PROJECT STRATEGIC GAMEFRAMEWORK

Eindverslag

Arunan Sriskantharasa
Kas Feenema
Rafael van den Berg
Tom Broenink
Wouter Haakmeester
Groep 7

Titel	Project Strategic Gameframework
Auteurs	Arunan Sriskantharasa Kas Feenema Rafael van den Berg Tom Broenink Wouter Haakmeester
Plaats uitgave	Groningen
Datum voltooiing	19-04-2015
School	Hanze Hogeschool Groningen
Opleiding	Informatica
Docent	L. Bialek
Versie	1.0

Versie	Commentaar
0.1	Opzet document
0.2	Inleiding, samenvatting toegevoegd
0.3	Hoofdstukken toegevoegd
0.4	Inhoudsopgave
0.5	Bijlagen
0.6	Concept
1.0	Eindverslag

Voorwoord

In de periode van 19 maart tot en met 17 april is er erg veel werk verzet om dit project te laten slagen. In dit document is te lezen en te zien hoe het eindresultaat is geworden. Dit project was in opdracht van een groot verzekeringsbedrijf en is door ons in behandeling genomen. We zouden een onder houdbaar framework realiseren waarin twee games konden worden gespeeld: Tic-Tac-Toe en Othello

We zijn iedere dag druk bezig geweest om het project te laten slagen. We zijn erg benieuwd hoe ver wij in het toernooi komen met onze kunstmatige intelligentie en ook wat de opdrachtgever vind van de user interface.

Ook willen wij graag onze begeleider Lech Bialek bedanken voor de begeleiding van het project.

Samenvatting

Het project moest worden gerealiseerd in de periode 19 maart tot en met 17 april. De opdracht hield in dat we een Game Framework moesten realiseren waarin twee games konden worden gespeeld, namelijk Tic-Tac-Toe en Othello. Dit framework moest gemakkelijk uitbreidbaar zijn met meerdere games. De nieuwe games moeten gemakkelijk geïnstalleerd kunnen worden. Uiteindelijk is dit ons ook gelukt.

Na ontvangst van het project zijn we bij elkaar gaan zitten om het project door te spreken. We hebben de taken verdeeld. Er moest een plan van aanpak, requirements document, een functioneel ontwerp en een technisch ontwerp gemaakt worden. In deze ontwerpen moesten de use-cases, sequence-diagrammen en klassendiagram zijn verwerkt. In het plan van aanpak hebben wij bijvoorbeeld ook vastgelegd welke ontwikkelmethodiek wij voor dit project gingen gebruiken. Wij hebben als projectgroep gekozen voor de Scrum-methodiek omdat iedereen hier al enige ervaring mee had. Tijdens deze bespreking hebben we ook Tom Broenink benoemd als projectleider van dit project.

In de eerste twee weken hebben we de taken voor het programmeren ook verdeeld. Eén Iemand ging het framework opzetten zodat we connectie konden maken met de server. De anderen gingen zich bezig houden met de spellen en de kunstmatige intelligentie. We zijn met Tic-Tac-Toe begonnen omdat deze ons het makkelijkste leek en we deze ook al eens eerder hadden gemaakt.

Na eerste twee weken bleek dat de kunstmatige intelligentie niet helemaal goed werkte. Hij gaf ons de slechtste move terug in plaats van de beste. Dit had te maken met een integer overflow. Nadat we ons er allemaal mee hadden bemoeid, was het probleem uiteindelijk toch opgelost en werkte de kunstmatige intelligentie naar behoren; hij gaf ons de beste move terug. Tijdens dit proces zijn we op de achtergrond ook alvast gestart met het bouwen van Othello. We kwamen erachter dat Othello toch een stuk lastiger bleek dan gedacht.

In de resterende weken hebben we de documentatie en de kunstmatige intelligentie van Othello afgemaakt. In de laatste week is de user interface voor de gebruiker ook nog wat veranderd. We vonden Swing niet zo heel goed, het werkte ook niet goed met OS X computers. We zijn op zoek gegaan naar een mooiere interface voor de gebruiker en hebben die gevonden en ook geïmplementeerd.

Wij vinden dat we als projectgroep goed hebben samengewerkt en het project tot een succesvol eind hebben gebracht. Het enige onderdeel die we nog wilden toevoegen was de tic-tac-toe AI, helaas zijn we hier niet meer aan toe gekomen.

Inhoud

Inleiding

Voor een groot verzekeringsbedrijf zijn wij, Doutzen Kroes Solutions (DK Solutions), gevraagd om een game framework te ontwikkelen waar gemakkelijk tweepersoons bordspelletjes op te spelen zijn. De spellen moesten gespeeld kunnen worden als speler-speler, speler-computer of computer-computer. Het verzekeringsbedrijf wil de spellen het game framework gebruiken bij hun reclamecampagne onder het motto ”Pijnig uw hersens, maar bezuinig niet op de polis”. Als dit een succes wordt willen ze in de toekomst ook andere bordspelletjes beschikbaar stellen op het zelfde platform. Herbruikbaarheid van het game framework is dan ook van groot belang.

Vooruitblik

In de inleiding is de omschrijving van het project te vinden. In hoofdstuk één worden de requirements die bij het project horen besproken en omschreven. In hoofdstuk twee wordt de architectuur van het framework besproken. In hoofdstuk drie wordt het ontwerp van het project besproken. Hierbij kun je denken aan de user interface, maar ook aan de kunstmatige intelligentie.

In hoofdstuk vier wordt besproken hoe we getest hebben gedurende het project. In hoofdstuk zes, het laatste hoofdstuk, wordt er een conclusie en de evaluatie van het proces besproken.

De bijlagen behorende bij dit document bestaan uit:

- Plan van Aanpak
- Klassendiagram
- Use-Cases
- Spel handleiding

Hoofdstuk 1 – Requirements

1.1 Functionele requirements

1.1.1 Gebruikers inloggen (Must have)

Een gebruiker kan inloggen op de server met een eigen gekozen gebruikersnaam. Na dat de speler is ingelogd is de speler zichtbaar voor andere spelers om er spellen tegen te gaan spelen.

1.1.2 Gebruikers uitloggen (Must have)

Een gebruiker kan nadat hij geen spellen meer wilt spelen uitloggen in het systeem. Na dat de speler is uitgelogd kan de speler niet meer worden uitgedaagd om een spel te spelen door andere spelers. De speler wordt teruggestuurd naar het hoofdscherm waar de speler zich weer opnieuw kan inloggen.

1.1.3 Spel starten (Must have)

Nadat de speler zich heeft ingelogd kan de speler een spel starten tegen een andere speler. In een drop down staan alle andere ingelogd spelers. Door een speler aan te klikken en op de knop 'Spel starten' te drukken wordt er een spel gestart tegen de gekozen speler.

1.1.4 AI Mens (Should have)

Als een spel gespeeld wordt kan er gekozen worden of de AI de mens of de computer is. Als er gekozen wordt voor de computer wordt het spel automatisch gespeeld. Als er wordt gekozen voor de AI mens speelt de speler zelf het zelf. Als je uitgedaagd wordt tot een spel krijg je een scherm waarin je kunt selecteren welke AI je wilt gebruiken.

1.1.5 Tic-tac-toe (Should have)

Als er een speler uitgekozen is kan het spel Tic Tac Toe gespeeld worden. Er verschijnt een speelveld met 9 vakken. De spelers kunnen een keuze maken uit één van deze vakken als set. Als één van de spelers drie rondjes of drie kruisjes op een rij heeft, heeft de desbetreffende persoon gewonnen. In het spel zijn er een viertal acties: gewonnen, verloren, gelijkspel en andere speler is aan zet. Het bord moet de juiste acties weergeven die speler doet.

1.1.6 Othello (Should have)

Als er een speler uitgekozen is kan het spel Othello gespeeld worden. Er verschijnt een speelveld van 64 vakken met in het midden twee zwarte en twee rondjes. De spelers kunnen het spel spelen door zwarte en witte rondjes te plaatsen in de daarvoor bestemde vakken. Bij dit spel zijn er ook vier acties mogelijk: gewonnen, verloren, gelijkspel of de andere speler is aan zet. Het bord moet de juiste acties weergeven die spelers doen. Zwart begint altijd met de eerste set, als er een set geplaatst kan worden is dat verplicht. Als je geen set kunt doen moet je passen en gaat de beurt naar de andere tegenstander.

1.2 Niet functionele requirements

Niet functionele requirements zijn eigenlijk requirements die ingaan op de performance van het systeem.

1.2.1 Beschikbaarheid (Must have)

De beschikbaarheid is in deze situatie lastig. De server wordt namelijk gehost door het verzekeringsbedrijf. Het verzekeringsbedrijf is er dus verantwoordelijk voor dat de server online blijft. Doordat het spel echt gespeeld wordt op de computer thuis van een speler en vanaf daar communiceert met de server is het ook lastig aan te geven hoe de beschikbaarheid is van de spelers. Dit hangt af van de computer van de klanten die het spel gaan spelen. Om daadwerkelijk de spellen te kunnen spelen is het erg belangrijk dat de server in ieder geval online blijft.

1.2.2 Schaalbaarheid (Must have)

Op de server worden alle verbindingen tussen de spelers tot stand gebracht. Alle communicatie verloopt dus op de server. Het aantal maximale verbinden die server aan kan, ook wel de schaalbaarheid genoemd, is de afhankelijk van het systeem waarop de klant de server online houdt. Bij een snellere server kan de server ook meer verbindingen aan. Op een minder snel systeem kan de server uiteraard minder verbindingen aan.

1.2.3 Stabiliteit (Should have)

Het spelen van de games zou stabiel moeten werken. Als er eventueel is mis gaat bij het spelen van het spel, komt er passende foutmelding die door het systeem wordt afgehandeld. Ook als er iets mis gaat tussen client en server zal er een passende foutmelding worden getoond.

1.2.4 Betrouwbaarheid (Should have)

De betrouwbaarheid is niets minder dan dat het systeem stabiel is en niet crasht. We kunnen als DK Solutions geen 100% garantie afgeven dat het systeem altijd werkt. Het systeem is uitvoerig getest om te zorgen dat het hele systeem goed werkt. Bij een crash mag de tijd dus up en down time zo kort mogelijk zijn. Mocht er een crash voorkomen dan kan men contact opnemen met DK Solutions.

1.2.5 Beveiliging (Must have)

In de huidige tijd van privacy van persoonsgegevens is het erg belangrijk dat er geen persoonsgegevens in verkeerde handen vallen. Door dat er voor het spel alleen een gebruikersnaam hoeft ingevuld te worden kunnen er geen privacygevoelige informatie gestolen worden.

1.2.6 Documentatie (Must have)

Voor de spellen die er gespeeld kan worden moet er wel duidelijk zijn hoe de spellen gespeeld moeten worden. Tevens is het de bedoeling van de opdrachtgever dat de spelen relatief eenvoudig gespeeld kunnen worden. Als men er als speler nog niet uitkomt, is er ons verslag een handleiding opgenomen tot het spelen van het spel met de desbetreffende spelregels. De documentatie kan eventueel ook zelf verschaft worden door de verzekeringsmaatschappij aan de klanten.

Hoofdstuk 2 – Architectuur

2.1 Packages

In de applicatie die wij gebouwd hebben zijn een viertal packages terug te vinden, namelijk:
- AI
- Framework
- Game
- Server

We hebben voor deze vier packages gekozen omdat dit de hoofdonderwerpen zijn waarin het project verdeeld kon worden. Zo konden wij goed overzicht houden welke classes waar moesten worden aangemaakt.
In de AI package hebben wij een abstracte klasse en de twee AI klassen voor de twee spellen staan. De abstracte AI klasse is gemaakt zodat het gemakkelijk is om voor andere spellen ook AI's toe te kunnen toevoegen. In deze klasse zijn de spelers, je tegenstander en het aantal hokjes van het spel al gedefinieerd.

In de Framework package zitten alle klassen die te maken heeft met ons specifieke framework. Zoals de schermen om in te loggen, de lobby waarin je de spelers ziet die online zijn op de server, maar ook alle knoppen waar je op kunt drukken in de lobby en als je een spel aan het spelen bent.

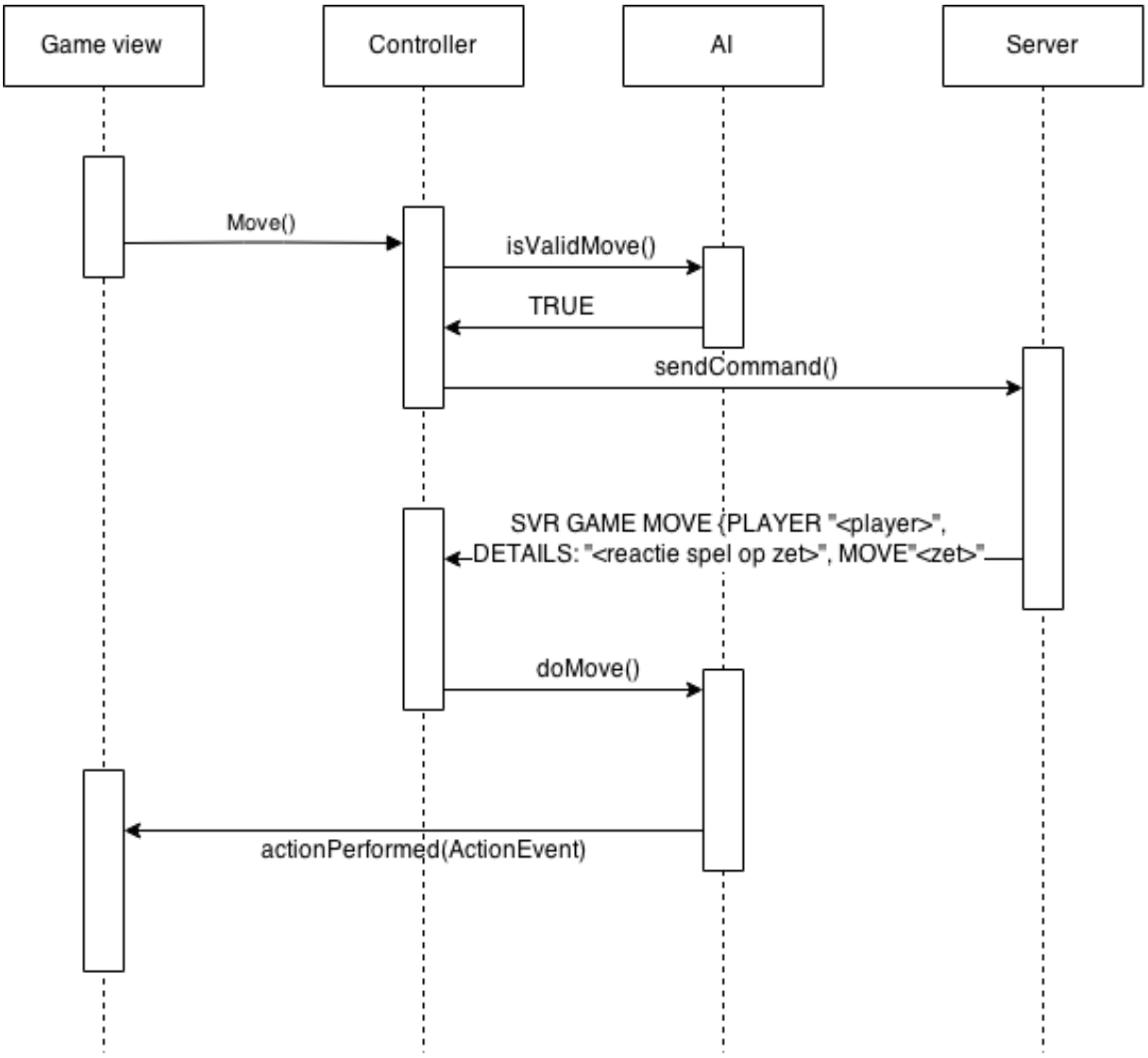
In de Game package zijn twee classes gemaakt. In de GameView class zitten de methodes om het bord te maken, het bord te updaten, de buttons toe te voegen, het bord te legen na dat een spel voorbij is en de hint functie die wij hebben ingebouwd. Wij draaien de AI altijd mee op de achtergrond. Als wij op de hint button klikken wordt een vakje op een bord een andere kleur. Dit vakje wordt door de AI bepaald als meest waardevolle zet op dat moment.

In de Server package zitten alle functionaliteiten die te maken hebben met de verbinding naar de server. Ook alles wat de server ons doorstuurt of wat wij naar de server doorsturen gaat via deze classes. Denk daarbij aan het doorgeven van een zet die je doet op het bord of om bijvoorbeeld de playerlist op te halen.

2.2 Klassendiagram

Het klassendiagram van dit project is te vinden in bijlage III

2.3 Sequence diagram



Versturen van een commando (in dit voorbeeld is het move commando uitgevoerd)

Voor het sturen van een commando wordt er in de game view een commando verstuurd. Deze view roept een functie aan van de gekoppelde controller. De controller controleert aan de hand van de ingeladen AI of het een legale zet is. Zo ja, dan wordt er TRUE teruggestuurd naar de controller die vervolgens een command aanmaakt om verstuurd te worden naar de server. Wanneer de server het command heeft ontvangen, dan wordt dit verwerkt door de server en een antwoord terug gestuurd naar de controller. De controller handelt dit af op de model. De model notificeert vervolgens de gekoppelde views met een actionPerformed call

2.4 Versiebeheer

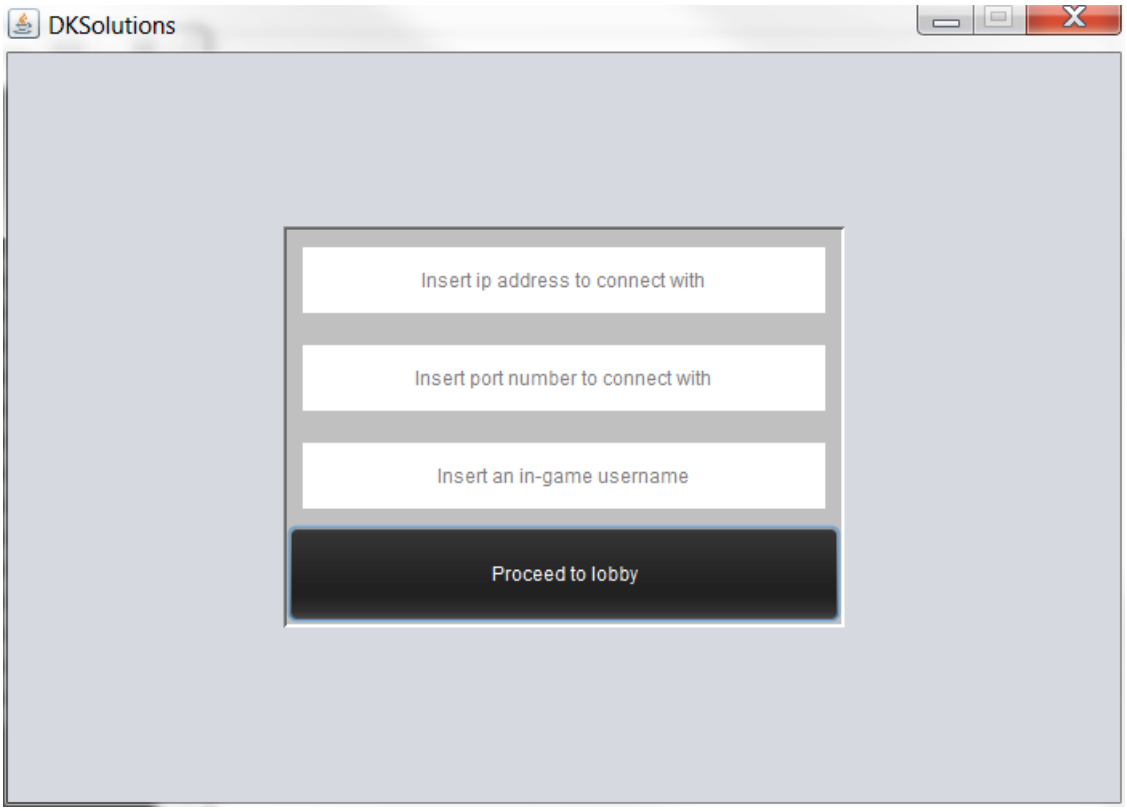
Wij hebben tijdens dit project gebruik gemaakt van git. Alle projectgenoten hadden al eerder gewerkt met git en vonden dit beter werken dan SVN. Doordat we gewerkt hebben met git hebben we goede versiebeheer kunnen toepassen. We hadden bijvoorbeeld wat wijzigingen gedaan maar die bleken later toch niet goed te zijn. We konden op deze manier gemakkelijk terugvallen op een vorige commit.

Hoofdstuk 3 – Ontwerp

3.1 Beschrijving User Interface

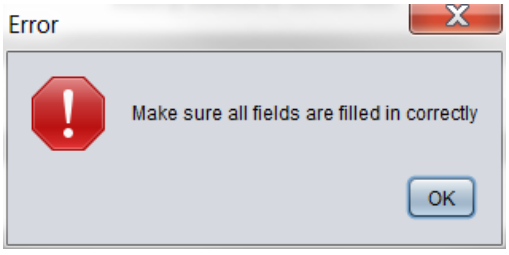
3.1.1 Inlogscherm

Als de applicatie wordt opgestart kom je op het login scherm terecht.



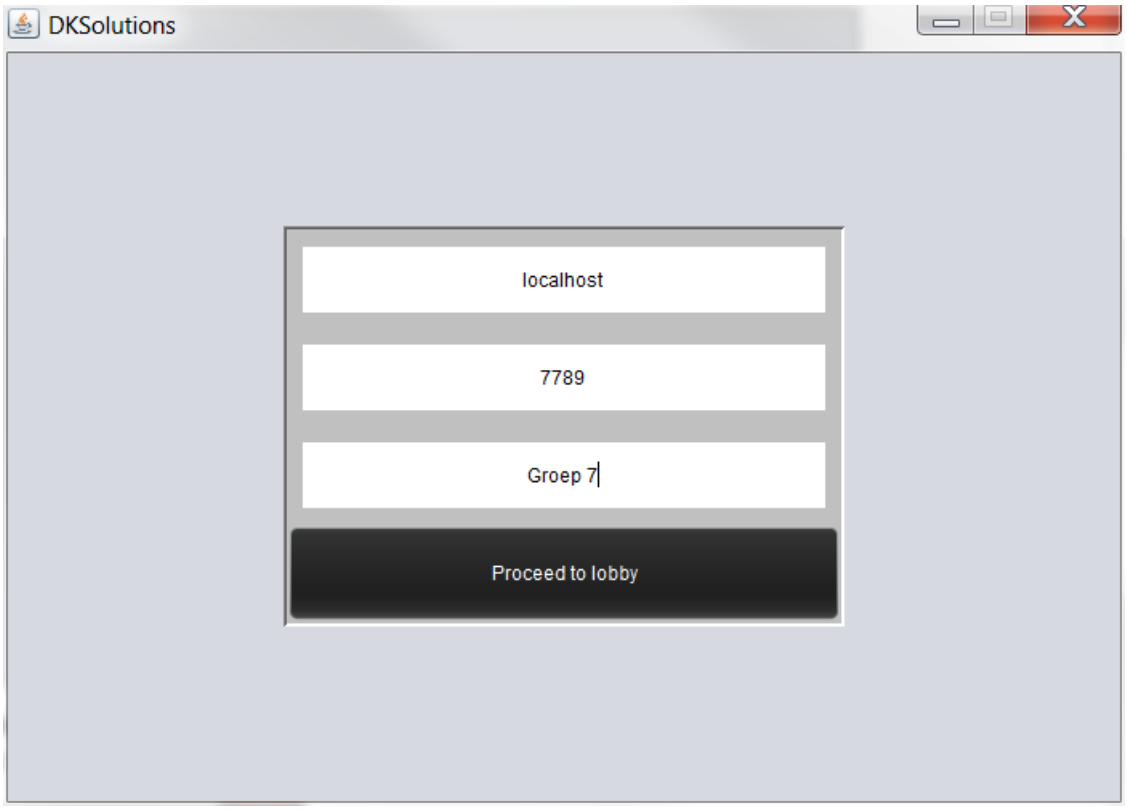
screenshot 1

In de tekst vakken staat wat er ingevuld moet worden om in te loggen. Er is een error handling aanwezig als 1 van de vakken bijvoorbeeld niet ingevuld is (screenshot 2).



screenshot 2

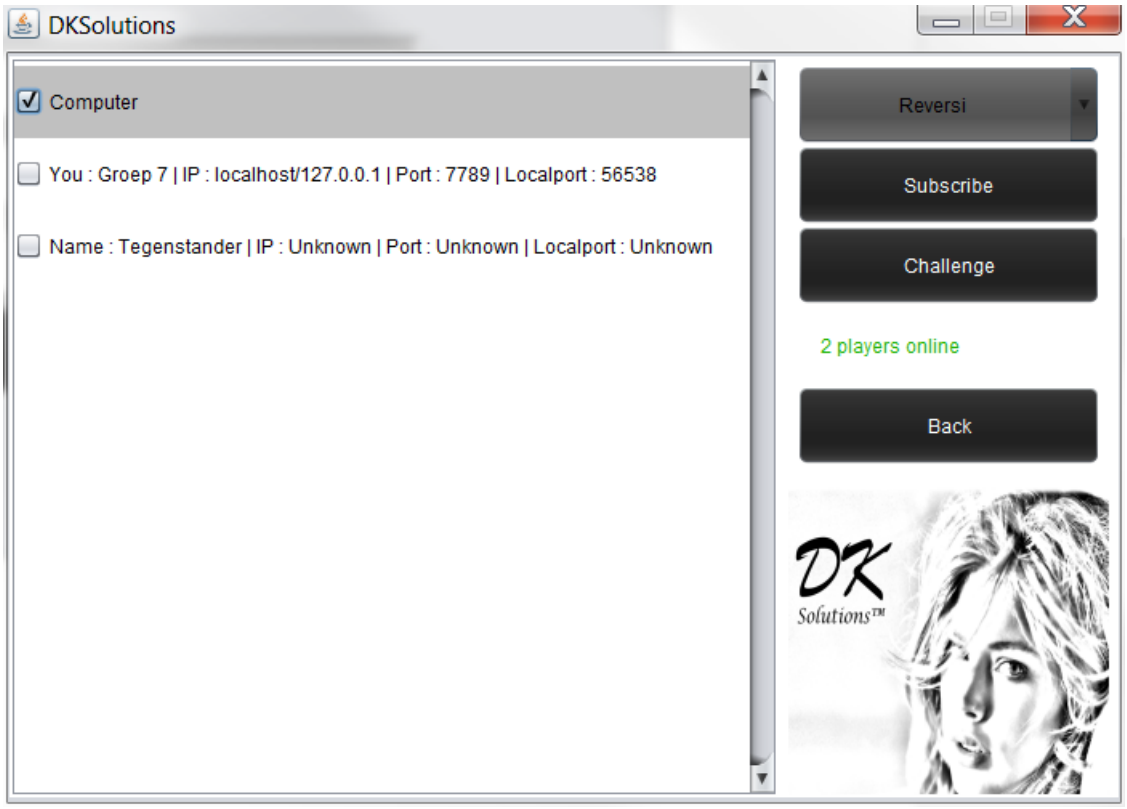
Als alles ingevuld is kan er op “Proceed to lobby” worden geklikt om door te gaan. Er is voor deze interface gekozen om het simpel te houden en zodat het is duidelijk wat er gedaan moet worden om naar de lobby te gaan. Screenshot 3 laat zien hoe het volledig ingevulde scherm eruit kan zien.



screenshot 3

3.1.2 Lobby

In de lobby staan aan de linkerkant alle spelers weergeven. Aan de rechterkant staan de buttons inclusief het aantal spelers online, en het DK Solutions logo.



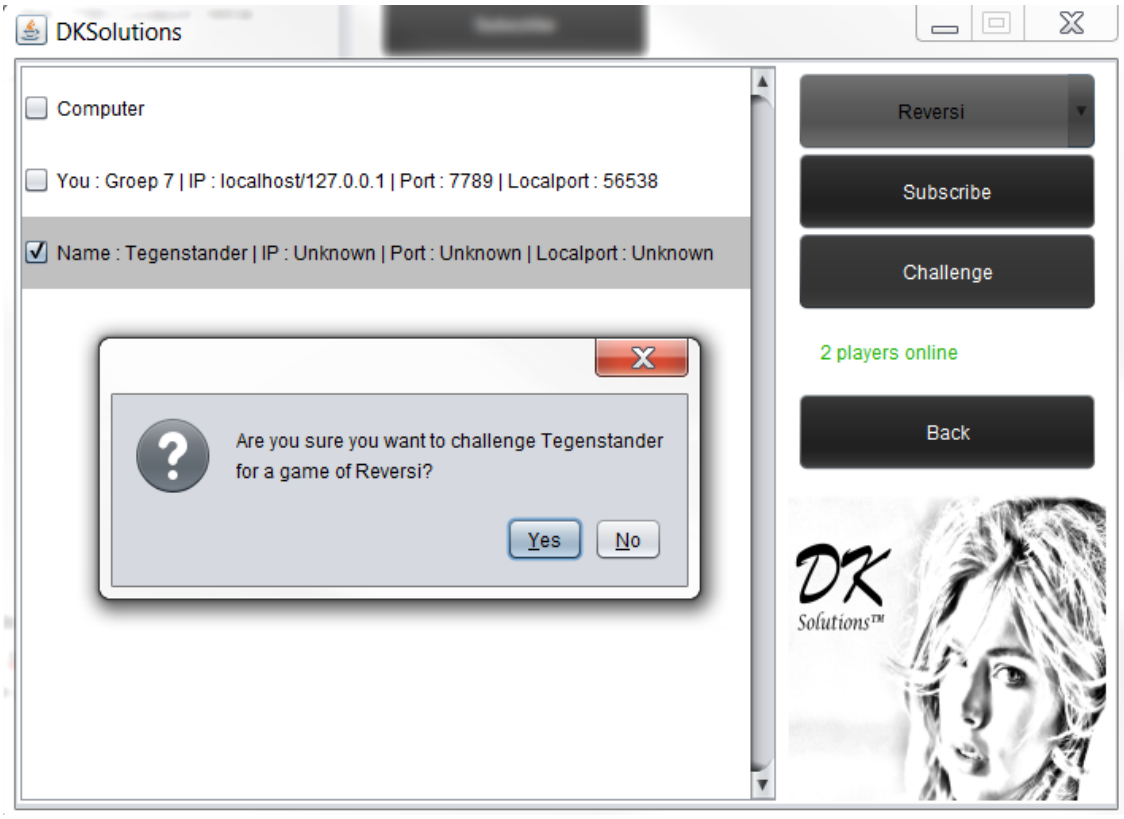
screenshot 4

De bovenste knop waar Reversi in staat is een drop-down menu met daarin alle games die gespeeld kunnen worden.

screenshot 5

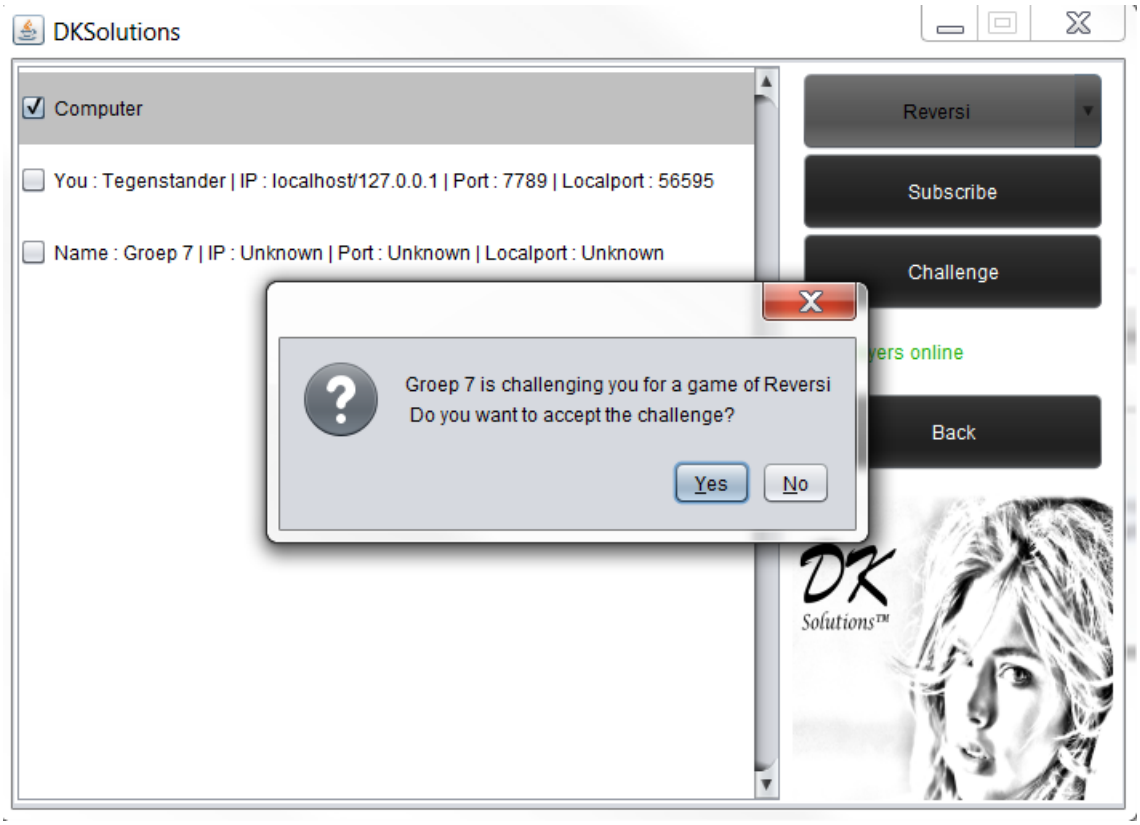
Met de Subscribe button is het mogelijk om jezelf toe te voegen aan een game. Als iemand anders in de lobby zich ook subscribed voor die game dan speel je automatisch tegen elkaar zonder elkaar uit te hoeven dagen.

De Challenge button is om elkaar uit te kunnen dagen. Je kunt een speler selecteren en vervolgens op Challenge klikken om deze speler uit te dagen voor de geselecteerde game. Er verschijnt dan een pop-up met de vraag of je zeker weet dat je de geselecteerde speler wilt uitdagen.



screenshot 6

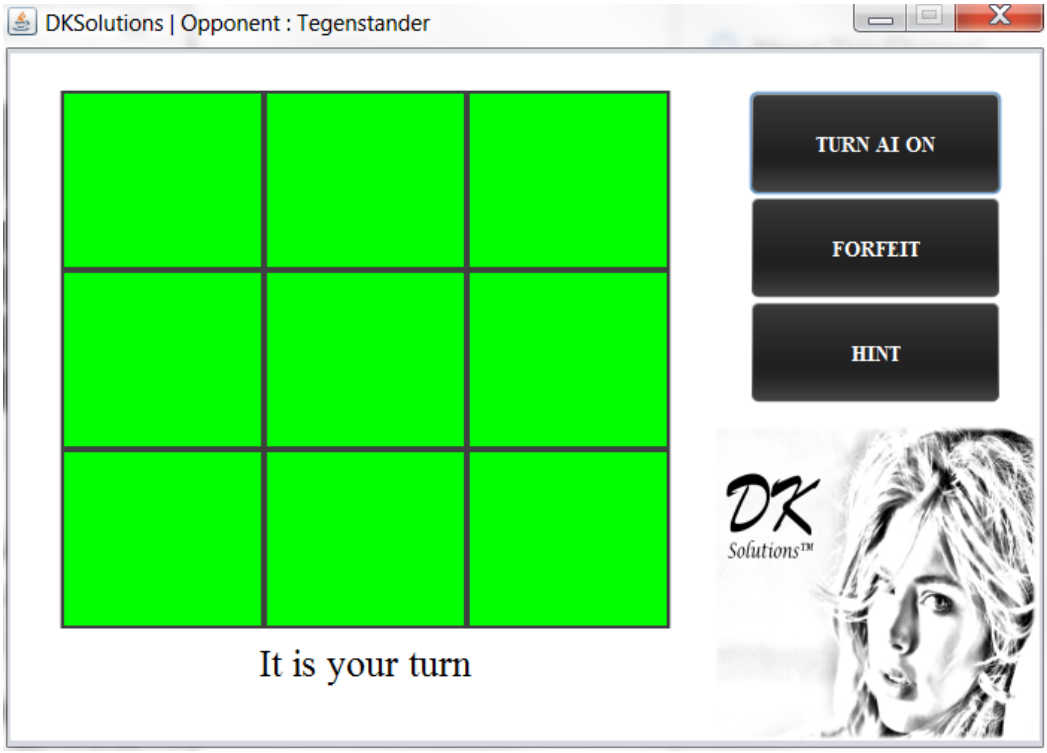
De uitgedaagde tegenstander krijgt dan een pop-up of hij de uitdaging wil accepteren. Te zien is wie jou uitdaagt en voor welk spel de uitdaging is. Als de uitdaging wordt geaccepteerd wordt het spel scherm weergeven, anders blijven de spelers in de lobby.



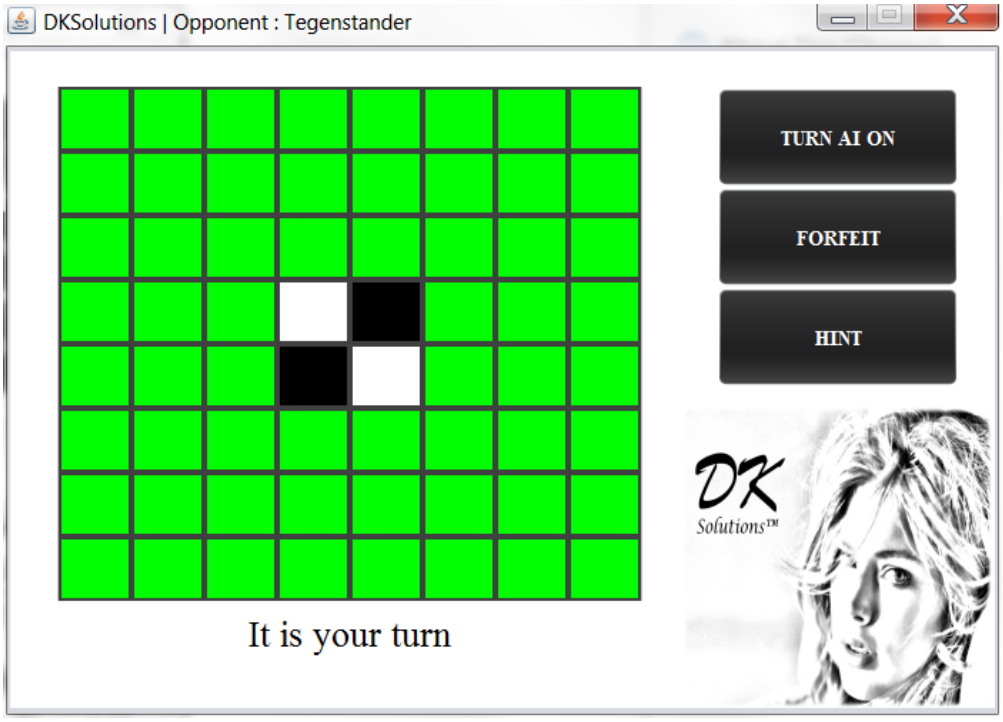
screenshot 7

3.1.3 Game scherm

Als de spelers Tic-Tac-Toe spelen ziet het scherm er uit zoals in screenshot 8. In screenshot 9 is de Reversi Gameview te zien. Deze is precies hetzelfde als de Tc-Tac-Toe View, maar dan met een speelbord van 8 bij 8 in plaats van 3 bij 3. Bovenin staat wie jouw tegenstander is, links is het speelscherm en rechts enkele buttons. De buttons die als requirement zijn opgegeven waren de AI en de forfeit. Het is hier mogelijk om de AI aan of uit te zetten. Dit kan pas als jij aan zet bent, want zodra de AI wordt aangezet speelt die ook gelijk een zet. Zelf is er een “Hint” button geïmplementeerd.

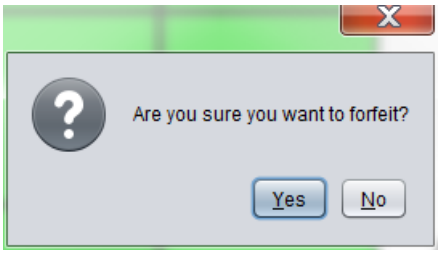


screenshot 8



screenshot 9

Met de forfeit button kan je opgeven. Als deze wordt ingedrukt dan komt er een pop-up met de vraag of je zeker weet dat je wilt opgeven. Als je op ‘Yes’ klikt dan verlies jij het spel en wint de tegenstander automatisch.

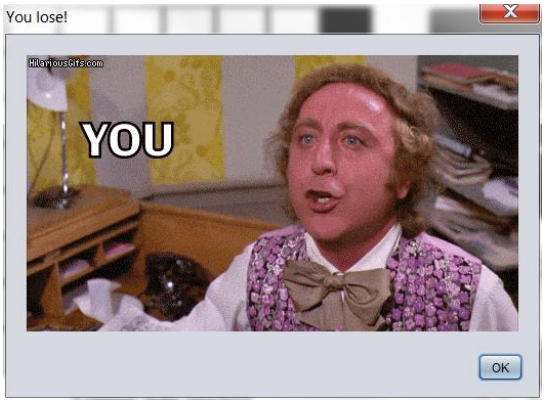


screenshot 10

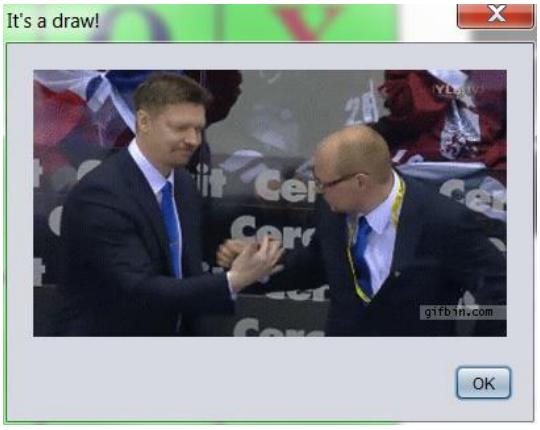
Bij hint wordt de AI gebruikt zonder dat de AI zelf de zet speelt. De AI toont alleen aan waar je het beste een zet kan plaatsen. De hint hebben wij er voor onszelf ingebouwd om tijdens het toernooi extra kans te maken tijdens de ronde waarbij iedereen met de hand ging spelen.

Er zijn ook drie gifjes ingevoegd om te laten zien of je hebt gewonnen, verloren of gelijk gespeeld.

screenshot 11



screenshot 12



screenshot 13

3.2 Client/Server protocol

Voor het implementeren van de interface is gekeken naar het client/server protocol. Er is hier gekeken naar welke commands er zijn en waar dan gemakkelijk gebruik van gemaakt kan worden.

De command voor het inloggen bestaat uit “Login <speler>”. Dit hebben wij opgesplitst naar alleen speler en als er verzonden wordt naar de lobby dan komt het login command er bij te staan. Server side is er dus wel een command maar client side is er niks van te zien.

In de lobby kon gebruik gemaakt worden van: “get playerlist”, “subscribe”, “get gamelist” en “challenge”. De ‘get playerlist’ is gebruikt om alle spelers die met de server zijn verbonden in de lobby weer te geven. De subscribe is samen met de game-mode button die daar boven stond gemaakt. Als er op subscribe drukt wordt dan subscribed de persoon zich dus voor de game die op dat moment is geselecteerd uit het drop-down menu. Hiervoor was ook de ‘get gamelist’ nodig om te kijken welke game modules er beschikbaar zijn. Deze games zijn dan in het drop-down menu verwerkt.

Bij de games hebben we gebruik gemaakt van het ‘forfeit command’ om de speler een mogelijkheid te geven om op te geven. Met de ‘command move’ is het mogelijk om een zet te plaatsen op het scherm. Onderin staat ook nog wie er aan de beurt is. Dit wordt opgevraagd via de ‘SVR GAME YOURTURN’. Als deze true is dan ben jij aan de beurt en anders de tegenstander.

3.3 Toevoegen van nieuwe game/game module

Om een nieuwe game toe te voegen moet er een nieuwe gamemodule worden ontwikkeld. Deze gamemodule kan het skelet van bestaande gamemodules gebruiken. Er zijn echter zijn restricties voor het toevoegen van een nieuw spel. Het spel wordt door twee spelers gespeeld. Het spel maakt gebruik van een twee dimensionaal bord. Verdere restricties worden bepaald door de server commando’s.

Na het toevoegen van een nieuwe gamemodule zullen er in enkele klassen wijzigingen moeten worden aangebracht. Denk hierbij aan de grootte van het bord, of eventuele kleuren als er van standaard zwart of wit wordt afgeweken.

Om de grootte van het nieuwe bord te kunnen gebruiken moet in de klasse FrameworkController.java de methode actionPerformed van de private chooseGameListener klasse worden gewijzigd. Er moet een nieuw if-statement worden toegevoegd voor de nieuwe gamemodule. In dit if-statement wordt doormiddel van het aanroepen van de methode changeBoardSize(int rows, int cols, int fontSizeOfPieces) die zich bevindt in de klasse GameView.java de grootte van het bord gewijzigd en geüpdatet.
Het kan zijn dat het gewenst is om voor beide spelers een unieke kleur te kiezen. In de klasse GameView.java bevindt zich een methode waarmee de kleuren van de characters kunnen worden gewijzigd. Dit is de methode paintColorsOnBoard(ArrayList<Jlabel> labelsOfTheBoard), ook hier zal een nieuw if-statement moeten worden toegevoegd of de standaard kleuren zullen moeten worden gewijzigd. Hiernaast hoeft er niet aangepast te worden om een nieuwe game toe te voegen.

3.4 Toevoegen van een artificial intelligence

Om een nieuwe Artificial Intelligence toe te kunnen voegen moet de code op enkele plekken worden aangevuld. In de klassen GameView.java en FrameworkController moeten de volgende wijzigingen worden gemaakt:

- In de klasse GameView.java moet de methode updateGameBoard(int position, String playerChar) worden voorzien van een if-statement die de placeOnBoard methode van de AIModule aanroept. Dit moet omdat deze methode kan verschillen van parameters

- In de klasse FrameworkController.java moet de nieuwe AI worden toegewezen aan de AIModule waarvan het framework gebruik maakt. Dit gebeurd in de methode actionPerformed van de private chooseGameListener klasse. Met de volgende regel kan een nieuwe AI worden toegevoegd: AIModule = new instanceOfNewAI();

Naast deze wijzigingen hoeft er niets te worden gedaan om de AI te laten functioneren.

3.5 Toelichting OthelloAI

Het algoritme van onze AI (Artificial Intelligence) is het greedy algortime. Het greedy algoritme houdt in dat er gekeken wordt naar het maximum. In dit geval is het halen van de meeste disks in een beurt.
Als we een kijkje nemen in de klasse OthelloAI zien we als eerste dat deze de abstracte klasse extend. Dit hebben we gedaan om zo gemakkelijk nieuwe AI's toe te kunnen voegen en ervoor te zorgen dat je op 1 plek alle variabelen krijgt die elke AI sowieso gaat gebruiken. Denk hierbij aan de mogelijke posities van een vakje, de spelers en methoden zoals getOpponent(). Het bord heeft een dimensie van 8 bij 8 die we bovenaan met de gamename definiëren. Daarnaast hebben we voor elke mogelijke richting ten opzichte van een bepaald blokje de richtingen gedefinieerd. Ook hebben we het bord waardes gegeven die kijkt of er een vakje belangrijker is dan een ander vakje.

Voor het ophalen van de beste zet hebben we een methode getBestMove() die een speler als parameter meekrijgt. In de methode getBestMove() wordt er een andere methode aangeroepen genaamd getMoveHighestScore(int player)met als parameter dezelfde player. De getMoveHighestScore() methode is waar alles bij elkaar komt. Hier wordt er eerst een loop gemaakt door het hele bord en kijk voor elke vakje of het een valid move is. Als er een vakje is gevonden waar een valid move wordt kan worden gemaakt, gaan we kijken hoeveel punten hij met deze move kan maken. Deze score houden we bij. Als er vervolgens een nieuwe move komt die een hogere score heeft, dan wordt deze opgeslagen. Uiteindelijk wordt de move met de hoogste score gereturned. Vervolgens wordt deze move geplaatst op het bord.

Hoofdstuk 4 – Testen

Het testen van de applicatie hebben we aangepakt door elke wijziging aan de applicatie goed te laten testen door de leden van de projectgroep. Na een wijziging in de applicatie gingen eerst de hoofdprogrammeurs de nieuwe wijziging lokaal testen. Als het niet werkte, gingen ze opnieuw aan het programmeren. Als het wel werkte, gaven ze een seintje aan de overige projectleden, en die gingen vervolgens zowel lokaal áls op de test-server de nieuwe wijziging testen.

Als de overige projectleden aan het testen waren, gingen de programmeurs verder met het schrijven van het programma, zodat het tempo erin bleef.

Het lokaal testen en het testen via de test-server wat gedaan werd door de overige projectleden, ging als volgt:

Ze startten een potje Tic-Tac-Toe of Othello tegen elkaar, en gingen tegen elkaar spelen. Dit deden ze 2 of 3 keer. Vervolgens gingen ze de situatie testen waarbij 1 speler bestuurd werd door de AI, en de andere door de persoon zelf. Na weer 2 of 3 potjes wisselden de spelers om van rol. Na ook hier 2 of 3 potjes van te hebben gespeeld, gingen ze de situatie testen waarbij beide spelers bestuurd werden door de AI. Ook in deze situatie werden er 2 of 3 potjes gespeeld.

Gedurende dit proces schreven de projectleden al hun bevindingen van het testen op. Enkele voorbeelden van zulke bevindingen zijn:

- Veld wordt niet geleegd bij starten nieuw potje
- AI doet onverwachte dingen
- Winst/verlies resultaat klopte niet
- Onverwachte invalid moves
- Etc.

Deze bevindingen gaven de testers door aan de programmeurs, en die gingen er vervolgens weer mee bezig om de bevindingen op te lossen.

Aangezien de testers zowel lokaal als via de test-server aan het testen waren, werd het mogelijk om problemen in hun geheel te elimineren, zodat de applicatie robuust genoeg was voor het toernooi.

De laatste paar dagen voorafgaand aan het toernooi hebben we via de test-server regelmatig potjes gespeeld tegen andere projectgroepen. Ook hier schreven we onze bevindingen van op, en keken we naar hoe het programma zich gedroeg. Dit bracht geen nieuwe dingen naar voren die we zelf nog niet hadden ontdekt tijdens het onderling testen, dus waren wij er van overtuigd dat onze applicatie het toernooi goed zou doorstaan.

Behalve dit, waren wij ook van plan om uitgebreide unit-tests te doen om de functionaliteit van de applicatie te testen. Vanwege alle drukte rondom het toernooi, besloten wij om dit pas in de allerlaatste fase van het project te doen, na het toernooi. Tom Broenink heeft deze taak op zich genomen, maar het is hem uiteindelijk niet meer gelukt. Hij had veel problemen met Eclipse en de libraries van jUnit, waardoor het hem niet meer gelukt is. Wegens tijdgebrek kon niemand anders deze taak nog op zich nemen, en daarom hebben wij als groep besloten dit te laten gaan.

Hoofdstuk 5 – Conclusie en evaluatie proces

Het project is tot een succesvol einde gebracht. Wij hebben het eindproduct binnen de deadline af weten te ronden. Voor de tic-tac-toe AI hadden we geen tijd meer, omdat het toernooi alleen om reversi ging hebben we er voor gekozen om de tic-tac-toe AI maar te laten voor wat het was en de reversi AI zo goed mogelijk te maken. Met de greedy AI zijn we uiteindelijk tweede geworden wat een goed gevoel gaf en het toch nog waard was om het tic-tac-toe AI te verwaarlozen.

Het proces tot het eindproduct is prima gegaan. De eerste dag is er een Slack groep opgezet voor communicatie en snelle screenshot uitwisseling. Ook een WhatsApp groep en Git repository waren gelijk opgezet voor communicatie en bestandsuitwisseling. Er is na elke meeting een lijstje gemaakt met wat er moet gebeuren en wie er mee bezig moet. Er werd elke avond ook even gekeken wat iedereen heeft gedaan en tijdens het programmeren kwam er elke keer een test online om te testen of wat er net is gemaakt ook echt gaat werken. Er was uiteindelijk wel wat tijdnood in verband met herkansingen en oplevering van documenten voor MBBC en Belbin, maar dit is goed opgepakt en het project is uiteindelijk voor de deadline af.