7.2 Technisch verslag van de stageopdracht

7.2.1 Het voorbereidend werk

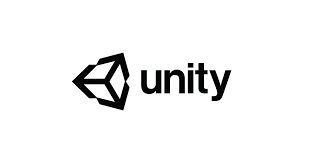
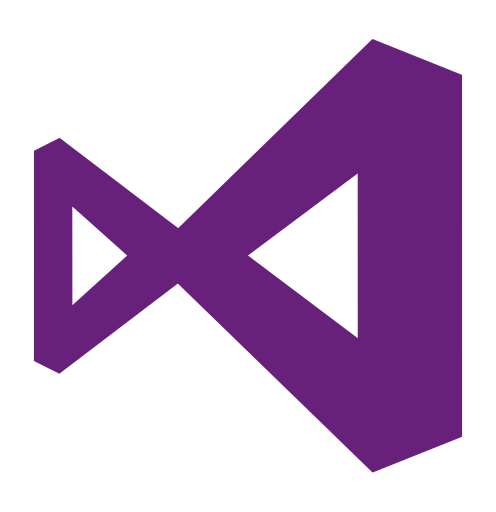
7.2.1.1 De game

Tijdens mijn eerste dag werd er besloten wat mijn stageopdracht concreet zou zijn. Dit gebeurde tijdens een kleine vergadering bij aanvang van de stage. Mijn stageopdracht werd een Virtual Reality parachute simulator. Virtual Reality is zoals de naam al verklaart een virtuele realiteit. Deze kan je betreden aan de hand van een VR-bril met bijhorende VR-controllers. Tijdens mijn stage maakte ik gebruik van de HTC Vive. Dit is een Virtual Reality bril van het merk HTC.

In de game begin je hoog in de lucht staand in een vliegtuig. De deur van het vliegtuig staat open. Je kan pas uit het vliegtuig springen als je op de startknop drukt. Eenmaal gedrukt op de startknop kan je uit het vliegtuig springen (of gewoon blijven staan) en moet je je navigeren doorheen ringen in de lucht. De game eindigt als je landt op het daarvoor voorziene bootje, het water of het terrein. Als je kan landen op het bootje krijg je je score te zien (aantal ringen waar je doorheen vloog), samen met de keuze om het spel te herstarten of om het spel af te ronden.. Als je op het terrein of het water land krijg je als melding dat de missie gefaald is en dan krijg je eveneens de keuze om te herstarten of het spel af te ronden.

7.2.1.2 Programma’s

De nodige programma’s om de opdracht te realiseren:

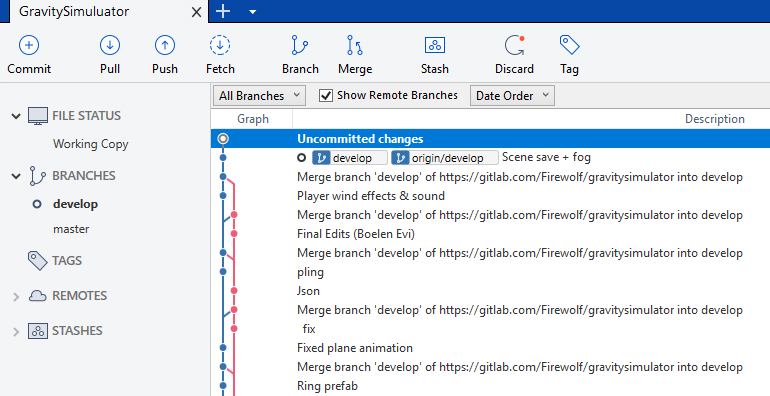
* Unity
* Visual Studio
* Gitlab
* SourceTree
* SteamVR

Unity werd gebruikt als game engine. Dit betekent dat de game die ik zal ontwikkelen in deze engine gemaakt zal worden. Unity ondersteunt scripts in de programmeertaal C# en C++.

Scripts zijn lijsten van commando’s die uitgevoerd worden door een script-engine. Scripts worden gebruikt om processen of acties te automatiseren.

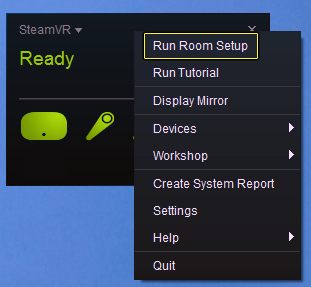
Visual Studio is de ontwikkelaarsomgeving waar scripts in ontwikkeld worden. Visual Studio biedt een mogelijkheid om programma’s te ontwikkelen in verschillende programmeertalen. Enkele van de ondersteunde programmeertalen zijn C, C++, C#, Python en Node.js.

Gitlab is een platform dat je de mogelijkheid geeft om samen te programmeren. Je kan projecten delen met elkaar aan de hand van een link. Je upload een programma op Gitlab, en er word een link gecreerd. Deze link kan je ingeven als een nieuw project op de SourceTree. De veranderingen die worden gedaan worden direct toegepast op het programma. Als een andere gebruiker ook aan het programma werkt, kan je SourceTree gebruiken om je veranderingen volledig vast te leggen. Pas als je op de SoureTree “pusht” (je veranderingen opslaat), en natuurlijk als de andere gebruiker deze gepushte dingen ook “pulled” (veranderingen van anderen op halen) worden de veranderingen ook toegepast. In de SourceTree werk je namelijk op de link van Gitlab. Als je pusht komen je aanpassingen dus ook terecht in het eigenlijke project.

SourceTree wordt gebruikt om samen te werken aan projecten en als een kleine backup. Op SourceTree kan je de link die je van Gitlab af haalde (van het programma waar gezamenlijk aan gewerkt word). In dit programma kan je op verschillende afdelingen werken waardoor er geen conflicten ontstaan als er twee programmeurs aan één project werken op het zelfde moment. Als dezelfde code is aangepast kan er gekozen worden tussen welke van de twee veranderingen toegepast zal worden. Zoals ik eerder zei word SourceTree ook gebruikt als back-up. Dit komt omdat je met SourceTree de veranderingen pas toepast (op Gitlab en SourceTree) als je hier toestemming voor geeft.

SteamVR is een Virtual Reality software die word gebruikt als ondersteuning voor de HTC Vive

HTC VIVE

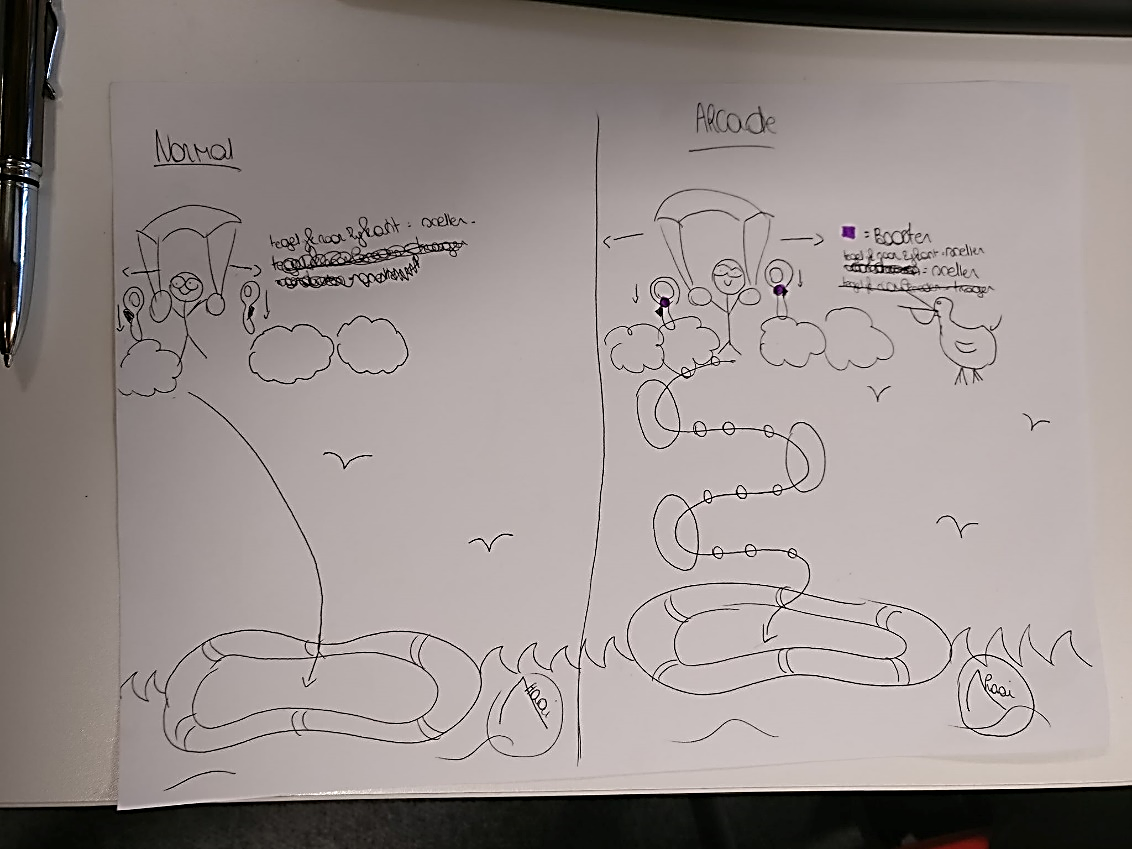
Om een Virtual Reality game te ontwikkelen moet je beschikken over een VR-bril. Deze kan je aan de hand van de net vermelde SteamVR software koppelen aan Unity. De steamVR software zit boordevol met scripts en andere leuke functies. Deze kan je gebruiken om de VR-bril samen met de controllers een functie te geven.

SteamVR zal eerst een “room setup” uitvoeren. Hier kan je je speelterrein markeren en je plaats in dit terrein aangeven.

In deze setup kan je ook aangeven waar de grond zich bevind.

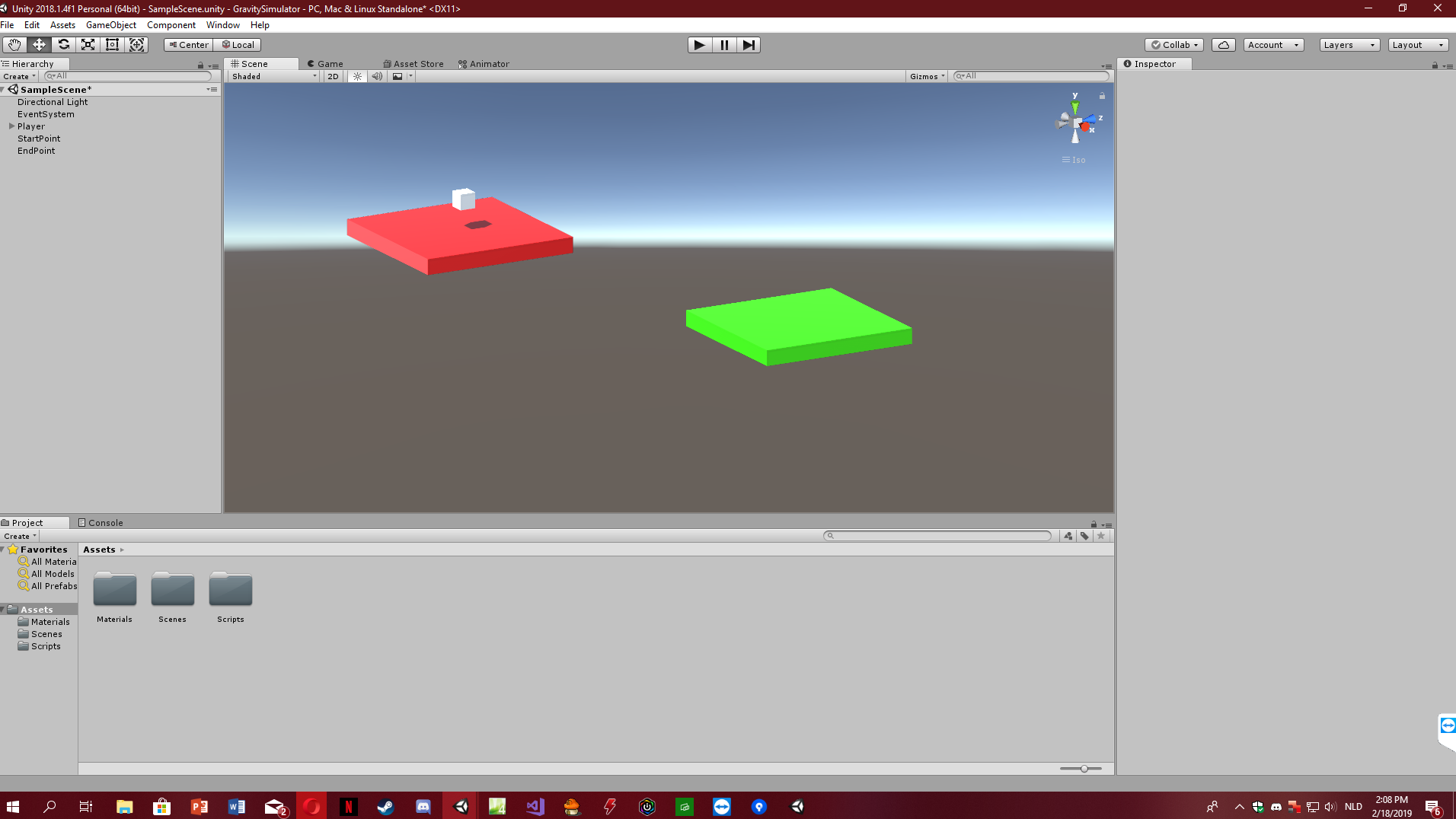


Als de HTC Vive klaar is voor gebruik, is het belangrijk om eerst mijn visie van het spel op papier uit te tekenen. Hieronder vind u een kleine schets van het spel.



7.2.2 De eerste bewegingen

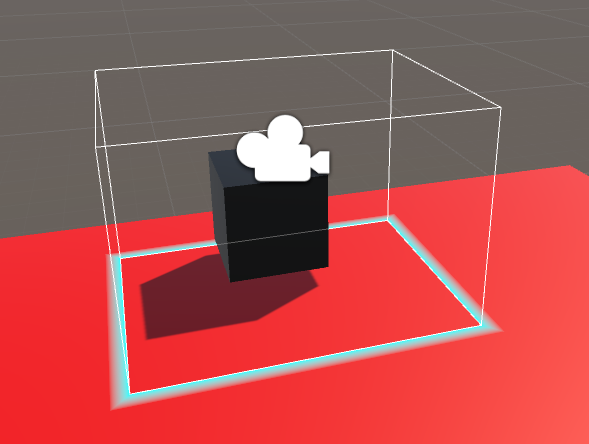
Vooraleer er aan het uiterlijk en het Virtual Reality aspect van de game gewerkt word, is het belangrijk om een beginsituatie te creëren. Voor mijn game bestond deze uit een startplatform (rood), een landplatform (groen) en een speler (wit). Alle objecten in deze game zijn op dit moment vierkanten, de platformen wat groter dan de speler.



De bewegingen van dit object werden gecontroleerd door een klein script. Hierin werd gekeken naar de invoer van de gebruiker. Op basis daarvan werd er een beweging uitgevoerd. Als er op het pijltje naar links gedrukt werd, beweegt de speler naar links, als het rechter pijltje ingedrukt werd, bewoog deze naar rechts.

7.2.3 VR Camera

Om een VR game te ontwikkelen moet je gebruik kunnen maken van een VR camera (die weergeeft wat je ziet in de VR-bril). De camera bevat een ‘head’ camera, een ‘eye camera’ en een vierkante box. De head camera verwijst naar de headset en de eye camera verwijst naar het scherm dat je op de VR bril ziet. Het vierkant is je speelomgeving.



7.2.4 VR Controllers

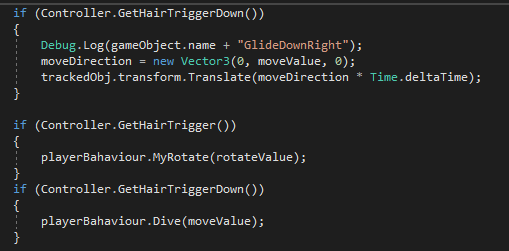
Vervolgens moeten de VR controllers ingesteld worden. Dit moest ik doen aan de hand van een externe website. Deze website is enkel toegankelijk vanuit een editor. De link kan ik hier dus helaas niet in plaatsen. In de editor koppel je acties aan de controller die je dan vervolgens kan oproepen in een script om hier een concrete beweging of actie aan toe te voegen.



Hier ondervond ik wat hinder bij. De SteamVR software was pas vernieuwd. Het systeem dat hierboven vermeld staat hoort bij de nieuwere versie. Hier was zeer weinig documentatiemateriaal over te vinden om mij op te baseren. Hier heb ik bijna een volledige dag opzoekwerk over gedaan. Helaas tevergeefs. Het uitzoeken van het nieuwe systeem ging te lang duren dus er werd me geadviseerd om opzoek te gaan naar een oudere versie van SteamVR. Hier was meer dan genoeg documentatiemateriaal over te vinden op websites zoals het Unity forum, Circuitstream, Youtube en nog heel veel andere. De rede dat ik van dit documentatiemateriaal geen links heb is omdat dit gewoonweg te veel was.

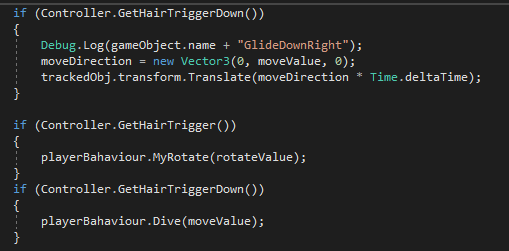
7.2.5 Controller acties inlezen

In de game zal de speler de triggers van de HTC Vive controller gebruiken om te sturen doorheen de lucht. Als deze ingedrukt worden moet de speler dus naar de gevraagde kant bewegen. Dit deed ik aan de hand van onderstaande lijn code.



Deze werd gezet in een ‘if-statement’. Een if-statement word gebruikt om na te gaan of iets aan een bepaalde waarde of actie voldoet. Zo ja zal de code die tussen de accolades er onder staat uitgevoerd worden.

De code die tussen de accolades staat is in dit geval:

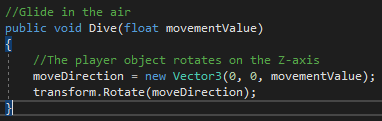


De moveDirection stelt zich voor als een Vector 3. Dit betekent dat hier 3 waardes ingevoerd kunnen worden. De waardes worden toegepast op 3 assen. De eerste waarde op de x-a, de tweede waarde op de y-as en de derde waarde op de z-as. Hier word zowel de beweging op de x-as als die op de z-as op 0 gezet. Dit komt omdat we willen dat de speler zal dalen als er op de trigger gedrukt word. Op de trigger drukken simuleert het trekken aan de parachute zodat je sneller daalt naar de kant waar je aan trekt. De y-as krijgt de waarde van de moveValue. De moveValue staat publiek, dit is een waarde die je kan veranderen in de inspector van Unity. Dit maakt het gemakkelijker om de waardes te veranderen om te testen want je kan deze waardes ook veranderen als je de game opstart in Unity.

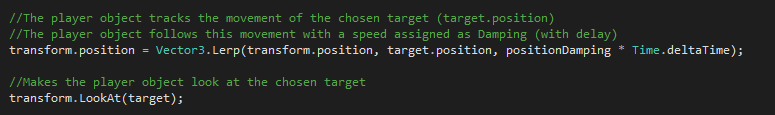
Het trackedObj staat publiek. Hier kan een object in gesleept worden vanuit de hierarchie. De rood gemarkeerde lijn code word uitgevoerd op het trackedObj. In die lijn code zal het object effectief gaan bewegen. De eerder vermelde moveDirection wordt maal Time.deltaTime gedaan. Time.deltaTime is de tijd die voorbij is gegaan sinds de laatste frame. Het is best dat je elke waarde maal Time.deltaTime doet als je een object zal bewegen.

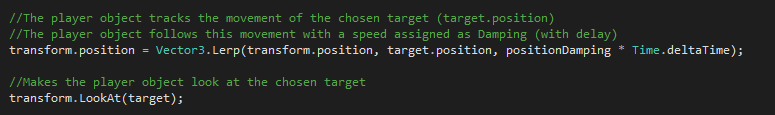
7.2.6 Een vlotte zachte beweging

Nu de speler instaat is om te bewegen word het tijd dat deze ook kan draaien. Dit wordt gedaan met een procedure. Er word een variabele doorgegeven vanuit het hoofdprogramma. Deze wordt ingevuld in de procedure. In dit geval is de variabele movementValue. Hier word de movementValue op de z-as gezet. Dit komt omdat de speler zal moeten roteren naar de gevraagde kant. In de plaats van ‘translate’ te gebruiken zoals in het vorige script maken we nu gebruik van Rotate. Deze laat het object roteren.



De beweging is nu a wat vlotter maar nog niet perfect. Wouter Spaas (collega bij Firewolf Studio’s) hielp me om de beweging optimaal te maken. Hij legde me uit dat ik gebruik kon maken van een ‘Lerp’ functie. Deze zorgt voor een vlotte beweging.



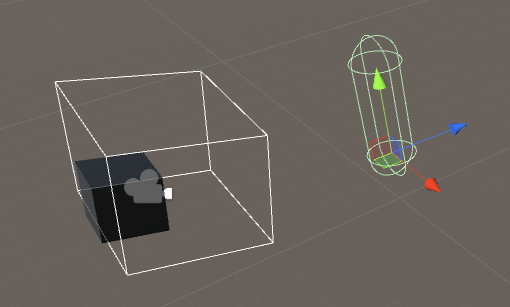


Bovenstaande code zoekt de beweging van het gekozen doelwit (in dit geval de speler) aan de hand van transfor.position. Dit werd eerder ook gebruikt om de speler te laten bewegen. Vervolgens word er gekeken naar de positie van de target (die publiek staat en vanuit de hiërarchie aangewezen kan worden). Als laatst word er een positionDamping toegevoegd. Deze zal zorgen dat de beweging met een vertraging word uitgevoerd. De rede van de vertraging is zodat de beweging niet plots gebeurd. Dit kan de speler doen verschrikken.

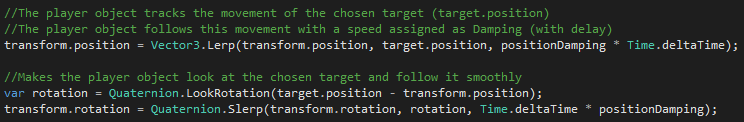
Deze functie zal staan in een FixedUpdate(). Dit is een functie die opgeroepen word aan de hand van elke stap die fysieke objecten maken in de game. In ons geval is dit de speler. Het script word gekoppeld aan de speler.

7.2.7 Een leeg object aan de speler koppelen

De speler heeft nu een VR-camera, controllers die ingelezen worden, kan vooruit bewegen en ook roteren. Tijdens het testen valt me op dat de speler redelijk stroef beweegt. Ik had geen idee goe ik dit moest oplossen. Wouter Spaas schoot me te hulp. Hij liet me weten dat een object dat een ander object volgt veel zachter beweegt. Ik moest dus een object maken, deze koppelen aan de speler en dan het object de bewegingen laten uitvoeren. Ik maakte het lege object aan, koppelde deze aan de speler en koppelde het script dat zorgt voor bewegingen aan het lege object.



Om de speler naar het lege object te laten kijken en mooi te laten roteren in de richting van de beweging van het lege object moest ik mijn bewegingsscript aanpassen.

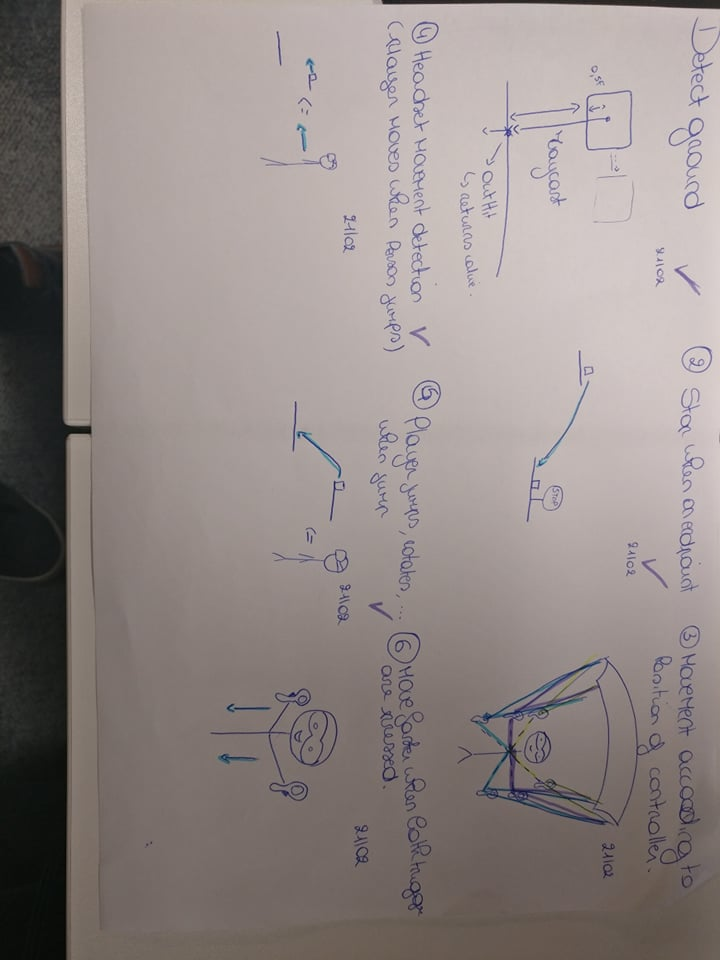


De eerste regel (niet //) van de hierboven geselecteerde code zorgt er voor dat er een nieuwe variabele aan gemaakt word die de waarde van de positie van je target (gevolgde object) verminderd met de waarde van jouw positie om een vaste minimum afstand van jou en je gevolgde object te berekenen. De regel daar onder geeft aan dat jouw rotatie mee gaat met de rotatie van het object met de nodige vertraging (positionDamping).

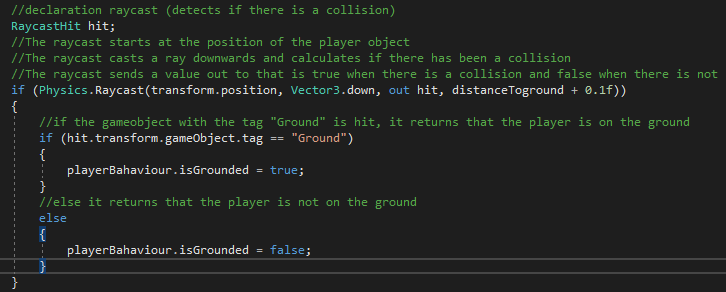
7.2.8 De grond detecteren

Om de grond of andere oppervakken te detecteren moet je gebruik maken van een Raycast. Deze lost het probleem op dat de speler eerst door de grond zou zakken. Dankzij de Raycast word er gedetecteerd dat de speler op een voorwerp staat.

7.2.9 De werking van een Raycast

Een Raycast is een functie die je kan gebruiken in je Unity scripts. Deze stuurt een straal vanuit het middelpunt van het object waar het script aan hangt, in de richting die je zelf kan aangeven in de functie. De straal detecteert of er objecten met colliders in de buurt zijn. Een collider zorgt voor botsingen tussen objecten. Als er een collider gedetecteerd word, stuurt de Raycast een waarde ‘true’ terug. Indien er geen collider gedetecteerd word, stuurt deze ‘false’ terug.

In onderstaand script word er gekeken of de collider die geraakt word de tag “Ground” heeft. Indien dit zo is zal er aangegeven worden dat de speler op de grond staat. Deze zal dan niet meer bewegen. Indien de Raycast een object met een andere tag detecteert, zal de speler geen actie ondernemen en gewoon door doen met de actie die uitgevoerd werd.



Om de speler volledig te doen stoppen als deze op een oppervlak met deze tag detecteert gebruik je de volgende lijn code.

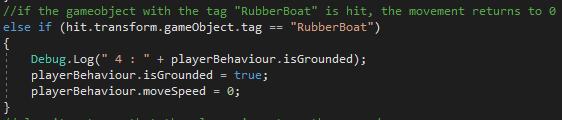
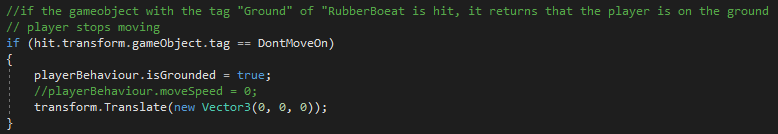
Deze zet de snelheid waar de speler mee beweegt naar 0, waardoor de speler niet meer zal bewegen.

7.2.10 Virtual Reality headset traceren

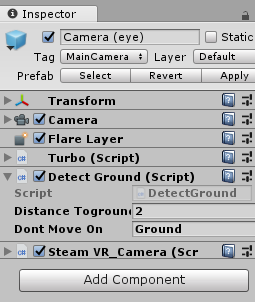
Om de beweging van de speler te traceren moet de headset de effectieve speler voorstellen. Er moet gedetecteert worden wanneer de speler van het platform af springt. Dit word eveneens gedaan met het eerder vermelde Raycast script. Het script word gehangen aan de ‘head camera’. Er word nu gedetecteerd wanneer de headset geen grond meer detecteerd via de Raycast. Als de grond niet meer gedetecteerd word dan word de movespeed terug op de waarde gezet waar deze eerst gelijk aan was en zal de speler gaan bewegen.

7.2.11 De speler kantelt door platformen

Met camera die als speler gezien word, kantelt de speler nu door het landplatform als deze hier op land. Dit werd opgelost door een kleine aanpassing te maken in het script dat de grond detecteert. In de plaats van dat de tag van het object waar de speler op land in de code gezet word, word deze vervangen door een string. Dit een object met als waarde een tekst.

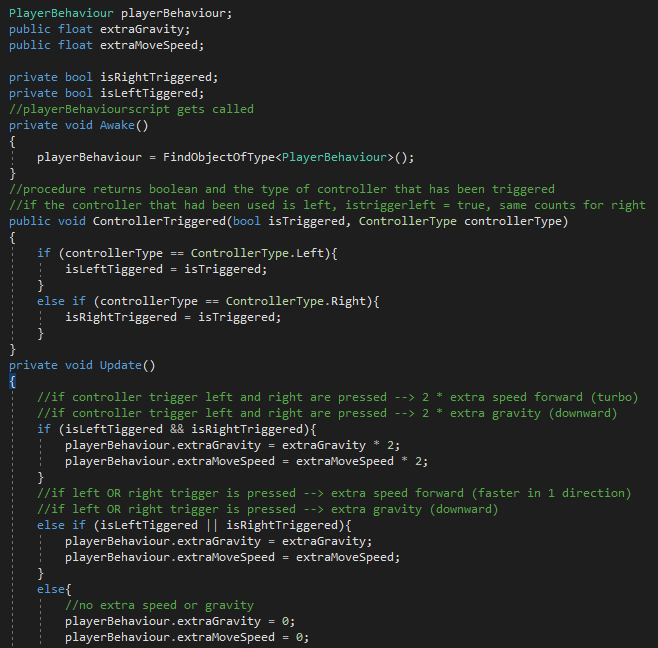


De tekst kan in de inspector veranderd worden. Deze kan op elk moment aangepast worden door een andere waarde in het rood gemarkeerde veld in te geven. Het script zal kijken of de tag van het object waar de speler mee botst gelijk is aan de waarde in het gemarkeerde vak.

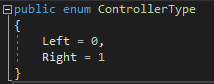


7.2.12 Turbo functie

Nu de speler goed zacht kan bewegen, de headset gedetecteerd word en de oppervlakken waar je op land gedetecteerd worden is het tijd om een extra functie toe te voegen. Dit doe je aan de hand van een script. Het doel van het onderstaande script is het versnellen in de richting van de controller waar de trigger van ingedrukt word. Indien beide controllers de trigger indrukken versnelt de speler naar voor.

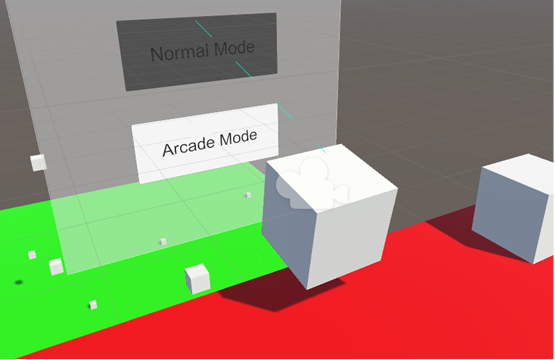
In onderstaande code staat er bij elke lijn uitgelegd wat er gedaan word.

7.2.13 Enums

Om bovenstaand script te kunnen gebruiken is er eerst nog een extra script nodig. Een enum script. In een enum script kan je alle mogelijke waardes die aangeroepen kunnen worden declareren. In ons geval stellen we een enum script op om na te gaan welke controller we gebruiken. In bovenstaand script roepen we ons enum script op in een procedure en erna in een Update(). Een Update() word elke frame uitgevoerd dus elke frame zal gekeken worden of de controllers ingedrukt worden.

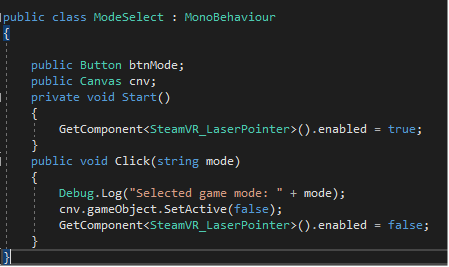
7.2.14 Het menu

Een game moet natuurlijk ook beschikken over een menu. In het menu zou de speler eerst de keuze krijgen om een game mode te kiezen. Maar twee game modes uitwerken in deze korte stageperiode was niet haalbaar. Hier onder kun je het prototype van het menu zien.



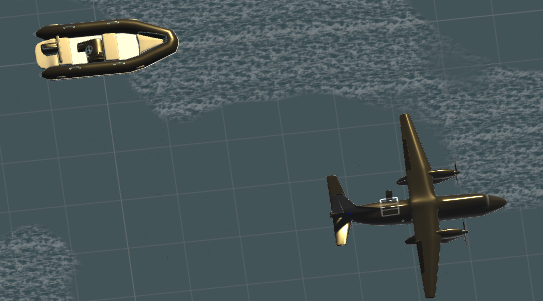
Het menu bestaat uit een canvas met daarop een paneel. Op dit paneel staan twee knoppen. De knoppen kunnen aangeklikt worden met de SteamVR Laserpointers. Deze kan je gebruiken door in de SteamVR map (die vanzelf verschijnt als je met de SteamVR software je headset en controllers hebt gekoppeld aan Unity) in je project opzoek te gaan naar het laserpointer script. Deze koppel je aan beide controllers.

Als er op een knop geklikt word, zal het canvas verdwijnen en de laserpointer uitgeschakeld worden zodat de speler kan beginnen met het spel. Dit wordt aan de hand van de onderstaande gemarkeerde code gedaan.



7.2.15 Het design

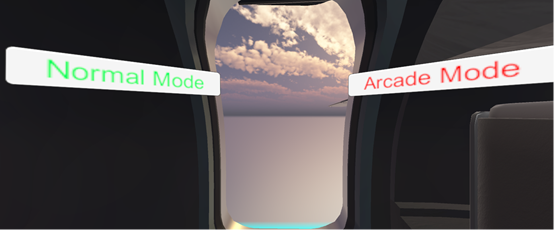
Nu de bewegingen volledig af waren en je de keuze had tussen twee game modes, kon ik de ontwerpen van Gialuca (3D design) toevoegen in het project. Ik kon de designs gewoon kopieerden en in de game plakken. Door deze in de hiërarchie te plakken en niet direct in de scene, blijft de positie waar Gianluca de designs had geplaatst behouden worden. In de plaats van dat de speler begint op een rood oppervlak begint deze nu in een vliegtuig. In de plaats van dat de speler landt op een groen platform, kan deze landen op een bootje. Er werd ook een zee toegevoegd. Het resultaat van deze toevoegingen zie je hier onder.



Een kijkje binnenin het vliegtuig (Dit kan de speler ook zien voordat hij uit het vliegtuig springt):



Het menu werd ook aangepast:



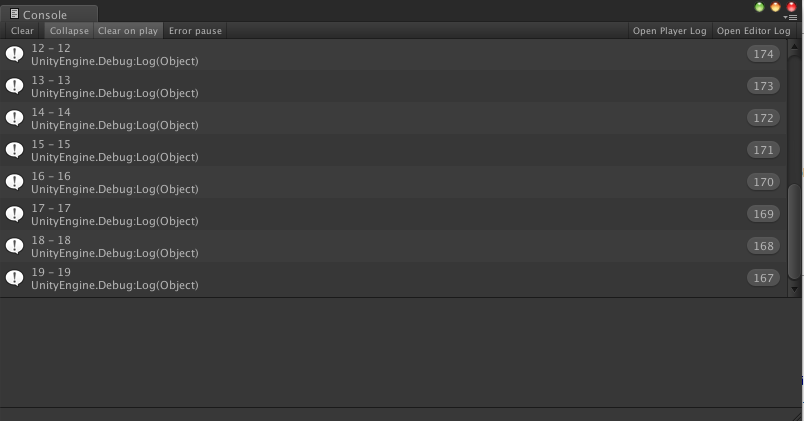
7.2.16 Script ringen

In de arcade mode moet je doorheen ringen vliegen, deze worden opgeteld en de score word afgebeeld op je scherm als je land op de rubberen boot. De ringen worden manueel in de game gezet. Je kan de ringen plaatsen op je eigen gewenste manier.

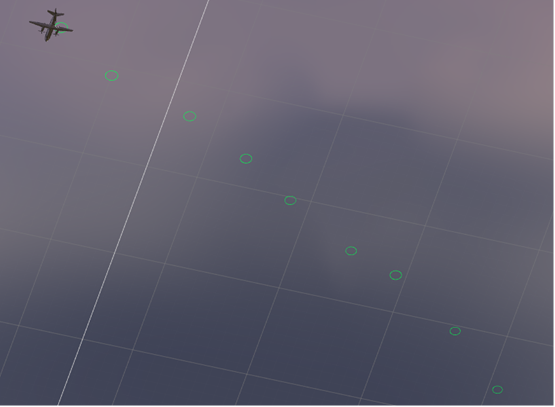
In onderstaand script word er gedetecteerd wanneer de collider van de speler (met de tag ‘player’) in contact komt met de collider van de ring. Indien dit zo is, word de ringCount plus één gedaan. De ringCount staat gedeclareerd in het playerBehaviour script. Dit is het script dat voor de beweging van de speler zorgt. De ringCount staat niet gedeclareerd in het onderstaande script omdat dit script word gekoppeld aan elke ring. Als deze aan elke ring hangt, zal elke ring zichzelf op tellen en niet hoger dan 1 geraken. Als deze vanuit een ander script word aangeroepen zal elke ring opgeteld worden en word de waarde in het andere script bijgehouden.



Een Debug.Log(“”); gebruik je om te testen of alles uitgevoerd word. Zodra de code uitgevoerd word, verschijnt de debug in de console window van Unity.



Om dit script uit te testen plaats je de ringen (groen) op een rechte lijn zoals het voorbeeld hieronder.



7.2.17 Een realistische zee

Om een realistische zee te kunnen gebruiken kocht Firewolf Studio’s een zee in de Unity AssetStore. In de Unity AssetStore vind je 3D ontwerpen, scripts, opmaak, etc. Je hebt zowel gratis ‘assets’ als betalende ‘assets’ in de AssetStore.

Helaas is deze zee enkel bruikbaar in een nieuwere versie van Unity. We zijn dus overgeschakeld naar een nieuwere versie. Hier bleek dat de oudere SteamVR software die we gebruikte niet compatibel was met deze nieuwere Unity versie. We moesten dus gebruik maken van de nieuwere SteamVR software. Hier had ik eerder in mijn stage problemen mee, vandaar de terug schakeling naar de oudere versie. Na een volledige dag opzoekwerk en hulp van collega’s werd er beslist om de zee niet te gebruiken en terug te schakelen naar de vorige Unity en SteamVR versie. Deze dag was leerrijk voor mij. Ik heb die dag niet veel programmeerwerk gedaan, maar veel opgezocht. Dit heeft mij veel kennis bijgebracht.

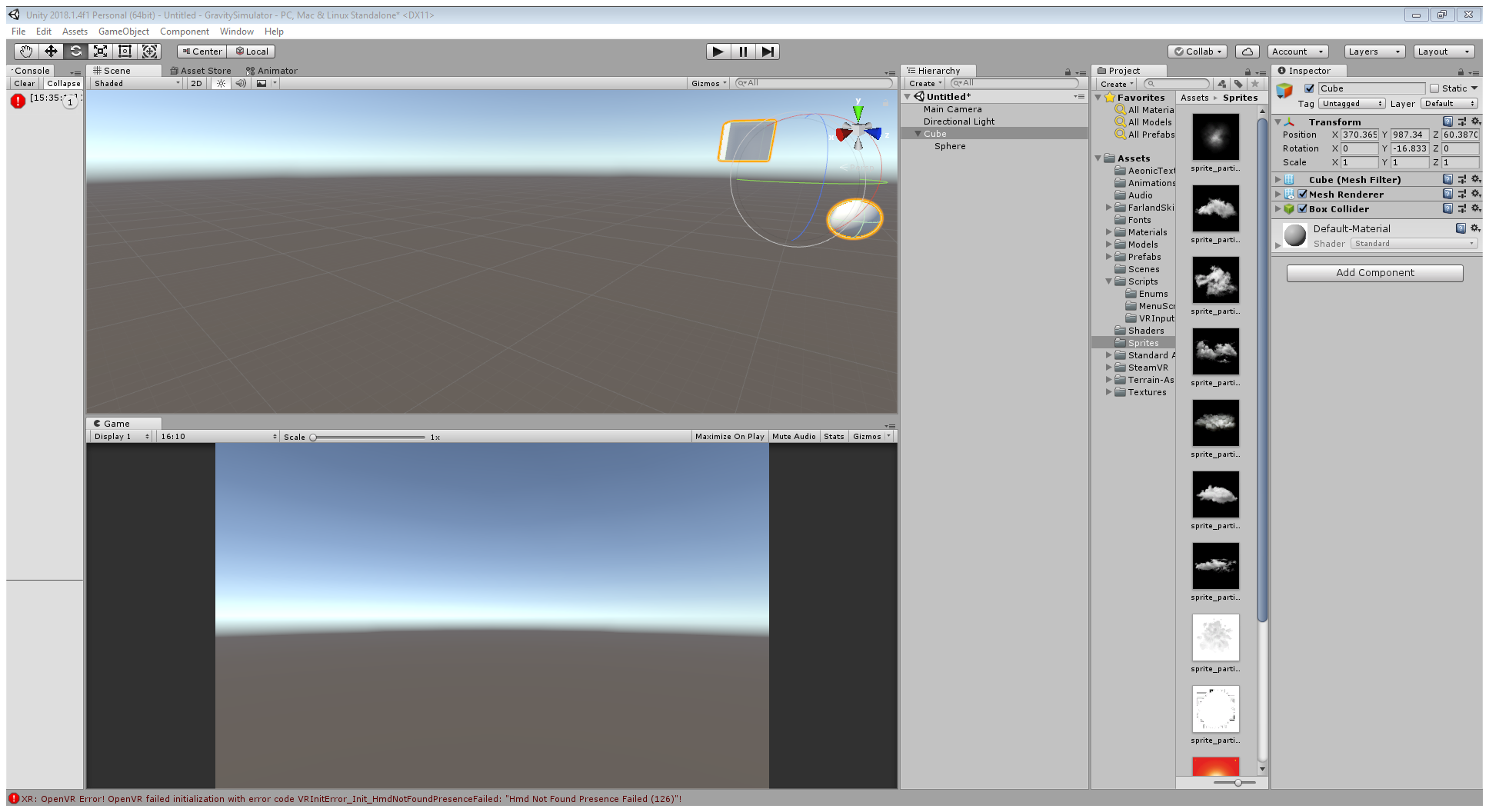
7.2.18 Aanpassen van de bewegingen

De bewegingen zijn helaas nog niet perfect. Dit kan aangepast worden door met de nummers in de scripts te spelen. De draaisnelheid, beweegsnelheid en de zwaartekracht van de speler werd aangepast zodat de bewegingen veel vlotter en realistischer waren voor de speler.

7.2.19 Positie van de ringen

De ringen zet je telkens op de juiste plaats zodat de speler hier net aan kan. Dit is echter niet altijd even makkelijk. Bij mij stonden de ringen te kort op elkaar. Dit moest aangepast worden. De speler moest makkelijk bij de ring geraken, maar hij zou ook bij volgende ringen moeten geraken als hij er ééntje miste. Dit was niet haalbaar bij mijn opstelling van de ringen. Na even uitzoeken hoe ik dit kon oplossen schoot Wouter mij te hulp. Hij begon uit te leggen hoe ik dit het beste aanpakte. Eerst snapte ik zijn uitleg niet, maar na er even over na te denken snapte ik het. Je moet de ‘parent’ van de ring (dit is een leeg object) op de startpositie van de speler zetten. Hierna test je de perfecte afstand van de speler tot de ring zodat de ring niet te hoog of te laag hangt. De speler moet niet constant de turbo functie gebruiken om een ring te kunnen halen. Als de perfecte positie is gevonden kopieer je de volledige ring (inclusief de parent). Deze plak je opnieuw in de game. Nu zet je de lege parent op de positie van de vorige ring. Hierdoor is de afstand tussen de twee ringen hetzelfde als de startpositie en de ring. De ring kan nog geroteerd worden zodat je niet in een rechte lijn naar beneden gaat.

Hieronder vind je een screenshot van hoe dit er uit ziet met normale objecten. In het filmpje zie je terug hoe de rotering van de parent (Cube) de child (Sphere) ook beïnvloed.



7.2.20 Het volledige menu

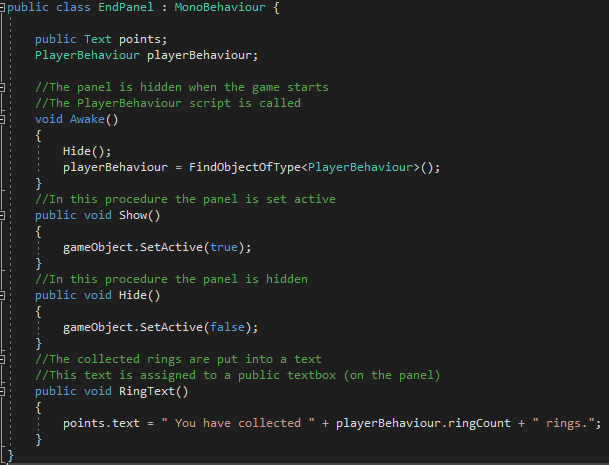
Het menu hangt vast aan de head camera. Zoals eerder vermeld stelt deze de VR-bril voor. Dit betekent dat het menu altijd voor de speler zal verschijnen.

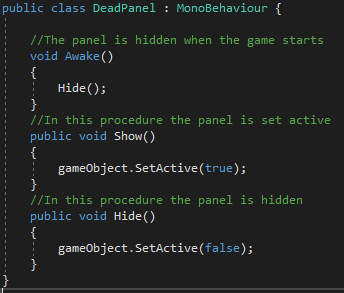
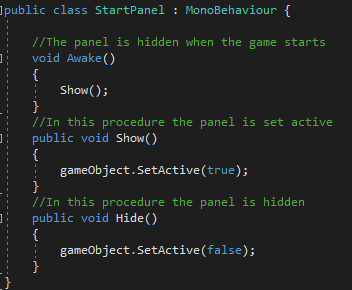
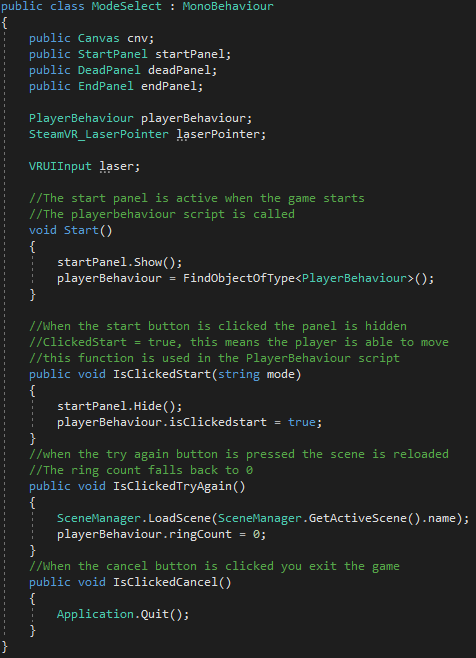


Als de speler op de startknop drukt (met de laserpointers) zal het spel starten en kan de speler van het platform af springen. Als er niet op start gedrukt word, zal de speler niet kunnen springen.

Aan het menu canvas zijn drie panelen gevestigd. Één paneel dat enkel zichtbaar is wanneer je start (het startpaneel). Één paneel dat enkel zichtbaar word als je op het bootje land. Hierop staan twee knoppen. Een cancel knop die je de game laat afsluiten en een try again knop die je terug naar het vliegtuig stuurt. Je score (aantal ringen waar je door ging) word eveneens zichtbaar. Het laatste paneel is enkel zichtbaar als je op het water of de bergen land. Hier krijg je de melding dat de missie niet geslaagd is.

De volgende scripts zijn aan de panelen gekoppeld.



Aan het menu canvas zit onderstaand script gekoppeld: Hier worden de functies aan de knoppen gegeven.

De functies worden aangeroepen in de inspector. Het mode select script word aangeroepen, met de functie IsClickedStart (uit het vorige script). Als er op de knop gedrukt word, zal de IsClickedStart fucntie uitgevoerd worden. In dit geval betekend dit dat het paneel waar de knop op staat zal worden verborgen en de speler kan vanaf dan ook uit het vliegtuig springen.

Nu zijn alle acties, functies, procedures en procedures toegepast. De ringen hangen op zijn plaats. Het aantal ringen word opgeteld. De speler kan zijn parachute besturen om door de lucht te vliegen. De speler kan landen op het daarvoor voorziene bootje. Hij krijgt de keuze om opnieuw te beginnen of het spel af te sluiten. De punten worden getoond. De speler kan ook landen op het terrein of het water en krijgt dan de keuze om opnieuw te beginnen of het spel af te sluiten.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | Naam Boelen Evi Vak Toegepaste Informatica  Klas 6ITN Project Technisch verslag van de stageopdracht | | | | | | |
| E v a l u a t i e d o c u m e n t G e ï n t e g r e e r d e P r o e f | | | | | | | | | |
| **Versie 1** af te geven op: 11/03/2019 | | | | | Paraf ouder | **Versie 2** af te geven op: ………. | | Paraf ouder | |
|  | Tijdig afgegeven | | | |  |  | Tijdig afgegeven |  | |
|  |  |
|  | 1 dag te laat (-10%)  Nieuwe deadline: binnen 24 uur in de uploadzone. Volgende les op papier afgeven. | | | | ……….…. |  | 1 dag te laat (-10%)  Nieuwe deadline: binnen 24 uur in de uploadzone. Volgende les op papier afgeven. | ……….…. | |
|  |  |
|  | Meer dan 24 uur te laat. (-50%). Laatste deadline afspreken met leerkracht:  .............. | | | | ……….…. |  | Meer dan 24 uur te laat. (-50%). Laatste deadline afspreken met leerkracht:  .............. | ……….…. | |
|  |  |
|  | Laatste deadline niet gehaald (score=0) | | | | ……….…. |  | Laatste deadline niet gehaald (score=0) | ……….…. | |
|  |  |
| **E v a l u a t i e** | | | | | | | | | |
| Cijfer | | Commentaar | | | | | | | |
|  | |  | | | | | | | |
| Handtekening leerkracht | | | | Handtekening ouder | | | | |
| .......................................... | | | | .......................................... | | | | |