Eindverslag afstudeerwerk

KUSOMARI

17 AUGUSTUS 2017

Auteur *: Jiska Baeten*

Schoolpromotor *: Wim van WeyenBerg, Pieter Jorissen*

KdG

Karel de Grote Hogeschool

Opleiding Multimedia en Communicatietechnologie

Inhoudsopgave

Inhoudsopgave

Omschrijving

Het idee

Het moest een pc-game worden dat 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 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 (mestkever) 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

Programma's

- Unity
- Arduino
- Adobe Photoshop
- Adobe Illustrator (om het font aan te passen)
- Icomoon.io (om het font aan te passen)
- Adobe Premiere
- www.smoothie-3d.com

Gebruikte tools/code 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
- Oplossing volume berekenen onregelmatige meshes http://answers.Unity3d.com/questions/1306365/mesh-volume-static-batching.html

➔ Verwijzing naar bronnen te vinden op de laatste pagina.

Zelf gemaakt

- Alle code van het besturen van de bal
- Alle code voor het oprapen van kleinere objecten (met uitzondering van het bereken van het volume van onregelmatige meshes)
- Alle code voor het bewegen van de camera
- De code voor het wisselen tussen shaders (om de outline te tonen)
- De code voor highscores op te slaan, terug op te vragen en te resetten (mits met wat hulp van Unity tutorials https://Unity3d.com/learn/tutorials/topics/scripting/high-score-playerprefs)
- Code rond timer, toggle muziek, toggle cursor visibility, wincheat…
- Code van Arduino met de hulp van Wim Van Weyenberg
- Code dat de arduinocode opvangt en verwerkt in Unity
- Aansluiten voor aan de arduino + de beschermde shell
- Alle modellen (mier en eend zijn gedeeltelijk in Smoothie gemaakt)
- Cijfers voor het Endutt font

- Alle tekeningen (voor de comic, loading screen, cursor en startup images)
- Animatie van de mestkever + animation states
- Het terrain (verschillende hoogtes en textures painten)

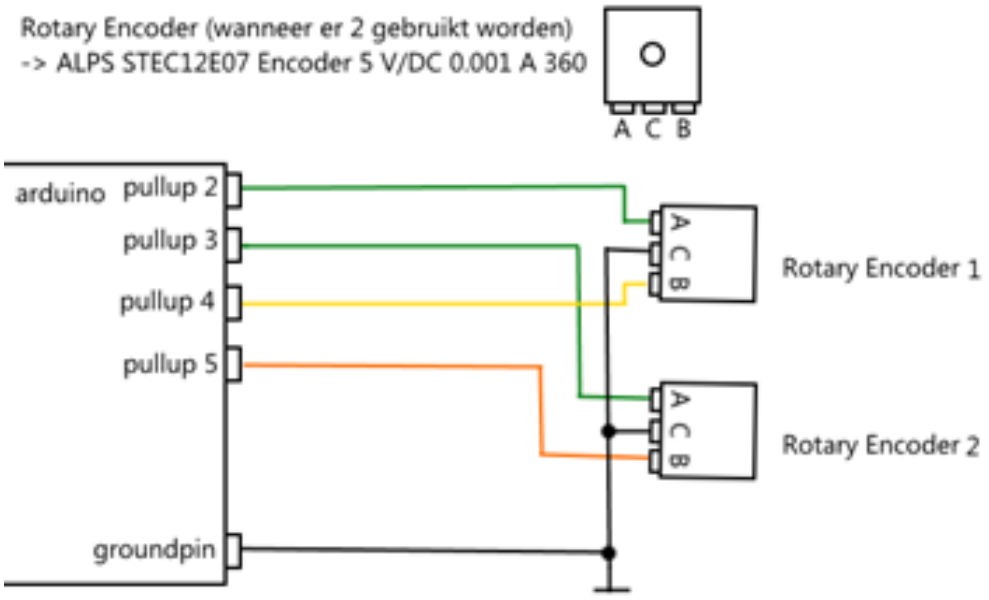
Opbouw eindwerk

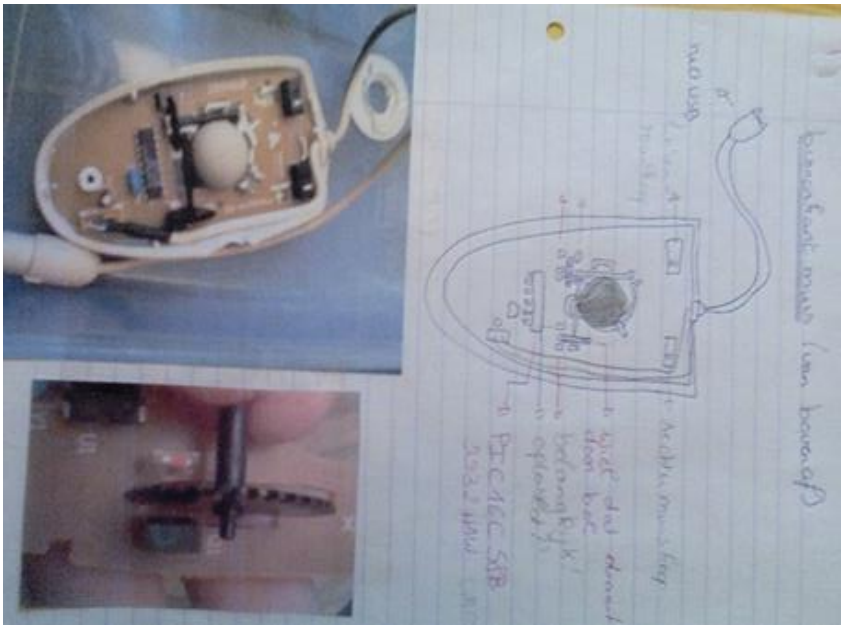
1. Code Unity

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.

2. Controller hardware en software

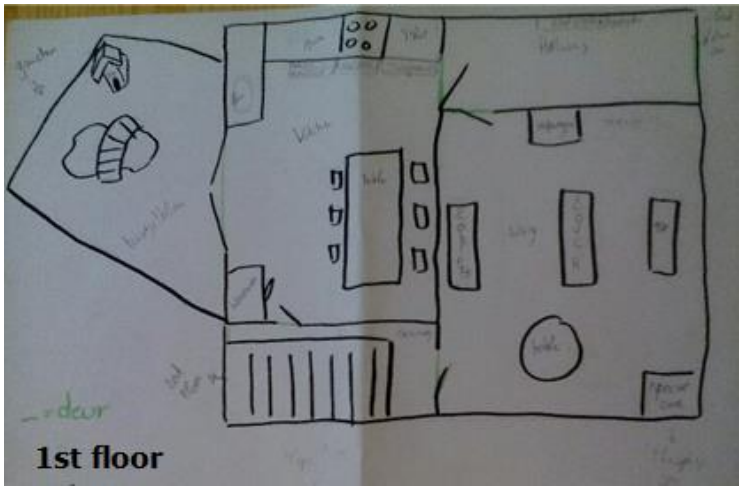
Ik wou ook erg graag dat de controller die ik voor dit game maakte, dat deze hardware en software al snel zou werken. Ik was van plan om de onderdelen van een oude muis te gebruiken. Een bal zou rollen tegen een tandwiel dat tussen een lichtdiode en ontvanger stond en zo een interrupt zou opwekken. Pas 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.





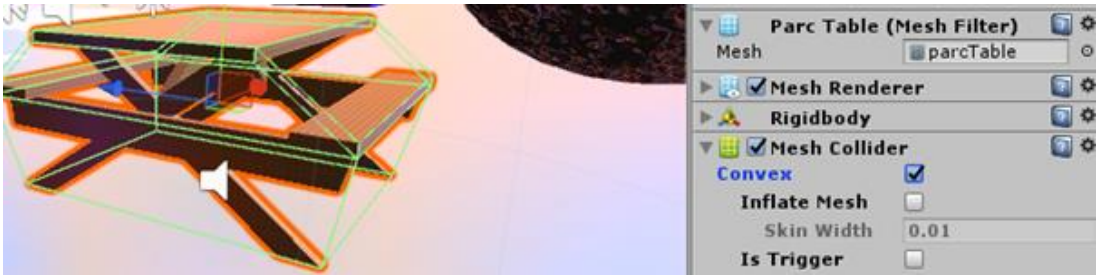
3. Eerste ontwerp levels

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.



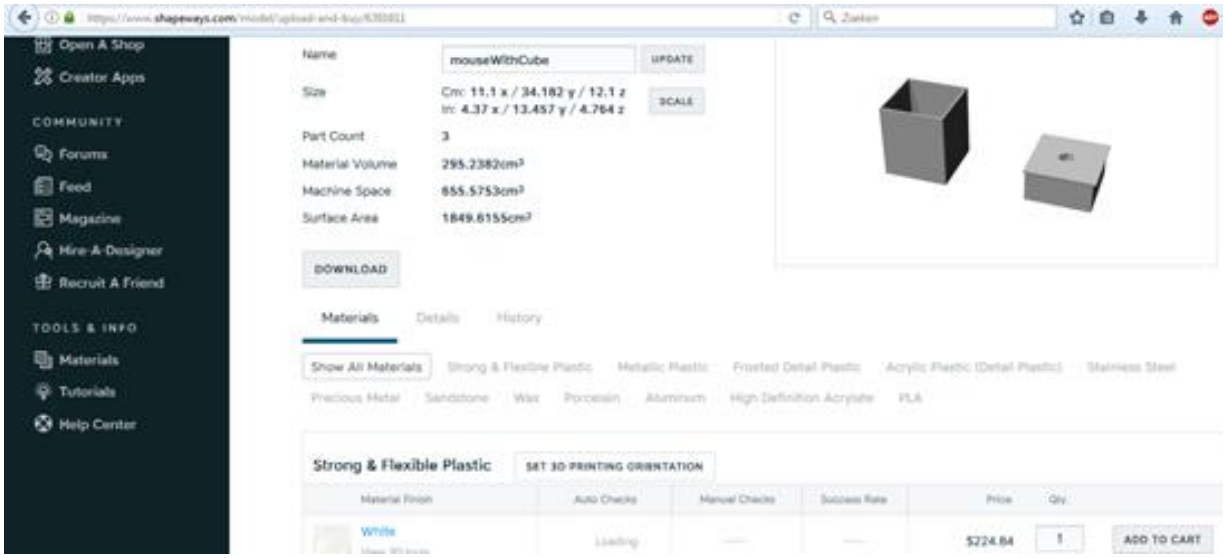
4. Eerste modellen en sandboxtesten

Ik maakte 3D modellen die 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.



5. 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.



6. Loading screen

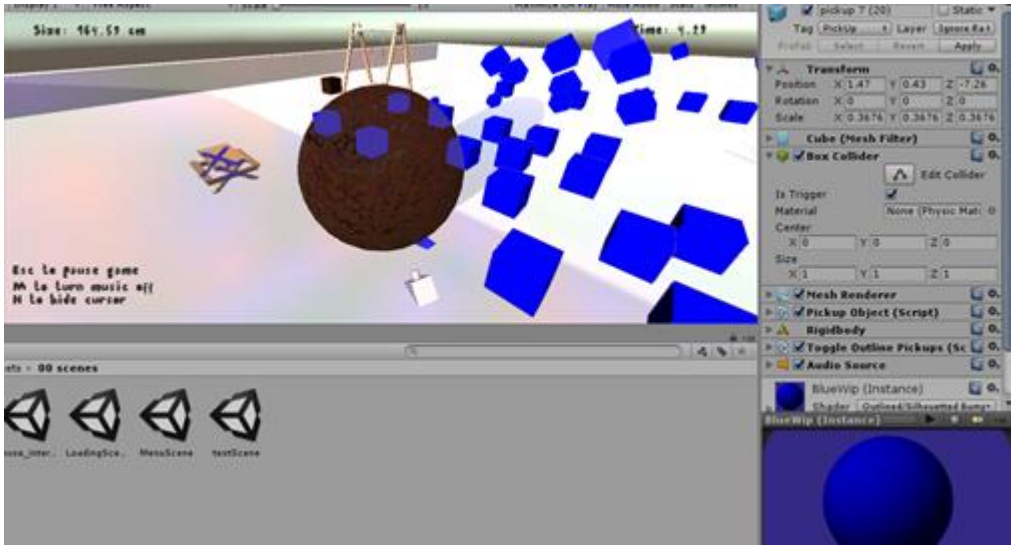
Heel graag wou ik een loading screen, om ongemakkelijke pauzes van Unity niet zichtbaar te maken wanneer hij iets aan het laden is (zoals een nieuw level). Na wat rondzoeken, kwam ik op een youtube-video terecht die een script had gemaakt voor een loading screen. Hij legde daarin uit hoe je zijn script 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 mestkever later aangepast is geweest, heb ik dan ook de animatie van het laadscherm aangepast.

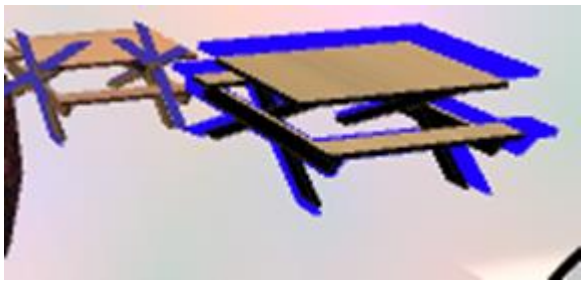
7. Timer

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.

8. 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.





9. 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.

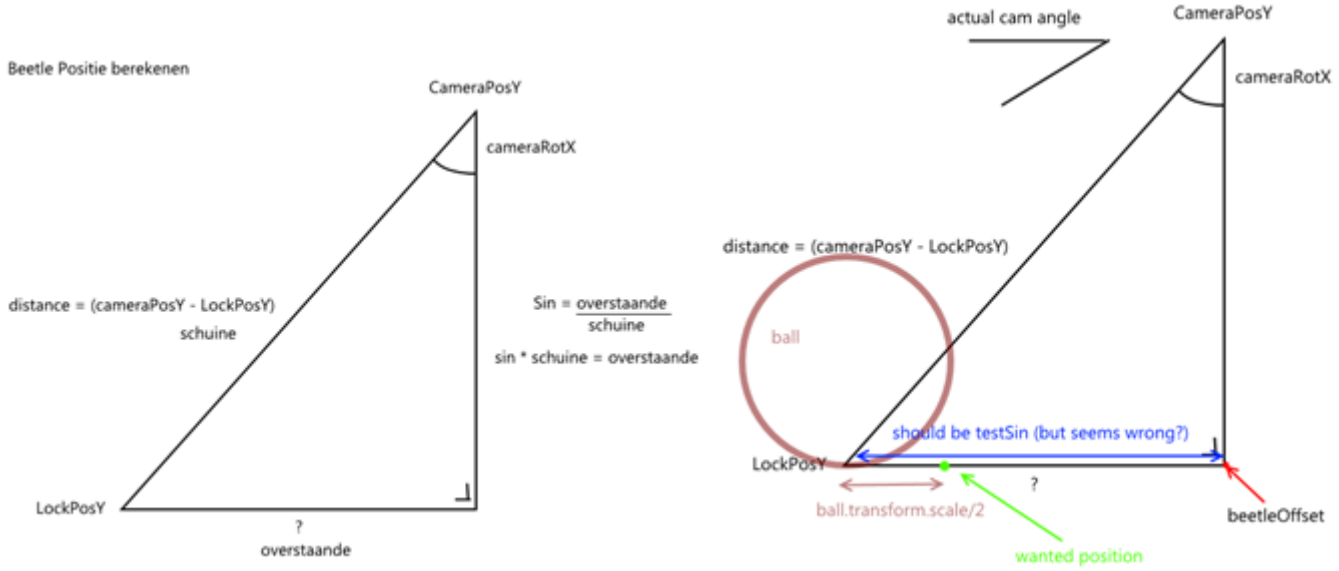
10. Mestkever volgt bal

De mestkever 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.

```
17    private Quaternion camRot;
18
19    void Start()
20    {
21      cc = cameraOb.GetComponent<cameraController>();
22    }
23    void Update()
24    {
25      beetleOffset = new Vector3(cameraOb.transform.position.x, lockPosition.position.y, cameraOb.transform.position.z);
26      camRot = Quaternion.Euler(cameraOb.transform.rotation.x, cameraOb.transform.rotation.y, cameraOb.transform.rotation.z);
27
28      // transform.position = lockPosition.position - beetleOffset; //beetleoffset is nu recht onder cam
29      //transform.RotateAround(lockPosition.position, Vector3.up, camRot);
30
31      //afstand lock - cam = s
32      //schuine, overstaande nodig -> sinus
33
34      //rotates with cam so beetle is always looking at the ball
35      transform.LookAt(lockPosition.position);
36    }
37  }
```

```
    public class FollowPlayer : MonoBehaviour
    {
      //used for the beetle following the ball, rotates with cam
      //DOESN't WORK YET
5.
      public Transform lockPosition;
      public GameObject cameraOb;

      private Vector3 offset;
10.   private Vector3 beetleOffset;

      private float distance;
      private float testSin;
```

```
15.    void Update()
       {
         beetleOffset = new Vector3(cameraOb.transform.position.x, lockPosition.position.y, cam
    eraOb.transform.position.z);

         distance = cameraOb.transform.position.y - lockPosition.position.y;
20.
         if (cameraOb != null)
         {
           testSin = Mathf.Sin((cameraOb.transform.rotation.x) * Mathf.Deg2Rad*100) * distance;
    //*100 since degrees are in comma
         }
25.      transform.position = beetleOffset - new Vector3(0f, 0f, testSin);

         //rotates with cam so beetle is always looking at the ball
         transform.LookAt(lockPosition.position);
       }
30.  }
```



11.Achtergrondmuziek

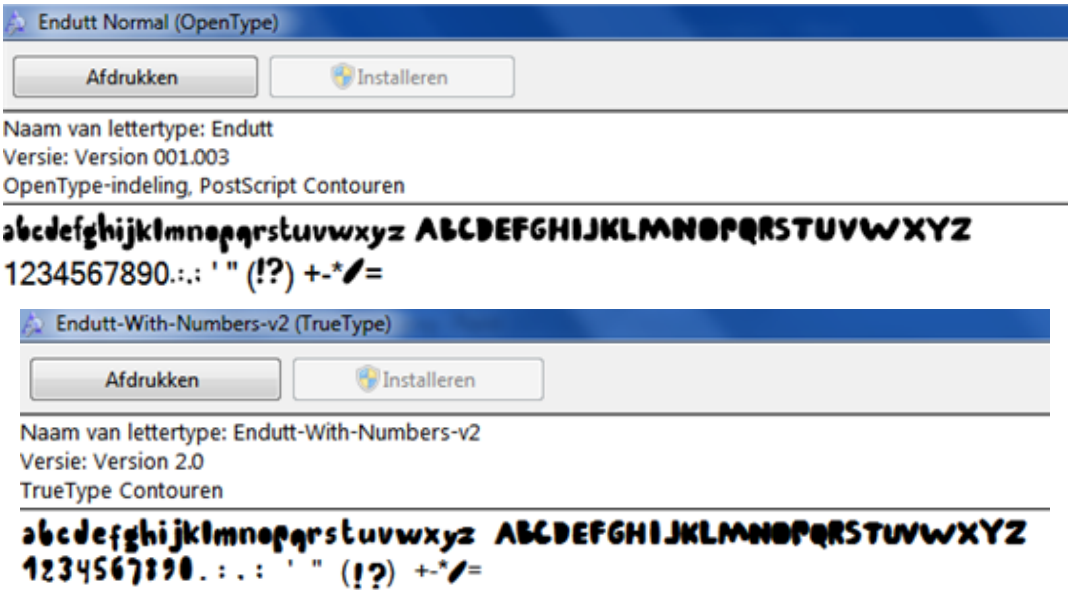
De levels waren doodstil. Het enige geluid dat ik had geïmplementeerd, was een geluid wanneer iets opgeraapt zou worden. Ook zou het nagaan van copyright een last zijn en de ontwikkeling van het game vertragen, dus ben ik gegaan voor klassieke muziek. Als je deze muziek versnelt, dan lijkt het soms wel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

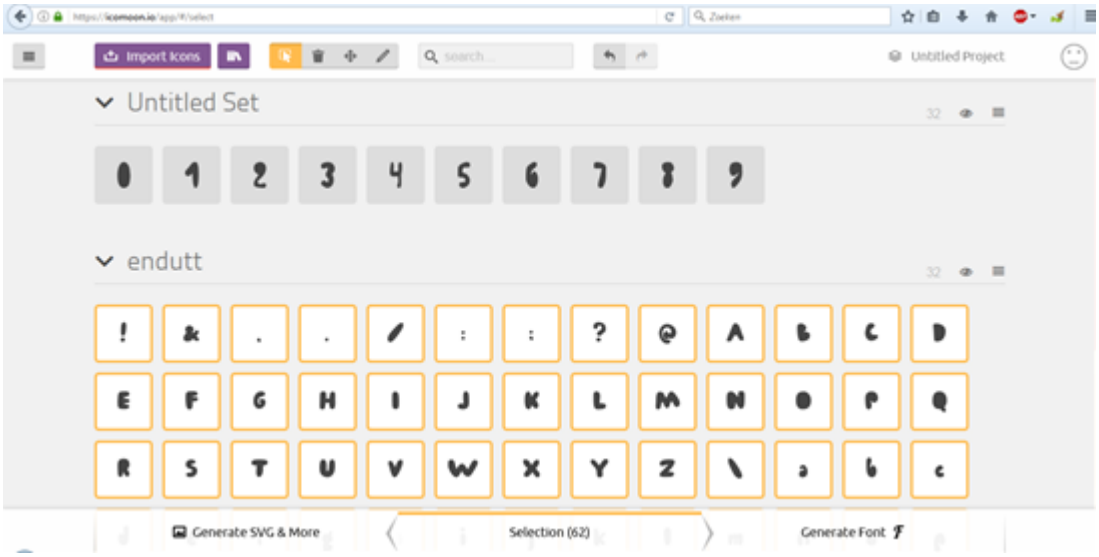
12. Pauzescherm

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.

13. Nummer aan font toevoegen

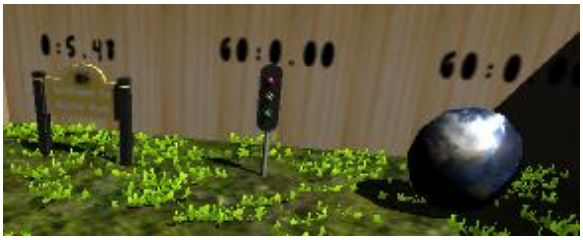
Ik merkte plots op dat het font dat ik gekozen had, geen nummers bevatte. Ik was ondertussen al gehecht geraakt aan het font, dus ik besloot om zelf de nummers aan het font toe te voegen. We hadden tijdens een graphics vak geleerd hoe we een font konden maken. Dus samen met Photoshop (om snel de vormen 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.

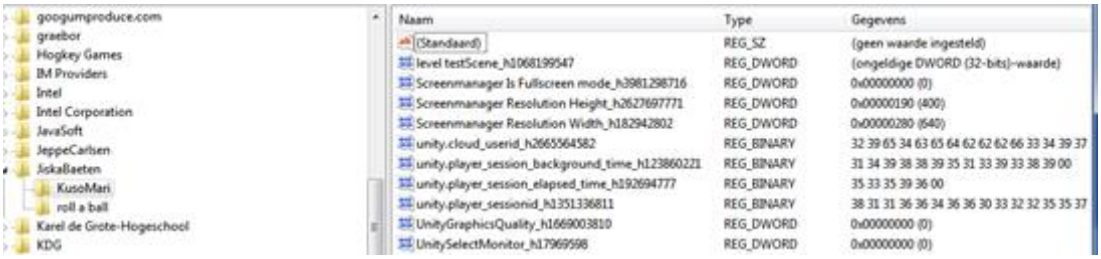




14.Opslagen en laden van highscores

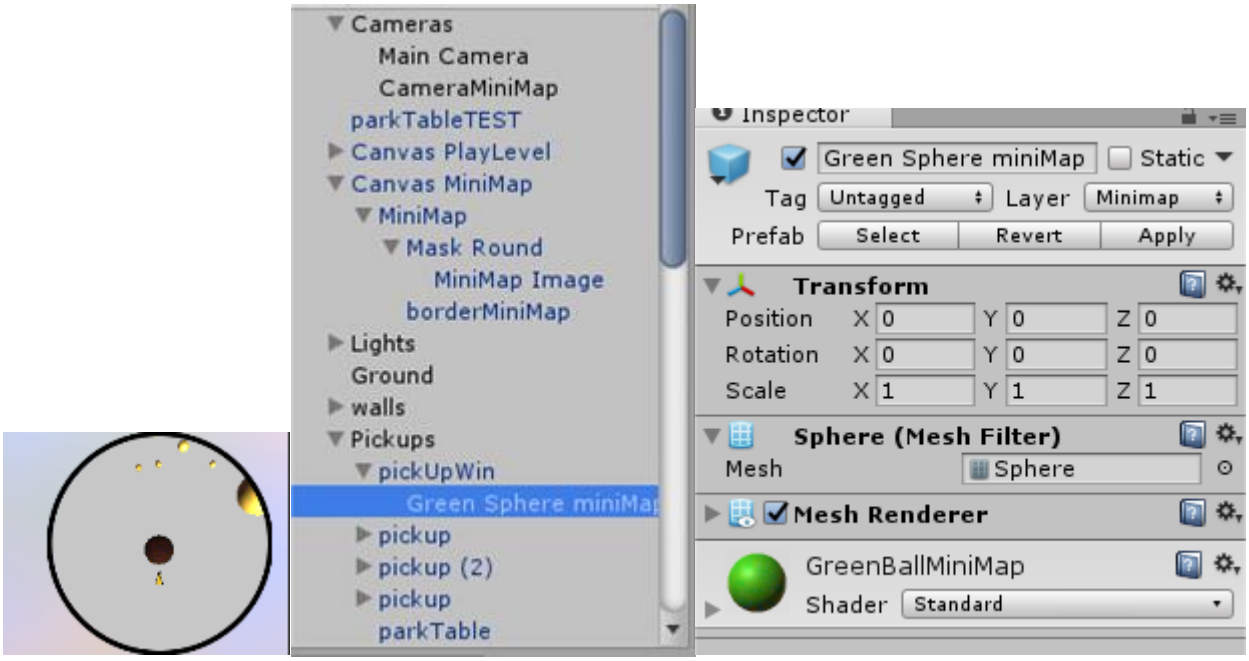
Om herspeelbaarheid te motiveren, was een implementatie van een highscore noodzakelijk. Ik wou dit niet lokaal in een bestand opslagen, omdat dit dan aanpasbaar was. Daarnaast vind ik het saven op een binaire manier erg omslachtig. Ik zocht een andere oplossing en kwam op lokale keys terecht. Een tutorial 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.





15.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).

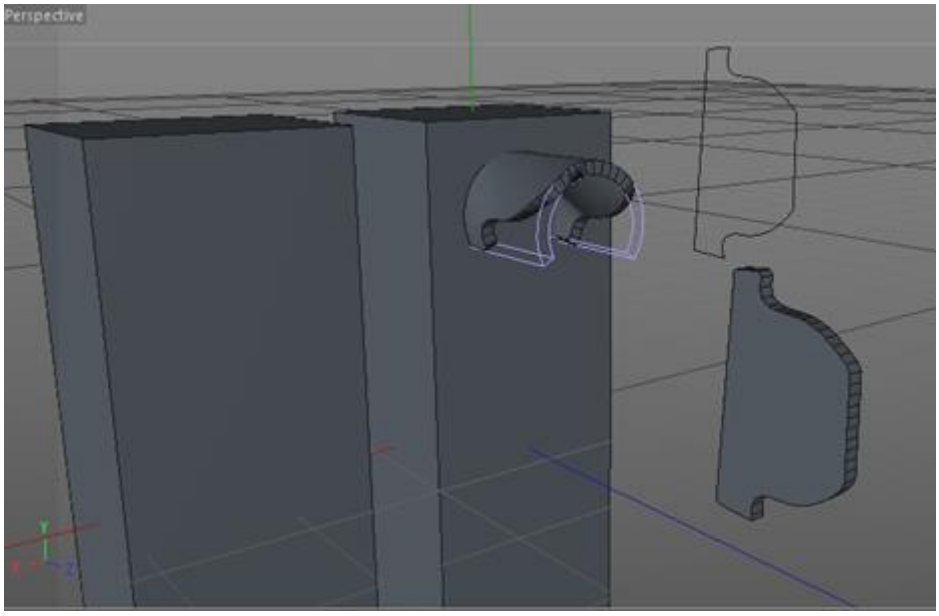


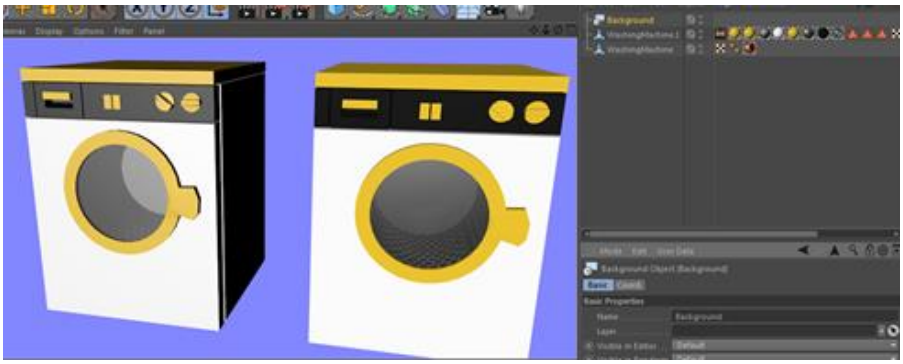
16.Fullscreen op lagere resolutie

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).

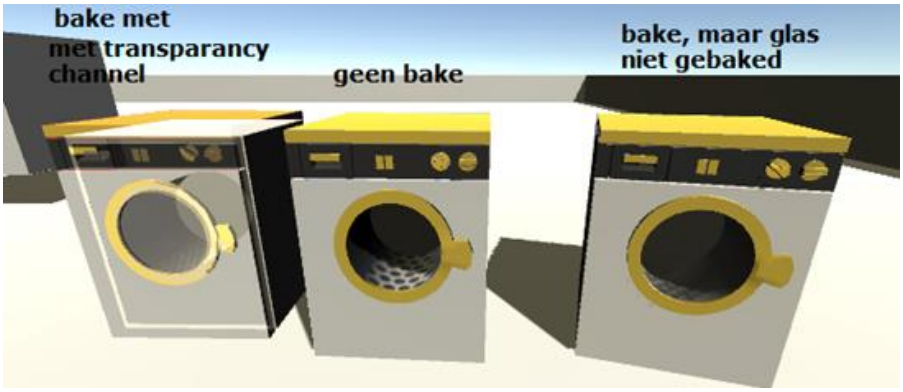
17.Testen i.v.m. baken en nieuwe ideeën om een 3D model te maken

Het was tijd om de levels in te vullen met 3D modellen. Ik maakte alle modellen die in het menu moesten 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.


Perspective

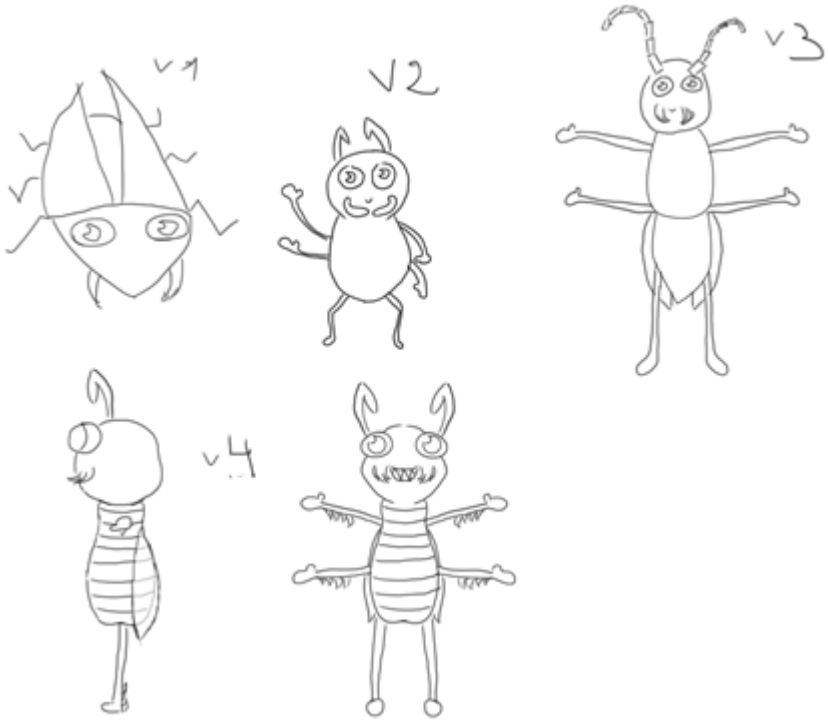



bake, met aangepaste alpha
bake, met glas
geen bake, versh. texturen
bake, zonder glas



18.Evolutie design mestkever

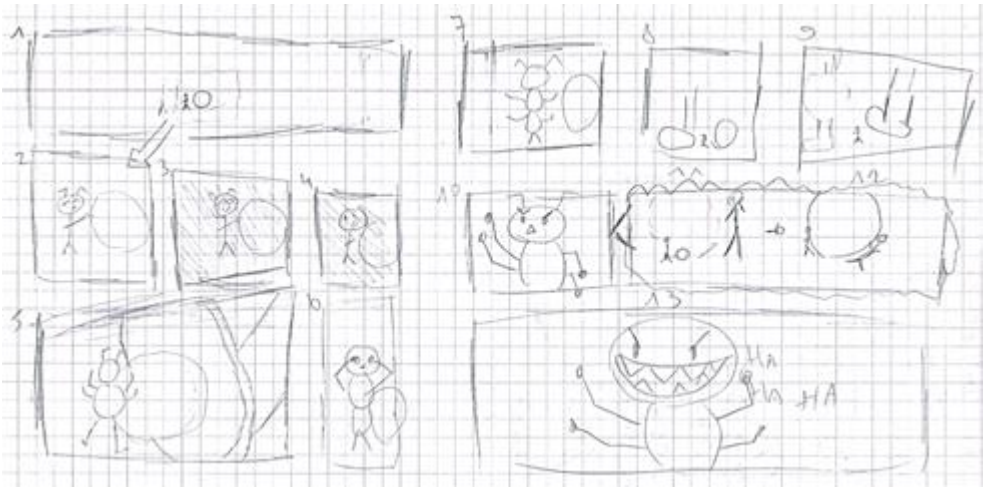
Ik vond het design van de mestkever toch niet interessant genoeg. Enkele schetsen later had ik een nieuw design waar ik dol op was. Het had iets cartoonachtig en het zag er ondeugend uit.

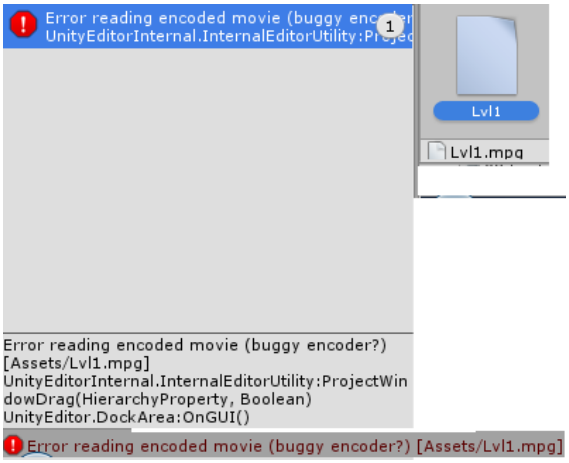


19.Comic video lvl1

Er was een motivatie nodig waarvoor de mestkever 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.





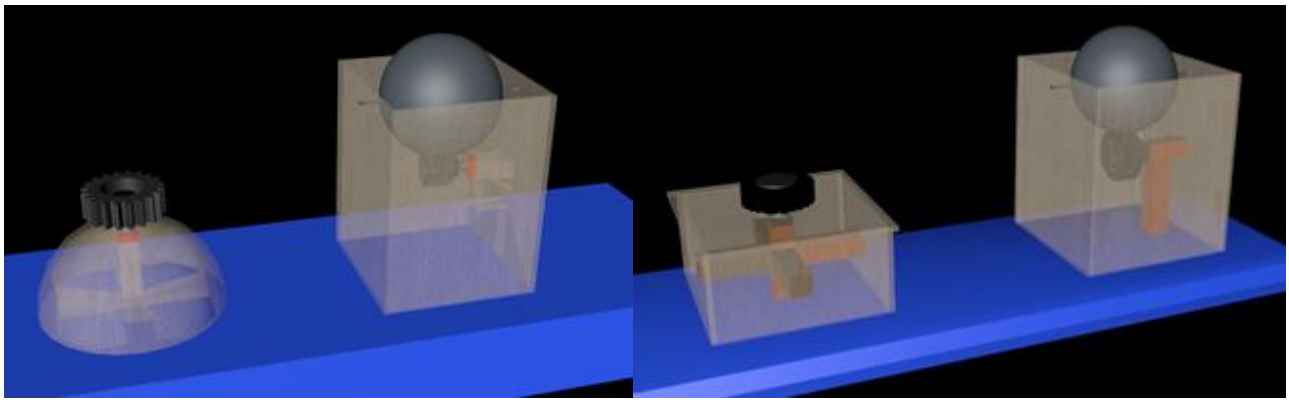
20.3D model mestkever

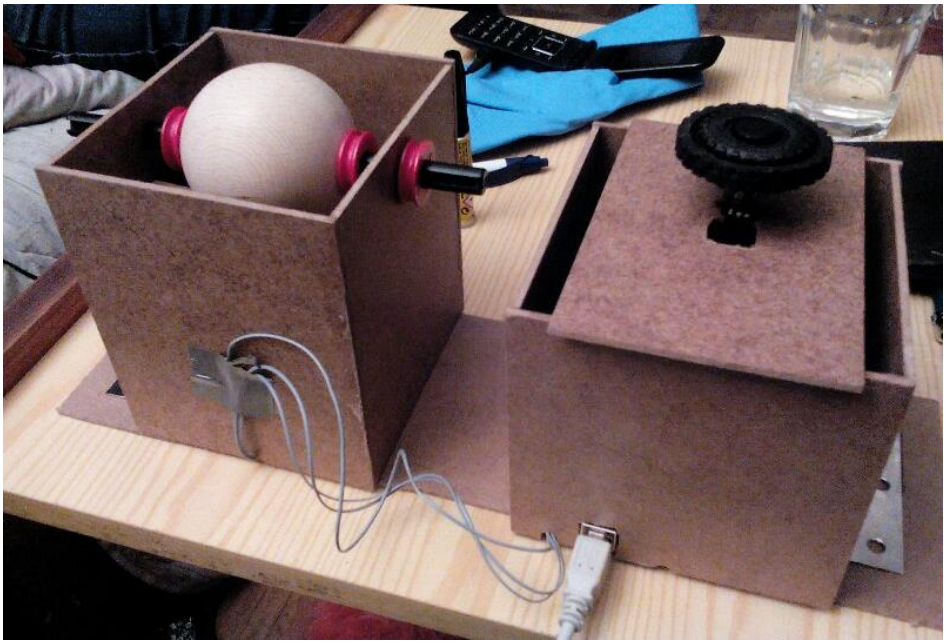
Nu de mestkever een vast ontwerp had, was er een 3D model nodig om achter de bal te laten lopen. Het is bijna gemaakt uit 1 object, met uitzondering van de vleugels en de ogen.

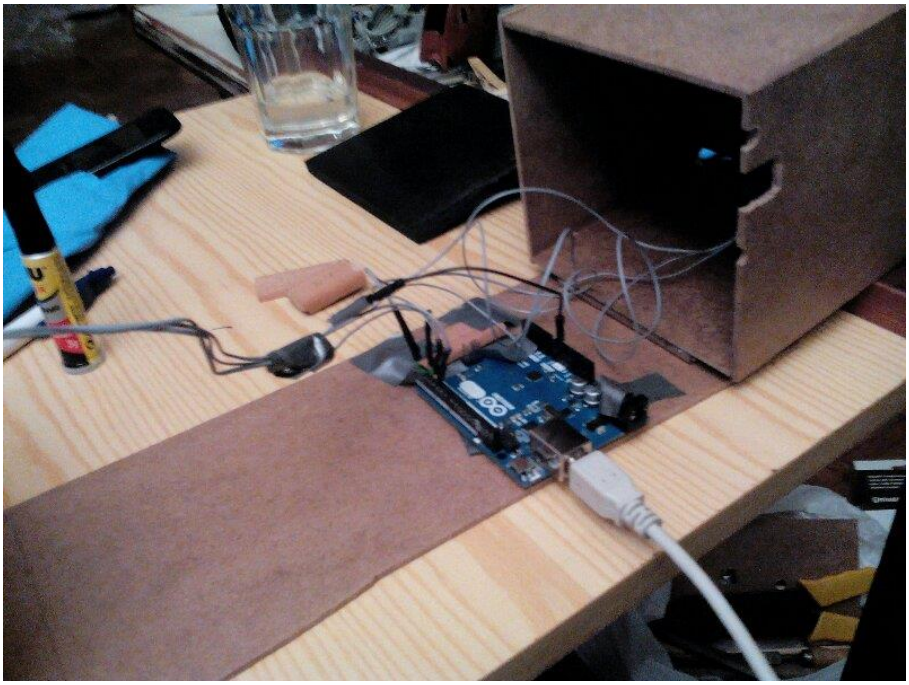
21.Bescherming controller deel 2

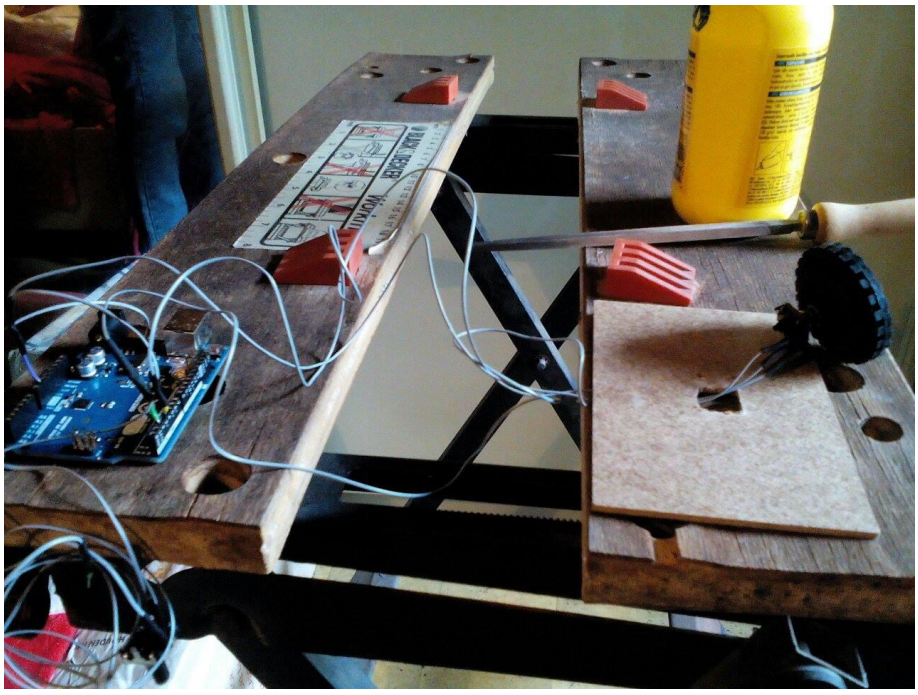
Het buitenste van de controller was nog steeds niet gemaakt. Het hout had ik al eerder gaan halen, dus er moesten nog gaten geboord worden en de stukken moesten nog aan elkaar gelijmd worden. Zo is het ontwerp van de controller langs de binnenkant nog enkele keren aangepast geweest tot een optimale opstelling.

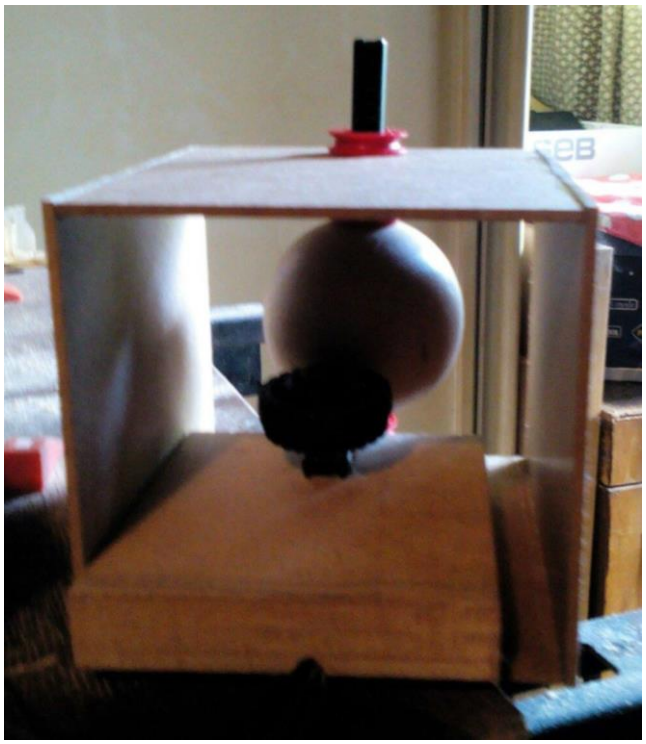

De bal bovenaan simuleert de Z-as in een game (vooruit).
Het stuk onderaan simulert het roteren langs de Y-as.
Gaatje in ondersteun voor draden?





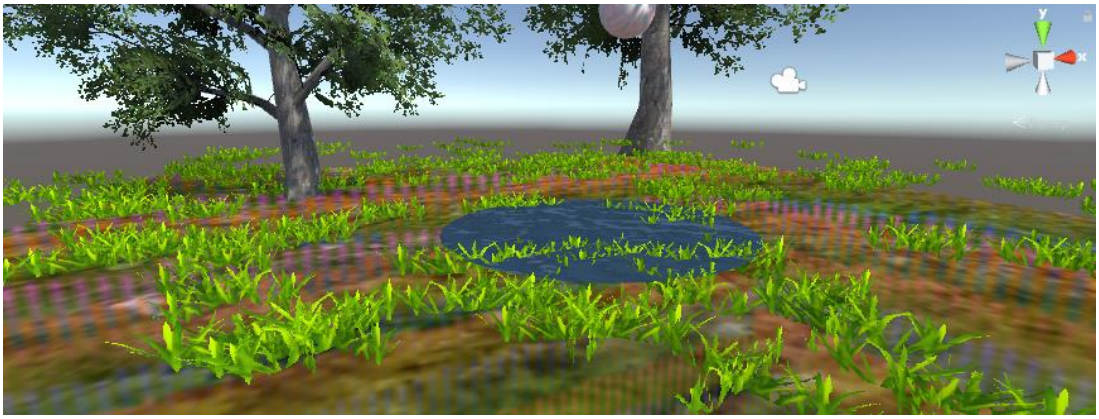






22.Speedtrees en terrain

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.



23.Code testen en aanpassingen doorvoeren

Ik was van plan om de camera automatisch traag te laten uit te zoomen, maar dit heb ik dan vervangen door 2 toetsen, waardoor de speler zelf de controle heeft over de camera.

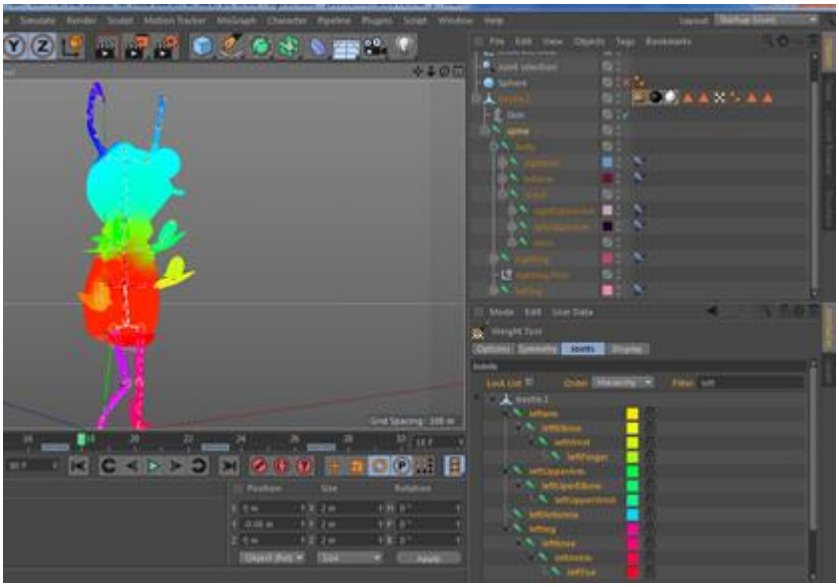
Ook is de code die de colliders beheren, zo aangepast dat ze sneller werken voor alle meshes (regelmatig en onregelmatig). Daarnaast werkt dit code nu ook sneller.

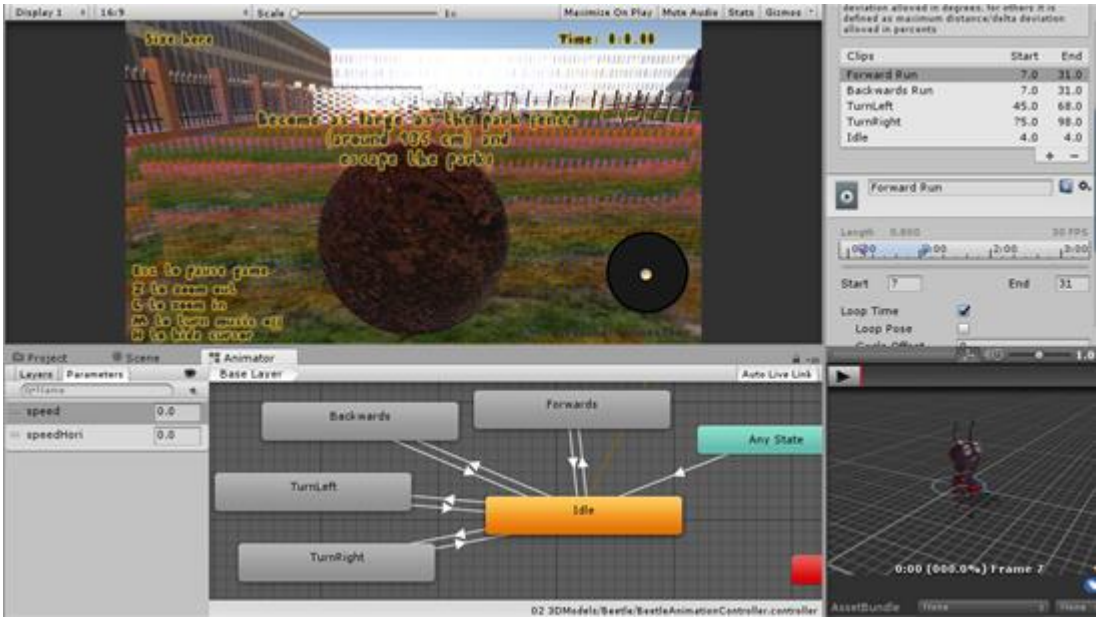
```
if (pickup.gameObject.GetComponent<MeshCollider>() != null) //when it has a mesh collider
{
  foreach (MeshCollider c in GetComponents<MeshCollider>()) // just in case I forgot to delete a collider
  {
    c.convex = true; // turn the mesh to convex -> so it can be triggered
    c.enabled = true; // turn the mesh collider on
    c.isTrigger = true; //turn on the trigger for the mesh
  }
}
```

24.Animatie toevoegen aan de mestkever

De mestkever bewoog nog niet als je de bal rolde. Ik had nog nooit geprobeerd om een 3d animatie over te brengen van cinema4D naar Unity. Daardoor wist ik niet dat je bij de .FBX-export ook nog animation track moest aanvinken. Ik wist ook niet hoe je een animatie kon splitsen (Kirsten had tijdens project 3 uitgelegd dat je alle verschillende states in 1 animation track kon animeren en dan later kon opsplitsen in Unity).

Eerst heb ik het model joints gegeven, daarna heb ik de weights van de joints licht aangepast omdat sommige delen vervormd waren en daarna heb ik alles geanimeerd.



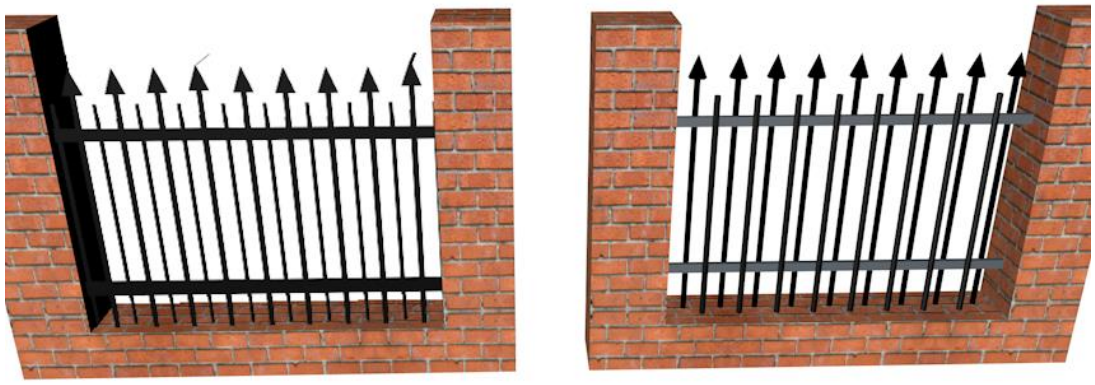


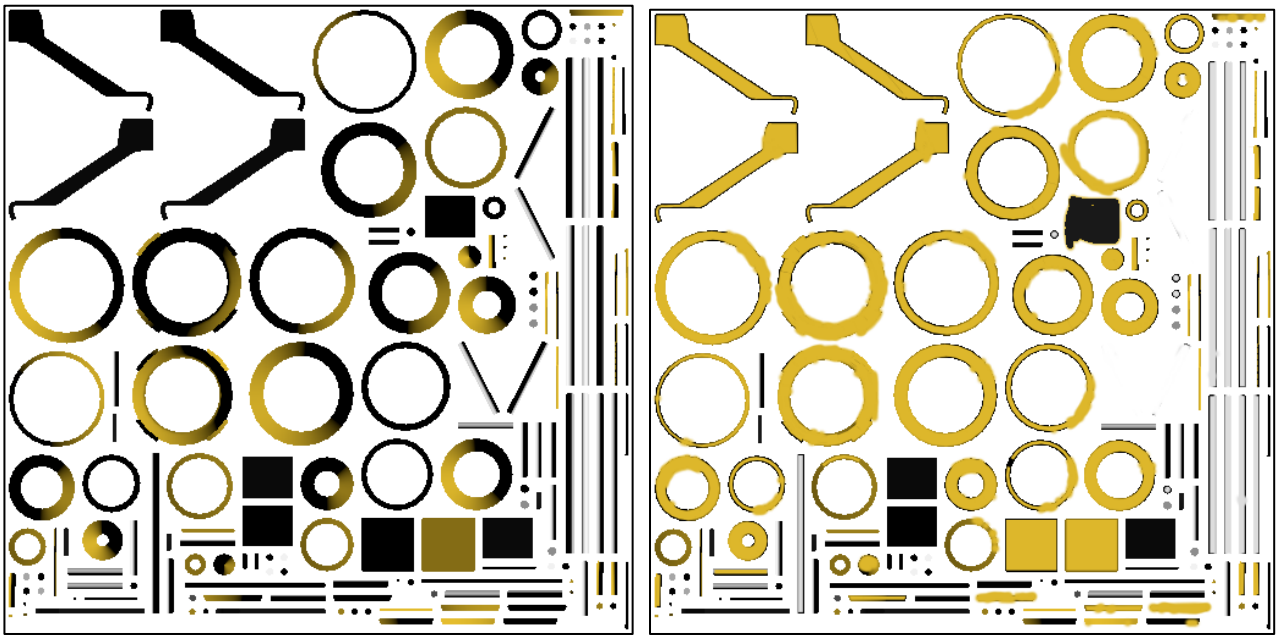
25. Modellen aanpassen en nieuwe modellen voor het park level

De modellen die voor het parklevel gemaakt waren, konden nog in polygonen verminderd worden. Daarnaast beschikten ze soms nog over meerdere texturen die herleidt konden worden naar 1 textuur, wat geheugen zou uitsparen en zou helpen bij de outline (die zich toont bij kleinere voorwerpen dan de bal). Daarnaast merkte ik bij de parkomheining op dat deze te snel opgerold kon worden bij het oude model. Daarom heb ik het volume hiervan vergroot en de platte alpha texture die ik voor de spijlen had gebruikt, vervangen door 3D modellen voor de spijlen. Zo werd het volume ook een beetje aangevuld.

Alle gebakte texturen waar ik surface color heb gebruikt, heb ik opnieuw aangepast omdat ze de schaduwen hierin ook al baken en dat wou ik niet (vooral omdat het er dan slecht uitzag).

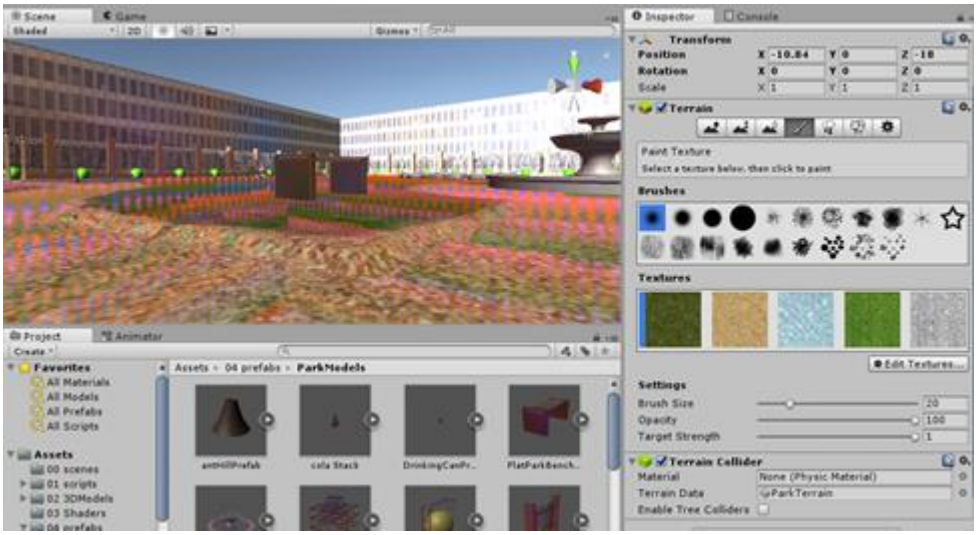
De stenen van het kleine meertje, zijn gesculpt en daarna in de vorm gebaked.





26.Terrain schilderen en hoogtes/laagstes painten

Op sommige plekken wou ik dat er hoogteverschillen waren. Als je bv. in water gaat, dat je toch iets lager kunt rollen dan de rest van het terrein. Zo heb ik ontdekt dat je op terrain texturen kan schilderen en dat hoogtes ook met een soort van schildertool werken. Je moest wel eerst hoogte hebben voordat je diepte kan creëren.



27. Laatste bugs eruit halen

Toen ik de 'Koka Kola'-blikjes in mijn scene stopte, merkte ik iets merkwaardig op. De bal was zeker 10 keer groter dan deze blikjes, maar volgens Unity waren ze groter dan de bal. De code om het volume van alle onregelmatige meshes te bereken was:

```
volumePickup = pickup.transform.localScale.x * pickup.transform.localScale.y *
pickup.transform.localScale.z;
```

Dit bleek niet de correcte code te zijn, sinds ze de scale van Unity gebruiken. Na wat opzoekwerk bleek dat het volume van onregelmatige meshes heel moeilijk te berekenen was. Ik vond de onderstaande code op http://answers.Unity3d.com/questions/1306365/mesh-volume-static-batching.html. Deze code zorgde voor betere resultaten. Daarnaast moest ik enkel zien dat de bal (die in een parent gameobject zit) dezelfde schaal had als de onregelmatige meshes, want voor de bal gebruik ik: `ballVolume = ((4 / 3) * Mathf.PI * Mathf.Pow(playerObject.transform.localScale.z / 2f, 3f));` -> Volume bal

Code voor volume berekenen onregelmatige meshes:

```
volumePickup = VolumeOfMesh(pickup.GetComponent<MeshFilter>().sharedMesh);

public float SignedVolumeOfTriangle(Vector3 p1, Vector3 p2, Vector3 p3)
  {
    float v321 = p3.x * p2.y * p1.z;
    float v231 = p2.x * p3.y * p1.z;
    float v312 = p3.x * p1.y * p2.z;
    float v132 = p1.x * p3.y * p2.z;
    float v213 = p2.x * p1.y * p3.z;
    float v123 = p1.x * p2.y * p3.z;
    return (1.0f / 6.0f) * (-v321 + v231 + v312 - v132 - v213 + v123);
  }
```

```
    public float VolumeOfMesh(Mesh mesh)
    {
      float volume = 0;
      Vector3[] vertices = mesh.vertices;
      int[] triangles = mesh.triangles;
                    for (int i = 0; i < mesh.triangles.Length; i += 3)
                    {
                      Vector3 p1 = vertices[triangles[i + 0]];
                      Vector3 p2 = vertices[triangles[i + 1]];
                      Vector3 p3 = vertices[triangles[i + 2]];
                      volume += SignedVolumeOfTriangle(p1, p2, p3);
                    }
                    return volume *= this.gameObject.transform.localScale.x *
                this.gameObject.transform.localScale.y *
                this.gameObject.transform.localScale.z;
}
```

Overzicht overwonnen problemen en oplossingen

- 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. **(Zie Opbouw eindwerk punt 4)**
- 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. **(Zie Opbouw eindwerk punt 4)**
- 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 Opbouw eindwerk punt 8).** 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.
- Als je een eigen ontwerp voor de cursor maakt, dan moet dat geen .cur-extensie hebben. Een gewone .png afbeelding werkt. **(Zie Opbouw eindwerk punt 9)**
- Ik wou de mestkever de bal laten volgen en de rotatie laten gebruiken van de camera (zodat de mestkever altijd voor de camera staat). Hiervoor wou ik wiskunde en rechthoekige driehoeken voor gebruiken, maar dit zorgde altijd voor een ander resultaat dan gewenst **(Zie Opbouw eindwerk punt 10).** 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 mestkever bevestigd. De mestkever 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. **(Zie Opbouw eindwerk punt 13)**

- 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. **(Zie Opbouw eindwerk punt 13)**
- 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 Opbouw eindwerk punt 17)**. 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).
- Blijkbaar kan Unity niet alle video extensies aan **(Zie Opbouw eindwerk punt 19)**. 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.
- Als je bij het baken van een texture de optie Surface color laat aanstaan, krijg je de kleuren + de schaduwen die hierop vallen, dat was niet de bedoeling. Als je enkel color baked, krijg je vaak een overbelichte versie van je textuur. Dus ik bake nu enkel in color en pas daarna de kleuren/texturen nog wat aan in Photoshop. **(Zie Opbouw eindwerk punt 25)**
- Blijkbaar als objecten te klein zijn en gravity hebben op hun rigidbody, dan is er de kans dat ze door collider gaan. Ik wou gravity gebruiken voor stapels zoals een blikjesmuur. Als je dan met de bal ertegen bots, dan vallen de blikjes omver. Bij modellen die hiervoor te klein waren, heb ik geen rigidbody gebruikt met gravity.
- De shader gaf de hele tijd een error aan dat de waardes die ik voor de kleur had ingegeven, niet genormaliseerd waren. Dit was op te lossen door de gekozen kleur zijn RGB waardes te nemen en elk getal te delen door 255, want de shader had een getal tussen 0 en 1 nodig.
- Modellen die vele keren kleiner waren dan de bal, werden niet opgeraapt. Dit heb ik opgelost door code op te zoeken die het volume van de meshes berekent en door de bal dezelfde schaal te geven mits zij wel met de localscale werkt (want als je de code van het berekenen van het volume van meshes constant gebruikt op de bal, dan is dit te zwaar). **(Zie Opbouw eindwerk punt 27)**

Conclusie

Wat geleerd?

- **Persoonlijk**: code kan op meerdere manieren opgelost worden, dus probeer niet dagen aan een stuk het op een wiskundige manier op te lossen wanneer het niet lijkt te lukken
- **Technisch**: Al heb je 4 blokgolven door je 2 rotary encoders, je hoeft enkel 1 pin van elke encoder op te volgen voor interrupts. De andere blokgolf kan je op het moment van de interrupt nakijken of deze hoog of laag staat.
- **Creatief / Technisch**: bij 3d modellen dacht ik soms: "Hoe zou ik het met papier gemaakt hebben," en dit paste ik dan toe
- **Technisch**: Naast een gewone bool bestaat er ook een Nullable bool. Hiermee kan je 3 states in steken (true, false en null). Dit hielp erg bij de controller via de arduino. Zo kon ik zonder strings doorgeven of we naar links/voor, rechts/achter of geen input hadden.
- **Technisch**: Hoe je makkelijk een minimap kan maken met een tweede camera, een mask en een rendertexture
- **Technisch**: hoe je een highscore met keys lokaal kan bijhouden, opvragen en resetten op een computer, zonder bestanden waar de speler in kan valsspelen
- **Technisch**: Hoe shaders werken en ik bestaande shaders snel kan aanpassen naar wens
- **Technisch**: Als je een kleur wilt ingeven bij shaders, moet dit een getal tussen 0 en 1 zijn (en dus niet tussen 0 en 255). Als je toch enkel de RGB waarden tussen 0 en 255 hebt, moet je dit delen door 255.
- **Technisch**: 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. Kies wel color want met surface color geeft ook de schaduwen.
- **Technisch**: 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).
- **Technisch**: Met speedtree kan je ook gras maken wat minder zwaar lijkt te zijn voor de processor.
- **Technisch**: Hoe je met terrains werkt (textuur schilderen, hoogtes aanpassen, trees schilderen…)
- **Technisch**: als je tree collider aanzet op een terrain en je schildert gras, kan kunnen er onverwachte collisions ontstaan.
- **Technisch**: Hoe 3D animaties in Unity te gebruiken en op te splitsen
- **Technisch / creatief**: een 3D model dat je wilt animeren, hoeft niet uit één perfecte mesh te bestaan. Je kan de verschillende objecten aan elkaar connecteren.
- **Technisch / Creatief / Persoonlijk**: de controller had steeds een nieuw design nodig omdat mijn materialen steeds veranderden. Dit zorgde voor een uitdaging waardoor er tijdens het bouwen, steeds een nieuw ontwerp in mij opkwam.

- **Technisch**: zelfgemaakte 3D modellen werken met een scaler. Dus je kan niet localscale gebruiken om hun effectief volume weer te geven
- **Technisch**: je moet oppassen bij mesh colliders en modellen met heel veel polygonen. Hiervoor moet je inflate mesh gebruiken zodat de collider net groter is dan je model.
- **Technisch**: als je door backface culling geen polygon kan zien (bv. je zet een plafond, maar geen dak), dan gaat het licht door je model heen

Zelfevaluatie

Ik heb hetgeen kunnen maken wat ik op voorhand had gemeld aan de lectoren: 1 groot level. Daarnaast heb ik ook nog een menulevel en een soort van tutoriallevel kunnen maken.

Het eerste level heeft ook zijn eigen introvideo, waarin wordt getoond waarom de mestkever actie neemt om verschillende voorwerpen op te rollen. Het is zo geanimeerd dat ze op de muziek wordt afgespeeld.

De mestkever is bij het tonen aan de lectoren, ook vaak nog veranderd van design: het was de bedoeling om een erg realistisch mestkever te modellen, maar ik had daar veel problemen mee. Ze zag er nooit uit als een kever, maar eerder als een blokachtig misvormd wezen. Ik had in het begon al beslist dat ik de omgeving redelijk realistisch wou maken, maar de figuren hoefden dat niet te zijn. Daarnaast kreeg ik ook een voorkeur voor de cartoonversie van de mestkever.

Het was leuk en uitdagend om zoveel mogelijk uit Unity te halen, met de tijd die er voor handen was. Zo ontdek je nieuwe oplossingen en mogelijkheden.

Soms duurde het modellen van enkele models zo lang, dat ik het niet eens merkte dat het al middag of avond was. Misschien model ik te traag, omdat ik wil dat alles er perfect uitziet. Aan de andere kant heb ik wel kleine zijweggetjes gevonden om het modellen te versnellen, terwijl het er goed uitziet.

Ik vind het wel spijtig dat ik niet alles heb kunnen doen wat ik wou: ik wou dat de dieren ook nog rondliepen, maar daar had ik geen tijd meer voor.

Verbeteringen en toevoegingen (als je nog 6 maanden extra had)

- Extra zachte afgeronde randen voor op de bescherming van de controller, zodat de speler zich zeker nooit kan bezeren.
- Level 1 zou betere gebouwen rond het park krijgen, gebouwen die je automatisch terug herkent in level 2.
- Level 2 zou zich verderzetten op de straat, waar je dan verkeerslichten, auto's, fietsers en uiteindelijk gebouwen zou kunnen oprollen. Het doel zou zijn om de mens die je bal heeft weggeschopt, zijn huis op te rollen om het level te winnen.
- Level 3 zou nog een stap verder gaan. Nu de mestkever zich oppermachtig voelt, rolt hij steden, bergen, bekende bezienswaardigheden (bv. Eiffeltoren, Atomium, Arc de Triomphe, operahuis van Australië, toren van Pisa…) en uiteindelijk rol je de planeet zelf op om het level te winnen.

- De animatie van de mestkever zelf zou verbeterd worden. Er zitten hier en daar nog wat stijve bewegingen die er dan uit zouden gehaald worden. Daarnaast zou hij een betere idle pose krijgen dan nu.
- Als je bal kleiner is dan een voorwerp dat je op kan raden en je zou ertegen botsen, dan zouden ze een animatie krijgen. In deze korte animatie zouden ze dan even onstabiel kunnen bewegen.
- Als voorwerpen aan de bal vastplakken, dan zouden ze eventueel een animatie krijgen. Zo zouden de mieren hun pootjes wild kunnen rondzwaaien.
- Als je het level wint, zou je niet meer meet de bal kunnen rond bewegen, maar zou er een overwinningsanimatie zijn. Daarnaast wordt je tijd in het midden van het scherm getoond en wordt er weergegeven of je een snellere tijd hebt neergezet of niet.
- I.p.v. muren maken die langs één kant geen vlak hebben, ervoor zorgen dat je door de muren kan kijken, zoals een soort masker dat dan de muur wegsnijdt.
- Er zou een interessanter (misschien zelfs interactiever) tutoriallevel zijn, waar je zelf niet op volgende of vorige moet drukken om door de turorial geleid te worden.
- De dieren / auto’s zou ik willen laten rondlopen, terwijl ze een animatie hebben.

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.
- **Serialcom for Unity.** https://github.com/DWilches/SerialCommUnity**.** Geraadpleegd op 5 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.
- **Splitting animations.** https://docs.Unity3d.com/Manual/Splittinganimations.html. Geraadpleegd op 8 augustus 2017.
- **Oplossing volume berekenen onregelmatige meshes**. http://answers.Unity3d.com/questions/1306365/mesh-volume-static-batching.html. Geraadpleegd op 10 augustus 2017.

Muziekbronnen

- **Muziek comic video**. Haydn, J., Hogwood, C. Academy of Ancient Music. (2007). *Haydn: Symphony No. 94 'Surprise'; Symphony No. 96 'Miracle'* [cd-rom]. 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.
- **Duck quack sound**. https://www.youtube.com/watch?v=Fw3RB7xnb80. Geraadpleegd op 10 augustus 2017.