BOLKE vs BSA

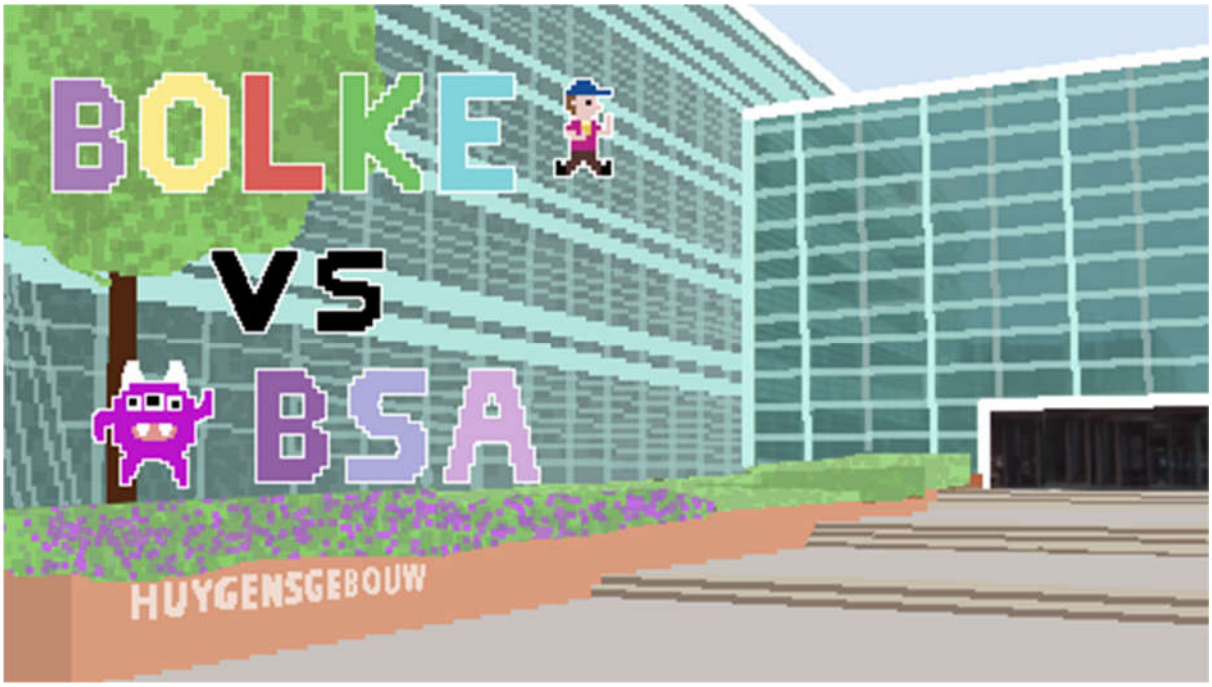
Astrid van der Jagt – s4571037

Taras Khrystenko – s4556429

Sean Snel – s4421116

Nienke Wessel – s4598350

Juni 2016



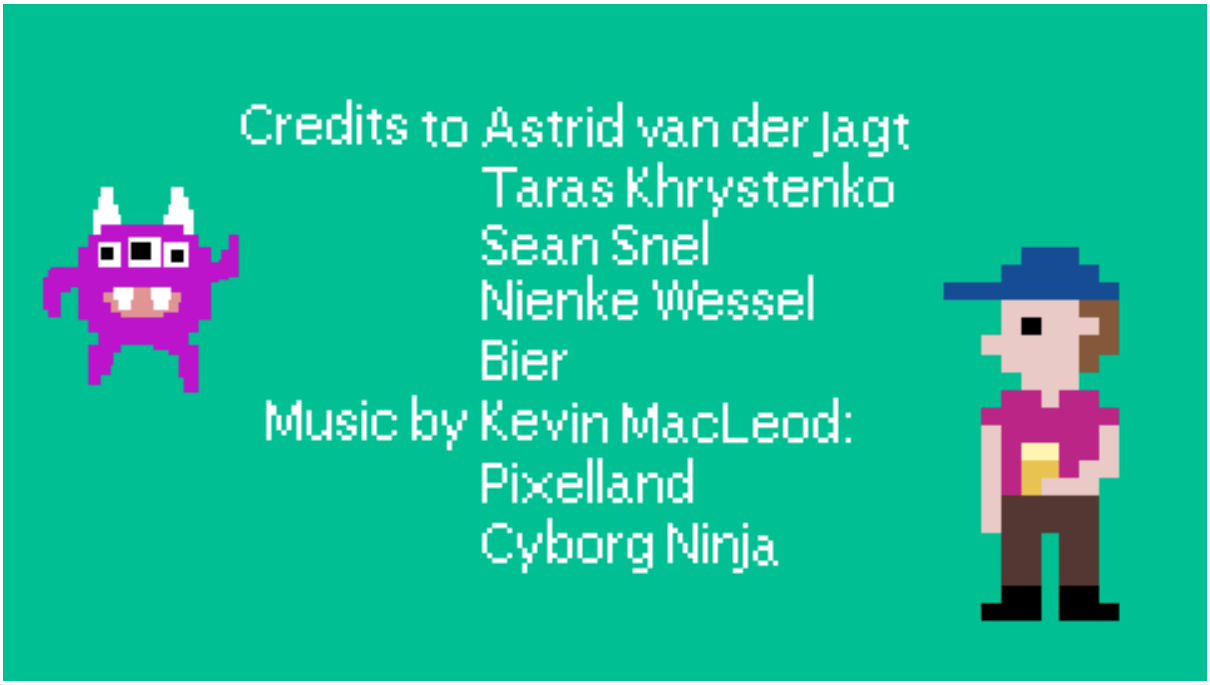
Voorwoord

Na vijf weken hard werken was het dan eindelijk zover: de app was af. Bolke vs BSA is een platformer geworden waar wij toch best een beetje trots op kunnen zijn (vinden wij zelf). En daarom hebben wij dit verslag gemaakt, waarin wij vastleggen wat de app is en hoe we de app hebben opgebouwd.

Eerst geven we een beschrijving van ons product met de belangrijkste eigenschappen. We geven aan waarom het bouwen van deze app de moeite waard was en wat ons spel toevoegt aan de bestaande, vergelijkbare spelletjes. Ook gaan we kort nog even in op de functionele en niet-functionele eigenschappen, aan de hand van use cases.

Vervolgens zullen we wat dieper ingaan op het ontwerpproces. Omdat we in de game engine Unity hebben gewerkt is ons ontwerpproces anders dan andere projecten. We zullen toelichten hoe dat werkte, welke componenten we gebruikt hebben en hoe deze componenten samenwerken. Daarnaast zullen we ons ontwerp verantwoorden.

Tenslotte hebben wij in ons document ook nog een reflectie toegevoegd. Dit gaat niet zo zeer over de kenmerken van onze app, maar onze persoonlijke kijk op de app zelf en op het ontwerpproces. We zullen ingaan op de problemen waar we tegenaan zijn gelopen, maar natuurlijk ook op de dingen waar we trots op mogen zijn.



Inhoud


LOADING...

Beschrijving

Inleiding

Bolke vs BSA is een gezellig, vrolijk spelletje. De categorie waartoe het spel behoort kan beschreven worden als 'Platformer'. De speler springt van platform naar platform om zo het einde van het level te bereiken. De gebruiker speelt als Bolke, die als student op de Radboud Universiteit studiepunten (ec) verzamelt en verschillende vijanden verslaat, om vervolgens het BSA monster te verslaan. Het speelt zich af in het Huygensgebouw, waar Bolke van platform naar Platform springt.

Productverantwoording

Bolke vs BSA is een gezellig spelletje. Het is leuk om te spelen. Het lijkt een beetje op de oude 8bit spelletjes van Super Mario Bros die we allemaal kennen, en draagt dus ook een beetje nostalgie met zich mee. Door de vrolijke kleuren, het muziekje en de geluidseffecten geeft het een vrolijke indruk.

Toch onderscheidt het spel zich duidelijk van de spelletjes die al bestaan doordat het studentgericht is. Als student kom je herkenbare dingen tegen, zoals het biertje dat Bolke altijd bij zich heeft, het feit dat hij studiepunten (ec) moet verzamelen en dat er met Heineken bier gegooid wordt. Dit spel is specifiek leuk voor de studenten die het Huygensgebouw kennen. Onze app voegt dus wat toe aan het app-landschap door zich specifiek te richten op de student en dan vooral de Radboud-studenten. Het herkenbare Huygensgebouw op het beginscherm en de achtergrond maken deze app uniek in z'n soort.

Specificaties

De gebruiker speelt als Bolke, een student op de Radboud Universiteit. Bolke springt van platform naar platform door het Huygensgebouw. Op de achtergrond is een vrolijk muziekje te horen. Op Afbeelding 1 is de start staat van het spel te zien. Bolke kan naar links, naar rechts, springen of double jumpen om verder in het spel te komen.

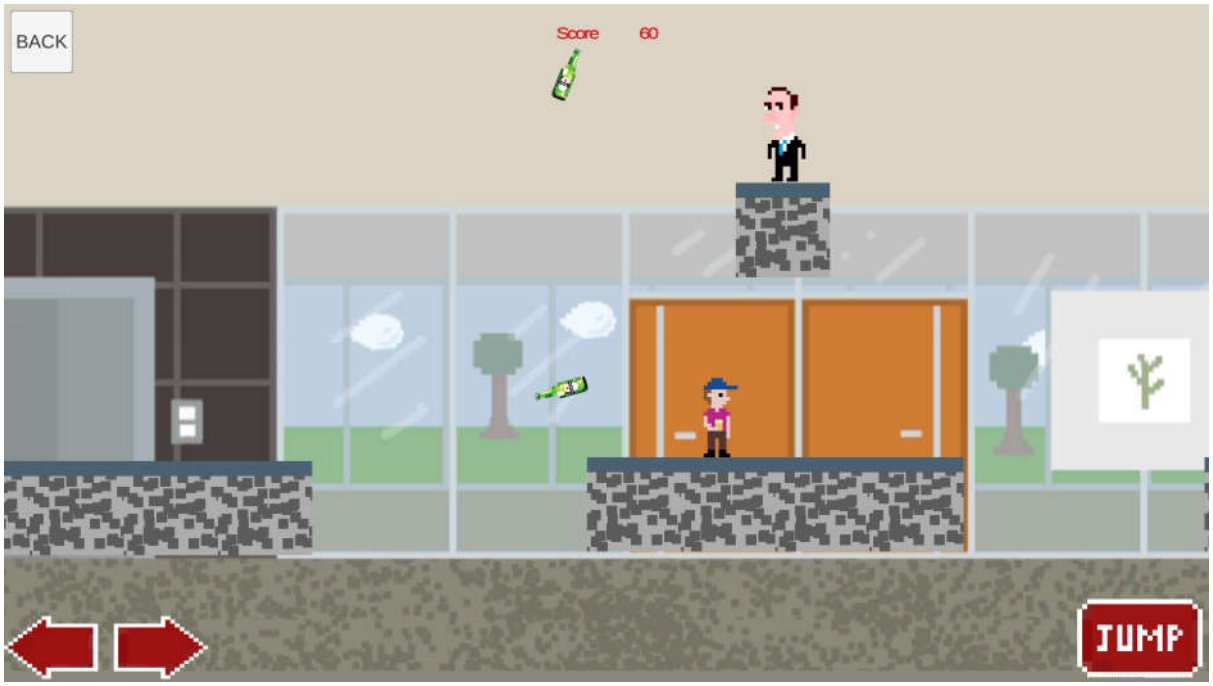
Afbeelding 1 – De startpositie van Bolke.

Onderweg komt Bolke verschillende vijanden tegen. De eerste vijanden die Bolke tegenkomt zijn de schoonmaakkarretjes (Afbeelding 1). Deze rijden op en neer. Als Bolke ze raakt moet hij het level opnieuw beginnen. Hij kan de karretjes ontwijken, of op ze springen om ze dood te maken. De karretjes verdwijnen dan.

De tweede vijand die Bolke tegenkomt is de vliegende docent (Afbeelding 2). Deze docent gaat op en neer. Ook bij deze vijand geldt dat Bolke hem niet mag raken, omdat hij anders opnieuw moet beginnen. Hij kan hem verslaan door op hem te springen, maar kan er ook voor kiezen hem te ontwijken en door te gaan.

Afbeelding 2 - De vliegende vijand

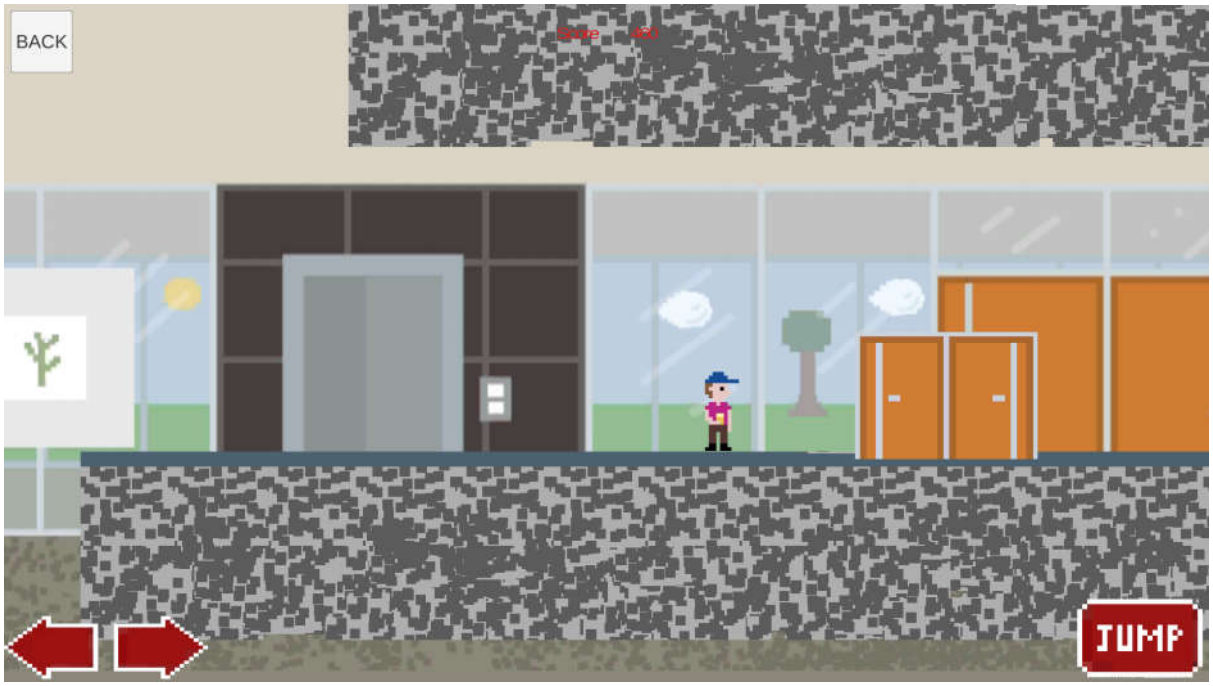
Verder komt Bolke ook een andere vijand tegen, de docent die Heineken bier in zijn richting gooit (Afbeelding 3). Als Bolke de docent of het bier raakt, is hij game over en moet hij opnieuw beginnen. Hij kan deze vijand niet verslaan, maar moet hem en de biertjes ontwijken.



Afbeelding 3 - De Heineken-bier gooiende vijand

Naast vijanden komt Bolke onderweg ook studiepunten (ec) tegen, die hij kan verzamelen (Afbeelding 4). Bij het oprapen van de muntjes wordt zijn score verhoogd met 10 punten.

Afbeelding 4 - ec

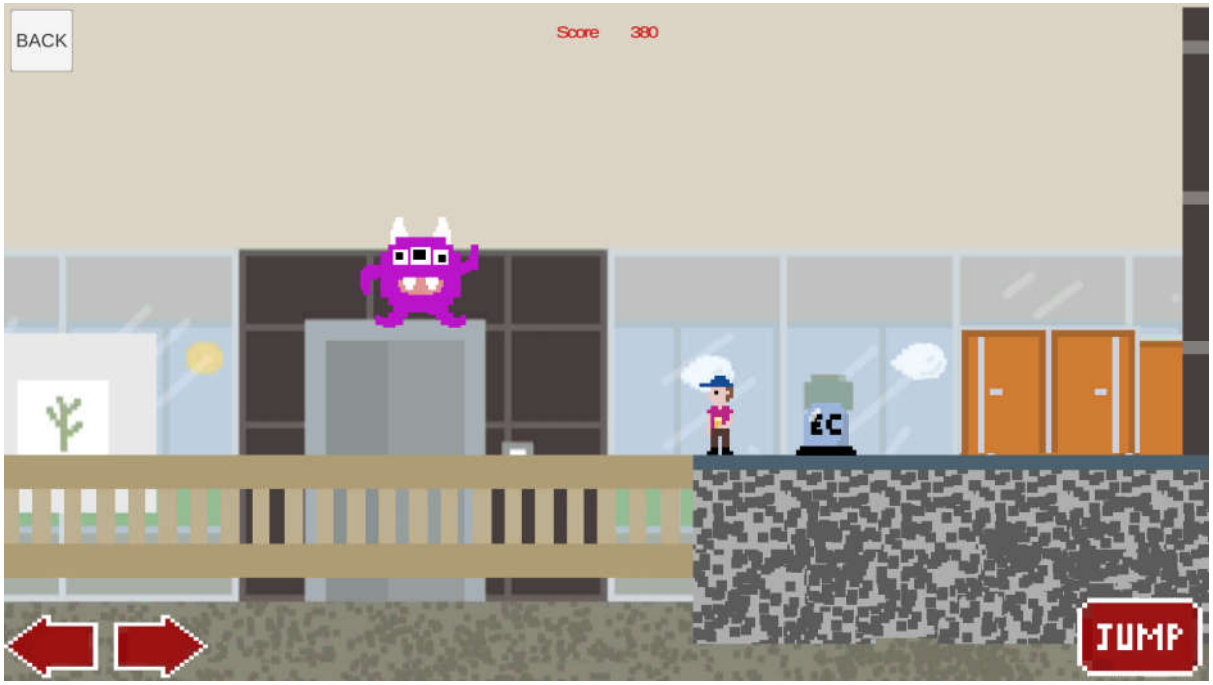


Afbeelding 5 - Bolke bereikt de deur die naar het BSA monster leidt

Als Bolke het level succesvol doorlopen heeft bereikt hij een deur (Afbeelding 5). Als hij door deze deur gaat komt hij bij de eindbaas: het BSA monster (Afbeelding 6). Het BSA monster loopt en springt heen en weer. Bolke mag hem niet raken. Doet hij dit wel, moet hij weer helemaal opnieuw beginnen.

Afbeelding 6 - Bolke is bij het BSA monster

Bolke kan het BSA monster verslaan door langs hem heen te lopen naar de knop met EC erop (Afbeelding 7).



Afbeelding 7 - Bolke is onder het BSA monster gelopen en heeft de knop bereikt

Vervolgens moet hij op de knop springen, waardoor het platform waarop het monster staat verdwijnt en het monster daarmee ook (Afbeelding 8). Dan gaat Bolke door de laatste deur en is het spel uitgespeeld.

Afbeelding 8 - Bolke heeft op de knop gedrukt waardoor het platform waar het BSA monster op staat verdwijnt. Als Bolke door de deur gaat heeft hij het spel uitgespeeld

Ontwerp

In dit stuk van het verslag bespreken wij het ontwerp van onze app. Dit doen wij op een iets andere manier dan voorgesteld is in de opdracht, aangezien wij onze app met de game-engine Unity hebben gemaakt. Daardoor is er niet echt sprake van klassen of methoden of attributen of maar in zeer beperkte mate in verschillende delen van ons ontwerp. Allereerst volgt daarom een kort stukje over hoe Unity heel globaal in elkaar zit zodat we daarna kunnen uitleggen hoe wij dit in ons ontwerp hebben gebruikt.

In de rest van dit hoofdstuk leggen wij uit hoe wij van deze opbouw van Unity gebruik hebben gemaakt bij het maken van onze app. In globaal ontwerp zullen we de verschillende scenes behandelen en uitleggen waar elke scene voor dient. In detailontwerp zullen wij de belangrijkste prefabs langs laten komen. Ten slotte zullen wij uitleggen waarom wij bepaalde keuzes hebben gemaakt.

Unity

In Unity maakt men geen klassen waar stukken codes aan toegevoegd worden, maar maakt men 'prefabs'. Dit is vergelijkbaar met een klasse, maar niet helemaal hetzelfde. Een belangrijk verschil is dat een prefab op een andere manier wordt bewerkt. Het grootste deel van de eigenschappen van een prefab pas je aan in de GUI. Je kiest welke eigenschappen je wilt dat jouw prefab heeft en dan kan je die aanpassen zodat ze precies voldoen aan jouw eisen. Bovendien kan je daarna een image toevoegen zodat je prefab ook een afbeelding heeft, mocht dat nodig zijn. Bij prefabs kan men denken aan concrete objecten zoals een karakter in een game of een muur, maar ook muziek, de achtergrond en de user controls zijn prefabs. Het voordeel van een prefab, net zoals een klasse, is dat je het maar één keer hoeft te maken en er daarna meerdere objecten van kan maken en gebruiken.

Een ander belangrijk verschil is dat er weinig methoden zijn die wij zelf gemaakt hebben. Een groot deel wordt door Unity gegenereerd en zien wij nooit als daadwerkelijk code-fragment. We hebben wel stukken code, maar dat zijn allemaal korte scriptjes die aan een of meerdere prefabs gekoppeld worden.

Om alle prefabs aan elkaar te koppelen en een echt spel te maken, maakt Unity gebruik van 'scenes'. Elke scene kan je door het klikken en slepen van prefabs zelf in elkaar zetten. Zo kan je dus een of meerdere levels los van elkaar in elkaar zetten, maar kan je wel dezelfde prefabs gebruiken in elk van de levels. En pas je wat aan in een prefab, dan wordt dat automatisch doorgevoerd op elke toepassing van die prefab in elke scene.

Globaal ontwerp

Het globale ontwerp van onze app bestaat eigenlijk uit zes verschillende 'scenes'. Zoals hierboven uitgelegd, is een scene een deel van het spel wat je in elkaar zet met prefabs. In onze app switch je tussen scenes door op buttons te klikken of (wanneer we het hebben over het eindgevecht), een deur door te lopen met je karakter in het level. Wij hebben de volgende scenes gebruikt:

StartScreen

Dit is het startscherm met het Huygensgebouw op de achtergrond. Deze scene is alleen bedoeld om de gebruiker verschillende opties te geven van wat hij wil doen. De opties zijn het spel spelen (andere scene), de credits bekijken (andere scene) of de app afsluiten. Behalve dat de buttons elk een actie uitvoeren, heeft deze scene verder geen verplichtingen.

Credits
Dit is een scherm waarop de credits te zien zijn, ergo wie de app gemaakt heeft en waar de muziek vandaan komt. Deze scene heeft alleen een button om terug te keren naar het startscherm.

Level
Deze scene is veruit de ingewikkeldste van de app. In deze scene hebben wij allerlei prefabs (zie volgende stuk voor welke prefabs wij gebruikt hebben) die het level opmaken geplaatst. Denk bijvoorbeeld aan het karakter, de omgeving, vijanden, muziek, de buttons om het spel te besturen, enzovoort. In deze scene speelt de gebruiker het spel. Er zit ook een back-button, waarmee de gebruiker terug kan keren naar het hoofdscherm. Daarnaast kan het karakter 'doodgaan' in het spel, waarna de death-scene wordt opgeroepen of kan de speler één van de twee deuren in het level bereiken waarna de speler bij het gevecht met de eindbaas komt, wat in een aparte scene zit.

BsaFight
Deze scene is ontwikkeld om te kunnen vechten met de eindbaas. Het is een soort uitbereiding op de scene die het level bevat. Ook deze scene bevat een groot aantal prefabs om de omgeving op te maken. Deze scene is alleen te bereiken door een van de twee deuren in de level-scene door te gaan. Vanuit deze scene kunnen zowel de death-scene als de win-scene bereikt worden, afhankelijk van het succes van de speler.

WinScreen
Deze scene laat de gebruiker zien dat hij of zij het spel heeft gewonnen. Deze scene kan alleen bereikt worden door in het gevecht met de eindbaas de deur door te gaan. Vanuit deze scene kan men terugkeren naar het startscherm.

DeathScreen
Als de speler ergens in BsaFight of Level 'doodgaat', dan krijgt hij deze scene te zien. Deze bevat de opties om weer naar het level te gaan, de credits te bekijken of de app af te sluiten.

Detailontwerp
In deze sectie behandelen we de belangrijkste (groepen) prefabs die wij gebruikt hebben in de app.

Player
De speler is natuurlijk een belangrijk element in de game. Deze bevat de afbeelding van de speler en verschillende eigenschappen van de speler, zoals de mogelijkheid om tegen dingen aan te botsen, zwaartekracht en de mogelijkheid om te springen en lopen. Ook wordt hier de vorm van het karakter vastgelegd, wat later weer wordt gebruikt bij het bepalen of het karakter met iets 'botst'

Background
Deze prefab bepaalt de achtergrond van het level en bsaFight. Dit is voornamelijk de afbeelding en een aantal gegevens zoals positie en de schaal

Omgeving (level design)
Dit is een grote verzameling prefabs die het level opmaken. Men moet hierbij denken aan de grond, de muren, platforms, enzovoorts. Deze prefabs hebben allemaal eigenschappen die ervoor zorgen dat een karakter er op kan staan of er tegen aan kan botsen.

Coin (collectable)
Deze staan in Unity onder level design, maar zijn toch net iets anders dan de andere prefabs in omgeving. Zo bevatten muntjes eigenschappen die ervoor zorgen dat het karakter er juist wel

doorheen kan lopen en dat ze verdwijnen zodra ze het karakter aanraken. Ook verhogen ze de score bovenaan het scherm.

HUDs

De HUDs bestaan bij ons uit twee dingen. Aan de ene kant hebben we alle buttons. Deze roepen scriptjes op om de speler te verplaatsen of een nieuwe scene te openen. Aan de andere kant hebben we HUDs die de gebruiker op de hoogte houden van hoe het er op dit moment voorstaat. Zo hebben we een loading screen, wat getoond wordt wanneer het level wordt geladen. Ook de score van de muntjes die verzameld zijn wordt getoond door middel van een HUD prefab.

Enemies

Enemies zijn de vijanden in het level die je voornamelijk wilt vermijden. De prefabs voor vijanden bevatten voornamelijk informatie over wanneer je karakter in contact is met een enemy en wat er dan moet gebeuren. Zo gebeurt er meestal wat anders wanneer men bovenop een enemy landt dan wanneer men er tegenaan loopt. Dit wordt per enemy prefab vastgelegd door middel van opties wel of niet selecteren en enkele scriptjes.

Audio

Er zitten ook meerdere audio prefabs in onze app, voornamelijk in de level en bsaFight scene. Van deze prefabs is voornamelijk vastgelegd welk geluidsfragment moet worden afgespeeld wanneer ze worden aangeroepen.

Ontwerpverantwoording

Allereerst is het gepast om in onze ontwerpverantwoording uit te leggen waarom wij gekozen hebben voor Unity. De voornaamste reden is dat het een stuk makkelijker is om een game te ontwerpen met een engine. Specifiek Unity omdat een lid van ons groep al ervaring had met deze engine. Verschillende aspecten die voor onze game heel belangrijk zijn, zoals zwaartekracht, het laten botsen van objecten en level ontwerp zijn in een game engine veel makkelijker dan in bijvoorbeeld Android studio. Daarnaast; waarom zou je het wiel opnieuw willen uitvinden? Unity is gemaakt voor het maken van dit soort spellen en heeft een groot deel van wat wij willen al in libraries die wij kunnen gebruiken. Als we dit zelf hadden moeten maken, was dit niet te doen geweest in de beschikbare tijd en was het nooit van deze kwaliteit geweest.

Dan volgen nu nog twee keuzes met toelichting die meer te maken hebben met het ontwerp van de app specifiek.

Een ontwerpkeuze waar we nog lang over gediscussieerd hebben, is de plaatsing van de knoppen waarmee je de speler bestuurt. Eerst wilden we voor een ontwerp gaan waarbij de knop om naar links te lopen aan de linkerkant van het scherm zat en de knop om naar rechts te lopen aan de rechterkant, aangezien dit als een intuïtieve indeling voelt. De jump-button zou dan in het midden zitten. Wij hebben echter gedacht dat dit onhandig zou zijn, aangezien je dan telkens moest wisselen met welke hand je springt: links als je naar rechts loopt en rechts als je naar links loopt. Daarom zijn wij uiteindelijk gegaan voor een besturing waarbij je met links de speler laat lopen en met rechts springt. Dit leek ons makkelijker voor de gebruiker. Bovendien komt dit overeen met de meeste game consoles die op dit moment gebruikt worden. Daardoor zou de gebruiker makkelijker onze app moeten kunnen gebruiken.

Een andere ontwerpkeuze die wij gemaakt hebben was de double-jump. Dit houdt in dat een karakter twee keer kan springen om zo hoger in de lucht te komen. Een alternatief was geweest om

de double jump weg te halen en de platforms dichter bij elkaar te zetten of de single jump hoger te laten komen. Wij hebben echter voor de double-jump gekozen omdat wij dachten dat dit een extra uitdaging aan het spel zou toevoegen. De double-jump goed beheersen kost wat extra moeite voor de speler en wij waren van mening dat dit het spel uitdagender en daardoor hopelijk ook leuker zouden maken. Wij hebben uiteindelijk besloten ook daadwerkelijk voor de double-jump te gaan, nadat wij een aantal studiegenoten hadden gevraagd wat ze leuker leek. Uit dit mini-onderzoekje bleek dat er meer voorkeur was voor de double-jump dan voor een single-jump.

Reflectie

We zijn erg tevreden over het resultaat van onze applicatie. Onze insteek was het een zo vrolijk, gezellig mogelijk spelletje te maken, wat vooral leuk zou zijn voor studenten. Om dit voor elkaar te krijgen was het noodzakelijk om gebruik te maken van de game engine Unity. Doordat we in Unity werkten hoefden we dingen zoals collisions en zwaartekracht niet zelf te maken.

Maar werken met Unity had ook haar keerzijdes. Sean was de enige van ons groepje die ervaring had met Unity. Voor de rest was het vaak lastig uit vinden hoe bepaalde dingen werkten. We hebben veel uren verspild aan het googlen van dingen, die, als we Unity gekend hadden, niet veel tijd gekost zouden hebben.

Ook het programmeren in C# was een uitdaging. De meesten van ons hadden dit nog nooit gebruikt. Hoewel de taal veel op Java lijkt, was het toch altijd even uitzoeken hoe je code precies opschreef. Je weet niet precies wat je in welke klasse moet zetten, en dat was even uitvinden.

Een andere uitdaging was dat we niet heel veel tijd hadden. Vijf weken is niet veel tijd om een app in elkaar te zetten en dat hebben wij ook gemerkt. We hadden graag meerdere levels geïmplementeerd of meer aan de usability van onze app gewerkt, maar dat is door tijdsgebrek niet mogelijk gebleken.

De samenwerking is altijd positief geweest. We hebben geen problemen met elkaar gehad, en ook kunnen genieten van het ontwerpen van deze app. In het begin ging de taakverdeling wat stroef doordat Sean en Taras vooral met Unity bezig waren, terwijl Nienke en Astrid vooral over de designkant nadachten.

Halverwege hadden we een feedbackmoment met onze begeleider, die ons er terecht op wees dat de verdeling niet helemaal goed was. Vanaf dat moment hebben we meer samengewerkt en ook onderdelen aangepakt die ons misschien in eerste instantie wat minder lagen. Daardoor hebben we niet alleen een product neer kunnen zetten waarvan we kunnen zeggen dat we er allemaal aan gewerkt hebben, maar hebben we ook veel van elkaar kunnen leren.

Het is fijn dat het gelukt is om een leuk spel neer te zetten, waaraan we echt alles zelf hebben gedaan. Alle graphics die te zien zijn hebben we zelf ontworpen. Het level hebben we zelf in elkaar gezet. Bolke kan double jumpen, de vijanden zijn geslaagd, en we hebben er leuke geluidseffecten

bij. Voor een volgende keer zullen we vanaf het begin beter op de taakverdeling letten en zorgen dat we de programmeeromgeving goed snappen. Maar voor nu zijn we trots op wat we bereikt hebben.