Project AACG : Visitor

Groep 4 :
Robin Marx
Nick Michiels
Jimmy Cleuren
Wouter Nivelle

Academiejaar 2008–2009

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Concept

Visitor is een 2.5D sidescrolling platform-game dat zich afspeelt in een middeleeuwse wereld die onder de voet gelopen wordt door hongerige zombies. De speler is een cyberninja uit de toekomst die toevallig op deze wereld terechtkomt en de zombies moet verslaan.
De gameplay wordt naast de traditionele spring, val aan en loop-acties vooral gekenmerkt door ver geïntegreerde physics en een interessant nieuw concept dat mousegestures gebruikt voor het uitvoeren van bepaalde acties.
Er is ook een online versus-modus waarin 2 spelers zich om ter snelst naar de andere kant van een level moeten begeven. Ze beginnen echter elk aan een andere kant en kunnen bepaalde stukken van het level opzettelijk aanpassen om het hun tegenstander moeilijker te maken als die daar later voorbij komt.
Het spel is verregaand aanpasbaar door het gebruik van editors en scripting, zodat nieuwe objecten en acties snel en gebruiksvriendelijk kunnen worden toegevoegd voor eindeloos spelplezier!
Dit concept is gebaseerd op een spel dat in de zomer van 2009 uitkomt, namelijk Trine (http://www.trine-thegame.com) voor de physics gameplay en het spel Black & White (http://www.lionhead.com/bw/) voor het gebruiken van gestures in games.

1.2 Aanpak

De projectgroep bestaat uit 4 leden, namelijk: Robin Marx, Jimmy Cleuren, Nick Michiels en Wouter Nivelle. Zij zijn alle 4 studenten in de 3e bachelor informatica-ict aan de Universiteit Hasselt.
Bij de initiële taakverdeling kreeg iedereen een aantal verantwoordelijkheden toegekend:

- Robin : Graphics, scripting, gestures
- Jimmy: Physics en allrounder
- Nick: Editors en allrounder
- Wouter: Modellen/animaties en netwerk

Figuur 1.1: Screenshot van het spel

In het laatste hoofdstuk zullen we zien dat deze verantwoordelijkheden niet altijd even strikt gebleven zijn.
Het concept dat we gekozen hebben had enkele voor- en nadelen in verband met de implementeerbare features. Zo konden we veel optimaliseren qua graphics en netwerk omdat het spel zich niet in een volledige 3D wereld afspeelt (alles gebeurt binnen een beperkte z-range). Ook konden we de AI relatief simpel houden en was er geen nood om hiervoor externe libraries te gebruiken. Moeilijkere zaken waren dan weer de implementatie van de physics en de grote verscheidenheid aan objecten, wat via scripting werd gedaan. Volgens de originele opgave waren dit de extra's. In onze voorbije jaren hebben wij echter al veel spelletjes gemaakt en daarbij ook steeds veel aandacht besteed aan de geziene netwerk en AI algoritmen. We vonden het dan ook interessanter om ons nu bezig te kunnen houden met totaal nieuwe dingen zoals physics en scripting en de basisvereisten niet verder uit te breiden dan echt nodig. Zo bekomen we een eindproduct dat voldoet aan deze basisvereisten maar toch veel interessante uitbreidingen bevat.
Er werd reeds in het begin een wiki opgezet om onze doelstellingen en vorderingen bij te houden en waarop we interessante informatie met elkaar konden delen. De meeste schema's in dit verslag komen rechtstreeks van deze wiki en werden tijdens de ontwikkeling gebruikt om de structuur duidelijk te maken voor de groepsleden.
We maakten gebruik van een door de universiteit voorziene SVN server voor versiebeheer. Op het moment van schrijven zaten we al over de 250 revisies.

De uiteindelijke bedoeling van de opgave was om zoveel mogelijk met externe libraries te werken om al deze zaken te verwezenlijken en uiteindelijk een soort van algemene game-engine te bekomen. Welke libraries dit geworden zijn en wat we er precies mee hebben gedaan leest u in het volgende hoofdstuk.

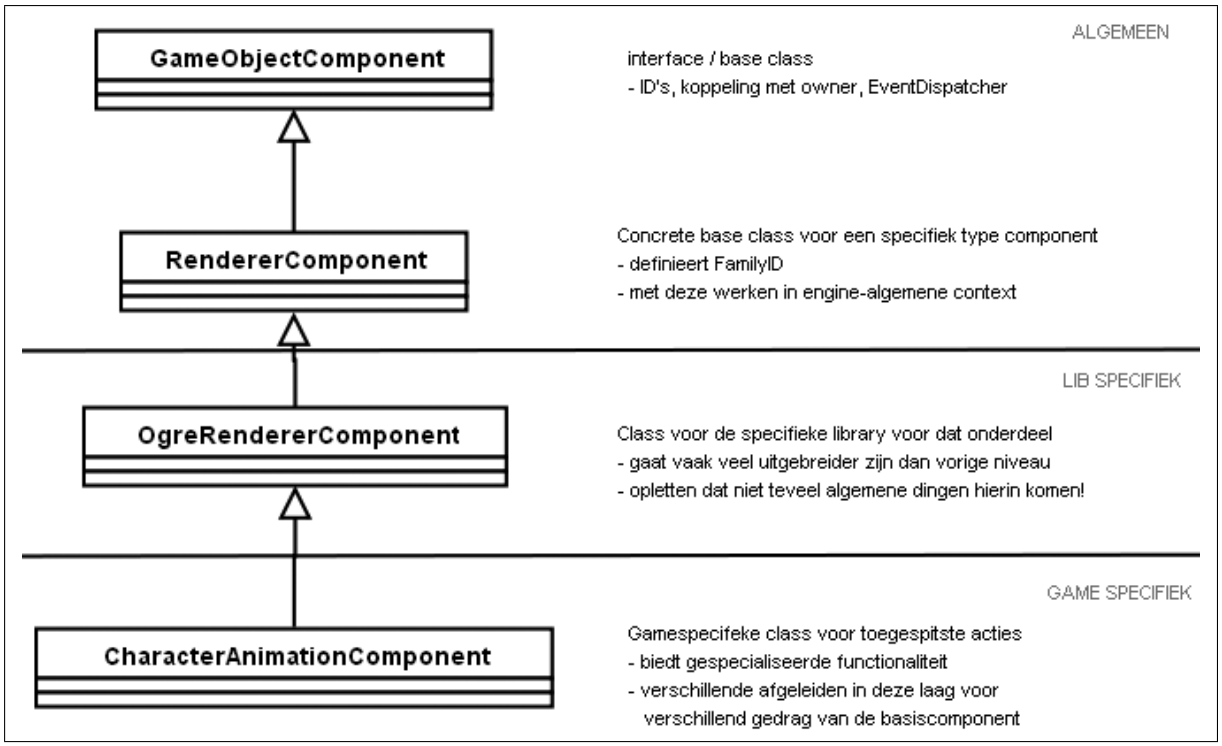
Hoofdstuk 2

Gebruikte technologieën

2.1 Basisstructuur

GameObject
GameObjectComponent
GameObjectFactory

In aangereikt lesmateriaal hadden we het concept van component-gebaseerd programmeren ontdekt. Waarbij men in traditioneel object-geörienteerd programmeren classes van elkaar gaat afleiden om nieuwe en uitgebreidere objecten mogelijk te maken, gaat men hier classes uitbreiden door ze bepaalde membervariabelen toe te kennen, de components. Deze membervariabelen bevatten dan de functionaliteit.
Deze structuur maakt het heel makkelijk om dynamisch extra functionaliteiten toe te kennen aan verschillende objecten zonder dat hiervoor nieuwe classes moeten worden geschreven. Vooral in game-development is dit zeer belangrijk omdat 1 object vaak zeer veel functionaliteit moet aankunnen en als we met afleiden zouden werken zou dit snel onoverzichtelijk worden.
De component-structuur geeft ook de mogelijkheid om de kern van de functionaliteit aan te pakken per component. Elke component heeft een duidelijke omschrijving en doel en hoeft maar weinig af te weten van andere aanwezige componenten om goed te kunnen werken.
Onze implementatie van deze structuur zit in de classes **GameObject** en **GameObject-Component** (GOC). GameObject is een class die eender welk object in het spel voorstelt. Wat er zoal voor functionaliteit op dat object gedefinieerd is, hangt af van de GameObject-Components die eraan hangen. In de verschillende GOC's zit er wel een hele structuur van afleidingen. Zo is er bijvoorbeeld 1 basiscomponent voor physics waarvan er dan 4 andere zijn afgeleid voor meer specifieke functionaliteit. Door deze combinatie van component-gebaseerd en object-georiënteerd werken is de structuur zeer dynamisch en uitbreidbaar.
We kunnen natuurlijk niet alle functionaliteit in de components totaal onafhankelijk maken van andere functionaliteit. We willen bijvoorbeeld dat er een animatie wordt afgespeeld wanneer het object door de spelwereld beweegt, dat het object reageert op andere dmv physics, dat er eventueel geluid wordt afgespeeld, etc. Het spreekt voor zich dat deze functionaliteit in aparte componenten zit. Het zou dan ook slecht zijn mocht 1 component functies op andere componenten aanroepen wanneer er iets gebeurt.
In plaats van functies op te roepen op de andere componenten gaat 1 component aan zijn parent GameObject laten weten dat er iets is gebeurd. Dit doet hij door een klein event-object



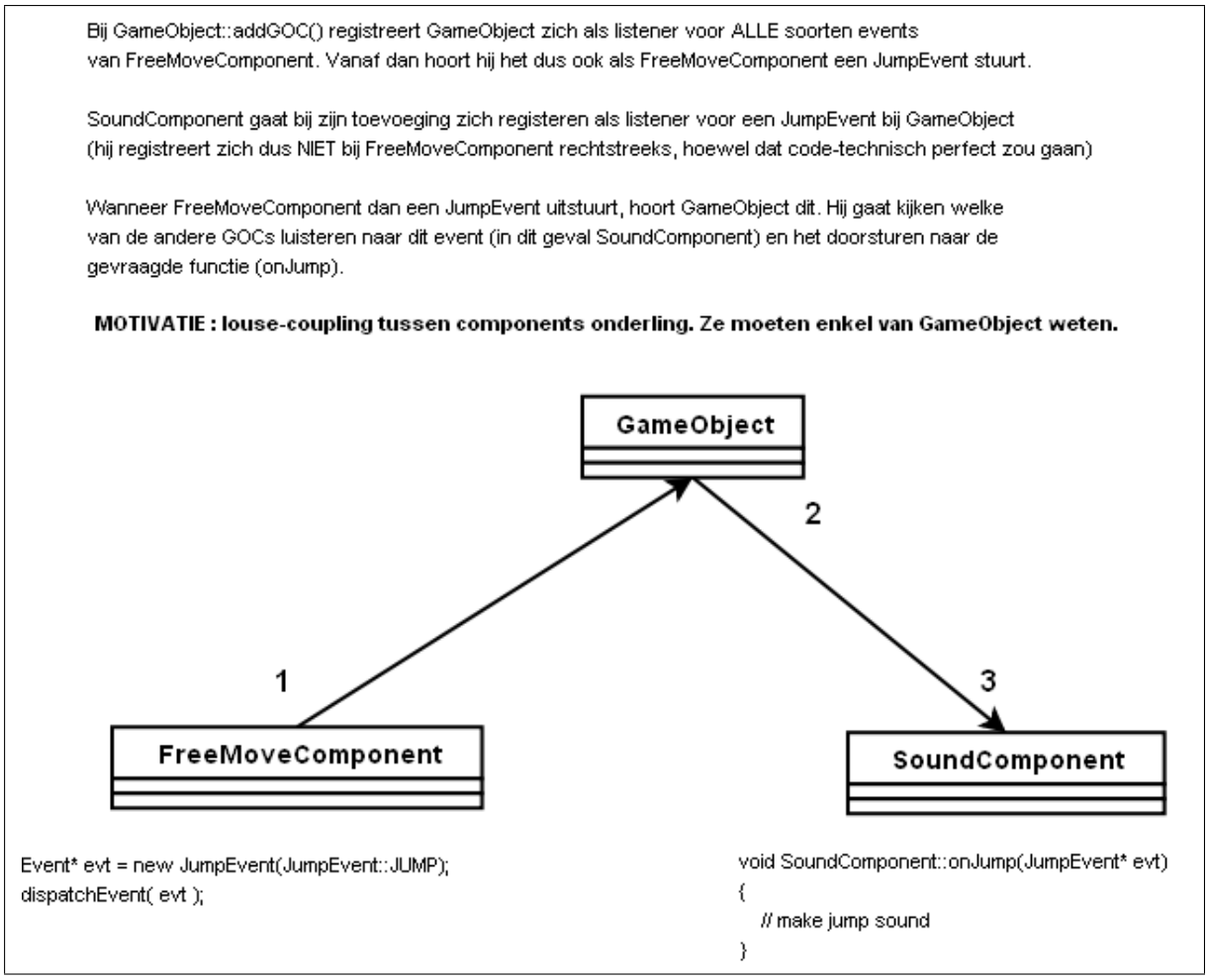
Figuur 2.1: Afleiden van GameObjectComponent

naar de parent te sturen. Deze parent stuurt dit event dan door naar alle andere GOC's die erin geïnteresseerd zijn. GOC's vertellen zelf aan de parent in welke events ze geïnteresseerd zijn.
Deze structuur zorgt ervoor dat het niet uitmaakt welk soort component bepaalde events uitstuurt. Zo kan een animatieComponent samenwerken met eender welke component die bijv. Walk-events uitstuurt, of dat nu een AIComponent, FreeMoveComponent of Network-Component is.

2.2 Levelstructuur

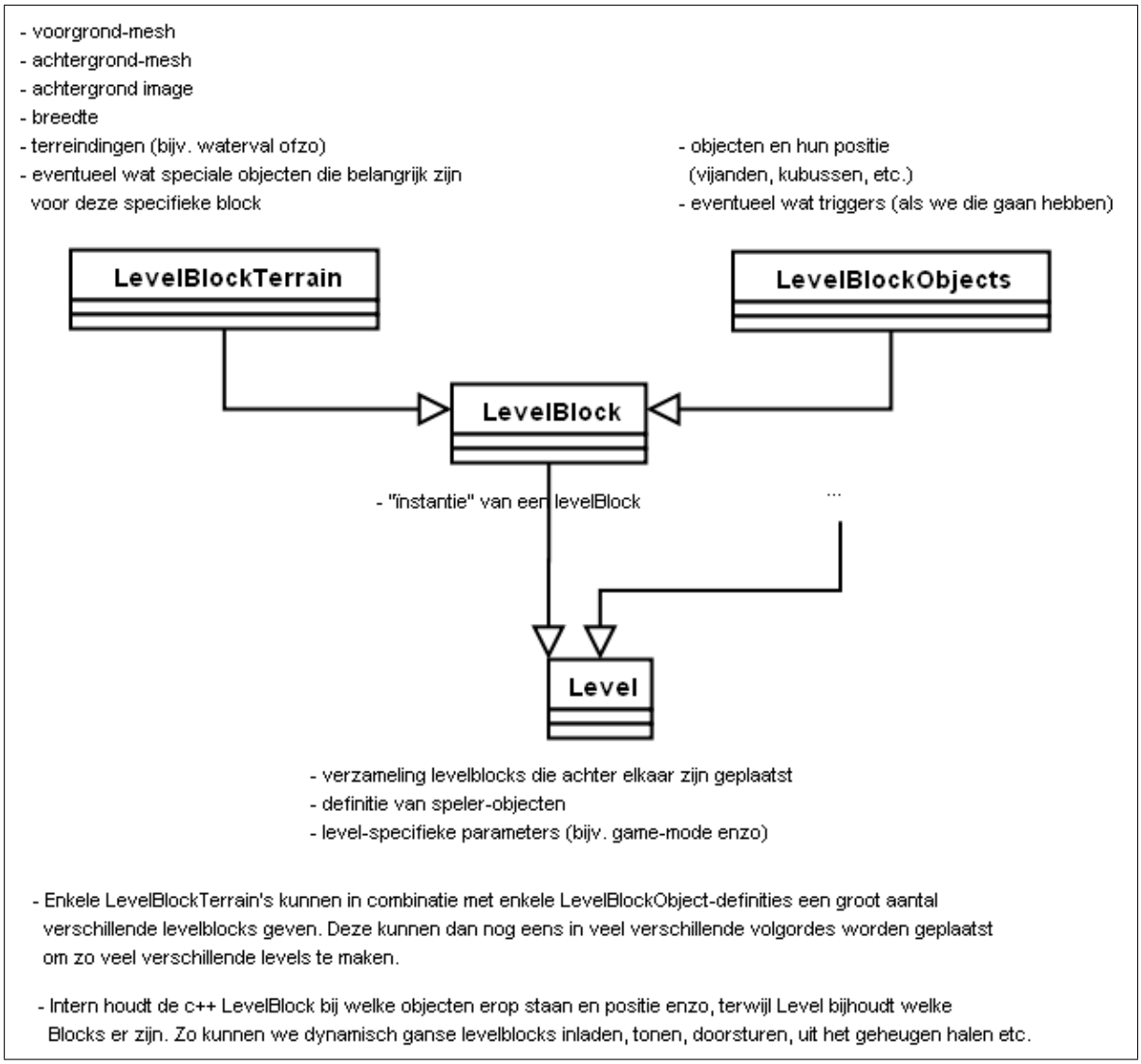
Level
LevelBlock
LevelBlockTerrain
LevelPagingManager
IDUtility

Het spel speelt zich af in een beperkte z-range en constant van links naar rechts (of andersom). We kozen er dan ook voor om de camera niet door de gebruiker vrij te laten besturen maar de speler te laten volgen vanop een afstand. Hierdoor weten we zeker dat de gebruiker niet zelf kan gaan rondkijken in een level en dit geeft goede mogelijkheden voor optimalisaties.
We werken met het principe van levelblokken. Een level is zo opgebouwd uit verschillende stukken met elk een eigen ondergrond en objecten. Omdat de camera niet door de gebruiker rechtstreeks wordt bestuurd kunnen we zorgen dat enkel de levelblokken die op dat moment



Figuur 2.2: De eventstructuur

nodig zijn in het geheugen moeten zijn. Standaard staat dit nu ingesteld op slechts 3 blokken tegelijk in het geheugen. Wanneer de speler naar een volgende blok overgaat worden de vorige voor hem afgesloten zodat hij niet terug kan en deze afgesloten blokken kunnen dan in de zeer nabije toekomst worden verwijderd uit het geheugen omdat ze niet meer nodig zijn. Het aantal blokken dat je kan teruggaan kan ook ingesteld worden.
De levelblokken zijn intern ook modulair opgebouwd zodat men bijvoorbeeld dezelfde achtergrondmesh kan gebruiken in meerdere blokken tegelijk en objecten makkelijk kunnen worden toegevoegd aan terreindefinities. Zo kunnen we met 1 terreindefinitie een hele reeks verschillende blokken maken door er andere objecten op te plaatsen. In het level zelf kunnen levelblokken worden herhaald, bijvoorbeeld voor een symmetrisch multiplayer level (zie ook later).
Deze opbouw had nood aan een duidelijke manier om alle blokken en objecten erop van elkaar te kunnen onderscheiden. Deze onderscheiding hebben wij gemaakt in de vorm van hiërarchisch opgebouwde ID-strings. Op deze manier konden de objecten via hun ID van elkaar worden gescheiden en ook voor het netwerk was dit zeer belangrijk (zie later).

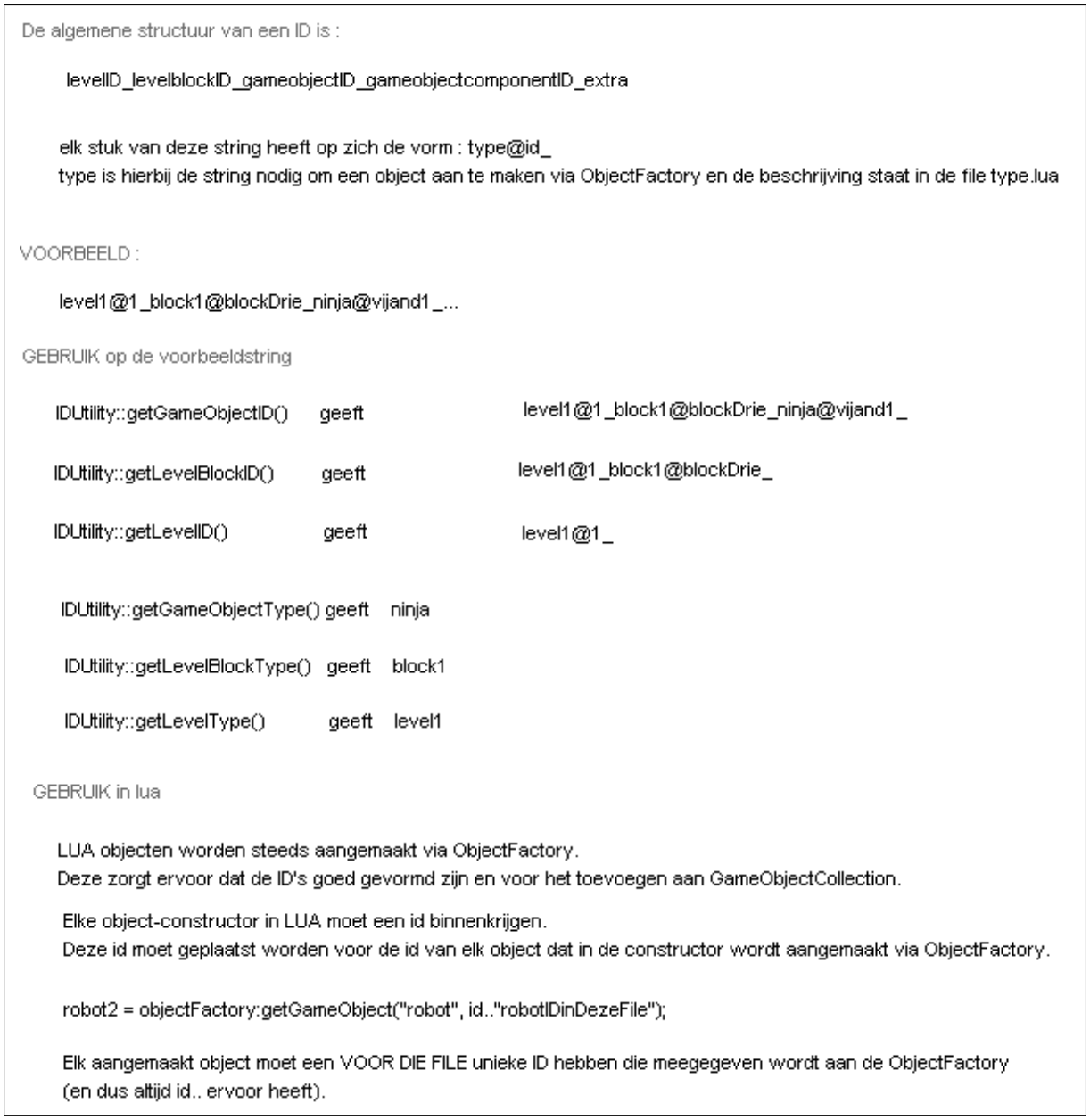


Figuur 2.3: Modulaire opbouw van de levels

2.3 Graphics

OgreRendererComponent
CharacterAnimationComponent
OgreLevel
OgreLevelBlock

Op aanraden van het onderwijsteam gebruiken we de rendering engine OGRE (http://www.ogre3d.org). OGRE is een object-georienteerd framework dat als een wrapper rond openGL of directX ligt en allerhande handige functionaliteit voorziet. Hieronder verstaan we onder andere resource management, makkelijk omgaan met meshes en animaties, makkelijke applicatie ontwikkeling, koppeling met allerhande uitbreidingen, etc.



Figuur 2.4: Gelaagde opbouw van de IDs

Ogre staat bij ons in voor het renderen van GameObjects en de levels. Ook gebruiken we de ingebouwde functionaliteit voor skeletal animation voor onze spelerobjecten en de vijanden. Ogre bestuurt ook de main game loop en stuurt frameEvents uit waarnaar op verschillende plaatsen wordt geluisterd. Ogre biedt ons ook de mogelijkheden van het Open Input System (OIS) voor keyboard- en mouseinput.
OGRE was zeer makkelijk in het gebruik. De meeste dingen wezen zichzelf uit en waren vaak zeer snel werkende. Het enige probleem was dat het middelpunt van de SceneNodes die OGRE gebruikt om objecten te positioneren in de wereld niet in de linkerbovenhoek van het object lagen, maar wel in het middelpunt van dat object. Hierdoor waren wat speciale

berekeningen nodig in bijvoorbeeld de levelopbouw.
We hebben ook heel wat objecten zelf gemaakt in Blender (www.blender.org). Deze meshes konden relatief simpel worden geëxporteerd naar een OGRE-compatibel formaat. De cyborg-mesh waarmee gespeeld wordt is zelf aangepast en geanimeerd en de levels zijn volledig zelf gemodelleerd en geanimeerd.
We hebben ook de addon Caelum gebruikt. Dit is een addon die een skybox simuleert met zonnebeweging, sterren, wolken, sfeerlicht, etc. Dit hielp ons om snel wat meer sfeer aan het spel te geven.

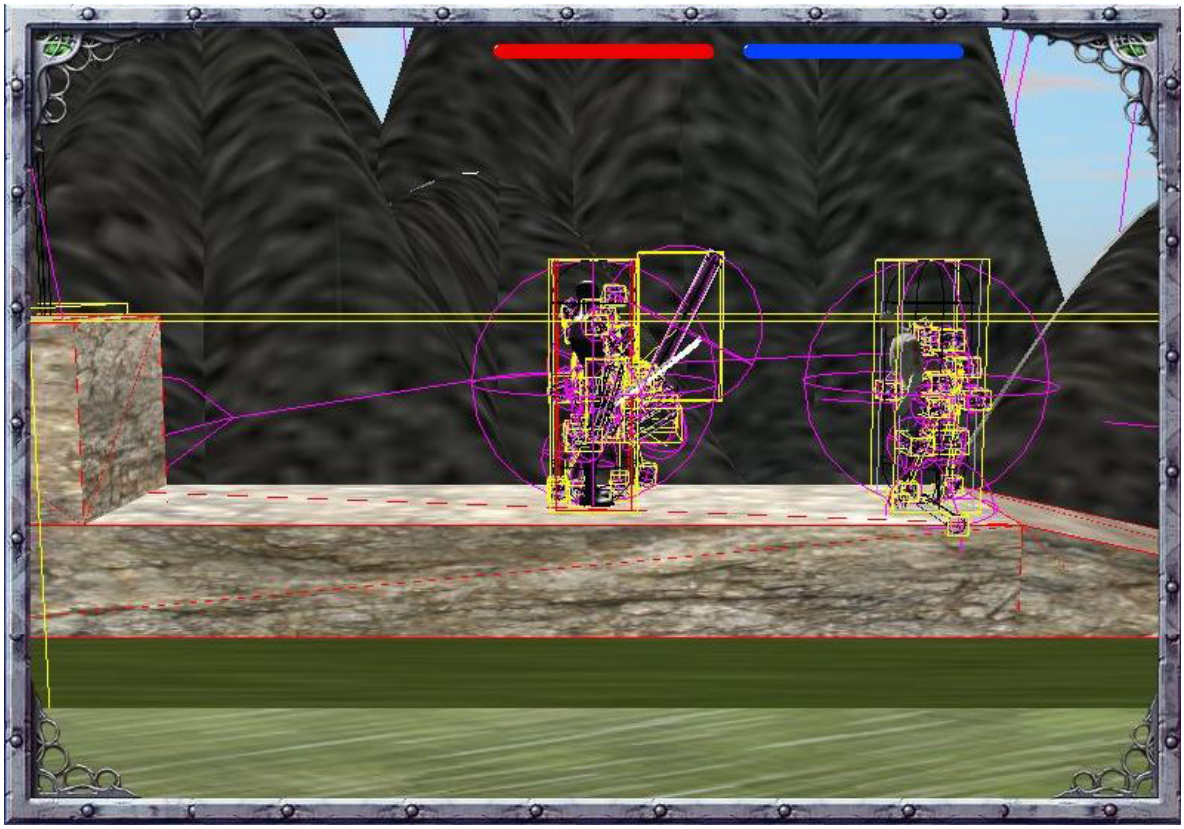
Figuur 2.5: Caelum zorgt voor belichting en wolken

2.4 Physics

PhysxPhysicsComponent
RagdollComponent
StaticPhysXPhysicsComponent
TriggerPhysXPhysicsComponent
Construction
Cube
Trigger

Voor de physics hebben we NXOGRE gebruikt (http://www.nxogre.org). Dit is een wrapper

voor de NVidia PhysX-library, dewelke sinds kort vrij beschikbaar is voor niet-commerciële doeleinden. We hadden verschillende mogelijkheden maar de PhysX-library leek ons de meest volwassen en meest uitgebreide van ze allemaal. Het feit dat er een binding met OGRE was via NXOGRE was ook een groot pluspunt.
Omdat we de physics veel wilden gebruiken is er veel tijd gekropen in het werken met NXOGRE. Zo zijn er automatisch gegenereerde convex-hulls voor alle meshes en voor Characters zoals de zombies en de cyborg zijn er automatisch gegenereerde ragdolls. Deze zorgen voornamelijk voor collision detection, wat op zijn beurt dan weer wordt gebruik om te zorgen dat de objecten goed op elkaar reageren, je kan vechten met vijanden en dat de objecten automatisch het terrein waar ze op staan volgen.
Naast deze toepassingen zijn er ook statische physics. Die zorgen ervoor dat de z-range wordt bewaard en men niet terugkan naar afgesloten levelblokken. Ook wordt hiermee automatisch het gedrag van de terreinen geregeld, zodat de objecten hiermee botsen. Een andere toepassing zijn zogenaamde triggers. Dit zijn onzichtbare objecten die collisions met objecten kunnen opvangen en doorsturen. Dit is bijvoorbeeld handig voor de AI om acties te definiëren wanneer een vijand in een bepaald gebied komt.



Figuur 2.6: Debug outlines van de ragdolls en physics

NXOGRE was heel wat moeilijker om mee te werken dan OGRE zelf. De library is nog niet zo volwassen en is niet altijd even goed gedocumenteerd. Zeker als we details nodig hadden was het soms moeilijk om te weten hoe alles precies kon gedaan worden. Uiteindelijk zijn de meeste dingen wel goed gelukt. Het enige probleem dat er nog was is dat de zombie-mesh

die we gebruiken niet zelf gemaakt was. Hierdoor zitten de bones fout en als we een gewone ragdoll wilden opzetten (bijvoorbeeld als de zombie dood zou gaan) gaf dit zeer bedenkelijke resultaten. De ragdoll wordt echter wel goed gebruikt voor het detecteren van collisions tussen bepaalde onderdelen van de Characters, bijvoorbeeld tussen het zwaard van de cyborg en de vijanden. Hierbij wordt de ragdoll bestuurd door de animaties van het object in plaats van door de physics, waardoor ze niet in elkaar valt maar toch gedetailleerde collision informatie kan sturen.

2.5 Scripting

LuaScriptComponent
LuaManager
LuaObjectFactoryImp

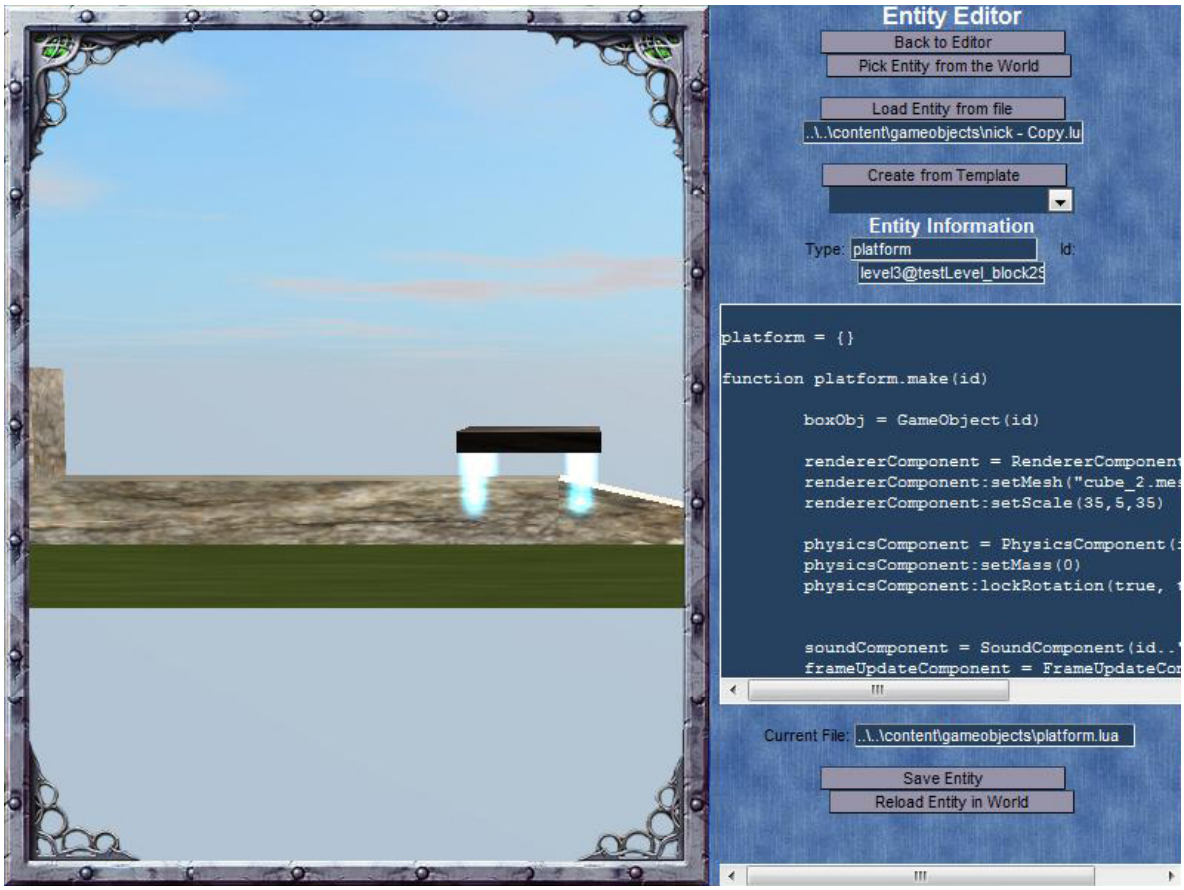
In eerdere projecten hadden we al goed de voordelen van data-driven design leren kennen. Nu kregen we echter de mogelijkheid om veel verder te gaan dan gewoon xml-files met configuratie informatie. De scriptingtaal LUA (http://www.lua.org/) is één van de meestgebruikte in allerhande grote games. LUA op zich is een taal met een zeer duidelijke en simpele syntax, maar de connectie met C++ is zeer low-level. Zo moet men bijvoorbeeld zelf variabelen op de stack zetten etc. Gelukkig bestaat er ook LUABIND (http://www.rasterbar.com/products/luabind/docs.html). Dit is een wrapper rond LUA die het heel wat makkelijker maakt om functies uit C++ beschikbaar te maken voor gebruik in de script-files. We gebruikten dus LUABIND voor de scripting van ons project.
Het idee is dat de constructors van de verschillende objecten elk in een LUA file zitten. Dit is handig omdat de objecten vaak opgebouwd zijn uit een specifiek soort GameObject, tesamen met bepaalde GameObjectComponents voor hun functionaliteit. Door deze verschillende GOC's beschikbaar te stellen aan LUA kunnen we alle high-level samenstellingen via scripting doen. Dit is echter niet alleen zo voor de GameObjects; ook de levelblokken zitten in de LUA-files, waardoor nieuwe levels snel kunnen gemaakt worden zonder iets te moeten compileren of aanpassen aan de broncode. Naast de constructors kunnen er ook eventlisteners worden toegevoegd via LUA. Via de LuaScriptComponent kan je in de LUA-files events opvangen en erop reageren in aparte functies.
De grote kracht achter deze aanpak is dat er heel wat minder moet gebeuren in C++ en dat je leert veel modulairder te werken. Je probeert de verschillende components zo onafhankelijk mogelijk van elkaar op te bouwen zodat je ze zo flexibel mogelijk kan combineren. Dit combineren gebeurt uiteindelijk in de LUA files. Dit zorgt ervoor dat er heel snel getest en uitgebreid kan worden. Een voorbeeld zijn de platformen. Deze waren niet voorbereid in de code en er waren geen specifieke components voor. Toen we dit idee krijgen is dit op niet meer dan 15 minuten kunnen toegevoegd worden zonder ook maar 1 regel te moeten veranderen in de C++ code. Toen we later ook bewegende platformen wilden was slechts een minieme aanpassing aan AreaAIComponent nodig om ook dit mogelijk te maken. Dit zijn twee relatief ingrijpende veranderingen gameplay gewijs die toch op minder dan een uur konden toegevoegd worden.
LUA is echter niet altijd zonnegeur en maneschijn. De LUABIND-library is spaarzaam gedocumenteerd en het is nogal een gedoe om ze gecompileerd te krijgen, mede omdat ze de boost-library gebruikt. Eens dit euvel was opgelost bleef het feit dat er goed moet opgelet

worden met hoe je functies aan LUA beschikbaar maakt. Dit is niet moeilijk via LUABIND maar de syntax in de LUA files moet goed in het oog gehouden worden.
Alle LUA functies worden ingeladen naar één zogenaamde LUA state. Dit heeft tot gevolg dat functies die hetzelfde noemen elkaar overschrijven; dit gebeurt ook met globale variabelen. Een functie make() in elke file was dus niet mogelijk. Dit is slechts 1 van de vele domme probleempjes die we ondervonden met LUA. Ook het gebrek aan een goede debugger was soms een reden tot frustratie. 1 enkele ; of vergeten) zorgde voor een crash zonder enige aanduiding waar de fout zat. Trial and error - methodes gecombineerd met een print() functie waren hard nodig. Door dit soort moeilijkheden hebben we ervoor gekozen de LUA niet super uit te breiden. Zo kan je bijvoorbeeld geen eigen GOC's definiëren in LUA en ze daarna gebruiken in andere LUA-files.

2.6 GUI en editors

AwesomiumUI
AwesomiumApplicationHandler

Voor de GUI besloten we al snel om te gaan voor een browser-gebaseerde library. De vaak gebruikte CeGUI, die goed met OGRE samenwerkt, is helemaal C++ gebaseerd. We besloten dan ook om Awesomium (http://princeofcode.com/awesomium.php) te gebruiken. Deze library gebruikt de rendering engine van Google Chromium en laat ons toe de GUI volledig in HTML, javascript en CSS op te bouwen. Dit heeft niet enkel het voordeel dat dit heel snel kan gebeuren (iedereen van ons kent immers zeer goed deze webtechnologieën) maar het is tegelijk het GUI equivalent van scripting. Er hoeft niks gecompiled te worden, wat aanpassingen aan de html files zijn genoeg. Awesomium werd gekozen in plaats van Navi omdat Awesomium op dat moment de beste was qua performance.
Naast de GUI besloten we ook om een editor te maken. In combinatie met onze scripting betekent dit in de praktijk dat we een in-game teksteditor voorzien om de LUA-files aan te passen en at runtime terug in te lezen. Er is ook een picking-mogelijkheid waarbij je kan klikken op een object in de spelwereld waarna de LUA-file voor dat object wordt geopend. In de editor kan je ook levels en levelblokken inladen en aanpassen.
Awesomium was eventjes moeilijk om aan het werken te krijgen, ook weer door gebrekkige en onbestaande documentatie, maar toen het eenmaal werkte ging het zeer vlot om de GUI en de editors op te bouwen. Het enige probleem dat we ondervonden lag niet aan Awesomium maar eigenlijk aan NXOGRE. Het idee was om ganse levels opnieuw inlaadbaar te maken via de editor, zodat je je aanpassingen aan levels kon bekijken zonder opnieuw te moeten opstarten. Omdat we bepaalde objecten van NXOGRE niet goed gedelete kregen en niemand ons hier goed mee kon helpen, is het dus niet mogelijk het level opnieuw in te laden zonder opnieuw op te starten. Voor de entity-editor en levelblok-editor lukt dit overigens wel. Je kan de lua files aanpassen en laten her-inladen. Als de volgende blok die ingeladen wordt (zie ook sectie 2.2) de aangepaste blok is of een aangepast object bevat, zal dit goed worden weergegeven zonder dat men moet herstarten.



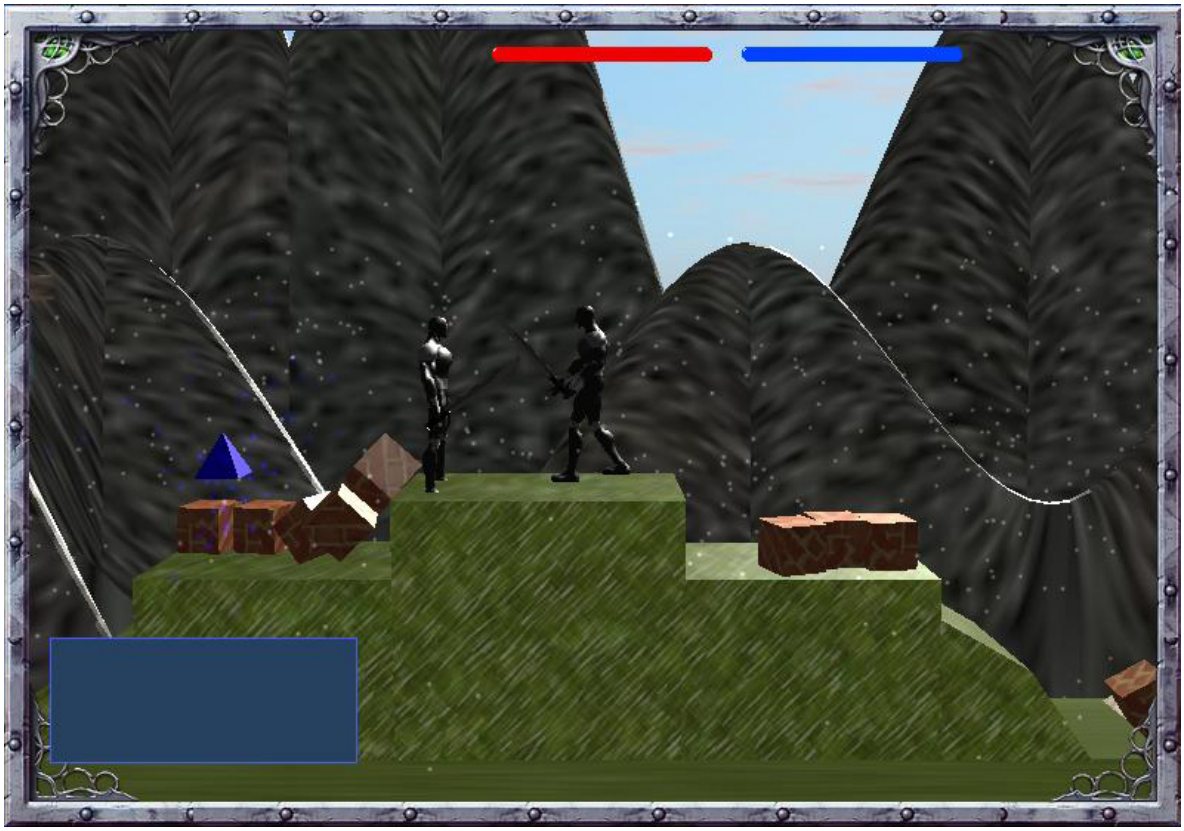
Figuur 2.7: De entity editor voor een platform

2.7 Netwerk

RakNetNetworkManagerImp
RakNetNetworkComponent
RakNetLevelBlockSerializer

Voor het netwerk bleek er maar 1 juiste keuze te zijn, en dat is RakNet (http://www.jenkinssoftware.com/), een library specifiek voor netwerkcommunicatie in games. Ze voorziet handige dingen zoals lobbies, makkelijk serializen van objecten, veilige transmissie, etc. RakNet is een zeer goed gedocumenteerde en vaak gebruikte library dus alles werkende krijgen ging vlot.
Voor het netwerk hadden we wat problemen met de keuze van concept en het gebruik van physics. Het origineel idee was om een soort van co-op speltype te doen over het netwerk waarbij 2 of meer spelers tegelijk doorheen het level gaan. Het probleem hierbij is dat het door ons ver doorgedreven gebruik van physics het noodzakelijk zou maken dat er zeer veel objecten perfect werden gesynchronizeerd (want een kleine afwijking kan soms grote gevolgen hebben voor reacties van andere objecten). Dit zou betekenen dat we een zeer ingewikkelde netwerkstructuur moesten opbouwen met wortels tot diep in de engine. Dingen als dead-reckoning moesten dan immers overal aanwezig zijn om delays etc. op te lossen. We hebben

hierover gepraat met een begeleider en er waren nog andere opties, zoals bijvoorbeeld 1 server de physics-simulaties te laten draaien en dan gewoon positie-updates doorsturen naar de clients. Dit voerde ons al ver voorbij de opzet van het project. Een interessante anekdote is dat het spel waarop we ons baseren, Trine (http://trine-thegame.com/), ook geen online co-op aanbiedt, en dit om precies dezelfde redenen. Er bestaat nog geen gekende standaardmanier om deze problemen aan te pakken, zo wist ook de begeleider ons te vertellen.
We hadden echter wel een ander goed idee voor multiplayer, namelijk een soort *versus* modus. Het idee is dat 2 spelers elk aan een andere kant van een level beginnen. Hun doel is zo snel mogelijk naar de andere kant van het level te raken. Het interessante aan deze modus is dat je keuzes in het eerste deel (voor je elkaar tegenkomt) een invloed kunnen hebben op de snelheid van je tegenstander als deze daar voorbij moet. Als je bijvoorbeeld zelf veel vijanden kan laten staan is het moeilijker voor je tegenstander en als je alle power-ups zelf weet te gebruiken kan je tegenspeler er al niet meer van profiteren.



Figuur 2.8: De 2 spelers komen elkaar tegen in de wereld

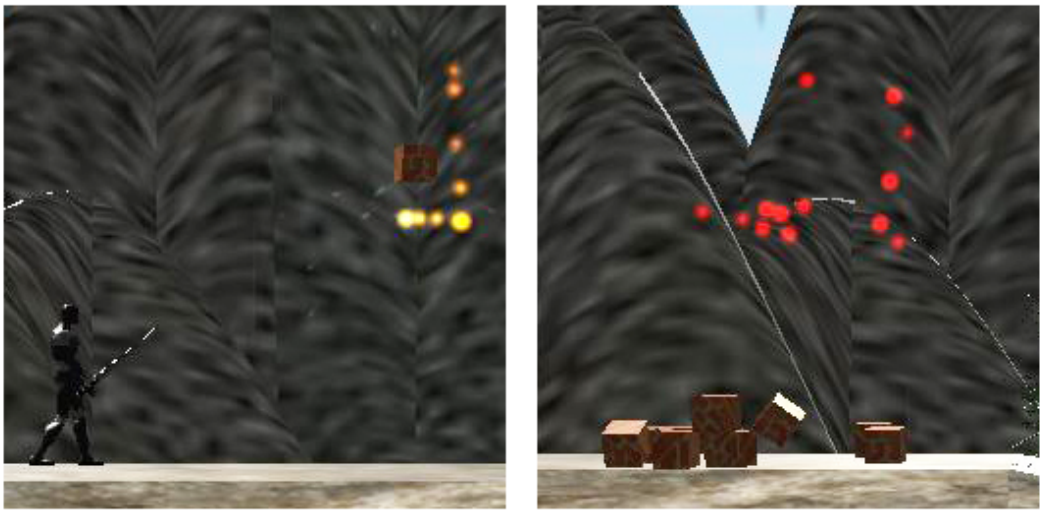
Deze modus paste eigenlijk perfect in onze levelstructuur. De spelers beginnen elk aan een andere kant van het level, dus ze hebben dan ook totaal andere levelblokken in het geheugen. Wanneer een speler voortgang maakt en er levelblokken worden afgesloten, kunnen deze rustig worden geserializeerd en doorgestuurd naar de tegenspeler. Als deze dan later aan die blok aanbelandt kan hij gewoon de eerder binnengekomen data terug deserializeren en de levelblok zal eruit zien zoals de tegenspeler hem had achtergelaten. Het enige probleem is dan: wat gebeurt er wanneer ze elkaar tegenkomen? Dit hebben we opgelost door, wanneer

de twee spelers op dezelfde levelblok zitten, hun posities wel regelmatig door te sturen. De physics staan echter even uit zodat de tegenspeler geen invloed heeft op de huidige levelblok, maar zo zien de tegenspelers elkaar wel voorbijkomen en weten ze van elkaar hoe ver de andere reeds zit in het level. Het positieve aan deze aanpak is dat we slechts 1x de positie en statusinformatie moeten versturen per object in de levelblokken, in tegenstelling tot meerdere keren per seconde. Dit maakt het netwerk zeer geoptimaliseerd en zorgt voor erg weinig verkeer, terwijl het toch een leuke multiplayer-ervaring kan opleveren.
Door tijdsgebrek is deze modus qua gameplay niet goed afgewerkt. Zo is er geen speciaal level dat echt inspeelt op het ik-ga-het-mijn-tegenstander-zo-moeilijk-mogelijk-maken-concept. Maar aangezien de opdracht daar ook niet echt rond draaide zijn we toch zeer tevreden met deze versus-modus.

2.8 Gestures

GestureCapture

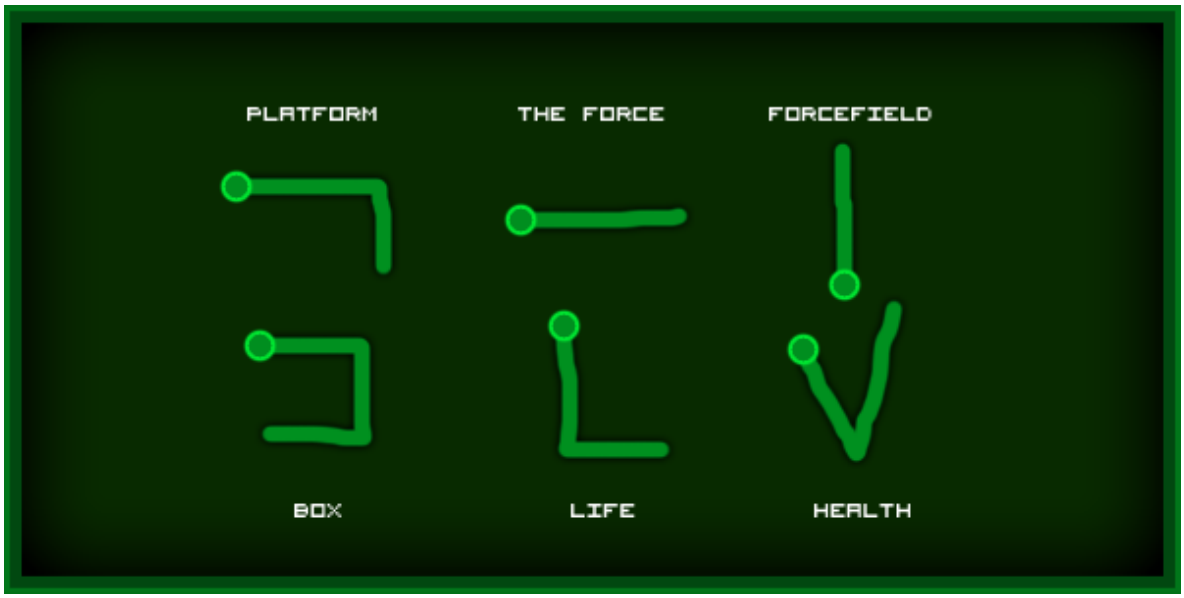
Het idee om gestures te gebruiken komt uit het voor die tijd heel innoverende Black & White (http://www.lionhead.com/bw/). In dit spel worden spreuken opgeroepen door de gestures in de wereld te tekenen. Dit leek ons een heel interessant idee om eens toe te passen in een platformgame (Trine doet dit trouwens ook in beperkte mate). Hiervoor maken we gebruik van aMiGolib (a Motion input Gesture output library). Deze is onderdeel van het bacheloreindwerk van één van de groepsgenoten, namelijk Robin Marx (http://bellerophon.be/gim/). Voor het inlezen van de gestures uit XML-files werd gebruik gemaakt van TinyXML (http://www.grinninglizard.com/tinyxml/).



Figuur 2.9: De gesture om een kubus te bekomen

Door gebruik te maken van deze libraries konden we gestures koppelen aan leuke gameplay features, zoals het aanmaken van nieuwe objecten of om de speler te helpen bij het verslaan van

vijanden of zijn health terug op een gezond pijl te krijgen. We zijn van mening dat de gestures een leuke insteek geven in dit soort spel en dat ze de speler heel wat meer mogelijkheden bieden om bepaalde puzzles of passages op te lossen. Waarom een zombie verslaan als je boven hem een platform kan tekenen en zo gewoon over hem springen? Gestures kosten echter wel mana, dus hierdoor kan het gebruik ervan wat worden ingeperkt, wat de balans van het spel ten goede komt.



Figuur 2.10: Verschillende gestures en hun resultaat

De gestures worden over het algemeen goed herkend maar soms zitten er toch kleine foutjes in, zeker als ze te snel gemaakt worden of niet helemaal goed worden afgerond. Zo krijg je heel af en toe een platform in plaats van een blok, maar dit is over het algemeen genomen zeldzaam en niet superstorend.

2.9 Kleinere extra's

Naast de bovenstaande besproken grote onderdelen zijn er ook nog een paar kleine extra's die we kunnen bespreken.

2.9.1 Geluid

FmodSoundComponent
SoundManager

Voor het geluid hadden we eerst openAL maar deze library was zeer moeilijk om mee te werken. Fmod bleek een stuk simpeler en zorgt in het spel voor achtergrondgeluid en geluidjes bij bepaalde gebeurtenissen. Via de FmodComponent kan men wachten op bepaalde events van andere components en daar dan gepaste geluidjes voor afspelen.

2.9.2 AI

AreaAIComponent
DumbAIComponent

Voor de AI hadden we niet echt uitgebreide algoritmen nodig. We zouden de zombies kunnen aansturen met een A* algoritme zodat ze de speler over het ganse level zouden volgen, maar we vonden dat dit maar weinig meerwaarde zou hebben. We hebben dan maar gekozen voor een eigen gemakkelijk algoritme. De zombies hebben een bepaalde streek in een levelblok waar ze op en af wandelen. Wanneer de speler binnen die streek komt gaan ze proberen de speler aan te vallen en er zo dicht mogelijk bij te raken. Via bepaalde parameters kan het gedrag van deze zombies toch uiteenlopend worden ingesteld zodat ze niet allemaal exact hetzelfde reageren.
Het leuke aan de AI was dat we de AreaAIComponent, die ervoor zorgt dat de zombies op en af lopen in hun streek, na een paar kleine aanpassingen konden gebruiken om er bewegende platformen mee te maken. De component was daar niet echt voor gemaakt maar doet zijn werk ook in andere omstandigheden dan het besturen van een zombie uitstekend.

2.9.3 Particles

ParticleEmitter
ParticleComponent
ParticleComponentEmitter

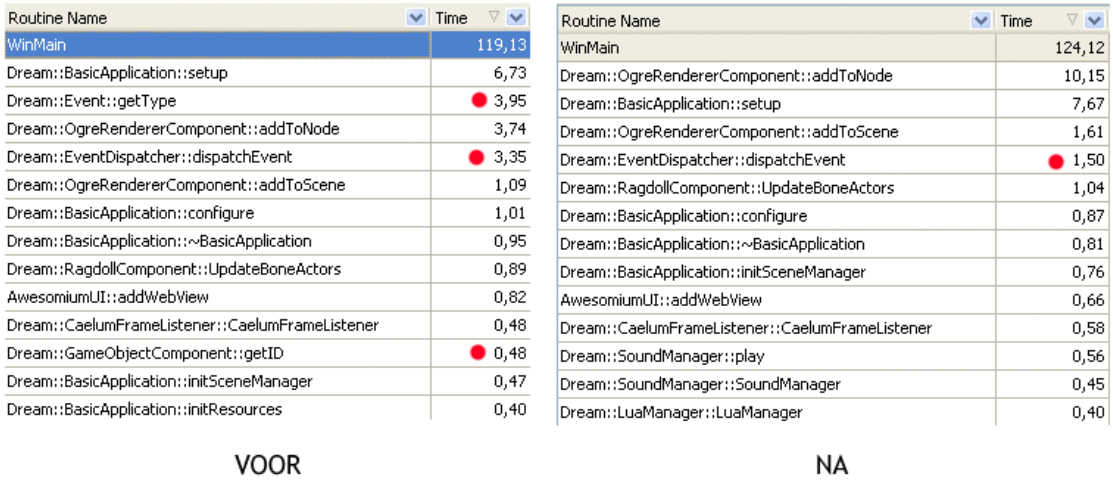
Nadat we van iemand uit een andere projectgroep hadden gehoord dat particles relatief simpel konden worden toegevoegd zijn we dit ook beginnen uitproberen. Het resultaat is dat we ze zeer veel gebruikt hebben in het spel. Ze helpen zeer goed bij het toevoegen van sfeer en geven de indruk dat de wereld echt leeft en in beweging is (zeker samen met de physics).
De particles zijn intern in Ogre aanwezig en zeer makkelijk op te roepen, We hebben heel wat van de voorbeeldscripts aangepast om onze eigen unieke particles te maken. Via de Component kunnen de particles aan een bepaald GameObject worden gehangen (zoals bijvoorbeeld op de platformen), terwijl de van GameObject afgeleide class ParticleEmitter zorgt voor particles los van een bepaald object, zoals de sneeuw en regen in de levels.

Hoofdstuk 3

Conclusie

3.1 Profiling

Na het schrijven van de laatste code hebben we een profiler gebruikt om te controleren waar er nog geoptimaliseerd kon worden (http://www.automatedqa.com/products/aqtime/). Het bleek dat vooral onze eventstructuur wat efficiënter kon en ook de AI deed een hoop overbodig werk.



Routine Name	Time
WinMain	119,13
Dream::BasicApplication::setup	6,73
Dream::Event::getType	3,95
Dream::OgreRendererComponent::addToNode	3,74
Dream::EventDispatcher::dispatchEvent	3,35
Dream::OgreRendererComponent::addToScene	1,09
Dream::BasicApplication::configure	1,01
Dream::BasicApplication::~BasicApplication	0,95
Dream::RagdollComponent::UpdateBoneActors	0,89
AwesomiumUI::addWebView	0,82
Dream::CaelumFrameListener::CaelumFrameListener	0,48
Dream::GameObjectComponent::getID	0,48
Dream::BasicApplication::initSceneManager	0,47
Dream::BasicApplication::initResources	0,40

VOOR

Routine Name	Time
WinMain	124,12
Dream::OgreRendererComponent::addToNode	10,15
Dream::BasicApplication::setup	7,67
Dream::OgreRendererComponent::addToScene	1,61
Dream::EventDispatcher::dispatchEvent	1,50
Dream::RagdollComponent::UpdateBoneActors	1,04
Dream::BasicApplication::configure	0,87
Dream::BasicApplication::~BasicApplication	0,81
Dream::BasicApplication::initSceneManager	0,76
AwesomiumUI::addWebView	0,66
Dream::CaelumFrameListener::CaelumFrameListener	0,58
Dream::SoundManager::play	0,56
Dream::SoundManager::SoundManager	0,45
Dream::LuaManager::LuaManager	0,40

NA

Figuur 3.1: Tijdswinst voor dispatchEvent. getType en getID zijn door inline-definitie uit de bovenste lagen van de lijst verdwenen.

Voor de events itereerden we in het begin over alle geregistreerde eventlisteners op GameObject en keken of ze geïnteresseerd waren in het huidige event. Dit gaf heel wat overhead omdat 90% van de listeners niet gebruikt moesten worden. Ook moesten we telkens ID's opvragen van de events. Dit zijn std::strings en hoewel die class redelijk efficient is, zijn onze ID's redelijk lang en als er zoveel events gestuurd worden (bijna een half miljoen op enkele minuten spelen).
Voor de AI werd er in elke frame opgevraagd in welke levelblok het AI object op dat moment was. Aangezien AI objecten nooit van blok kunnen veranderen tijdens het spel was het dus

veel beter dit gewoon 1 maal op te vragen en bij te houden voor de volgende frames. Dit verminderde de aanroepaantallen van de functie van ongeveer 7000 naar 10. Dit gaf ons meteen een extra grote optimalisatie omdat de functie die we gebruikten om de huidige blok op te vragen zelf niet zo efficient bleek. Nadat ook deze functie aangepast was zodat ze maar 1 maal de ID opvroeg van het gezochte object in plaats van in elke vergelijking, hadden we heel wat performantie-winst behaald.

Routine Name	Hit Count
Dream::Event::getType	7768946
Dream::GameObjectComponent::getID	852720
Dream::EventDispatcher::dispatchEvent	485148
Dream::Event::Event	267875
Dream::GameObject::dispatchEventToGOCs	217273
Dream::OgreFrameUpdateComponent::frameStarted	150533
Dream::FrameEvent::FrameEvent	150533
Dream::PhysXPhysicsComponent::onNewFrame	125966
Dream::GameObject::getGOC	88943
Dream::OgreRendererComponent::onOrientationChanged	64402
Dream::OrientationChangedEvent::OrientationChangedEvent	64402
Dream::PhysXPhysicsComponent::onOrientationChanged	64291
Dream::OgreRendererComponent::onPositionChanged	50675
Dream::PositionChangedEvent::PositionChangedEvent	50348

VOOR

Routine Name	Hit Count
Dream::Event::getType	718463
Dream::EventDispatcher::dispatchEvent	718463
Nocturnal::RefCountBase<Nocturnal::Void>::DecrRefCount	628356
Dream::Event::Event	402555
Dream::GameObject::dispatchEventToGOCs	315908
Dream::OgreFrameUpdateComponent::frameStarted	218670
Dream::FrameEvent::FrameEvent	218670
Dream::PhysXPhysicsComponent::onNewFrame	182940
Dream::GameObject::getGOC	127022
Dream::OgreRendererComponent::onOrientationChanged	94885
Dream::OrientationChangedEvent::OrientationChangedEvent	94885
Dream::PhysXPhysicsComponent::onOrientationChanged	94757
Dream::OgreRendererComponent::onPositionChanged	86404
Dream::PositionChangedEvent::PositionChangedEvent	86077

NA

Figuur 3.2: Een factor 14 minder hits wanneer getType in een lokale variabele bijgehouden wordt. getID verdwijnt helemaal van de bovenste rijen. De hogere aantallen voor de rest komen door een langere speelduur.

3.2 Net-niet-erin

Uiteindelijk zijn er ook enkele dingen die we in het begin wel wilden maken maar waar door redenen van tijdsgebrek of technische moeilijkheden geen resultaat is uitgekomen. Enkele voorbeelden zijn:

- Constructions : Het idee was om dozen en grotere structuren op te bouwen uit allemaal verschillende onderdelen. Op die manier zouden deze voorwerpen in stukken uiteen vallen als ze kapot gingen. We hebben hier de classes voor gemaakt en lang zitten testen maar onder meer door de beperking in diepte en het moeilijke werken in NXOGRE is deze feature nooit helemaal af geraakt.
- Co-op : Door de physics was het zeer moeilijk een werkende co-op spelvorm te bouwen via een netwerk. We hebben er lang over nagedacht en hulp gevraagd aan begeleiders en dan toch besloten er niet aan te beginnen.
- Besturing met wiimotes : De wii-remote leek ons een zeer interessant besturingsmiddel voor een spel zoals visitor. Ze kunnen via bluetooth gewoon aangesloten worden op de pc en de laatste versies van OIS hebben ondersteuning ervoor. Puur door tijdsgebrek is het ons niet gelukt dit erin te krijgen. We spendeerden liever iets meer tijd aan de afwerking van de bestaande implementatie.

- Beter uitgedachte levelopbouw : Dat informatici niet zomaar kunnen ingezet worden als leveldesigners is bij dit project ook weer duidelijk geworden. De gemaakte levels zijn vooral voor het testen van de verschillende features, niet om een aangenaam spel op te leveren. Met wat meer tijd hadden we een uitdagender versus-level kunnen opbouwen en misschien ook een singleplayer level met wat leuke puzzels.
- Afwerking van de graphics : Informatici zijn ook geen grafici. We konden wel een beetje overweg met 3D tools zoals Blender maar om hier professioneel ogende resultaten uit te krijgen is toch een beetje te hoog gegrepen, zeker op het gebied van textures en dergelijke. Ook hadden we graag wat meer en betere animaties en eigen modellen gehad. We zijn weliswaar zeer tevreden van ons werk op dit gebied maar het is nog altijd geen grafisch hoogstandje.
- Herinladen van een level : Ondermeer door het gebruik van NXOGRE bleek het moeilijk om alle destructors goed werkende te krijgen. Hierdoor kan men geen volledige levels opnieuw inladen en moet men het spel dus telkens opnieuw opstarten als men een nieuw level wilt beginnen. Hier hebben we heel veel tijd in gestoken maar uiteindelijk was er geen oplossing te vinden.

3.3 Uiteindelijke taakverdeling

Zoals reeds geopperd in de inleiding is de taakverdeling die we in het begin gemaakt hadden onder invloed van heel wat factoren veranderd. In de bijlagen vindt u gedetailleerde logs van onze activiteiten, maar hier is een kort overzicht:

Jimmy Cleuren :

- Physics met NXOGRE. Ragdolls, static physics, triggers, constructions, collisions, forceField
- Netwerk met RakNet : serializen, doorsturen, deserializen van levelblokken, gamesetup
- Eventstructuur
- Basis van sound
- Uitbreiden van camera
- Uitbreiden van health/mana/attack systeem voor Characters
- Verdere implementatie van destructors
- Code cleaning voor aflevering

Robin Marx :

- Basisstructuur voor GameObjects en GameObjectComponents
- Graphics met OgreRendererComponent, CharacterAnimationComponent
- Mouse en keyboard input, basis van camera

- Uitbreiden van levelstructuur, pagingmanager
- Scripting met LUA en LUABIND, bindings met c++, structuur van LUA-files, GameObjectFactory
- Platformen
- ID-structuur
- AI
- Verslag
- Basisimplementatie van destructors

Nick Michiels :

- Volledige GUI en editors met Awesomium
- Gestures met Amigolib, inlezen gestures via TinyXML
- Basisstructuur van level en levelblokken, renderen van levelblokken, afsluiten van levelblokken door muren, respawning en game-end
- Uitbreiden van camera
- Particles
- Uitbreiden van sound, SoundManager
- Basisimplementatie van destructors

Wouter Nivelle :

- Animeren cyborg model
- Level voorgrond en achtergrond meshes, object meshes zoeken en maken
- Content aanmaken in LUA, levelopbouw
- Onderzoek naar RakNet

3.4 Conclusie

We zijn allemaal zeer tevreden van ons werk aan dit project. We hebben veel tijd gestoken in de structuur en de opbouw van de code en dat heeft ons op het einde zeer goed geholpen om het spel snel maar goed af te werken. Ondanks de occasionele problemen met de gebruikte libraries hebben we ook op dat vlak zeer veel bijgeleerd.
We zijn er van overtuigd dat een project van deze omvang echt wel nood heeft aan bestaande en goed bevonden software pakketten en dit zal zeker ook in onze latere carriére ook het geval zijn. We menen dan ook dat een goede informaticus niet alleen zelf goed moet kunnen programmeren maar ook snel moet kunnen inzien hoe andere software en code werkt en hier goed mee overweg kan na slechts een kleine leerperiode. Het feit dat we een redelijk

goede kennis van alle gebruikte libraries hebben weten te bekomen in de beperkte tijd voor dit project vonden we een zeer positief iets voor onze ontwikkeling als informaticus en een aanduiding dat we goed bezig zijn.
Veel van de gebruikte technologieën zullen we zeker nog opnieuw gebruiken in latere projecten. Denken we hierbij zeker aan RakNet, Lua, Awesomium, Fmod en onze eventstructuur.
Alle leden van het team hebben goed hun bijdrage geleverd waarbij de samenwerking meestal zeer vlot ging, mede dankzij het gebruik van de SVN server en de wiki. Duidelijke afspraken in verband met de taakverdeling, deadlines en te gebruiken structuren hebben bijgedragen tot het vlot kunnen afwerken van dit project.

Hoofdstuk 4

Bijlagen

4.1 Logs

4.1.1 Jimmy Cleuren

19/02 2 Ogre tutorials bekijken + proberen
19/02 3 NxOgre aant het werk krijgen samen met Nvidia PhysX
21/02 4 EventDispatcher geschreven en getest
24/02 3 Opzoeken NxOgre collision detection
26/02 2 Structuur bespreken
02/03 3 EventDispatcher herzien zonder rare memory output
06/03 2 Onderzoek naar ragdolls in NxOGre
09/03 4 PhysicsComponent en PhysXPhysicsComponent gemaakt
11/03 3 CubeCharacter gemaakt om te testen + FmodSoundComponent
15/03 7 RagdollComponent
17/03 6 RagdollComponent
27/03 1 DebugFrameListener gemaakt
30/03 3 PhysxComponent draait tegoei mee met character
04/04 3 Collision detection proberen
05/04 4 Collision detection implementatie
06/04 5 Huiske bouwen met breakable joints
09/04 6 Ragdoll draait mee
12/04 4 Collisions ragdoll - gewone physx objecten part 1
13/04 2 Collisions part 2 met source en targetpart
14/04 1 Ragdoll paar foutjes
19/04 4 gravity tegoei, physx op terrein
21/04 3 ragdoll volgt nu ook het terrein en kan springe
21/04 2 broadcast server en client
22/04 1 gameserver
27/04 2 release build
29/04 1 ragdoll terrein laten volgen, memory leaks
03/05 3 netwerk gui + html cleanup
04/05 3 data sturen proberen + serialize levelbock en gameobject
05/05 2 rare bug van ragdoll met lua -¿ damn you setMass
06/05 2 ragdoll kan geen verticale muur meer op + serialize gameobject

07/05 2 EventDispatcher : removeEventListenersFor
08/05 4 Netwerk: serialize/deserialize + chat
09/05 2 Z range limit door onzichtbare muren
14/05 4 collisions implemenatie aangepast ahv IDUtility
14/05 2 Triggers
15/05 5 physx + ragdoll component aangepast aan geneste scenenodes
18/05 8 DEATH event, applyForce op physx, vechten in gameplay verwerken door colisions, schade doen, globale singleton voor veel te gebruiken data
19/05 10 vechten in gameplay, vijanden kunnen dood gaan, health en mana systeem, smooth camera, schade door dooskes, health door orbs, force y positie op ragdoll en physx
20/05 10 manabonus door vijand te doden, destructor bug zoeken
21/05 10 schade door blokken, destructor trigger, netwerk luistert op events als spel gestart is en netwerk spel, ontvange blocks die nog nie ingelade zijn worden opgeslaan, physx bij nieuw platform ook kapot doordat y positie niet kan geupdate worden, forcefield gesture, nieuw ontvangen objecten via netwerk aanmaken, extra netwerkevents
22/05 10 netwerk events, serialize, aardbeving, kleine foutjes
23/05 10 forcefield, netwerk, lobby, force y op ragdoll, kleine foutjes + testen
24/05 5 testen, profiling + optimalisaties

4.1.2 Robin Marx

Donderdag 19 februari 6 wiki opzetten + doelstellingen + taakverdeling
Woensdag 23 februari 2 LUA en LUABind onderzoek
Dinsdag 24 februari 8 LUA en LUABind onderzoeken en gecompileerd krijgen
Woensdag 25 februari 4 Ogre basisimplementatie
Donderdag 26 februari 3 PhysX en Fmod installatie
Donderdag 26 februari 3 Opbouwen structuur inputControllers en FrameListeners
Vrijdag 27 februari 7 inputcontrollers, framelisteners en GameObjects
zaterdag 28 februari 2 GameObjects en components
zondag 1 maart 7 GameObjects en components, Controllers
maandag 2 maart 6 Controllers en Listeners
dinsdag 3 maart 3 Aether structuur erin brengen
donderdag 5 maart 4 FreeMoveComponent
zondag 15 maart 4 before en afterParentChange, CharacterAnimationComponent
maandag 16 maart 3 CharacterAnimationComponent
maandag 16 maart 2 SideScrollingCamera
zaterdag 28 maart 3 BasicApplication
zaterdag 28 maart 2 LUA voor gameobjects - begin
maandag 30 maart 3 LUA voor gameobjects
dinsdag 31 maart 2 LuaManager
dinsdag 31 maart 5 File (haha ja, da ding deed zeeer moeilijk)
woensdag 1 april 4 GameObjectFactory
zaterdag 11 april 2 ScriptComponent
zondag 12 april 6 levels in LUA
vrijdag 24, zaterdag 25 april 9 ID's en PagingManager
maandag 4 mei 3 Release build en Config

maandag 4 mei 3 Lua bindings en CharacterAnimationComponent
dinsdag 5 mei 6 Allerhande aanpassingen, voornamelijk LUA
donderdag 7 mei 4.5 Cleanen van destructors, reload level
woensdag 13 mei 3 Release build terug werkende krijgen, LUA uitbreiden
donderdag 13 mei 6 Triggers, AI, particles, LUA uitbreiden
vrijdag 14 mei 8 Triggers, AI, particles, LUA uitbreiden
vrijdag 14 mei 8 Triggers, AI, particles, LUA uitbreiden
maandag 18 mei 6 levelstructuur voorbereiden op netwerk
dinsdag 19 mei 8 levelstructuur voorbereiden op netwerk
woensdag 20 mei 4 allerhande aanpassingen, radius
donderdag 21 mei 9 allerhande aanpassingen, AI, voorbereiding netwerk, netwerk
vrijdag 22 mei 9 AI, netwerk, bewegende platformen
zaterdag 23 mei 9 AI, verslag
zondag 23 mei 8 verslag

4.1.3 Nick Michiels

Donderdag 19 februari 10:00 - 12:30 2:30 Ogre bestuderen (tutorials)
Donderdag 19 februari 13:00 - 14:30 1:30 Ogre bestuderen (tutorials) + algemene concepten bestuderen
Maandag 23 februari 9:30 - 11:30 2:00 Zoeken naar Awesomium voor de GUI
Maandag 9 maart 18:00 - 22:00 4:00 Zorgen dat alle libraries zijn geinstalleerd + opzoekwerk awesomium + klein UI uitwerken
Donderdag 13 maart 19:00 - 21:00 2:00 Awesomium invoegen in ons project (nog een probleem bij rendering, geeft enkel wit beeld)
Maandag 16 maart 18:30 - 21:00 2:30 Rendering van awesomium gefixed + Framelistener, keylistener en mouselistener toegevoegd aan awesomium
Dinsdag 31 maart 14:00 - 15:00 1:00 Samenkomst groep voor bespreking vorderingen en technieken
Dinsdag 31 maart 15:30 - 16:40 1:10 Proberen muis op te zetten: niet gelukt
Donderdag 2 april 10:00 - 12:30 2:30 Uittekenen objectstructuur voor level en interpreteren van andere klasses
Zaterdag 4 april 10:30 - 12:20 1:50 Basisklasses voor low-level objectstructuur van level (Level, LevelBlock, LevelBlockTerrain)
Zaterdag 5 april 14:00 - 17:30 3:30 Verdere uitwerking van levelstructuur naar high level voor ogre
Zondag 5 april 11:00 - 13:00 2:00 Renderingsysteem van ogre voor level uitgewerkt
Zondag 5 april 15:00 - 16:40 1:40 Debugging van renderingsysteem level
Dinsdag 7 april 10:00 - 12:30 2:30 Displacements van de levelblocks goed zetten -¿ op basis van boundingbox van voorgaande
Dinsdag 7 april 13:00 - 14:30 1:30 Foreground werkt, nu ook getracht de background toe te voegen maar hij wil deze niet aan dezelfde node hangen. (offset gebeurt in de negatieve Z richting).
Donderdag 9 april 10:30 - 13:00 2:30 Facade voor awesomium (gemakkelijk nieuwe webviews aanmaken) -¿ probleem met het redrawen
Donderdag 9 april 13:20 - 17:50 4:30 Facade voor awesomium verder afwerken + toevoegen

in structuur van onze applicatie
Donderdag 9 april 18:15 - 18:50 0:35 Problemen met het renderen van level opgelost. Displacement van achtergrond werkt nu (probleem was het maken van een node per entity + schaal en positie van objecten correct voor het aanroepen van de rendering (opgelost door extra test op NULL).
Maandag 13 april 12:30 - 12:50 0:20 Scenemanager onafhankelijk gemaakt van level maar afhankelijk van rendering level
Dinsdag 14 april 13:10 - 14:19 1:09 muis opzetten + afmetingen muisinput juist gezet + keyhandlers voor awesomium via keyboardhook
Dinsdag 15 april 14:20 - 17:15 2:55 file inlezen en uitschrijven + LUA via awesomium inlezen aanpassen en wegschrijven
Dinsdag 21 april 18:10 - 22:30 4:20 html coderen voor awesomium + graphics in photoshop
Zaterdag 25 april 13:20 - 15:45 2:25 entity editor: html + lua inladen met behulp van picking
Zaterdag 25 april 16:10 - 17:55 1:45 entity editor: html + lua inladen met behulp van picking
Zondag 26 april 9:30 - 12:00 1:45 entity editor: lua save + lua load + javascript pop-up + load from file + drop down
Zondag 26 april 12:30 - 18:00 5:30 entity editor: lua save + lua load + javascript pop-up + load from file + drop down + current level editor + dynamic block editing
Zondag 3 mei 9:30 - 12:30 3:00 destructors van level fatsoenlijk opbouwen + deleten van scene (je moet altijd node deleten en voor levelobjects moet je de entity deleten via de node en de renderer uit de scenemanager smijten
Zondag 3 mei 13:10 - 17:45 4:35 verderwerken aan de destructors van level + lua file herinladen van level (werkt), maar hij loopt vast bij het de rendering van nieuwe level, ik vermoed dat het ligt aan de framelistener en/of aan het main character
Maandag 4 mei 8:30 - 11:45 3:15 resize van awesomium bijgewerkt -¿ zeker zien dat de dll's van MVS9 aanwezig zijn + herloaden nog wat bijgewerkt
Maandag 4 mei 12:15 - 17:00 4:15 nog kleine foutjes in resize + html nog wat aangepast zodat ook de editor in een iframe zit + debugging reloaden level
Maandag 4 mei 19:20 - 21:25 2:05 leveleditor uitgebreidÉ samengevat zijn de mogelijkheden: 1) bij choose level kunde een level kiezen uit de map van levels, deze kan je inladen door edit this level te drukken, 2) edit current level gaat de lua van de huidig ingelade level in ogre openen, 3) existing levelblocks bevat alle levelblocks aanwezig in de overeenkomstige map, ook deze kan kan je hier editen door op de knop te drukken, 4) in de current file staat altijd de filenaam van het huidig geopende lua bestand. Deze gaat ook gebruikt worden om de lua file op te slaan, 5) naargelang je een level of levelblock aant editen ben kan je "save lua of levelöf "save current levelblock"kiezen. Deze wijst zijneigen uit, hier heb ik nog een klein popupje voor gemaakt zodat ge nog kunt annuleren. Let op: zoals ik terjuist al zeiÉ hij gaat in de current file opslaan (goed zien da je niets verkeerd doet), 6) reload level in ogre zal uiteindelijk de huidige ingeladen lua file van level inladen in ogre, maar deze werkt nog niet
Dinsdag 5 mei 20:30 - 0:00 3:30 destructors cleanen + scenenodes cleanen voor herinladen
Woensdag 6 mei 9:00 - 11:00 2:00 destructors cleanen + herladen
Maandag 11 mei 10:30 - 11:30 1:00 Menu mouse over + uitbreiding singleplayer
Maandag 11 mei 12:00 - 17:30 5:30 Entity uitgewerkt met een previeuw van de entities -¿ 1 background plane en 1 ground plane waar de entity opstaat + camera gemanipuleerd voor ze mooi in beeld te krijgen. Heet wat moeilijkheden met die planes en staat nog niet helemaal op punt. De entities durven nog naar voor schuiven
Maandag 11 mei 19:00 - 21:00 2:00 Picking uitgewerkt zodat ook de de preview van entity

daarop reageert
Donderdag 14 mei 10:00 - 11:30 1:30 Ogre Particle System getest
Donderdag 14 mei 12:30 - 17:15 4:45 Ogre Particle System getest + geimplementeerd als gameobjectcomponent in ons systeem
Zondag 17 mei 9:30 - 13:30 5:00 Particle script ingesteld (rain, snow, campfire, campfiresmoke) + character events
Zondag 17 mei 14:00 - 19:00 5:00 Character events uitsturen en opvangen en deze opvangen in de UI. Vooral opvangen in UI geeft veel problemen. Javascriptinstellingen voor healthbar en manabar aan te passen heeft veel tijd gekost. + entity editor (vooral in samenwerking met physics)
Zondag 17 mei 19:30 - 20:12 0:42 Entity editor in combinatie met physics werkende gekregen + enkele foutjes eruit gehaald waardoor picking nu beter werkt
Maandag 18 mei 14:15 - 14:45 0:30 Team meeting en werkverdeling
Maandag 18 mei 14:45 - 18:00 3:15 Gesture herkenning met de Amigo library + XML parser voor gestures in te lezen met TinyXml
Maandag 18 mei 18:30 - 0:30 6:00 Gesture herkenning met de Amigo library + acties op gesture zoals aanvallen + als er een vierkant wordt getekend een solidbox laten vallen op de plaats waar het vierkant is getekend. Hiervoor moet een planehittest worden uitgevoerd. Ook particlecomponent uitgebreid zodat er een duration op de particle kan gedefineerd worden
Dinsdag 19 mei 09:30 - 11:30 2:00 picking toegevoegd aan gestures zodat ook gestures op bepaalde objecten kunnen uitgevoerd worden
Dinsdag 19 mei 12:30 - 18:00 5:30 gesture interactie verder uitgewerkt + fmodsoundcomponent uitgewerkt + koppeling met UI verbeterd
Dinsdag 19 mei 19:00 - 23:30 4:30 interactie sounds en particles verbeterd + timeout particles in lua + restart particles
Woensdag 20 mei 19:00 - 0:15 5:15 foutjes uit GUI, mouseemitter gemaakt + soundmanager gemaakt
Donderdag 21 mei 9:30 - 13:00 3:30 footstep sound werkt nu enkel als er gelopen wordt, gesture om platform te tekenen + starten aan netwerklobby
Donderdag 21 mei 13:30 - 18:00 4:30 netwerk lobby voor server en client afgewerkt + mana en health change van gestures in xml van gesture gestopt en uitgevoerd op character
Donderdag 21 mei 18:40 - 23:20 4:40 debugging aan lobby + sound nog wat bijgeschaafd + andere details
Vrijdag 22 mei 9:30 - 12:15 2:45 Kleine foutjes uit GUI gehaald + force gesture verder utigewerkt met radius en sound
Vrijdag 22 mei 13:15 - 16:20 3:05 aanvallen met linkermuis, gestures met rechtermuis + scrolling voor in en uit te zoomen + error geluid als gesture niet mag + GUI errors
Vrijdag 22 mei 20:05 - 01:30 5:25 Z begrenzing voor camera's, doublejump beperking + chat aangepast zodat manneke nimmer meebeweegt
Zaterdag 23 mei 9:40 - 01:00 3:20 Achtergrond + chat koppelen met toets en corrigeren voor multiplayer
Zaterdag 23 mei 13:30 - 18:30 5:00 Muren + animatie gezet bij levelblockovergangen (rekening houden of het van links naar rechts gaat of omgekeerd) + finish object gezet en kijken of de speler finisht (ook koppeling met GUI) + respawnen aan het begin van blok na een leven kwijt
Zaterdag 24 mei 19:40 - 0:30 5:20 Verder werken aan respawning + netwerktest + kleine foutjes eruit halen

Zondag 25 mei 15:50 - 18:00 2:10 Enkele algemene tests van UI + testen voor het maken van een youtubefilmpje
Zondag 25 mei 19:00 - 21:00 2:00 Editor tests + oplossen

4.1.4 Wouter Nivelle

Donderdag 19 februari 2 Ogre verkrijgen 2,5 Ogre bestuderen + wiki lezen
Maandag 23 februari 1,75 3ds Max bekijken
Woensdag 23 februari 2 3ds Max tutorials proberen 1,25 3ds Max tutorials proberen
Woensdag 23 februari 5 Level creeeren + animatie uitproberen
Donderdag 26 februari 2 Structuur bespreken
Donderdag 5 maart 2 3ds max uninstallen en blender afhalen / bestuderen 1 blender naar ogre meshes exporter aan het werken krijgen 5 Blender tutorials
Woensdag 11 maart 2,5 voorlopige levels aan het werken krijgen in Ogre
Woensdag 11 maart 2,5 voorlopige levels aan het werken krijgen in Ogre
Zaterdag 14 maart 3 Texturen / materials renderen
Vrijdag 20 maart 4 Blender tutorials voor subsurf modelling uitproberen
Donderdag 2 april 1 Level met material aan het werken gekregen in Ogre zodat Nick verder kon
Zaterdag 4 april 1 Terreinklasse toegevoegd
Dinsdag 7 april 3 Aanpassen levels voor het wegkrijgen spleten en texturen aan het werken proberen te krijgen
Dinsdag 14 april 3,25 RakNet aan het werken krijgen
Dinsdag 21 april 2 Serializen, deserializen
Zaterdag 25 april 5 UV wrapping / unwrapping tutorials in blender voor texturen op meshes
Dinsdag 28 april 3,5 Levels bewerkt, in Ogre gekeken en online gezet
Donderdag 30 april 4,25 animatie tutorials 2 Nog meer animatie tutorials
Vrijdag 1 mei 5 Zoeken van modellen, aanpassen van modellen, animeren van die modellen
Zaterdag 2 mei 4 Animeren modellen + combineren met ragdoll
Zondag 3 mei 2 Aanpassen bottenstructuur Cyborg
Maandag 4 mei 2 Bottenstructuur weer aanpassen voor cyborg
Vrijdag 8 mei 4 2 nieuwe backgrounds + animatie verbeterd
Dinsdag 12 mei 1 Kleine lua update met blocks
Donderdag 12 mei 3 Dieper front mesh zodat er geen leegte meer is
Zaterdag 16 mei 4 Werken aan level opbouw en modellen
Zondag 17 mei 2 Level afwerken
Maandag 18 mei 4 Orbs (piramides) gemaakt + werken aan kubus
Dinsdag 19 mei 5,5 Algemeen werken aan level + animaties + modellen
Dinsdag 19 mei 6 Algemeen werken aan level + animaties + modellen
Donderdag 21 mei 5 Opbouw level, nieuwe modellen, extra texturen voor modellen etc, animaties
Vrijdag 22 mei 5,5 Level2 en level3 bijgemaakt met bijhorende opbouw + wat algemene bugfixes
Zaterdag 23 mei 4 Algemene bugfixes
Zondag 24 mei 3 Algemene bugfixes in level + platforms verbeteren