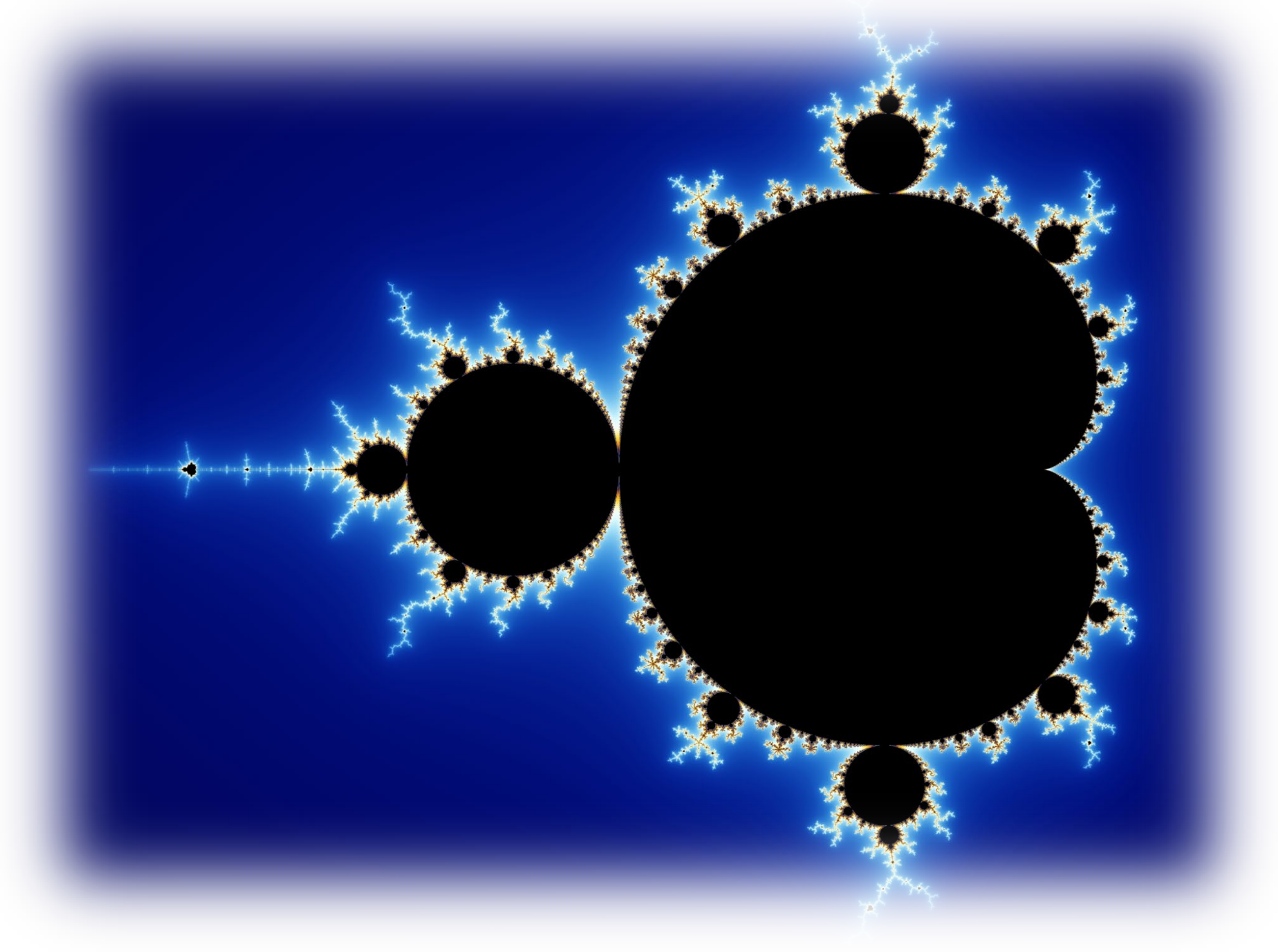


AI & Pathfinding

Ci410

What is AI?

The appearance of intelligence in Agents

How does this manifest itself?

- Finding the player
- Blocking an attack
- Picking the correct attack
- Running away if fight can’t be won

Breaking it down

REMINDER: AI is there to entertain

Think of it as Artificial Stupidity

Needs to give a good game experience

Provides challenge whilst giving player win opportunities

With computers Intelligence is overrated and best faked

Come find me

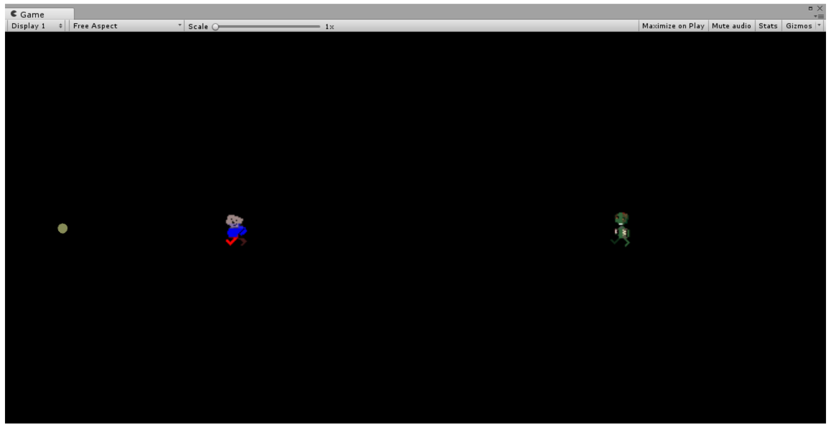
Simple pathfinding

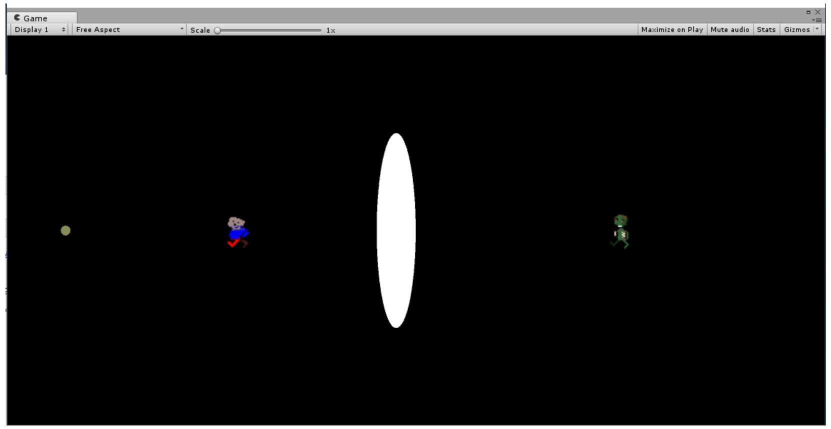
Vpath=Vdest-Vcurrent

Distance = |Vpath|

One optimal path

How about now?





Other consideration

Appearing more intelligent

- Facing your opponent
- Running away, with haste
- Slowing down when getting close

Anticipation

- Aiming where player will be, rather than where they are

Others?

Sensing the real world

Positions

- Points in world space

Colliders

- Volumes / Areas in world space

Anticipating

- Extrapolation, path between next & future
- Interpolation, path between last and next
- Raycasting

Cheating

- Knowing vs. guessing velocity

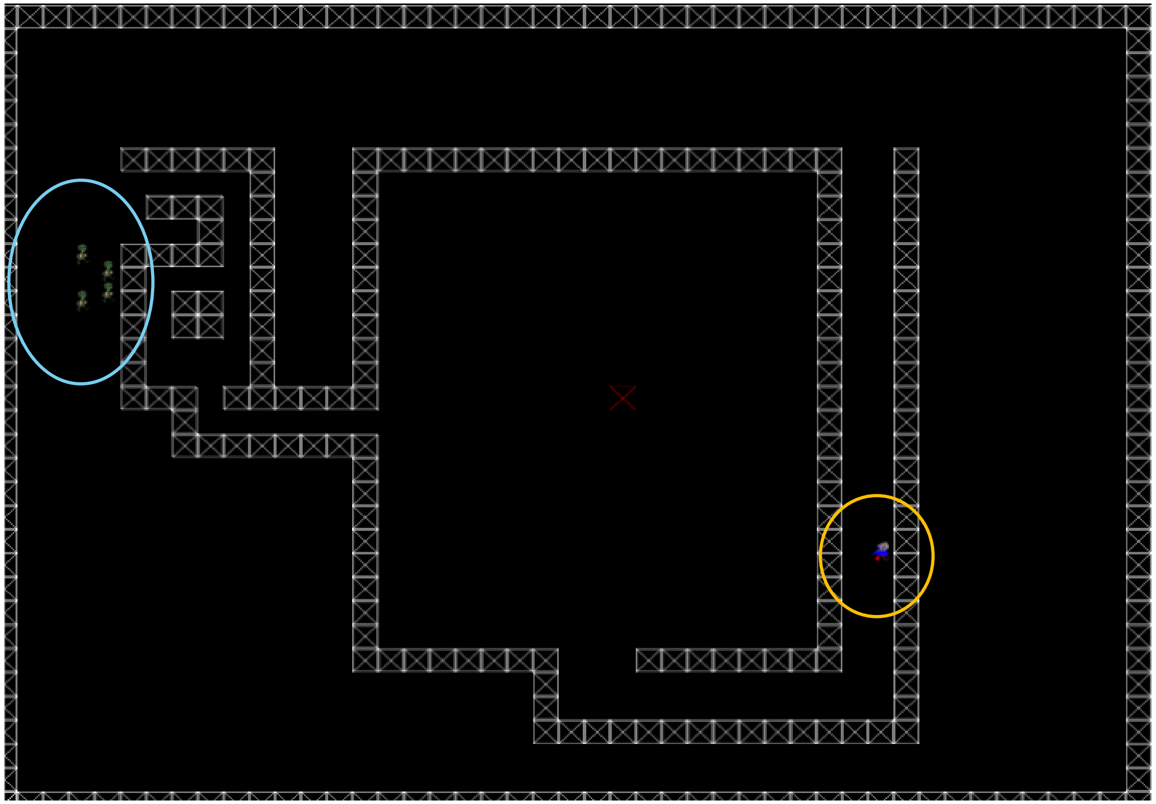
Pathfinding

A* (called “A Star”)

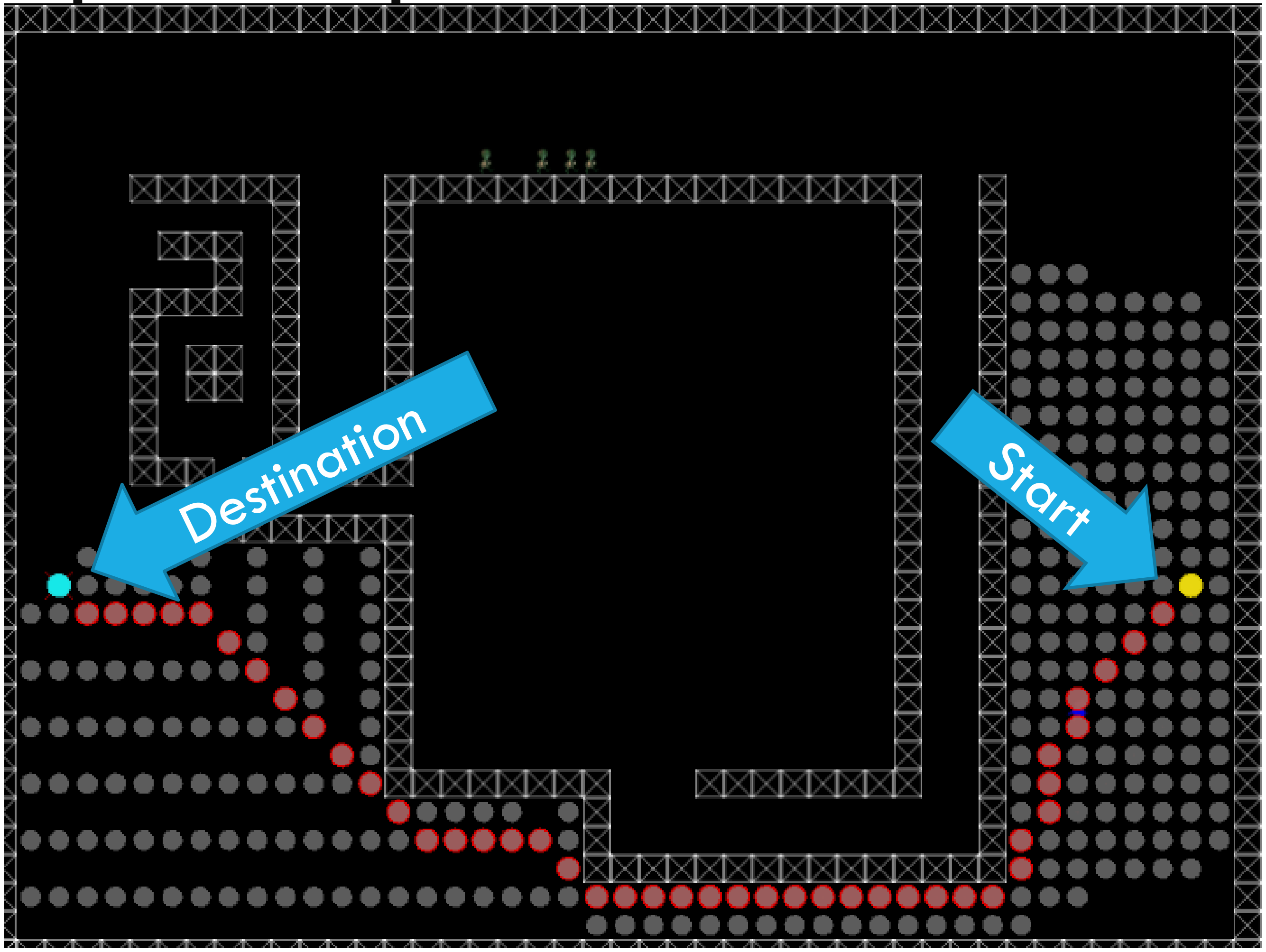
- A very common path finding algorithm
- Works by coming up with the least “cost” path between origin & destination
- Quite complex, you are not expected to remember it, source code sample on GitHub
- Requires knowledge of what is “navigable” i.e. can be walked on
- Can be computationally expensive

Zombie eat brains

http://modulo17.com/unity/astar/



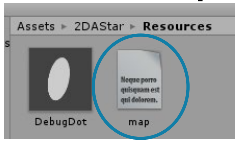
Evaluates possible paths



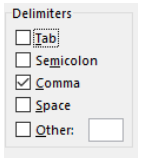
Change level layout

Excel as level editor

Load map.csv in into Excel

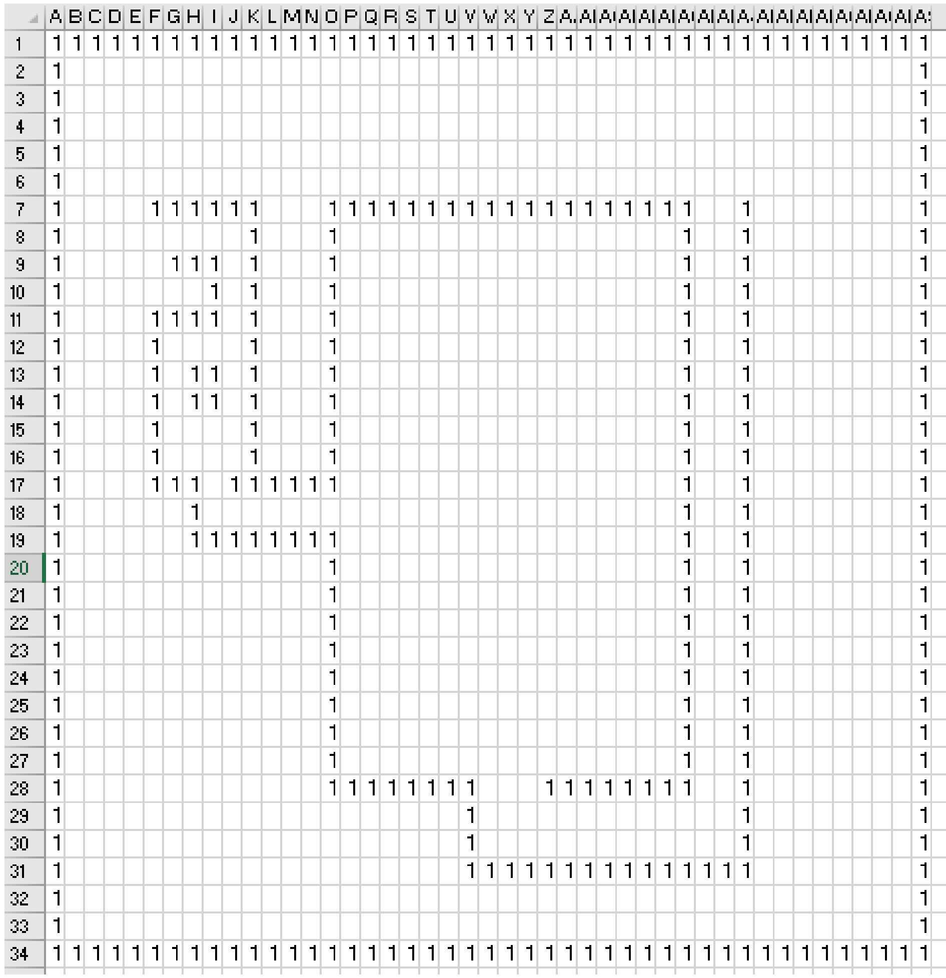


Make sure you set text import delimiter to ,


Delimiters
- [ ] Tab
- [ ] Semicolon
- [✓ Comma]
- [ ] Space
- [ ] Other:

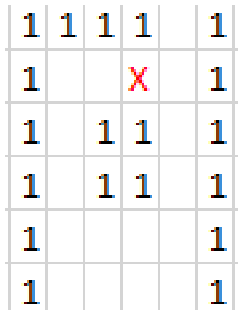
Edit level & make notes on A* behaviour, include level pictures

1 = wall



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A.	AI	AI	AI	AI	AI	AI	AI	AI	A.	AI	AI	AI	AI	AI	AI	AI	AI	A:
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1																																												1
3	1																																												1
4	1																																												1
5	1																																												1
6	1																																												1
7	1					1	1	1	1	1	1				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1									1
8	1										1				1																		1			1									1
9	1						1	1	1		1				1																		1			1									1
10	1								1		1				1																		1			1									1
11	1					1	1	1	1		1				1																		1			1									1
12	1					1					1				1																		1			1									1
13	1					1		1	1		1				1																		1			1									1
14	1					1		1	1		1				1																		1			1									1
15	1					1					1				1																		1			1									1
16	1					1					1				1																		1			1									1
17	1					1	1	1		1	1	1	1	1	1																		1			1									1
18	1							1																									1			1									1
19	1							1	1	1	1	1	1	1	1																		1			1									1
20	1														1																		1			1									1
21	1														1																		1			1									1
22	1														1																		1			1									1
23	1														1																		1			1									1
24	1														1																		1			1									1
25	1														1																		1			1									1
26	1														1																		1			1									1
27	1														1																		1			1									1
28	1														1	1	1	1	1	1	1	1				1	1	1	1	1	1	1	1			1									1
29	1																					1														1									1
30	1																					1														1									1
31	1																					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1									1
32	1																																												1
33	1																																												1
34	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Navigating with A*



1	1	1	1		1
1			X		1
1		1	1		1
1		1	1		1
1					1
1					1

A* works well with an array

- It relies on rapidly knowing where it can step next
- It uses this to calculate routes from its current position to the destination
- Blocked paths are "closed", they no longer form part of the search
- It picks open routes which get it closer to the target at the least cost
- http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

Navigating with Unity

NavMesh is a map of static walkable areas, it is baked in

NavAgent uses A* on a NavMesh for path finding around static objects

NavAgent uses RayCast to avoid other NavAgents & NavObstacles real-time

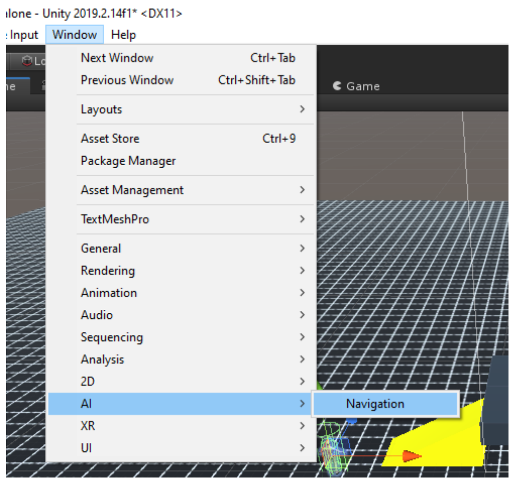
Some good resources here

- https://docs.unity3d.com/Manual/nav-NavigationSystem.html

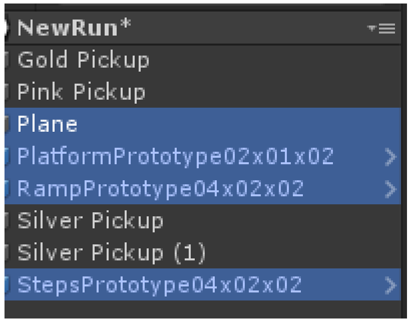
Before we can use navigation

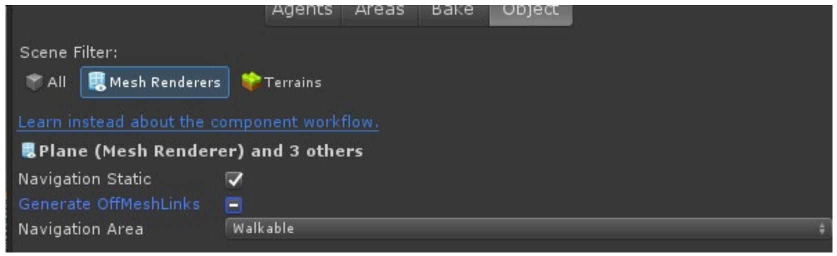
We need to "Bake a NavMesh"

This is done from the Navigation Window



Select the objects in the scene to include & make them walkable





Plane (Mesh Renderer) and 3 others

Navigation Static ✓

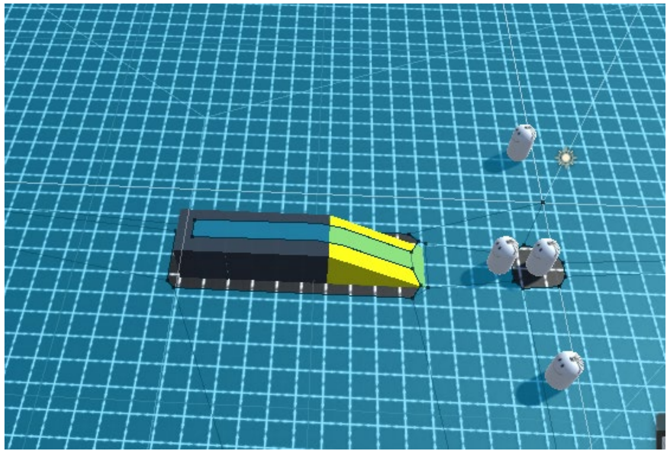
Generate OffMeshLinks −

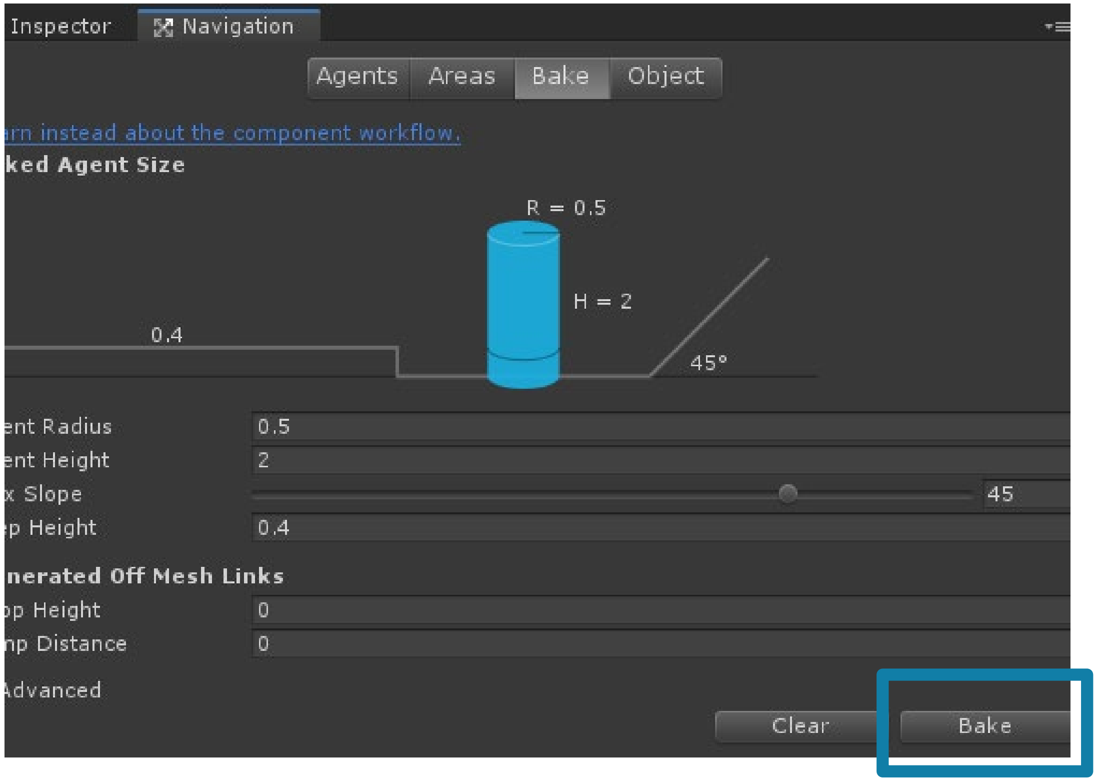
Navigation Area: Walkable

They can then be baked

On the Bake Tab

Once baked blue is walkable





arn instead about the component workflow.

ked Agent Size

ent Radius 0.5

ent Height 2

x Slope 45

p Height 0.4

nerated Off Mesh Links

op Height 0

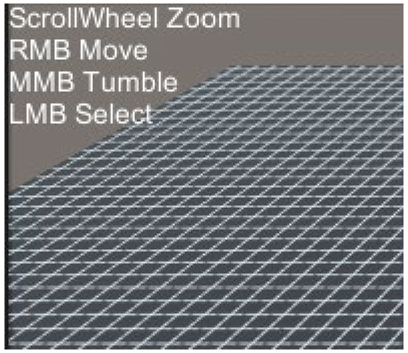
np Distance 0

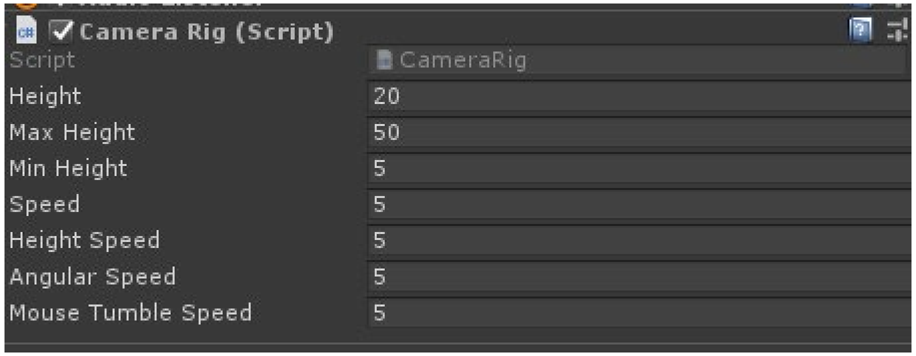
Advanced

Camera Rig (CameraRig.cs)

Top down with some push movement

- Inspect the code in the lab, it will allow the scene to be viewed
- You can attach the script to an existing camera or delete the existing and use the script to make one
- It has lots of defaults you can use, controls on screen

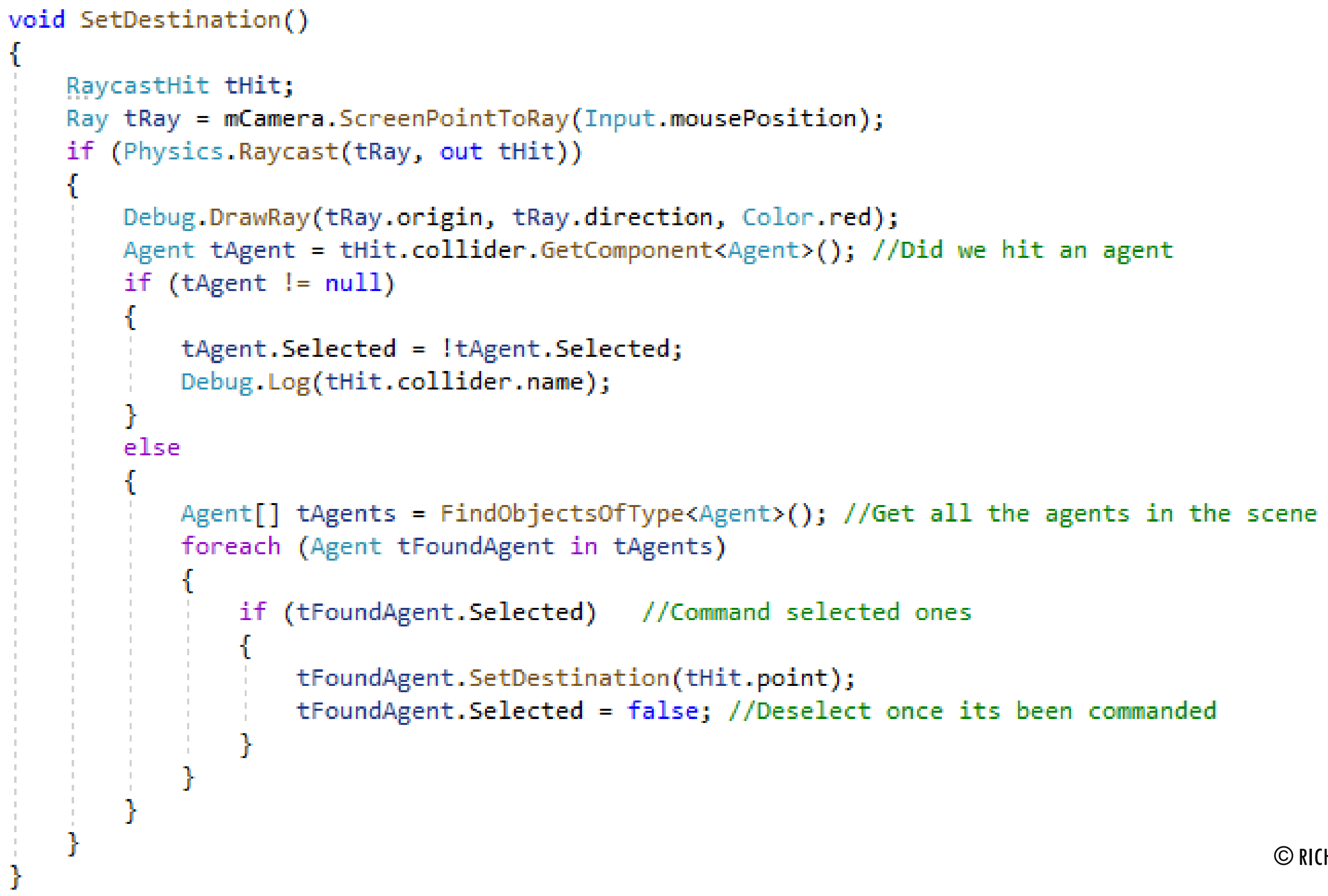

ScrollWheel Zoom
RMB Move
MMB Tumble
LMB Select


Camera Rig (Script)

Script	CameraRig
Height	20
Max Height	50
Min Height	5
Speed	5
Height Speed	5
Angular Speed	5
Mouse Tumble Speed	5

The Camera also directs the Agents

Using a Raycast to see what's under the mouse, if its an agent toggle selection, if not then send all the selected agents there

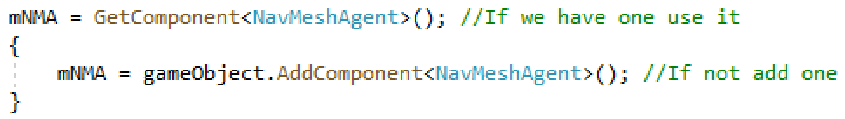


```
void SetDestination()
{
    RaycastHit tHit;
    Ray tRay = mCamera.ScreenPointToRay(Input.mousePosition);
    if (Physics.Raycast(tRay, out tHit))
    {
        Debug.DrawRay(tRay.origin, tRay.direction, Color.red);
        Agent tAgent = tHit.collider.GetComponent<Agent>(); //Did we hit an agent
        if (tAgent != null)
        {
            tAgent.Selected = !tAgent.Selected;
            Debug.Log(tHit.collider.name);
        }
        else
        {
            Agent[] tAgents = FindObjectsOfType<Agent>(); //Get all the agents in the scene
            foreach (Agent tFoundAgent in tAgents)
            {
                if (tFoundAgent.Selected)   //Command selected ones
                {
                    tFoundAgent.SetDestination(tHit.point);
                    tFoundAgent.Selected = false; //Deselect once its been commanded
                }
            }
        }
    }
}
```

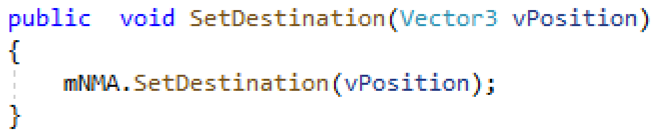
The Agents (Agent.cs)

They will navigate on the NavMesh

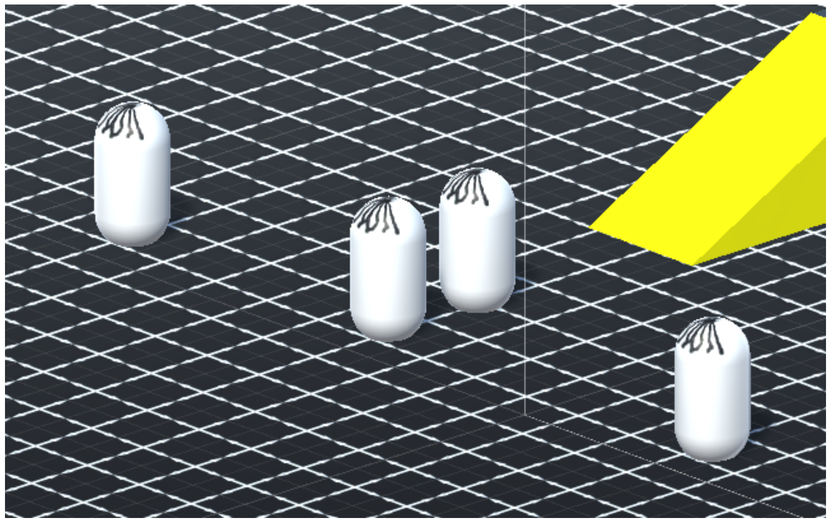
- They have a selected flag to mark them as selected or not
- Use the NavMeshAgent to get to destination



```
mNMA = GetComponent<NavMeshAgent>(); //If we have one use it
{
    mNMA = gameObject.AddComponent<NavMeshAgent>(); //If not add one
}
```

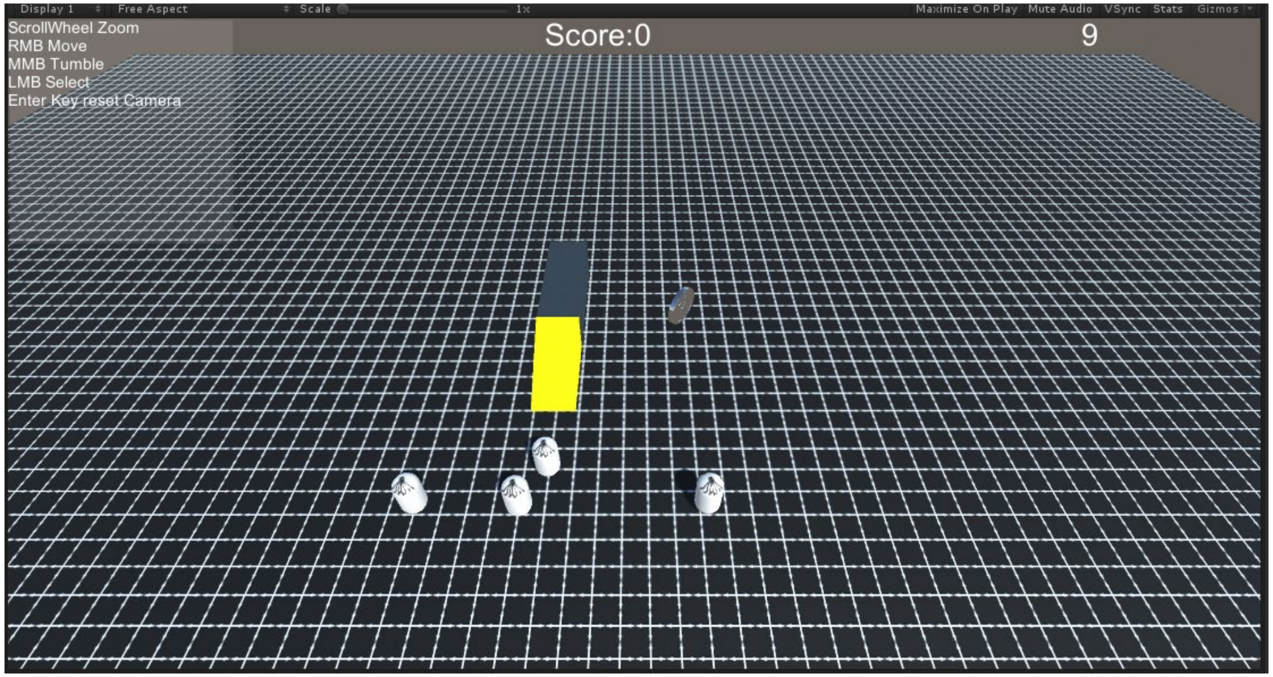


```
public  void SetDestination(Vector3 vPosition)
{
    mNMA.SetDestination(vPosition);
}
```



The GM (GM.cs)

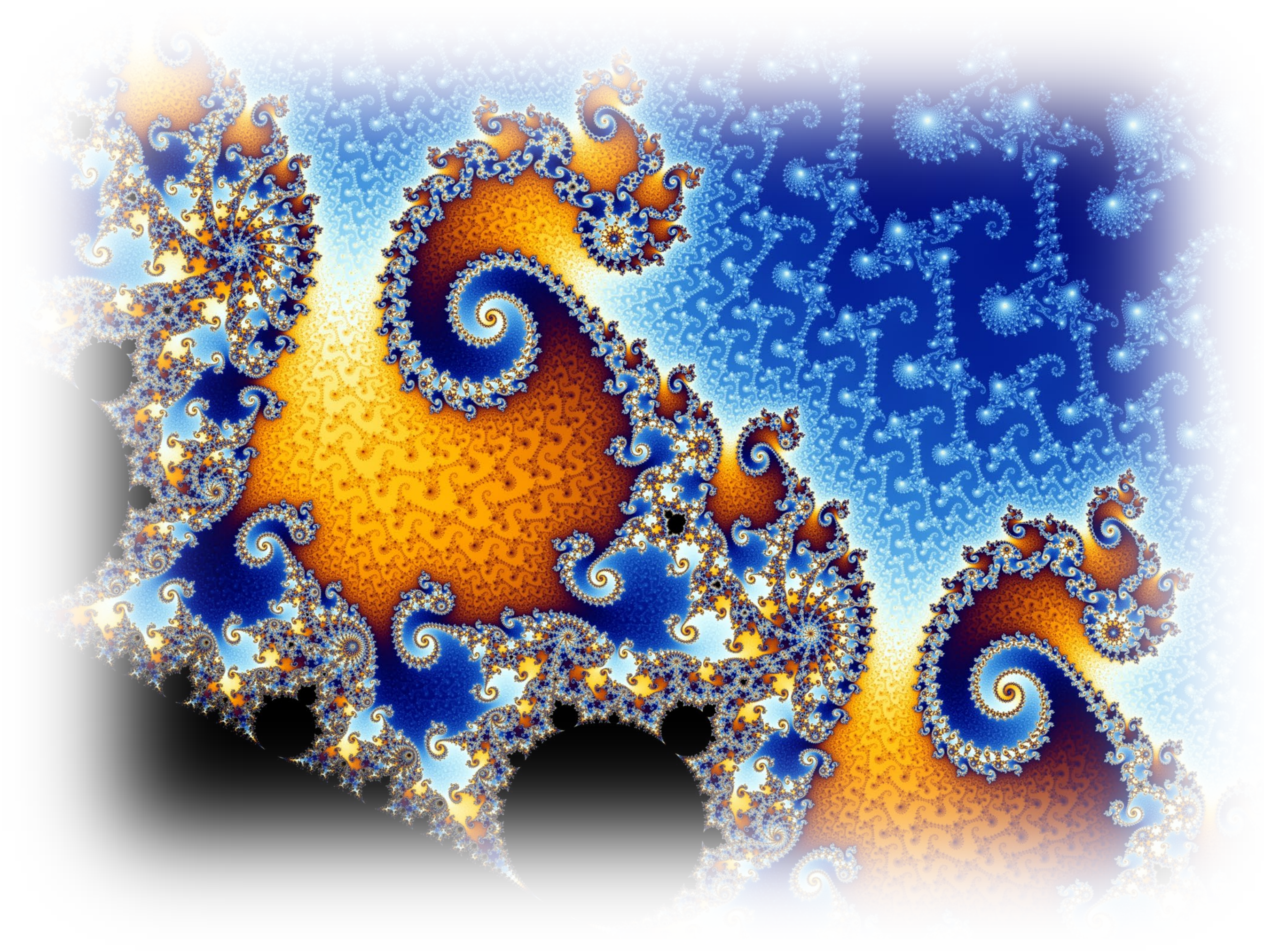
Pretty much the same as last week, handles score & time



ScrollWheel Zoom
RMB Move
MMB Tumble
LMB Select
Enter Key reset Camera

Score:0

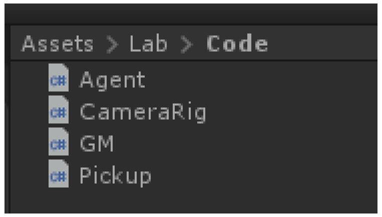
9



WORKSHOP

Workshop, make a AI based game

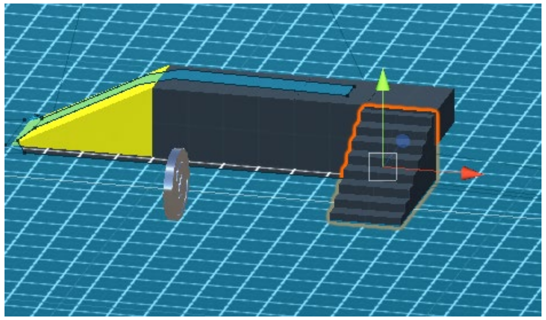
1. Review the scripts
   - Agent.cs
   - CameraRig.cs
   - Pickup.cs
   - GM.cs

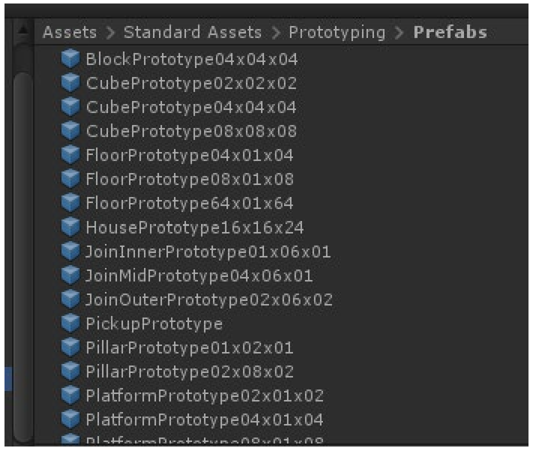


Assets > Lab > **Code**

- Agent
- CameraRig
- GM
- Pickup

2. Try the supplied Scene and see the limitations of the NavMesh
3. Add more Geometry to the scene from standard assets
4. Rebake



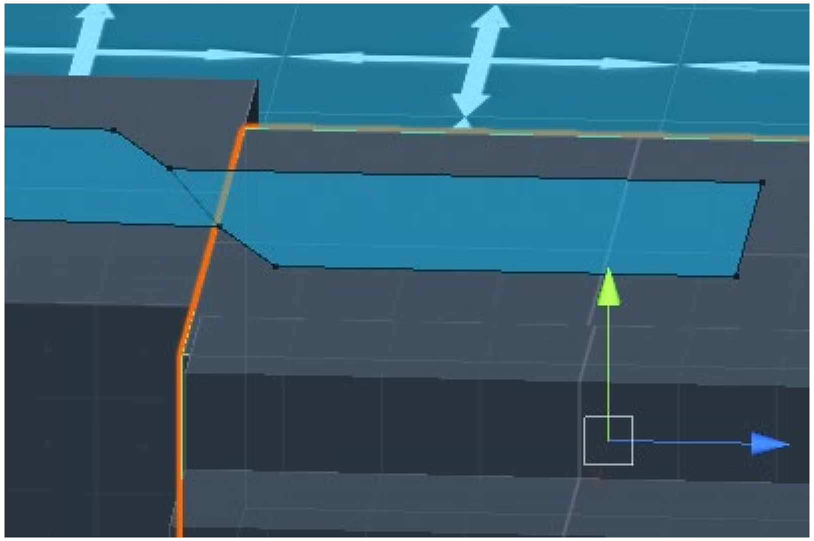


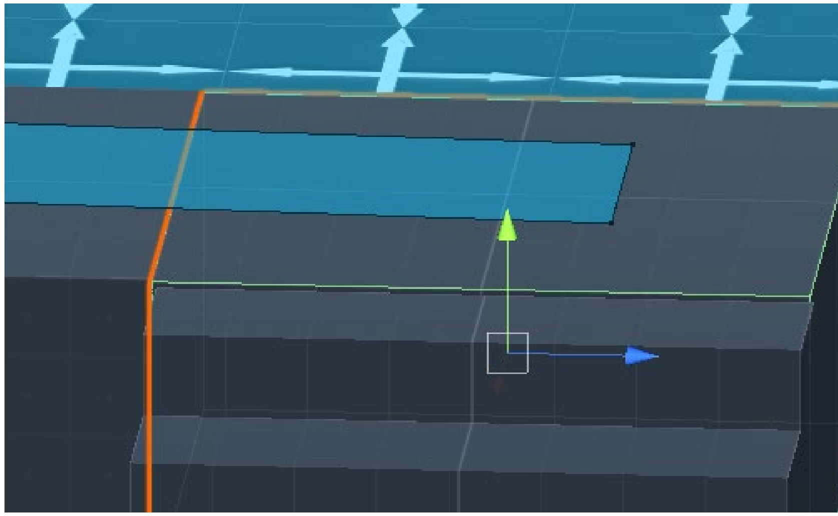
Assets > Standard Assets > Prototyping > **Prefabs**

- BlockPrototype04x04x04
- CubePrototype02x02x02
- CubePrototype04x04x04
- CubePrototype08x08x08
- FloorPrototype04x01x04
- FloorPrototype08x01x08
- FloorPrototype64x01x64
- HousePrototype16x16x24
- JoinInnerPrototype01x06x01
- JoinMidPrototype04x06x01
- JoinOuterPrototype02x06x02
- PickupPrototype
- PillarPrototype01x02x01
- PillarPrototype02x08x02
- PlatformPrototype02x01x02
- PlatformPrototype04x01x04

Fix any issues with Geometry

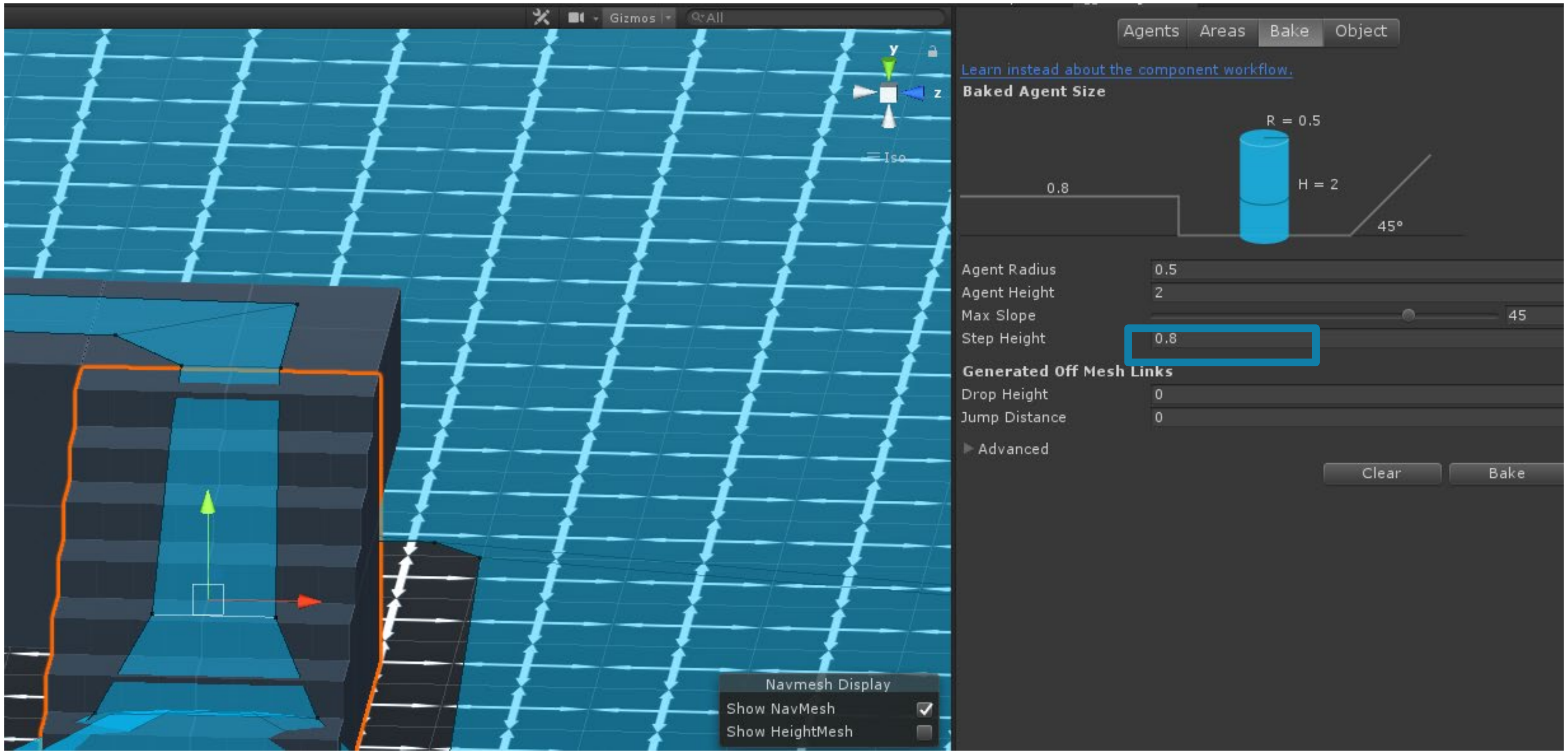
Such as gaps





Steps too high?

Change Step Size



Learn instead about the component workflow.

Baked Agent Size

Agent Radius	0.5
Agent Height	2
Max Slope	45
Step Height	0.8

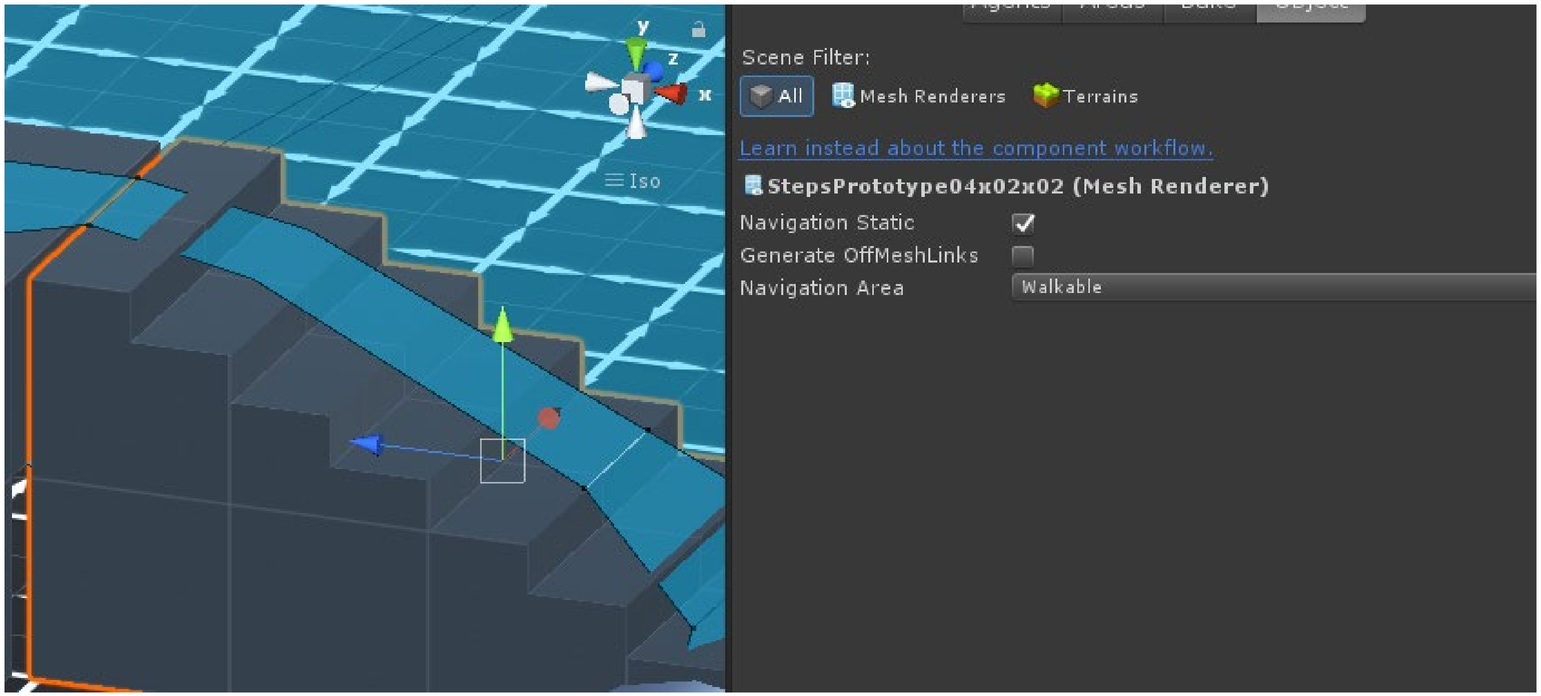
Generated Off Mesh Links

Drop Height	0
Jump Distance	0

▶ Advanced

Show NavMesh ✓
Show HeightMesh

When baking make sure items are marked as Navigation static



StepsPrototype04x02x02 (Mesh Renderer)

Navigation Static ✓

Generate OffMeshLinks

Navigation Area: Walkable

Make a small game

1. Build a maze (using Standard asset proto prefabs)
2. Ensure the NavMesh works
3. Hide pickups in the maze
4. Send the Agents in to find them

➢ They will get stuck, how would you know this? What could you do about it?