Artificial Intelligence

Scriptie | Dennis Haverhals

Student nummer: 1589419
Opleiding: Technische Informatica

Datum: 17 maart 2015
Plaats: Utrecht

Bedrijf: &Ranj
Bedrijfsbegeleider: Frank Bovenkerk
Docentbegeleider en eerste examinator: Ander de Keijzer
Tweede examinator: Gerald Ovink

Het voorwoord

Life is a game.

Dat is het thema van het spel waarvoor ik tijdens dit project artificial intelligence heb gemaakt. In het spel heeft de speler net als in het echte leven een doel voor ogen en doet hij alles om zo snel mogelijk bij zijn doel te komen. Tijdens zijn reis kan hij geluk hebben of tegenslagen tegenkomen maar hij bepaalt zelf hoeveel risico hij wilt nemen. Uiteraard wilt hij zo veel mogelijk tegenslagen voorkomen maar vooral als hij achter staat wanneer het einde nadert moet hij wel. Gelukkig heeft hij nog een verzekeringskaart in zijn hand die hem een steuntje in de rug geeft zodat hij toch nog zijn doel kan bereiken.

Life is a game.

Nu ik terug kijk op het project zie ik dit niet alleen geld voor het leven in het algemeen maar dat ook dit project een spel is geweest met als doel het maken van deze scriptie en de AI Module. Ik heb vervelende bugs overwonnen door een goede architectuur op te hebben gezet en ik ben de gevolgen van ziek zijn voor geweest door een goede planning op te hebben gezet.

Maar niets van dit had ik kunnen doen zonder mijn persoonlijke verzekeringskaarten: Het bedrijf, de begeleiders en mijn familie en vrienden.

Ik wil graag iedereen bij &Ranj bedanken voor de geweldige tijd die ik tot nu toe bij jullie heb doorgebracht en mijn familie en vrienden voor de moral support de afgelopen maanden.

In het bijzonder wil ik graag Frank bedanken voor het ondersteunen van de loop van het project, Ivo voor de technische ondersteuning, Peter en Bruno voor hun bijdrage aan het functionele onderzoek en Ander voor de begeleiding vanuit school.

Bedankt!

Dennis Haverhals

De management samenvatting

In opdracht van &Ranj is gekeken naar de mogelijkheden van artificial intelligence voor een turn-based cross-platform online multiplayer bordspel in de vorm van een virtuele speler. Het toevoegen van een virtuele speler biedt veel nieuwe gameplay mogelijkheden. De virtuele speler kan bijvoorbeeld dienen als waardige tegenspeler voor spelers van dit spel wanneer geen andere spelers online zijn.

Voordat een virtuele speler gemaakt kan worden, moet eerst duidelijk zijn wat, waarom en hoe het moet functioneren. Om dit uit te zoeken is er een tweedelig onderzoek uitgevoerd naar de functionele- en technische specificaties van de virtuele speler. Het functionele onderzoek heeft de waarden in het spel in kaart gebracht waar de virtuele speler rekening mee moet houden tijdens het beslissen van de volgende zet. Daarnaast is samen met de ontwerpers van het spel verder gekeken naar de wijze van implementatie en de gameplay mogelijkheden. Tijdens het technisch onderzoek zijn een tiental literaire bronnen gebestudeerd over artificial intelligence voor bordspellen. Dit heeft geleid tot vier mogelijke strategieen en drie pathfinding algorithmen. Aan het eind van de onderzoeksfase zijn de resultaten van het complete onderderzoek verwerkt in een MoSCoW lijst van functionele en technische eisen.

Op basis van deze eisen is de software architectuur van de virtuele speler opgezet in samenhang met de bestaande software van het spel. De software architectuur bestaat uit de requirements architectuur en de solution architectuur. De requirements architectuur beschrijft wat de virtuele speler moet doen en aan welke voorwaarden de virtuele speler moet voldoen. De solution architectuur verwerkt de functionele beschrijving tot een blauwdruk voor de virtuele speler. Deze blauwdruk is gebruikt om de klassen van de virtuele speler op te zetten. Hierna zijn drie strategieën en twee algorithmen uit het technische onderzoek geimplementeerd.

De verschillende strategieen en algorithmen zijn tegen de voorwaarden uit de requirements architectuur getest. Uit de testresultaten is naar voren gekomen dat de state machine strategie en het A* pathfinding alorithme de beste resultaten geven. De aanbeveling is om deze strategie en dit algorithme te gebruiken mocht de virtuele speler als uitbreiding van het spel ingezet gaan worden.

De inhoudsopgave

1 De inleiding

&Ranj is een serious game bedrijf dat spellen maakt voor marketing of educatieve doeleinden. Een van de opdrachten van het bedrijf is het maken van een turn-based cross-platform online multiplayer bordspel. Voor een uitbreiding van het spel zijn de gameplay mogelijkheden en mogelijke technische uitwerkingen van een virtuele speler onderzocht en uitgewerkt in een prototype. De focus van dit verslag ligt op de uitwerking en de testresultaten van het prototype.

Het onderzoek heeft de functionele eisen en verschillende technische mogelijkheden (strategieen en algorithmen) van de virtuele speler in kaart gebracht. Deze eisen en mogelijkheden zijn daarna meegenomen in de basis van de uitwerking van het prototype: de systeem architectuur. De systeem architectuur bestaat uit twee delen: de requirements architectuur en de solution architectuur. De requirements architectuur beschrijft wat de virtuele speler moet doen en de voorwaarden waaraan de functionaliteit moet voldoen. De solution architectuur vormt de basis van de technische uitwerking van de functionaliteit. De verschillende strategieen en algorithmen zijn vervolgens geimplementeerd en getest tegen de voorwaarden uit de requirements architectuur.

Hoofdstuk 2 geeft een introductie van &Ranj en het team wat verantwoordelijk is voor het maken van het spel. Hoofdstuk 3 beschrijft de kwestie en hoofdstuk 4 worden de opdracht en doelstellingen doorgenomen. In hoofdstuk 5 worden de opzet en de resultaten van het onderzoek toegelicht. Hoofdstuk 6 beschrijft de functionele- en technische specificaties van de virtuele speler. In hoofdstuk 7 wordt de systeem architectuur ingeleid. In hoofdstuk 8 wordt in de requirements architectuur in detail behandeld en hoofdstuk 9 beschrijft de solution architectuur. In hoofdstuk 10 zijn de verschillende stategieën en algorithmen doorsproken en hoofdstuk 11 beschrijft de testresultaten. In hoofdstuk 12 worden conclusies en aanbevelingen beschreven en hoofdstuk 13 sluit het verslag af met een evaluatie van het project.

2 Het bedrijf

&Ranj is een serious game bedrijf in Rotterdam dat spellen maakt met een serieuze toepassing (marketing of educatie). Vaak zijn dit verhaal gerichte spellen waarin mensen leren zichzelf te ontwikkelen op sociale gebieden, al is het spel waar deze opdracht aan gekoppeld is een ander soort spel: een marketing game.

Met ongeveer dertig medewerkers is &Ranj een van de grotere serious game bedrijven in Nederland en Europa. Projecten worden uitgevoerd in multi-disciplinaire teams van zes tot twintig mensen. De teams bestaan o.a. uit game designers, artists, developers en projectleiders.

2.1 Het team

Het spel waar deze opdracht aan gekoppeld is, wordt gemaakt door Team Orange. Dit team is verantwoordelijk voor het maken van een turn-based cross-platform online multiplayer bordspel.

2.1.1 Het spel

Het spel is een online bordspel voor twee spelers waarin de gebruikers opdrachten krijgen die afgeleid zijn van speciale vrijetijdsbestedingen zoals het voorbereiden op een VIP feest en het rondleiden van je schoonmoeder. Deze opdrachten zijn te behalen door op verschillende punten van het speelbord te komen. De spelers kunnen op deze punten komen door middel van het spelen van loopacties. De afstand die ze vervolgens kunnen afleggen is gekoppeld aan hun levenspunten. Naast loopacties kunnen ook andere acties gespeeld worden waarmee ze onder andere hun eigen levenspunten kunnen verhogen en/of de levenspunten van de tegenstander kunnen verlagen.

Daarnaast zijn er Life Events: punten op het bord die de gebruiker een willekeurige tegenslag of boost kunnen geven. Tot slot kunnen spelers verzekeringskaarten spelen die hun levenspunten beschermen tegen negatieve acties van de tegenstander en/of negatieve life events.

3 De kwestie

In de huidige versie van het bordspel is het voor spelers alleen mogelijk om tegen elkaar te spelen. Dit betekent dat spelers niet kunnen spelen als niemand anders op dat moment ook wil spelen. Daarnaast tellen alle spellen die spelers tegen elkaar spelen mee voor de ranglijsten terwijl sommige spelers meer tijd nodig hebben dan anderen om goed te worden in het spel.

Een onderdeel binnen Artificial Intelligence (AI) in de game industrie is het simuleren van een (virtuele) speler door de computer. Dit kan een vijand zijn in spellen als Mario of een tegenstander als Gary in pokemon. In dit bordspel gaat het om het laatste: een tegenstander.

Het maken van virtuele tegenstanders is begonnen in de jaren 50, met als eerst de focus op spellen waar al de informatie altijd beschikbaar is als schaken en dammen. Deze virtuele spelers zijn nu zo goed dat ze op wereld niveau ingezet kunnen worden. (Billings; Davidson; Schaeffer; Szafron, 2001, p.1). Het voordeel van deze spellen is dat al de informatie altijd beschikbaar is.

In de laatste tien jaar is steeds meer onderzoek gedaan naar spellen waarbij niet altijd al de informatie beschikbaar is. Zo is gekeken naar Poker (Billings; Davidson; Schaeffer; Szafron; 2001), Kolonisten van Catan (Szita; Chaslot; Spronck; 2010) en Dominion (Jansen; Tollisen, 2014). Dit bordspel valt ook in deze categorie.

Een virtuele speler geeft veel nieuwe mogelijkheden voor het bordspel:

- Het maakt het mogelijk voor spelers om tegen de virtuele speler te spelen. Zo kan de gebruiker oefenen, bepaalde strategieen uit te proberen en het spel ook spelen wanneer niemand anders beschikbaar is.
- Het kan spelers laten zien hoe het spel gespeeld moet worden door de stappen van een virtuele speler te volgen.
- Het kan gebruikt worden om de balans in het spel te testen als je veel spellen laat spelen tussen twee virtuele spelers. Zo kunnen andere uitbreidingen of veranderingen goed en snel getest worden.

4 De opdracht

Het eindproduct van de afstudeeropdracht is de te ontwikkelen AI module die virtuele spelers het bordspel kan laten spelen. De virtuele speler moet op basis van de gegeven informatie intelligente keuzes maken om de kans op winnen zo groot mogelijk te maken.

4.1 De doelstelling

Op welke manier(en) de virtuele speler ingezet gaat worden staat nog niet vast, om deze reden is de opdracht zo opgezet dat het op meerdere manieren in te zetten valt. Het hoofddoel is het maken van een AI module die een virtuele speler het spel kan laten spelen. Dit kan tegen een normale speler of tegen een virtuele speler zijn.

4.2 De deelopdrachten

Het project is ingedeeld in vijf deelopdrachten met elk een eigen eindproduct en leiden samen tot het maken van een goede AI Module en sriptie.

4.2.1 Onderzoeksrapport over de AI module

Het onderzoeksrapport zal inzicht geven op de te maken functionaliteiten van de AI module en waar deze module zich in het programma zal bevinden (server-side of client-side). In het rapport zullen verschillende mogelijkheden naast elkaar gelegd worden en vanuit daar een onderbouwde keuze gemaakt worden tussen deze mogelijkheden.

4.2.2 Systeem Architectuur document

Wanneer de functionele eisen zijn opgesteld worden deze verwerkt tot requirements architectuur en vervolgens wordt dit gebruikt om een solution architectuur op te zetten. Dit kan vervolgens gebruikt worden om de AI module te gaan maken.

4.2.3 AI Module

Het eindproduct van deze afstudeeropdracht is een AI module voor het bordpsel. Daarnaast zal deze module compleet geintegreerd worden met het bestaande spel, op alle 3 de platforms. Zie hoofdstuk 8 voor meer informatie over deze AI module.

4.2.4 Testrapport

Het systeem wordt na het ontwikkelen getest tegen de opgezette functionele eisen. De resultaten hiervan worden beschreven in het testrapport.

4.2.5 Scriptie

De scriptie bevat het onderzoeksrapport, systeem architectuur document, de implementatie van de AI module en het testrapport.

5 Het onderzoek

Het doel van dit onderzoek is het vinden van de functionele en technische specificaties voor de virtuele speler in het spel - een multplayer online bordspel - met behulp van het beantwoorden van de onderzoeksvragen die opgesteld zijn aan het begin van het onderzoek. Dit verslag geeft een samenvatting van het onderzoek en zal enkel de punten bespreken die direct invloed hebben op rest van het verslag. Het volledige onderzoek is te vinden in Bijlage 3: Onderzoeksrapport.

5.1 De onderzoeksvragen

De onderzoeksvragen vormen de richtlijn voor het onderzoek en zorgen dat zo veel mogelijk bruikbare informatie verkregen wordt om de functionele en technische specificaties mee op te zetten.

5.1.1 De hoofdvraag

Welke AI module moet worden ontworpen en ontwikkeld om virtuele spelers het bordspel tegen andere (virtuele) spelers op elk niveau te kunnen laten spelen met als focus de normale speler een consistent uitdagende spelervaring te geven, zonder dat de speler het idee heeft dat de virtuele speler zonder reden acties onderneemt of dat de virtuele speler vals speelt?

5.1.2 De deelvragen

- Wat is een virtuele speler?
- Wat voegt de virtuele speler toe aan de spelervaring?
- Welke functionele specificaties worden gesteld aan de virtuele speler?
    - In welke maten moet de virtuele speler intelligentie bevatten?
    - Vanuit welke basis moet de virtuele speler keuzes maken?

- Welke technische specificaties worden gesteld aan de virtuele speler?
    - Hoe kan de virtuele speler toegepast worden in het bestaande spel?
    - Op welke manier verkrijgt de virtuele speler zijn kennis?

5.2 De resultaten

Het onderzoek bestaat uit twee onderdelen: het functionele onderzoek en het technische onderzoek. Het functionele onderzoek heeft de functionele specificaties in kaart gebracht en heeft een beter beeld gegeven op het implementatie proces. Het technische onderzoek heeft zicht gegeven op verschillende technische mogelijkheden voor de implementatie van de virtuele speler.

5.2.1 Het functionele onderzoek

Tijdens het functionele onderzoek is gekeken naar de gameplay factoren waar de virtuele speler rekening mee kan houden tijdens het spel. Het spel heeft negen bekende factoren en vier onbekende factoren. Uiteraard mag de virtuele speler geen onbekende factoren meenemen in zijn besluit.

Bekende factoren	**Onbekende factoren**
Levenspunten zelf en tegenstander	Kaarten in de trekstapel
Eigen kaarten	Kaarten van tegenstander
Eigen plaats op het bord	Consequenties van life events
Eigen verzekeringen	Inhoud van verzekeringen tegenstander
Plaats op het bord van de tegenstander	
Aantal kaarten tegenstander	
Plaats van (sub)objectives	
Plaats van life events	

Daarnaast heeft het functionele onderzoek inzicht gegeven op het feit dat het ontwerp en ontwikkel process iteratief moet zijn. Het is belangrijk dat er zo snel mogelijk tegen een virtuele speler gespeeld kan gaan worden. De daadwerkelijke intelligentie van de virtuele speler staat daardoor op de tweede plaats.

5.2.2 Het technische onderzoek

Het tweede deel van het onderzoek is het technische onderzoek. Hierin is duidelijk geworden dat de virtuele speler twee hoofd componenten moet gaan bevatten om het spel te kunnen spelen: Pathfinding en Strategy. Beide van deze componenten kunnen op verschillende wijzen geimplementeerd worden. Naarmate de orde van grootte van de implementatie stijgt, zal de implementatie tijd en de snelheid van het pathfinding algorithmen of intelligentie van de strategie van de virtuele speler verhoogd worden.

Orde van grootte	**Pathfinding**	**Strategy**
0		Random
1	Dijkstra	State Machine
2	A*	Monte Carlo
3	Bi-Directional	Neural Network

Daarnaast heeft het technische onderzoek inzicht gegeven op het feit dat het huidige spel compleet aan de server kant gespeeld wordt en dat de client het spel enkel visueel maakt voor de spelers. Hierdoor is het voor de AI Module enkel mogelijk om aan de server kant geimplementeerd te kunnen worden.

5.3 De conclusie

De twee onderzoeken samen hebben geleidt tot een tussentijds antwoord op de hoofdvraag:

Het bouwen van de AI module is een dynamisch process waarbij verschillende strategieen en algorithmen opeenvolgend geimplementeerd en getest moeten gaan worden. Hierdoor blijft Team Orange op de hoogte van de voortgang kan er vooruit gedacht worden over de mogelijke inzet van de virtuele speler. De eerste versie van de AI module zal daarom het Dijkstra pathfinding algorithme en een random strategy structuur bevatten om vervolgens te testen of dit een waardige uitdaging biedt aan de speler. Vervolgens zal een virtuele speler gemaakt gaan worden met A pathfinding algorithme en het state machine strategy structuur. Bij de state machine strategie zal de speler rekening houden met de bekende factoren in het spel. Tot slot zal mogelijk de Monte Carlo strategy geimplementeerd en getest worden.*

Dit zal de leiddraad zijn voor de rest van het ontwerp en ontwikkel process van de virtuele speler voor het digitale bordspel.

6 De specificaties

Uit het onderzoek zijn naast de antwoorden op de hoofd- en deelvragen ook de functionele en technische specificaties opgezet. Deze zijn op basis van MoSCoW prioritering gerangschikt. In deze lijst zijn de zwarte punten de functionele specificaties en de grijze punten de technische specificaties. De specificaties zijn opgezet aan de hand van de resultaten en conclusies uit het onderzoek.

6.1 De Must Haves

De must haves zorgen voor de basis van de AI Module. Het zorgt dat spelers het spel tegen een virtuele speler kunnen spelen. Deze virtuele speler zal de strategie en het pathfinding algorithme implementeren met de kleinste orde van grootte om zo snel mogelijk resulaat te kunnen zien.

- Spelers moeten tegen AI kunnen spelen.
- De virtuele speler moet een pad van A via B naar C kunnen lopen.
- De virtuele speler moet keuzes maken op basis van de huidige mogelijkheden.

- De AI module moet gemakkelijk te integreren zijn met het spel.
    - Met de huidige versie maar ook met toekomstige versies.
- Meerdere virtuele spelers op de server tegelijkertijd.
- Random strategy.
- Dijkstra Pathfinding.

6.2 De Should Haves

De should haves vormen de basis van de eerste intelligente virtuele speler. Deze virtuele speler zal de strategie en het pathfinding algorithme implementeren met de op een na kleinste orde van grootte om wederom het snelst te kunnen zien of de intelligentie van deze virtuele speler genoeg uitdaging biedt. Het tweede pathfinding algorithme zal

- De AI moet alle bekende factoren meenemen in zijn besluit
- De AI moet geen onbekende factoren meenemen in zijn besluit

- A* Pathfinding.
- State Machine Strategy.

6.3 De Could Haves

De could haves zorgen voor een tweede strategie voor de intelligente virtuele speler. Een tweede strategie zorgt voor beter onderbouwde testresultaten en kan met meer zekerheid gezegd worden dat de beste strategie ook daadwerkelijk het beste is.

- AI die beslissingen maakt op basis van simulaties en/of ervaringen van eerder gespeelde spellen.
- Virtuele spelers kunnen het spel tegen elkaar spelen.
    - Virtuele spelers genereren gevonden bugs.

- Monte Carlo Strategy

6.4 De Won't Haves

Dit project richt zich op het maken van een functionele virtuele speler waartegen spelers een uitdagend spel kunnen spelen. Door twee intelligente strategieen en twee pathfinding algorithmen naast enkel te kunnen zetten kan een goed beeld worden opgemaakt over de mogelijke inzetbaarheid van een virtuele speler als uitbreiding op het bestaande spel. Hierdoor zullen de strategie en het pathfinding algorithme met de hoogste orde van grootte buiten de scope vallen van dit project.

- Bi-Directional BFS Pathfinding
- Neural Network Strategy

7 De systeem architectuur

De systeem architectuur zet de AI Module in kaart aan de hand van modellen. Deze modellen zijn in twee groepen onder te brengen: de requirements architectuur en de solution architectuur. De requirements architectuur geeft een globaal beeld van de onderdelen van het systeem en hoe deze samenwerken om de wensen van de gebruiker te vervullen. De solution architectuur verwerkt deze onderdelen tot een blauwdruk van het systeem.

De AI Module is een uitbreiding op het bestaande spel en uit het onderzoek is gebleken dat deze uitbreiding het best aan de server-side van het programma geimplementeerd kan worden. Om de AI Module zo goed mogelijk te kunnen integreren met de rest van de applicatie is het van belang zicht te hebben op de architectuur van de server-side van het bestaande spel. Bij de onderdelen waar een direct verband is tussen het bestaande spel (de server) en de mogelijke uitbreiding (de AI module) wordt eerst de server en vervolgens de AI Module besproken.

8 De requirements architectuur

De requirements architectuur bestaat uit het use case model en het contraints model. Het use case model geeft zicht op wat het systeem voor de gebruiker moet doen. Het constraints model bevat informatie over waar het systeem aan moet voldoen. Hier worden performance en tijd restricties besproken.

8.1 Het use case model

Het use case model bevat het use case diagram en activity diagrams. Het use case diagram geeft een beeld van de use cases van het systeem en hoe deze samen de wensen van de gebruikers vervullen. De activity diagrams beschrijven een globale uitwerking van elk van de use cases.

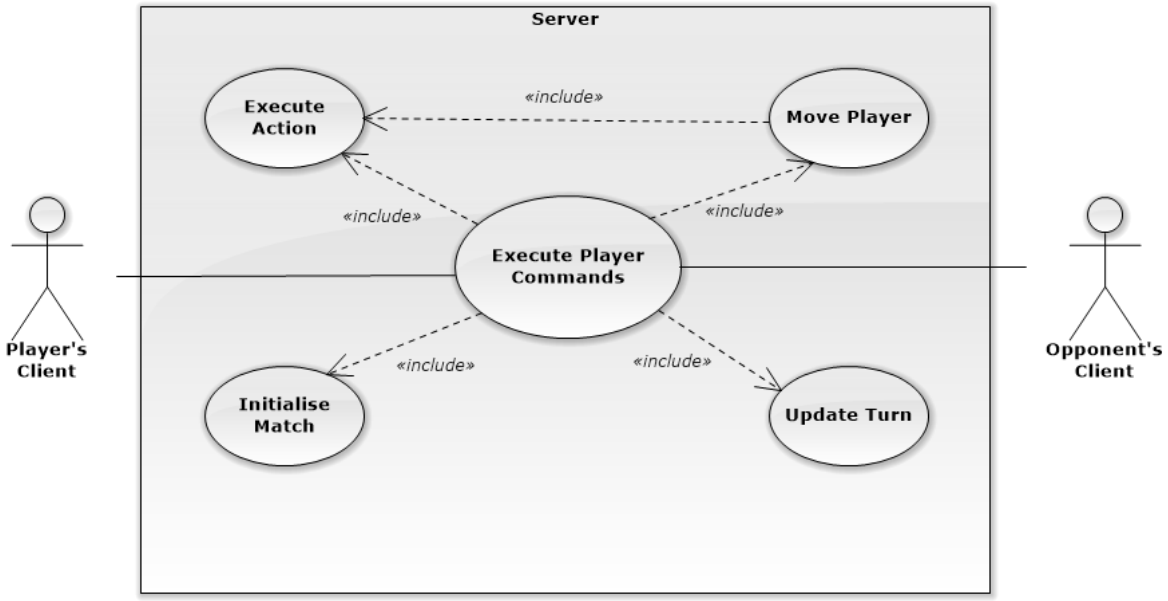
8.1.1 De gebruikers van de server

Het bestaande spel wordt gespeeld door twee spelers: de player en de opponent. Deze sturen via de client berichten naar de server. De gebruikers van de server zijn niet de player en de opponent zelf maar vanuit de server gezien zijn de gebruikers de clients van de player en de opponent.

Gebruiker	Beschrijving
Primair: Player's Client	De client van de player visualiseert en bestuurt de gebruiker voor een van de gebruikers
Primair: Opponent's Client	De client van de opponent visualiseert en bestuurt de gebruiker voor de andere speler.

8.1.2 Het use case diagram van de server

De server heeft één hoofddoel: het uitvoeren van de commando's die de clients van de player en de opponent naar de server sturen. Als een speler begint met spelen wordt de match geinitialiseerd, vervolgens spelen de player en opponent kaarten met bepaalde acties of lopen



ze over het bord. Tot slot wordt de beurt van de gebruiker bijgewerkt.

8.1.3 De gebruikers van de AI Module

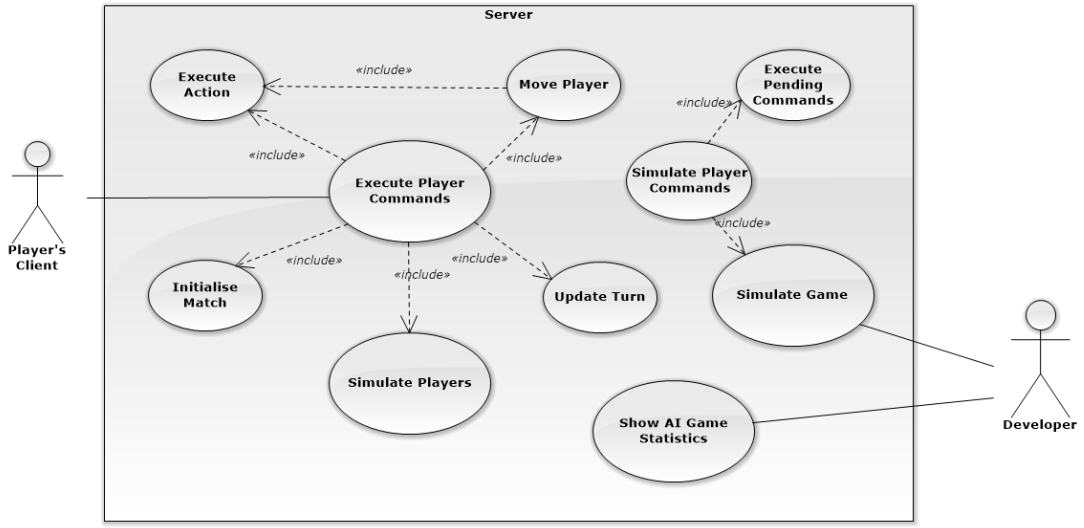
Een spel tegen een virtuele speler wordt door één speler gespeeld. Daarnaast kan een developer in een management tool statistieken zien van de spellen met virtuele spelers en zelf spellen starten die door twee virtuele spelers gespeeld worden.

Gebruiker	Beschrijving
Primair: Player's Client	De client van de player visualiseert en bestuurt de match voor een van de gebruikers
Primair: Developer	Bekijkt statistieken van spellen met virtuele spelers en start spellen waarbij twee virtuele spelers tegen elkaar spelen.

8.1.4 Het use case diagram van server en de AI Module

Het systeem met de AI Module heeft vier doelen

- Het uitvoeren van de commando's die de player's client naar de server stuurt.
- Het simuleren van speler commando's (door een virtuele speler) en deze uitvoeren.
- Het simuleren van een geheel spel (dat gespeeld wordt door twee virtuele spelers).



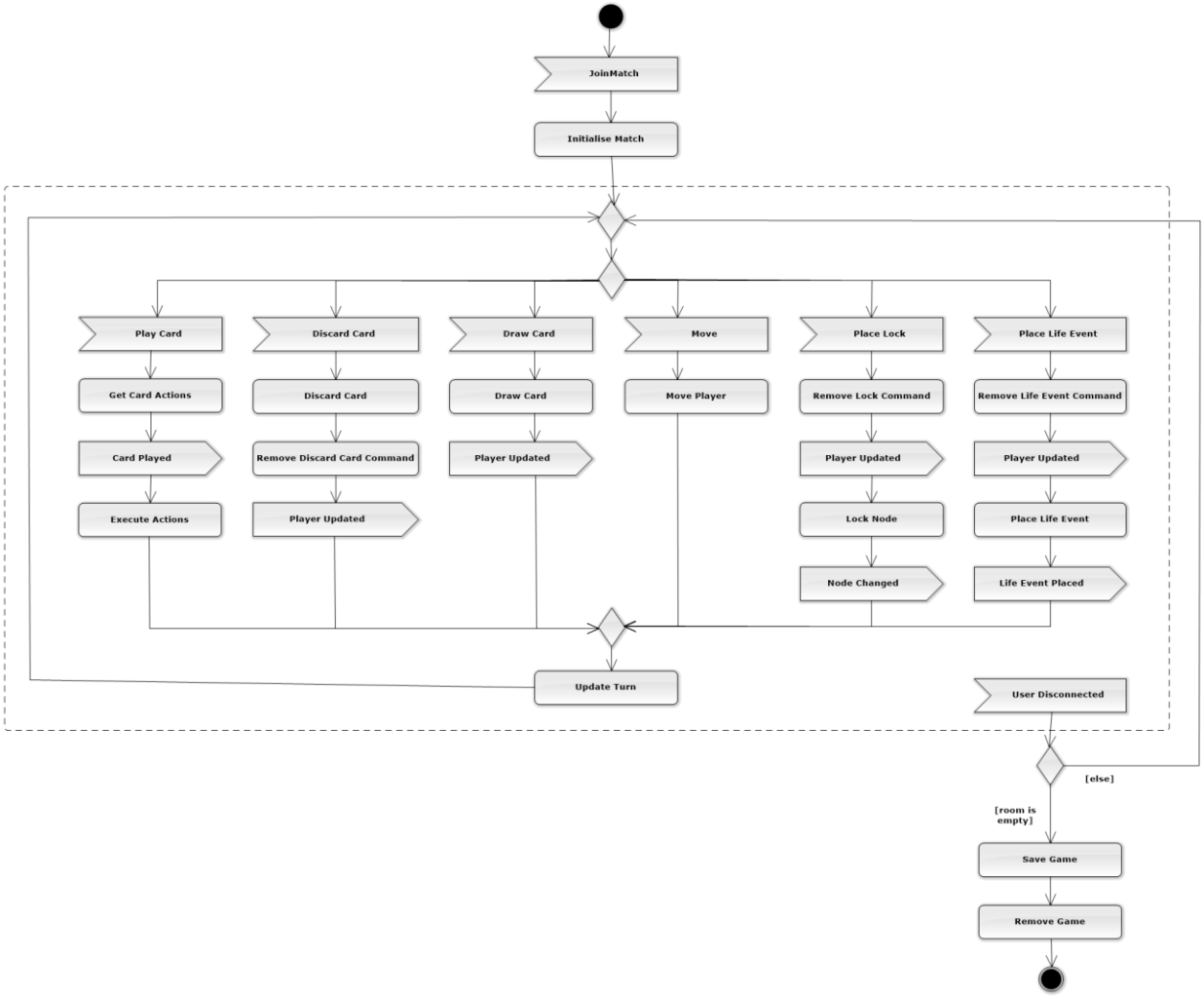
- Het laten zien van de statistieken van spellen met virtuele spelers.

8.2 De activity diagrams

Activity diagrams geven een beeld van de acties waaruit de use cases bestaan. Ook is hier te zien op welke manier de use cases op elkaar aansluiten. Aangezien een grote overlap bestaat tussen de activity diagrams van de server en van de AI Module zullen deze paralel besproken worden. De activity diagrams (AD's) zijn in te delen in drie groepen: activity diagrams van de server en de AI module die van elkaar verschillen, activity diagrams van de server en de AI Module die hetzelfde zijn en activity diagrams alleen van de AI Module.

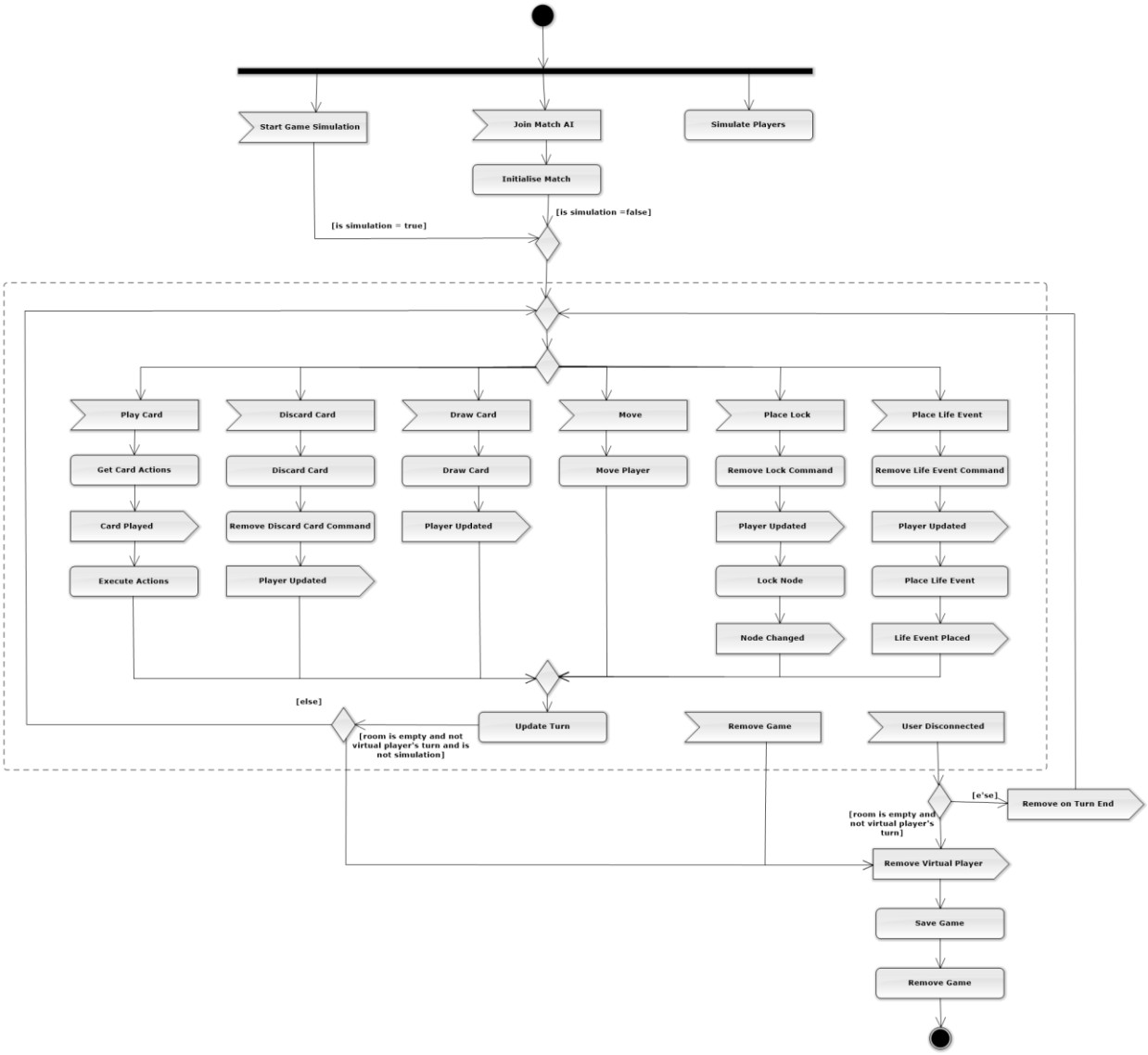
Van elkaar verschillende AD's	Aan elkaar gelijke AD's	AD's voor de AI Module
Execute player commands	Execute action	Simulate Players
Initialise match	Update turn	Simulate Player Commands
	Move player	Execute Pending Commands
		Simulate Game
		Show AI Game Statistics

8.2.1 Het activity diagram voor use case “Execute Player Commands” van de server

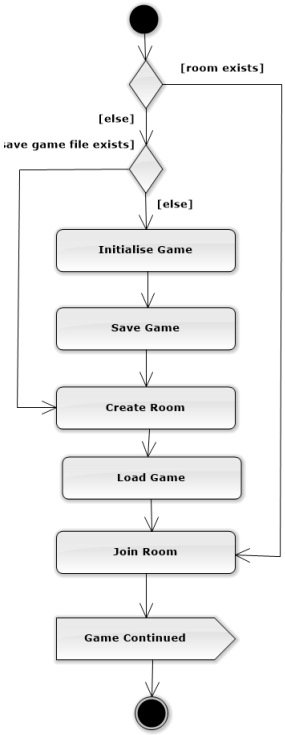


Wanneer de gebruiker een match begint te spelen, wordt de match geinitialiseerd. Tijdens de initiaisatie zal de gebruiker naar de room gestuurd waar de match zich bevindt. De server zal in de room luisteren naar binnenkomende berichten en deze uitvoeren wanneer de desbetreffende gebruiker aan de beurt is. Daarna wordt de beurt bijgewerkt en mogelijk overgedragen naar de andere speler en zal de server weer wachten op binnenkomende berichten. Dit herhaalt zich tot beide gebruikers room hebben verlaten. Wanneer de room leeg is zal de match opgeslagen worden en de room verwijderd worden.

8.2.2 Het activity diagram voor use case “Execute Player Commands” van de AI Module

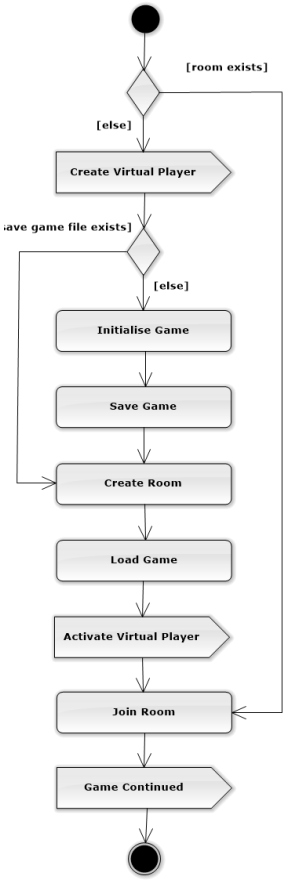


Naast het initialiseren van een match en het uitvoeren van speler commando’s zal in het begin de use case “simulate players” gestart worden waardoor als de gebruiker een match begint, de server een virtuele speler aan kan maken. Wanneer de gebruiker de room verlaat, zal de virtuele speler zijn beurt afmaken als hij aan de beurt is en vervolgens de match opslaan en de room en zichzelf verwijderen. Als de gebruiker zelf aan de beurt is, wordt de match direct opgeslagen en de room en de virtuele speler verwijderd.



8.2.3 Het activity diagram voor use case “Initialise Match” van de server

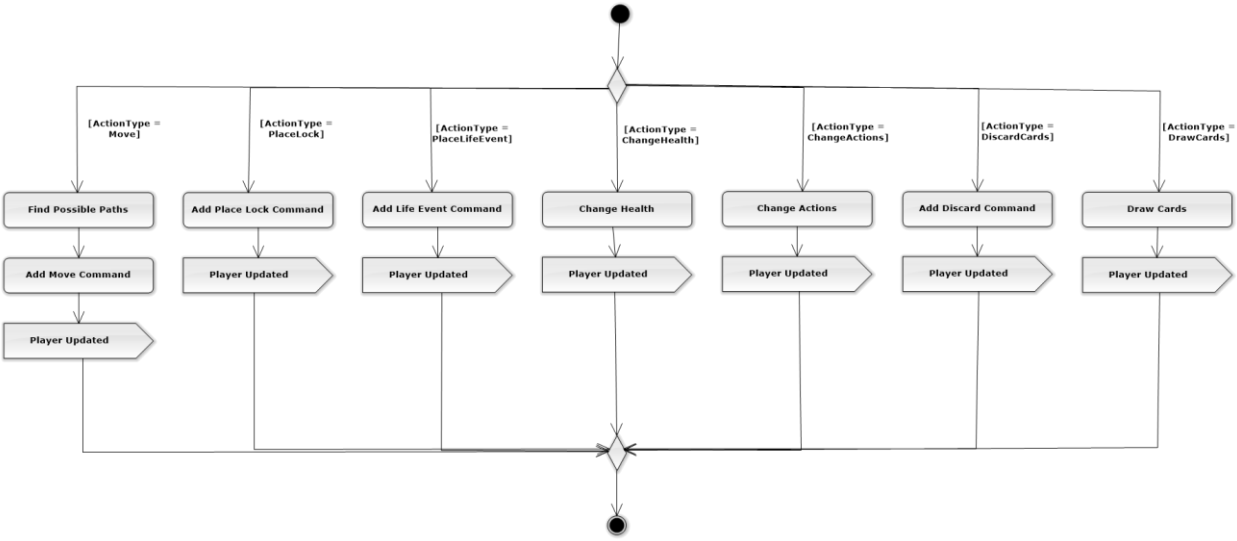
Tijdens de initialisatie van een match zal eerst gekeken worden of de room met de desbetreffende match al bestaat. Als dit zo is zal de gebruiker aan de room worden toegevoegd. Zo niet kijkt de server of een savegame bestaat. Als dit niet zo is zal de server een nieuwe match aanmaken en opslaan. Vervolgens wordt de room aangemaakt waarna de match wordt toegevoegd aan de room. Tot slot zal de gebruiker aan de nieuwe room toegevoegd worden. De gebruiker zal bij het binnenkomen van de room een Game Continued bericht krijgen zodat hij weet dat de match kan visualiseren en berichten kan sturen naar de server.



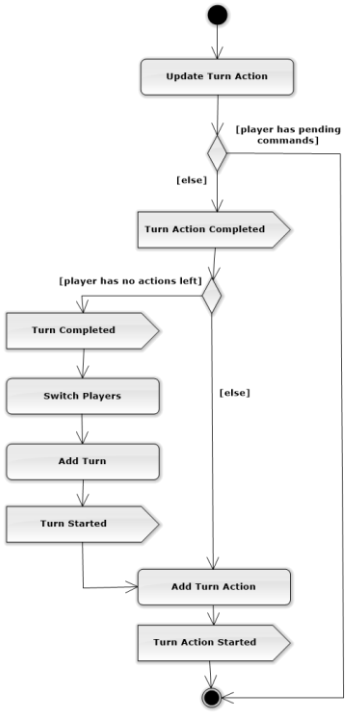
8.2.4 Het activity diagram voor use case “Initialise Match” van de AI Module

Bij een spel tegen een virtuele speler zal de server tijdens de initialisatie een virtuele speler aanmaken wanneer de room nog niet bestaat. Vervolgens zullen de match en de room worden gemaakt en zal zodra de match geladen is een bericht worden gestuurd om de virtuele speler de activeren. Tot slot zal de gebruiker aan de room worden toegevoegd en zal de gebruiker een bericht krijgen dat de match gestart is.

8.2.5 Het activity diagram voor use case “Execute Action” van de server en de AI Module

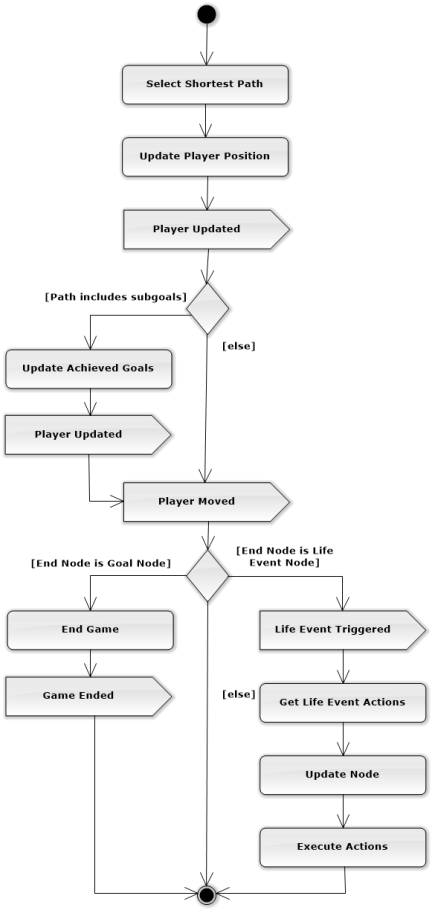


De acties van kaarten of life events maken kleine aanpassingen aan de match. Wanneer extra informatie van de match nodig is - bijvoorbeeld welke kant de match op wil lopen, waar de match een life event wil plaatsen of welke kaart hij weg wil gooien - stuurt de server een bericht naar de gebruiker of naar de virtuele speler om deze informatie te verkrijgen. Zowel bij aanpassingen als bij berichten voor meer informatie wordt een Player Updated bericht gestuurd.



8.2.6 Het activity diagram voor use case “Update Turn” van de server en de AI Module

Bij zowel de server als bij de AI Module wordt de beurt van de huidige speler aan het eind van een actie bijgewerkt. Dit gebeurt door eerst de huidige actie te bewerken. Vervolgens wordt gekeken of de speler openstaande commando’s heeft - zoals het weg moeten gooien van kaarten. Als dit zo is wordt er gewacht tot deze commando’s zijn verwerkt. In het geval dat de speler geen openstaande commando’s heeft wordt een Turn Action Completed bericht gestuurd. Wanneer de speler geen acties over heeft wordt de beurt overgedragen aan de andere speler. Hierbij worden de berichten Turn Completed en Turn Started gestuurd naar de (virtuele) spelers van de match. Tot slot wordt een nieuwe actie aan de beurt toegevoegd en wordt het bericht Turn Action Started gestuurd.



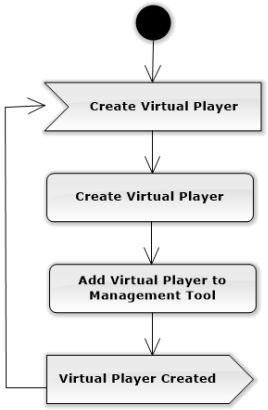
8.2.7 Het activity diagram voor use case “Move Player” van de server en de AI Module

Tijdens het verplaatsen van een speler over het bord wordt eerst het kortste pad gezocht naar de destinatie die de speler heeft aangegeven. Vervolgens wordt de positie van de speler bijgewerkt en wordt een Player Updated bericht gestuurd naar de clients van de spelers in de match. Daarna wordt gekeken of het gekozen pad een subdoel bevat. Als dit zo is wordt de lijst van behaalde subdoelen bijgewerkt en wordt er wederom een Player Updated bericht gestuurd. Vervolgens wordt het gekozen pad naar de (virtuele) spelers gestuurd door middel van een Player Moved bericht.

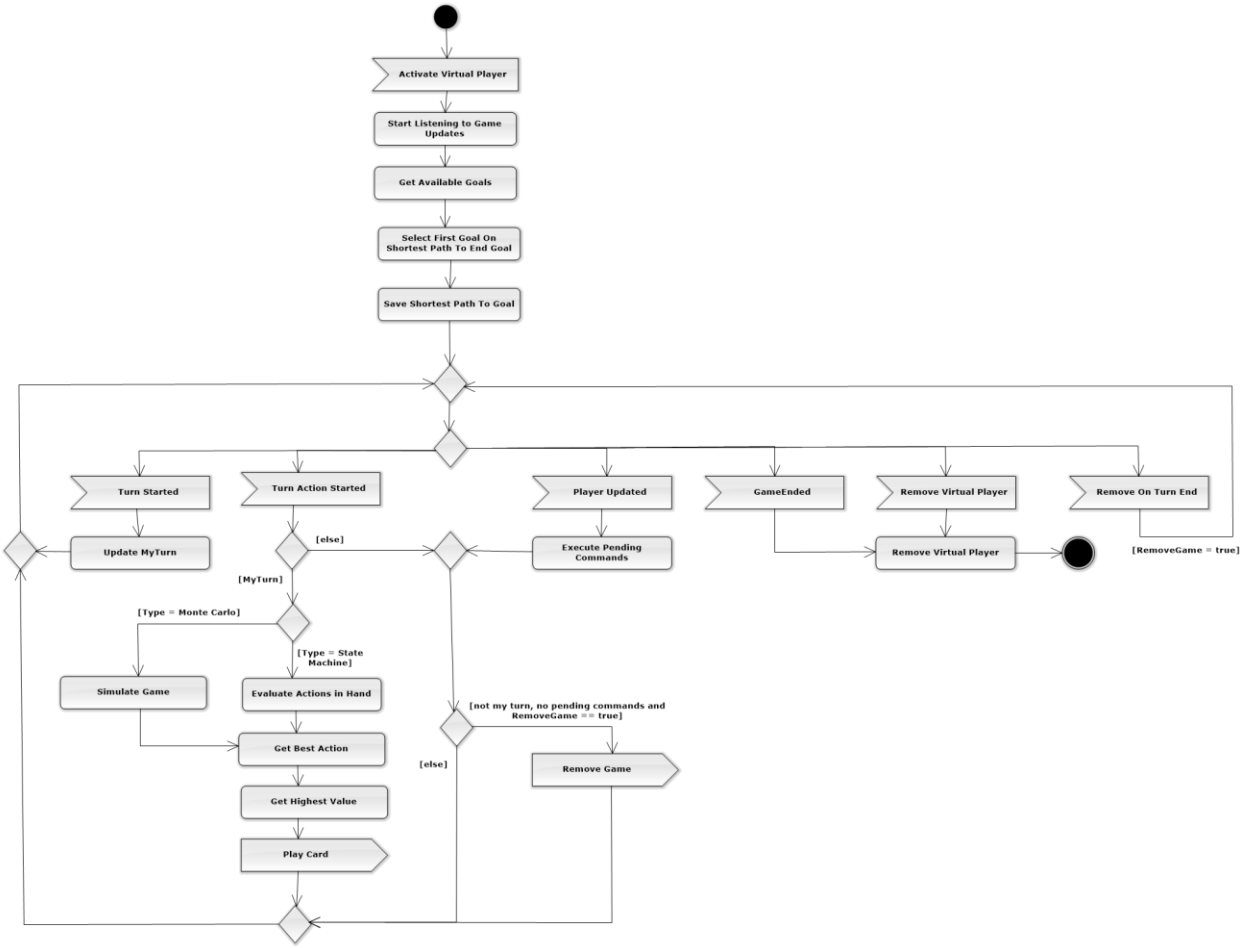
Tot slot wordt de laatste node bekeken. Als het de doel node is, wordt de match beeindigd en wordt er een Game Ended bericht gestuurd. Als het een Life Event node is, wordt er een Life Event Triggered bericht gestuurd en worden de acties van het Life Event opgezocht en het Life Event wordt van de node verwijderd. Tot slot zullen de acties van het Life Event uitgevoerd worden.

8.2.8 Het activity diagram voor use case “Simulate Player” van de AI Module

Wanneer het bericht “Create Virtual Player” binnenkomt zal een virtuele speler aangemaakt worden. Vervolgens zal de management tool op de hoogte worden gesteld van de aangemaakte virtuele speler. Tot slot zal het bericht “Virtual Player Created” verstuurd worden.



8.2.9 Het activity diagram voor use case “Simulate Player Commands” van de AI Module.

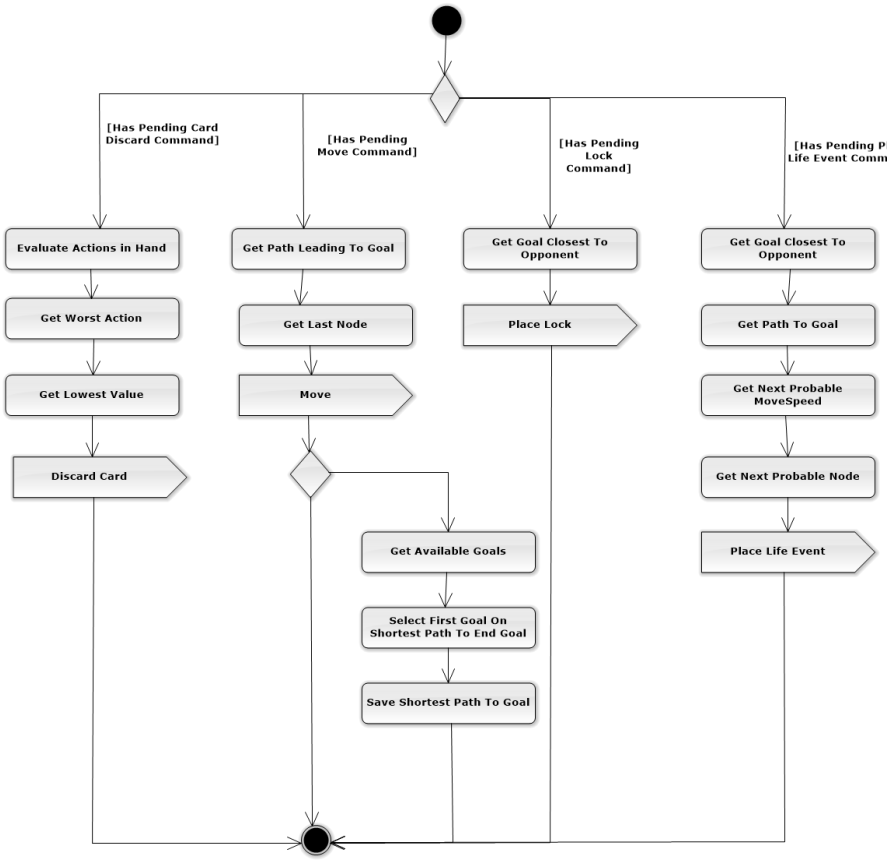


Wanneer een virtuele speler aan is gemaakt wacht hij totdat hij geactiveerd wordt. De activatie koppelt de virtuele speler aan een bepaalde match. Vervolgens zal gekeken worden of hij aan de beurt is en besloten worden naar welk (sub)doel hij toe gaat en welk pad hij zal nemen om zijn doel te bereiken.

Wanneer dit gebeurt is, zal hij wachten op berichten van de match.

- Bij een turn started bericht wisselt de staat of hij aan de beurt is of niet.
- Bij een turn action started bericht zal hij als hij aan de beurt is een kaart kiezen en spelen. Het kiezen kan door het simuleren van spellen en het kijken met welke kaart hij het meest wint, of door de waarden van de match op dat moment te bekijken zoals het aantal levenspunten, aantal kaarten in zijn hand, etc.
- Bij een player updated bericht reageert hij op openstaande commando’s.
- Bij een remove on turn end wordt de remove game variabele gezet om de match aan het eind van zijn beurt af te sluiten.
- Bij een game ended of remove virtual player bericht wordt de virtuele speler verwijderd.

8.2.10 Het activity diagram voor use case “Execute Pending Commands” van de AI Module.



Om openstaande commando’s uit te voeren wordt eerst gekeken welk van de commando’s open staat. Vervolgens wordt per commando een reactie gegenereerd.

Discard card command
De waarden van de kaarten worden bekeken en de minst waardevolle kaart wordt weggegooid.

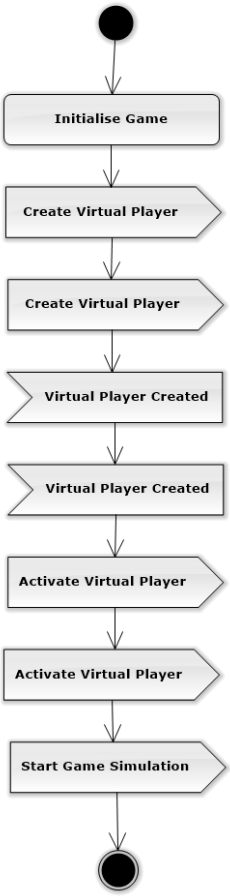
Move player command
Kijk welk van de mogelijke paden naar het huidige doel leidt en verplaats naar de laatste node van dat pad. Wanneer het doel bereikt is, kies een nieuw doel en vind het kortste pad naar het nieuwe doel.

Place lock command
Zoek het dichtstbijzijnde doel van de tegenstander en lock dat doel.

Place life event command
Zoek het dichtstbijzijnde doel van de tegenstander, zoek de node waar de tegenstander waarschijnlijk heen gaat lopen, plaats een life event op die node.

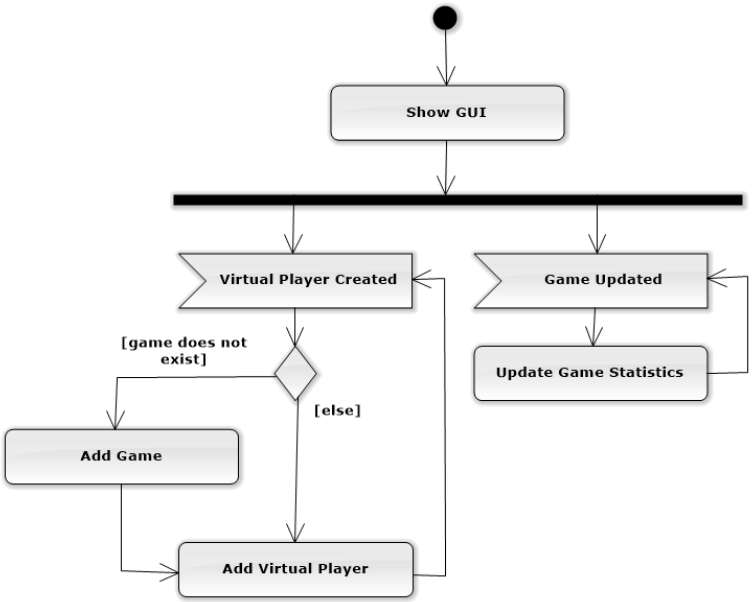
8.2.11 Het activity diagram voor use case “Simulate Game” van de AI Module.

Om een game te simuleren wordt eerst een spel geinitialiseert, vervolgens twee virtuele spelers aangemaakt en nadat deze aangemaakt zijn geactiveert. Tot slot wordt de match gestart.



8.2.12 Het activity diagram voor use case “Show AI Game Statistics” van de AI Module.

Bij het aanmaken van de management tool wordt de GUI van de tool getoont. Vervolgens wacht de tool op Virtual Player Created en Player Updated berichten vanuit de verschillende spellen en zal hij de informatie toevoegen of bijwerken.



8.3 Het constraints model

Het constraints model bevat voorwaarden waaraan de AI module moet voldoen. Deze voorwaarden worden in de testresultaten doorgenomen om te valideren dat het systeem aan de opgestelde eisen voldoet.

Constrainttype	Identificatie	Beschrijving	Eis	Verificatie
Accuracy	Doelgericht lopen over het bord	De virtuele speler moet doelgericht over het bord lopen (niet in circels zonder iets te behalen)	Minimaal 99% de goede kant op lopen	Playtesting analyse
	Het spelen van zinvolle kaarten	De virtuele spelers zonder random strategie mogen geen kaarten spelen die geen zin hebben	Minimaal 99% zinvolle kaarten spelen	
Performance	Spelen van een kaart	Tijd dat het duurt voor de virtuele speler om een kaart te spelen	Maximaal 3 sec	Tijd analyse:
	Bijwerken van de management tool	Tijd dat het duurt om een game update te visualiseren in de management tool	Maximaal 1 sec	
	Aanmaken en activeren van een virtuele speler	Tijd dat het duurt voordat de virtuele speler de match aan het spelen is.	Maximaal 500 ms	
	Win/Lose ratio	Percentage van games die de virtuele speler moet kunnen winnen van een gemiddelde speler	Minimaal 30%	

8.3.1 De playtesting analyse

Het analyseren van de acties die de virtuele speler doet. De acties die de virtuele speler uitvoert worden bijgehouden in de AI Management Tool en kunnen worden teruggevonden in tekstbestanden die de staat van het spel en de acties die de spelers uitvoeren bijhoudt.

8.3.2 De tijd analyse

Het analyseren van de tijd dat het duurt om de beschreven acties uit te voeren. Dit wordt gedaan door de tijden bij te houden in de AI management tool en op te slaan in de tekstbestanden

9 De solution architectuur

Het tweede deel van de software achitectuur is de solution architectuur. Hier worden de functionaliteiten van de requirements achitectuur omgezet naar een blauwdruk voor de code. De solution achitectuur bestaat uit drie delen: het class model, het concurrency model en het dynamic model. In het class model worden objecten in kaart gebracht door middel van een klassendiagram, in het cuncurrency model worden de objecten tot taken verwerkt en in het dynamic model worden de taken tot in detail uitgewerkt.

Ook in dit hoofdstuk wordt bij de onderdelen waar een direct verband is tussen het bestaande spel (de server) en de uitbreiding (de AI module) eerst de server en vervolgens de AI Module besproken.

9.1 Het class model

Het doel van het class model is het opstellen van een initieel klassendiagram. Dit begint met het vinden van objecten en vervolgens wordt gekeken hoe deze objecten zich tot elkaar verhouden.

9.1.1 De objecten van de server

Door de bestaande code te onderzoeken zijn de onderstaande objecten gevonden.

<u>De Controller objecten</u>

Controller objecten voeren functies uit die de staat van de speler of de match beinvloeden of verkrijgen data van entity objecten voor andere controllers die de staat van de speler of de match aanpast.

Object	**Beschrijving**
GameServerExtension	Portaal waar gebruikers op inloggen.
JoinMatchHandler	Start een nieuwe match of voegt een gebruiker toe aan een bestaande match.
GameServerRoom-Extension	Stuurt berichten van gebruikers door naar het gamemodel, en stuurt berichten van het gamemodel terug naar de gebruikers.
GameModel	Onvangt commando’s van de gameserverroomextension, begeleid de verwerking van de commando’s. Stuurt berichten terug tijdens de verwerking van het commando.
GameLogic	Verkrijgt de data die nodig is voor het bijwerken van het gamemodel en het sturen van de berichten naar de client.
CardsModel	Verkrijgt data van kaarten.
GoalsModel	Verkrijgt data van doelen.
BoardModel	Verkrijgt data van de verschillende borden en werkt deze bij.

LifeEventsModel	Verkrijgt data van Life Events.
TurnModel	Verkrijgt data van beurten en werkt deze bij.
GameSettingsModel	Verkrijgt data van instellingen.
TurnAction	Houdt data bij van een actie en voert de deze acties uit.
InsuranceEffect	Houdt data bij van insurance effecten en voert deze effecten uit.
GameEffect	Houdt data bij van game effecten en voert deze effecten uit.

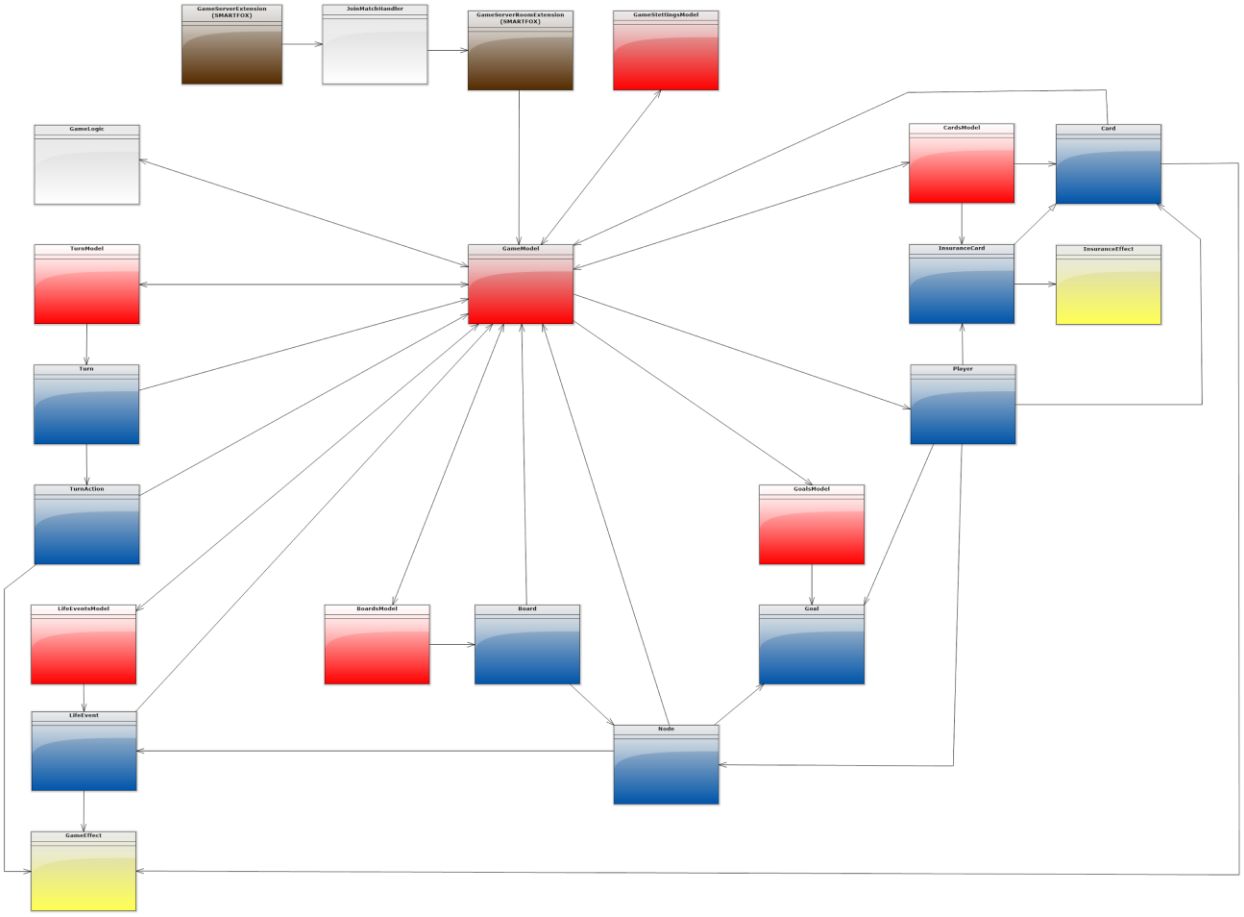
De entity Objecten

Entity Objecten zijn objecten die data bijhouden over een deel van de match.

Player	Houdt data bij van een van de gebruikers.
Card	Houdt data bij van kaarten.
InsuranceCard	Houdt data bij van insurance kaarten.
Goal	Houdt data bij van een doel.
Board	Houdt data bij van een bord.
Node	Houdt data bij van een node.
LifeEvent	Houdt data bij van een life event.
Turn	Houdt data bij van een beurt.

9.1.2 Het Initiele klassendiagram van de server

Door de associaties tussen de objecten in de bestaande code te onderzoeken is het onderstaande initiele klassendiagram van de server gemodelleerd.



9.1.3 De objecten van de AI Module

Door de use cases en activity diagrams te analyseren zijn objecten gevonden die aan de server toegevoegd moeten worden om de AI module te verwezenlijken. In deze paragraaf zijn enkel de objecten van de server meegenomen waar de AI Module mee communiceert. Het complete klassendiagram is te vinden in paragraaf 9.4

De controller Objecten

Controller objecten voeren functies uit die de staat van de gebruiker of de match beinvloeden of verkrijgen data van entity objecten voor andere controllers die de staat van de gebruiker of de match aanpast.

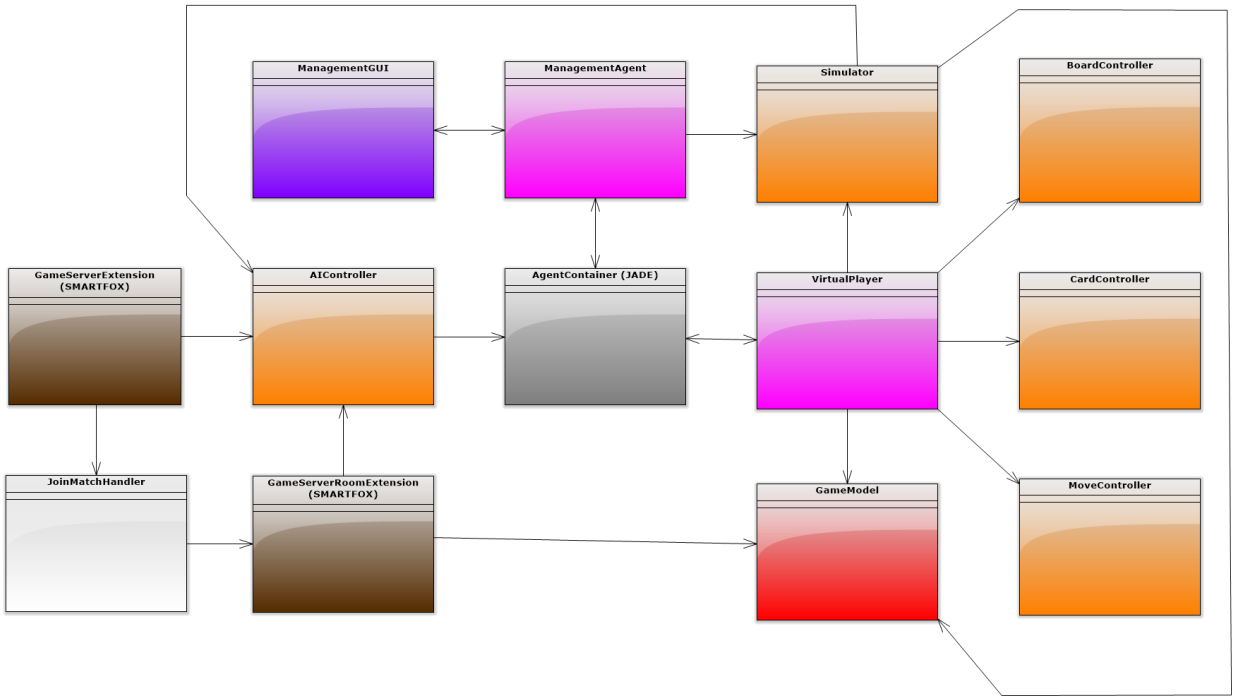
Object	**Beschrijving**
GameServerExtension	Portaal waar gebruikers op inloggen.
JoinMatchAIHandler	Start een nieuwe match en stuurt bericht om een virtuele speler aan te maken of voegt een gebruiker toe aan een bestaande match.
GameServerRoomExtension	Activeert de virtuele speler, stuurt berichten van gebruikers door naar het gamemodel en stuurt berichten van het gamemodel terug naar de gebruikers.
GameModel	Onvangt commando's van de gameserverroomextension en virtuele spelers en begeleid de verwerking van de commando's. Stuurt berichten terug tijdens de verwerking van het commando.
AIController (use case: Simulate Player)	Stuurt berichten om virtuele spelers aan te maken. Ontvangt berichten van aangemaakte virtuele spelers. Stuurt deze berichten door naar de geintresseerden.
AgentContainer	Beheert alle agents.
ManagementTool (use case: Show AI Game Statistics)	Beheert gegevens over spellen met virtuele spellen.
Virtual Player (use case: Simulate Player Commands)	Simuleert speler commando's en voert deze uit.
Simulator (use case: Simulate Game)	Start spellen met twee virtuele spelers.
BoardController (use case: execute pending commands)	Simuleert speler commando's: Place Lock & Place Life Event
CardController (use case: execute pending commands)	Simuleert speler commando's: Play Card en Discard Card
MoveController (use case: execute pending commands)	Simuleert speler commando: Move Player

De boundary Objecten

Object	Beschrijving
ManagementGUI	GUI voor het laten zien van statistieken van spellen met virtuele spelers en het starten van simulaties.

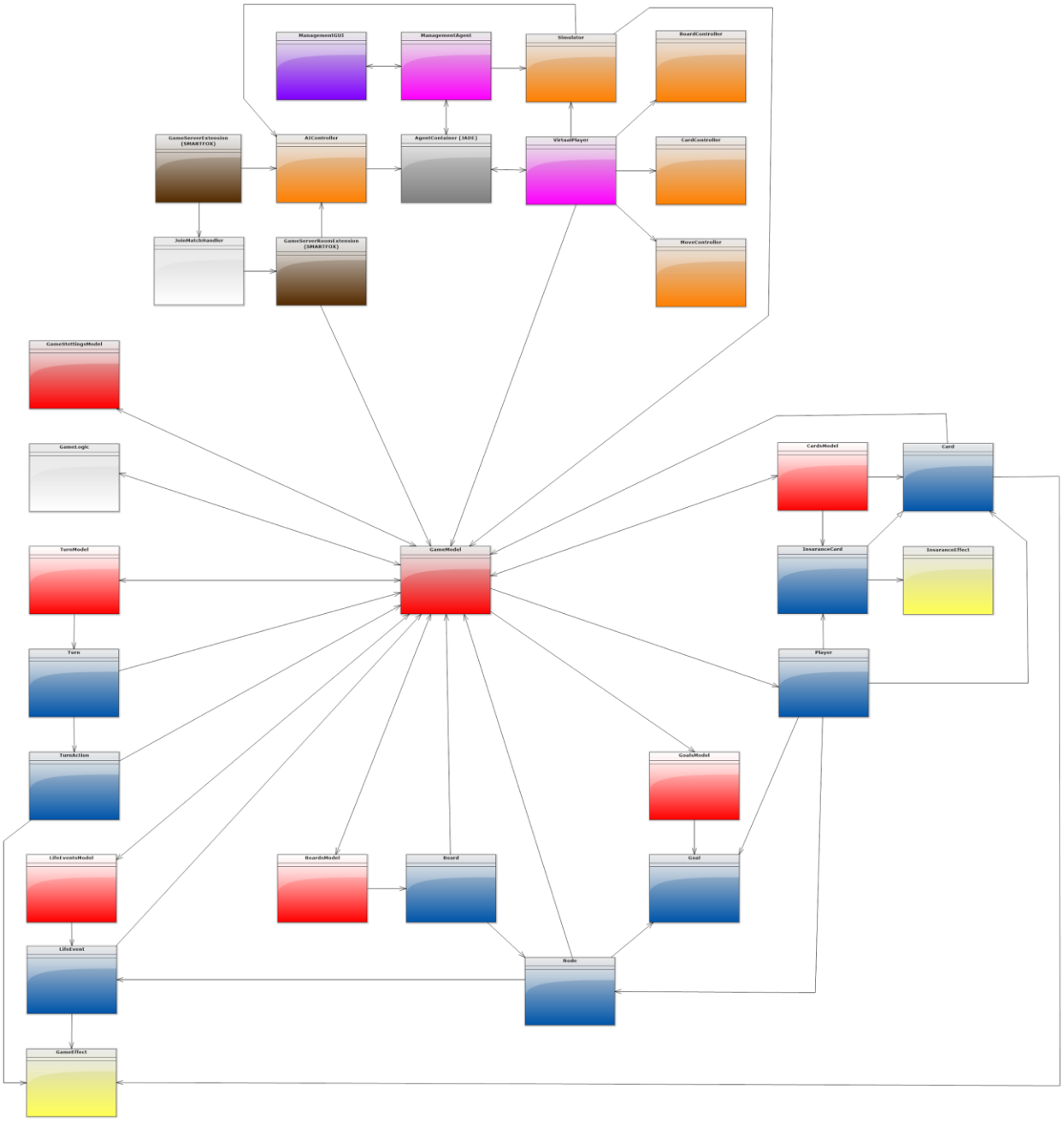
9.1.4 Het initiele klassendiagram van de AI Module

Door de associaties van de objecten te onderzoeken in het use case diagram en de activity diagrams is het volgende klassendiagram tot stand gekomen.



9.1.5 Het complete initiele klassendiagram

Een overzicht van het complete klassendiagram op basis van entity, controller en boundary objecten.



9.2 Het concurrency model

Het concurrency model leidt tot het maken van een concurrency diagram. Een concurrency diagram beschrijft de uitwisseling van informatie tussen taken. Hiervoor worden eerst taken opgesteld en vervolgens wordt gekeken naar de informatie uitwisseling tussen deze taken.

9.2.1 De taak structurering van de server

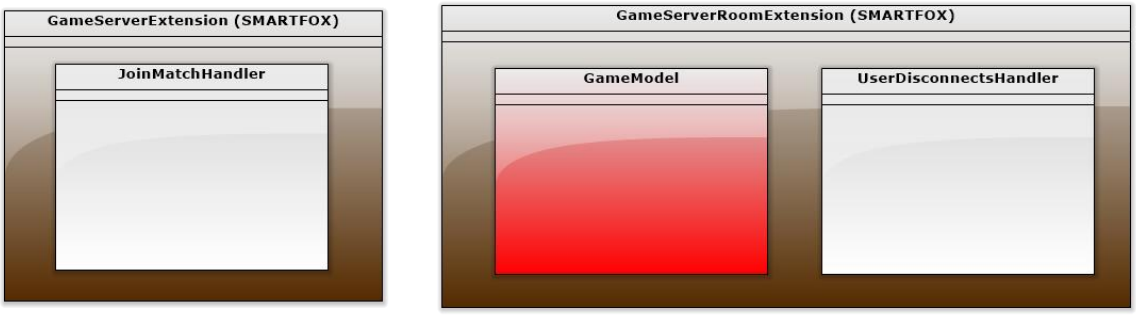
Taken beheren elk een deel van de objecten uit het initiele klassendiagram. De taken van de server zijn uit de bestaande code gehaald, de taken van de AIModule zullen opgesteld zijn.

De taken van de server

De taken van de server zitten in de standaard structuur van de SmartfoxServer die gebruikt is om de server-side te programmeren.

Taken	**Beschrijving**
GameServerExtension - JoinMatchHandler	De gameserverextension taak regelt alles vanaf het moment dat de gebruiker verbonden is met de server totdat de gebruiker een room binnen gaat.
GameServerRoomExtension - GameModel - UserDisconnectsHandler	De gameserverroomextension taak is de taak waarin de gebruiker met de match kan communiceren. Het houdt de match bij en verandert het op basis van binnenkomende commando's. Daarnaast stuurt hij de veranderingen door naar de gebruikers in de room. Wanneer een gebruiker de match afsluit wordt de gebruiker uit de room verwijderd en zal wanneer de room leeg is de match opslaan en de room zelf ook verwijderen.

9.2.2 Het task structure diagram van de server



9.2.3 De taak structurering van de AI Module

Om de taken op te stellen van de AI Module moeten de boundary- en controller objecten eerst gecategoriseerd worden.

De objecten van de AI Module

Om de objecten van de AI Module te kunnen categoriseren is voor elk object het type, de deadline en de prioriteit vastgesteld.

Type

De AI Module bevat twee type objecten: asynchroon interne objecten en asynchroon I/O objecten. Asynchroon betekent dat het object alleen aangeroepen wordt door een actie van een ander object. Intern betekent dat het object geen interactie heeft met de wereld buiten het systeem. I/O heeft juist interactie met de wereld buiten het systeem.

Deadline

De deadlines zijn gehaald uit het constraints tabel van de requirements achitectuur.

Prioriteit

De prioriteit is opgesteld aan de hand van de deadlines en zal dienen om taken mee op te stellen.

Object	Type	Deadline	Prioriteit
AIController	Asynchroon Intern	500ms	1
AgentContainer	Asynchroon Intern	500ms	1
ManagementTool	Asynchroon Intern	1 sec	2
Virtual Player	Asynchroon Intern	3 sec	3
Simulator	Asynchroon Intern	1 sec	2
BoardController	Asynchroon Intern	3 sec	3
CardController	Asynchroon Intern	3 sec	3
MoveController	Asynchroon Intern	3 sec	3
ManagementGUI	Asynchroon I/0	1 sec	2

Samenvoegen van de objecten

Het samenvoegen van de objecten gebeurt op basis van drie vormen van cohesie: temporele cohesie, sequentiele cohesie en control cohesie. Hierbij wordt gelet op de eerder opgestelde deadlines en prioriteiten van de objecten.

Temporele cohesie

Temporele cohesie is mogelijk wanneer verschillende objecten door hetzelfde event actief worden (bijvoorbeeld een klok op dezelfde frequentie). Dit is niet van toepassing in de AI Module.

Sequentiele Cohesie

Wanneer objecten in sequentie worden uitgevoerd - Object A activeert B en B activeert C - en Object A wacht op C, dan is er sprake van sequentiele cohesie. Ook dit komt niet voor in de AI Module.

Control Cohesie

Control Cohesie vindt plaats wanneer Object A als enigste communiceert met Object B.

- De VirtualPlayer communiceert als enigste met de BoardController, de CardController en de MoveController.
- De ManagementTool cummuniceert als enigste met de ManagementGUI.
- De AIController communiceert als enigste met de AgentContainer.

De simulator is toegevoegd aan de ManagementTool en de VirtualPlayer omdat de functionaliteit te weinig is om in een aparte taak te draaien.

Samengevoegde taken van de AI Module

Taken	**Beschrijving**
AIController - AgentContainer	De AIController taak beheert het aanmaken van virtuele spelers.
ManagementTool - ManagementGUI - Simulator	De managamentAgent taak deelt informatie van de spellen met virtuele spelers met de developer en kan spellen simuleren.
VirtualPlayer - BoardController - CardController - MoveController - Simulator	De virtuele speler taak simuleert speler commando's en voert deze uit in het game model. De veranderingen worden naar alle (virtuele) spelers in de match gestuurd. Ook de virtuele speler kan simulaties van de match starten om zo een keuze te kunnen maken wanneer een kaart gespeeld moet worden.

Samenvoeging van de server en AIModule

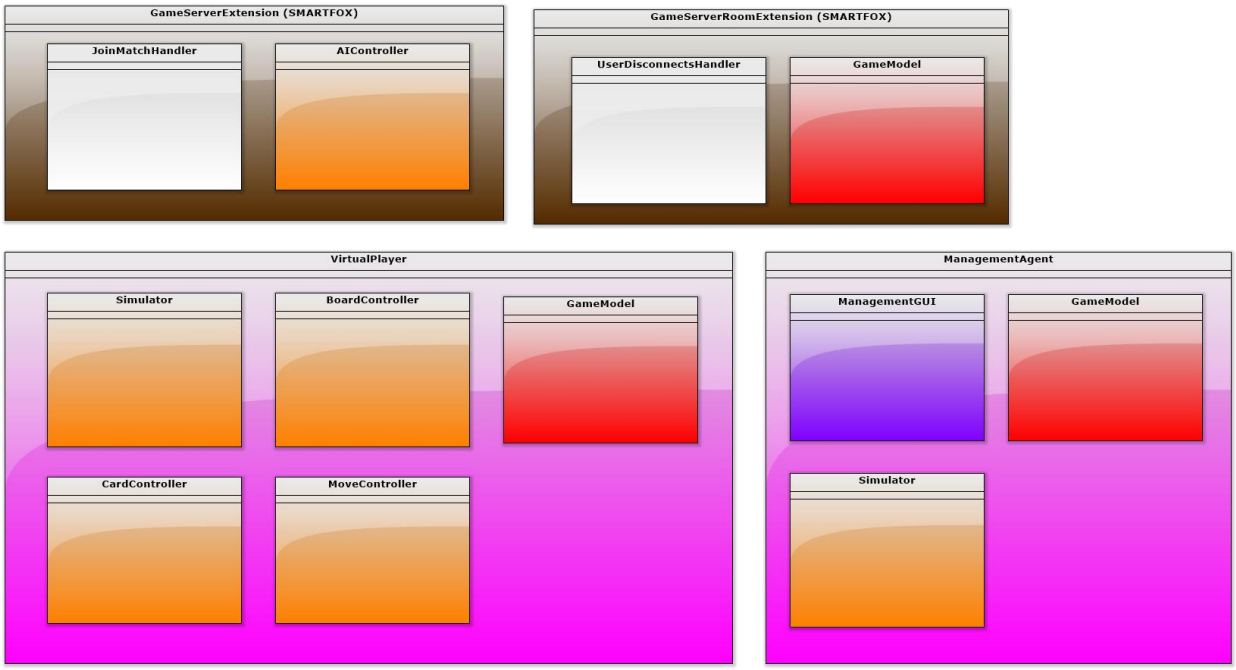
De gameserverextension is de enigste die met de AIController communiceert. Op basis van control cohesie worden deze ook samengevoegd.

Compleet overzicht van de taken

Taken	**Beschrijving**
GameServerExtension - JoinMatchHandler - AIController	De gameserverextension taak regelt alles vanaf het moment dat de gebruiker verbonden is met de server totdat de gebruiker een room binnen gaat. De JoinMatchHandler regelt de overplaatsing naar de room. De AIController creert virtuele spelers en stuurt berichten wanneer de virtuele spelers aangemaakt zijn.
GameServerRoomExtension - GameModel - UserDisconnects-Handler	De gameserverroomextension taak is de taak waarin de gebruiker met de match kan communiceren. Het houdt de match bij en verandert het op basis van binnenkomende commando's. Daarnaast stuurt hij de veranderingen door naar de gebruikers in de room. Wanneer een gebruiker de match afsluit wordt de gebruiker uit de room verwijderd en zal wanneer de room leeg is de match opslaan en de room zelf ook verwijderen.
ManagementTool - ManagementGUI - Simulator - GameModel	De managamentTool taak deelt informatie van de spellen met virtuele spelers met de developer en kan spellen simuleren. Het simuleren gebeurt door twee virtuele spelers aan te maken en een nieuw GameModel aan de virtuele spelers te koppelen.
VirtualPlayer - BoardController - CardController - MoveController - Simulator - GameModel	De virtuele speler taak simuleert speler commando's en voert deze uit in het GameModel. De veranderingen die aan het GameModel gemaakt worden door deze commando's worden naar alle (virtuele) spelers in de match gestuurd. Ook de virtuele speler kan simulaties van de match starten - simulaties die beginnen in de huidige stand van de match - om zo een keuze te kunnen maken wanneer een kaart gespeeld moet worden.

9.2.4 Het task structure diagram van de server en de AI Module

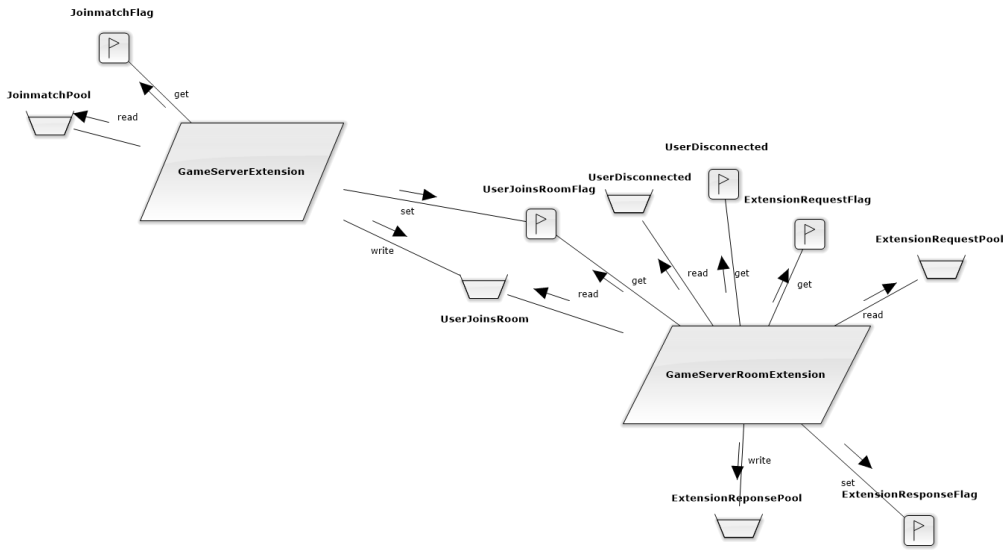
De samengevoegde taken vormen samen het compelete task structure diagram.



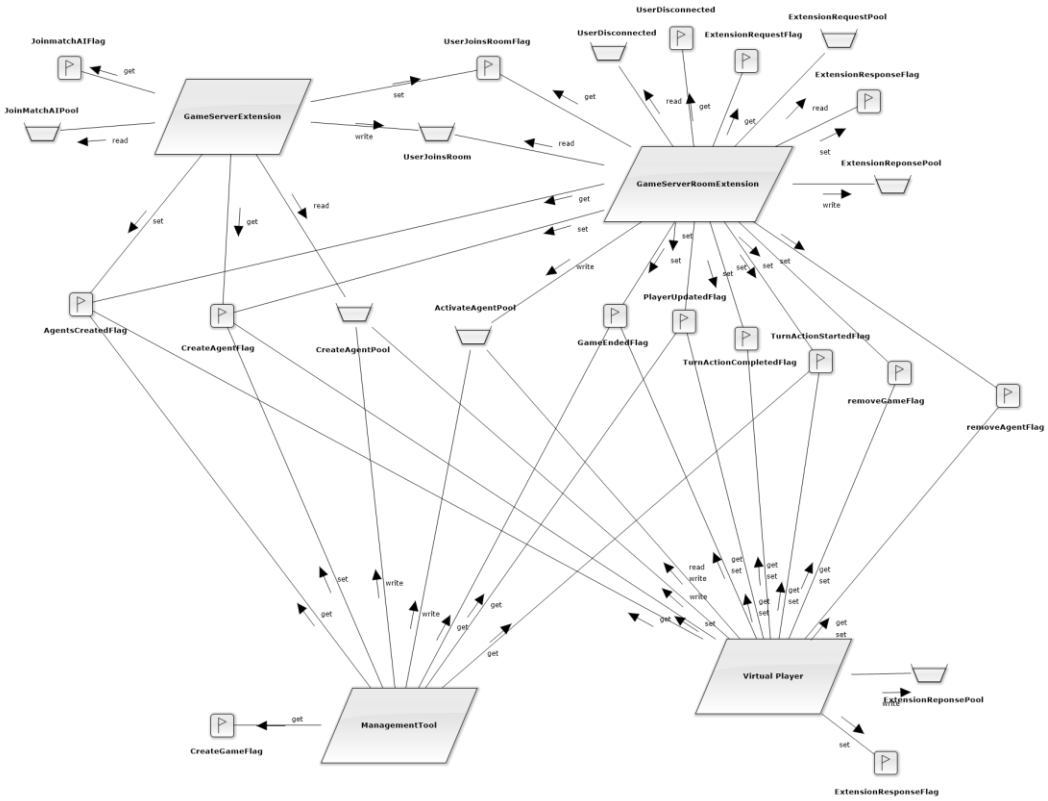
9.2.5 Het concurrency diagram

Het concurrency diagram brengt de in de tabel beschreven communicatie tussen de taken in beeld.

<u>Concurrency diagram van de server</u>



Concurrency diagram van de server en de AI Module

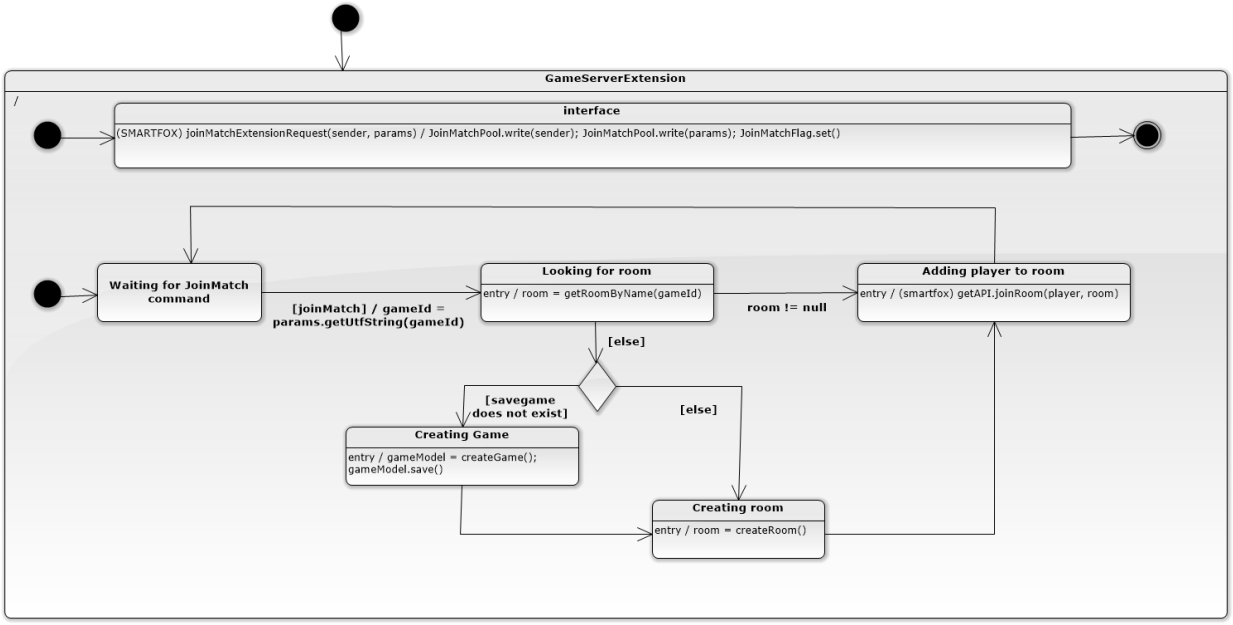


9.3 Het dynamic model

Het dynamic model werkt de taken in detail uit in state transition diagrams. Deze diagrammen laten voor elke taak zien wanneer de taak op welke events reagert en wanneer de taak events uit stuurt naar andere taken.

De state transition diagrams worden beschreven aan de hand van de doelen, de inkomende en uitgaande events.

9.3.1 Het state transition diagram voor de “GameServerExtension” taak van de server



Doelen:

- Het starten van een match in een room
- Het toevoegen van spelers aan een room.

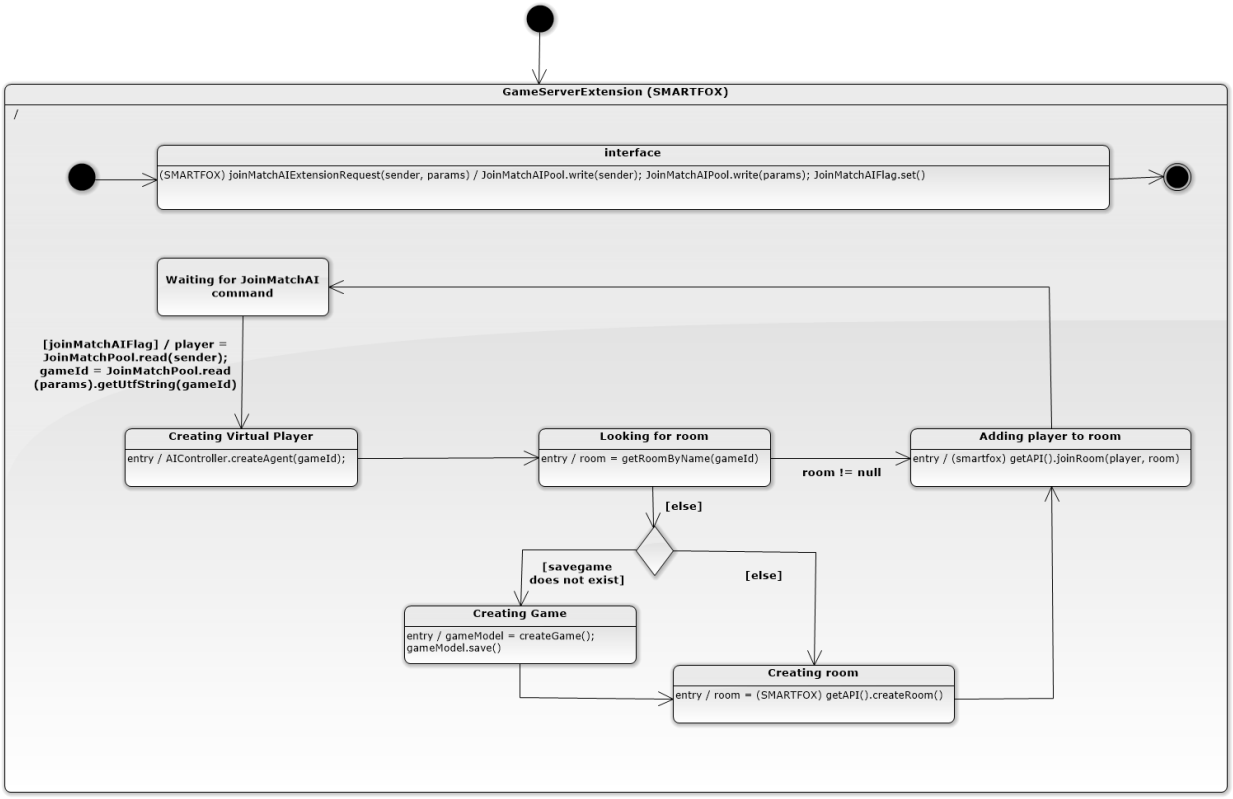
Inkomende Events:

- Het JoinMatch event treed op wanneer een gebruiker een JoinMatch commando stuurt.

Uitgaande Events:

- Wanneer een room beschikbaar is, zal de GameServerExtension een joinRoom event sturen naar de GameServerRoomExtension.

9.3.2 Het state transition diagram voor de “GameServerExtension” taak van de AI Module



Doelen:

- Het starten van een match in een room.
- Het toevoegen van spelers aan een room
- Het aanmaken van een virtuele speler die in dezelfde room de match kan spelen.

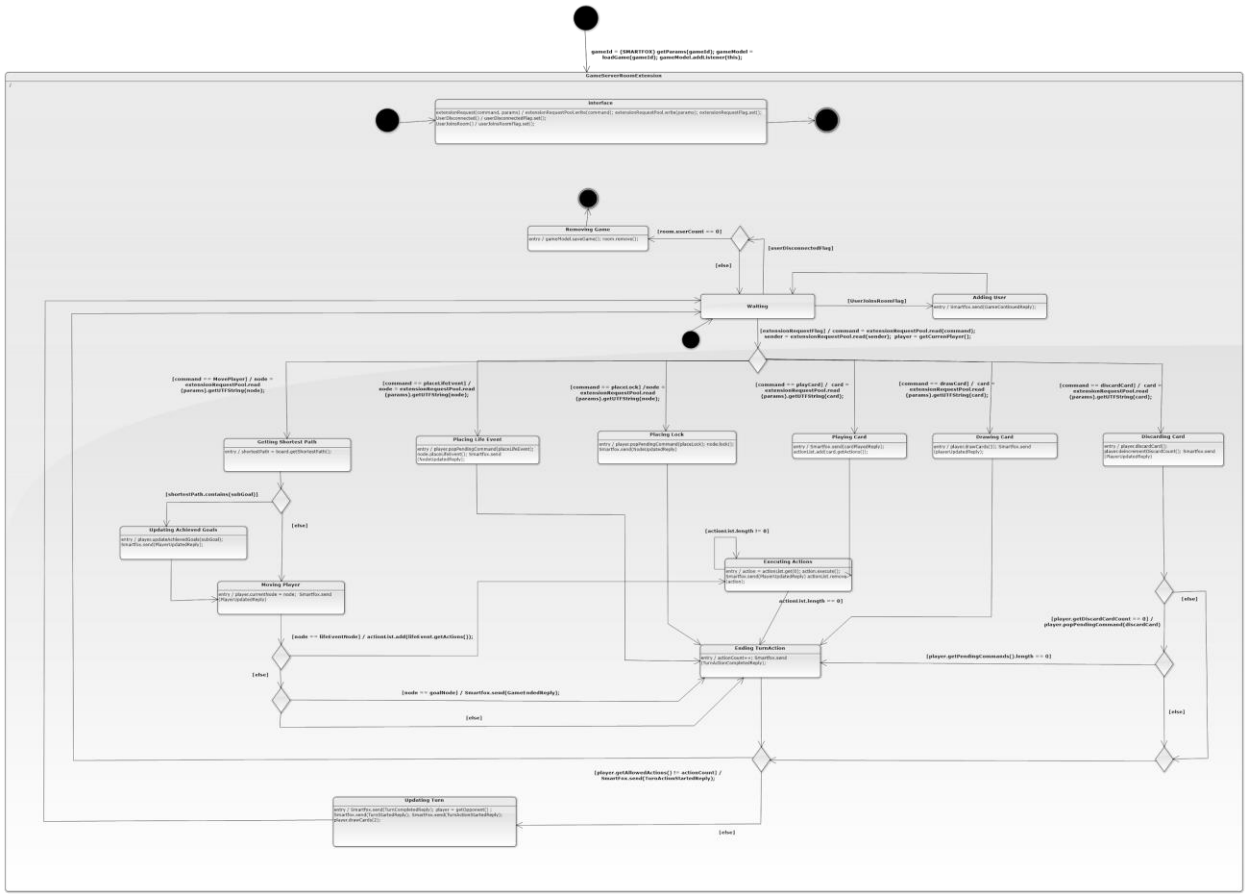
Inkomende Events;

- Het JoinMatchAI event treed op wanneer een gebruiker een JoinMatchAI commando stuurt.

Uitgaande Events:

- Wanneer een room beschikbaar is, zal de GameServerExtension een joinRoom event sturen naar de GameServerRoomExtension.
- Zodra het JoinMatchAI event op treed stuurt de GameServerExtension een createAgent event naar de AIController.

9.3.3 Het state transition diagram voor de “GameServerRoomExtension” taak van de server



Doelen:

- Het uitvoeren van commando’s van de gebruikers.
- Het sturen van berichten met updates van de match naar de gebruikers.
- Het sturen van de huidige staat van de match wanneer een room binnenkomt.
- Het opslaan van de game en afsluiten van de room wanneer de room leeg is.

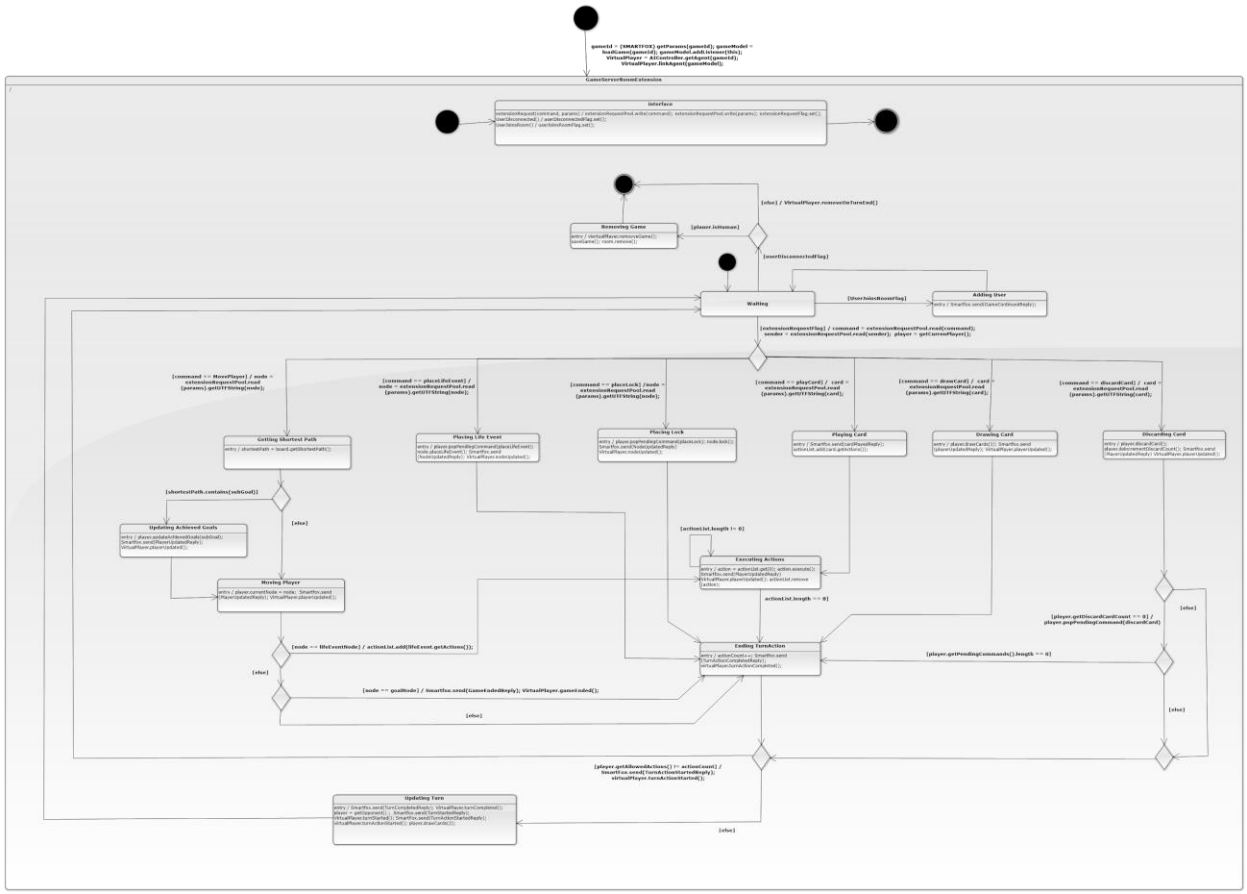
Inkomende Events;

- Het ExtensionRequest event treedt op wanneer een speler een spelcommando stuurt.
- Het UserJoinsRoom event treedt op wanneer een gebruiker de room binnenkomt.
- Het UserDisconnects event treedt op wanneer een gebruiker de room verlaat.

Uitgaande Events:

- Bij een verandering van een speler wordt een PlayerUpdated event gestuurd.
- Bij een verandering van een node wordt een NodeUpdated event gestuurd.
- Bij de verplaatsing van een speler wordt een MovePlayer event gestuurd.

9.3.4 Het state transition diagram voor de “GameServerRoomExtension” taak van de AI Module



Doelen:

- Het uitvoeren van commando’s van de gebruikers.
- Het activeren van de virtuele speler wanneer de room aangemaakt wordt.
- Het sturen van berichten met updates van de match naar de (virtuele) spelers.
- Het sturen van de huidige staat van de match wanneer een room binnenkomt.
- Het opslaan van de game en afsluiten van de room wanneer de room leeg is als de virtuele speler niet aan zet is.

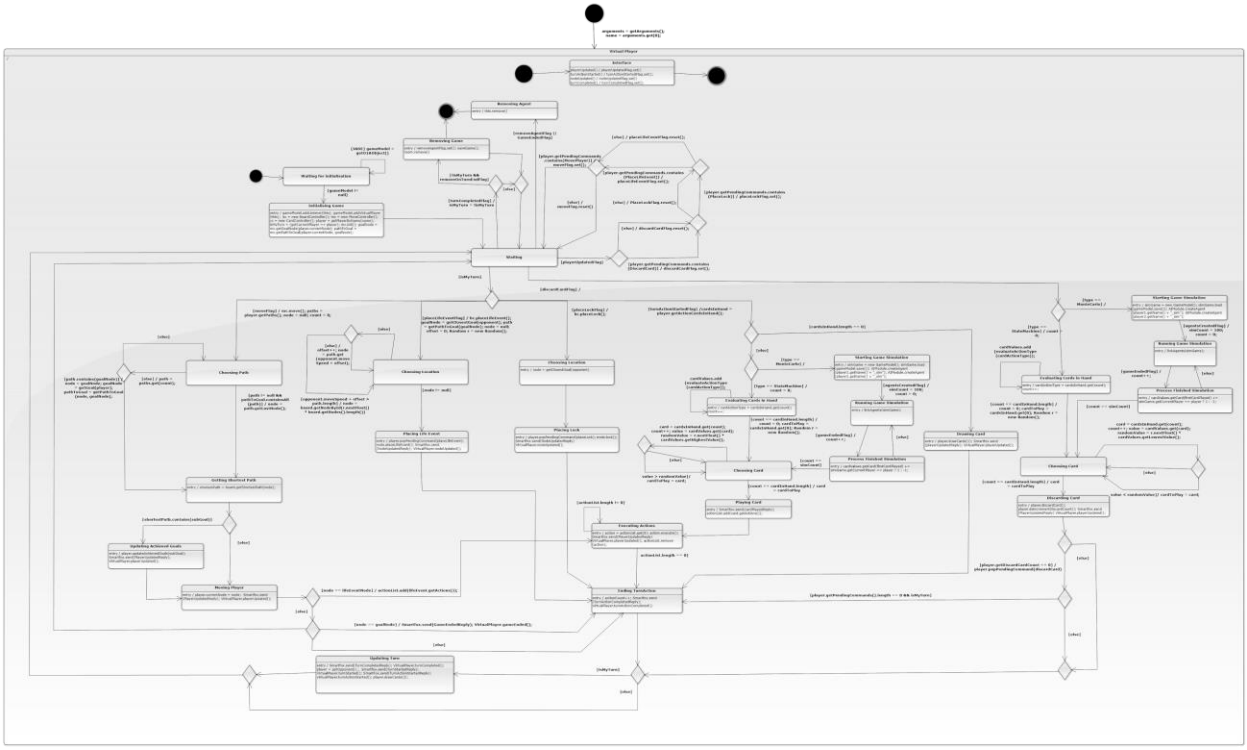
Inkomende Events;

- Het ExtensionRequest event treedt op wanneer een speler een spelcommando stuurt.
- Het UserJoinsRoom event treedt op wanneer een gebruiker de room binnenkomt.
- Het UserDisconnects event treedt op wanneer een gebruiker de room verlaat.

Uitgaande Events:

- Bij een verandering van een speler wordt een PlayerUpdated event gestuurd.
- Bij een verandering van een node wordt een NodeUpdated event gestuurd.
- Bij de verplaatsing van een speler wordt een MovePlayer event gestuurd.

9.3.5 Het state transition diagram voor de “VirtualPlayer” taak van de AI Module



Doelen:

- Het simuleren van commando’s voor de virtuele speler en deze uitvoeren.
- Het sturen van berichten met updates van de match naar de (virtuele) spelers.
- Het opslaan van de game en afsluiten van de room wanneer de room leeg is zodra de virtuele speler niet meer aan zet is.

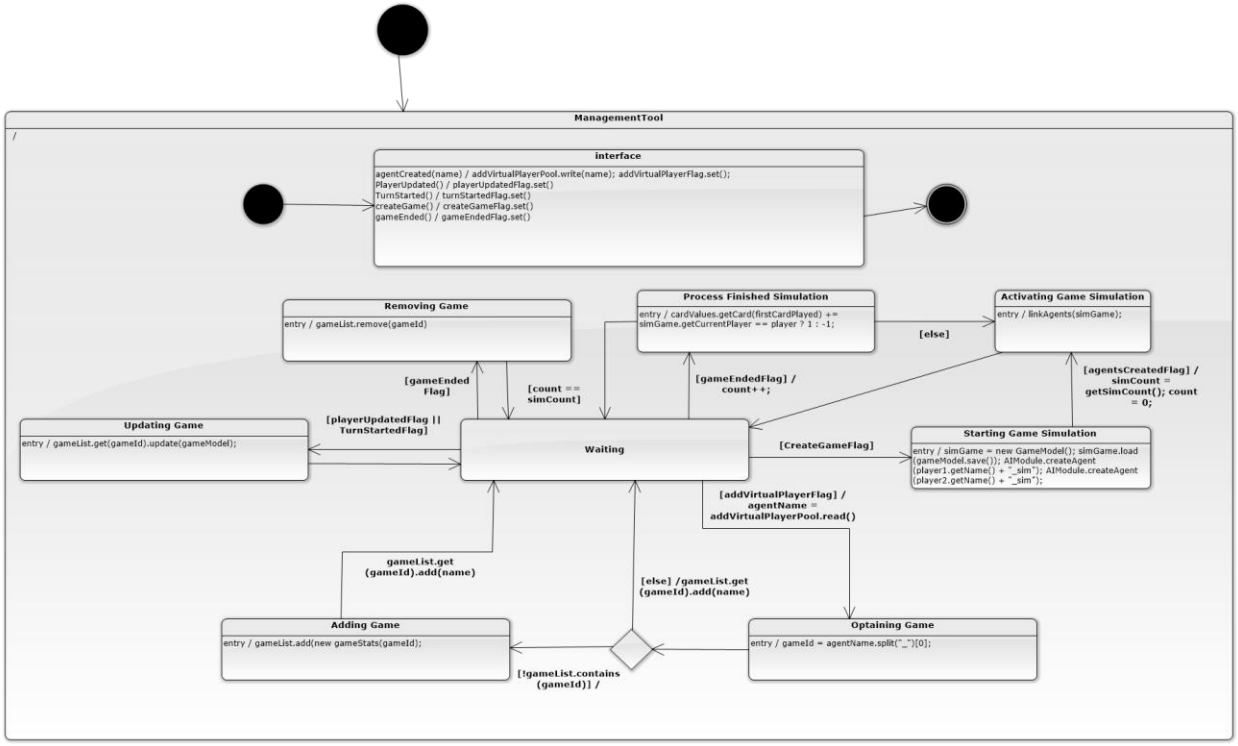
Inkomende Events;

- Het PlayerUpdated event treedt op wanneer een verandering van een speler plaatsvindt..
- Het NodeUpdated event treedt op wanneer een verandering van een node plaatsvindt.
- Het MovePlayer event treedt op wanneer een speler verplaatst wordt.

Uitgaande Events:

- Bij een verandering van een speler wordt een PlayerUpdated event gestuurd.
- Bij een verandering van een node wordt een NodeUpdated event gestuurd.
- Bij de verplaatsing van een speler wordt een MovePlayer event gestuurd.

9.3.6 Het state transition diagram voor de “ManagementTool” taak van de AI Module



Doelen:

- Het tonen van de statistieken van games met virtuele spelers.
- Het starten van game simulaties met twee virtuele spelers.

Inkomende Events

- Het AgentCreated event treedt op als een virtuele speler aangemaakt is.
- Het CreateGame event treedt op als de developer op de knop add game klikt.
- Het PlayerUpdated event treedt op wanneer een verandering van een speler plaatsvindt..
- Het TurnStarted event treedt op wanneer een beurt wordt overgedragen aan de andere speler.
- Het GameEnded event treedt op wanneer een spel is afgelopen.

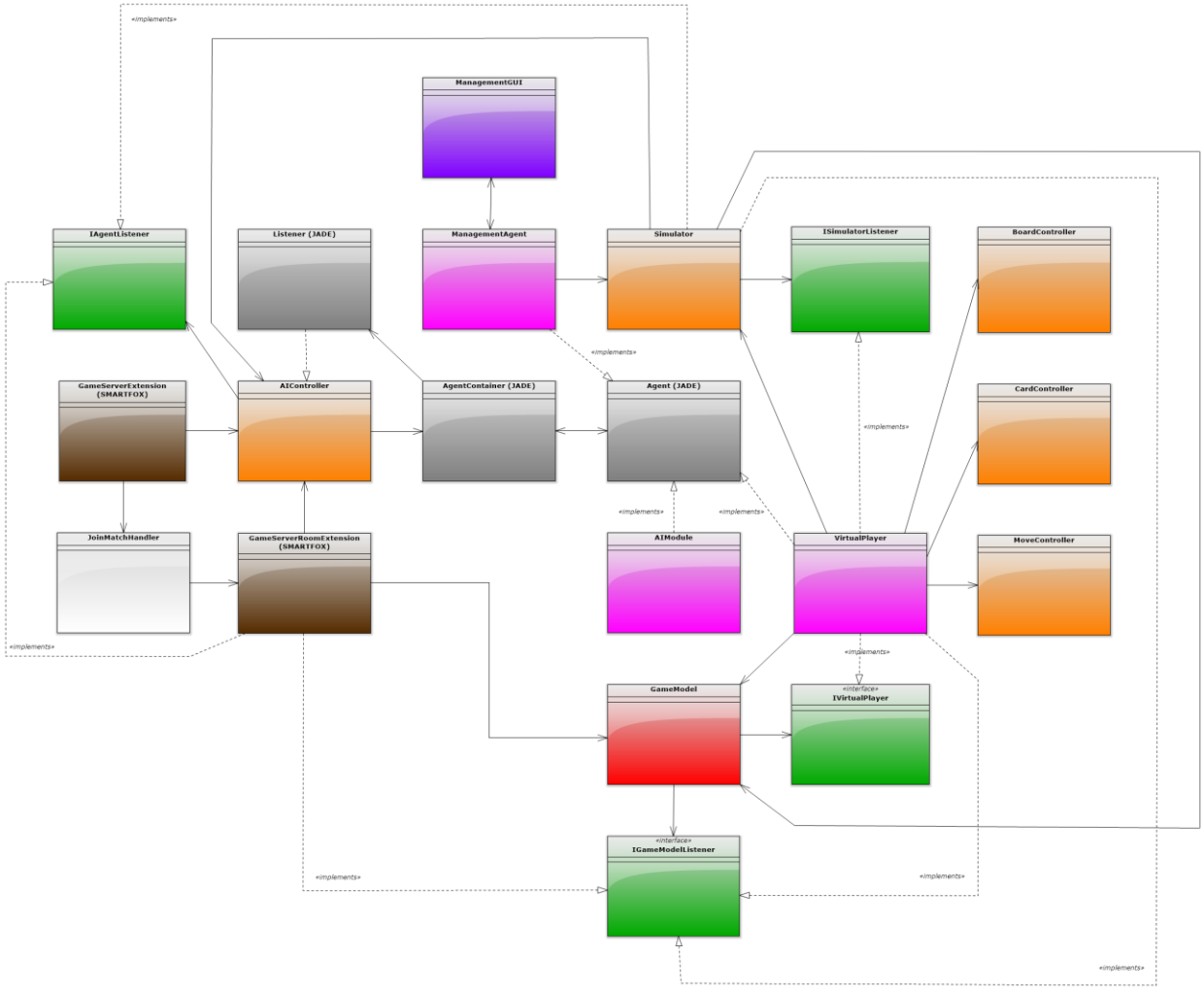
Uitgaande Events

- Het CreateAgent event wordt gestuurd op het moment dat een simulatie gestart wordt.

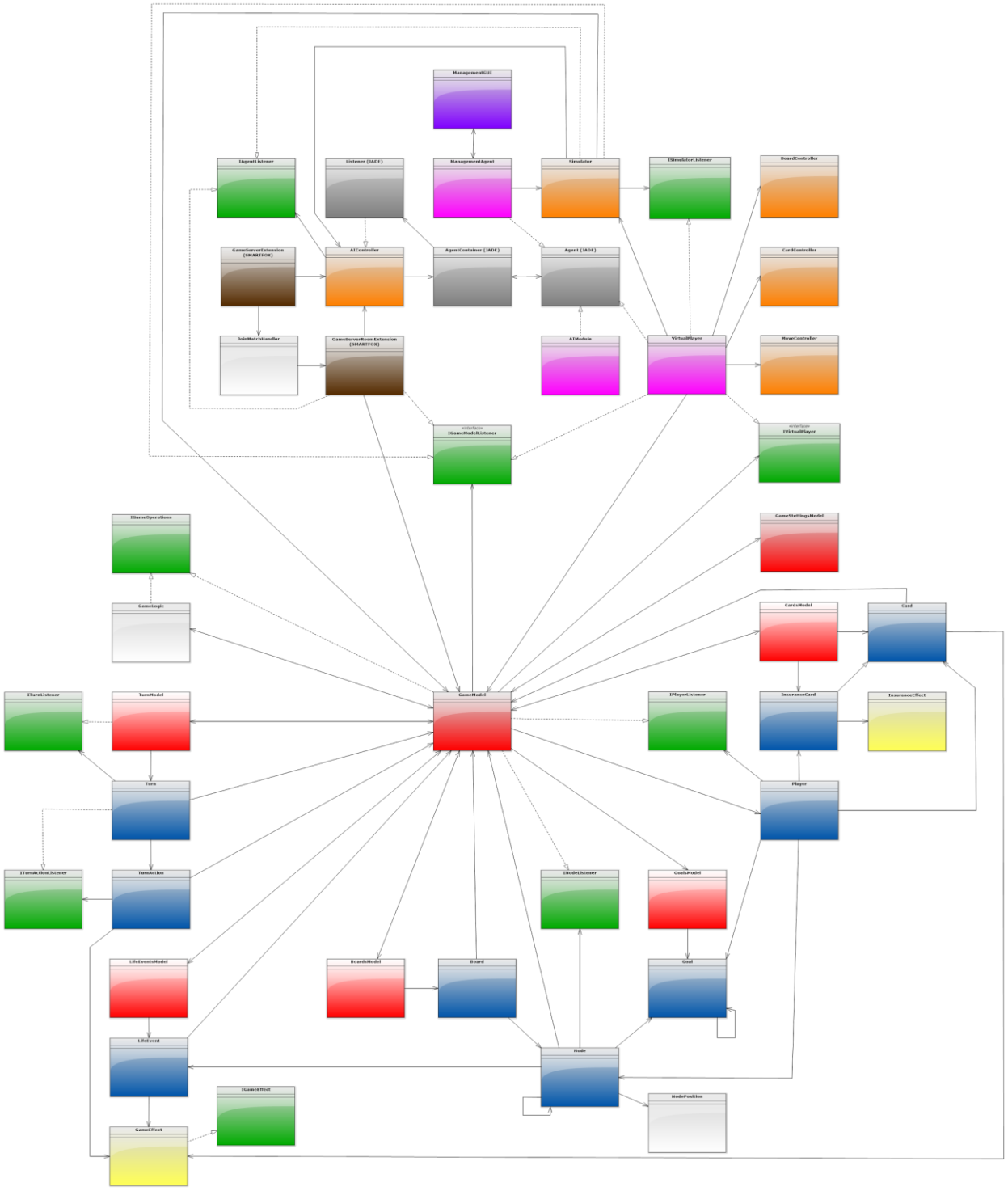
9.4 De complete klassen diagrammen

Het laatste deel van de solution architectuur is het in beeld brengen van een allesomvattend klassendiagram. Aan deze klassendiagrams zijn interfaces toegevoegd en laten zien welke klassen naar elkaar luisteren. Door middel van de interfaces kunnen verschillende taken met elkaar communiceren. In de functies van de interfaces worden flaggen gezet en pools gevuld waar de andere taak op zal reageren.

9.4.1 Het complete klassendiagram van de AI Module



9.4.2 Het complete klassendiagram voor de server en de AI Module



10 De strategieën en algorithmen

Tijdens het onderzoek zijn vier mogelijke strategieën en drie mogelijke pathfinding algorithmen naar voren gekomen. Hiervan zijn drie strategieen en twee algorithmen in de virtuele speler geimplementeerd. De verschillende strategieen hebben allemaal betrekking tot het spelen en/of weggooien van kaarten. De pathfinding algorithmen zorgen voor informatie over de te belopen weg naar het einddoel en dienen tevens als informatie bron voor het plaatsen van Locks en Life Events.

10.1 De strategieën

De geimplementeerde strategieën zijn: random, state machine en Monte Carlo. Een virtuele speler met random strategie speelt willekeurige kaarten uit zijn hand en gooit ook willekeurig kaarten weg. Een virtuele speler met state machine strategie besluit op basis van de staat van de heuristiek van het spel welke kaart hij gaat spelen of weggooien. Een virtuele speler met Monte Carlo strategie start een groot aantal simulaties van het spel op en houdt van de simulaties bij hoe vaak de eerst gespeelde kaart leidt tot het winnen of het verliezen van het spel. Vervolgens speelt hij de kaart met het grootste kans op winnen.

10.1.1 De state machine strategie

Een virtuele speler met state machine strategie speelt kaarten op basis van de heuristiek van het spel op dat moment. Voor elk type actie kaart in zijn hand wordt een waarde opgezet op basis van de factoren waar de waarden van dat tiype kaart op gebaseerd zijn. De virtuele speler kan zes typen actie kaarten in zijn hand hebben: Cards, Health, Move, Brainstorm, Lock en Surprise.

<u>Cards</u>

Kaarten met cards acties zullen de virtuele speler meer kaarten geven en soms laten weggooien of zullen ze de tegenstander kaarten geven of laten weggooien. Hierbij wordt gekeken naar hoeveel kaarten de virtuele speler en de tegenspeler in hun hand hebben.

Factor	Waarde
Standaard waarde	0.2
Aantal kaarten kleiner dan 3	4
Aantal kaarten groter dan 12	0
Aantal kaarten tegenspeler kleiner dan 3	huidige waarde keer -1 [1]

Health

Health kaarten geven de virtuele speler meer health en/of geven de tegenspeler meer of minder health. Bij het spelen van health kaarten wordt gekeken naar de huidige health van de virtuele speler en de tegenspeler, de plaats op het bord van de virtuele speler en of het de eerste actie van de beurt is.

Factor	Waarde
Standaard waarde	0.5
Health tegenspeler groter dan eigen health	1
Eigen health kleiner dan of gelijk aan 3	3
Het is de eerste actie van de beurt	huidige waarde +2
Een doel is net niet in bereik	10
Eigen health is gelijk aan 14	0
Heath van de tegenspeler is kleiner dan 4	huidige waarde keer -1 [1]

Move

Kaarten met move acties laten de virtuele speler over het bord bewegen om zo doelen te kunnen behalen. De factoren die meegenomen worden in de waarde van move kaarten zijn het aantal behaalde subdoelen van de virtuele speler en de tegenspeler, of het de tweede actie in de beurt is, de health van de virtuele speler, het aantal actieve verzekeringen en of een doel te behalen valt.

Factor	Waarde
Standaard waarde	1
Aantal behaalde subdoelen is kleiner dan aantal behaalde subdoelen van de tegenspeler	4
Aantal actieve verzekeringen groter dan 0	waarde +3
Als een doel binnen loopafstand is	10
Health kleiner dan 4	0

[1] als de waarde negatief is zullen de kaarten met negatieve gevolgen voor de tegenstander overgeslagen worden.

Brainstorm

Brainstorm geeft de virtuele speler vijf extra kaarten waarna hij vier kaarten moet weggooien. Het bijzondere aan deze kaart is dat de speler daarnaast ook een extra actie krijgt in de zelfde beurt. Aangezien deze kaart een betere versie is van een Cards kaart, is de waarde 1 hoger dan andere Cards kaarten.

Lock

Een lock plaatsen moet zorgen dat de tegenspeler zijn (sub)doel tijdelijk niet kan bereiken maar alleen wanneer de virtuele speler niet ook op weg is naar dat (sub)doel of dat de virtuele speler niet binnen twee beurten bij dat (sub)doel kan komen.

Factor	Waarde
Standaard waarde	0
Als alleen de tegenspeler opweg is naar dat doel en minder dan 5 stappen van het doel staat	5 min de afstand tussen de tegenspeler en het doel.
Als beide spelers opweg zijn naar het hetzelfde doel en de tegenspeler minder dan 5 stappen van het doel staat en de virtuele speler meer dan 1 keer lopen verder weg staat als de tegenspeler	

Surprise

Surprise geeft de virtuele speler de mogelijkheid om een life event op het bord te plaatsen. Aangezien deze vaak negatieve gevolgen hebben is het goed om deze te plaatsen op de plaats waar de virtuele speler denkt dat de tegenspeler heen zal gaan lopen.

Factor	Waarde
Standaard waarde	0
Op de plaats waar de tegenspeler waarschijnlijk heen gaat lopen kan een life event geplaatst worden.	3

Keuze van het type actie kaart

Een menselijke speler neemt niet altijd al de factoren mee in zijn besluit om een kaart te spelen. Om de virtuele speler in dat opzicht menselijker te maken is de keuze in de kaarten enigszins willekeurig gemaakt. Deste hoger de waarde van het type actie kaart deste groter de kans is dat de virtuele speler die actie kaart zal kiezen.

10.1.2 De monte Carlo strategie.

Een virtuele speler met Monte Carlo strategie start simulaties van het spel op het moment dat hij een volgende kaart moet gaan spelen om te besluiten welke kaart de virtuele speler gaat spelen. In de AI Management Tool kan aangegeven worden hoeveel simulaties de virtuele speler met Monte Carlo strategie start. De simulaties worden parallel uitgevoerd met een maximum van 20 spellen tegelijk. Dit maximum geld echter enkel voor de laptop waarop de AI Module is gemaakt en getest en zou mogelijk hoger gezet kunnen worden of lager gezet moeten worden op de server waar het spel op draait.

Deze strategie is in het prototype enkel geimplementeerd voor het spelen van kaarten. Het is niet geimplementeerd voor het weggooien van kaarten, voor het plaatsen van locks of life events en voor het besluit naar welk van de (sub)doelen de virtuele speler wilt gaan lopen.

Het spelen van kaarten heeft de grootste invloed op het spel. Door te testen of een virtuele speler met Monte Carlo strategie voor het spelen van kaarten voldoet aan de eisen die gesteld zijn in de requirements architectuur zal duidelijk worden of deze strategie uberhaupt in aanmerking komt om ingezet te worden als uitbreiding van het spel. Al de andere besluiten die gemaakt worden, worden op basis van de state machine strategie besloten.

<u>De simulaties</u>

Een simulatie van het spel is een spel dat gespeeld wordt door twee andere virtuele spelers die random kaarten spelen en weggooien. Het spel wordt gestart in de zelfde staat als het spel waar de virtuele speler met Monte Carlo strategie zich op dat moment in bevindt. Door virtuele speler houdt bij voor elke simulatie of de eerste kaart die in de simulatie gespeeld is leidt tot het winnen of verliezen van het spel. Wanneer al de spellen van de simulaties zijn beeindigd zal de virtuele speler met Monte Carlo strategie de kaart spelen die de hoogste win/lose ratio heeft. Aangezien het aantal keer dat in de simulaties een kaart gespeeld wordt kan verschillen per keer dat de Monte Carlo simulator de simulaties start, is hier al een willekeurig element ingebouwd.

De simulaties hebben echter ook een groot nadeel. Door dit spel te simuleren neemt de virtuele speler met Monte Carlo strategie soms ook onbekende factoren in het spel mee. De simulaties spelen bijvoorbeeld op een exact kopie van het bord. Wanneer een kaarten combinatie wordt gespeelt dat leidt tot een Life Event met een extra actie is de kans veel groter dat de virtuele speler wint.

10.2 De pathfinding algorithmen

Naast een strategie bevat de virtuele speler ook een pathfinding algorithme. Van dit algorithme zijn ook twee soorten geimplementeerd: Dijkstra en A*. Beide algorithmen zoeken naar het kortste pad van de huidige plek naar het gekozen doel. Voor het kiezen naar welk doel de virtuele speler gaat lopen zijn drie versies geimplementeerd, zie paragraaf 10.3.

10.2.1 Het Dijkstra algorithme
Het Dijkstra algorithme zoekt vanaf de huidige positie naar het doel door middel van het iteratief analyseren van de opgelagen posities en vervolgens de buren van die positie op te slaan om geanalyseert te worden. Het algorithme begint met het opslaan en analyseren van de huidige positie en zal stopppen wanneer het einddoel is gevonden.

Het nadeel van dit algorithme is dat het alle kanten tegelijk opzoekt. Elke positie op een bepaalde afstand van het begin wordt opgeslagen en geanalyseert en pas daarna worden posities met een grotere afstand van het begin bekeken.

10.2.2 Het A* algorithme
Het A* algorithme zoekt ook de snelste weg van de huidige positie naar het doel door posities te analyseren en de buren op te slaan, echter rangschikt dit algorithme de buren op de afstand tot het doel. Hierdoor krijgen de posities die dichter bij het doel liggen voorrang en zal de weg sneller gevonden worden.

10.3 De keuze van doelen
Het kiezen naar welk doel de virtuele speler loopt is een essentieel deel van het spelen van het spel. Zonder dit besluit zou de virtuele speler letterlijk doelloos over het bord gaan rondlopen, in de hoop dat hij toch bij (de juiste) doelen uitkomt. Voor het maken van deze keuze zijn drie versies geimplementeerd: random goal, closest goal en to endgoal. In al de versies van het maken van de keuze wordt eerst een analyse gedaan naar welk van de doelen op dat moment open staan zodat de random en closest goal keuzes niet direct naar het einddoel zou lopen zonder eerst naar de subdoelen te lopen.

10.3.1 De random goal keuze
De virtuele speler kiest een willekeurig doel uit de openstaande doelen.

10.3.2 De closest goal keuze
De virtuele speler kiest het doel die op dat moment het dichtst in de buurt is. Om de beslissingen van de closest goal en de to engoal keuzes menselijker te maken is hier een kleine kans ingebouwd dat een doel gekozen wordt dat net iets verder weg ligt

10.3.3 De to endgoal keuze
De virtuele speler berekent de totale afstanden van alle combinaties van subdoelen die naar het einddoel leiden en kiest vervolgens het eerste doel met de kortste afstand naar het einddoel. Ook hier is een kleine willekeurige factor toegevoegd om de beslissing menselijker te maken.

11 De testresultaten

Door de virtuele spelers met verschillende strategieen en algorithmen te testen kan ondervonden worden welke het beste het aan de constraints voldoet die in de requirements architectuur zijn opgezet. In dit hoofdstuk worden de resultaten van de tests van de constraints doorgenomen.

De constraints opgezet in de requirements architectuur zijn opgedeeld in twee groepen: accuracy en performance constraints. De accuracy constraints zijn getest aan de hand van het analyseren van de acties die de virtuele speler heeft gedaan tijdens het playtesten. De performance constraints zijn getest door de reactietijd van het systeem te analyseren tijdens het playtesten.

Het playtesten

De virtuele spelers zijn getest tijdens vijf playtesting sessies:

- State machine vs State machine
- State machine vs Monte Carlo
- Monte Carlo vs Monte Carlo
- Human vs State Machine*
- Human vs Monte Carlo*

Elke playtesting sessie bestaat uit tien gespeelde spellen tussen de twee virtuele spelers. Van deze tien spellen zijn steeds twee op elk van de vijf verschillende borden gespeeld en heeft de virtuele speler in vijf spellen de Dijkstra to endgoal pathfinding algorthme en de andere vijf keer de A* to endgoal pathfinding algorithme. De andere versies van de keuze van doelen zijn niet meegenomen in de playtesting sessies omdat de focus ligt op het testen van de verschillende strategieen en pathfinding algorithmen.

De playtesting sessies met een menselijke speler bestaan uit twaalf gespeelde spellen tegen een virtuele speler met state machine of monte carlo strategie en altijd een A* to endgoal pathfinding algorithme.

De gespeelde acties en reactietijden worden weergegeven in de AI Management Tool die gemaakt is statistieken van spellen te bekijken. Echter kunnen soms de statistieken alleen niet genoeg vertellen over of de gespeelde zet echt zinvol is geweest. Daarom is de voortgang van de spellen ook opgenomen en in de vorm van tekstbestanden opgeslagen. De inhoud van deze opnames, inclusief de gespeelde zetten en reactietijden van het spelen van kaarte voor spellen tussen twee virtuele spelers zijn te vinden in Bijlage 4: Testrapport.

* De 24 spellen waarin mensen gespeeld hebben tegen de virtuele spelers zijn op aanvraag beschikbaar mits het online bordspel dat nu nog in ontwikkeling is, is uitgebracht. De reden hiervoor dat de deze opnames informatie bevatten die vertrouwelijk is tot het spel uit. De aanvraag kan gestuurd worden naar d.haverhals@gmail.com.

11.1 De playtesting analyse

De virtuele speler bevat twee constraints getest door playtesting: doelgericht lopen over het bord en het spelen van zinvolle kaarten. Deze constraints zijn getest door het analyseren van de acties die gedaan zijn door virtuele spelers met verschillende strategieen en algorithmen in spellen die de virtuele spelers tegen elkaar gespeeld hebben.

11.1.1 Het doelgericht lopen over het bord (99%)

In de analyse van dertig gespeelde spellen tussen virtuele spelers met Dijkstra of A* pathfinding en state machine of Monte Carlo strategie is geen enkele keer te ondervinden dat een virtuele speler niet doelgericht over het bord loopt. Zowel het Dijkstra algorithme als het A* algorithme voldoen aan deze eis.

11.1.2 Het spelen van zinvolle kaarten (99%)

In de dertig gespeelde spellen hebben de opgenomen virtuele spelers samen meer dan 1000 kaarten gespeeld waarvan alle kaarten zinvol waren voor zowel de virtuele speler met state machine strategie als de virtuele speler met monte carlo strategie. Beide strategieen voldoen aan deze eis.

11.2 De tijd analyse

De performance constraints zijn deels getest door de opnames te analyseren en deels door de AI Management Tool zelf te analyseren. Het spelen van kaarten en de win/lose ratio is geanalyseert in de opnames en het bijwerken van de management tool en het aanmaken van de virtuele speler is getest door de management tool te analyseren.

11.2.1 Het spelen van een kaart (3 sec)

In de opnames van de spellen te zien dat de state machine virtuele speler altijd binnen 2 seconden een kaart speelt en voldoet hiermee aan de eis. De monte carlo virtuele speler speelt een kaart in ongeveer 10 tot 50 seconden en voldoet niet aan de eis.

11.2.2 Win/Lose ratio (30%)

Van de twaalf gespeelde spellen heeft de state machine virtuele speler vier keer van mensen gewonnen. Dit betekent dat met een 33% win/lose ratio de state machine virtuele speler aan de eis voldoet. In de twaalf gespeelde spellen tegen mensen heeft de virtuele speler met Monte Carlo strategie acht keer kunnen winnen. Met een 80% win/lose ratio voldoet deze dus ook aan de eis.

11.2.3 Het bijwerken van de management tool (1 sec)

De management tool loopt op een klok van 1 seconde. Elke keer dat de klok af gaat, wordt de management tool geupdate. De tijd van tussen de updates is te zien boven in de management tool en is altijd kleiner dan of precies gelijk aan 1 seconde.

11.2.4 Aanmaken en activeren van een virtuele speler (500 ms)

In de management tool is te zien dat de virtuele spelers aangemaakt worden in 1 tot 20 milliseconden. (zie Creation Time (ms))

12 De conclusie

Nu de software architectuur is opgezet, de AI module is gemaakt, de strategieën en algorithmen zijn geïmplementeerd en getest is het tussentijdse antwoord op de hoofdvraag beschreven in paragraaf 5.3 niet langer acuraat. Dit hoofdstuk richt zich op het herdefiniëren van de hoofd- en deelvragen met de nieuwe kennis die is opgedaan.

12.1 De deelvragen

Wat is een virtuele speler?

Een virtuele speler kan vele vormen aannemen maar is in het geval van het online bordspel een digitale tegenspeler die acties uitvoert op basis van beslissingen die gemaakt worden door algorithmen, simulaties en/of strategieen.

Wat voegt de virtuele speler toe aan de spelervaring?

De virtuele speler geeft het spel veel nieuwe gameplay mogelijkheden. De virtuele speler kan ingezet worden als waardige tegenspeler en als begeleider van nieuwe spelers die het spel spelenderwijs kunnen leren.

Welke functionele specificaties worden gesteld aan de virtuele speler?

In welke maten moet de virtuele speler intelligentie bevatten?

Om een uitdagend spel te bieden voor de speler is gebleken uit de tests dat een virtuele speler rekening moet houden met de bekende factoren in het spel en niet de onbekende factoren.

Vanuit welke basis moet de virtuele speler keuzes maken?

Uit de resultaten van de tests blijkt dat beslissingen maken op basis van de staat van het spel op dat moment de beste resultaten geeft. Hierbij is een willekeurige kans toegepast zodat de speler niet door heeft op welke basis de beslissing precies is gemaakt.

Welke technische specificaties worden gesteld aan de virtuele speler?

Hoe kan de virtuele speler toegepast worden in het bestaande spel?

Aangezien het spel op de server gespeeld wordt is er maar één optie mogelijk: aan de server kant.

Op welke manier verkrijgt de virtuele speler zijn kennis?

De testresultaten geven aan dat de state machine virtuele speler het best aan al de eisen die opgezet zijn in de requirements architectuur voldoet.

12.2 De hoofdvraag

Welke AI module moet worden ontworpen en ontwikkeld om virtuele spelers het bordspel tegen andere (virtuele) spelers op elk niveau te kunnen laten spelen met als focus de normale speler een consistent uitdagende spelervaring te geven, zonder dat de speler het idee heeft dat de virtuele speler zonder reden acties onderneemt of dat de virtuele speler vals speelt?

De AI Module die ontworpen en ontwikkeld is aan de server-side van het bestaande spel bevat virtuele spelers met twee soorten strategieen en twee pathfinding algorithmen. Uit de tests is gebleken dat de virtuele speler met state machine strategie het best aan de eisen van het systeem voldoet.

Daarnaast neemt deze virtuele speler altijd alleen de bekende factoren in het spel op dat moment mee in zijn beslissingen. Hierdoor zal de virtuele speler geen doelloze acties uit voeren en zal de tegenspeler nooit het idee hebben dat de virtuele speler vals speelt - mits de waarden van de type actie kaarten goed ingesteld zijn.

Wanneer de willekeurige kans op het spelen van kaarten kleiner gemaakt wordt naar mate het level van de tegenspeler hoger wordt groeit de virtuele speler met de tegenspeler mee. Op deze manier biedt de virtuele speler met state machine algorithme een consistent uitdagende spelervaring voor spelers op elk niveau.

13 De aanbevelingen

In de conclusie is duidelijk de aanbeveling naar voren gekomen voor het inzetten van de virtuele speler met state machine strategie.

Naast de keuze van strategie van de virtuele speler bevat de virtuele speler ook een pathfinding algorithme. Ondanks dat beide algorithmen aan alle eisen voldoen is de aanbeveling om te kiezen voor het A* algorithme.

Het A* algorithme heeft altijd een 1 : 1 relatie tussen de lengte van het uiteindelijke pad en de snelheid waarmee het wordt uitgevoerd waar het Dijkstra algorithme een exponentiele relatie tussen de lengte van het uiteindelijke pad en de snelheid waarmee het wordt uitgevoerd.

De virtuele speler kan op verschillende manieren gebruikt gaan worden als het ingezet gaat worden als uitbreiding op het bestaande spel. Mijn aanbeveling is om het niet enkel te houden op een tutorial modes. Een tutorial modus zal maar een klein deel van de kracht en de mogelijkheden die deze virtuele speler bevat gebruiken.

14 De evaluatie van de procesgang

Het proces van het maken van de AI Module en de verschillende soorten virtuele spelers is over het algemeen erg goed gelopen. De verschillende fasen zijn in het begin opgesteld en binnen de tijd van het project zijn al deze fasen doorlopen en hebben samen geleid tot een werkend eindproduct.

Mogelijk zou de oorspronkelijke planning (het waterval model) toch beter gepast hebben bij dit project. Door de loop van het proces is hierdoor soms door fasen heen gevlogen door tijdsgebrek. Op deze manier is meer tijd verloren gegeaan in latere fasen omdat de basis nog niet acuraat en compleet genoeg was opgezet.

De deadlines van de tussenproducten hebben aan de andere kant veel geholpen om de prioriteit van de fasen weer recht te zetten.

15 De bronvermeldingen

Billings, D; Davidson, A; Schaeffer, J; Szafron, D (2001) The Challenge of Poker
Department of Computing Science, University of Alberta, Edmonton, AB, Canada T6G 2H1
http://ac.els-cdn.com/S0004370201001308/1-s2.0-S0004370201001308-main.pdf?_tid=5acd627a-7ba3-11e4-a36f-00000aab0f01&acdnat=1417690406_568fe64462a1be0b5b9c16516d94a181

Szita, I; Chaslot, G; Spronck, P (2010) Monte-Carlo Tree Search in Settlers of Catan
Advances in Computer Games, Lecture Notes in Computer Science Volume 6048, p.21-32

Jansen, J.V.; Tollisen, R; (2014) An AI for dominion based on Monte-Carlo methods
University of Agder
http://brage.bibsys.no/xmlui/bitstream/handle/11250/221382/IKT-590-G%20-%202014%20-%20Spring%20-%20Master%27s%20thesis%20-%20Jon%20Vegard%20Jansen%20and%20Robin%20Tollisen.pdf?sequence=1&isAllowed=y