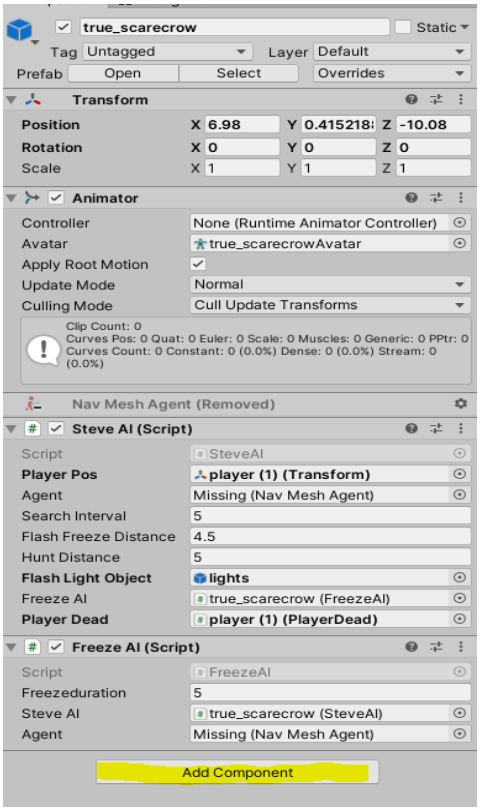
3D Game development project

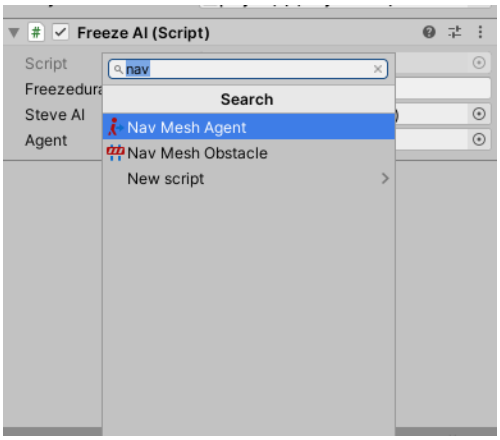
Geschreven door: Kobe bogaerts

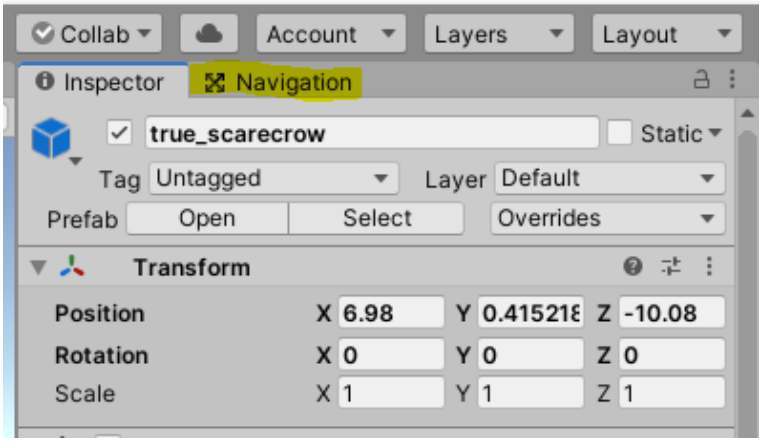
Voor mijn contributie aan het project heb ik een simpele AI gemaakt die de speler achtervolgt en probeert te doden. Als er met een zaklamp op de ai word geschenen zal deze stoppen met het uitvoeren van de acties waar deze mee bezig is op dat moment. Voor het achtervolgen van de spelen, dit is gebeurd met path finding die in Unity is ingebouwd. Dit is een zeer simpel te maken door middel van nav mesh. Er zijn 2 componenten om path finding te late werken met nav mesh: een agent en een baked path. Een agent is een component die zal beslissen waar de AI naar toe zal gaan, met welke snelheid en nog andere eigenschappen zoals welke hoeken deze mag beklimmen. Het 2de component is een backed path dit zijn de gebieden die de agent mag betreden en welke niet, zodat de agent kan bereken hoe hij naar de bestemming kan gaan.

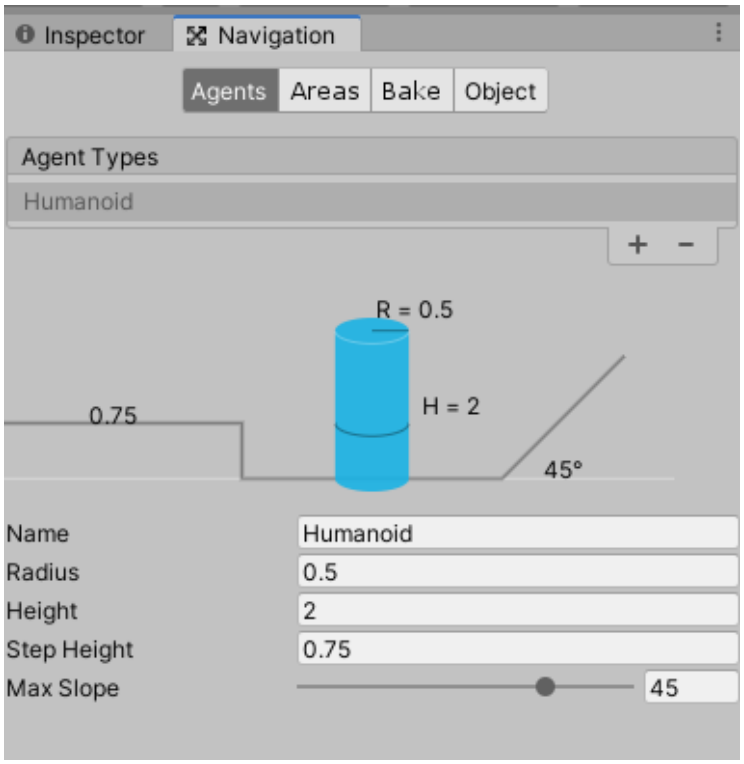
Editor:



Om nav mesh te gebruiken in je project zal je eerst een agent component moeten toevoegen aan een game-component.



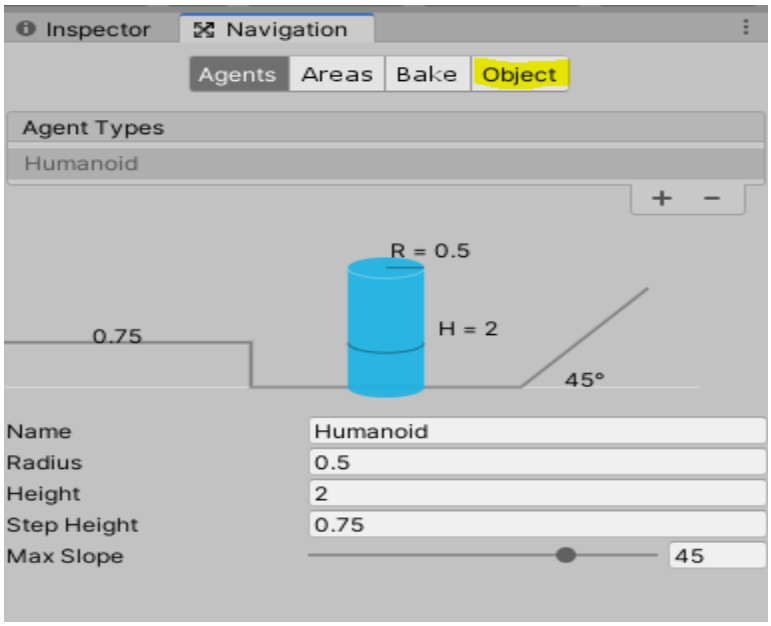


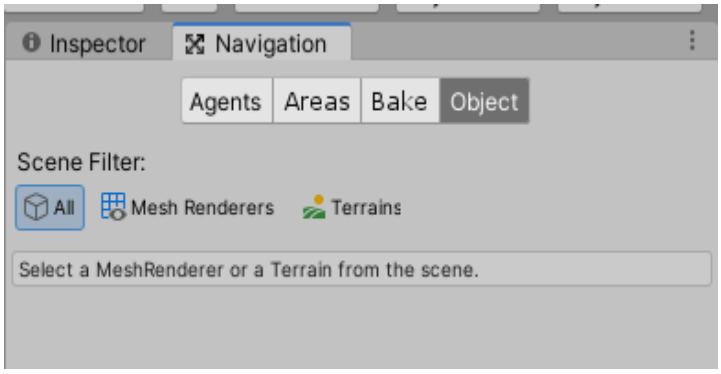


Daarna ga je naar de tab navigatie die naast de inspector tab staat. Hier vind je alle opties om je nav mesh te configureren.

In de agent tab kan je de grote en de hoogte dat de agent kan beklimmen.

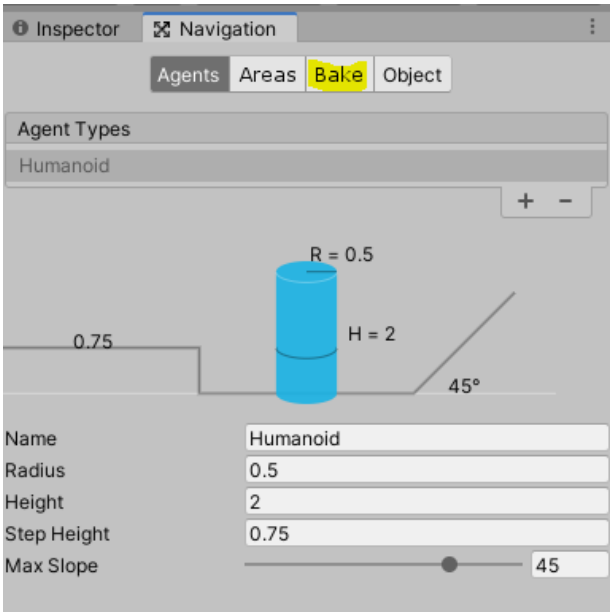
Je kan ook meerde types van agents maken bv, een oger in een game is groter en zou in bepaalde gebieden niet geraken. Deze types kan je dan aan je agent meegeven zodat er rekening mee kan gehouden worden in de berekeningen van de path finding.

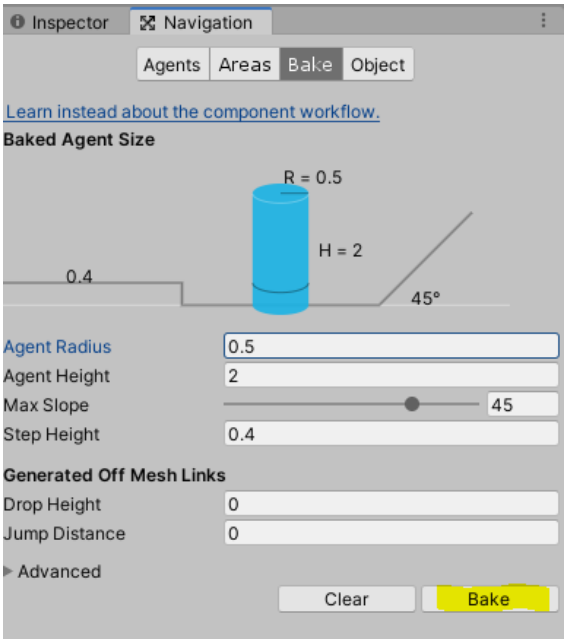


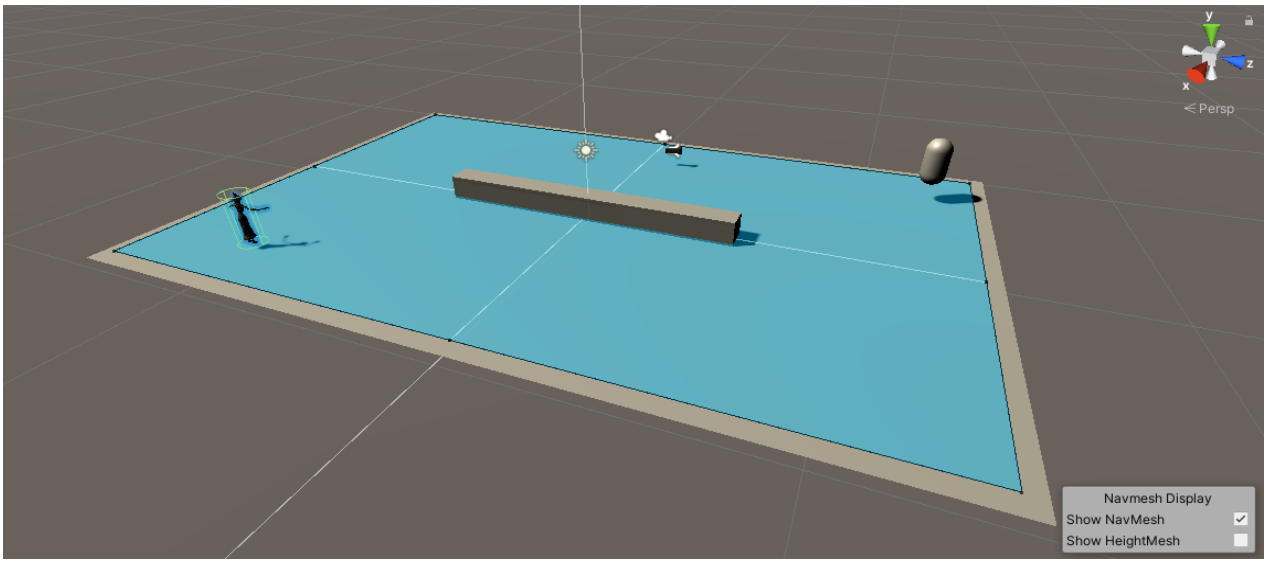


In de object tab kan je filters selecteren om bepaalde types als wandelbaar te maken en sommige niet. Ook zulle deze filters bepalen welke de agent moet ontwijken.

Als je als laatste zal je naar de bake tab moeten gaan. Het baken van een path is het statisch aanmaken van stukken waar de agent kan lopen en welke niet. Dit is nodig omdat anders de agent niet weet waar hij wel en niet mag komen. Als je na het baken van je mesh naar de Unity editor in scene kijkt (met de navigatie tab open). Kun je in het blauw de wandelbare delen zijn.







Dit is groten deels hoe je een nav mesh moet maken. Maar de agent moet nu nog via code weten naar welk punt in de wereld die moet navigeren. Dit is het 2de deel van een simplistische AI.

Code:

Voor de agent te laten bewegen naar een positie zal je als eerste een publieke referentie moeten maken van een NavMeshAgent, deze moet je dan assigne in de editor. Daarna kan je met agent.SetDestination(Vector3) de x,y,z positie zetten naar waar de agent zich naartoe moet verplaatsen.

```
public Transform playerPos;
public NavMeshAgent agent;
public float searchInterval = 5f;
public float flashFreezeDistance = 4.5f;
public float huntDistance = 5f;
public GameObject flashLightObject;
public FreezeAI freezeAI;
public PlayerDead playerDead;

float timer = 0.0f;
0 references
void Update()
{
    if (timer > searchInterval)
    {
        agent.SetDestination(playerPos.position);
        timer = 0;
    }
    timer += Time.deltaTime;
```

In het voorbeeld links is er een timer die om de 5 seconde de destination zal zetten naar de player postitie. Dit is handig als je wilt dat je agent de player achtervolgd.

In de code hier onder is wat uitbreiding op de AI en de code er boven, maar is eigenlijk zeer simpel. Als een bepaald gameobject de player of een andere te dicht bij de AI komt zal er een actie gebeuren, de afstanden kunnen variabel worden aangepast. In onze game kan je met een zaklantaren op de AI schijnen zodat deze stop met bewegen dus dat is de freeze methode. De freese methode zal het script uitzetten en na een tijd terug aan zodat het verder kan werken. Daarnaast is er ook een kill methode die word aan geroepen als de AI zo goed als in de player staat en in onze game ben je dan dood.

```
if(Vector3.Distance(transform.position, playerPos.position) < huntDistance)
{
    agent.SetDestination(playerPos.position);
}

//check if light is on
if (flashLightObject.activeSelf)
{
    if (Vector3.Distance(transform.position, flashLightObject.transform.position) < flashFreezeDistance)
    {
        print("Steve freeze");
        freezeAI.Freeze();
    }
}

if(Vector3.Distance(transform.position, playerPos.position) < 1.0f)
{
    playerDead.Kill();
}
```

Conclusie

Het is zeer simpel om een achtervolging AI te maken met praktisch geen code. Dit is allemaal mogelijk dankzij de simplistische configuratie van navmesh in Unity. Met 1 component 2 lijnen code en een paar drukken op een knop, kan je al een gameobject zelf een pad laten vinden en je scene.

Contributie aan het project

Mijn contributie aan dit project is de AI van de enemy, Player movement en acties zoals de zaklamp met zijn ui, het samenvoegen van alle delen, menuhandeling, afbakening van het terrain, pickup system met icon en bugfixes. Meeste van deze delen waren niet zo moeilijk te implementeren of om te doen. De meest tijd bestedende delen waren vooral de AI en het samenvoegen van alle delen en deze met elkaar te laten werken.