Eindverslag Minor

Game development

25-1-2019  
Wouter Heslinga  
[Hesl1500@student.nhl.nl](mailto:Hesl1500@student.nhl.nl)  
Studentnummer: 432148

Opleiding NHL Stenden Hogeschool - HBO Informatica   
Rengerslaan 10,   
8917 DD Leeuwarden

# Voorwoord

Dit eindverslag gaat in op hoe de minor is verlopen. Deze minor maakt deel uit van het 4e jaar van de opleiding Informatica aan de NHL Stenden Hogeschool te Leeuwarden. Er was vrije keuze in het kiezen van een minor opdracht.

Ik heb gekozen om mij bezig te houden met game development, om zo de beste aansluiting te krijgen op mijn interesses en hobby’s.

Gedurende dit semester heb ik mij gericht op het ontwikkelen van 3 games voor het pc-platform.  
Twee van deze games zijn gemaakt met Unity en de andere is gemaakt met het framework “Monogame”.

Dit verslag zal terugblikken op wat ik heb geleerd heb en wat ik bereikt heb tijdens deze minor periode.

# Samenvatting

Dit eindverslag gaat in op wat er binnen deze minor is gedaan en wat er uiteindelijk is behaald.  
Mijn opdracht was het designen en ontwikkelen van een game die in Unity gemaakt wordt, daarnaast zijn er nog 2 games die ontworpen en ontwikkeld zijn tijdens deze minor waarvan 1 met het framework “Monogame” is gemaakt.

De games die ik tijdens deze minor gemaakt heb zijn:

**Cannibal Town**

Cannibal Town is een game die gemaakt is voor de game jam “Ludum Dare 43”.  
Het thema van deze game jam was “Sacrifices must be made”, op basis van dit thema ben ik op het idee gekomen om een kolonie simulator game te maken waarin je jouw eigen inwoners moet op offeren om zo grondstoffen te krijgen waarmee je vervolgens je kolonie kan uitbreiden.  
Cannibal Town is gemaakt met Unity en C#, de feedback op de game was erg positief en de feedback is gebruikt bij het ontwikkelen van de andere games.

**ShootIT**

ShootIT is een game die gemaakt is voor de game jam “Game-off 2018”.  
Het thema van deze game jam was “Hybrid”, op basis van dit thema heb ik een hybrid gemaakt tussen 2D en 3D. Zo heeft de game een overworld(2D) en een aantal dungeons(3D) waarin de speler zich kan bewegen. De speler kan met behulp van wapens vijanden uitschakelen om vervolgens een aantal objecten te verzamelen, wanneer de speler alle objecten heeft verzameld heeft de speler gewonnen, deze objecten kunnen zowel in de 2D als 3D wereld voor komen.  
ShootIT is gemaakt met het framework “Monogame” en geprogrammeerd in C# tijdens het ontwikkelen van deze game heb ik veel nieuwe dingen geleerd zoals het maken van een Particle systeem.

**ProjectRPG:**

ProjectRPG is een game die gemaakt is voor de minor van school in het kader van game development.  
ProjectRPG is een 2D role playing game die de speler uitdaagt om zo sterk mogelijk te worden door het verzamelen van voorwerpen en door het behalen van levels. Om de game interessanter te maken zijn de voorwerpen en vijanden random gegenereerd. Hierdoor krijgt de speler veel meer items tot zijn beschikking en is het voor de speler ook een stuk lastiger om voor alle karakters een goed item te bemachtigen. De game is voorzien van AI omdat de vijanden veel keuzes moeten maken tijdens gevechten. De speler moet zelf keuzes maken voor het aanvallen van vijanden zo kan de speler ervoor kiezen om altijd zwakke vijanden aan te vallen of juist sterke vijanden te gaan verslaan. Hoe sterker de vijand des te meer voorwerpen de speler zal krijgen.

Inhoudsopgave

[Voorwoord 2](#_Toc536557409)

[Samenvatting 3](#_Toc536557410)

[Inleiding 5](#_Toc536557411)

[Gemaakte Games 6](#_Toc536557412)

[Cannibal Town 6](#_Toc536557413)

[Villagers 7](#_Toc536557414)

[Grondstoffen 7](#_Toc536557415)

[Gebouwen 8](#_Toc536557416)

[Jobs 8](#_Toc536557417)

[Ludum Dare 9](#_Toc536557418)

[ShootIT 2D/3D Shooter 11](#_Toc536557419)

[Wereld 14](#_Toc536557420)

[Tiled 14](#_Toc536557421)

[Project RPG 15](#_Toc536557422)

[Items 15](#_Toc536557423)

[Karakters 17](#_Toc536557424)

[Reflectie 19](#_Toc536557425)

# Inleiding

Dit hoofdstuk is ter inleiding van dit eindverslag voor mijn minor. De reden achter mijn opdracht zal uitgelegd worden en de resultaten en ondervingen zullen besproken worden.

De keuze om games te maken is gekomen omdat ik tijdens mijn vorige minor ook een game heb gemaakt met een aantal andere studenten. Dit was me erg goed bevallen en heb dus besloten om deze minor periode ook een game te gaan maken maar dit keer in 2D terwijl ik mij tijdens de vorige minor gefocust had op 3D.  
  
Met deze minor wou ik mij wat meer richten op het design gedeelte van game development, dus hoe bouw ik een game die leuk is voor de speler. Hierbij komen veel aspecten kijken zoals bijvoorbeeld uitdaging en hoe lang de speler het spel zou kunnen spelen zonder dat het verveeld.  
  
Omdat ik mij iets meer wou focussen op het design gedeelte van game development heb ik ervoor gekozen om i.p.v. één game te maken mij te storten op de wereld van game-jams.  
Game-jams zijn evenementen waarbij je alleen of met een groepje in een korte periode een volledig werkende game moet maken op basis van een thema, dus van design tot het programmeren. Hierbij komen veel uitdagingen naar voren zoals het plannen van je tijd tot het designen van een game die leuk, kort en speelbaar is.

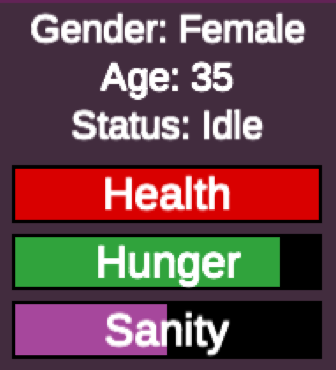
# Gemaakte Games

Tijdens deze minor periode heb ik mij beziggehouden met het maken van 3 games. In de volgende hoofdstukken zal ik in gaan op de verschillende games, hoe ik deze gemaakt heb en wat de speler kan verwachten van de games.

# Cannibal Town

Cannibal Town is een 2D top down kolonie simulator game die ik gemaakt heb voor de game jam “Ludum Dare 43”.  
Het thema van de game jam was “Sacrifices must be made”. Toen het thema eenmaal bekend was,   
was het tijd om te brainstormen over ideeën, na ~2 uren brainstormen lagen er een 5-tal game ideeën klaar.   
Maar omdat ik altijd al eens een kolonie sim game had willen maken was dat het idee die ik verder heb uitgewerkt.  
  
Een kolonie sim is een game waarbij de speler zijn kolonie opdrachten geeft om hiermee zijn kolonie uit te breiden en te beschermen tegen vijanden.  
Meestal werken deze games met het verzamelen van grondstoffen als hout, steen, ijzer etc. maar omdat ik mij aan het thema van de jam wou houden ben ik op het idee gekomen om i.p.v. normale grondstoffen, botten, vlees en voedsel te gaan gebruiken.  
Omdat het thema “Sacrifices must be made” was had ik het idee om de inwoners van je kolonie te op te offeren om zo grondstoffen te krijgen om je kolonie uit te kunnen breiden.   
Hiervoor was er dus geen hout o.i.d. nodig want de huizen zouden van botten en vlees gemaakt kunnen worden die je van je eigen inwoners zou krijgen. Dit was het basisidee en hier ben ik op verder kaan borduren.  
  
Op de volgende pagina’s zal ik verder ingaan op de features van Cannibal Town.  


Figure 0‑1 Cannibal Town met erg veel villagers

VillagersVillagers zijn de basis van de game, zij voeren opdrachten uit en kunnen opgeofferd worden voor grondstoffen. Een villager heeft een aantal stats zoals, hunger, sanity, current job, gender en age.   
Op basis van deze stats kon er een AI geschreven worden die probeert om alle stats in evenwicht te houden zodat de villager niet dood gaat van de honger.  
  
De AI die ik geschreven heb is een Finite State Machine met de volgende states:

* **Idle** – Willekeurig rondrennen, deze state wordt alleen uitgevoerd wanneer er geen opdrachten zijn die uitgevoerd hoeven te worden
* **SearchFood** – Wanneer de AI in deze state komt betekent het dat de villager zijn hunger onder een bepaald niveau is. De villager zal opzoek gaan naar eten. Wanneer er geen eten is zal de villager weer verder gaan met werken.
* **Breed** – Wanneer de villager oud genoeg (18+) is zal de villager op zoek gaan naar een partner die voldoet aan de volgende eisen, 18+, ander geslacht, genoeg plek voor nieuwe villagers, niet in breeding state geweest de afgelopen 30 seconden. Wanneer de villager een partner heeft gevonden zullen er 1 of 2 nieuwe villagers gemaakt worden en zal de villager minimaal 30 seconden niet meer in de breed state gaan.
* **Going Insane** – Een villager kan maar zoveel hebben voordat hij gek wordt. Wanneer er een villager in de buurt van een ander villager wordt op geofferd zal de overgebleven villager gek worden omdat er net iemand voor zijn neus opgeofferd is. De villager zal gek worden en zal proberen om andere villagers te vermoorden. De enige manier om villagers uit deze state te halen is om ze op te offeren.

GrondstoffenZoals als eerder vermeld heeft de game 3 grondstoffen namelijk Botten, Vlees en eten. Naast de grondstoffen is er ook een maximaal aantal villagers die de kolonie mag hebben dit kan worden uitgebreid door het bouwen van huizen.  
  
Grondstoffen zijn nodig voor het maken van gebouwen en het voeden van de villagers.  
Wanneer er een villager op geofferd wordt zal er een lijk op de grond komen van die villager, dit lijk kan op worden gehaald door een andere villager om vervolgens naar de slager te brengen om het lijk te verwerken naar vlees en botten. Het verwerken van lijken geeft 1-3 botten, 1-4 vlees en 3 voedsel.

### Gebouwen

De game heeft 4 gebouwen die de speler kan maken dit zijn:

* **House –** Een house maakt de kolonie groter en zorgt ervoor dat het maximaal aantal villagers met 5 omhooggaat.
* **Butchery –** Een butchery is een werkplek voor een villager om een lijk te verwerken naar grondstoffen. Wanneer het lijk verwerkt is zal de villager de grondstoffen opslaan in een kist die vervolgens naar het warenhuis gebracht kan worden.
* **Warehouse –** Een warehouse kan niet gemaakt worden maar staat klaar wanneer de speler het spel start, dit is gedaan omdat het spel zo gemaakt is dat het niet zonder warehouse kan worden gespeeld. Een warehouse is de plek waar alle grondstoffen gebracht en gehaald worden, wanneer een grondstof niet in het warenhuis ligt, kan deze ook niet gebruikt worden.
* **Altar –** Een altar is het gebouw die de speler moet maken om te winnen. Wanneer deze gebouwd is kan de speler er alsnog voor kiezen om met zijn huidige kolonie verder te spelen.

### Jobs

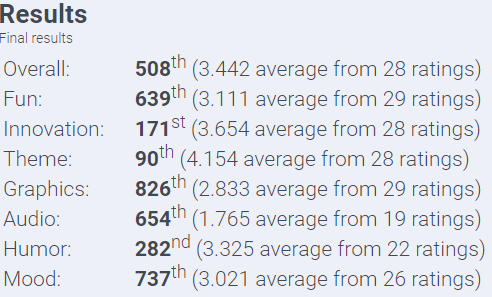
Wanneer de speler wil dat een villager een taak uitvoert zal de speler een Job aan moeten maken voor een villager, dit kan de speler op de volgende manieren doen

* **Klikken op grondstof/Lijk –** Wanneer de speler op een grondstof klikt zal er een nieuwe CollectJob aangemaakt worden die aan de JobQueue toegevoegd zal worden. Een CollectJob is simpelweg het ophalen en opslaan van de grondstof.  
  Wanneer de speler op een lijk klikt zal er een ButcheryCollectJob aangemaakt worden die ervoor zorgt dat een villager het aangeklikte lijk naar de butchery brengt.
* **Gebouw creëren –** Wanneer de speler een nieuw gebouw maakt zullen er jobs gemaakt worden op basis van het aantal grondstoffen die nodig zijn voor het gebouw. Mochten er 20 grondstoffen nodig zijn, worden er 20 jobs gecreëerd om de grondstoffen naar de bouwplaats te brengen.
* **Automatisch –** Een voorbeeld van een job die voor de speler gemaakt wordt is het verwerken van een lijk in de butchery, wanneer er lijk(en) beschikbaar zijn wordt er een nieuwe job aangemaakt.

## Ludum Dare

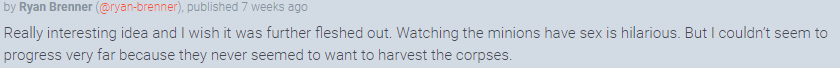
Ludum dare is een internationale game jam waar iedereen aan mee kan doen. Om mee te doen hoef je je alleen in te schrijven.  
Om tot een thema te komen kan iedereen 2 weken van te voren thema’s opsturen die gebruikt kunnen worden voor de komende jam.  
Vervolgens kunnen mensen thema’s liken en disliken zodat er aan het eind van de 2 weken een aantal thema’s overblijven waar nogmaals op gestemd kan worden, wanneer de jam begint word het thema met de meeste stemmen bekent gemaakt en kan iedereen beginnen  
  
Ludum dare duurt 72 uren van zaterdag nacht 3:00 tot maandag nacht 3:00. In deze tijd kun je werken aan je game. Als je voor de eindtijd je game uploadt naar de website kun je mee doen aan de beoordeling periode. De beoordeling periode is een periode van 2 weken waarbij mensen je game kunnen spelen en beoordeling op de volgende punten:

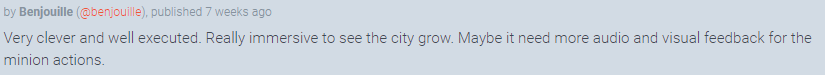
* Overall
* Fun
* Innovation
* Theme
* Graphics
* Audio
* Humor
* Mood

Ook kun je na het spelen van een game commentaar achterlaten voor de maker(s) van de game. Zo kun je bijvoorbeeld laten weten dat er een bug in de game zit of dat het springen niet goed werkt. Op basis van deze beoordelingen krijgen deze games aan het einde van de beoordeling periode een cijfer in alle bovenstaande punten.  
  
De beoordelingen van mijn game zijn hier in beeld gebracht.  
Ratings gaan van 0(niet goed) tot 5(goed)

Tijdens Ludum Dare 43 waren er 7489 deelnemers. Waarvan 2514 games geüpload zijn.  
De plaatsen achter scores zijn dus op basis van de 2514 games die tijdens deze jam zijn gemaakt.  
Mijn game is inmiddels meer dan 100 keer gespeeld door andere mensen, dit is ook leuk voor mij omdat ik hierdoor ook veel feedback op mijn werk heb gekregen.  
Over het algemeen ben ik erg tevreden met deze scores omdat ik een erg uitgebreide game had ontworpen om in een weekend te maken waardoor er veel functionaliteiten eigenlijk niet gemaakt zijn omdat de tijd op was en ik keuzes moest maken welke functionaliteiten ik het belangrijkst vond.  
Voor audio zijn de scores erg laag omdat ik eigenlijk geen tijd meer had om ook nog geluid te implementeren. Hieronder zijn een aantal reacties op mijn game van andere mensen te zien.













# ShootIT 2D/3D Shooter

ShootIT is een combinatie van een 2D en 3D shooter, we hebben er een 2D en 3D gedeelte in gemaakt omdat het thema van deze game jam “Hybrid” was. Dus ik heb een Hybrid tussen 2 en 3D gemaakt.  
ShootIT is gemaakt voor de game jam “Game-off 2018”, omdat ik wat dieper in de code wou duiken heb ik deze game gemaakt met het framework “Monogame”.  
Monogame is een open source C# game framework voor het maken van cross platform games ik heb tijdens deze game jam gekozen voor Monogame omdat deze game jam een maand duurde, en ik ook eens een game wou maken zonder gebruik te maken van een complete game engine die het meeste vuile werk voor mij doet.  
  
ShootIT is een top down shooter met een 3D aspect, het idee van de game was het combineren van een top down survival shooter met wolfenstein3D. Het spel zou zich vooral afspelen in de overworld(2D), daarnaast zouden er een aantal dungeons in de game zitten waar de speler in kon gaan om zo de transitie naar 3D te maken.  
In de dungeons was het doel om alle vijanden dood te maken en vervolgens een voorwerp op te pakken om vervolgens weer naar de oppervlakte te gaan om verder te gaan met het 2D gedeelte van de game.  
  
Monogame was even heel wat anders als Unity waar ik veel ervaring mee heb, i.p.v. het beginnen met gameplay programmeren was het eerst een goed idee een goede basis voor mijzelf te maken zodat ik de meeste basis functionaliteiten beschikbaar zou hebben denk hierbij aan:

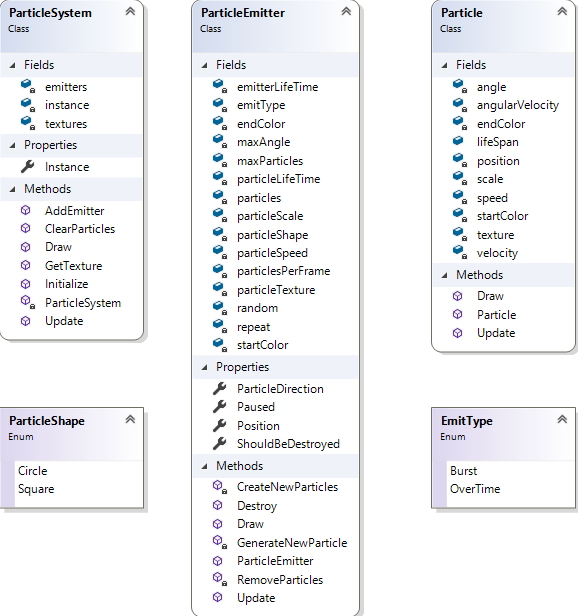
* **ScreenManager –** Een screen manager zorgt ervoor dat ik meerdere schermen in mijn game kan maken, zoals een game screen en een main-menu screen, de screen manager handelt het wisselen van schermen af.
* **InputManager –** Om een game te spelen heeft de game input van de speler(s) nodig, hiervoor is een inputManager geschreven die de input van de speler afhandelt van het scrollen van de muis tot het instappen in een voertuig.
* **3D raycast renderer –** Om de 3D wereld te laten werken met sprites heb ik een renderer gemaakt die met behulp van raycasting de sprites op de goeie plaats renderd zodat het een 3D wereld lijkt.
* **Sound Manager –** Om de game wat “levender” te maken heb ik tijdens deze game ook een complete sound Manager geschreven die alle geluiden kan afspelen uit de game van het schieten van een geweer tot het geluid van de auto.
* **Sprites –** Om iets op het scherm te tekenen makkelijker te maken heb ik eens basis sprites class gemaakt die gelijk getekend kan worden zonder dat je veel parameters hoef in te vullen of de state van de sprites speciaal moet bijhouden.
* **Levels –** Om levels in de game te maken was het ook van belang om een level editor te hebben zodat dit niet tile voor tile in de code verwerkt hoefde te worden. Hiervoor is “Tiled” gebruikt.  
  Tiled is een tile editor waar je je eigen Sprite sheets en sprites kan inladen om hiermee vervolgens je eigen levels te maken.  
  Om een Tiled level om te zetten naar een level die bruikbaar is in Monogame was veel onderzoek en code nodig.
* **Entities –** Elke object waar de speler iets mee kan doen is binnen deze game een “Entity”.  
  Hiervoor heb ik een systeem gemaakt die ervoor zorgt dat alle entities in de huidige level geüpdate en gerenderd worden zodat ik niet voor elk object de render en updatemethodes hoef aan te roepen.

Naast de functionaliteiten die nodig waren voor een goede basis voor een game, zijn er ook een aantal extra functionaliteiten toegevoegd zoals:

* Particle systeem – Om wat meer leven in de game te brengen heb ik een particle systeem gemaakt die verschillende soorten particles kan maken zoals vuur, rook en bloedspetters. Deze details geven de speler het idee dat alles realistischer is.
* Animaties – Om animaties in de game te laten werken, zijn de sprites voor de spelers en vijanden in een sprite sheet gezet zodat de sprites elkaar kunnen afwisselen om zo de entities wat leven te geven.
* Pathfinding – Om vijanden van punt A naar punt B te laten lopen was er path finding nodig,  
  Om dit te realiseren moest ik onderzoek doen naar de beste oplossing voor deze game, omdat de levels opgedeeld zijn in een grid van 500x500 tiles, was A\* een goede oplossing voor dit probleem.

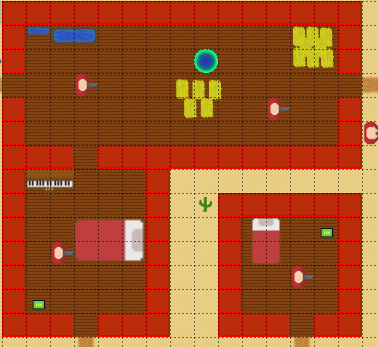
Om een inzicht te geven wat er op de achtergrond allemaal gebeurt met particles is hier een overzicht van de classes, methodes en variabelen die nodig zijn om simpele particles zoals de uitlaatgassen van een auto kunnen renderen.

Een particle is een object zonder bijv. collisions of Pathfinding, het is een dom object wat alleen weet welke kant hij op moet bewegen en welke vorm en kleur hij heeft. Particles zijn eigenlijk sprites die ervoor zorgen dat een game er levender uitziet bijvoorbeeld bij de uitlaat van een auto. Als er niks visueels aanwezig is kan de speler niet goed zien of de auto aan of uit staat, maar zodra je particles voor uitlaatgassen gaat implementeren krijgt de speler gelijk het gevoel dat de auto iets aan het doen is omdat er iets nieuws op het scherm te zien is.



## Wereld

Voor de wereld is gekozen voor een 2D tilemap, een tilemap is niks meer dan een array van tegels die op het scherm in de goede volgorde getekend kan worden. Voor deze game wouden we een aantal dingen die de wereld moest kunnen bevatten zo was er pathfinding nodig en moesten er ingangen komen naar het 3D gedeelte van de game.  
Om dit te realiseren is er gekozen om met behulp van Tiled een wereld te maken en deze te importeren naar Monogame, Om dit mogelijk te maken was er veel reken werk nodig om alle tiles op de goede plek te krijgen omdat er vanuit Tiled niet goed rekening werd gehouden met rotatie en schaal van een tile.

Tiled  
Tiled is een world editor gebaseerd op XML.  
Wanneer je een wereld in Tiled maakt krijg je een XML of JSON file die je vervolgens in je game kan verwerken om zo het proces van het maken van een wereld te versimpelen.  
  
Rechts is een klein stuk van de wereld te zien, hierin kun je goed zien dat de wereld is opgedeeld in stukken van 32x32 pixels.  
De objecten(Bedden, auto’s, etc.) worden ook in Tiled geplaatst alleen worden deze objecten niet 32x32 maar hangt het van de grootte van het object af. Ook wil je dat de objecten in de game interacteer baar zijn zodat je ook daadwerkelijk de auto kan besturen.  
  
Om de objecten iets anders te maken dan een simpele tile hebben we alle objecten een unieke naam gegeven die we in de code kunnen opvangen en op basis van de JSON   
kunnen zien wat het is, hoe groot het moet zijn   
en waar dit object zich moet bevinden.

Om ervoor te zorgen dat spelers en vijanden niet door muren en gebouwen heen kunnen lopen was er collision nodig, om dit te realiseren zijn er in Tiled een aantal tiles gedefinieerd waar de speler niet door heen kan. Hiermee konden we ook een JSON-file generen die we in de code konden gebruiken om te kijken of een specifieke tile collisions heeft of niet.

# Project RPG

Het hoofdproject tijdens mijn minor was het maken van een 2D RPG. Een RPG is een Role Playing Game,  
dit houdt in dat het een spel is waarbij de speler door het verslaan van monsters steeds sterker wordt door het behalen van levels en nieuwe items die de speler kan gebruiken bij het verslaan van steeds sterkere vijanden.  
Voor deze game wou ik niet een simpele RPG maken maar wou ik wat extra’s toevoegen om het spel interessanter te maken. Dit heb ik gedaan doormiddel van het maken van dynamische systemen voor items, vijanden en classes die de speler kan spelen.  
Met dynamische systemen bedoel ik dat het heel erg makkelijk is voor de ontwikkelaar om nieuwe content toe te voegen, zo kan ik met een druk op de knop nieuwe vijanden, items en skills toevoegen aan de game.

Om alle nieuwe content en skills een plek te geven in de game worden alle userinterfaces vanuit de code gegenereerd, dus mocht een class 5 nieuwe skills krijgen zal de code dit omzetten naar een nieuwe userinterface die gebruikersvriendelijk is.

Items  
Items zijn erg belangrijk om een goede RPG-game te ontwikkelen. Om in mijn game de items nuttig en sterk te maken heb ik een systeem ontwikkeld die op basis van een aantal criteria items kan genereren.  
Spelers beschikken over 8 verschillende stats die hun karakters sterker maken, de 8 stats zijn als volgt:

* Strength – Increases melee damage dealt.
* Dexterity – Increases ranged damage dealt.
* Intelligience – Increases magic damage dealt.
* Stamina – Increase maximum health.
* CritChance – Gives the character a #% chance to deal double damage.
* Haste – Determines the speed of the character
* Damage – Increases all damage dealt.
* Armor – Decreases all damage taken.

Om het generen van items wat restricties te geven heb ik de 8 stats verdeeld in 2 groepen namelijk, Suffixes en Prefixes. Dit zorgt ervoor dat het systeem niet 4 sterke stats op 1 item kan zetten en zo kan ik als developer de items ook beter balanceren. De stats zijn verdeeld in de volgende groepen.

Suffix en prefix verdeling van stats.

|  |  |
| --- | --- |
| Prefixes | Suffixes |
| Strength | CritChance |
| Dexterity | Haste |
| Intelligience | Damage |
| Stamina | Armor |

Om het balanceren van items nog beter te maken hebben items nog een aantal andere stats die niet zichtbaar zijn voor de speler zoals “ItemLevel”.  
Het ItemLevel van een item bepaald hoeveel van een bepaalde stat op een item kan zitten dus een laag itemlevel item zou bijvoorbeeld 1-3 strength kunnen hebben terwijl een hoog itemlevel 10-13 strength zou kunnen hebben. Hieronder is een tabel te zien van de itemLevels en stat ranges die items kunnen hebben.

|  |  |  |
| --- | --- | --- |
| ItemLevel | Minimum stat | Maximum Stat |
| 1 | 1 | 3 |
| 2 | 4 | 6 |
| 3 | 7 | 9 |
| 4 | 10 | 13 |
| 5 | 14 | 15 |

Item level – stat ranges

De game bevat 6 verschillende basis items die de speler kan vinden, zodra de speler een item vindt wordt er op basis van de regels een item gegenereerd.  
De speler kan vervolgens dit item gebruiken door hem aan een van zijn karakters te geven of te verkopen bij een winkel. De prijs van een item is als volgt (random (5,10) + (5 \* aantal stats)), dit geeft de speler het gevoel dat items met meer stats veel zeldzamer zijn omdat ze ook voor meer verkopen in de winkel.  
  
De speler heeft een inventory van 20 slots, hierin kan de speler zijn items opslaan, ook heeft elk karakter 6 slots om items te gebruiken. Zodra de speler een item gebruikt op een karakter zullen de stats van het item toegevoegd worden aan de stats van het karakter. Hieronder is een overzicht van de inventory te zien. 

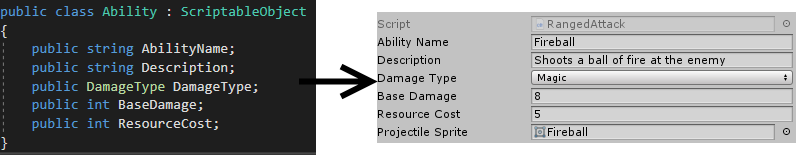
## Karakters

De speler beschikt in het begin van de game over 3 karakters die in zijn party zitten.  
Deze 3 karakters zijn ook gelijk alle classes die momenteel in de game zitten namelijk:

* **Warrior** – Een class die alles met melee wapens doet
  + Belangrijke stats
    - Strength
    - Stamina
  + Skills
    - Melee Hit – een normale aanval met zijn wapen
    - Bash – een aanval die meer schade doet dan een normale aanval maar wel energy kost
* **Mage** – Een class die zich focust op het casten van spells
  + Belangrijke stats
    - Intelligience
    - Stamina
  + Skills
    - Melee hit – een normale aanval met zijn wapen
    - FireBall – Een spell die een vuurbalt maakt om de vijand schade te doen
* **Ranger** – Een class die zich focust op het schieten van bogen en kruisbogen.
  + Belangrijke stats
    - Dexterity
    - Stamina
  + Skills
    - Melee hit – een normale aanval met zijn wapen
    - Shoots – Het schieten van een boog of kruisboog

Elk karakter heeft zijn eigen stats en op basis van de class van een karakter kan de speler de juiste items aan een karakter geven, zo zal een item met strength niet veel doen voor een mage maar zou het voor een warrior wel goed zijn.

Elke class heeft zijn eigen skills zo heeft de mage een vuurbal terwijl de warrior zijn wapens al het vuile werk laat doen. Om ervoor te zorgen dat elke skill nuttig voelt moet de schade die een skill doet wel goed gebalanceerd zijn. Om hiervoor te zorgen heb ik het zo gemaakt dat alle attributen van een skills makkelijk aangepast kunnen worden. Zo heb ik in Unity gebruikt gemaakt van zo genaamde “Scriptable Objects” dit zijn als het ware data objecten die binnen Unity aangepast kunnen worden.



Vervolgens kan de game deze data gebruiken om de classes te voorzien van skills, zo word de user interface in de combat scene aangepast aan het aantal skills die de class heeft.

# Reflectie

Ik heb tijdens deze minor periode veel geleerd, vooral tijdens het maken van de game ShootIT, dit komt omdat ik voor deze game het framework “Monogame” heb gebruikt. Bij dit framework moest ik erg veel zelf maken b.v. het implementeren van pathfinding tot het renderen van sprites. Dit was een leuke uitdaging en heeft mij ook meer inzicht gegeven in hoe een game engine nou eigenlijk werkt en wat het allemaal op de achtergrond doet.

De game jam games waren erg leuk om te maken omdat het een vrij korte periode is waarin je veel moet doen van het ontwerpen van een leuke game die in ~5 minuten te spelen is tot het schrijven van alle code. Dit heeft mij ook een goed beeld gegeven hoe moeilijk het eigenlijk is om een goede game te maken in zo’n korte periode.

Mocht ik ooit nog eens een game gaan maken zal ik van tevoren beter mijn ontwerp maken zodat ik tijdens de game niet veel tijd in het ontwerpen van specifieke features hoef te stoppen.  
Hierdoor zal de workflow ook beter lopen omdat je dan in één stuk door kan ontwikkelen zonder eerst de nieuwe features te hoeven ontwerpen.