Games maken (4)

René Suiker

Terugblik

En zo komen we weer bij elkaar voor Game Design. Vorige keer had ik grootse plannen, maar was de beschikbare ruimte in de SoftwareBus niet toereikend. Als Game Designer kan ik me daar druk om maken, maar als hoofdredacteur van de SoftwareBus weet ik waar we vandaan komen en doet het me deugd dat we ons magazine zo vol krijgen, met aandacht voor zo veel thema's. Dus ik klaag niet, ik probeer in deze reeks telkens vier pagina's te vullen. Misschien iets minder of iets kleinere plaatjes, maar het moet natuurlijk wel begrijpelijk blijven.

Wat kunnen we eventueel doen?

Als ik structureel veel ruimte te kort ga komen, dan houd ik het in deze artikelen iets oppervlakkiger en kan ik natuurlijk online wat verdiepingen plaatsen. In eerste instantie op een afgeschermd deel voor abonnees, maar net als de SoftwareBus zelf na verloop van tijd voor iedereen beschikbaar. Op die manier kan ik misschien mijn verhaal iets makkelijker maken, maar het vergt nog wel enige planning, want ik wil voorkomen dat je telkens tussen papier en Internet moet schakelen om een artikel te begrijpen. Iets voor de toekomst dus, vandaag nog niet.

Programmeren

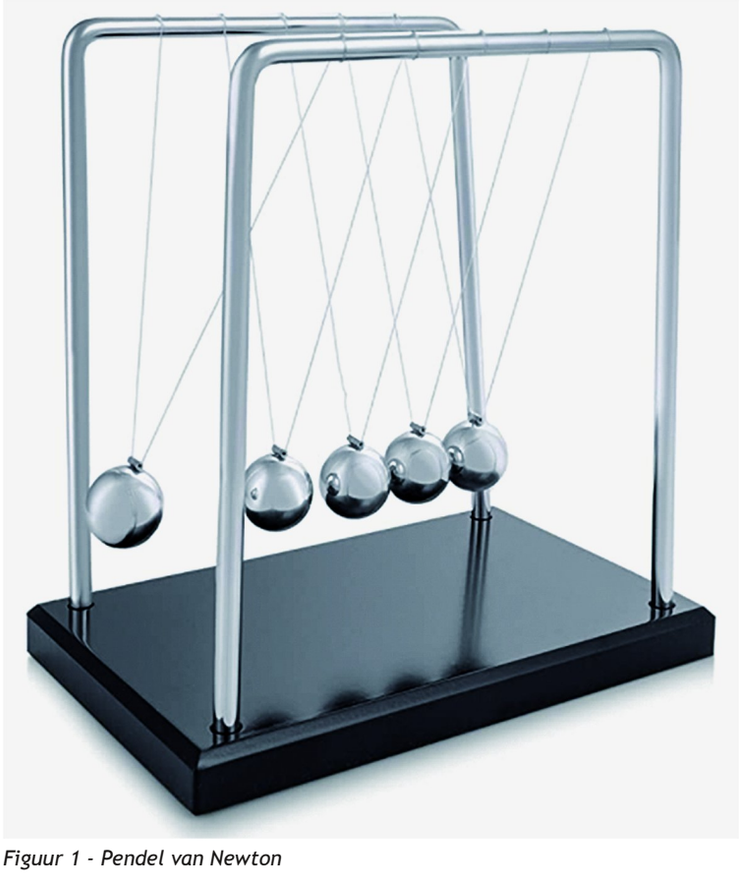
Wij zijn als CompUsers van de toepassingen, we zijn niet de IG Programmeren, maar ik vind het zelf wel leuk. Maar omdat we de expertise niet echt in huis hebben, is deze reeks een samenwerking met de IG Programmeren en deze samenwerking loopt echt op rolletjes. Via de Website van deze IG (**https://programmeren.hcc.nl**/) kun je zien waar ze nog meer mee bezig zijn en als je programmeren leuk vindt, dan kun je bijna elke dag in de week bij hen een online event bijwonen, over verschillende onderwerpen. Ze zijn actief met o.a. Liberty Basic, Visual Basic, Python, Pascal en dus Games. Wel is het zo, dat het verstandig is een aantal keren achter elkaar bijeen te komen, vooral als je nog niet zo veel weet, want de sessies (per thema) borduren voort op wat de week ervoor is behandeld. En af en toe starten ze weer een reeks; er is bijvoorbeeld voor Liberty Basic op 30 maart jl. weer een sessie gestart voor beginnende gebruikers. Zo kun je met een avond in de week je kennis snel naar een hoger niveau brengen. En ik doe dus mee op de vrijdagavond, wanneer Marco Kurvers zijn kennis over Unity met ons deelt. En Marco heeft intussen al menig game zelf uitgebracht, vooral voor Windows, maar intussen dus ook al voor Android. Daar valt voor ons en zeker voor mij dus nog een hoop van te leren.

Planning

De plannen die we hadden voor deze reeks artikelen is intussen ietwat bijgesteld. Deels omdat de leercurve aan de ene kant wat minder steil is dan verwacht, aan de andere kant omdat het toch wel interessanter is om de C# (spreek uit: 'see sharp') te gebruiken. Dus met ingang van volgend artikel beginnen we daar ook mee. Wel is het plan nog steeds om ons dit jaar te beperken tot twee dimensionale scenes/projecten. En bij leven en welzijn gaat de reeks volgend jaar nog door. Het is dan wel zaak dat CompUsers doorgaat, maar dan als gewone interessegroep binnen HCC.

Deze keer

De vorige keer hebben we de wetten van de mechanica een beetje getart en hebben we wat fysieke onmogelijkheden uitgehaald. Deze keer wilden we wat dichter bij de natuurwetten blijven. Daarbij zal ook blijken dat Unity de natuurwetten benadert. Dat ligt misschien niet alleen aan Unity, maar misschien gedeeltelijk aan onze beperkingen om het allemaal te modelleren, maar uit de diverse meldingen binnen de Unity-community lijkt het erop dat niemand dit tot nu toe perfect heeft kunnen modelleren. Wat was namelijk het plan: de pendel van Newton.



Figuur 1 - Pendel van Newton

Ik denk dat we allemaal wel weten wat dit is. Op basis van de wetten van de mechanica hangen alle vijf de ballen recht naar beneden en hangt alles stil in de rusttoestand. Maar het feitelijk effect is dat als je één bal pakt en je aan de slinger omhoog beweegt en dan loslaat, dan valt hij naar beneden, komt tegen de volgende bal aan, die geeft de kracht door enzovoort tot de laatste bal, die kan de kracht niet doorgeven en gaat dus zelf omhoog langs de slinger. Daarna, op grond van de zwaartekracht, keert die weer terug, zorgt voor weer een botsing, blijft hangen maar dan gaat aan de andere kant de eerste bal weer omhoog. En als er geen demping van de beweging zou zijn op basis van wrijving zou dit proces zich eeuwig herhalen. Nu is er altijd wrijving en luchtweerstand, dus het dempt uit. En als je aan de start twee ballen tegelijk pakt en loslaat, zie je aan de andere kant twee ballen omhoog schieten en weer terugvallen. Hetzelfde effect, maar dus met dubbele kracht. En dit wilden we dus nabouwen binnen Unity, zonder nog code te moeten gebruiken. Spoiler alert: dat valt nog niet mee en lukt ook niet helemaal.

Huiswerk?

Hoewel ik wel huiswerk heb opgegeven en ik daar ook zeker wel op in wil gaan, slaan we dat nu even over. Het komt nog wel, geen paniek. We kijken straks nog even terug op onze deeltjesversneller. De wijziging in volgorde komt een beetje voort uit het feit dat de acties niet meer synchroon lopen tussen Unity en Scratch en dat gaat ook niet meer gebeuren. Zoals ik al eerder aangaf: er zijn grote verschillen tussen de twee en met Unity kan veel meer, maar met Scratch gaan we veel harder. In deze SoftwareBus trouwens weer zowel Scratch als Unity, maar het spel dat ik bij Scratch beschrijf, dat kan in Unity veel mooier, maar daar zijn we bij Unity nog niet aan toe.

Nog een stapje verder terug

Ik gaf al aan, dat Unity een stuk complexer is dan Scratch en daarom is het misschien toch goed even een stapje terug te doen en eens goed naar het concept te kijken. Allereerst is Unity wel een game engine, maar het wordt niet alleen daarvoor gebruikt. In feite kun je er veel meer mee, maar we richten ons in deze reeks op Unity als game engine. En misschien is het dan toch goed om daar eens iets dieper op in te gaan.

Wat is een game engine?

Volgens Wikipedia is een engine de softwarematige basis van een computerprogramma. En bij computerspellen wordt het dus meestal de ‘game engine’ genoemd. In feite is engine niets anders dan het Engelse woord voor motor en de game engine is in feite de motor voor een spel. Verder gaat Wikipedia in op de verschillende modules die in een game engine kunnen zitten, waar we later ook mee te maken krijgen.

Zo heb je:

- De rendering engine, die berekent wat op het scherm moet komen;
- De physics engine, die ervoor zorgt, dat objecten zich enigszins natuurgetrouw gedragen (of juist niet, als je een kunstmatige wereld wilt creëren);
- De scripting engine, die alle acties reguleert.

Overigens is Unity niet de enige game engine, maar wel een goede en veel gebruikte. Voor meer details verwijs ik graag naar **https://nl.wikipedia.org/wiki/Engine** waar veel uitleg te vinden is.

Een spel, wat komt erbij kijken?

In het kader van stapjes terug: we moeten bij het maken van een spel niet gelijk de engine in en beginnen te modelleren en programmeren; een goed plan maken is ook bij deze vorm van softwareontwikkeling van belang. In een eerdere fase heb ik al eens uitgelegd dat de wereld van de computerspellen een grote markt is, waar intussen al meer geld in omgaat dan in de filmindustrie. Dat komt niet voort uit de eenvoudige spelletjes die je gratis op je telefoon speelt, alhoewel daar ook al flinke verdienmodellen achter zitten, maar met name in de grote spellen, waar vele duizenden gebruikers meespelen en die daar maandelijks een bijdrage voor betalen.

Maar zo’n spel ontstaat niet even op een achternamiddag, daar is veel bloed, zweet en tranen aan voorafgegaan en ook veel geld in geïnvesteerd. En zo moeten we bij het maken van een spel dus nadenken over het spelconcept, de karakters, het spelverloop en de moeilijkheidsgraad. En dan hebben we het dus ook over hoe je een en ander documenteert, zodat je eenmaal aan het werk ook nog steeds snapt wat je aan het doen bent.

Bij grote spellen liggen deze taken ook niet allemaal bij één persoon, maar ligt elke taak bij een speciaal team. En dan heb je nog projectmanagement, om alle opleveringen op elkaar af te stemmen. En moderne ontwikkeling gaat uit van een eerste release en dan steeds toevoegingen.

Op die manier kun je al een beperkt spel op de markt brengen en daar uitbreidingen aan toevoegen. En die uitbreidingen, die liggen soms al vast, maar vaak komen daar nog nieuwe inzichten bij en kunnen prioriteiten in de loop van de tijd veranderen.

Het plan is gewijzigd

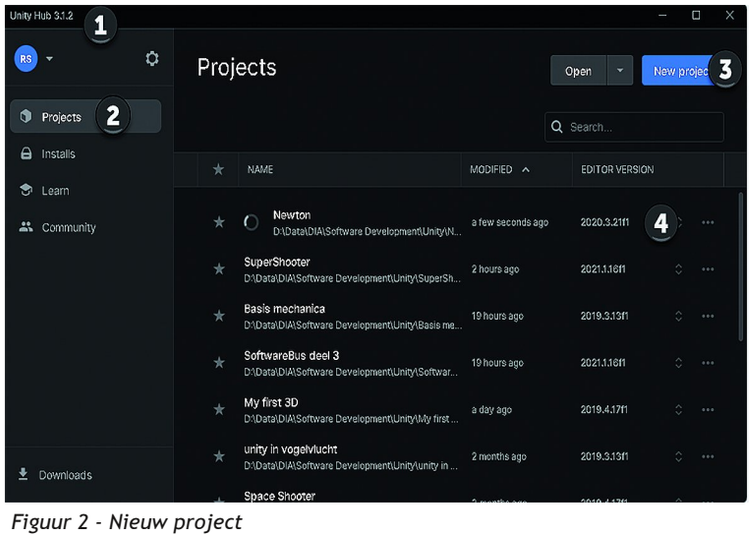
We hadden voor de uitleg over Unity een mooi plan bedacht, waarbinnen de pendel van Newton mooi paste en we gaan hier nog wel op in, net als op het huiswerk, maar het plan is nu iets veranderd. In de loop van deze reeks gaan we een echt spel maken, een first person shooter (FPS), in 3D. Een FPS is een spel waarin de speler zich vereenzelvigt met de held en hij het beeld beziet door de ogen van de spelfiguur, alsof hij dat dus zelf is. En dat shooter slaat er dan op dat hij vijanden om zeep moet helpen.Los daarvan laat ik soms wat concepten zien, zoals de deeltjesversneller en de pendel van Newton, om te laten zien wat je zoal kunt zonder te programmeren.

In het originele plan zouden we pas in december met scripts (in C#) gaan beginnen en pas volgend jaar in 3D, maar ik denk dat het nieuwe concept iets aantrekkelijker is, omdat we gelijk al naar iets tastbaars toewerken.

Maar goed, we hadden een plan en we hebben een gewijzigd plan, ook dat kom je in softwareontwikkeling vaker tegen. En nu dus even naar het huiswerk, dan naar de pendel en dan een begin maken met onze FPS-game

Het huiswerk van de vorige keer

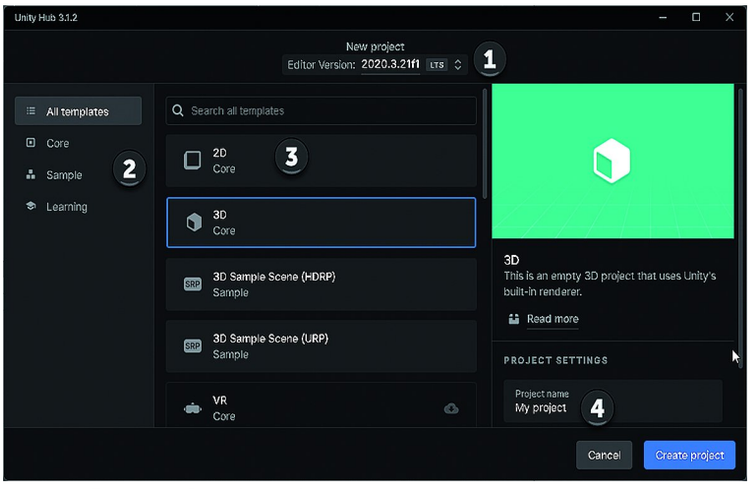
Weten we het nog? We moesten het startscherm maken van onze deeltjesversneller. We moesten de ruimte maken en een balletje dat er doorheen geschoten werd. We beginnen met het opstarten van de Unity Hub:



Figuur 2 - Nieuw project

Je kiest in de Hub (1) voor het blad ‘Projecten’ (2) en daarbinnen kies je voor ‘Nieuw Project’ (3). Als het project gemaakt wordt zie je het al verschijnen in het projectoverzicht (4).

Als je knop (3) indrukt krijg je de gelegenheid details over je nieuwe project op te geven:

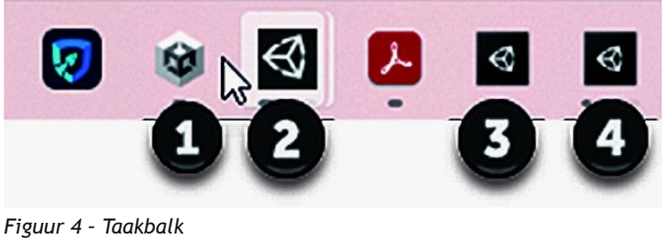


Figuur 3 - Details van het nieuwe project

Bij (1) kies je de versie van Unity die je wilt gebruiken. Binnen de Hub kun je meerdere versies van Unity installeren, zie een vorig artikel in deze serie. Bij (1) kun je uit de geïnstalleerde versies kiezen welke je hier wilt gebruiken. Daarbij kun je een template uitkiezen bij (3). Bij (2) kun je de beschikbare templates filteren.

Bij (4) vul je de projectnaam in en daaronder (viel even buiten beeld) kun je ook nog de folder opgeven waarin je project wordt opgeslagen.

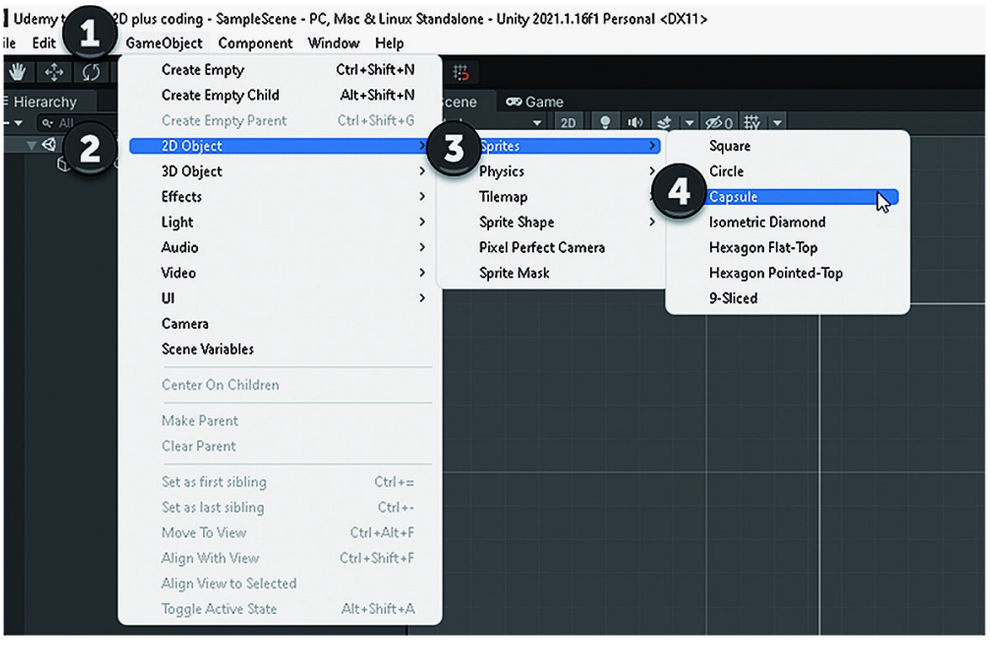
Je kunt trouwens vanuit de Hub verschillende projecten tegelijk openen. Voor zover ik het nu kan beoordelen worden die op de taakbalk per Unity versie gegroepeerd, dus maak je met verschillende versies van Unity projecten, dan hebben ze allemaal een eigen icoontje op de taakbalk, en heb je vier projecten open vanuit dezelfde Unity versie, dan zie je maar één icoon op de taakbalk, maar daarbinnen zie je dus wel de verschillende instanties:



Figuur 4 - Taakbalk

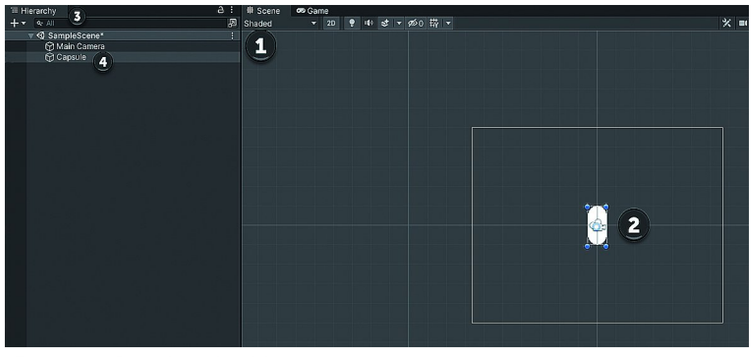
Bij (1) zie je het icoontje voor de Unity Hub. Bij (2) zie je een icoontje dat toevallig twee Unity-projecten omvat, bij (3) zie je een andere Unity versie, die maar één project open heeft en bij (4) zijn op dit moment twee projecten actief, die beide nog geopend worden. Het aantal actieve projecten per icoon kun je zien door met de muis erboven te hooveren. Helaas lukte het me niet om dat in een screenshot te vangen, maar u weet wel hoe dit werkt.

We hadden vorige keer beschreven: we gebruiken capsules voor de wanden. Die maak je aldus:



> *Figuur 5 - Capsule maken*

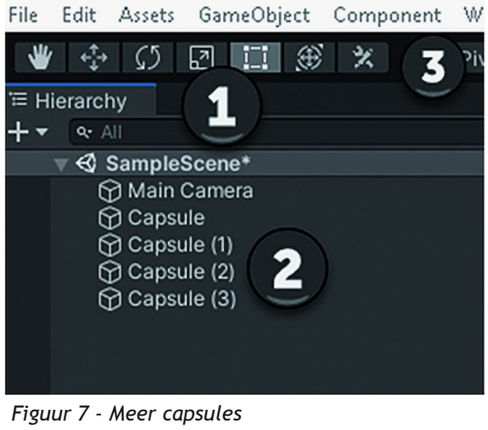
Je gaat dus naar het menu 'GameObject' (1), daarbinnen kies je '2D object' (2), daarbinnen kies je 'Sprites' (3) en daarbinnen kies je 'Capsule' (4). Als je dat doet verschijnt een capsule in beeld en wordt deze ook opgenomen in de hiërarchie:



Figuur 6 - Capsule in beeld

Je ziet hier binnen de scene (1) de capsule (2) afgebeeld. Daarnaast zie je in de hiërarchie (3) de capsule opgenomen (4). Als je heel goed kijkt zie je in de scene dat de capsule ook geselecteerd is en in de hiërarchie zie je dat ook, omdat de regel geselecteerd is.

Op dezelfde manier kun je meerdere capsules aanmaken, want we hebben er vier nodig. Maar het kan ook eenvoudiger. Klik in de hiërarchie met de rechtermuisknop op de capsule en klik dan op 'copy'. Klik vervolgens ergens eronder met rechts en kies dan 'paste'. Doe dat drie keer en je hebt vier capsules staan:



Figuur 7 - Meer capsules

Bij (1) zie je nog even dat we in de hiërarchie kijken. Bij (2) zie je de diverse capsules staan. In de scene zie je trouwens maar één capsule staan, althans bij mij is dat het geval, maar ze zijn er echt allemaal, al staan ze achter elkaar. Bij (3) zie je trouwens enkele bewerkingssymbolen voor het actieve game-object. Met een game-object geselecteerd kun je deze trouwens ook met een toets bereiken, en wel met QWERTY respectievelijk.

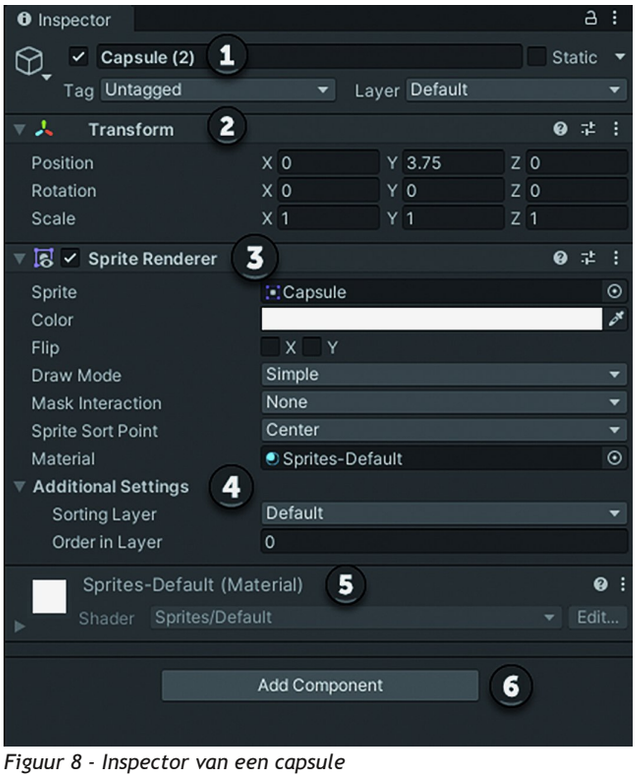
De bewerkingen zijn:
1. ***View***
2. ***Move***
3. ***Rotate***
4. ***Scale***
5. ***Resize***
6. ***Gizmo***

Met View kun je het object bekijken, met Move kun je het door de scene verplaatsen, met Rotate kun je het laten draaien. Met Scale en Resize kun je de omvang veranderen, het subtiele verschil daartussen komt in een later verhaal aan de orde. Met de Gizmo kun je meerdere bewerkingen tegelijk uitvoeren.

We gaan nu stuk voor stuk de vier capsules aanpassen, zodat de eerste de linker zijwand wordt, de tweede wordt de rechter zijwand, de derde wordt het plafond en de vierde wordt de vloer. Overigens, de objecten kopiëren kan dus wel in de hiërarchie, maar niet in de scene.

Klik op de bovenste capsule in de hiërarchie en druk vervolgens de W in. Daarmee wordt dus de Move-bewerking ingeschakeld. Je ziet dan in onze 2D-scene een rode pijl naar rechts en een groene pijl naar boven, voor bewegingen langs de X- en de Y-as. Aangezien de eerste capsule de linker zijwand wordt, schuiven we deze capsule naar links. Dat doe je door op de rode pijl te klikken en dan deze naar links te slepen. De capsule beweegt mee. Selecteer vervolgens de tweede capsule en beweeg deze naar rechts, ook met de rode pijl.

De derde capsule schuif je naar boven en de vierde naar beneden met behulp van de groene pijl. Als je trouwens één van de capsules hebt geselecteerd, bijvoorbeeld de derde, dan kun je in de inspector rechts de gegevens van deze capsule bekijken (en bewerken):



Figuur 8 - Inspector van een capsule

De inspector geeft je ontzettend veel mogelijkheden, zeker als we al wat verder zijn met Unity, om de werking van je spel te beïnvloeden. Hier kunnen we bij (1) de naam aanpassen. Hier vullen we nu 'Plafond' in. Daarmee kun je later ook in de code aan deze sprite refereren. We maken nog geen code in dit verhaal, maar in de toekomst gaan we dat dus wel doen. Het is dan aan te raden om elk object zijn eigen naam te geven.

Bij (2) geven we wat transformatieparameters op. In dit geval zijn dat de positie, de rotatie en de schaal. We zien dat in dit plaatje de Y een positieve waarde heeft, dat wil zeggen dat hij boven het midden staat. De X is 0, dat wil zeggen, dat hij in horizontaal opzicht precies in het midden staat. De Z staat op 0, maar dat is in 2D niet zo relevant. De Z-as speelt echter wel degelijk een rol, met name bij rotatie. Overigens speelt de Z-positie geen rol bij het voor of achter elkaar langs gaan of staan, daarvoor heb je de 'sorting layer' en de 'order in layer' bij de additionele instellingen (4). Bij (3) heb je de sprite renderer, die bepaalt dus hoe de sprite wordt weergegeven. Hier kijken we nu nog even niet naar. Bij (5) kun je materiaal van de sprite bepalen, ook daarop komen we pas in een later stadium terug. Ten slotte zie je bij (6) de knop om componenten toe te voegen.

Alles is in Unity een game object en een game object wordt bepaald door zijn eigenschappen. En bij de eigenschappen horen ook weer componenten die je de mogelijkheid geven extra eigenschappen aan een object toe te kennen. Dit gaan we allemaal nog verder uitwerken in volgende artikelen. Nog heel veel te doen dus voordat we een echt spel hebben. Ik zei al eerder, dat gaat in Scratch sneller en eenvoudiger, maar daar heb je dan wel minder mogelijkheden bij.

Als we vervolgens bij het plafond de Y-waarde op 4 instellen, de rotatie bij Z op 90 en de scale bij X een waarde 0.5 en bij Y een waarde 5 geven, dan hebben we het plafond iets dunner gemaakt, iets breder gemaakt en hem dus plat gelegd. We hadden natuurlijk de rotatie niet nodig, we hadden hem ook gewoon kunnen rekken en duwen, maar dit effect is mooier.

Kiezen we vervolgens de bovenste capsule in de hiërarchie en vullen we die in met positie X = -5 en Y = 0 en scale X = 0.5 en Y = 4 dan sluit deze mooi aan op het plafond. Noem deze capsule iets als Zijwand_L.

Vul vergelijkbare waarden in bij de tweede capsule uit de hiërarchie. Noem deze Zijwand_R en vul dan bij positie X = 5 in.

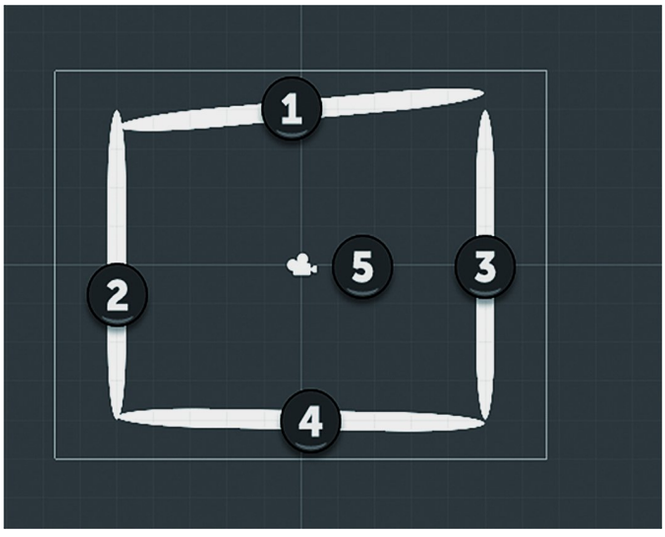
Ten slotte nog de vloer, de laatst overgebleven capsule.

Met deze waarden is het veld gesloten en kunnen we een bal erbinnen laten bewegen. Alleen, we wilden een schuine vloer, om de bal straks niet recht te laten stuiteren. En we wilden ook een gaatje om de bal te laten ontsnappen. Daarvoor hebben we dus nog wat kleine aanpassingen nodig.

Dit kunnen we vrij eenvoudig doen door het plafond in plaats van 90 graden over de Z-as te laten draaien, er vijf graden bij op te tellen, dus 95 graden. Dan hebben we een schuin plafond en gelijk een opening rechtsboven.

Als we ook de vloer schuin willen laten lopen en geen echte opening willen creëren, kunnen we de hoek iets minder aanpassen. De vloer ligt dan niet recht, dus een stuiterende bal krijgt een lichte afwijking, maar niet zoveel als op het plafond. Voor ons doel doet dat er nog niet echt toe, we zijn nog niet zo ver dat we ons daar druk om maken. Uiteraard, als we straks met echte games bezig zijn, dan moet je je na de prototype-fase ook druk gaan maken om hoe het spel eruit ziet, want anders wil niemand het spelen. Maar het is eerst zaak om alles feitelijk aan het werk te krijgen. En dan komt de rest later wel.

Ik heb de vloer dus maar 89 graden laten draaien en dan zie je dus het beeld zoals hieronder weergegeven:



Figuur 9 - De versneller

Bij (1) zie je dus het plafond, bij (2) de linker zijwand, bij (3) de rechter zijwand en bij (4) de vloer. De vloer en het plafond lopen niet helemaal kaarsrecht, de wanden staan wel loodrecht. Bij (5) zie je trouwens de camera. Als je die selecteert, zie je rechts onder in de scene het beeld dat de camera ziet. Dat is ook het beeld dat je krijgt als je de scene speelt. Je kunt dat uitproberen door play te activeren. Verder gebeurt er uiteraard niets, het zijn gewoon vier capsules die niet bewegen.

Bij (1) zie je dus dat we in de scene zijn. We hebben de camera geselecteerd. De rotatiebewerking was nog actief, vandaar de cirkels om de camera, maar daar doen we nu niets mee. Rechtsonder zie je het beeld dat de camera ziet (2).

Je ziet dat onze versneller bijna beeldvullend is, in elk geval in verticaal opzicht.

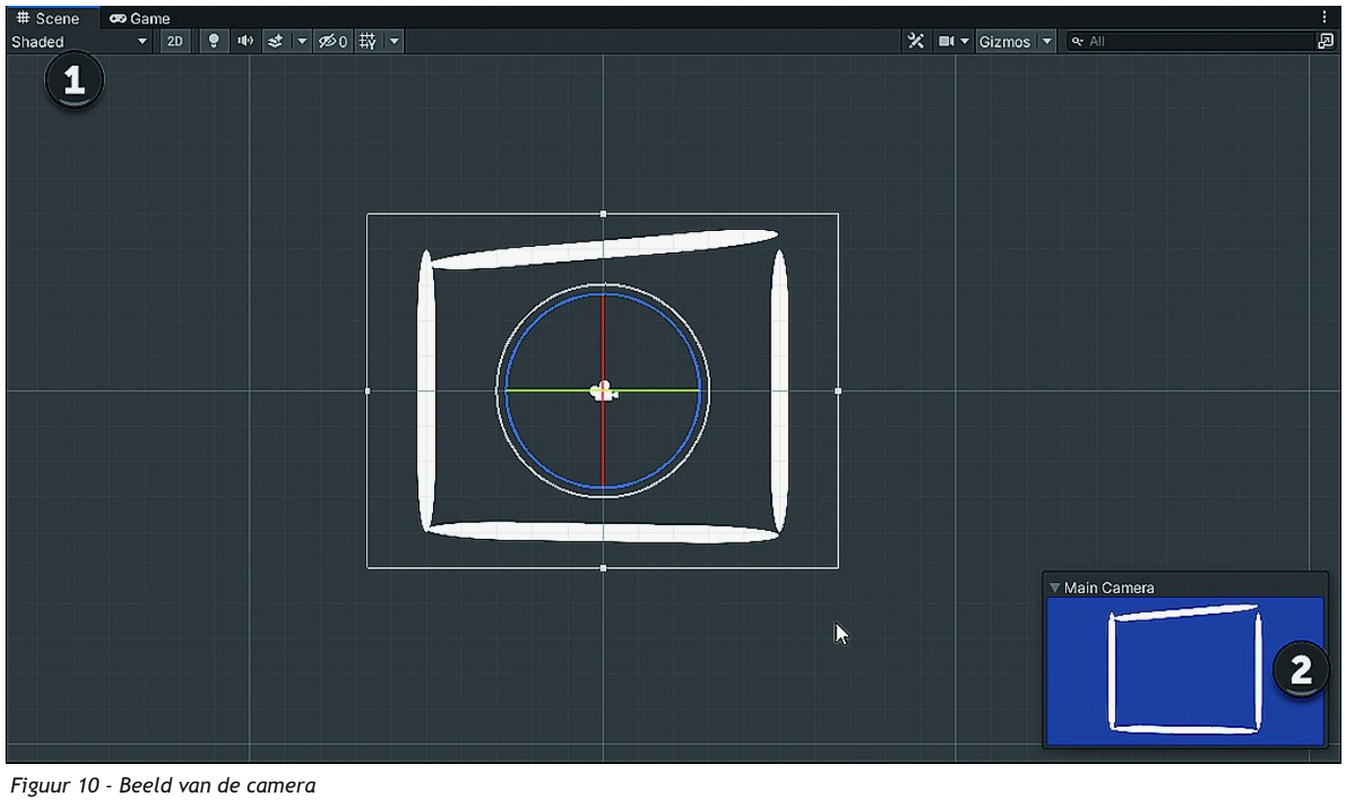
Al met al zijn we nog niet aan de Pendel van Newton toegekomen, dus daar moeten we de volgende keer serieus werk van maken. Het zijn kleine stapjes die je met Unity maakt, omdat er zo veel is om uit te leggen.

De volgende keer ga ik wel even verder met deze versneller. We brengen een bal in het veld en laten die dan stuiteren volgens de zwaartekracht én steeds iets harder stuiteren, zodat hij vervolgens steeds sneller gaat. Komt hij dan bij de uitgang, dan moet hij al een heel behoorlijke snelheid hebben.

Daarnaast gaan we de volgende keer even iets uitleggen over de Pendel van Newton en we gaan alvast een klein stapje zetten in de richting van onze FPS. Al met al best wel iets om naar uit te kijken, hoop ik.

Huiswerk

Het huiswerk voor deze keer is simpel. Maak het project na dat ik hier heb gedaan en zet alvast een balletje in het spel. Maak dat balletje rood en zo groot, dat het eventueel wel door de uitgang kan schieten.



Figuur 10 - Beeld van de camera