

Zombieland

**Gemaakt door:**

Wouter Grutter  
Olivier Lindenbergh  
Stefan van der Ham

## Versiebeheer

|  |  |
| --- | --- |
| 22-11-16 | Eerste inlevering Documentatie. Plan van aanpak, functioneel ontwerp gemaakt en technisch ontwerp mee bezig. |
| 29-11-16 | Technisch ontwerp toegevoegd, Inleiding kopjes bij functioneel ontwerp en technisch ontwerp, kopje ontwikkel omgeving bijgewerkt, prioritering van de product backlog gemaakt, volgorde van sprints verbeterd, layout en verbeterd en plaatjes toegevoegd. Class diagram en test plan toegevoegd. |
| 6-12-16 | Layout verbeterd. |
| 14-12-16 | Client – Server relatie toegevoegd.  Sprint 1 afgerond en ingevuld.  Logboek geupdate. |

Inhoudsopgave

[Versiebeheer 1](#_Toc469522094)

[Project voorstel 4](#_Toc469522095)

[Informatieanalyse 8](#_Toc469522096)

[Plan van aanpak 9](#_Toc469522097)

[Doel reden van de opdracht 9](#_Toc469522098)

[Spel idee 9](#_Toc469522099)

[Organisatie 9](#_Toc469522100)

[Eindproducten 9](#_Toc469522101)

[Activiteiten en planning 10](#_Toc469522102)

[Taken en activiteiten: 10](#_Toc469522103)

[Globale planning 10](#_Toc469522104)

[Begroting 11](#_Toc469522105)

[Deadlines 11](#_Toc469522106)

[Randvoorwaarden 11](#_Toc469522107)

[Inrichting van de ontwikkelomgeving 11](#_Toc469522108)

[Functioneel ontwerp 13](#_Toc469522109)

[Product backlog 13](#_Toc469522110)

[Ontwerpen van de schermen 14](#_Toc469522111)

[Menustructuur 14](#_Toc469522112)

[Gameplay 15](#_Toc469522113)

[World/levels 15](#_Toc469522114)

[Karakters 17](#_Toc469522115)

[Technisch ontwerp 18](#_Toc469522116)

[Activity diagram 18](#_Toc469522117)

[Class diagram 19](#_Toc469522118)

[Testplan 24](#_Toc469522119)

[Client – Server relatie 24](#_Toc469522120)

[Voordelen 24](#_Toc469522121)

[Nadelen 24](#_Toc469522122)

[Conclusie 25](#_Toc469522123)

[Sprints 26](#_Toc469522124)

[Sprint backlog 26](#_Toc469522125)

[Sprint 1: 27](#_Toc469522126)

[Sprint backlog 27](#_Toc469522127)

[Codevoorbeelden 27](#_Toc469522128)

[Testresultaten 31](#_Toc469522129)

[Review 31](#_Toc469522130)

[Retrospective 31](#_Toc469522131)

[Sprint 2: 32](#_Toc469522132)

[Sprint backlog 32](#_Toc469522133)

[Codevoorbeelden 32](#_Toc469522134)

[Testresultaten 32](#_Toc469522135)

[Review 32](#_Toc469522136)

[Retrospective 32](#_Toc469522137)

[Sprint 3: 33](#_Toc469522138)

[Sprint backlog 33](#_Toc469522139)

[Codevoorbeelden 33](#_Toc469522140)

[Testresultaten 33](#_Toc469522141)

[Review 33](#_Toc469522142)

[Retrospective 33](#_Toc469522143)

[Bijlagen: 34](#_Toc469522144)

[Notulen: 34](#_Toc469522145)

[Logboek 36](#_Toc469522146)

# Project voorstel

**Inleiding**

Ons projectteam bestaat uit 3 personen. Olivier, Stefan en Wouter.

In dit document gaan wij ons eindproject voorstel beschrijven. Hierin gaan wij het hebben over welke technieken wij gaan gebruiken in ons project en team. Verder gaan wij uitgebreid ons project-idee beschrijven.

**Projectteam**

Ons projectteam bestaat uit 3 personen:

* Wouter Grutter

**Opleiding:** GameDeveloper

**Leerling nummer:** 400017487

* Stefan van der Ham

**Opleiding:** GameDeveloper

**Leerling nummer:** 400017683

* Olivier Lindenbergh

**Opleiding:** GameDeveloper

**Leerling nummer:** 400018155

**Gegevens**

**Klant:**

Christiaan Albers  
Wooden Planks (Co-Founder)

**Doelgroep:**

Onze doelgroep is vanaf 16 jaar en ouder.  
Voor informatieanalyse en testen gebruiken wij de ICT klassen op ROCA12 in Ede.

**Doel / Reden**

Wij willen laten zien wat wij kunnen en tijdens ons school traject hebben geleerd. Aan het eind willen wij een product opleveren waar in te zien is wat wij hebben gemaakt met de kennis die wij op school hebben geleerd.

**Taakverdeling**

|  |  |
| --- | --- |
| Render engine | **Wouter Grutter** |
| Sound engine | **Wouter Grutter** |
| Character development | **Stefan van der Ham** |
| Level design | **Olivier Lindenbergh** |
| World collision | **Olivier Lindenbergh** |
| Other collision | **Olivier Lindenbergh** |
| Weapon technique development | **Stefan van der Ham** |
| Physics engine implementation | **Wouter Grutter** |
| NPC development | **Stefan van der Ham** |
| AI Development | **Wouter Grutter** |
| Multiplayer client-side development | **Wouter Grutter** |
| Multiplayer server-side development | **Olivier Lindenbergh** |
| Model, texture and sound creation | **Stefan van der Ham** |

**Projectbeschrijving**

Ons idee is een 3D game gemaakt met OpenGL, OpenAL en een physics engine.  
De game is een first person zombie survival game tijdens de nacht.

**Basic Game**  
Het basis spel heeft één gebied waarin gespeeld kan worden. Het thema van het gebied is het bos. In het bos is een hutje waarin de speler kan teleporteren naar het safe gebied. Hier kan de speler ammo voor zijn wapens kopen.   
In het basisspel kan de speler in het bos de zombies verslaan en hier punten mee verdienen. Hier kan de speler dan weer wapens en ammo van kopen.  
Verder hebben wij multiplayer ingebouwd waardoor de speler met anderen kan spelen en de zombies verslaan.  
De zombies komen altijd naar de dichtstbijzijnde speler toe in een rechtstreekse motie. Hoe langer je levend blijft, hoe moeilijker het wordt. Het spel wordt moeilijker doordat de zombies steeds sterker en sneller worden.   
Wij doen alle physics en andere berekeningen die nodig zijn voor alle speler in het spel op de server. Hierdoor zorgen wij ervoor dat de spelers sowieso niet vals kunnen spelen en dat alle clients minder werk hebben met de berekeningen die elke frame gedaan moeten worden.  
Op de clients sturen wij dan ook alleen de input door naar de server. Voor de rest renderen wij op de clients alle objecten en spelers op de goede plekken.

**Extended Game**Bij de map is een extra gebied gekomen. Het thema van dit gebied is een casino.   
De speler kan niet zelf naar andere gebieden lopen.  
Om van gebieden te wisselen gebruikt de speler een teleporter.  
De collision boxes zijn beter afgesteld en overal betere collision.  
De zombies hebben een beter pathfinding gekregen zoals A\*.  
In de gebieden zijn meer wapens beschikbaar en de speler kan een speciaal wapen krijgen door meerdere onderdelen te vinden en in het safe gebied het wapen te craften.

**Full Game**Bij de map is nog een extra gebied gekomen. Het thema van dit gebied is een treinstation. De teleporten tussen de gebieden zijn vervangen door een trein. Deze trein rijdt op een vast schema rondjes tussen alle gebieden.  
Vanaf het treinstation is er een 2e trein die vertrekt van een ander spoor die de speler naar het safe gebied brengt.  
De treinen wachten nooit op de spelers en wanneer het tijd is om te vertrekken gaat de trein ook.  
Wapens kan je in het safe gebied upgraden met items die je kan vinden in de gebieden.

**Op te leveren producten**

Aan het einde van dit project hebben wij de volgende producten om op te leveren.

* Documentatie
* Game met alle gameplay elementen
* Minimaal de basic game en de game is uit te breiden naar de extendedof full game.
* In de game moeten de omgevingen goed gemaakt zijn.
* Een server met alle functionaliteit die nodig is om naar de client te sturen.

|  |  |  |
| --- | --- | --- |
| **Kerntaak** | **Werkproces** | **Projectverantwoording** |
| Kerntaak 1 Ontwerpen van de applicatie, (cross)media-uiting of game | 1.1 werkproces: Stelt de vraag en/of informatiebehoefte vast | Voor het project is er overleg met de opdrachtgever en onze bevindingen worden samen gevoegd in een mindmap en de beschrijving van het project in het PvA |
| 1.2 werkproces: Maakt een plan van aanpak | Er wordt een PvA gemaakt en besproken met de klant. Hierop geeft de opdrachtgever een go/no go |
| 1.3 werkproces: Levert een bijdrage aan een functioneel ontwerp | Het projectteam levert een basisontwerp op waarin de functionaliteit van de applicatie wordt beschreven. |
| 1.4 werkproces: Maakt een technisch ontwerp | Het projectteam levert een technisch ontwerp op waarin de technische onderdelen van het project worden besproken. Hier komt ook het Classen diagram in |
| 1.5 werkproces: Richt een ontwikkelomgeving in | Het projectteam richt ieder zijn ontwikkelomgeving in en laat dit weten aan het team. |
| Kerntaak 2 Realiseren van de applicatie, (cross)media-uiting of game | 1.1 werkproces: Programmeert games of game-onderdelen | Ieder lid van ons projectteam is verantwoordelijk voor een deel in de ontwikkeling van de game. Wij gaan de delen verdelen in onze documentatie, daarna maken wij een planning waarin wij per persoon duidelijk maken waar hij verantwoordelijk voor is en wanneer dit af moet zijn. |
| 1.2 werkproces: Test het ontwikkelde product | Tijdens de ontwikkeling gaan wij het product op meerdere momenten volledig testen. Wij hebben verschillende tests die wij dan gaan uitvoeren. Een functionele test, technische test en wij gaan het product testen met de doelgroep waardoor wij duidelijk resultaten krijgen waarmee wij de game kunnen verbeteren. |
| 1.3 werkproces: Optimaliseert de game | Om het product goed te laten werken moeten wij het goed optimaliseren. Dit gaan wij van tevoren in de documentatie uitleggen en tijdens de ontwikkeling implementeren. |
| 1.4 werkproces: Bewaakt de voortgang en evalueert het project | Wij gaan de voortgang bewaken en het project evalueren door onder andere vaak gesprekken te hebben met de klant. Hierdoor krijgen wij duidelijkheid over hoe de voortgang in het project is en wat de klant ervan vindt. Ook denken wij na over de benodigde tijd die wij nodig hebben per deelproduct in de documentatie.  Tijdens ons project gaan wij gebruik maken van de projectmethode Scrum. Voor de planning van alle deelproducten en onderdelen die in het eindproduct komen gaan wij gebruik maken van Trello. |
| Kerntaak 4 Onderhouden en beheren van de applicatie, (cross) media-uiting of game | 1.3 Documenteert en archiveert gegevens | Het projectteam maakt een uitgebreide documentatie waarin onder andere de volgende onderdelen aan bod komen:   * Vraag en informatieanalyse * Plan van aanpak * Functioneel ontwerp * Technisch ontwerp * Testen   Daarbij gebruiken wij in dit project de techniek Scrum. Hiervoor hebben wij ook nog de volgende punten:   * Product backlog * Sprint planning * Sprint backlog * Sprints * Testen per sprint * Review * Retrospective |

# Informatieanalyse

Wij hebben met het team overleg gehad met de klant(Christiaan Albers).  
in dit overleg hebben wij ons project voorstel besproken en bepaald of dit genoeg is of te veel voor drie mensen. Christiaan heeft gezegd dat ons totaal plaatje wat te veel is voor drie mensen dus hebben wij het project opgedeeld in drie delen. Het eerste deel is wat wij minimaal moeten maken om aan het eind van deze periode een spel in te leveren. In het tweede deel breiden wij het spel uit naar een spel dat ons goed haalbaar lijkt met wat geavanceerdere technieken en meer gameplay. In het derde voorstel breiden wij het spel nog meer uit door het spel mooier te maken en verschillende gameplay aspecten te verbeteren waardoor het een beter en leuker spel wordt.

# Plan van aanpak

## Doel reden van de opdracht

Het doel van ons project is om te laten zien wat wij hebben geleerd zowel in programmeren als in samenwerken. Ook willen wij laten zien dat we in een bepaalde periode een goed product kunnen opleveren.

## Spel idee

In ons projectvoorstel wordt het spelidee in groot detail beschreven, hier kunt u dan ook het spel idee vinden.   
[Klik hier om naar het projectvoorstel te gaan.](#_Project_voorstel)

## Organisatie

De klant is 2 keer per week aanwezig om mee te kunnen praten. Wanneer er plotselinge veranderingen of problemen op komen, kunnen we hem altijd via de mail nog benaderen.

Het project team bestaat uit 3 personen. Wij hebben elke ochtend een scrum meeting waarin wij de voortgang, veranderingen en problemen met elkaar bespreken. Wanneer er iets verkeerd dreigt te gaan, kunnen we het met het hele projectteam oplossen.

Na elke sprint hebben wij een oplevering van een deelproduct. Dan hebben we opnieuw een meeting om te bekijken of het deelproduct voldoet aan de eisen.

## Eindproducten

Het eindproduct moet minimaal voldoen aan de volgende eisen:

* Core gameplay moet goed en vloeiend werken.
* Verschillende events moeten geluid effecten geven waardoor het spel een stuk duidelijker en leuker wordt.
* Er komen visual effecten in waardoor het spel er een stuk beter uitziet en leuker is om te spelen.
* Een goed werkend menu met opties en help. Daarnaast kun je in het menu ook het spel afsluiten of starten.

Tussentijds hebben wij na elke sprint een werkend deelproduct die word opgeleverd. De documentatie moet zo snel mogelijk worden opgeleverd zodat het voor de opdracht gever duidelijk is wat er gaat gebeuren en hoe het spel gemaakt gaat worden.  
Voor elk deelproduct wordt aan het begin van de sprint een doel gezet met een sprint backlog en een sprint planning. Hierdoor hopen wij elke sprint een goed deelproduct te kunnen opleveren met elke keer meer functionaliteit.

## Activiteiten en planning

### Taken en activiteiten:

* Plan van aanpak
* Functioneel ontwerp
* Technisch ontwerp
* Ontwikkelomgeving
* Testplan
* Realisatie
* Testen
* Oplevering

### Globale planning

|  |  |  |
| --- | --- | --- |
| Week | Items | |
| Week 1 14-11-2016 | Begin documentatie |  |
| Week 2 21-11-2016 | Finalizing documentatie | Start sprint 1 |
| Week 3 28-11-2016 | Werken aan sprint 1 | Einde sprint 1 |
| Week 4 05-12-2016 | Start sprint 2 |  |
| Week 5 12-12-2016 | Werken aan sprint 2 | Einde sprint 2 |
| Week 6 19-12-2016 | Start sprint 3 |  |
| Week 7 26-12-2016 | Werken aan sprint 3 | Einde sprint 3 |
| Week 8 02-01-2017 | Begin Testen |  |
| Week 9 09-01-2017 | Testresultaten bekijken en verwerken | |
| Week 10 16-01-2017 | Inleveren |  |

### Begroting

|  |  |
| --- | --- |
| **Soorten kosten** | **Kosten** |
| Personeelskosten | €96.000,- |
| Apparatuurskosten | €2.400,- |
| Softwarekosten | €1.938,- |
| Onverwachte kosten | €8.000,- |
| **Totale kosten:** | €108.338,- |

**\* Personeelskosten:**

* Uurloon: 80 euro per uur.
* Aantal uren: (10 weken) 400 uren pp
* Aantal werknemers: 3

**\* Apparatuurskosten:**

* Elk team lid moet een goede computer hebben om op te werken.

**\* Softwarekosten:**

* Visual studio professional voor elk teamlid : €646,- \* 3

**\* Onverwachte kosten:**

Deadlines

Het is de bedoeling dat het product opgeleverd wordt op 25 januari 2017.  
Het product dat opgeleverd moet is eerder in deze documentatie al beschreven. Het zijn alle spel eisen met de documentatie.  
De documentatie Willen wij en nagekeken hebben voor het eind van de tweede week zodat de klant de planning en de kosten weet.

## Randvoorwaarden

Welke software en hardware hebben we nodig om het product goed op te leveren.  
Elk team lid heeft een computer nodig en een plek om te werken.  
De software die wij gebruiken om het spel te maken zijn de volgende:

* Visual studio community 2015.
* Voor de documentatie gebruiken we Microsoft office.
* Voor 3D modellen gebruiken wij 3DS max 2015.
* Voor de textures gebruiken wij Photoshop.

## Inrichting van de ontwikkelomgeving

Wij gebruiken de volgende software:

* Visual studio
* Photoshop
* 3DS Max
* Github

In Visual Studio moeten we alle libraries en include directories instellen zodat we OpenGL en andere libraries goed kunnen gebruiken. Verder moeten we Visual Studio koppelen aan GitHub zodat we allemaal veilig kunnen werken zonder elkaars bestanden aan te passen. En daarna alles goed bij elkaar kunnen voegen zodat het een product wordt.

Photoshop gebruiken we om simpele sprites en bitmaps te maken die we kunnen implementeren in de game. Mooiere en moeilijkere sprites en bitmaps halen we van internet.

3DS Max gebruiken we om simpele 3D modellen te maken. De lastigere 3D modellen halen we van internet anders gaat het te veel tijd kosten en kunnen we minder programmeren.

Github gebruiken we voor de source control. We hebben in github één master branch en daaronder heeft iedereen zijn eigen branch om in te werken en veilig op te slaan zonder dat de anderen daar last van hebben.

# Functioneel ontwerp

In het functioneel ontwerp bespreken wij de product backlog, het ontwerp van de schermen, de menustructuur, de gameplay, de world/levels en de karakters

Product backlog

Wij hebben met het team overleg gehad met onze klant(Christiaan Albers), Te vinden in de notulen. In dit overleg hebben wij de product backlog besproken.

|  |  |  |  |
| --- | --- | --- | --- |
| **Prioriteit** | **Als een?** | **Ik wil / wil ik** | **Zodat** |
| 3 | Klant | Gamemenu’s | Voor navigatie en starten van het spel |
| 3 | Klant | Lobby voor multiplayer | Voor het joinen van spellen |
| 2 | Klant | Safe area | Zodat de speler rust krijgt |
| 1 | Klant | Werelden | Voor replayabilty |
| 1 | Klant | Player interactions | Het spel gespeeld kan worden |
| 2 | Klant | Zombies | Gameplay, vijanden |
| 4 | Klant | Score(highscores) en levens | Gameplay, voortgang |
| 2 | Klant | Wapen systeem | Gameplay, Doel om verder te komen |
| 1 | Klant | Multiplayer | Voor het spelen met andere mensen. |

## Ontwerpen van de schermen

Het **main menu** scherm ziet er als volgt uit:

De play knop begint het afspelen van de game.  
Wanneer je op de settings knop drukt opent het settings menu.  
En wanneer je op de exit knop drukt, sluit de game af.

Het **settings menu** ziet er als volgt uit:

In het settings menu heb je verschillende opties voor de game.  
Bijvoorbeeld welke knop wat moet doen, hierdoor kan de user de game aanpassen zoals hij/zij dat wil.  
Grafische opties voor de game. Niet alle computers zijn even goed, hierdoor kan de ene computer de game met betere opties renderen als de andere.  
Geluids opties voor de game. Sommige gebruikers vinden geluiden harder dan andere. Het is dan fijn om de geluidsinstelling aan te kunnen passen.

De back knop zorgt ervoor dat je terug gaat naar het menu waar je vandaan kwam.

In de **game** ziet het menu er als volgt uit:

Het menu ziet er grotendeels hetzelfde uit als het main menu maar in plaats van de play knop is er een resume knop. Deze zorgt ervoor dat het menu weer sluit en je verder kan met de game.  
De settings knop opent het settings menu.  
De exit knop laat je de game verlaten en naar het main menu te gaan. Hier kun je dan beslissen om de game af te sluiten of wat anders te gaan doen.

## Menustructuur

**Main Menu:**

Dit is het start scherm wanneer je de game opent. Hier kun je de game **starten**, **afsluiten** en **opties** voor de game instellen.

In het **optiemenu** kun je het volgende instellen:

* **Volume:** Geluiden harder of zachter instellen.
* **Keys:** Opties voor welke knop op het toetsenbord wat moet doen.
* **Grafische opties:** Grafische opties voor de game aanpassen zodat de computer de game het beste kan afspelen.

**In Game Menu:**

Dit is het menu voor wanneer je de game aan het spelen bent. Hier kun je de game **hervatten**, **opties** instellen en **terug naar main menu**.

Het optie menu is overal hetzelfde. In de game heb je alleen als achtergrond de game.

## Gameplay

Het doel in onze game is om zo lang mogelijk levend te blijven.

De zombies in de game blijven constant naar je toe komen behalve wanneer je in de save-zone bent. In de save-zone ben je veilig voor de zombies, hier kun je ook je wapens upgraden en ammo kopen.

De zombies worden steeds steker naarmate je langer levend blijft. Hierdoor moet je na een bepaalde tijd je wapens wel upgraden omdat het anders te moeilijk wordt om ze steeds te kunnen verslaan.

Je kunt wapens upgraden met items die over de hele map zijn verspreidt.   
Wanneer je voor de eerste keer deze items hebt gevonden, kun je in de save-zone met deze items een beter wapen craften.   
Nadat je het wapen hebt gecraft, spawnen de items weer opnieuw over de hele map. Nu kun je met de items je wapens upgraden. Bij het upgraden wordt het wapen sterker waardoor het makkelijker wordt om de zombies te verslaan.

Wanneer je een zombie hebt verslagen krijg je hier punten van. Het puntenaantal verschilt per keer en maakt uit waar je de zombie raakt.

Je kunt ook naar een ander gebied gaan. In sommige gebieden is het makkelijker om de zombies te verslaan dan in andere gebieden. Hierdoor kun je een tactisch voordeel hebben.

## World/levels

In de game hebben wij een paar gebieden.

Het bos. Het gebied in het bos is buiten en open. Er zijn overal bomen en soms een klein huisje waar je in zou kunnen schuilen. Er is niet veel licht, en is aardig donker waardoor het een griezelig gevoel geeft.



De safe-zone. De safe-zone is in een klein gebouwtje. Hierin ben je veilig van de zombies omdat deze hier niet kunnen komen. In de safe-zone kun je eventueel items craften of upgraden.   
Ook kun je hier ammo kopen voor je wapens met de punten die je hebt gekregen voor het verslaan van de zombies. In de safe-zone is het goed verlicht en duidelijk wat waar te doen is.

Het casino. Het casino deel is ingericht zoals een normaal casino. Verder zijn er veel lichten zoals dat in het echt ook is.

Het treinstation. In het treinstation zijn verschillende platformen waar treinen kunnen stoppen. Je kunt vanuit dit gebied ook naar de safe-zone komen.

## Karakters

**Spelers:**

Er zijn drie speler karakters, voor elke speler een ander karakter. Je kan je eigen karakter bewegen door middel van de toetsen die daar voor ingesteld(standaard wasd). Het karakter kan springen, en om zombies dood te maken kan hij schieten met het wapen dat hij heeft geselecteerd. Een karakter kan maximaal twee wapens tegelijk hebben, wil hij een ander wapen hebben dan zal hij een van zijn wapens er voor moeten inruilen.

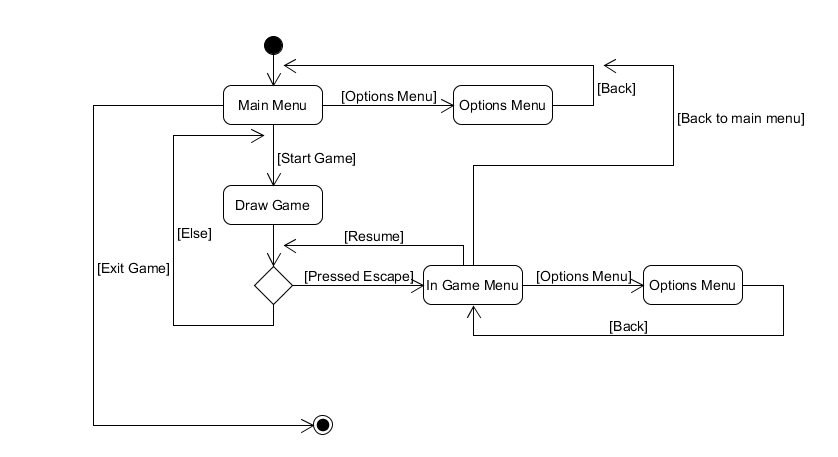
**Zombies:**  
Er zijn twee verschillende zombie karakters. De manier waarop ze lopen en naar mensen toe gaan is hetzelfde maar ze zien er alleen wat anders uit. Als we genoeg tijd hebben kunnen we de verschillende zombies anders laten lopen of sterker maken om wat meer diversiteit toe te voegen.

# Technisch ontwerp

In het technisch ontwerp bespreken wij de verschillende diagrammen. Wij hebben ervoor gekozen om drie activity diagrammen te maken. We hebben twee technische activity diagrammen, één voor de client en een voor de server waarin we laten zien hoe de code loopt en wat er gebeurt. En we hebben een activity diagram voor het doel van het spel de keuzes die er gemaakt kunnen worden en hoe je voortgang maakt.

Verder hebben we twee class diagrammen één voor de classes van de client(de renderer) en één voor de classes van de server(het spel).

## Activity diagram

****

Wanneer je de game opstart, begin je bij het **Main Menu**. Hier heb je 3 keuzes:

* **Start Game.** Start je het spelen van de game.
* **Options.** Hier ga je naar het options menu. Hier kun je verschillende opties voor de game instellen.

Onder andere de geluids instellingen. Deze worden dan opgeslagen en in de game gebruikt. Wanneer je het spel afsluit en daarna weer opstart, zijn de settings de laatste waarde die jij ze gegeven hebt.  
Wanneer je in het settings menu op back drukt, ga je terug naar het **Main Menu**.

* **Exit Game.** Dit sluit de game meteen af.

Tijdens het spelen van de game kun je op escape drukken. Wanneer je dit doet, opent het **In Game Menu**. Hier heb je bijna dezelfde opties als in het **Main Menu**.

* **Resume Game.** Het spel hervatten waar je gebleven was.
* **Options.** Hier ga je naar het options menu. Hier kun je verschillende opties voor de game instellen.

Onder andere de geluids instellingen. Deze worden dan opgeslagen en meteen in de game gebruikt. Wanneer je het spel afsluit of terug gaat naar het main menu, zijn de settings de laatste waardes die jij ze gegeven hebt.  
Wanneer je in dit menu op back drukt, ga je terug naar het **In Game Menu**.

* **Back to main menu.** Hier ga je terug naar het main menu.

## KnipselClass diagram

**Model:**

Model is de basis class die voor de player, enemy en modellen zelf gebruikt wordt.  
In de model class is een positie, rotatie en scale van het model die gebruikt kunnen worden om bijvoorbeeld een transformation matrix te maken zodat het model goed op het scherm komt.

Ook kun je een velocity meegeven. Dit kan gebruikt worden voor de physics van het model.

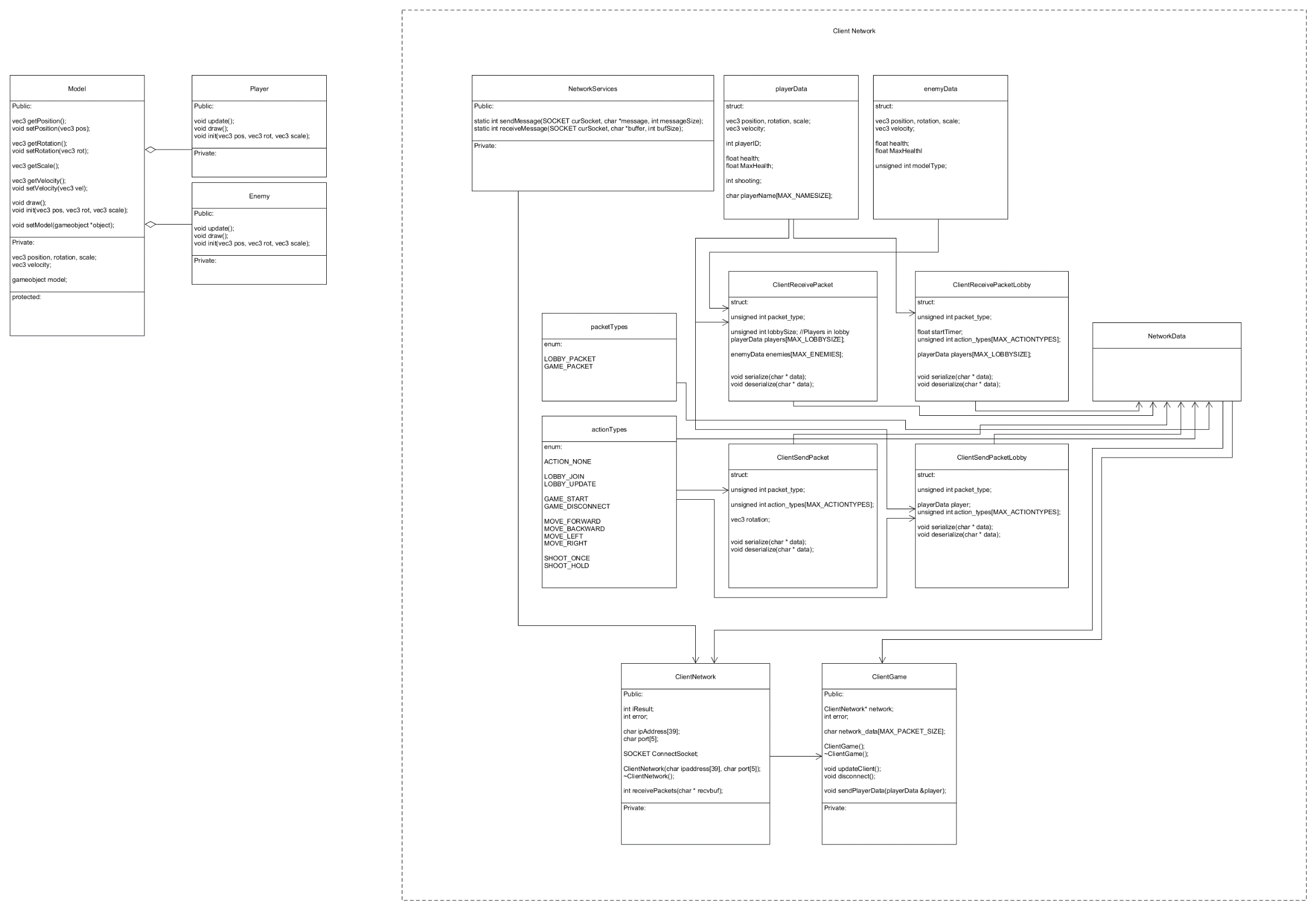
Het gameobject genaamd model is het model dat gebruikt kan worden om het model uiteindelijk te tekenen.

**Player:**

De player class heeft alle informatie van de speler. Hierin staat onder andere alle update code van de speler.

**Enemy:**

De enemy class heeft alle informatie van de enemy. Hier zit onder andere alle AI code en update code van de enemy.

**NetworkServices:**

In networkservices zijn de functies die packets over het netwerk sturen en ontvangen. Deze informatie kan dan in een buffer gestopt worden of uit een buffer gehaald worden.

**playerData:**

playerData is een struct dat alle informatie van de speler bevat die over het netwerk gestuurd zouden kunnen worden.

**enemyData:**

enemyData is een struct dat alle informatie van de enemies bevat die over het netwerk gestuurd zouden kunnen worden.

**packetTypes:**

packetTypes is een enum dat aangeeft wat voor een packet type het is. Packets met een ander type hebben vaak ook een andere packet grootte. Met dit type kun je in de code kijken hoe groot het packet hoort te zijn.

**actionTypes:**

actionTypes is een enum dat aangeeft wat voor een actie heeft plaatsgevonden. Omdat wij alle processing van posities en collisions op de server doen, geven wij elke frame alleen door wat er die frame gebeurt is.

**ClientReceivePacket:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden en van de server afkomt tijdens het spelen van de game.  
Hier staan alle spelers en enemies in die gebruikt worden in de game.

**ClientSendPacket:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden en naar de server gaat tijdens het spelen van de game.  
Hier staan alle actietypes in die deze frame zijn plaatsgevonden en de rotatie van de speler. Dit is namelijk gebonden aan de veranderingen van de muis van de speler, daarom is dit beter om mee te geven.

**ClientReceivePacketLobby:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden en van de server afkomt tijdens dat de spelers in de lobby zitten voordat de game gestart is.  
Hier staan onder andere alle spelers in, hier kan de client dan de player namen uithalen. Ook staat er een startTimer in. Wanneer de game gestart wordt, gebeurt dit met een timer. Deze tijd wordt van de server naar de clients gestuurd.

**ClientSendPacketLobby:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden en naar de server wordt gestuurd tijdens dat de spelers in de lobby zitten voordat de game gestart is.  
Hierin staat onder andere de huidige speler data zodat de client’s spelersnaam naar de server wordt verzonden en naar alle andere spelers gaat.

**NetworkData:**

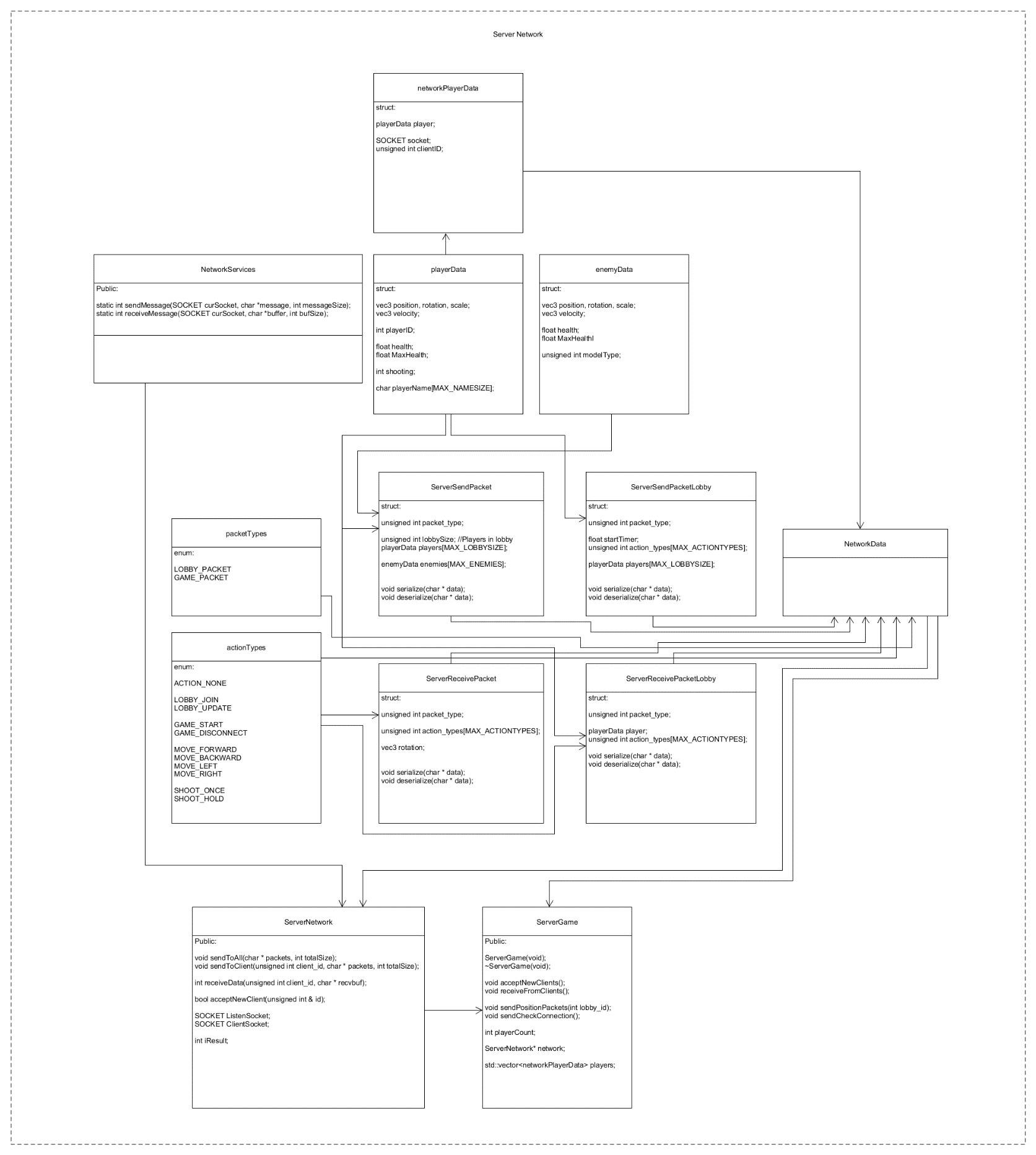
NetworkData is een header class dat alle nodige bestanden bij elkaar houdt om verder te gebruiken in de code.

**ClientNetwork:**

ClientNetwork maakt een connectie met de server over een IP-adres en port en slaat deze connectie op om later hier data naartoe te kunnen sturen.  
Ook zit er in deze class een functie die alle packets ontvangt en deze in een buffer stopt om uit te kunnen lezen.

**ClientGame:**

ClientGame is een class die vooral packets gaat versturen en alle packets die ontvangen zijn uitleest en behandelt.  
Hier kan in onder andere mijn playerData versturen en disconnecten van de server.

**networkPlayerData:**

networkPlayerData is een struct dat alle informatie van de client bijhoudt. Hierin zit onder andere zijn playerData struct, Socket en clientID. Met deze informatie kan de server packetjes naar de goede client sturen.

**NetworkServices:**

In networkservices zijn de functies die packets over het netwerk sturen en ontvangen. Deze informatie kan dan in een buffer gestopt worden of uit een buffer gehaald worden.

**playerData:**

playerData is een struct dat alle informatie van de speler bevat die over het netwerk gestuurd zouden kunnen worden.

**enemyData:**

enemyData is een struct dat alle informatie van de enemies bevat die over het netwerk gestuurd zouden kunnen worden.

**packetTypes:**

packetTypes is een enum dat aangeeft wat voor een packet type het is. Packets met een ander type hebben vaak ook een andere packet grootte. Met dit type kun je in de code kijken hoe groot het packet hoort te zijn.

**actionTypes:**

actionTypes is een enum dat aangeeft wat voor een actie heeft plaatsgevonden. Omdat wij alle processing van posities en collisions op de server doen, geven wij elke frame alleen door wat er die frame gebeurt is.

**ServerSendPacket:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden. Dit packet wordt naar de client gestuurd en heeft alle informatie die de client nodig heeft tijdens het spelen van de game.

**ServerReceivePacket:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden. Dit packet wordt van de client op de server ontvangen. Hierin staat alle informatie die de server moet gaan verwerken.

**ServerSendPacketLobby:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden. Dit packet wordt naar de client gestuurd en heeft alle informatie die de client nodig heeft wanneer hij nog in de lobby zit voor het begin van de game.

**ServerReceivePacket:**

Het packet dat uiteindelijk over het netwerk gestuurd gaat worden. Dit packet wordt van de client op de server ontvangen. Hierin staat alle informatie die nodig is voordat het spel gestart is.

**NetworkData:**

NetworkData is een header class dat alle nodige bestanden bij elkaar houdt om verder te gebruiken in de code.

**ServerNetwork:**

ServerNetwork is een class die packetjes kan versturen of ontvangen van de clients.  
Ook kan deze class een nieuwe client accepteren als er iemand is die de game wil komen spelen.

**ServerGame:**

ServerGame is een class die uiteindelijk het meeste in de server gebruikt gaat worden voor de netwerken. Hierin wordt bijgehouden welke clients er allemaal zijn, hoeveel er zijn en makkelijke functies die de informatie van de clients kan verwerken en weer versturen.

## Testplan

Tussentijds wordt er natuurlijk ook getest. We beschrijven hieronder lijst met functionaliteiten en gameplay elementen die getest moeten worden

|  |  |
| --- | --- |
| **Functionaliteit** | De speler kan bewegen |
| De speler kan kogels afvuren |
| De client kan een connectie maken met de server |
| De client kan pakketjes ontvangen en versturen naar de server |
| De server kan pakkertjes ontvangen en versturen naar de clients |
| De enemy’s spawnen op de goede plaatsen |
| De speler kan de pick-ups oppakken |
| De speler heeft collision met het level en de objecten/spelers en enemy’s daarin |
| De enemy’s hebben collision met het level en de objecten/spelers en enemy’s daarin |
| De speler kan teleporteren naar de ander levels |
| De speler ontvangt punten als hij een enemy’s verslaat |
| De speler kan wapens kopen van de muur |
|  |
| **Gameplay** | Bestuurt de speler goed |
| Hebben de spelers genoeg health |
| Hebben de enemy’s genoeg health |
| Schiet de speler goed |
| Zijn de pick-ups goed te vinden |
| Zijn de wapens goed te vinden |
| Worden er genoeg punten uitgedeeld |
| Bewegen de zombies snel genoeg |
| Is er genoeg licht aanwezig (is het te donker?) |

## Client – Server relatie

Wij gebruiken de client puur en alleen voor het tekenen van de objecten in de game en het versturen van de gebruikers acties.  
Op de server krijgen we de informatie van de gebruikers acties en kunnen deze dan weer gebruiken om de speler in een richting te bewegen, te laten springen en collisions te hebben met de wereld. Ofwel alle berekeningen gebeuren op de server.

### Voordelen

Een voordeel om de relatie zo te hebben is dat de client alle rekenkracht in het tekenen van de game kan stoppen en dat de server die ergens anders draait, zich alleen bezig hoeft te houden met het berekenen van de nieuwe posities van alle objecten.  
Wel is er geen mogelijkheid dat de speler kan cheaten door opeens te teleporten naar een hele andere plek omdat de posities allemaal door de server berekend worden.

### Nadelen

Om de relatie op deze manier te maken heeft een groot nadeel. Wanneer de verbinding tussen de client en server niet zo goed is, komen misschien niet alle playeracties binnen op de server waardoor de berekeningen die de server doet niet altijd goed zijn. Ook is het mogelijk dat de client niet genoeg informatie van de server krijgt over alle objecten. Hierdoor kunnen allemaal vreemde acties gebeuren.

### Conclusie

Het is een goede manier om de verbinding te doen wanneer je er zeker van kan zijn dat de verbinding goed is en zo blijft. Dit is bijvoorbeeld mogelijk door de game op een lokaal netwerk te draaien.

# Sprints

## Sprint backlog

|  |  |  |
| --- | --- | --- |
| **Item** | **Prioriteit** | **Taakverdeling** |
| Werelden | 1 | Stefan |
| Player Controlling | 1 | Olivier |
| Player interactions (Collisions) | 1 | Olivier |
| Multiplayer (Client) | 1 | Wouter |
| Multiplayer (Server) | 1 | Wouter |
| Safe area | 2 | Wouter |
| Player interactions (With world) | 2 | Stefan |
| Zombies (Collisions) | 2 | Olivier |
| Zombies (AI) | 2 | Wouter |
| Zombies (Spawning system) | 2 | Olivier |
| Wapen systeem (Weapons) | 2 | Stefan |
| Wapen systeem (Bullets) | 2 | Stefan |
| Animaties (Create) | 3 | Stefan |
| Animaties (Use) | 3 | Wouter |
| Sound | 3 | Wouter |
| Game menu’s (Main Menu) | 3 | Olivier |
| Game menu’s (Lobby Menu) | 3 | Olivier |
| Game menu’s (In Game Menu) | 3 | Stefan |
| Multiplayer (Lobby System) | 4 | Wouter |
| Score (highscores) en levens | 4 | Stefan |

## Sprint 1:

## Sprint backlog

|  |  |  |
| --- | --- | --- |
| Order | Prioriteit | Item |
| 1 | **1** | Multiplayer (Client) |
| 1 | **1** | Multiplayer (Server) |
| 1 | **1** | Werelden (Design world and get models) |
| 2 | **1** | Player interactions (Collisions) |
| 2 | **1** | Player Controlling (Control player with mouse and keyboard) |

### Codevoorbeelden

Wouter:

De volgende code maakt een netwerk connectie aan en probeert te verbinden met de server met het ip adres \_ipAddress en port \_port.  
Wanneer dit gelukt is wordt de verbinding opgeslagen in ConnectSocket. Hier kan de client dan packetjes naar de server sturen.

//Constructor

//Initialise the clientNetwork and connect to server with ipAddress and port.

ClientNetwork::ClientNetwork(char \_ipAddress[39], char \_port[5])

{

// Create WSADATA object.

WSADATA wsaData;

// Intialise socket.

ConnectSocket = INVALID\_SOCKET;

// Holds address info for socket to connect to.

struct addrinfo \*result = NULL,

\*ptr = NULL,

hints;

// Initialize Winsock

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

// Check if it didn't create an error.

if (iResult != 0) {

printf("ERROR: -- WSAStartup failed with error: %d\n", iResult);

errors.push\_back(WSA\_STARTUP\_ERROR);

}

// Set address info

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP; //Use TCP connection!

// Resolve ipaddress and port.

iResult = getaddrinfo(\_ipAddress, \_port, &hints, &result);

// Check if resolving didn't create an error.

if (iResult != 0)

{

printf("ERROR: -- getaddrinfo failed with error: %d\n", iResult);

WSACleanup();

errors.push\_back(GET\_ADDR\_INFO\_ERROR);

}

// Attempt to connect to an address until one succeeds.

for (ptr = result; ptr != NULL; ptr = ptr->ai\_next) {

// Create a SOCKET for connecting to server

ConnectSocket = socket(ptr->ai\_family, ptr->ai\_socktype, ptr->ai\_protocol);

//Check if the socket is not an INVALID\_SOCKET.

if (ConnectSocket == INVALID\_SOCKET) {

printf("ERROR: -- socket failed with error: %ld\n", WSAGetLastError());

WSACleanup();

errors.push\_back(CREATE\_SOCKET\_ERROR);

}

// Connect to the server.

iResult = connect(ConnectSocket, ptr->ai\_addr, (int)ptr->ai\_addrlen);

// Check if connecting to the server didn't create an error.

if (iResult == SOCKET\_ERROR)

{

closesocket(ConnectSocket);

ConnectSocket = INVALID\_SOCKET;

printf("ERROR: -- The server is down or was unable to connect\n");

errors.push\_back(CONNECT\_SOCKET\_ERROR);

}

}

// No longer need address info for server.

freeaddrinfo(result);

// Check if creating the socket didn't fail.

if (ConnectSocket == INVALID\_SOCKET)

{

printf("ERROR: -- Unable to connect to server!\n");

WSACleanup();

errors.push\_back(ALL\_CONNECTING\_SOCKETS\_ERROR);

}

// Set the socket I/O mode: In this case FIONBIO

// enables or disables the blocking mode for the

// socket based on the numerical value of iMode.

// If iMode = 0, blocking is enabled;

// If iMode != 0, non-blocking mode is enabled.

u\_long iMode = 1;

iResult = ioctlsocket(ConnectSocket, FIONBIO, &iMode);

// Check if the socket didnt create an error.

if (iResult == SOCKET\_ERROR)

{

printf("ERROR: -- ioctlsocket failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

errors.push\_back(SET\_NONBLOCKING\_ERROR);

}

// Disable nagle.

char value = 1;

setsockopt(ConnectSocket, IPPROTO\_TCP, TCP\_NODELAY, &value, sizeof(value));

}

Olivier:

Stefan:

### Testresultaten

In deze sprint gaan wij het volgende testen:

|  |  |
| --- | --- |
| **Functionaliteit** | De client kan een connectie maken met de server |
| De client kan pakketjes ontvangen en versturen naar de server |
| De server kan pakkertjes ontvangen en versturen naar de clients |
| De speler kan bewegen |
|  |
| **Gameplay** | Bestuurt de speler goed |

Functionaliteits test:

De client kan succesvol een connectie maken met de server. Hierna kan de client packetjes versturen en ontvangen van en naar de server.  
Op de server wordt de client succesvol ontvangen en opgeslagen. Hierna worden de packetjes van de client goed gehandeled en daarna kan er weer een packetjes terug verstuurd worden.

### Review

### Retrospective

## Sprint 2:

## Sprint backlog

|  |  |  |
| --- | --- | --- |
| Order | Prioriteit | Item |
| 1 | **2** | Zombies (Collisions) |
| 1 | **2** | Zombies (AI) |
| 1 | **2** | Wapen systeem (Weapons) |
| 2 | **2** | Safe area |
| 2 | **2** | Zombies (Spawning system) |
| 2 | **2** | Wapen systeem (Bullets) |
| 3 | **2** | Player interactions (With world) |

### Codevoorbeelden

### Testresultaten

### Review

### Retrospective

## Sprint 3:

## Sprint backlog

|  |  |  |
| --- | --- | --- |
| Order | Prioriteit | Item |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Codevoorbeelden

### Testresultaten

### Review

### Retrospective

## Bijlagen:

### Notulen:

**Datum:** 14-11-2016  
**Gesprek met:** Christiaan Albers  
**Aanwezig:** Wouter Grutter, Stefan van der Ham, Olivier Lindenbergh  
**Gesprekspunten:**

* Source control:

De klant heeft aangegeven dat het handig is om een goede vorm van source control te hebben. Wij hebben voor GitHub gekozen omdat het ons het handigst en makkelijkst te gebruiken leek.

* Informatieanalyse:

Te vinden in de Informatieanalyse.

* Product Backlog besproken.

**Datum:** 17-11-2016  
**Gesprek met:** assessoren(Mr. Jaspers)  
**Aanwezig:** Wouter Grutter, Olivier Lindenbergh  
**Gesprekspunten:**

* Doelgroep zoeken. Spel uitleggen en bespreken na elke sprint
* Volgende week dinsdag (22-11-2016) verslag inleveren bij assessoren. Groeiend verslag met verschillende versies.
* Afspraak gemaakt voor volgend assessoren overleg dinsdag 23-11-2016.

**Datum:** 22-11-2016  
**Gesprek met:** assessoren (Mr. Jaspers)  
**Aanwezig:** Wouter Grutter, Olivier Lindenbergh  
**Gesprekspunten:**

* Layout verbeteren. Arial 12 is normaal geen 36 want is bladvulling.
* Github toevoegen bij ontwikkel omgeving. Word weghalen.
* Tekst tussen kopjes, kleine inleiding waar het volgende kopje over gaat.
* Prioritering bij de product backlog.
* Volgorde van de sprints: Sprint backlog, codevoorbeelden plaatjes, test resultaten, review, retrospective.

**Datum:** 29-11-2016  
**Gesprek met:** assessoren (Mr. Jaspers)  
**Aanwezig:** Wouter Grutter, Olivier Lindenbergh, Stefan van der Ham  
**Gesprekspunten:**

* Layout overleggen met mevrouw Pol.
* Plaatjes bij menu’s.
* Uitgebreider Classen diagram beschrijven.
* Server – Client relatie beschrijven in technisch ontwerp. Wat gebeurt op de client en wat doet de server met voor en nadelen.

**Datum:** 06-12-2016  
**Gesprek met:** assessoren (Mr. Jaspers)  
**Aanwezig:** Wouter Grutter, Olivier Lindenbergh, Stefan van der Ham  
**Gesprekspunten:**

* Plaatjes bij menu’s.
* Uitgebreider Classen diagram beschrijven.
* Server – Client relatie beschrijven in technisch ontwerp. Wat gebeurt op de client en wat doet de server met voor en nadelen.

**Datum:**   
**Gesprek met:**   
**Aanwezig:**   
**Gesprekspunten:**

### Logboek

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Datum | Wouter | Olivier | | Stefan | |
| 28-11-2016 | Begonnen met de client en verder gewerkt aan documentatie. | |  | |  |
| 29-11-2016 | Verder met de client en documentatie. | |  | |  |
| 30-11-2016 | Verder met de client. | |  | |  |
| 01-12-2016 | Verder met de client en begonnen aan de server. | |  | |  |
| 02-12-2016 | Gewerkt aan de client en server, kunnen nu packetjes naar elkaar sturen. | |  | |  |
| 04-12-2016 | Client en server zijn bijna af. | |  | |  |
| 05-12-2016 | Nog een klein beetje gewerkt aan de client en server. | |  | |  |
| 06-12-2016 | Beetje verder gewerkt aan die connectie en wat de client en server naar elkaar sturen. Verder heb ik ook gewerkt aan de documentatie. | |  | |  |
| 07-12-2016 | Begonnen aan de AI van de enemies op de server. De enemy draait nu meteen naar de dichtstbijzijnde speler toe. | |  | |  |
| 08-12-2016 | Verder gegaan aan de AI van de enemy. Ook begonnen aan het ontwerpen van de safe area in de game. | |  | |  |
| 09-12-2016 | AI zo goed als af gemaakt door de enemy naar de dichtstbijzijnde speler ta laten draaien over een bepaalde tijd wanneer een andere speler opeens dichterbij is. Ook verder gewerkt aan het ontwerpen van de safe area.  Ik ben ook nog even bezig geweest met het verbeteren van de client. Deze gebruikt de data die over het netwerk verstuurd wordt nu wat beter en stuurt dan ook weer wat terug. | |  | |  |
| 11-12-2016 | Bezig geweest aan het exporteren van de safe are modellen naar obj’s zodat ik deze kan gaan gebruiken in de game. | |  | |  |
| 12-12-2016 | De safe area models inladen, bezig geweest met het exporteren van de textures naar .bmp bestanden en de meeste nu ingeladen in de game. | |  | |  |
| 13-12-2016 | De safe area is nu volledig ingeladen. Er moet alleen nog getweaked worden met de model settings.  Ook werkt de server en client connectie nu zo dat hij de player posities goed verstuurd en ontvangt. | |  | |  |
| 14-12-2016 | Ik heb vandaag wat veranderingen gemaakt aan de safe area. Sommige models zijn veranderd zodat ze er beter uit zien.  Ook heb ik Olivier geholpen met bijna het laatste van de physics in de game. | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |
|  |  | |  | |  |