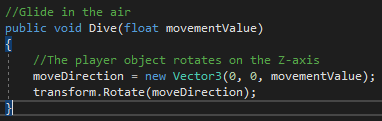
# Dag verslag Woensdag 20-02-2019

## Parachute rotatie

Vandaag begon ik toen ik aankwam op mijn stage direct verder aan de beweging van mijn speler. Aangezien de game een parachutesprong simuleert is het belangrijk om dit dus zo echt mogelijk te laten lijken.

Om een parachute te simuleren is het nodig om de waarde van de Z-as aan te passen. Dit heb ik gedaan aan de hand van een kleine aanpassing in mijn PlayerBehaviour script (het script dat er voor zal zorgen dat de speler kan bewegen).

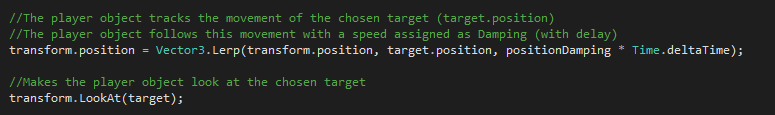


Hier word de richting waar de speler zich naartoe beweegt (moveDirection) aangepast aan de hand van een waarde (movementValue) die publiek ingegeven en aangepast kan worden in de inspector window van Unity. Deze staat op de Z-as omdat de speler rond de Z-as moet roteren om een parachute vlucht te simuleren.

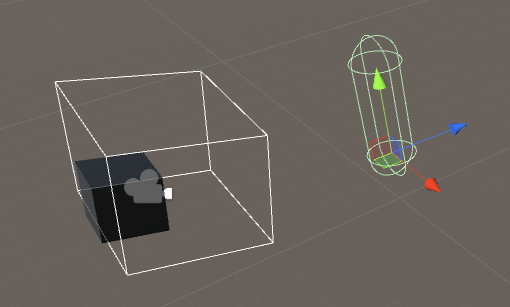
Een publieke waarde geven aan een gegeven doe je aan de hand van “public” voor de variabele te zetten als je deze declareert in je programma.

Aan de hand van deze toevoeging in het PlayerBehaviour script werkt de rotatie.

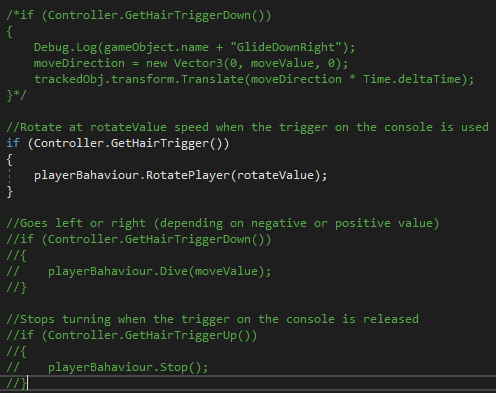
## Probleem 1:

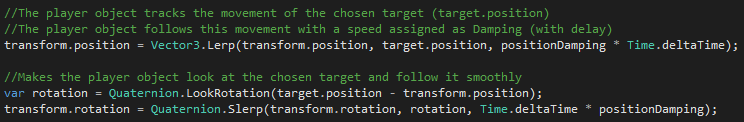
Tijdens het testen merkte ik dat de beweging echter niet vlot loopt. Na veel opzoek werk zonder enige concrete vooruitgang besloot Wouter mij een handje te helpen. Hij vertelde me dat ik een nieuw object in de scene moest aan maken, en mijn speler dit object moet laten volgen aan de hand van een nieuw script. Dus ik zorgde er voor dat er een nieuw leeg object in de scene was waar ik mijn bewegingen op kon baseren. Aan dit leeg object heb ik mijn Player Movement script gekoppeld zodat het lege object de bewegingen maakte. Dit heb ik gedaan omdat de speler het lege object op basis van een script zou gaan volgen, dus deze zelf geen handelingen meer zou doen.

Hierboven zie je de dan ook hoe ik er voor heb gezorgd dat mijn speler het lege object zal volgen. De transform.position staat voor de verandering van de positie van de speler. Deze word gelijk gesteld aan de positie van het target (hetgene dat gevolgd word). De positionDamping zal er voor zorgen dat het object waar dit script aan gekoppeld is het andere object zal volgen met een kleine vertraging (of groot als je dit zo instelt). \* Time.deltaTime zorgt er voor dat de tijd genormaliseerd word. Op de volgende pagina vind u een screenshot van hoe het de speler het lege object volgt in de lucht.



## Probleem 2:

De rotering werkt nu goed omdat de speler een ander object volgt. Het leeg object heeft nu stroeve bewegingen terwijl de speler er heel vlot achter aan beweegt. Vanwege een aanpassing van het script is er een probleem opgedoken met de camera, deze keert om bij de bewegingen. Voor een vloeiende beweging en het exact volgen van het object op de juiste manier ben ik op zoek gegaan naar een oplossing op het internet. Hier heb ik heel wat tijd in geïnvesteerd gehad. Wouter merkte dat ik hier redelijk lang over deed en besloot me een duwtje in de juiste richting te geven. Hij keek mijn code nog eens goed na en merkte dat ik redelijk veel overbodige code had staan in mijn script. Deze werden er uit gehaald aan de hand van commentaarlijnen.

Vervolgens gaf hij een kleine hint door te zeggen dat ik met een Slerp functie zou moeten werken.

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 jou positie om een vaste minimum afstand van jou en je gevolgde object te berekenen. De regel daar onder geeft aan dat jou rotatie mee gaat met de rotatie van het object met de nodige vertraging (positionDamping).