Far Away

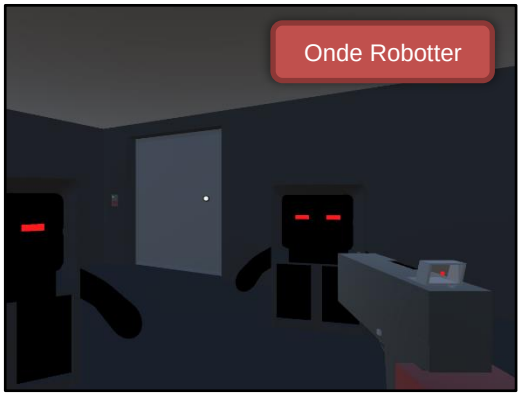
Martin Jørgen Pedersen
DATAMATIKERUDDANNELSEN
4 SEMESTER

Synopsis i 3D Spiludvikling
VALGFAGSPRØVEN 2021
VEJLEDER: MARTIN KIERKEGAARD

Indholdsfortegnelse

Far Away

Eksploderende, onde robotter i flokke og kraftige, altødelæggende våben, er man som spilleren med til at redde dagen i det her sjove, unikke univers. Jeg føler, at noget af det mest vigtige i ethvert spil er selve gameplayet, og det afspejles tydeligt i mit eget spil jeg har lavet i forbindelse med dette projekt.



Indledning

Denne synopsis tager udgangspunkt i Game Development valgfaget og udviklingen af et simpelt, grundlæggende 3D First Person Shooter spil ved brug af Unity Engine som dets værktøj.

Motivation

Jeg har altid været betaget af hvor gode spil er til at formidle historier på, der virkelig drager personer ind og gør dem en aktiv del af deres universer. Jeg har lyst at skabe mit helt eget, og undersøge hvad der skal til, og det er netop derfor dette emne er blevet valgt som mit fokusområde for 4. semesterets individuelle projekt.

Problemformulering

Hvordan udvikler jeg et 3D-spil i første person med Unity Engine?

- Hvordan implementerer jeg rigid bodies, colliders, triggers og ray i spillet?
- Hvad skal jeg gøre for at skabe en navigerende, kunstig intelligens til 3D objekter i Unity?
- Hvordan laver jeg en simpel angrebsmekanisme for spilleren, som har en fysisk effekt på elementer i 3D-banen, herunder eventuelle fjender og forhindringer?

Planlægning

Jeg har nedenunder udarbejdet en oversigt/skema af projektets proces, hvor jeg for hver uge, i den afsatte tid på 5 uger, har en række mål som skal nås og laves.

Uge Nr.	Ugentlige Mål
17	Research, oprettelse af et Unity Projekt, simpel controller og et 3D level, samt synopsisskrivning
18	Research, udvidelse af den simple controller og selve 3D levelet, samt synopsisskrivning
19	Mere research og implementering af AI, samt synopsisskrivning.
20	Endnu mere research og implementering af Ray Physics, samt yderligere udvidelse af 3D banen og synopsisskrivning
21	Finjustering af 3D-banen, funktionaliteten og synopsisskrivning, samt aflevering af den

For at sikre, at jeg har et færdigt produkt til når det er slut, har jeg valgt at først lave et fundament, som jeg for hver uge kan bygge videre ovenpå – det vil sige en rotation imellem de forskellige metoder jeg kommer ind på i afsnittet *Metode* på næste side.

Metode

Til udarbejdelsen af synopsis og besvarelse af problemformuleringen, har jeg valgt at anvende fire forskellige metoder, herunder:

- **Research:** søgning efter og redegørelse af relevant materiale, der kan hjælpe mig med udviklingen af 3D-spillet og problemformuleringens besvarelse. Materialet vil bestå af kurser, bøger, artikler, forummer, videoer, tutorials og lignende.
- **Undersøgelse:** dette vil være undersøgelsen af det som kan uddrages af research materialet i praksis, som kommer til udtryk i form af prototyper, forsøg og mere, og undersøgelsesresultaterne vil blive noteret.
- **Udførelse:** i dette punkt, vil jeg anvende alt det relevante der kan uddrages af research materialet, og dets praktiske undersøgelse, til at udvikle mit eget 3D spil som stemmer overens med problemformuleringen. Spillets udviklingsproces vil blive tydeligt dokumenteret og fremvist.

Research

Jeg vil, som nævnt tidligere under metodeafsnittet, researche efter relevant materiale som hjælp til at kunne besvare min problemformulering og dens problemstillinger. Dette afsnit vil dermed redegøre for hvilken teoretisk viden og praktiske foranstaltninger der skal til, for at kunne udvikle et førstepersonsspil i 3D.

Physics[1]

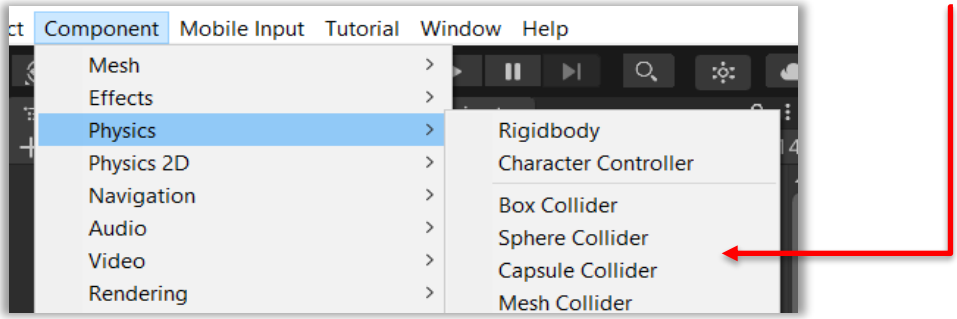
Unity Engine værktøjet består af en komponent kaldet 'Physics', som gør det muligt at manipulere med 3D objekter på en måde, der stemmer overens med naturens egne kræfter, som f.eks. tyngdekraft, fysisk interaktion, hastighed og friktion. Alt efter hvad der ønskes simuleret på et 3D objekt, benytter Physics sig af tilsvarende underkomponent(er) - det kan f.eks. være colliders', 'rigidbodies', 'rays', 'triggers' og flere, som alle vil blive redegjort for nedenunder.

[1] https://learn.unity.com/tutorial/intro-to-the-unity-physics-engine-2019-3#5f7cf02eedbc2a002070e081
https://www.youtube.com/watch?v=dLYTwDQmjdo

Colliders

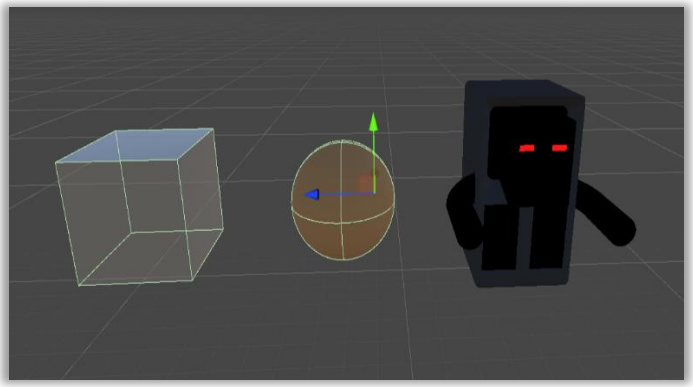
Collider er en komponent der bruges til at lave en fysisk afgrænsning af et 3D objekt, og kan sammen med rigidbody-komponenten gøre det muligt for Unity at vurdere når der forekommer kollisioner mellem forskellige objekter.[2]

Der er flere typer colliders som hver især passer bedst til den tilsvarende mesh form, herunder 'Box' og 'Sphere', og 'Mesh' hvis 3D objektets topologi er mere kompleks i det. [3]



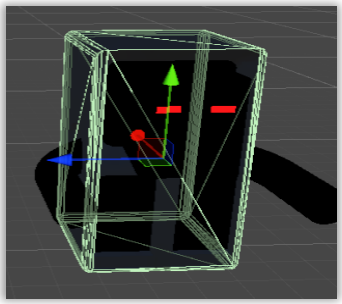
Når man tildeler et 3D objekt en collider, vil den får et grønt, omridsene kant for at indikere, at objektet netop besidder sådan en.

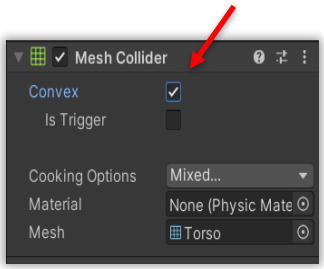
Figuren til højre bruger mesh collider - det vil sige, at collideren tager figurens mesh og konverterer den om til en collider, og grundet dens mere komplekse natur, er der ingen omrids.



Derudover vil objekter, der ikke har en optimeret mesh collider, ikke være i stand til at registrere når andre objekter prøver at interagere med dem via deres rigidbodies.

Hvis man ønsker at optimere en mesh collider til dette formål, skal man blot klikke på convex og der vil nu komme en grøn omrids bestående af trekanter.





I scripts kan man bruge visse funktioner til at bestemme hvad der skal ske når to spil objekter kolliderer med hinanden, herunder *OnCollisionEnter(), -Stay og -Exit.*

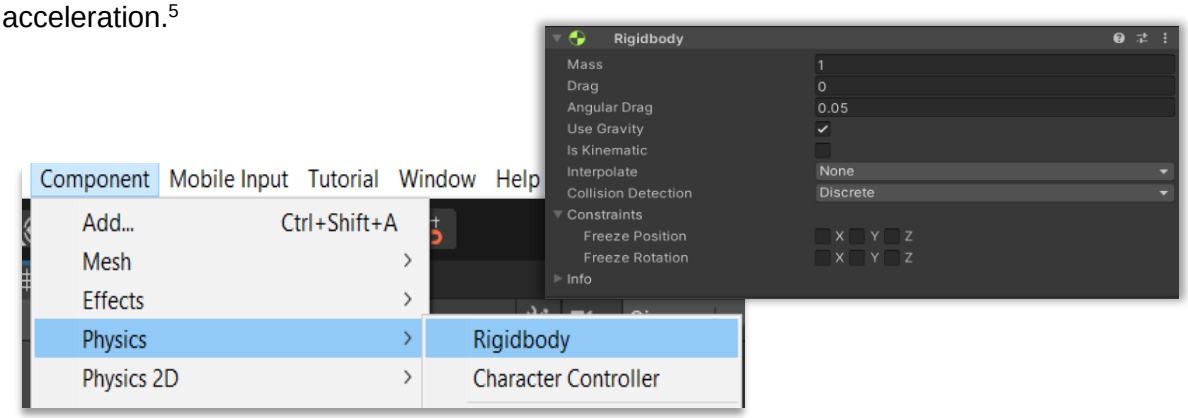
Trigger er en funktion som alle colliders besidder og kan aktiveres ved at klikke på 'Is Trigger'. Når funktionen er slået til, vil

[2] https://www.youtube.com/watch?v=6C4KfuW2q8Y
[3] https://docs.unity3d.com/Manual/CollidersOverview.html

dens fysiske afgrænsning i universet blive slået fra og vil i stedet kunne bruges som en slags aktiveringszone ved at registrere andre objekters tilstedeværelse i den. Man kan via scripts bruge følgende funktioner; 'OnTriggerEnter()', 'OnTriggerStay()' og 'OnTriggerExit' til at bestemme udfaldet af disse interaktioner.[4]

Rigidbody

Dette er en komponent som fortæller, at 3D objektet den er tilføjet til, skal kunne blive påvirket af, og være en del af den indbyggede fysik i Unity, såsom tyngdekraft, kollision og acceleration.[5]



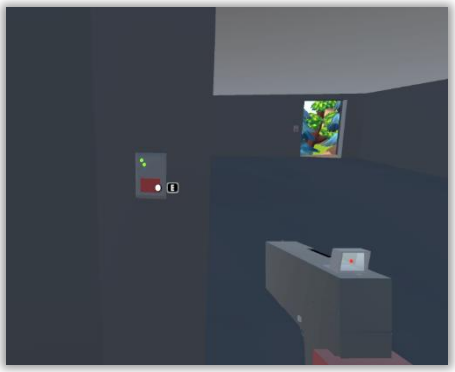
Efter en Rigidbody komponent er blevet tilføjet, er der en række egenskaber der kan manipuleres og 'tweakes med', så der opnås den ønskede fysiske effekt på 3D objektet.
De vigtigste egenskaber i udviklingen af et 3D spil i mit tilfælde er henholdsvis *Mass, Use Gravity* og *Collision Detection,* hvor den sidstnævnte handler om hvordan den skal håndtere kollisioner med andre objekter.
Der er desuden adskillige funktioner man kan bruge når det kommer til programmering og Rigidbodies, og den som er mest relevant for mit projekt er *AddForce()*[6] i forbindelse med skydning af projektiler. Sammen med AddForce kan man bruge *Instantiate*[7] funktionen, som skaber en klon af en vilkårlig prefab, der besidder en rigidbody og skyder den ud med en bestemt kraft.

[4] https://www.youtube.com/watch?v=F4TC7H6trLo
[5] https://docs.unity3d.com/ScriptReference/Rigidbody.html
[6] https://docs.unity3d.com/ScriptReference/Rigidbody.AddForce.html
[7] https://docs.unity3d.com/ScriptReference/Object.Instantiate.html

Ray

Hvis man ønsker at skabe en usynlig forbindelse mellem to objekter, kan Ray, der er en slags 3D stråle, bruges til netop dette formål. Det fungerer således at Ray ud fra dets startpunkt skaber en streg frem til dets slutpunkt, og penetrerer eventuelle objekter der befinder sig på dens vej.[8]

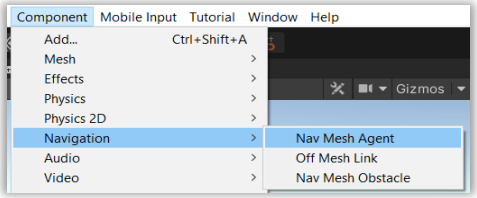
Det kan anvendes til en masse forskellige formål, såsom skydemekanismer, transformationer, trigger og Unity Events. I mit projekt bruges det til en simpel mekanisme, der åbner og lukker en dør ved at bruge ray til at interagere med den.



NavMesh

NavMesh (- Navigation Mesh), er en metode i Unity der kan bruges til at lave og tildele objekter en navigerende kunstig intelligens. [9]

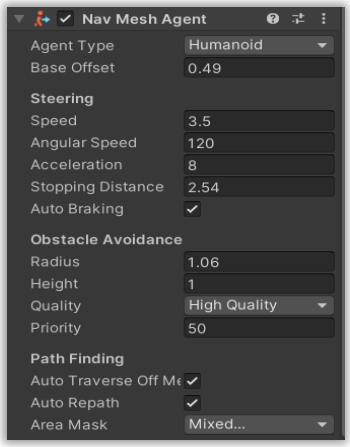
For at gøre dette, skal man først og fremmest give de gældende objekter en NavMeshAgent komponent, der giver dem de krævede forudsætninger for at kunne bruge et NavMesh underlag.



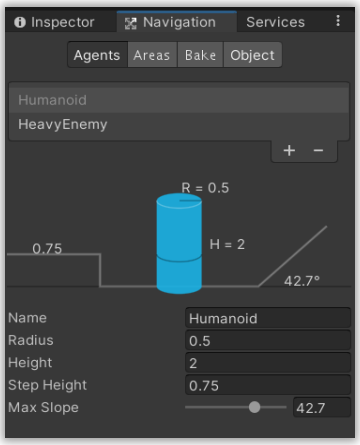
Når man tilføjer en NavMeshAgent komponent åbnes der en ny fane op, hvor alt med navigation kan oprettes og redigeres.

[8] https://docs.unity3d.com/ScriptReference/Physics.Raycast.html
https://www.youtube.com/watch?v=_yf5vzZ2sYE
[9] https://medium.com/@moesio.f/https-medium-com-moesio-f-unity-navmesh-tutorial-en-bfeeccd6169a
https://learn.unity.com/tutorial/unity-navmesh

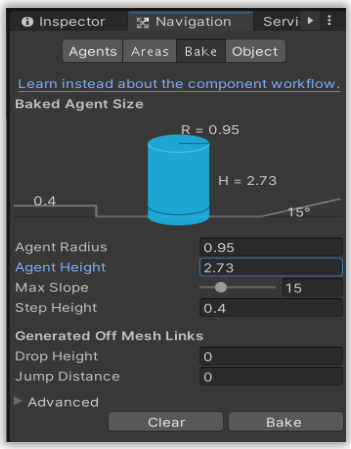
I den første fane under Navigation kan man oprette 'agents' som er en slags skabelon for hvilken fysiske egenskaber hver enkelte agent besidder i forhold til at navigere i spillets univers. En mere omfattende redigering foregår derimod i selve komponenten, hvor man skal blandt andet vælge hvilken agenttype den anvender, samt dets styring, forhindring og Path Finding.



NavMeshAgent Komponent



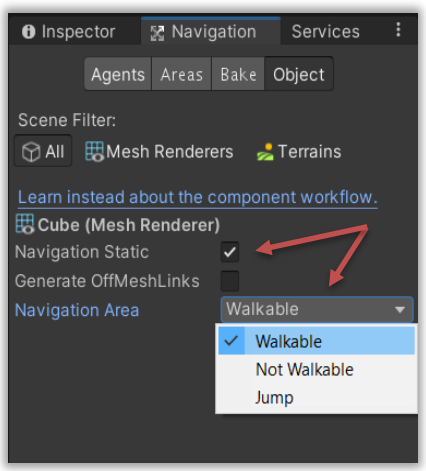
Navigation → Agents



Navigation → Bake

I den næste fane, 'Bake', tages alle de instillede værdier fra agent(erne) og objekt(erne) der blevet tildelt en navigation static, og bliver opført i 3D banen. Hver eneste gang man er under Naviagtion, vil alt det agenterne kan tilgå (eller ikke) blive markeret med et blåt lag ligesom fremvist i billedet nedenunder.

Hvis man ønsker at indstille hvad er navigerbart – eller ikke – skal man blot gå ind på *Navigation → Object,* og derefter trykke på det ønskede 3D objekt i sin 'scene'. Der vil nu komme nogle indstillinger frem og man skal trykke på *Navigation Static* for at gøre det bestemte objekt en del af NavMesh i banen.



Herefter skal man vælge hvilken type område, objektet skal symbolisere, hvor 'Walkable' gør det navigerbart og 'Not Walkable' til en forhindring på NavMeshen.

Når de ønskede objekter i banen er indstillet, skal man huske at påbegynde 'bakingprocessen' under 'Bake' for at gøre dem aktuelle og up-to-date.

Konklusion

Der kan konkluderes en del ud fra dette projekt som hjælper med besvarelse af problemformuleringen og dens problemstillinger - jeg vil dog blot skrive om sidstnævnte her i synopsis, og vente med problemformuleringen til selve fremlæggelsen.

Hvordan implementerer jeg rigid bodies, colliders, triggers og ray i spillet?

Unity gør det muligt, uden at skrive noget som helst kode, at implementere henholdsvis rigidbodies, colliders og triggers, der hver især består af nogle redigerbare 'default' værdier. Man skal blot markere de ønskede 3D objekter og tilføje komponenterne under sektionen, 'Components'

Ray er en mere kompliceret proces sammenlignet med implementering af de andre komponenter, da man bliver nødt til at programmere det i en C#-fil og tilføje det i banen på et objekt, der fungerer som dets startpunkt.

Man kan sige, at alle disse physics komponenter, er selve fundamentet til et vilkårligt spil, herunder mit eget - der følger en hvilken som helst fysik – som det kan udfolde sig på.

Hvad skal jeg gøre for at skabe en navigerende kunstig intelligens til 3D objekter?

Unity gør det også nemt for en når det kommer til implementering af en navigerende kunstig intelligens, da man med programmets indbygget komponent, kaldet *NavMesh,* ikke behøver at skrive koder, for at have intelligensen med i sit spil. Man har blandt andet mulighed for at indstille hvor hurtig de hver især er til at bevæge sig og hvor gode de er til at forudse forhindringer via *NavMesh komponenten* og *Navigationsfanen.* I mit kommer det til udtryk i at robotterne nemt følger efter spilleren, og undviger eventuelle forhindringer der placeret rundt omkring på banen - og endda andre robotter - på deres vej.

Hvordan laver jeg en simpel angrebsmekanisme for spilleren, som har en fysisk effekt på elementer i 3D-banen, herunder eventuelle fjender og forhindringer?

For at kunne lave en angrebsmekanisme, i mit tilfælde skydevåben, skal man bruge en prefab med en rigidbody og collider som projektil, og en 'projektil spawner' som kan skyde dem afsted. Man skal derudover have lavet et C#-fil hvor man skal gøre bruge af funktionerne *Instantiate*, som spawner en klon af projektilet, og *AddForce* der bestemmer hvor kraftig de skal skydes ud fra spawnen.

Refleksion

Forløbet af projektet har haft sine op- og nedture. Først havde jeg utroligt svært ved at komme i gang med at udvikle selve 3D banen fordi jeg følte at det var vigtigt at få historien bag spillet på plads – indtil jeg ikke længere ville udskyde det og begyndte improviserende at lave noget. Stille og roligt faldt alle brikkerne på plads, og jeg havde pludseligt en klar idé om hvilken retning jeg skulle tage spillet i, samtidig med at jeg lærte noget nyt; det hjælper nogle gange bare at gå i gang uden at alle detaljerne er på plads.

Jeg oprettede to scener i Unity, en 3D bane som spillet skulle foregå i og en anden hvor jeg udviklede alt funktionaliteten og prefabs. Det gjorde udviklingsprocessen af spillet meget mere effektiv og overskueliggjort, og pludselig havde man et lille fungerende et, som jeg så gradvist byggede videre ovenpå. Fremgangsmåden minder også om det jeg har lært i *Systemsudviklingsfaget,* hvor man udvikler et funktionelt produkt allerede i starten af processen, som man så kan udvikle videre på.

Jeg føler, at jeg fulgte det planlagte skema godt med især research og udvikling, hvor med synopsisskrivning haltede det lidt efter – dog endte det med at falde på plads til sidst alligevel.

For at runde det af, så den eneste bemærkelsesværdig udfordring jeg havde gennem hele projektet, var som nævnt tidligere, det med at få bolden rullende i begyndelsen og jeg har kunne bruge noget af det som jeg har lært fra uddannelsen til noget produktivt i forhold til dette projekt.

Kildehenvisning

Kildehenvisninger i numerisk rækkefølgen (1-9)

[1] Physics

- **Intro to the Unity Physics Engine - 2019.3,** *Unity Technology*, https://learn.unity.com/tutorial/ o-....
  Senest besøg: 28-05-2021

- **Unity3D Physics - Rigidbodies, Colliders, Triggers,** *Jason Weimann.*
  https://www.youtube.com/watch?v=dLYTwDQmjdo
  Senest besøg: 28-05-2021

[2] Collider

- **Unity Collisions Tutorial - How To Use Colliders and Triggers in Unity | Unity 5 Tutorial,** *Omnirift,*
  https://www.youtube.com/watch?v=6C4KfuW2q8Y
  Senest besøg: 28-05-2021

[3] Collider

- **Colliders,** *Unity Technology,* https://docs.unity3d.com/Manual/CollidersOverview.html
  Senest besøg: 28-05-2021

[4] Collider + Trigger

- **Unity3D Scripting: OnTriggerEnter, OnCollisionEnter, & Tags**, *Renaissance Coders,*
  https://www.youtube.com/watch?v=F4TC7H6trLo
  Senest besøg: 28-05-2021

[5] RigidBody

- **Rigidbody,** *Unity Technologies,* https://docs.unity3d.com/ScriptReference/Rigidbody.html
  Senest besøg: 28-05-2021

[6] RigidBody

- **Rigidbody.AddForce,** *Unity Technologies*
  https://docs.unity3d.com/ScriptReference/Rigidbody.AddForce.html
  Senest besøg: 28-05-2021

[7] Instantiate

- **Object.Instantiate,** *Unity Technologies,* https://docs.unity3d.com/ScriptReference/Object.Instantiate
  Senest besøg: 28-05-2021

[8] Ray

- **Physics.Raycast,** *Unity Technologies,* https://docs.unity3d.com/ScriptReference/Physics.Raycast.html,
  Senest besøg: 28-05-2021

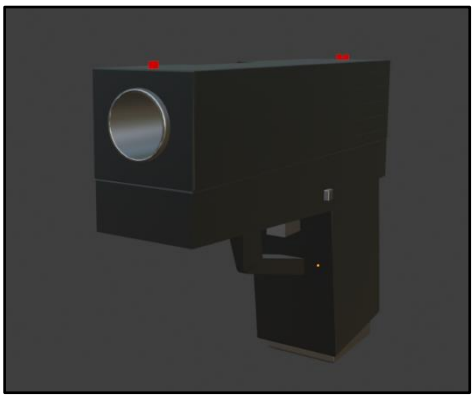
- **Selecting Objects with Raycast – Unity Tutorial,** *Infallible Code,*
  https://www.youtube.com/watch?v=_yf5vzZ2sYE
  Senest besøg: 28-05-2021

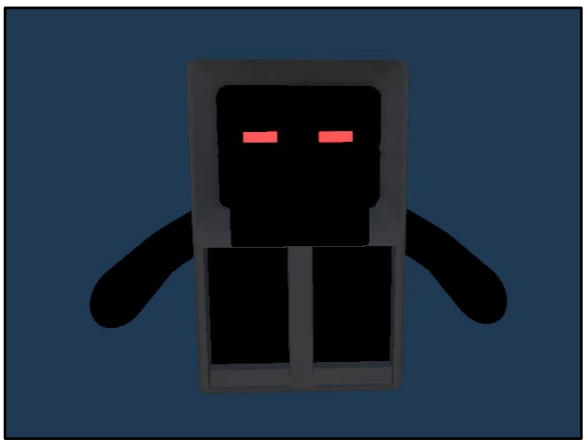
[9] NavMesh

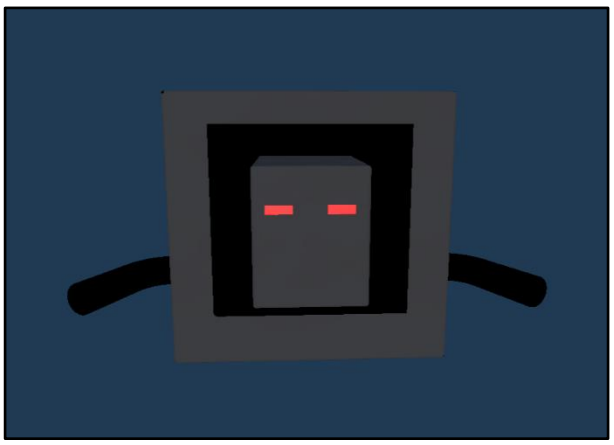
- **Unity – NavMesh Tutorial (EN),** *Moésio Filho,*
  https://medium.com/@moesio.f/https-medium-com-moesio-f-unity-navmesh-tutorial-en-bfeeccd6169a
  Senest besøg: 28-05-2021

- **Unity NavMesh,** *Brackeys,* https://learn.unity.com/tutorial/unity-navmesh
  Senest besøg: 29-05-2021

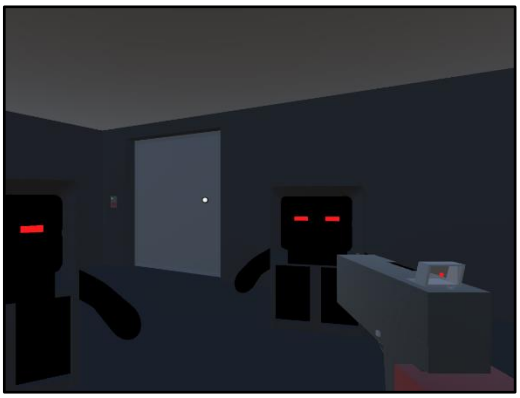
Bilag

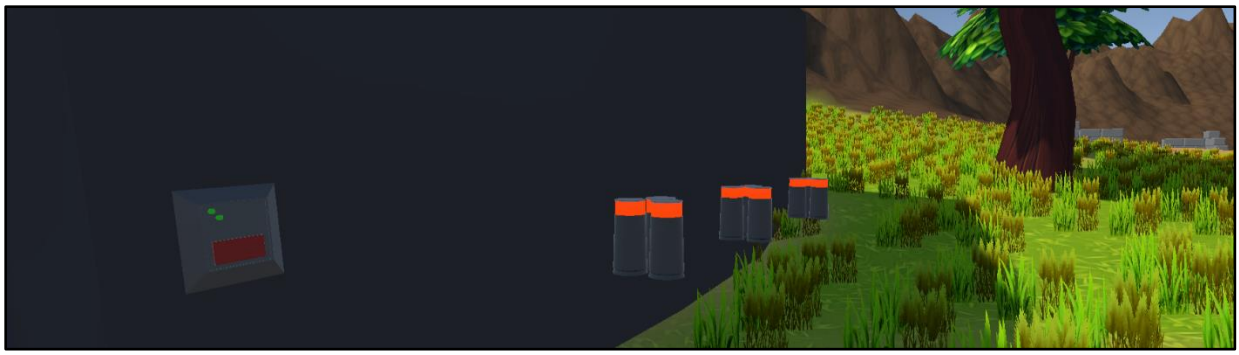


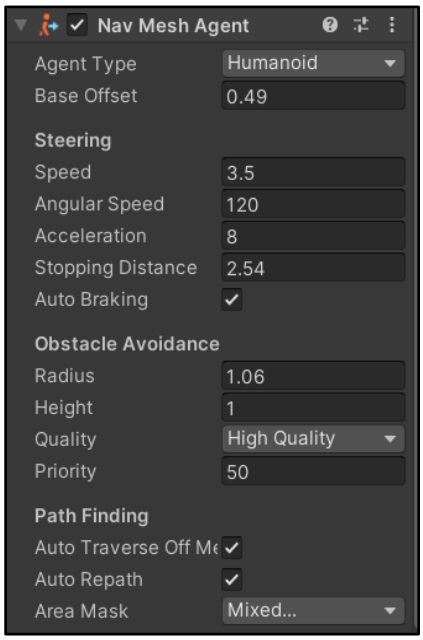


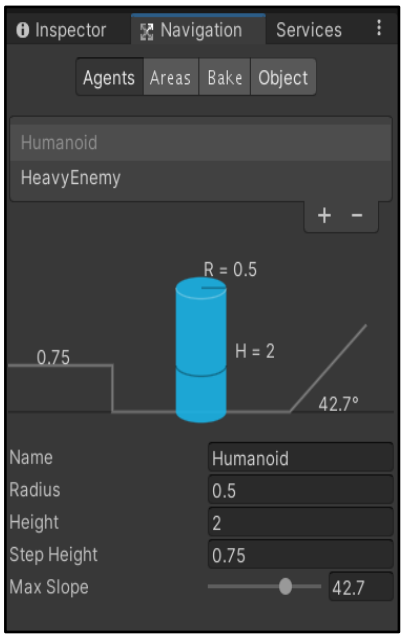


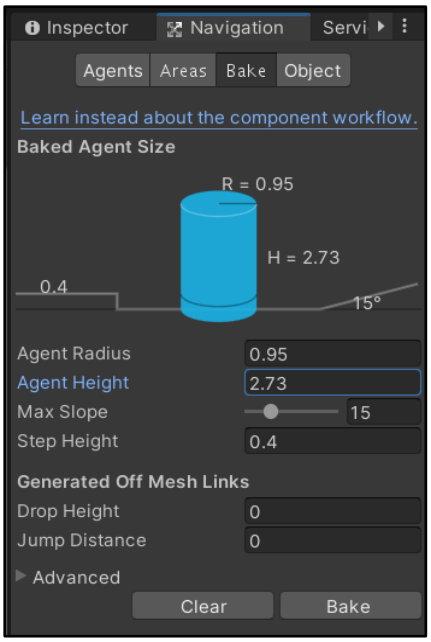

51

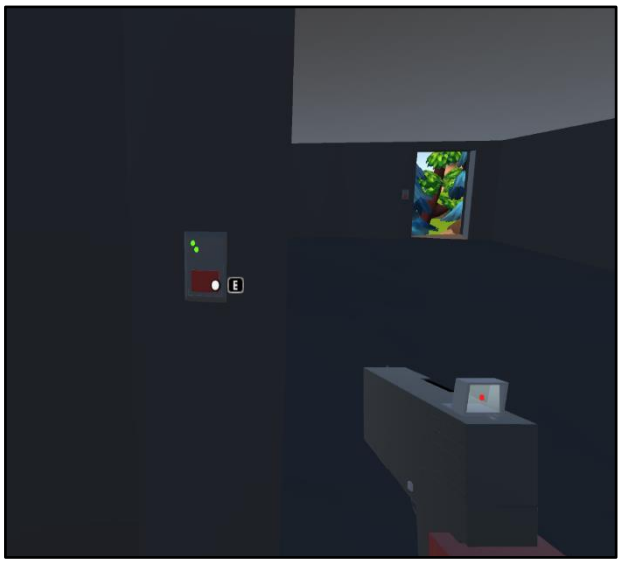












```
Rigidbody clone;

clone = Instantiate(bullet, bulletSpawner.position, bulletSpawner.rotation);
clone.AddForce(bulletSpawner.forward * shootingPower);
```