Opdracht 3 - Documentatie

Game Development, Kernvak 3, jaar 2

Jeffrey Saydam

Design, Fun en Geloofwaardigheid

In deze game “A Blood Plague Family”, moet de speler de Mother verslaan om te kunnen ontsnappen. De Mother kan pas aangevallen worden, wanneer al haar kinderen dood zijn. De Mother zal rondlopen en energy verzamelen bij de EnergySources. Ook kan zij haar kinderen healen als ze daar om vragen. Wanneer ze genoeg energie heeft kan ze bij de Womb een nieuw kind baren. De speler verliest wanneer hij zelf dood gaat.

De fun en geloofwaardigheid zit hem er vooral in de audio en gameplay. De audio is erg overdreven designed waardoor de horror setting comisch wordt. Ook wakkerd het het sadisme aan bij de speler. Daarnaast kan de speler dood door de muur gedrukt worden als hij tussen een bewegende muur staat en ook zo verliest hij. De design van de AI’s helpen bij de geloofwaardigheid, door gebruik te maken van humanisatie, zo hebben de AI’s ogen en een mond en kijken ze naar hun doel, en dus ook mogelijk naar de speler.

Voor realisme en mysterie zijn er bewust geen controls en objectives aangegeven.

2 AI’s met teminste 3 verschilende behaviours/states.

Beiden AI’s hebben vele verschillende behavious/states. De nuances zijn te lezen in opdracht 2. Zo heeft de MotherAI 6 states: Idle, Moving, Giving Birth, Healing Child, Energizing Womb en Energizing Pocket. En de ChildAI 6 states: Idle, Healing, Crying, Combining Powers, Attacking Player en Searching.

Interactie tussen Speler en AI’s

De interactie tussen de speler en de AI’s vertaald zich door combat. Zo heeft de speler een speer waarmee hij de ChildAI’s kan aanvallen, waarna de MotherAI. De ChildAI’s vallen ook terug aan met als standaard een melee attack en met een speciale ability, een ranged attack. Als de ChildAI’s klaar zijn om te vechten zullen ze de speler zelf opzoeken, dit word ook gecommuniceerd d.m.v. audio.

Interactie tussen AI en AI

De interacties tussen AI’s gaat via ChildAI en ChildAI of ChildAI en MotherAI.
ChildAI en ChildAI: Als beide ChildAI’s de speler gevonden hebben. Vragen ze zichzelf eerst af of ze een combinatie attack kunnen doen, als ze dat kunnen gaan ze kijken of er toevallig een andere ChildAI in de buurt is die dat ook kan. Zoja dan combineren ze hun krachten en kunnen ze een ranged attack doen.

ChildAI en MotherAI: Wanner de ChildAI merkt dat hij bijna dood gaat, probeert hij te vluchten en probeert hij de MotherAI te zoeken. Als hij haar niet kan vinden gaat hij huilen en probeert hij haar nogmaals te zoeken. Als hij haar gevonden heeft, gaat hij haar vragen om hem te healen. Dit is ook te merken via audio in de game. Als de moeder in staat is om hem te healen zal ze zit doen. Beiden AI's staan dan stil en gaan verder met hun taken wanneer ze klaar zijn.

Interactie tussen AI en wereld

De AI's zijn instaat om te wachten voor bewegende muren.
De MotherAI zal eerst haar energie portemonnee vullen, dit doet ze door naar één van de EnergySources te gaan. Deze staan statisch verspreid over de wereld. De MotherAI is in staat om de meest potentiële EnergySource te kiezen. Wanneer haar portemonnee vol is zal ze de energie lozen bij de Womb, deze staat helemaal ergens anders vergeleken met de EnergySources. Wanneer de Womb genoeg energie heeft om een baby te baren zal de moeder in weeën raken en deze uitpersen.

Zelflerende AI

Dit is geïmplementeerd in de ChildAI. Wanneer een ChildAI de speler gevonden hebben, word de spelen in hun geheugen opgeslagen. Daardoor zijn ze in staat de speler te volgen. Wanneer de ChildAI gaat vluchten om te healen zal hij de speler weer vergeten.

Design patterns

Creational patterns: Het AI systeem is geprogrammeerd d.m.v. een Abstract Factory Pattern. De ChildAI en MotherAI zijn een instantie van AIBasics. AIBasics bevat het: Pathfinding Systeem, Health Systeem en overige functies zoals de Die state en UI Visuals. Dit was de beste optie, omdat beide AI's vele overeenkomsten hadden.
Structural patterns: Het wereld systemen zoals de EnergySource en Womb, maken gebruik van een Flyweight Design Pattern. Al deze objecten staan los van elkaar en hebben geen invloed op elkaar. Elk hebben ze hun eigen taak. Het enigste wat hun met elkaar verbind is het AI systeem.
Behavioral patterns: De behaviour van de AI's is gedesigned met een State Design Pattern. De AI's hebben vele verschillende states die gedetermineerd worden door de BehaviourTree.

Implementatie BehaviourTree

De BehaviourTree heb ik eerst gedesigned op papier om de core van de AI's vast te stellen, daarna heb ik deze uitgewerkt in meer details, deze zijn te zien in opdracht 2.

De BehaviourTree fungeert in mijn code als een keuzemaker. D.m.v. het vooraf designen van de states kon ik makkelijk de keuzes vertalen met gebruik van if-statements. Dit systeem maakt ook gebruik van een int variable die bijhoud in welke state de AI zich bevind. Hiermee kunnen continue functies aangeroepen worden per state in de Update functie.

Ik ben er later achter gekomen dat mijn BehaviourTree een Final State Machine bleek te zijn. Na goede uitleg en hulp van de docent, heb ik mijn Final Stae Machine kunnen omzetten tot een BehaviourTree. Wat veranderd is en ik dus geleerd heb is dat een BehaviourTree flexiebeler is, omdat he gebruik maakt van Nodes. Deze manier van AI's schrijven is veel handiger en deze zal ik zeker vaker gebruiken in de toekomst.

Implementatie Pathfinding

Het Pathfinding system heeft verschillende componenten om correct en flexiebel te kunnen werken. Het syteem bestaat uit een Grid, Nodes en een Pathfinding Component voor de AI's.
Nodes: Elke node houd bij zijn positie in de grid en in de wereld, zijn parent node, of hij een muur is en de node cost om aan te geven hoe efficiënt hij is voor het pad.
Grid: De grid is een zelfstandig object die alle nodes in een array bijhoud. Ook geeft het grid visueel weer in de engine hoe de nodes eruitzien. Met deze flexiebele grid kun je de node sizes, grid sizes en ruimte tussen nodes aanpassen.
Pathfinding Component: Elke AI heeft zijn eigen Pathfinding Component, waarmee hij d.m.v. het grid en A* calculatie het beste pad kan finden.

Door bewegende muren in het spel was A* Pathfinding de meest efficiënte optie. Ter extra efficiëntie word A* Pathfinding alleen gebruikt wanneer de AI beweegd, deze word dan geüpdated in de Update functie. Een extra is dat de AI's wachten wanneer hun pad verbroken word door de bewegende muren. Zo lopen ze niet dom tegen de muren aan of heen.

Bronnen

Programming: All het programmeren is door mij Jeffrey Saydam gedaan, met gebruik van Unity en Visual Studio 2015. Research met gebruik van Youtube en Unity3D Documentatie.
3D art: All het 3D is door mij Jeffrey Saydam gemaakt. Met gebruik van Maya, Substance Painter en Substance Designer.
2D art: Het Concept art voor de 3D meshes is door mij Jeffrey Saydam gemaakt, met gebruik van Photoshop. Overige plaatjes zijn van Google Images.
Audio: Audio resources zijn van youtube gehaald en door mij geëdit met behulp van Adobe Audition.
https://www.youtube.com/watch?v=paRR1vXGgck
https://www.youtube.com/watch?v=NjkNNDuAb9A
https://www.youtube.com/watch?v=0BnH0AVdgv0