Examensarbete, Högskolan på Åland, Utbildningsprogrammet för Informationsteknik

VIRTUAL REALITY-SPEL

- utvecklat i plattformen Unity

Joannis Pouliou

2022:46

Datum för godkännande: 20.12.2022
Handledare: Björn-Erik Zetterman

EXAMENSARBETE
Högskolan på Åland

Utbildningsprogram:	Informationsteknik
Författare:	Joannis Pouliou
Arbetets namn:	Virtual Reality-Spel - utvecklat i plattformen Unity
Handledare:	Björn-Erik Zetterman
Uppdragsgivare:	Joannis Pouliou

Abstrakt

I mitt examensarbete skriver jag om virtuell verklighet (VR), spelmotorn Unity och processen jag använde för att skapa ett VR-spel. Syftet med arbetet är att öka mina färdigheter i att använda Unity. Resultatet är en prototyp av ett skjutspel som kan spelas med de flesta VR-plattformarna och headseten på marknaden.

Nyckelord (sökord)

VR, Virtual Reality, Unity, Videospel

Högskolans serienummer:	ISSN:	Språk:	Sidantal:
2022:46	1458-1531	Svenska	31 sidor

Inlämningsdatum:	Presentationsdatum:	Datum för godkännande:
4.12.2022	16.12.2022	20.12.2022

DEGREE THESIS
Åland University of Applied Sciences

Degree Programme:	Information Technology
Author:	Joannis Pouliou
Title:	Virtual Reality Game - developed with the Unity platform
Academic Supervisor:	Björn-Erik Zetterman
Commissioned by:	Joannis Pouliou

Abstract

In my thesis, I am writing about virtual reality (VR), the game engine Unity and the process which I used to create a VR game. The purpose of the project is to increase my skills in using Unity. The result is a prototype of a shooter game playable with most VR platforms and headsets on the market.

Keywords

VR, Virtual Reality, Unity, Video Game

Serial number:	ISSN:	Language:	Number of pages:
2022:46	1458-1531	Swedish	31 pages

Handed in:	Date of presentation:	Approved:
4.12.2022	16.12.2022	20.12.2022

INNEHÅLLSFÖRTECKNING

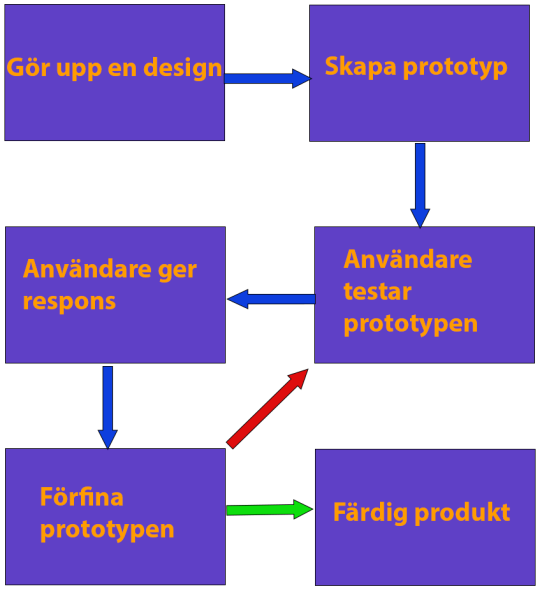
1. INLEDNING

1.1 Syfte

Eftersom jag har ett intresse av spelutveckling ville jag bli bekant och någorlunda kompetent i att använda den populära spelmotorn Unity och samtidigt lära mig hur VR-spel i synnerhet utvecklas. Syftet med detta projekt är att jag skall lära mig hur man kan implementera ett 3D-spel med VR-användargränssnitt, specifikt i detta fall med Unity 3D.

1.2 Metodik

För att få ett färdigt projekt krävs initialt inläsning och experimenterande, för att på basen av en kravspecifikation (se kap 3) arbeta inkrementellt, ta ett steg i taget, ett delmål, ett krav för att uppnå resultat. Metodiken kallas för *Evolutionary prototyping*, och beskrivs enklast som ett sätt där utvecklaren (eller teamet) bygger små delar, i små steg för att testa och ta steg framåt mot de mål som finns uppställda (Sherrell, 2013). Figur 1 illustrerar arbetsflödet.



Figur 1. Diagram över evolutionary prototyping

1.3 Avgränsning

Spelet är inte produktionsklart, då det är en prototyp med några viktiga element för att visa en demonstration av vad man kan göra med VR, och innehåller endast ett fåtal element såsom

endast en typ av fiende och vapen. Verktygen jag använder är de samma som används för utveckling av AR-applikationer. AR står för Augmented Reality och är skilt från VR och kommer därför inte att gås in i här.

2. TEORI

2.1 Virtual Reality (VR)

Virtual Reality innebär “virtuell verklighet” på svenska (Svenska Akademins Ordbok, 2021). “Virtual reality is a simulated 3D environment that enables users to explore and interact with a virtual surrounding in a way that approximates reality, as it is perceived through the users' senses” (Sheldon, 2022).

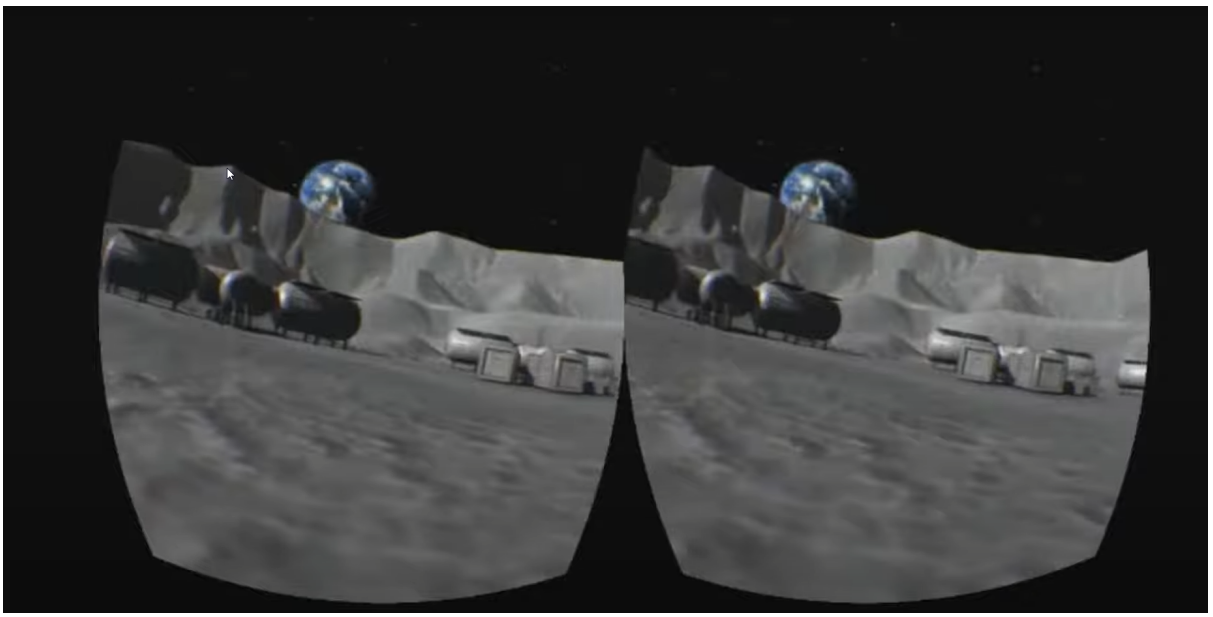
VR är en teknologi som gradvis börjar leta sig in i samhället. Teknologin ämnar simulera verkligheten för videospel men även andra viktiga syften såsom, utbildning, vetenskap, handel och industriproduktion (Bardi, 2019).

Teknologin man sätter på huvudet har många namn inklusive “VR-glasögon”, “VR-headset” och “HMD”, vilket står för “head mounted display”, alltså en huvudburen skärm. Framöver kommer jag att beskriva uppbyggnaden av VR-glasögon med särskild fokus på modellen jag har använt under projektet, vilket är en HP Reverb G2 Version 2, visad i figur 2.

Figur 2. HP Reverb G2 Version 2

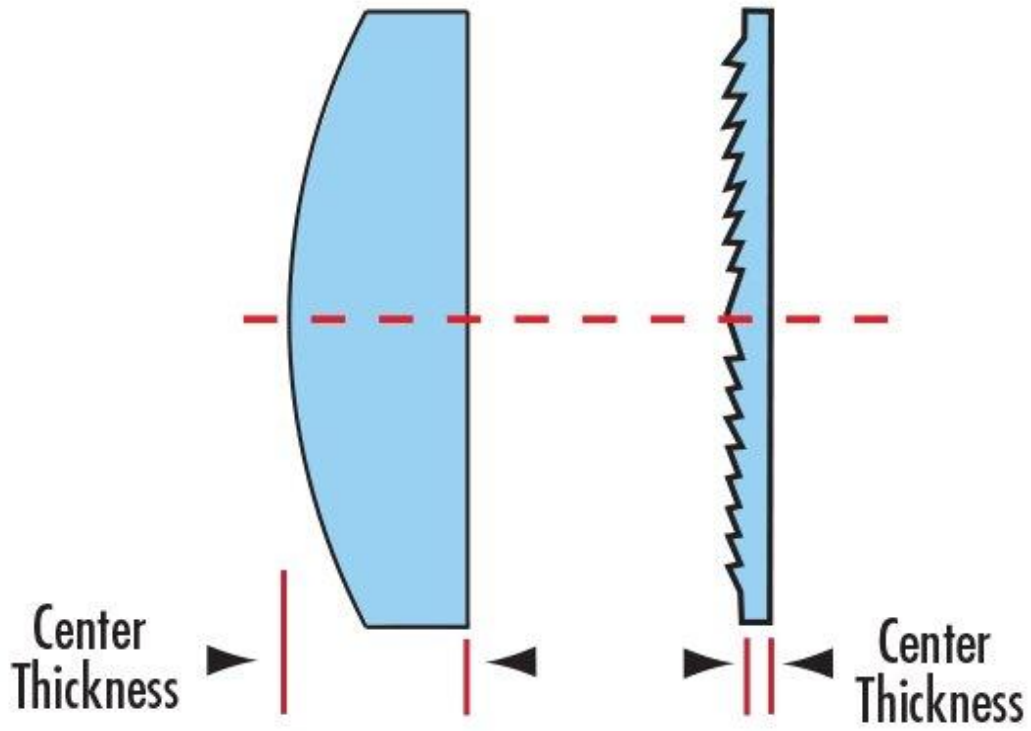
2.1.1 Linser

VR-glasögon innehåller två skärmar med linser framför för att optiskt ge “rätt intryck” åt användaren. Eftersom det mänskliga ögat inte kan fokusera på någonting som är så nära behöver ögats linser hjälp av linser framför skärmarna. Linserna böjer skärmarnas ljus så att då det träffar näthinnorna uppfattar hjärnan det som att ljuset kommer långt bortifrån fast det egentligen kommer från rakt framför ögonen. Två bilder av samma scen visas med en viss förskjutning mellan skärmarna (se figur 3). Resultatet blir att hjärnan kombinerar bilderna till en tredimensionell bild. Detta kallas för stereoskopisk 3D (Tikkanen, 1998). Med datorbundna HMD genereras bilderna av datorns grafikkort och hanteras av VR-headsetets drivrutin. Det finns även *stand-alone* headset där allt sker i headsetet. Eftersom två separata bilder måste genereras för att VR ska fungera är VR-applikationer dubbelt så resursintensiva på grafikkortet jämfört med om samma program skulle köras i ett läge utan VR.



Figur 3. Höger ögas bild är förskjuten jämfört med vänster (Advanced VR fare, 2021)

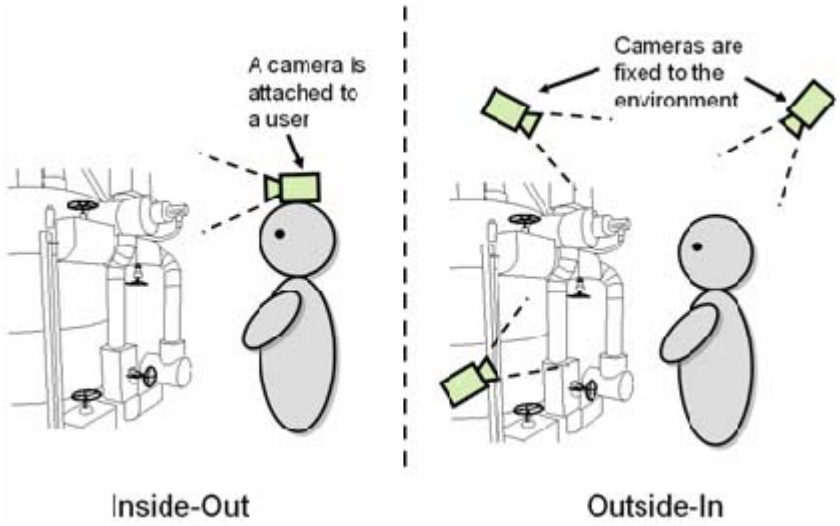
För tillfället används Fresnellinser i huvuddelen av VR-glasögon inklusive HP Reverb G2 eftersom de är billiga och lättviktiga men så kallade "pannkakslinser" har börjat användas i de senaste headseten på marknaden (SPIE Europe Ltd, 2022). Fresnellinser skapas genom att avlägsna stora delar av linserna som inte böjer ljus. Kvar blir en lins med samma funktion som originalet men med mycket mindre volym och massa (se figur 4).



Figur 4. Fresnellinser tar mindre rum än vanliga linser (Optics, 2011)

2.1.2 Spårning

Det finns två huvudkategorier inom VR-spårningsteknologi, outside-in och inside-out. Glasögon som använder outside-in-spårning kräver att sensorer sätts ut i rummet för att hålla koll på glasögonen och kontrollerna. HP Reverb G2 använder sig istället av inside-out-spårning vilket innebär att spårningsteknologin finns inbyggd i headsetet istället (Ishii, 2010, p. 320). Runt om headsetet finns 4 svartvita kameror med låg upplösning. Två fram på och en på vardera sida. Drivrutinerna kalkylerar headsetets position i rummet genom att kombinera visuell data från kamerorna med IMU[1]-data från andra sensorer i headsetet (Sean-Kerawala, 2020). För att spåra kontrollerna med inside-out spårning finns många små infraröda lampor i en ring ovanför själva greppet så att kamerorna kan identifiera dem och deras orientering (Gajsek, 2022).



Figur 5. Inside-out jämfört med outside-in spårning (Ishii, 2010, p. 320)

2.2 Unity

Unity är en spelmotor med förmågan att användas för utveckling av både tredimensionella och tvådimensionella applikationer. En spelmotor är en utvecklingsmiljö som gör det lättare att utveckla spel. De snabbar upp spelutveckling enormt genom att ge en färdig grund för mycket vanliga men svårutvecklade system såsom fysiksimulation och artificiell intelligens (Arm Ltd, 2022).

[1] IMU står för inertial measurement unit (Vectornav, 2022)

2.2.1 Unity editor

I figur 6 har jag markerat funktioner i editorn som kommer att förklaras under bilden. Det här är standardpositioner för verktygen men de behöver inte finnas på samma ställen som i bilden eftersom alla fönster och menyer kan flyttas till valfri position i editorn. Fler fönster kan även öppnas ifrån “Window”-menyn i verktygsfältet.

Figur 6. Unitys editor. Jag har market relevanta områden för beskrivning nedan (Unity Technologies, 2022c)

Följande listas numrering motsvarar siffrorna i bilden och representerar väsentliga funktioner för att använda Unity (Unity Technologies, 2022e):

(1) Projektmenyn innehåller alla filer som tillhör projektet. Filerna behöver inte vara i användning någonstans i spelet men de är en del av projektet och är redo att användas. Vanliga filer inkluderar scener, material, texturer, ljud, och prefabs. Importerade resurser från Unity Asset Store eller andra källor hamnar här.

(2) Hierarkimenyn visar alla objekt som existerar i den öppna scenen. Allt som ses här kan återfinnas i spelvärlden. Då ett objekt dras ovanpå ett annat blir det ett barnobjekt till det andra objektet. Barnobjekt följer med föräldern då den rörs och barnets position utgår från förälderns position istället för världens origo (x=0,y=0,z=0). Om man dubbelklickar ett objekt i hierarkimenyn tas man till objektet i scenvyn.

(3) Scenvyn är var objekt placeras ut i världen. Man kan flyga runt i första person i spelvärlden och editera den. Man kan flytta position, rotera och skala storleken på

objekt. Objekt kan markeras både i scenvyn och hierarkimenyn. Flera scener, alltså världar, kan existera i projektet. Man kan öppna dem i scenvyn via projektmenyn och de kan hoppas mellan då spelet körs genom en funktion i ett skript.

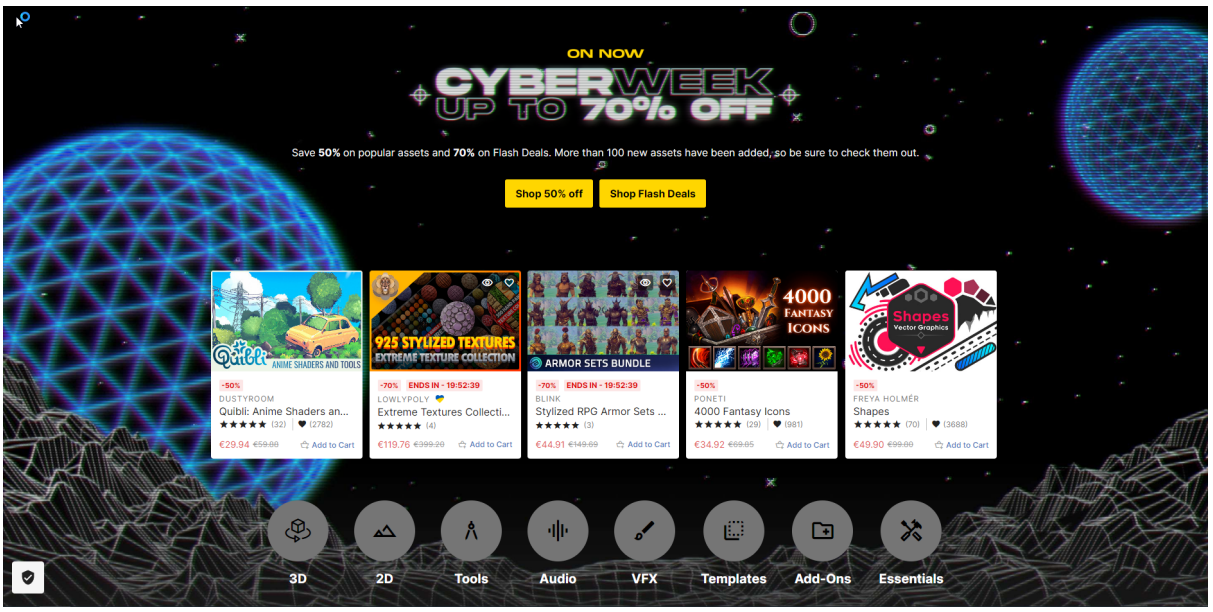
(4) Då ett objekt är markerat visas inspectormenyn. I inspectormenyn syns alla egenskaper objektet har. Alla objekt har en "Transform" som bestämmer var i världen de befinner sig i relation till ettdera världens mittpunkt eller föräldraobjektet. Många olika egenskaper kan läggas till på ett objekt, till exempel olika typer av colliders vilka gör att objektet kolliderar med andra objekt som också har colliders, eller Rigid Body vilket gör att objektet påverkas av gravitation. C#-skript kan läggas till på objekt som en egenskap. I dem kan man med kod påverka objektet eller andra relaterade objekt.

(5) Man kan testköra spelet genom att trycka på pilen. Då växlas man automatiskt till spelvyn och får ta kontroll över spelet. Projektet kan fortfarande editeras i de andra vyerna medan spelet körs men alla förändringar går förlorade då man slutar testspela.

(6) Om man trycker på den här fliken växlar man från projektmenyn till konsolfönstret. Här visas error meddelanden och strängar passerade till Debug.Log-metoden i ett C# skript.

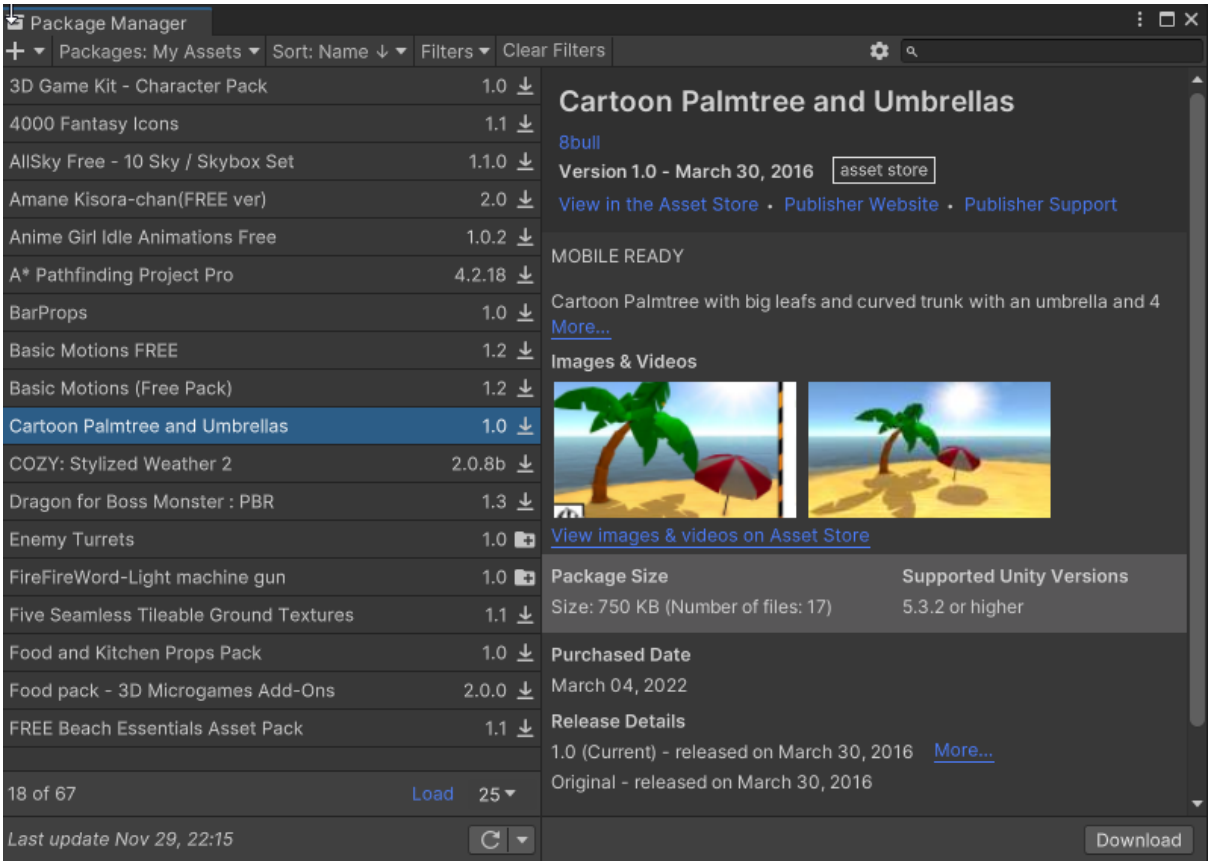
2.2.2 Unity Asset Store

Unity har en egen butik där utvecklare kan köpa en stor mängd olika resurser för att göra arbetet lättare. I figur 7 visas butikens framsida under en rea. Till salu finns 3D-modeller och 2D-ikoner, ljudfiler inklusive musik, kraftfulla verktyg som helt ändrar hur man använder Unity och mycket mer. Det finns även många gratis resurser i butiken vilka jag har utnyttjat några av i mitt projekt (Unity Technologies, 2022b).



Figur 7. Framsidan på Unity Asset Store under Black Friday rean (Unity Technologies, 2022d)

När man äger en resurs från butiken kan man importera den till sitt projekt genom "Package Manager"-fönstret (figur 8) som kan öppnas från "Window"-knappen i verktygsfältet (Unity Technologies, 2021).



Figur 8. Package manager (Unity Technologies, 2021)

2.2.3 XR Interaction Toolkit

Om man har tänkt utveckla sitt VR-spel för en specifik utgivares plattform kan man använda plugins skapade av ägarna av plattformen man siktar på såsom SteamVR[2] eller Oculus/Meta[3]. I alla andra fall lönar det sig att använda sig av Unitys XR Interaction Toolkit (Unity Technologies, 2022h). Spel skapade med XR Interaction Toolkit kan lätt köras på alla plattformar. XR står för "Extended Reality" vilket är en samlingsterm för VR och AR (Arm Blueprint staff, 2022).

För att börja använda XR Interaction Toolkit skapar man en "XR Origin" i spelvärlden. Det är ett spelobjekt med ett kopplat skript för att hålla reda på headsetets position. Kameran som representerar spelarens syn i spelet blir ett barnobjekt till XR Origin (Unity Technologies, 2022i). För att kunna använda kontrollerna skapar man fler spelobjekt som barn till XR Origin. I spelobjekten som representerar spelarens händer sätts ett av två "interactor"-skript. XR Ray Interactor, vilken använder en laserstråle för att interagera med saker på långt håll (Unity Technologies, 2022j), eller XR Direct Interactor, vilken bara kan interagera med saker inom spelarens räckvidd (Unity Technologies, 2022f). I alla saker som spelaren skall kunna interagera med sätts XR Grab Interactable-skriptet. *Interactable*-skriptet har inställningar för hur objektet ska bete sig då det hålls av spelaren. Exempelvis om det ska följa handen ohindrat eller kollidera med världen (Unity Technologies, 2022g).

För att spelet ska fungera med kontroller ifrån alla möjliga typer av VR headset, översätts knapptryck till standardiserade input "*actions*". Interactable-skriptet lyssnar efter en rad olika händelser från en närvarande interactor och kör specifierad kod då det sker. En metod från ett script på samma objekt kan användas eller så kan man ändra enkla saker såsom objektets material utan kod. Följande actions kan lyssnas efter (Unity Technologies, 2022a):

- First Hover Entered
  Aktiveras då en interactor kommer nära men inte om en till interactor kommer nära medan den första är kvar.
- Last Hover Exited
  Aktiveras då alla närvarande interactors har avlägsnat sig.

[2] Steam VR är Valves VR plattform (Valve Corporation, 2022)
[3] VR plattformen som förut hette Oculus har köpts av Facebooks ägare, Meta (Crunchbase, 2014)

- Hover Entered

  Aktiveras varje gång en interactor kommer nära.
- Hover Exited

  Aktiveras varje gång en interactor avlägsnar sig.
- First Select Entered

  Kallas då en interactor trycker på Select(välj)-knappen medan ingen annan interactor redan har valt den.
- Last Select Exited

  Kallas då alla interactors har slutat välja interactablen.
- Select Entered

  Kallas då en interactor trycker på Select-knappen.
- Select Exited

  Kallas då en interactor slutar välja interactablen.
- Activated

  Kallas då en interactor trycker på Activate-knappen efter att tidigare ha valt samma objekt med Select-knappen.
- Deactivated

  Kallas då Activate-knappen släpps.

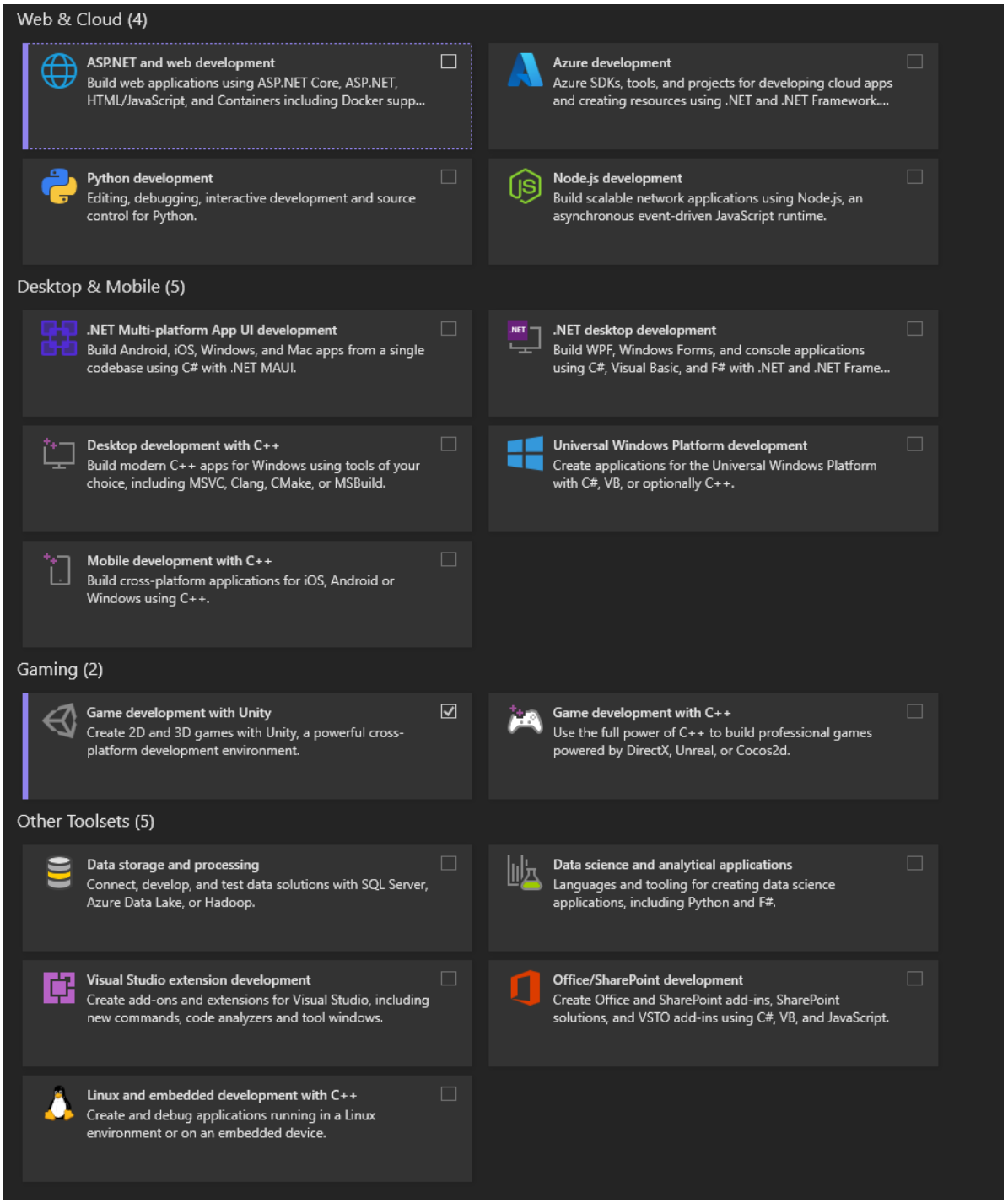
2.3 C#

Skript i Unity skrivs i huvudsak i programmeringsspråket C# med möjlighet att använda några andra såsom Rust. C# uttalas "*C Sharp*" efter musikens höjda noter. Det är ett objektorienterat språk med statiska datatyper utvecklat av Microsoft. Språket har automatisk sophantering vilket innebär att oanvända objekt automatiskt förstörs så att minnesläckor inte uppstår (Hejlsberg et al., 2008). Det släpptes som *open source* 2014 vilket medförde att det kunde användas för utveckling utanför Windows (Martin, 2014).

2.4 Visual Studio 2022

Visual Studio är ett IDE utvecklat av Microsoft vilket släpptes samtidigt som C# och .NET. Det ämnas i huvudsak för deras .NET ekosystem men det stöder också allt i figur 9 genom

moduler som kan installeras via en skild installationsapp. Det finns en särskild modul för integration med Unity som ses vara installerad genom bocken i figur 9 (TerryGLee, 2022).



Figur 9. Visual Studios moduler (Microsoft, 2022)

3. KRAVSPECIFIKATION

“I examensarbetet används begreppet kravspecifikation och definieras enligt följande: ett dokument innehållande en förteckning över de krav och behov beställaren har” (Franzén & Narse, 2009, p. 8). Beställaren i det här fallet är jag själv så därför skapade jag en lista på vaga riktlinjer innan jag börjat jobba på mitt projekt. Ska kraven har högst prioritet och måste finnas i den färdiga produkten medan bör kraven utförs om det finns tid. Alla krav för projektet uppnåddes.

3.1 Skakrav

1. 3D-spelmiljö, grafisk Virtual Reality som fungerar med VR-headset.
2. Stationära måltavlor (grafiska objekt), som registrerar träff/interaktion
3. Vapen som spelare kan använda för att skjuta på mål (hit scan vapen)
4. Startmeny för spel
5. Tillstånd då spelet är slut, resultat?

3.2 Börkrav

6. Rörliga mål som registrerar träff/interaktion

4.IMPLEMENTERING AV PROTOTYP

4.1 VR-miljön

Jag började med att skapa spelvärlden (figur 10). Unity har ett fåtal enkla former inbyggda så jag började att använda dem. Öknen är en *plane* utsträckt till 100x100 meter. Planes är ytor utan djup. Huset i öknen är en restaurang som jag skapade då konceptet för spelet var en matlagningssimulator. Väggarna på huset är omskalade kuber. De mer komplicerade 3D-modellerna såsom träden är gratis produkter från Unity Asset Store.

Figur 10. Överblick av spelvärlden

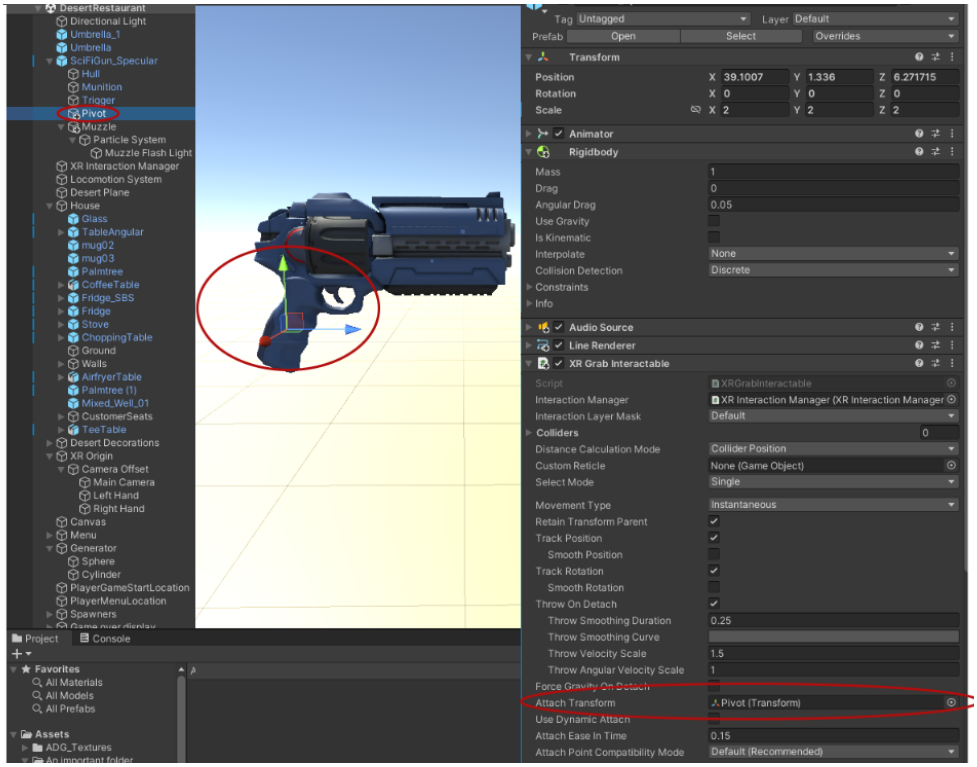
För VR-integrering använde jag unitys XR-system. Jag skapade en XR Origin där spelaren skall stå. XR Origin innehåller script som håller reda på var spelarens huvud är. XR Origin har ett barnobjekt som heter “Camera Offset” (figur 11). Det är ett tomt objekt som vars position används för att bestämma hur lång spelaren är. Det vill säga hur långt ifrån marken huvudet ska vara. Camera offset har sedan egna barn som har skript för händernas funktionalitet bifogade.

Figur 11. XR Origin och dess barnobjekt

Jag tog bort händerna som automatiskt kommer med eftersom de använder sig av Ray Interactors vilket jag inte tänkte använda. Istället ersatte jag dem med tomma objekt och bifogade vänster respektive höger grip interactor komponent. Jag använde mig av en fil som kommer med XR-paketet för att automatisk fylla i standardinställningar för höger och vänster kontrollers knappbindningar.

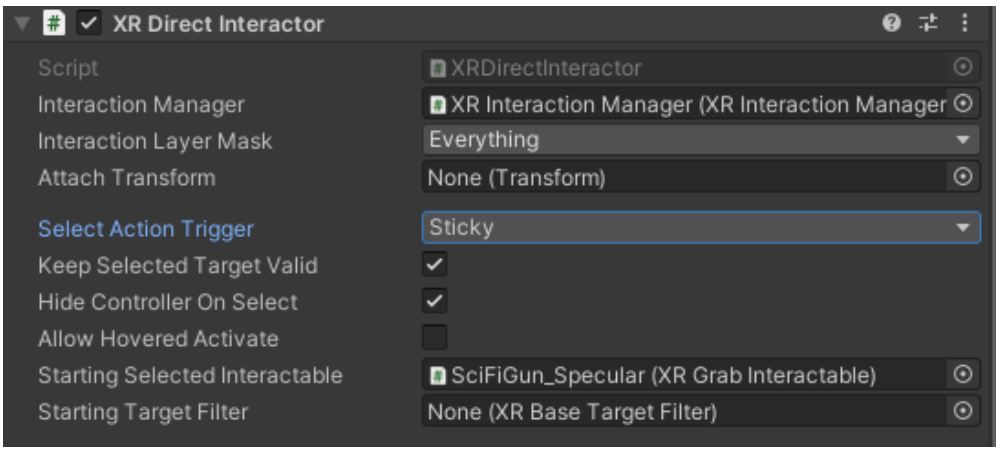
4.2 Vapen

Jag importerade ett lämpligt vapen från Unity Asset Store och lade en XR Interactable-komponent på det. För att vapnet skulle hållas i handen rätt skapade jag ett tomt spelobjekt på greppet som pekar åt rätt håll och lade en referens till dess Transform i XR Grab Interactable-skriptet (figur 12).



Figur 12. Punkten vapnet hålls i handen från och var man kan ge en referens till den

Jag gjorde så att vapnet automatiskt hålls i spelarens högra hand då scenen laddas genom en referens i interactor skriptet. Sedan valde jag "*sticky*" för Select Action Trigger variabeln i samma skript så att vapnet inte släpps direkt (figur 13).



Figur 13. Mina interactorinställningar

Nu började jag kodandet med att skapa ett skript för vapnets funktionalitet. Jag valde att använda mig av *raycasting* för att skapa vad som kallas för ett "hitscan"-vapen inom skjutspelsgemenskapen. Hitscan-vapen använder sig inte av projektiler utan istället dras ett streck (raycast) framåt från vapnets mynning och hämtar information om vad det träffar. Jag bifogade skriptet "Revolver" till vapnet och skapade en metod "TriggerPulled" i skriptet. Nu kan den kallas från XR Grab Interactable-skriptet då "Activate"-*action* händer, alltså när utlösarknappen trycks (figur 14).

Figur 14. Activate action kopplad till TriggerPulled funktionen

I TriggerPulled börjar jag med att starta en *coroutine* för att hantera de visuella effekterna som jag har satt i en annan funktion "ShotEffect". *Coroutines* är unika för Unity och är alltså inte en standard del av C#. De körs parallellt med den andra koden men använder inte *multithreading*. I ShotEffect spelar jag upp ett ljud, visar ett ljus i mynningen och sätter på en laserstråle för ett ögonblick (figur 15).

```
//Start ShotEffect coroutine to turn the laser line on and off
StartCoroutine(ShotEffect());

1 reference
private IEnumerator ShotEffect()
{
    //Make shooting sound
    source.PlayOneShot(fireSound);
    //Show muzzle flash
    muzzleFlash.Play();

    //Turn on line renderer
    laser.enabled = true;

    //Wait for 0.07 seconds
    WaitForSeconds shotDuration = new WaitForSeconds(0.07f);
    yield return shotDuration;

    //Deactivate line renderer after waiting
    laser.enabled = false;
}
```

Figur 15. Koden för skottens visuella effekt

Efter att ha satt igång *coroutine* ser jag till att riktningen vapnet pekar och laserns startpunkt är uppdaterade. Nu skickar jag ut *raycast* och beroende på om det träffar eller inte hanteras slutpunkten på lasern annorlunda. Om den träffar sätter jag slutpunkten på lasern till träffpunkten (figur 16).

```
if (Physics.Raycast(muzzle.position,muzzleDirection,out hit, range))
{
    //If the raycast hit set the endposition to the hit location
    laser.SetPosition(1, hit.point);
```

Figur 16. Koden då raycast träffar

Om raycasten missar finns det ingen träffpunkt att sätta lasern till och därför sätter jag den till en punkt i riktningen vapnet pekar, vid vapnets maximala räckvidd (figur 17).

```
else
{
    //If laser did not hit, shoot laser to max range
    laser.SetPosition(1, muzzle.position + (muzzleDirection * range));
}
```

Figur 17. Koden då raycast missar

Om strålen träffar någonting med kollision i spelvärlden undersöker jag ifall det träffade objektet har en utav de fyra skripten jag har gjort för interaktion med världen (figur 18). Om

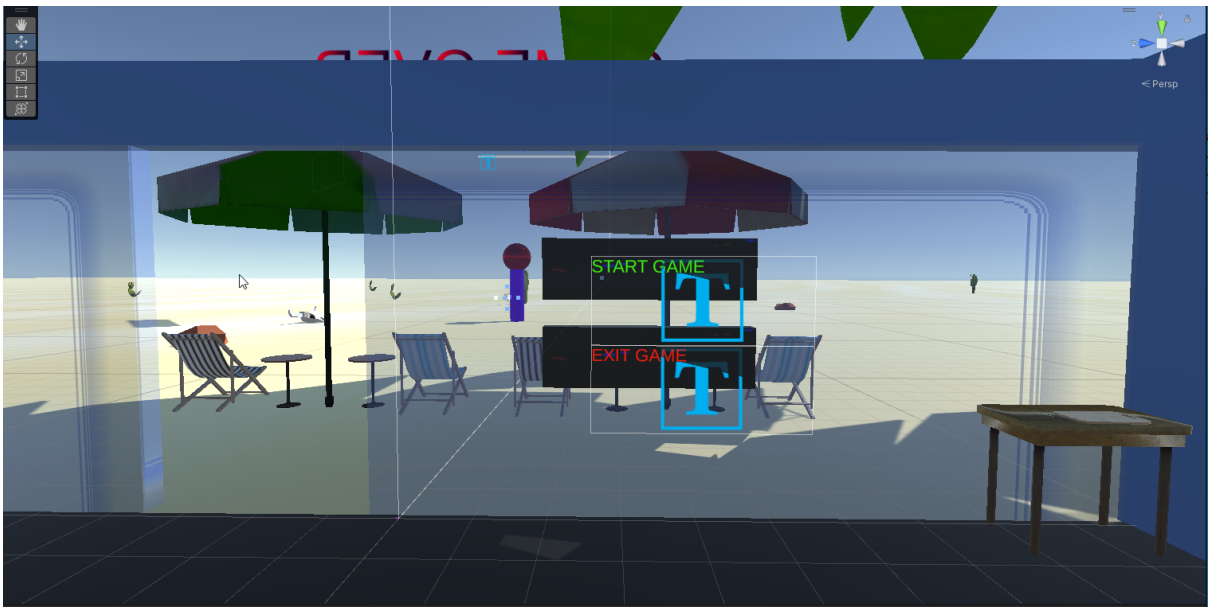
så är fallet kallas en publik funktion från skriptet. De här skripten förklaras i kommande kapitel.

```
//Enemies and destructables will have the "Target" script that removes health from them
Target targetScript = hit.transform.GetComponent<Target>();
if(targetScript != null)
{
    targetScript.TakeDamage(damage);
}

//Menu buttons below!

//Hit start button
StartScript startScript = hit.transform.GetComponent<StartScript>();
if (startScript != null)
{
    startScript.StartGame();
}
//Hit exit button
CloseGameScript closeScript = hit.transform.GetComponent<CloseGameScript>();
if (closeScript != null)
{
    closeScript.ExitGame();
}
//Hit continue button after losing
RestartScript restartScript = hit.transform.GetComponent<RestartScript>();
if (restartScript != null)
{
    restartScript.ReloadScene();
}
```

Figur 18. Koden som kallar rätt funktion då ett interaktivt objekt träffas

4.3 Startmeny

I de flesta applikationer består menyer av tvådimensionella knappar och text. I en VR-applikation måste menyerna också vara i spelvärlden för att spelaren skall kunna interagera med dem. I mitt fall har jag valt att sätta min meny i huset från tidigare (figur 19).



Figur 19. Startmenyn

På väggen har jag satt två block med kollision bakom med textobjekt framför. Då spelaren skjuter ett alternativ kallas relevant kod som förklarat i föregående kapitel. “EXIT GAME” stänger ner applikationen med den inbyggda Application.Quit-funktionen. “START GAME” gör tre saker för att sätta igång spelet. Först teleporteras spelaren framför generatorn som ska skyddas, sedan aktiveras alla fiendeframbringare (se kap 4.4) så att drönare kan börja attackera, och sist startas en timer som håller reda på hur länge spelaren överlever (figur 20).

```
public void StartGame()
{
    //Teleport player
    player.transform.position = startLocation.position;
    //Start the enemy spawners
    spawnnerRef.SetActive(true);
    //Start game timer
    timerStarted= true;
}
```

Figur 20. Koden då spelet startar

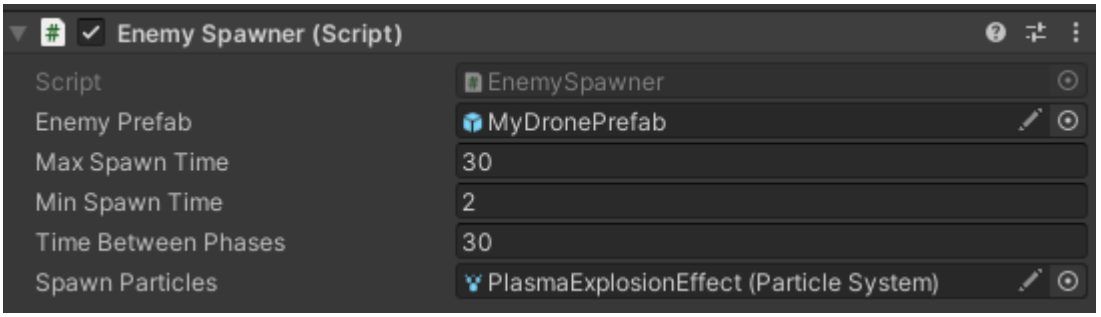
Timern fungerar genom Unitys Update-funktion. Update kallas en gång varje *frame*, alltså varje gång bilden uppdateras, så för att för att hålla koll på tiden ökar jag en variabel med Time.deltaTime varje gång (figur 21). Time.deltaTime är tiden det tog mellan förra *frame* och den nuvarande.

```
private void Update()
{
    if (timerStarted)
    {
        gameTimer += Time.deltaTime;
    }
}
```

Figur 21. Koden som ökar timern

4.4 Frambringare

Frambringarna (eng. *spawner*) som aktiveras av startskriptet är åtta osynliga tomma objekt med skript bifogade. De har som uppgift att skapa nya instanser av fiender. Genom frambringarna har jag implementerat en ökande svårighetsgrad. Svårighetsgraden för individuella frambringare är lätt att justera via de synliga variablerna i inspektorn (figur 22). De skulle även kunna frambringa andra saker bara genom att byta ut prefabreferensen. Prefabs är objekt som inte finns i spelvärlden men finns i projektmenyn, redo att instansieras i världen.



Figur 22. Variabler som kan ändras för att anpassa frambringare

I koden använder jag två separata timers som fungerar på samma sätt som timern i förra kapitlet (figur 23). En timer räknar ner tills att en ny fiende skapas medan den andra räknar ner till en svårighetsökning för den frambringaren. Tiden mellan svårighetsgradsökning är konstant men tiden mellan nya fiender slumpas mellan givna max- och minvärden.

```
private void Start()
{
    difficultyTimer = timeBetweenPhases;
    enemySpawnTimer = Random.Range(minSpawnTime, maxSpawnTime);
}

//Spawn enemies with random timeintervals. Max wait time grows shorter over time.
Unity Message | 0 references
void Update()
{
    //count down till difficulty increase
    difficultyTimer -= Time.deltaTime;
    //count down till enemy spawn
    enemySpawnTimer -= Time.deltaTime;
```

Figur 23. Koden för de två timerna

Nya fiender skapas genom att instantiera en kopia av fiendens prefab. Efter att en ny fiende skapats slumpas timern igen (figur 24).

```
if (enemySpawnTimer <= 0)
{
    //particle effect when spawning enemy
    Instantiate(spawnParticles, transform);
    //spawn enemy
    Instantiate(enemyPrefab, transform);

    //reset timer
    enemySpawnTimer = Random.Range(minSpawnTime, maxSpawnTime);
}
```

Figur 24. Koden för att skapa en fiende

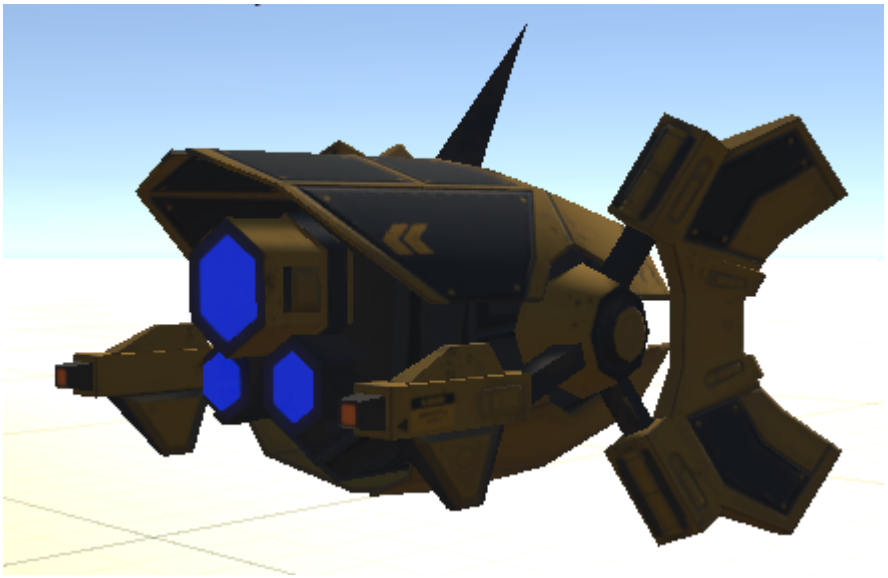
Vad svårighetsökningen innebär är att den maximala tiden mellan nya fiender blir kortare. Detta fortsätter ända tills maximumvärdet når minimumvärdet varpå maxvärdet inte sjunker lägre och fiender fortsätter skapas med minimitiden som uppehåll (figur 25).

```
if (difficultyTimer <= 0)
{
    //decrease maximum spawntime, increasing difficulty/spawnrate
    maxSpawnTime -= 1f;
    //limit maxSpawnTime decrease to the level of minSpawnTime
    if(maxSpawnTime<= minSpawnTime)
    {
        maxSpawnTime= minSpawnTime;
    }

    //reset timer
    difficultyTimer = timeBetweenPhases;
}
```

Figur 25. Koden som ökar svårighetsgraden

4.5 Fiender

Fienderna i spelet är drönare. De ser ut som i figur 26 och deras mål är att förstöra den abstrakta formen framför restaurangen. Jag har föreställt mig den som en science fictioninspirerad sköldgenerator och kallar den därför hädanefter för en "generator".



Figur 26. Fiendernas utseende

Efter de skapats av en framkallare vänder de sig mot generatorn med LookAt-funktionen och rör sig närmare i en rak linje med MoveTowards-funktionen. När drönaren kommer inom den specificerade räckvidden slutar den röra sig och aktiverar en laser med start och slutpunkten

hårdkodade fram på drönaren respektive på generatorns klot. Generatorn skadas varje *frame* drönaren är inom räckvidd genom att kalla en publik funktion i ett skript på generatorn.

4.6 Spelets slut

Spelet är över då generatorns livspoäng når noll. Tre avstängda spelobjekt aktiveras då framför spelaren. Två textobjekt och en menyknapp av samma typ som i startmenyn (figur 27).

Figur 27. Game over skärmen i editorn

Det övre textobjektet visar alltid texten “GAME OVER” men det undre objektet uppdateras i koden för att visa antalet sekunder spelaren överlevde som har hållits koll på i skriptet från startmenyn. Framkallarna stängs av och därmed försvinner även alla drönare eftersom de instantieras som barnobjekt till framkallaren som skapade fienden. En explosionseffekt loopas oändligt på generatorn. Då spelaren vill spela igen skjuter den “continue”-knappen vilket startar om spelet från början genom att ladda samma scen på nytt (figur 28).

```
public void ReloadScene()
{
    //Restart the game by reloading the scene
    SceneManager.LoadScene(SceneManager.GetActiveScene().name);
}
```

Figur 28. Koden som startar om spelet

5. SLUTSATSER

Tack vare det här projektet har jag blivit mycket bättre på att använda Unity. Alla uppsatta mål uppnåddes. Problem uppstod så klart under utvecklingen men eftersom Unity är en populär plattform är det lätt att hitta hjälp. Att ta på och av VR-headsetet varje gång jag ville testa en ändring försenade utvecklingen jämför med utveckling av spel för vanliga skärmar. Trots det var arbetet behagligt eftersom XR Interaction Toolkit är ett logiskt verktyg för VR-utveckling.

KÄLLFÖRTECKNING

Advanced VR fare. (2021, July 27). *How does a VR headset work and how it can be used*. https://www.youtube.com/watch?v=CoQyhIWNSK8

Arm Blueprint staff. (2022, April 1). *xR, AR, VR, MR: What's the Difference in Reality?* Arm Blueprint. https://www.arm.com/blogs/blueprint/xr-ar-vr-mr-difference

Arm Ltd. (2022). *What is a Gaming Engine?* Arm | The Architecture for the Digital World. https://www.arm.com/glossary/gaming-engines

Bardi, J. (2019, March 26). *What Is Virtual Reality: Definitions, Devices, and Examples*. 3D Cloud by Marxent. https://www.marxentlabs.com/what-is-virtual-reality/

Crunchbase. (2014). *Meta acquires Oculus - 2014-03-25 - Crunchbase Acquisition Profile*. Crunchbase. https://www.crunchbase.com/acquisition/facebook-acquires-oculus-vr--30357598

Franzén, C.-O., & Narse, M. (2009). *GRUND TILL KRAVSPECIFIKATION*. Högskolan I Jönköping. https://www.diva-portal.org/smash/get/diva2:209643/FULLTEXT01.pdf

Gajsek, D. (2022). *Overview of VR controllers: The way of interacting with the virtual worlds*. https://www.circuitstream.com/blog/vr-controllers-the-way-of-interacting-with-the-virtual-worlds

Hejlsberg, A., Torgersen, M., Wiltamuth, S., & Golde, P. (2008). *The C# Programming Language*. Google Books. https://books.google.com/books/about/The_C_Programming_Language.html?hl=sv&id=ICe7ea4RscUC

Ishii, H. (2010). *Augmented Reality: Fundamentals and Nuclear Related Applications*. https://www.researchgate.net/publication/241686793

Martin, J. (2014, April 3). *C# Compiler Released As Open Source*. InfoQ. https://www.infoq.com/news/2014/04/roslyn_oss/

Microsoft. (2022). *Install Visual Studio*. https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022

Optics, E. (2011). *Advantages of Fresnel lenses*. https://www.edmundoptics.com/knowledge-center/application-notes/optics/advantages-of-fresnel-lenses/

Sean-Kerawala. (2020). *How inside-out tracking works - Enthusiast Guide*. https://learn.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/tracking-system

Sheldon, R. (2022, August 3). *virtual reality*. WhatIs.com; TechTarget. https://www.techtarget.com/whatis/definition/virtual-reality

Sherrell, L. (2013). Evolutionary Prototyping. *Encyclopedia of Sciences and Religions*, 803–803.

SPIE Europe Ltd. (2022). *Report: emerging optics to displace Fresnel lenses in XR headsets*. Optics.org. https://optics.org/news/13/6/8

Svenska Akademins Ordbok. (2021). *VR*. https://svenska.se/so/?id=193813&pz=7

TerryGLee. (2022). *Overview of Visual Studio*. https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide

Tikkanen, A. (1998, July 20). *stereoscopy*. Encyclopedia Britannica. https://www.britannica.com/technology/stereoscopy

Unity Technologies. (2021). *Package Manager window*. https://docs.unity3d.com/Manual/upm-ui.html

Unity Technologies. (2022a). *Interactable Events*. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/interactable-events.html

Unity Technologies. (2022b). *Introduction to asset store*. https://unity.com/pages/introduction-to-asset-store

Unity Technologies. (2022c). *The scene view*. https://docs.unity3d.com/Manual/UsingTheSceneView.html

Unity Technologies. (2022d). *Unity asset store - the best assets for game making*. https://assetstore.unity.com/

Unity Technologies. (2022e). *Unity - Manual: Unity's interface*. https://docs.unity3d.com/2023.1/Documentation/Manual/UsingTheEditor.html

Unity Technologies. (2022f). *XR Direct Interactor*. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-direct-interactor.html

Unity Technologies. (2022g). *XR Grab Interactable*. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-grab-interactable.html

Unity Technologies. (2022h). *XR Interaction Toolkit*. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.2/manual/index.html

Unity Technologies. (2022i). *XR Origin*. https://docs.unity3d.com/Packages/com.unity.xr.core-utils@2.1/manual/xr-origin.html

Unity Technologies. (2022j). *XR Ray Interactor*. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.0/manual/xr-ray-interactor.html

Valve Corporation. (2022). *SteamVR*. https://store.steampowered.com/steamvr

Vectornav. (2022). *What is an Inertial Measurement Unit?* VectorNav. https://www.vectornav.com/resources/inertial-navigation-articles/what-is-an-inertial-measurement-unit-imu