Eindverslag afstudeerwerk

Kusomari

17 augustus 2017

**Auteur** : Jiska Baeten

**Schoolpromotor** : Wim van WeyenBerg, Pieter Jorissen



Opleiding Multimedia en Communicatietechnologie

|  |  |  |
| --- | --- | --- |
|  |  |  |

Inhoudsopgave

[Omschrijving 1](#_Toc488756734)

[Het idee 1](#_Toc488756735)

[Concept 1](#_Toc488756736)

[Doelgroep 2](#_Toc488756737)

[Technische overzicht 3](#_Toc488756738)

[Technieken en tools 3](#_Toc488756739)

[Opbouw eindwerk 3](#_Toc488756740)

[Screenshots 7](#_Toc488756741)

[Overzicht overwonnen problemen en oplossingen 16](#_Toc488756742)

[Conclusie 19](#_Toc488756743)

[Wat geleerd? 19](#_Toc488756744)

[Zelfevaluatie 19](#_Toc488756745)

[Verbeteringen 19](#_Toc488756746)

[Bronnen 20](#_Toc488756747)

[Algemene bronnen 20](#_Toc488756748)

[Muziekbronnen 20](#_Toc488756749)

Omschrijving

Het idee

Ik wou een pc-game maken waarin ik ongeveer dezelfde spelconcepten had als het bestaande playstation game: katamari. Sinds dit nog niet leek te bestaan, leek het me wel interessant om dit game dan te maken.

Ik wou een variant hierop maken en teruggrijpen naar de roots van het idee zelf. Waarschijnlijk kwam het originele idee van de dierenwereld: een mestkever (beetle) die een bal van mest vooruitduwt.

Hierbij was het plan om een speciale controller te maken die speciaal ontworpen was om de bal realistisch en makkelijk te besturen.



Concept

In katamari speel je met een klein personage dat een bal vooruitduwt en andere voorwerpen die kleiner dan de bal zijn, opraapt. Deze voorwerpen blijven aan de bal hangen en maken de bal groter. Ik heb dit idee overgenomen, maar beperk wel hoeveel objecten er zichtbaar aan de bal kleven om geheugen uit te sparen.

Bij katamari wordt er ook getoond hoe groot de bal momenteel is en wordt er een timer bijgehouden die aftelt. Ik hou ook bij hoe groot de bal is, maar laat de timer optellen, zodat de speler meer dan genoeg tijd heeft om een level te voltooien. Hierdoor kan de snelste tijd bijgehouden worden en kan de speler steeds proberen om zijn eigen score te verbeteren.

Ik heb ervoor gezorgd dat het level pas kan voltooien als je een bepaalde grootte hebt gehaald en daarmee bv. uit het park kan ontsnappen door de omheining op te rollen met je bal. Dus je rolt altijd een bepaald voorwerp op, om het level te voltooien.

Ik dacht ook wel dat enkele hulpmiddelen zouden kunnen helpen, o.a.: een minimap waarop alles dat kleiner is dan je bal, getoond zou worden. Voorwerpen die het level zouden voltooien worden op een andere manier getoond op deze map. Daarnaast heeft de speler ook de mogelijkheid om een omlijning rond de voorwerpen te tonen die kleiner zijn dan de bal, door een druk op een bepaalde toets.

Ik vond het ook noodzakelijk dat je het spel op verschillende manieren van besturing kon spelen. Zo is het spel dat niet enkel speelbaar met de speciale controller, maar ook door de pijltjestoetsen, de nummertoetsen en zelfs de computermuis. Zo kunnen mensen die de controller niet hebben, het spel ook spelen.

Doelgroep

Ik vond dat iedereen het spel zou moeten kunnen spelen. Daarom is er een timer die optelt i.p.v. aftelt.

Technische overzicht

Technieken en tools

(welke heb je zelf gemaakt, welke heb je gebruikt van andere bronnen, vermeld ook duidelijk waar die bronnen vandaan komen)

**Programma’s**

* Unity
* Arduino
* Adobe Photoshop
* Adobe Illustrator (om het font aan te passen)
* Icomoon.io (om het font aan te passen)
* Adobe Premiere

**Gebruikte tools en andere**

* Het Endutt font kwam van <http://www.dafont.com/endutt.font>
* Speedtrees komen van unity zelf (package > environment)
* Loadscreen kwam van <https://www.youtube.com/watch?v=xJQXoG3caGc>
* Textuur gras <http://www.reinerstilesets.de/3dtextures/billboardgrass0002.png>
* Verwijzing naar bronnen te vinden op de laatste pagina.

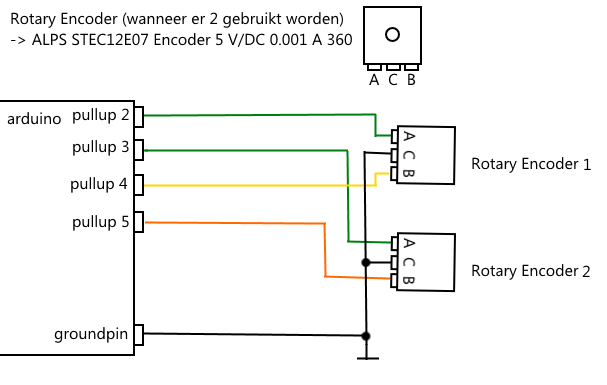
Opbouw eindwerk

1. **Code unity**

Ik heb eerst geprobeerd om alle code (bal rollen, alles kleiner dan de bal oprapen bij colliden…) op te stellen. Later heb ik kleine aanpassingen gemaakt aan deze code zodat ze beter zou runnen. Als de basis al werkte zoals verwacht, dan had ik ten minste al een werkend game.

1. **Controller hardware en software**

Ik wou ook erg graag dat de controller die ik voor dit game maakte, dat deze hardware en software al snel zou werken. Ik was van plan om de onderdelen van een oude muis te gebruiken. Een bal zou rollen tegen een tandwiel dat tussen een lichtdiode en ontvanger stond en zo een interrupt zou opwekken. Pas later kwam ik tot de conclusie dat ik niet wist hoe een muis weet of je vooruit of achteruit bewoog tussen de diode en ontvanger. Mr. Van Weyenberg stelde daarop voor dat een rotary encoder te gebruiken. Ik dacht dat je voor elke rotary encoder 2 interrupts nodig zou hebben (omdat ze gebruik maakt van 2 blokgolven). Hierdoor was het probleem dat ik niet genoeg interrupts zou hebben op de arduino omdat ik 2 rotary encoders wou gebruiken. Ik vroeg raad aan mr. Van Weyenberg en hij hielp mij enorm. Nu meet ik de neergaande flank van 1 input van elke rotary encoder en als deze interrupt optreedt, dan kijken we naar de blokgolf van de andere input en op basis daarvan weet je of de encoder naar links or rechts is gedraaid.



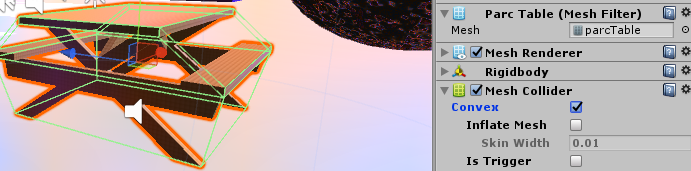
1. **Eerste ontwerp levels**

Daarna besloot ik het level dat ik wou maken, te ontwerpen op papier. Ik wou het eerste level laten beginnen binnenshuis, totdat ik merkte dat als de bal te groot wordt, hij niet meer tussen de deuren past. Daarom heb ik besloten om het eerste level in een park te doen en de keuken (binnenshuis) met tuin te gebruiken als het menulevel.

Plannen inscannen

1. **Eerste modellen en sandboxtesten**

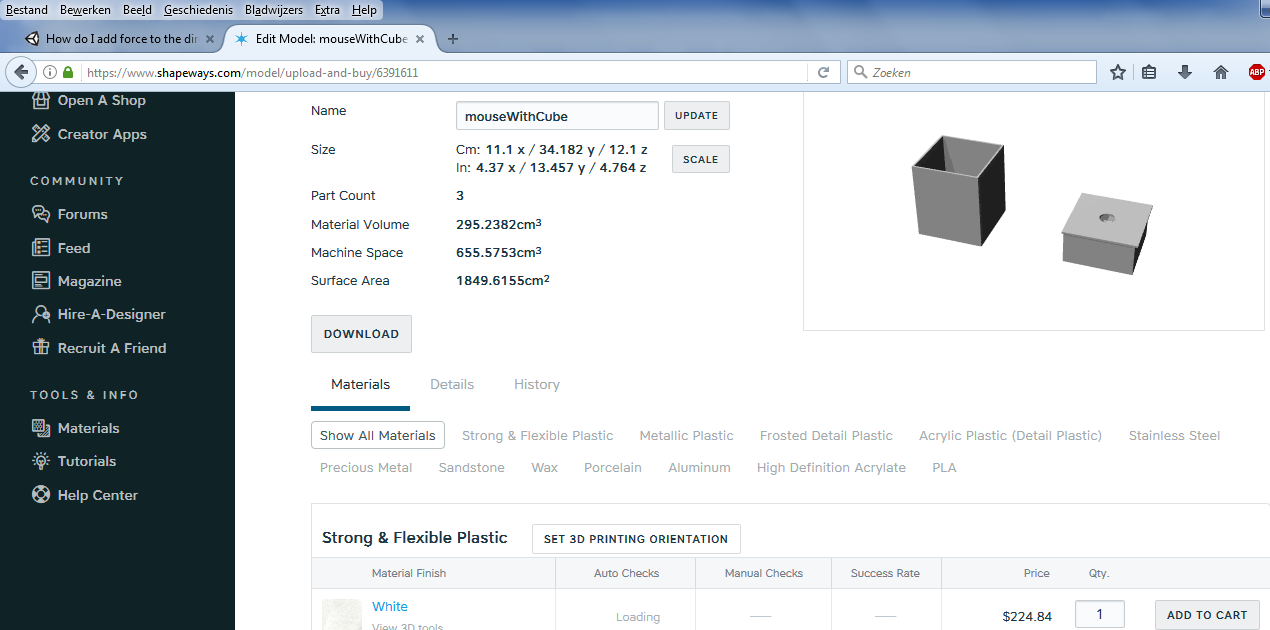
Ik maakte 3D modellen de ik wou gebruiken in het parklevel. Deze gebruikte ik dan om sandbox testen uit te voeren. Zo ontdekte ik dat mesh colliders op convex moesten staan om te kunnen triggeren. Dus om onder dingen zoals parkbanken te rollen (of om erop te rollen m.b.v. een plank), had ik meerdere colliders nodig.





1. **3D printen bescherming controller**

Ik wou op dit moment de buitenste bescherming van de controller maken, door ze te laten 3D printen door ShapeWays. Dit zou te duur uitkomen en ik wou het risico niet lopen dat het dan toch niet allemaal zou passen zoals gepland. Het design van dit is meerdere malen aangepast geweest.



1. **Loading screen**

Heel graag wou ik een loading screen, om ongemakkelijke pauzes van unity niet zichtbaar te maken wanneer hij iets aan het laden is (zoals een nieuw level). Na wat rondzoeken, kwam ik op een youtube-video terecht die een script had gemaakt voor een loading screen. Hij legde daarin uit hoe je zijn script kon gebruiken en dit in een nieuwe scene kon voorbereiden. Daarnaast vertelde hij ook hoe je het loaden kon oproepen (<https://www.youtube.com/watch?v=xJQXoG3caGc>). Ik moest enkel nog een image voorbereiden om te vullen zoals een loading bar, een image (die eventueel geanimeerd was) tijdens het laden en een image die het einde van het laden aantoonde. Omdat het design van de beetle later aangepast is geweest, heb ik dan ook de animatie van het laadscherm aangepast.

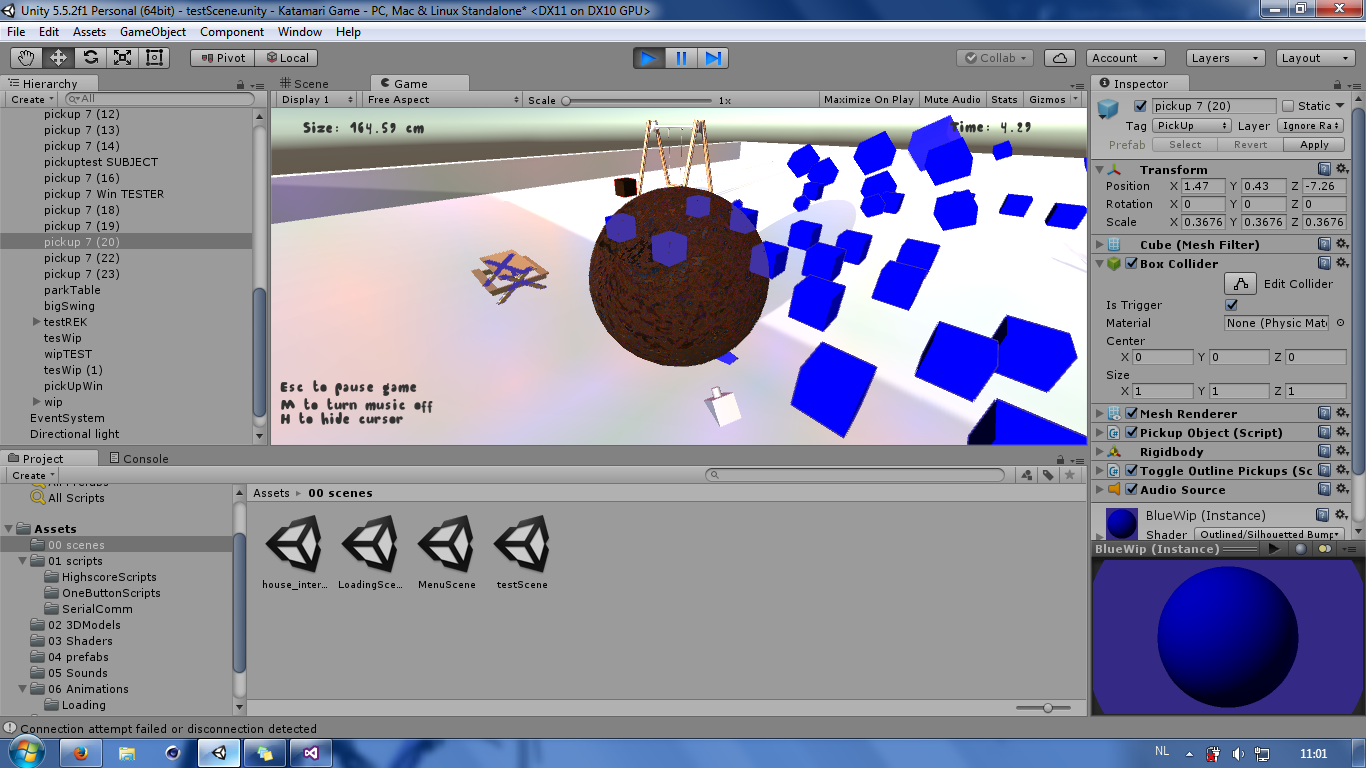
 

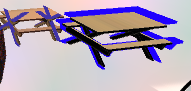
1. **Timer**

Een timerscript is erna toegevoegd. Omdat de tijd bijhouden in seconden niet echt goed was (als je het spel voor een lange tijd zou spelen), heb ik later de conversie naar minuten en seconden veranderd.

1. **Outline voor kleinere objecten**

Ik vond dat het nodig was om alle voorwerpen die kleiner zijn dan de bal, op te laten vallen t.o.v. de rest van de voorwerpen die niet kleiner zijn dan de bal. Het moest niet altijd zichtbaar zijn, behalve als er een knop ingedrukt zou zijn. Ik kwam terecht bij een speciale shader. Hij omlijnde de voorwerpen die kleiner waren, maar het was niet een dikke lijn. Daarom is er later een combinatie gekomen van de shader die ik had gevonden en een toon shader. Het enige probleem dat hierbij kwam, was: voorwerpen met meerdere texturen, konden niet allemaal van shader veranderen. Enkel de maintexture toonde een outline. Dit heb ik op een later moment kunnen oplossen.





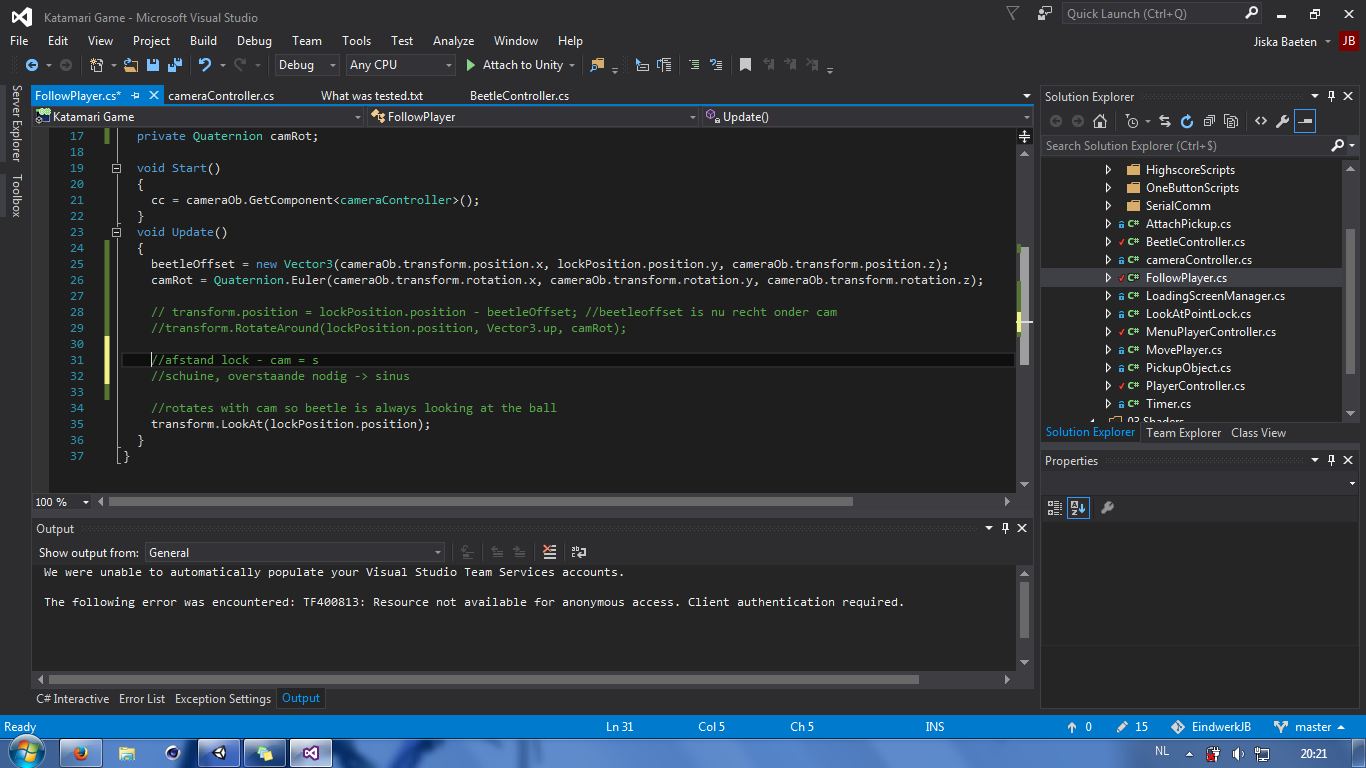
1. **Cursor**

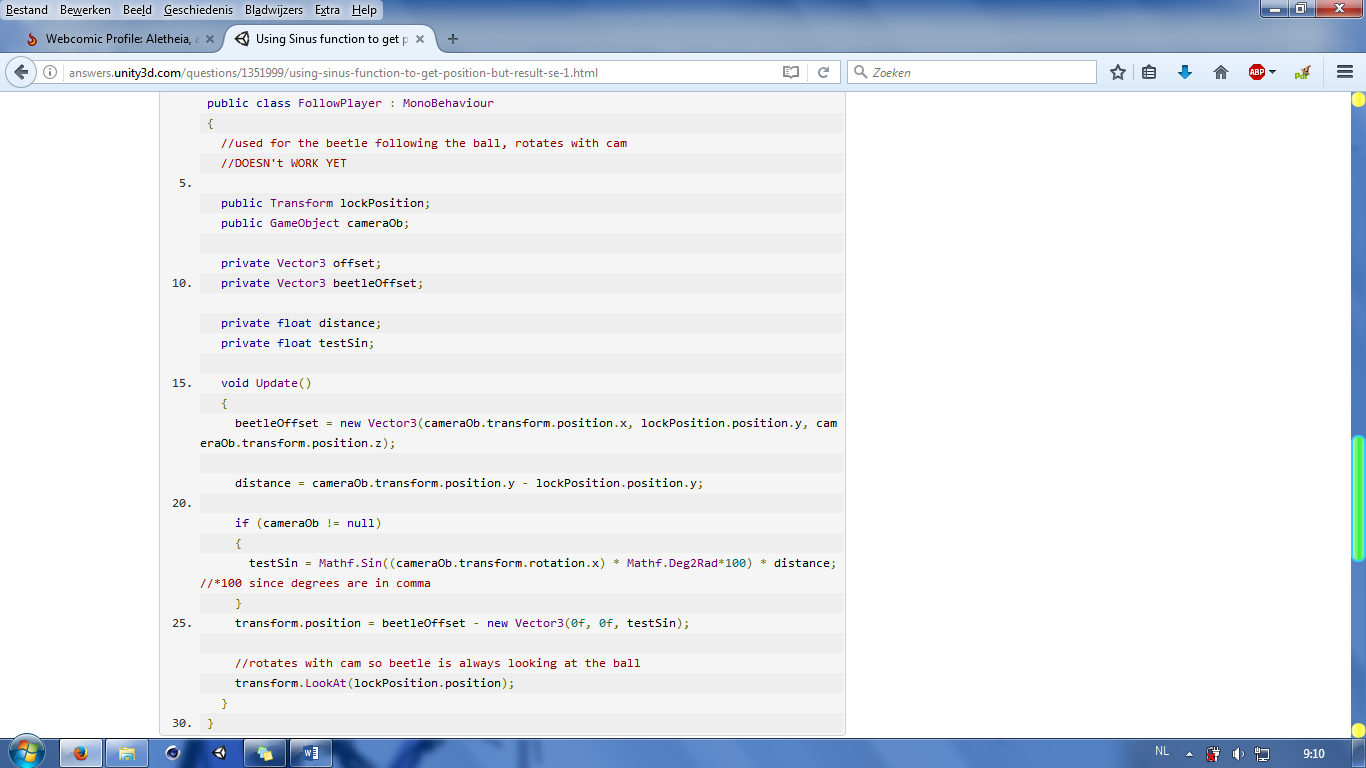
Het spel heb ik wat gepersonaliseerd door een zelfgemaakte cursor te maken. Daarvoor heb ik de ideale afmetingen voor unity cursors opgezocht en dan de image in het kleine vakje gemaakt. Ik dacht dat de image dan nog geconverteerd moest worden naar een .cur-extensie, maar dat bleek dan niet nodig te zijn.

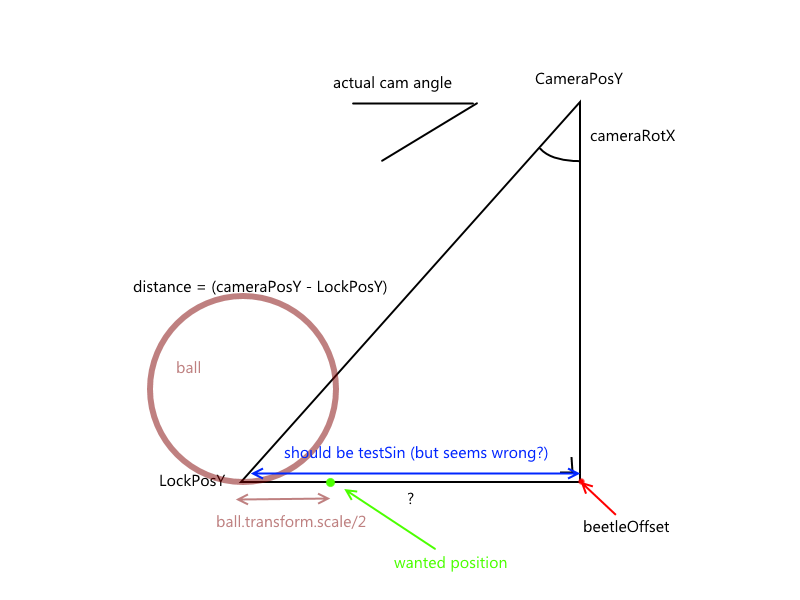
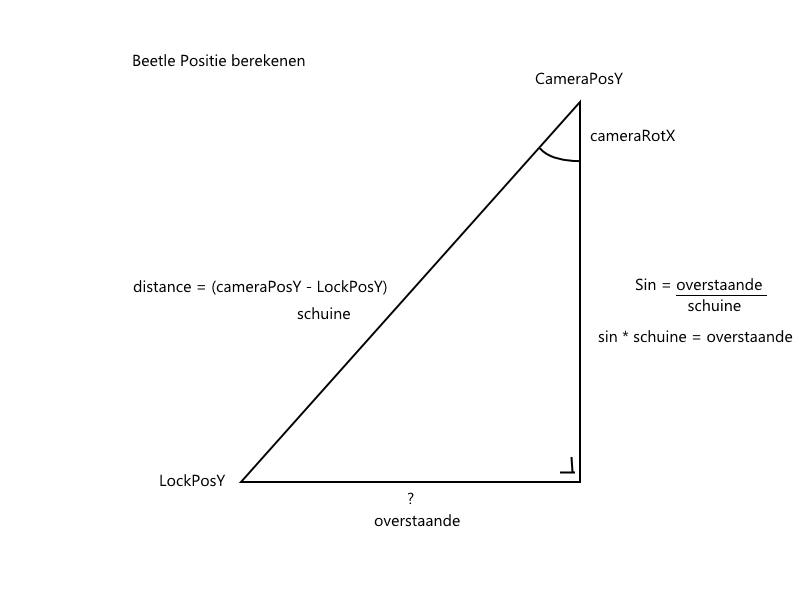
C:\Users\Gebruiker\Documents\GitHub\EindwerkJB\Animations\beetleCursorOutline.png

1. **Beetle volg bal**

De beetle moest de bal kunnen volgen en op dezelfde rotatie als de camera hebben t.o.v. de bal. Hier moest nog een script voor geschreven worden. Ik wou dit wiskundig oplossen, maar dit heeft toch veel langer geduurd om te programmeren dan verwacht.







1. **Achtergrondmuziek**

De levels waren doodstil. Het enige geluid dat ik had geïmplementeerd, was een geluid wanneer iets opgeraapt zou worden. Ook zou het nagaan van copyright een last zijn en de ontwikkeling van het game vertragen, dus ben ik gegaan voor klassieke muziek. Als je deze muziek versnelt, dan lijkt het soms wel of er een klein insect rondloopt.

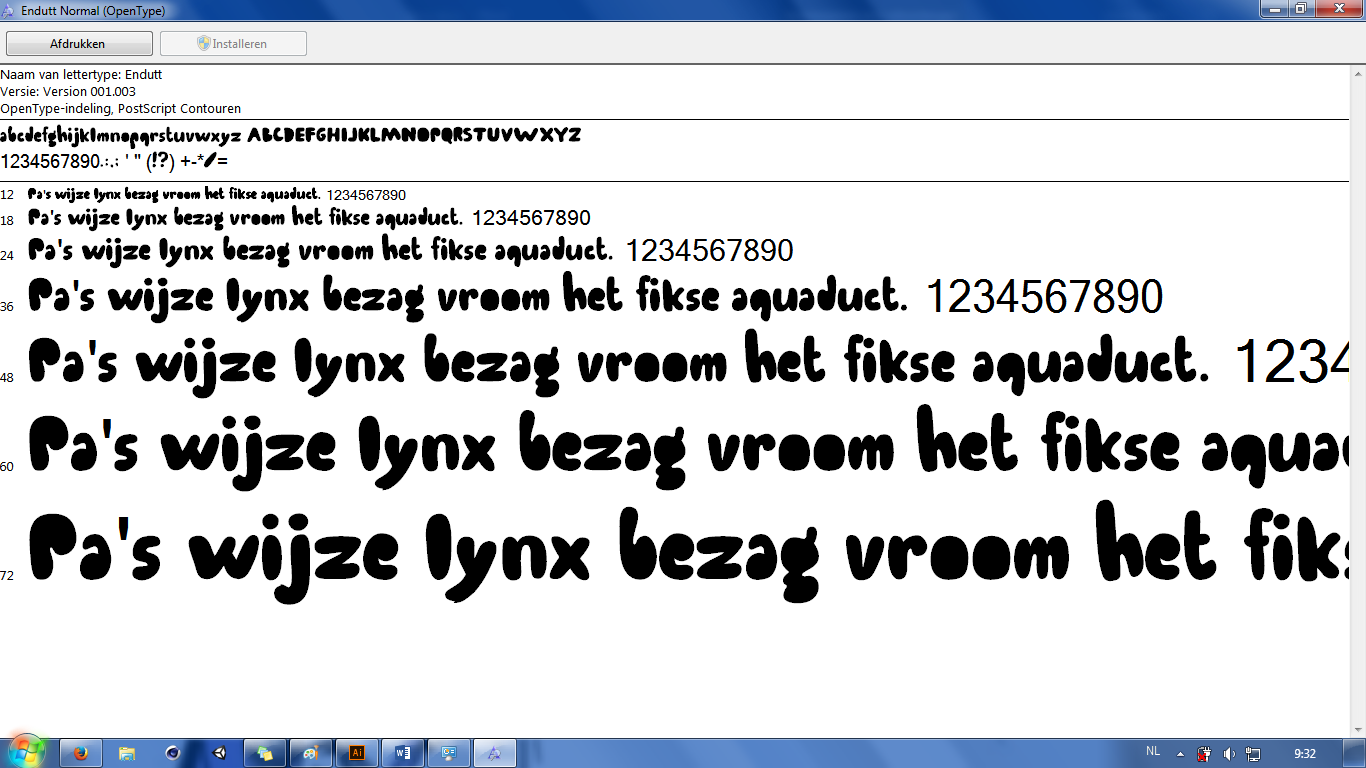
1. **Pauzescherm**

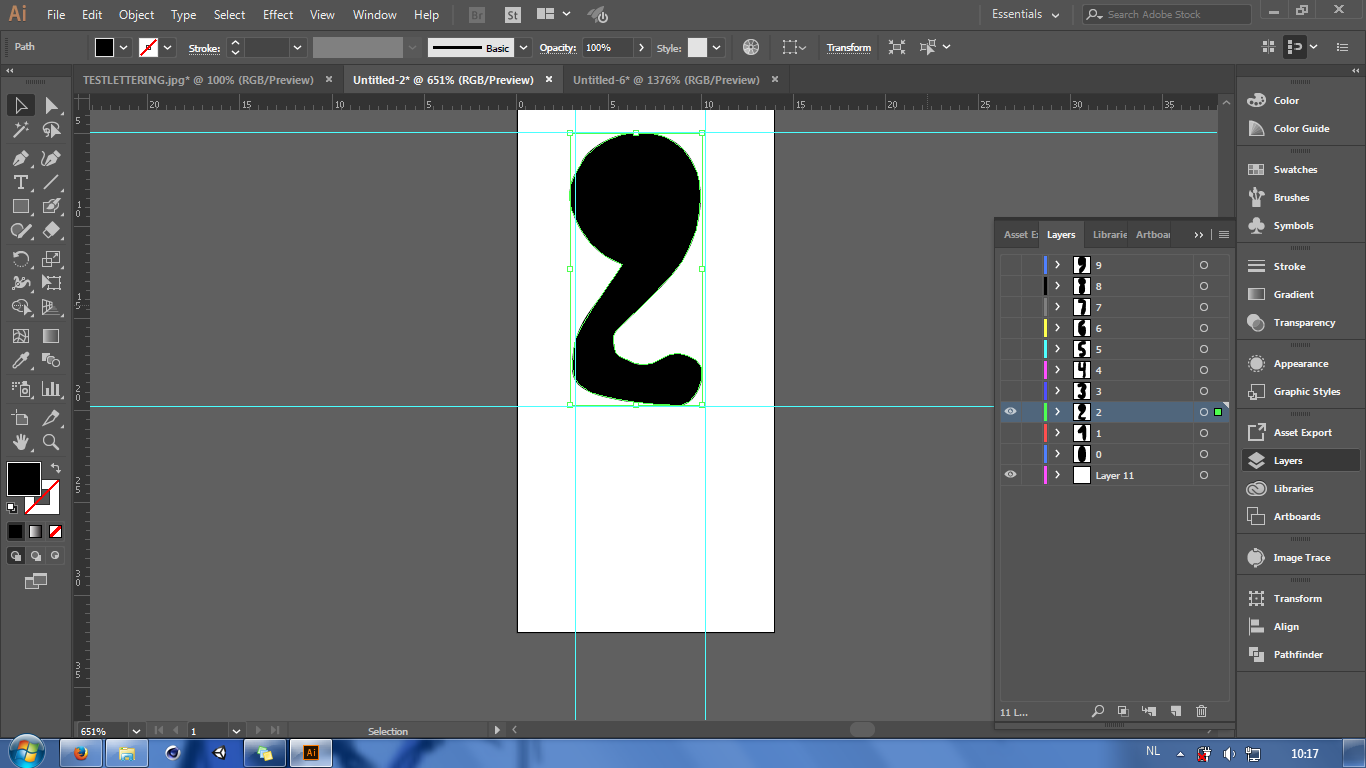
Het was noodzakelijk dat je het spel kon pauzeren, dat je eventueel terug naar level select zou kunnen gaan en dat je het spel kan afsluiten wanneer je maar wilt. Ik maakte een pauzescherm waar dit allemaal mee mogelijk was. Op dit moment heb ik ook sneltoetsen gemaakt waarmee je de cursor onzichtbaar kon maken en de muziek kon uitzetten wanneer je dit wou. Daarnaast heb ik ook een zichtbare indicator gezet die je vertelde of de arduino geconnecteerd was of niet.

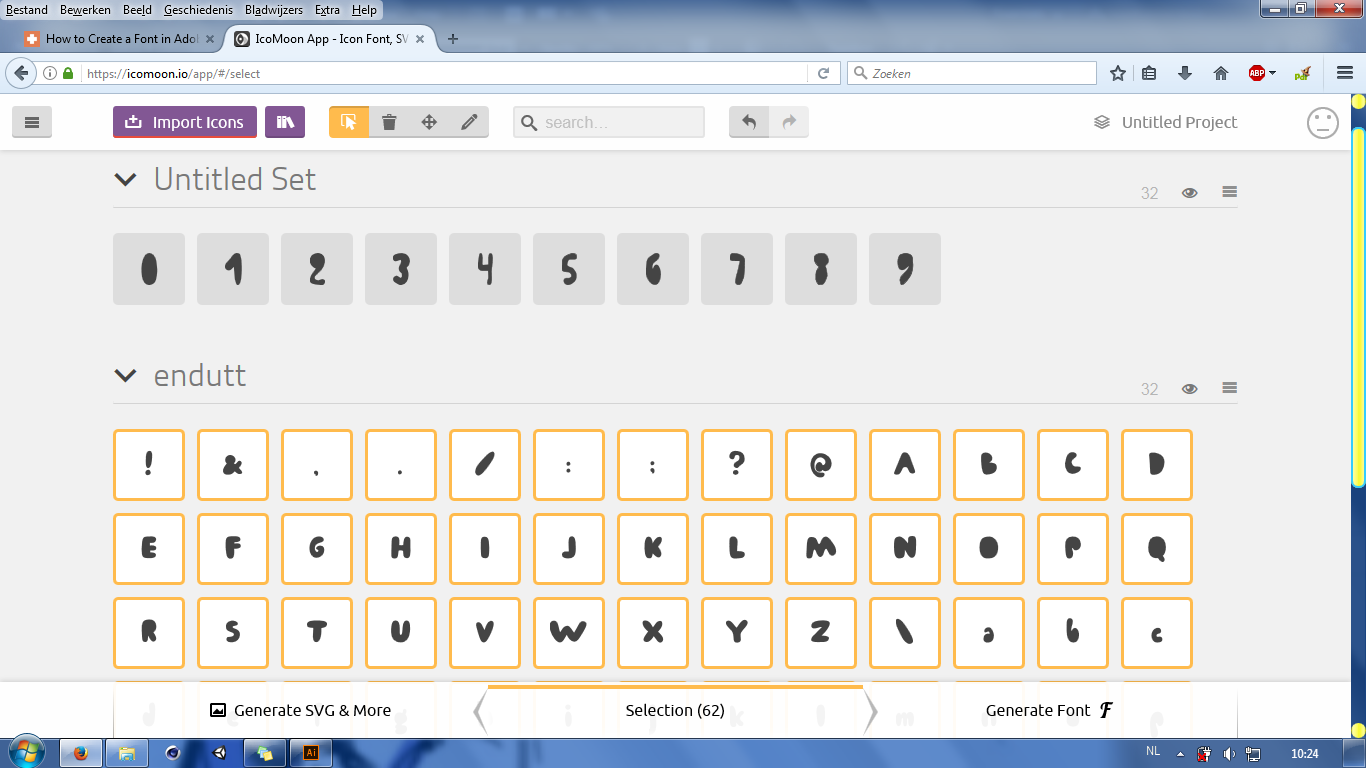


1. **Nummer aan font toevoegen**

Ik merkte plots op dat het font dat ik gekozen had, geen nummers bevatte. Ik was ondertussen al gehecht geraakt aan het font, dus ik besloot om zelf de nummers aan het font toe te voegen. We hadden tijdens een graphics vak geleerd hoe we een font konden maken. Dus samen met photoshop (om snel de vormen uit te tekenen), illustrator (om er vectortekeningen van te maken) en icomoon (om alles bijeen te plaatsen om er een font van te maken) is dit gelukt. Er kwamen wel enkele problemen te boven omdat het niet een genormaliseerd font was (stukjes van de letters vielen er af). Na wat aan te passen in icomoon, was dit opgelost. Ook was het niet makkelijk om een spatie in een font te steken, want een lege ruimte wordt niet geregistreerd.

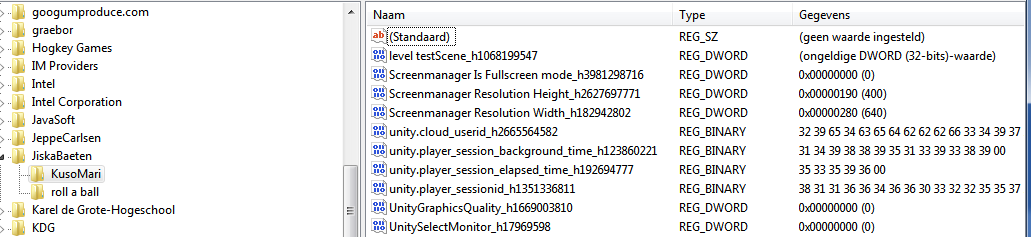






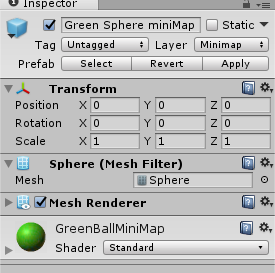
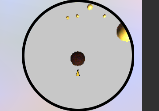
1. **Opslagen en laden van highscores**
2. Om herspeelbaarheid te motiveren, was een implementatie van een highscore noodzakelijk. Ik wou dit niet lokaal in een bestand opslagen, omdat dit dan aanpasbaar was. Daarnaast vind ik het saven op een binaire manier erg omslachtig. Ik zocht een andere oplossing en kwam op lokale keys terecht. Een tutorial van unity zelf, kon mij hierbij helpen (<https://unity3d.com/learn/tutorials/topics/scripting/high-score-playerprefs>). Hierbij merkte ik zelfs op dat je verschillende datatypes in keys kunt opslaan. Normaal wordt dit gebruikt voor playerprefs maar het werkte ook perfect voor een highscore.





1. **Minimap**

Ik vond dat er nog een andere manier moest zijn dat je kon zien of een voorwerp kleiner is dan je speelbal. Ik bedacht dat een minimap nog een interessante aanvulling zou kunnen zijn. Ik wist niet dat het zo makkelijk was (<http://blog.theknightsofunity.com/implementing-minimap-unity/>). Je maakt een tweede camera aan en maskt de vorm eruit die je wilt hebben voor je map. Je maakt ook een render texture aan en plaats deze op de camera. Je maakt aan de voorwerpen die op de map moeten verschijnen, een image/object vast en zet deze op een nieuwe layer ‘minimap’. Nu moet je enkel de culling mask van de camera’s aanpassen naar de lagen die elke camera moet kunnen zien (minimap cam enkel minimap layer, andere cam alles behalve minimap layer).

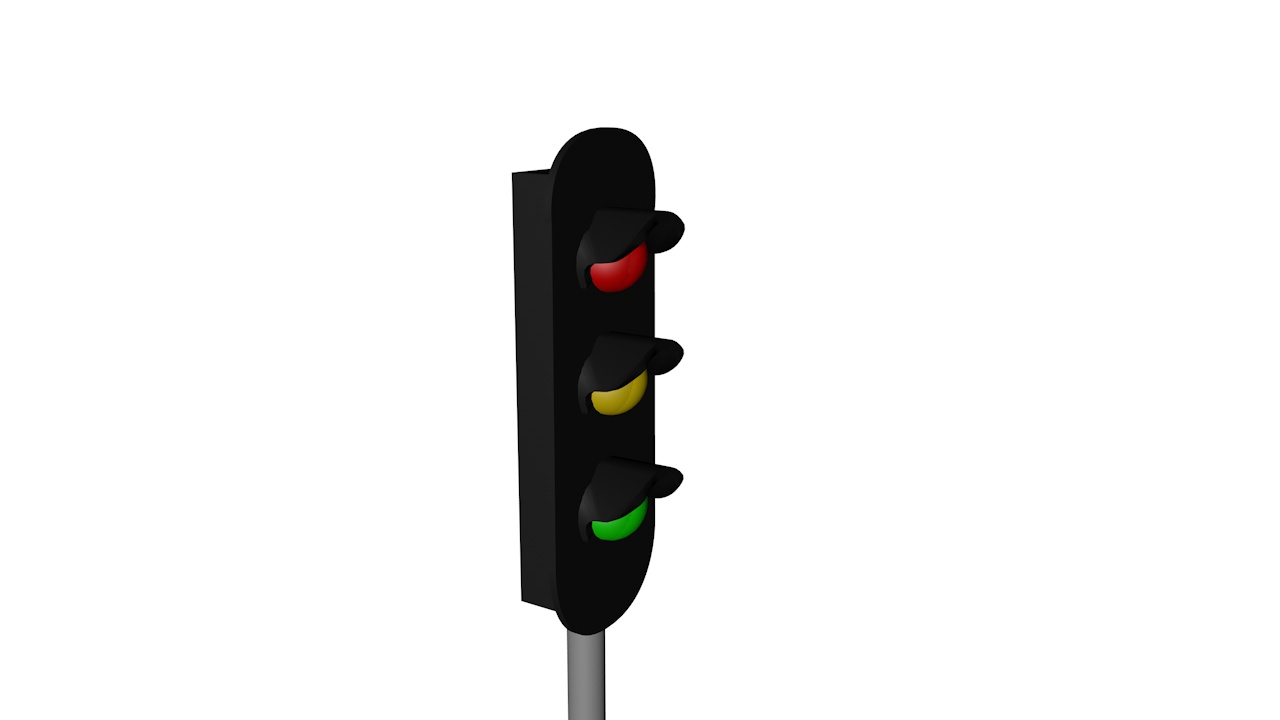
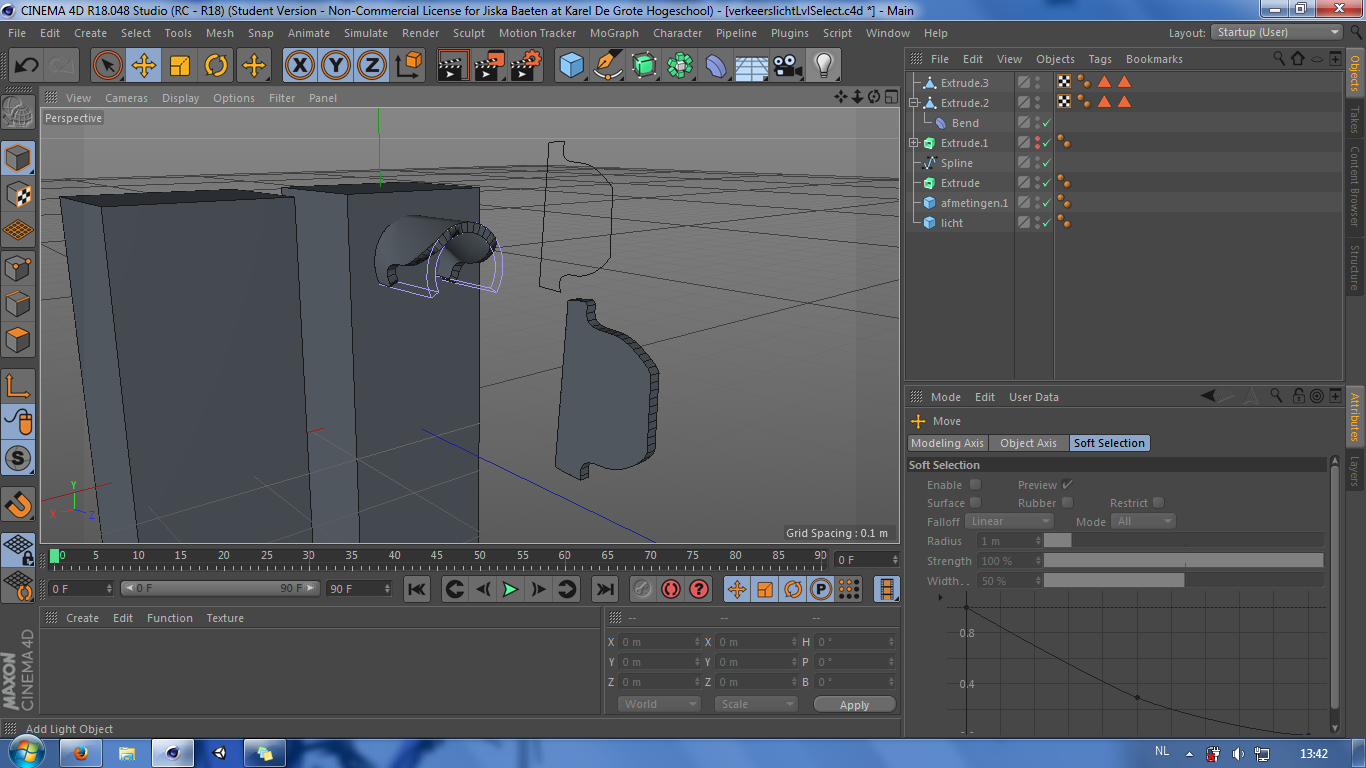


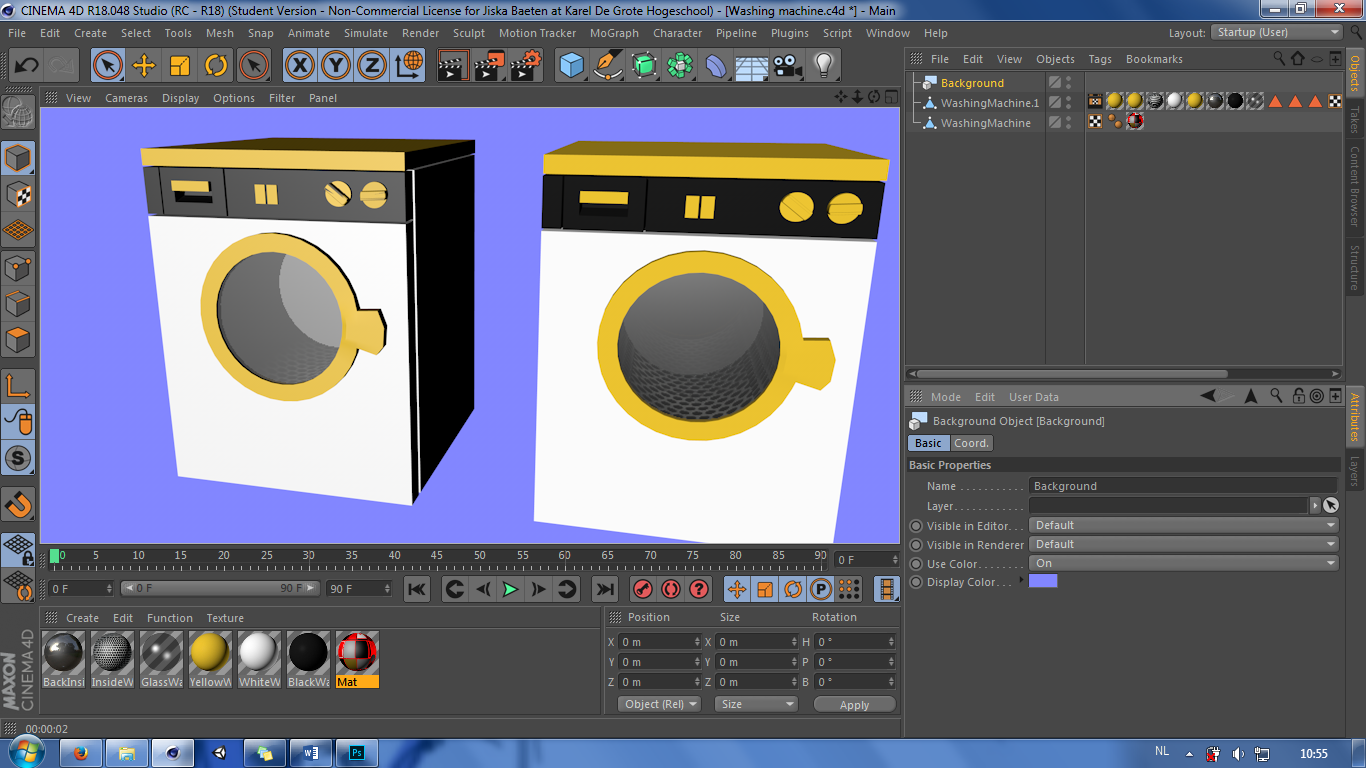
1. **Fullscreen op lagere resolutie**

Omdat mijn laptop niet de sterkste is en toch een fullscreen beeld wou, heb ik opgezocht of dit mogelijk was. Samen met unitydocs, kwam daar als snel een antwoord voor (<https://docs.unity3d.com/ScriptReference/Screen-fullScreen.html>).

1. **Testen i.v.m. baken en nieuwe ideeën om een 3D model te maken**

Het was tijd om de levels in te vullen met 3D modellen. Ik maakte alle modellen die in het menu moesen komen en heb vele tests gedaan met de wasmachine. De textuur zag er nooit uit zoals ik het wou en op dit moment had ik besloten om te proberen om alle texturen te verlagen in aantallen en om de maximale textuur-grootte op 512 x 512 mb zou zetten. Daarnaast vond ik nieuwe manieren op 3D modellen te maken: bedenken hoe je het in het echt met papier zou kunnen maken.



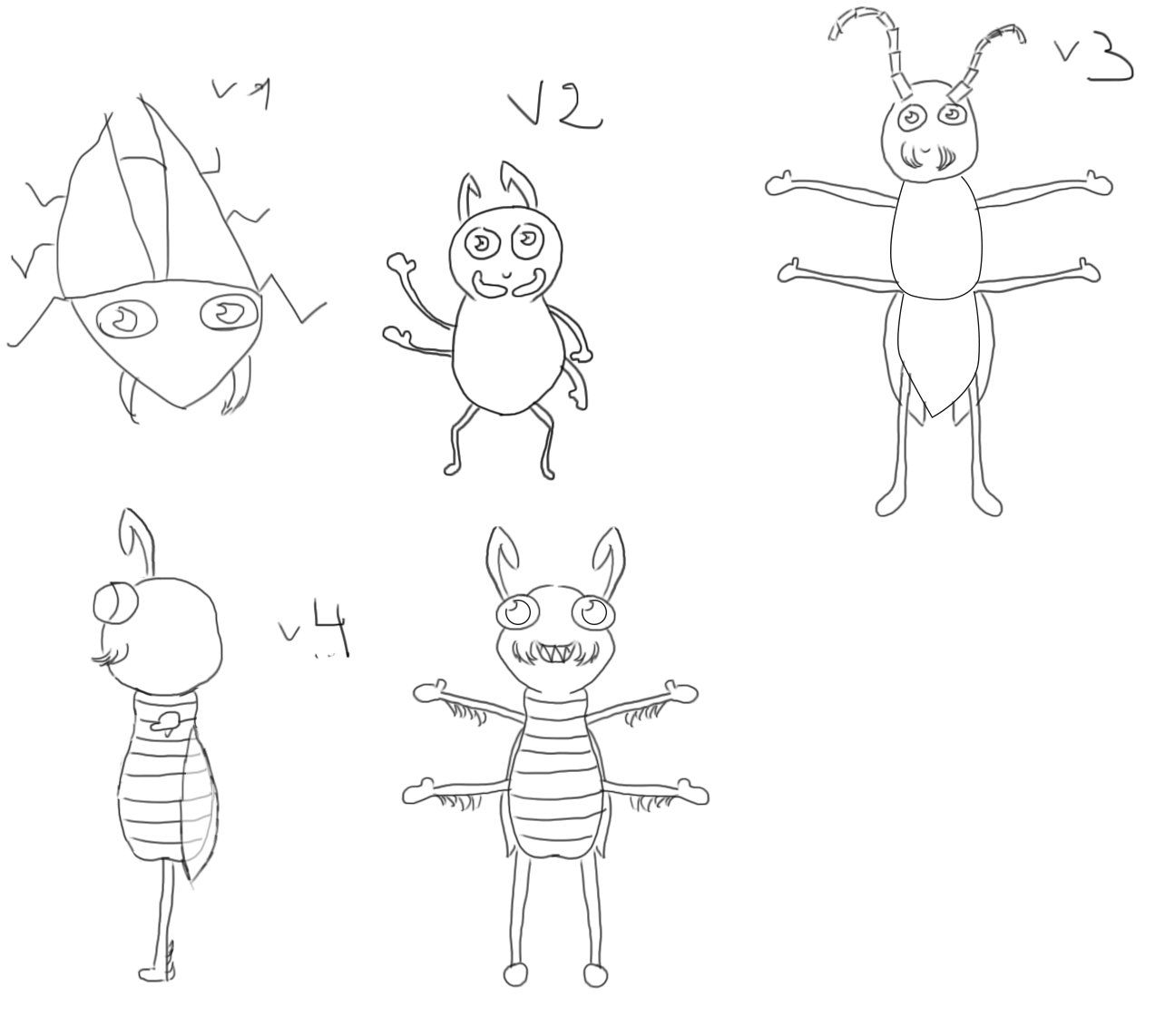






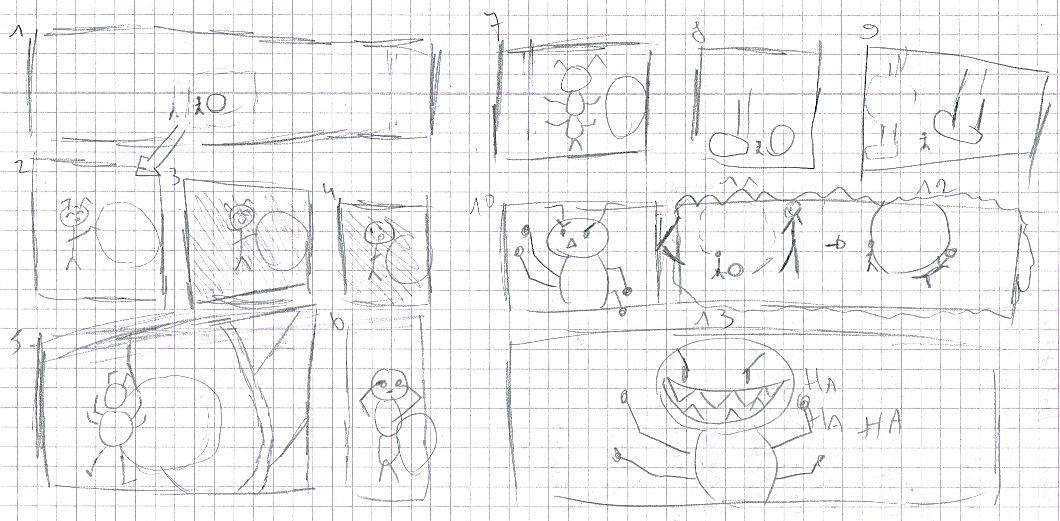
1. **Evolutie design beetle**

Ik vond het design van de beetle toch niet interessant genoeg. Enkele schetsen later had ik een nieuw design waar ik dol op was. Het had iets cartoonachtig en het zag er ondeugend uit.

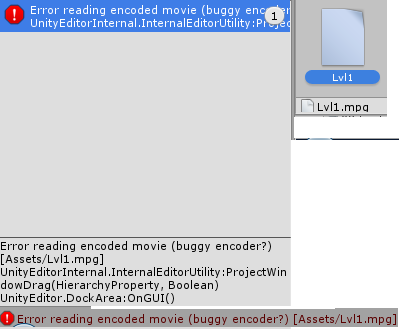


1. **Comic video lvl1**

Er was een motivatie nodig waarvoor de beetle handelde. Dit was het beste en het snelste uit te leggen via een video. Ik was geïnspireerd door een ander spel: Tadpole Treble. Hierin werd het via een stripboek stijl uitgelegd hoe het verhaal verder ging. Eerst heb ik een schets van het storyboard op papier gemaakt, daarna heb ik dit in photoshop uitgewerkt. Elk paneel is apart opgeslagen en daarna geïmporteerd naar premiere. Daar heb ik de gehele comic geanimeerd en de muziek erop geplaatst. Om het in unity te brengen, kon je een gewone texture van bv. een raw image gebruiken om het filmpje af te spelen. Een texture kan geen geluid afspelen, dus ik moest nog wel een audio source invoeren met de muziek van de video.

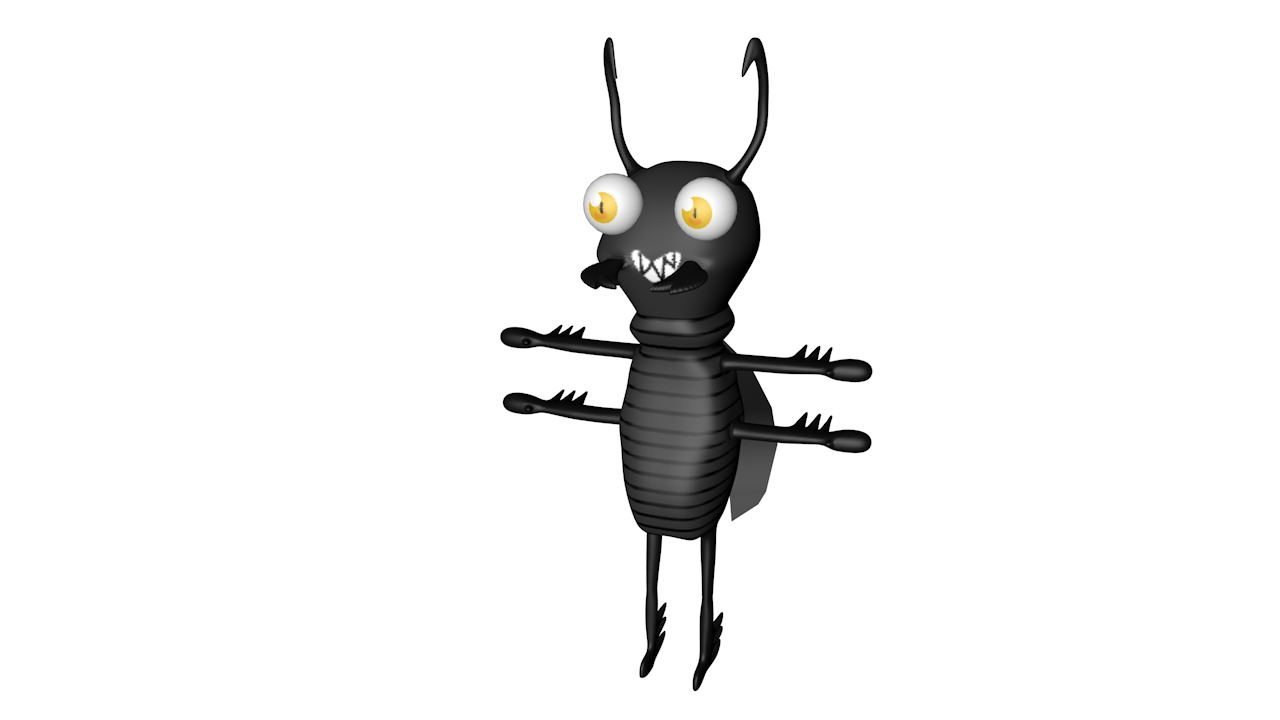






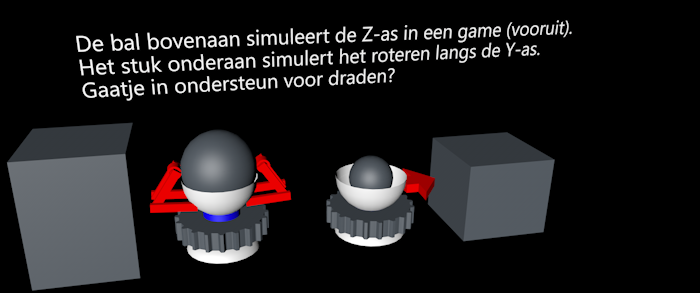
1. **3D model beetle**

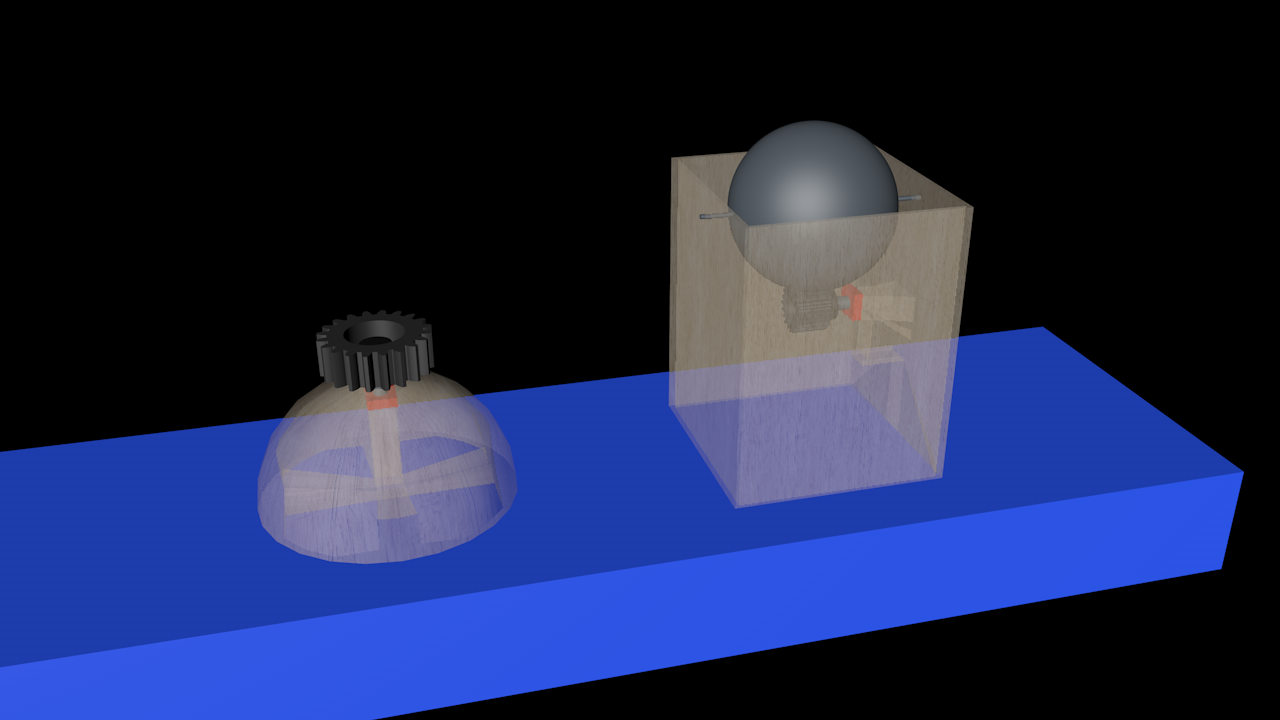
Nu de beetle een vast ontwerp had, was er een 3D model nodig om achter de bal te laten lopen. Het is bijna gemaakt uit 1 object, met uitzondering van de vleugels en de ogen.

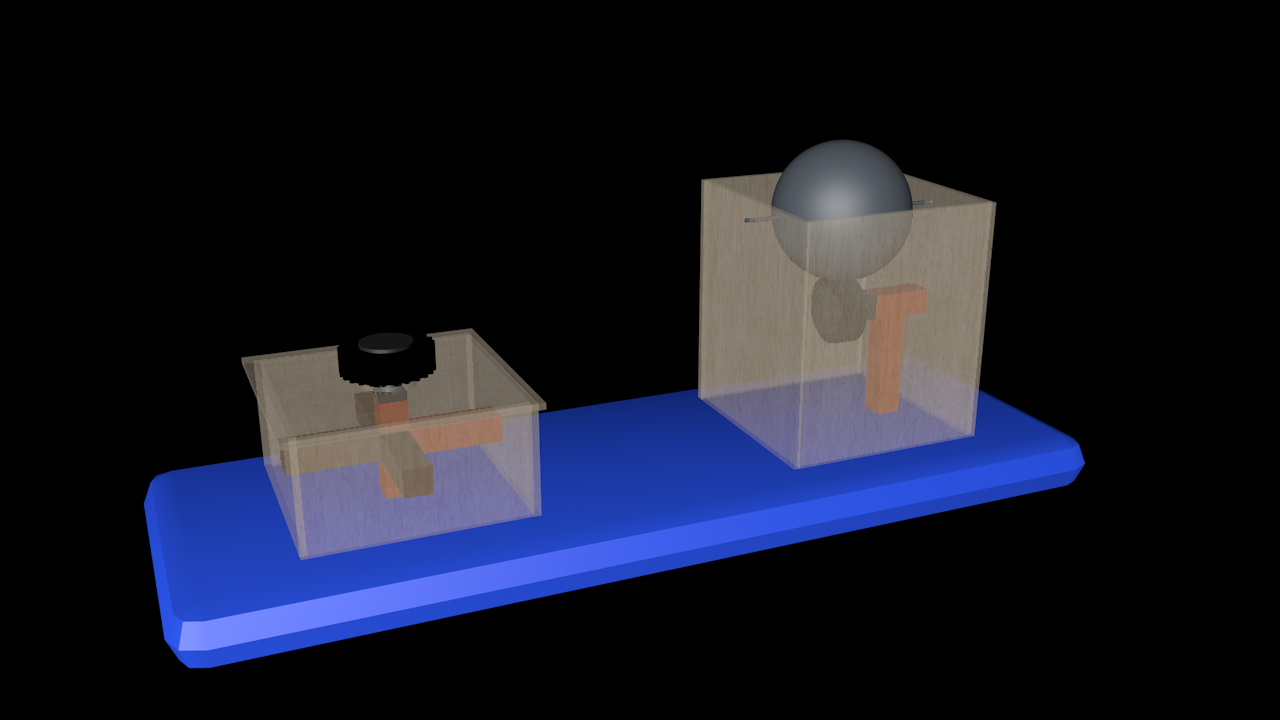


1. **Bescherming controller deel 2**

Het buitenste van de controller was nog steeds niet gemaakt. Het hout had ik al eerder gaan halen, dus er moesten nog gaten geboord worden en de stukken moesten nog aan elkaar gelijmd worden. Zo is het ontwerp van de controller langs de binnenkant nog enkele keren aangepast geweest tot een optimale opstelling.



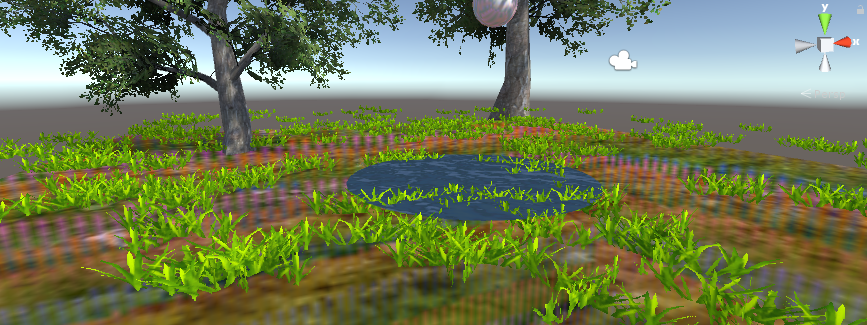


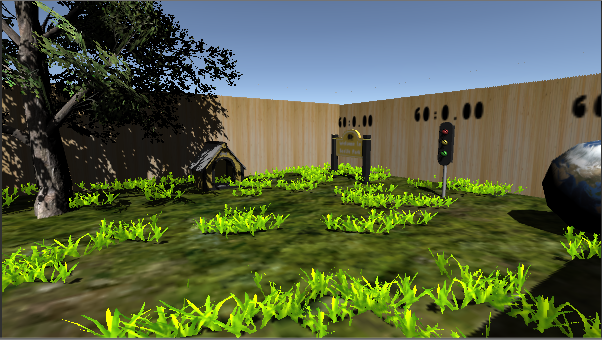


**LAATSTE VERSIE!!!!!!**

1. **Speedtrees en terrain**

Ik heb moeite met het maken van bomen. Als het wel lukte, dan waren de bomen zo zwaar i.v.m. polygon count, dat het spel last had van vertragingen, dus ik besloot om gebruik te maken van terrain en speedtrees. De speedtrees zaten al ingebouwd in het unity package > environment. Het gras heb ik wel zelf gemaakt met de tree editor. De textuur die ik hiervoor gebruikt heb, is wel van het internet (<http://www.reinerstilesets.de/3dtextures/billboardgrass0002.png>). Omdat unity denkt dat het grass ook een speedtree is, kan ik dit object te schilderen op een terrain.





Overzicht overwonnen problemen en oplossingen (nummers nog aanpassen)

* Als een object kleiner wordt dan de speelbal, dan zou de trigger aan moeten springen van een object dat opgeraapt kan worden. Bij mesh colliders leek dit een probleem te zijn. Als bij mesh collider **convex niet aangevinkt is, dan kan je geen trigger gebruiken.**
* **Bij convex trigger kwam er een ander probleem. Als een collider convex is en de bal is kleiner dan het object (bv. bij een parkbark), dan kan de bal niet onder het object rollen. Dit heb ik dan opgelost om tijdelijk meerdere box colliders te gebruiken.**
* **Ik wou de beetle de bal laten volgen en de rotatie laten gebruiken van de camera (zodat de beetle altijd voor de camera staat). Hiervoor wou ik wiskunde en rechthoekige driehoeken voor gebruiken, maar dit zorgde altijd voor een ander resultaat dan gewenst (zien screenshot punt 3). Dus uiteindelijk heb ik een punt aangemaakt onderaan de bal dat altijd op local ypos = 0 blijft t.o.v. de bal. Daaraan zit de beetle bevestigd. De beetle staat op een afstand die gelijk is aan 2/3 van de diameter van de bal, zodat hij niet te dicht of veraf van de bal staat. Het punt dat onderaan de bal staat, kijkt altijd naar de bal.**
* **Toen ik de nummers aan het font wou toevoegen, waren enkele letters afgekapt (bv. l, p, f…). Dit bleek fout zijn gelopen bij de conversie naar het font bij icomoon. Omdat Endutt geen genormaliseerd font was, stonden er geen afstanden en spacing geprogrammeerd zijn. Dus ik heb alle letter en nummers correct moeten positioneren in de ruimte t.o.v. de rest van de letters en cijfers, zodat alles even hoog/laag staat.**
* **Een spatie aanmaken in een font bleek lastiger te zijn dan verwacht. Elke keer als ik dit probeerde, leken alle letters aan elkaar vast te hangen, ondanks dat ik een spatie had getypt. Uiteindelijk heb ik dit opgelost door een lijn te tekenen in Adobe Illustrator en deze een opacity van 0% te geven zodat ze onzichtbaar werd. Dit leek de oplossing te zijn.**
* **Als je een eigen ontwerp voor de cursor maakt, dan moet dat geen .cur-extensie hebben. Een gewone .png afbeelding werkt.**
* **Ik wou minder verschillende textures, die op bepaalde polygonen waren geplaats met polygon selection, herleiden naar 1 texture (zoals je bij UV mapping zou doen). Daarvoor moest ik de texture baken. In cinema4d ziet het eruit zoals verwacht, maar in unity was dat niet zo (zie punt 6 bij screenshots). Ik heb veel verschillende manieren geprobeerd om het gewenste effect te verkrijgen. Dus ik heb ervoor gekozen om transparante texturen op objecten, apart te houden en deze niet te baken. Dus eerst voeg ik alle aparte stukken behalve het glas object samen door connect, dan bake ik alle texturen tot 1 textuur en daarna connect ik het glazen onderdeel. Hierdoor kan ik zelf kiezen hoe groot de texturen zijn in filegrootte en beperk ik alle objecten tot ong. 2 textures (textuur van alles + glastextuur).**
* **Daarnaast heeft het baken van de textuur een oplossing gegeven voor een ander probleem dat ik tegenkwam: de outline rond variërende meshes met meerdere texturen werkt niet zoals ik het zou willen (zie punt 7 bij screenshots). Door dit naar 1 textuur te brengen, werkt het zoals ik het zou willen. Het had nog wel kleine aanpassingen nodig bij de shader zelf, maar het was al vele beter zoals ik het verwachte.**
* **Blijkbaar kan unity niet alle video extensies aan (Zie punt 10 bij screenshot). Ik heb even rondgezocht online welke extensies het wel aankon, en kwam bij een .ogv-extensie. Dit kon je online converteren op** <http://video.online-convert.com/convert-to-ogv>**.**

Conclusie

Wat geleerd?

* **technisch**
* **creatief**
* **persoonlijk**
* Creatief / technisch: bij 3d modellen dacht ik soms: “Hoe zou ik het met papier gemaakt hebben,” en dit paste ik dan toe (zie screenshot punt 5)
* Hoe minimap
* Hoe highscore met keys bijhouden
* Hoe shaders werken (shaders aanpassen)
* Niet altijd de moeilijke weg proberen, soms kan het ook makkelijk
* Je kan verschillende materials toewijzen met polygon selection (tags) en daarna alle aparte stukken samenvoegen tot 1 object. Daarna kan je de verschillende textures tot 1 texture samenvoegen door de texture te baken.
* Een movietexture kan je als een normale texture gebruiken (je kan het bv. in een rawimage steken). Voor sommige extensies kon unity niet de correcte encoder vinden (MPEG2).
* Met speedtree kan je ook gras maken wat minder zwaar lijkt te zijn voor de processor. Hoe je met terrains werkt.

Zelfevaluatie

* uhhh….

Verbeteringen

Wat zou je verbeteren/toevoegen als je nog 6 maanden zou kunnen doorwerken.

Bronnen

Algemene bronnen

* **Unity docs algemeen**. <https://docs.unity3d.com/Manual/index.html>. Verschillende momenten.
* **Arduino Rotary Encoder**. <http://playground.arduino.cc/Main/RotaryEncoders>. Geraadpleegd op 3 maart 2017.
* **Meer info en code voor de Arduino Rotary Encoder**. W. Van Weyenberg, lector en schoolpromoter KdG Multimedia en CommunicatieTechnologie, persoonlijke communicatie, 3 maart 2017.
* **Outline met shader**. <http://wiki.unity3d.com/index.php/Silhouette-Outlined_Diffuse>. Geraadpleegd op 1 april 2017.
* **Loading screen**. <https://www.youtube.com/watch?v=xJQXoG3caGc>. Geraadpleegd op 2 april 2017.
* **Endutt font**. <http://www.dafont.com/endutt.font>. Geraadpleegd op 9 mei 2017.
* **Highscore met playerprefs**. <https://unity3d.com/learn/tutorials/topics/scripting/high-score-playerprefs>. Geraadpleegd op 10 mei 2017.
* **Minimap**. <http://blog.theknightsofunity.com/implementing-minimap-unity/>. Geraadpleegd op 11 mei 2017.
* **MovieTexture**. https://docs.unity3d.com/Manual/class-MovieTexture.html. Geraadpleegd op 20 juli 2017.
* **Converteren naar .OGV.** <http://video.online-convert.com/convert-to-ogv>**.** Geraadpleegd op 20 juli 2017.
* **SpeedTree Grass uitleg**. <https://forum.unity3d.com/threads/best-way-of-making-grass-looks-a-lot-like-speed-tree-grass-runs-as-fast-and-looks-just-as-good.339792/>. Geraadpleegd op 25 juli 2017.
* **SpeedTree Grass Texture.** <http://www.reinerstilesets.de/3dtextures/billboardgrass0002.png>. Geraadpleegd op 25 juli 2017.
* **SpeedTrees.** Package Environment. Geïmporteerd op 25 juli 2017.

Muziekbronnen

* **Muziek comic video**. Haydn, J., Hogwood, C. [Academy of Ancient Music](https://www.amazon.com/s/ref=dp_byline_sr_music_3?ie=UTF8&field-artist=Academy+of+Ancient+Music&search-alias=music). (2007). *Haydn: Symphony No. 94 'Surprise'; Symphony No. 96 'Miracle'* [cd-rom]. [Decca Music Group Limited](https://www.discogs.com/label/266418-Decca-Music-Group-Limited)
* **Muziek park level 1**. Waltz of the flower (door Tchaikovsky). <https://www.youtube.com/audiolibrary/music>. Geraadpleegd op 8 mei 2017.
* **Muziek menu level**. Skaterwaltz (door Waldteufel). <https://www.youtube.com/audiolibrary/music>. Geraadpleegd op 1 juli 2017.