this is to mimic a jumping motion.

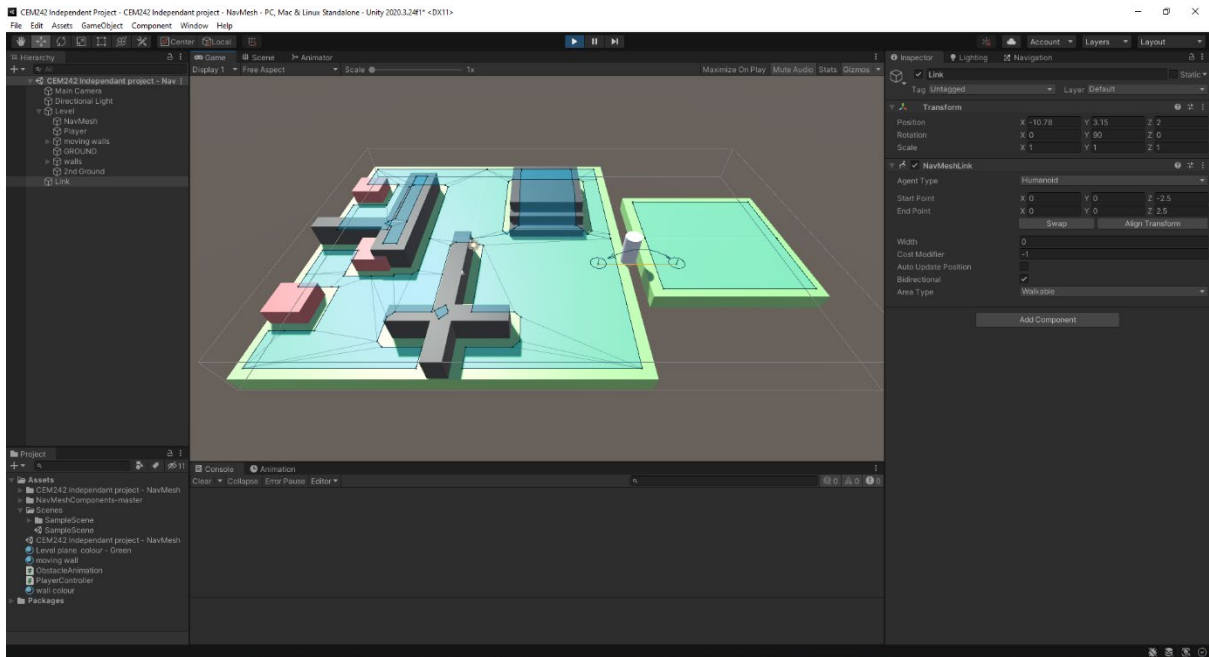


Figure 21 - in game traversing NavMeshes

The NavMeshLink component allows a wide range of customisability from controlling what agents are allowed to cross to the widths of the crossing point and if the crossing is one directional. Refer to fig 2.

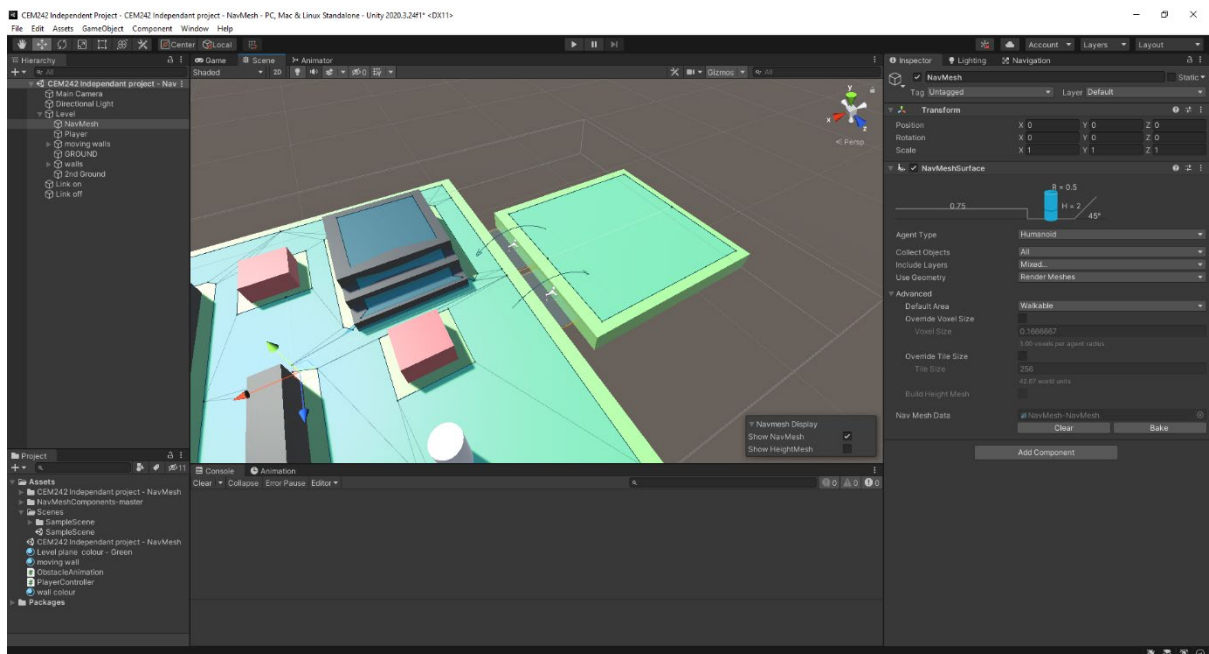


Figure 22 - widened link area and made the direction on direction

## REFLECTION, PROTOTYPE AND FUTURE USES.

The NavMesh is an extremely useful and easy to use tool that allows for users to create traversable areas very quickly, using either the built-in unity tools or external models.

The 4 key components add a verity of tools and customisations to create endless worlds. All 4 components were easy, intuitive, and offered an element of play with the customisation they offered.

NavMesh works well on flat surfaces but can struggle on external imported models and on textured bumpy surfaces.

Some potential use cases for the tool could be for small or large crowd simulation through creating multiple agents that travers to a set location. The attached prototype investigates this at a smaller scale using the created world from the detailed activities. Set waypoints or objects could also be applied to the agents.

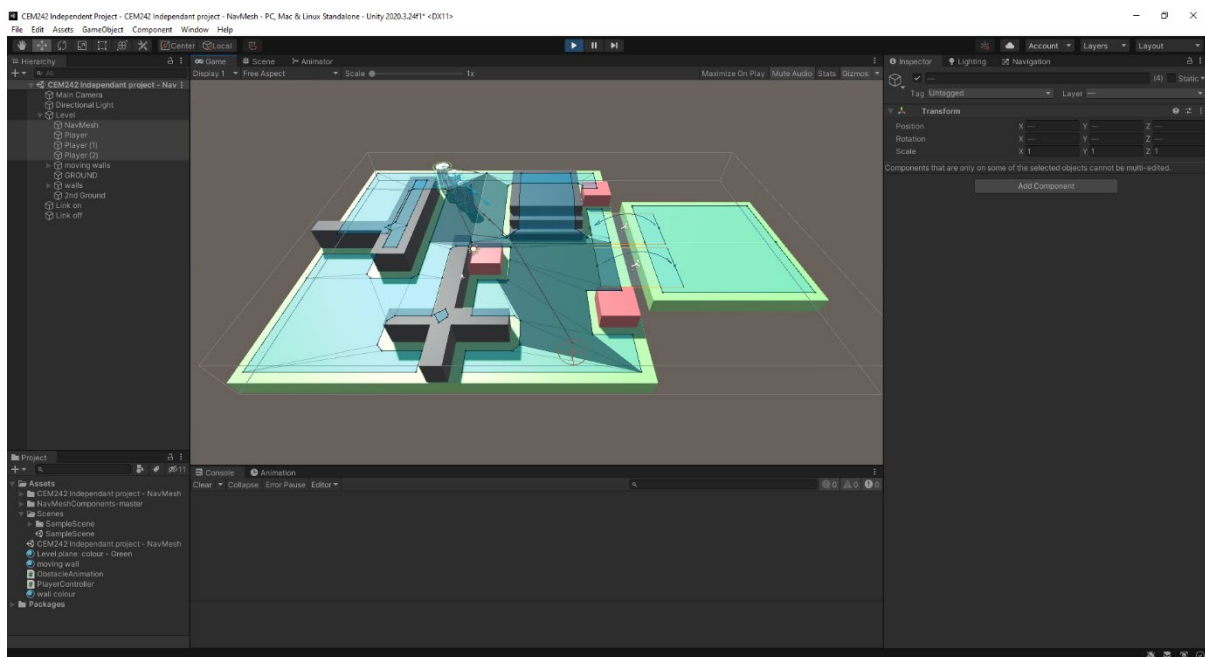


Figure 23 - Prototype crowd simulation

## **BIBLIOGRAPHY**

Unity documentation, 2020. *Unity - Manual: Navigation System in Unity*. [online] Docs.unity3d.com. Available at: <<https://docs.unity3d.com/Manual/nav-NavMeshAgent.html>> [Accessed 4 December 2021].

Unity documentation, 2020. *Unity - Manual: Nav Mesh Obstacle*. [online] Docs.unity3d.com. Available at: <<https://docs.unity3d.com/Manual/class-NavMeshObstacle.html>> [Accessed 15 December 2021].

## **REFERENCES**

Unity User Manual 2020.3 (LTS) (<https://docs.unity3d.com/Manual/index.html>)

NavMesh Baking by Unity Technologies (<https://learn.unity.com/tutorial/navmesh-baking>)

Working with NavMesh Agents by Unity Technologies (<https://learn.unity.com/tutorial/working-with-navmesh-agents>)

How to use Unity NavMesh Pathfinding! (Unity Tutorial) by code monkey (<https://youtu.be/atCOd4o7tG4>)

Unity AI (NavMesh) Part 01 by UIW 3D Animation and Game Design (<https://youtu.be/vlwVHfwvg-g>)

Unity AI (NavMesh) Part 02 by UIW 3D Animation and Game Design (<https://youtu.be/rbVf3I2BV6Y>)