Trots op Sonja

Groepsnaam: Sonja.
Auteurs: Nick Tönissen, Arjen Theunis en Tessa Schlief.

26 juni 2014

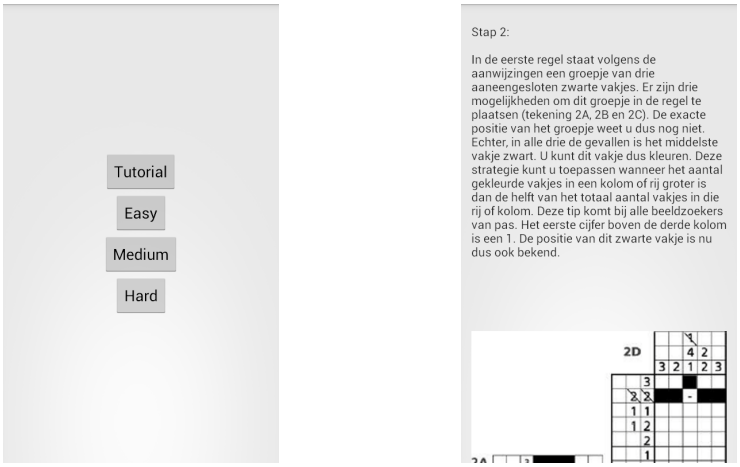
1 Voorwoord

Dit verslag is het eindverslag voor de cursus Research & Development. In deze cursus hebben wij twee verschillende applicaties moeten maken. De eerste applicatie was om het werken met Android te leren kennen. De tweede applicatie is ons eindproduct voor deze cursus. Wij hebben een applicatie gemaakt waarmee men Japanse puzzels op kan lossen. Wat deze applicatie allemaal kan, wordt uitgelegd in de sectie Beschrijving. Vervolgens wordt in de sectie Ontwerp verteld waarom er gekozen is voor de functionaliteit van de applicatie zoals deze is. Ook wordt het ontwikkelproces van de applicatie uitgebreid beschreven. Hoe de applicatie in elkaar zit, welke problemen we in het ontwerp tegenkwamen en hoe we deze problemen hebben opgelost staat allemaal in de sectie Ontwerp. Als laatste schrijven wij de reflectie, waarin staat hoe we het vonden om aan deze applicatie te werken, wat er goed en minder goed ging en of we tevreden zijn over het hele proces en het eindresultaat.

2 Beschrijving

2.1 Inleiding

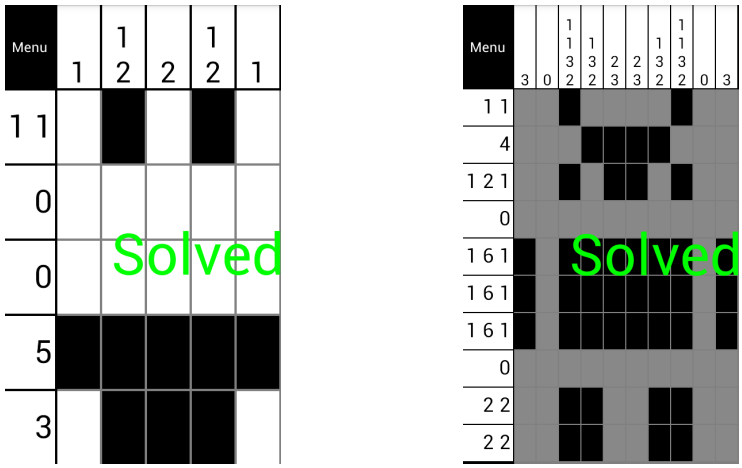
Sonja is een applicatie voor android smartphones die de gebruiker in staat stelt om Japanse puzzels, ook wel een nonogram genoemd, op te lossen. Het is een Japanse beeldpuzzel die uit drie delen bestaat. Een leeg raster in het midden met daarboven en links van het raster rijen getallen voor elke kolom en rij van het raster. Elk getal geeft aan hoeveel aaneengesloten zwarte blokjes er moeten zijn in de desbetreffende rij of kolom. De gebruiker kan door op een vakje te klikken dat vakje van kleur laten veranderen. Een vakje kan met drie verschillende kleuren worden ingevuld. Wit is de default kleur en heeft voor de rest niet echt een betekenis. Door op een wit vakje te tappen wordt het vakje zwart, zwart betekend dat je zeker weet dat dat vakje gekleurd moet zijn. De zwarte vakjes vormen uiteindelijk ook de oplossing. Door op een zwart vakje te tappen wordt het grijs, dit is bedoeld om aan te geven dat je zeker weet dat daar geen zwart vakje kan, dit is vooral makkelijk als je een grote puzzel aan het maken bent. Als je op een grijs vakje tapt wordt het weer wit, zo kan de gebruiker makkelijk wisselen tussen verschillende toestanden van een vakje. Voor meer informatie over hoe de gebruiker het beste kan spelen biedt Sonja naast verschillende moeilijkheidsgraden puzzels ook nog de mogelijkheid om een tutorial te lezen zodat zelfs mensen waarvoor dit soort puzzels nieuw zijn makkelijk met de applicatie kunnen spelen. Door de vakjes uiteindelijk kloppend in te kleuren krijgt de gebruiker een afbeelding te zien, de oplossing van de puzzel.



(a) Menu (b) Tutorial

Figuur 1: Begin van applicatie

Dan als een speler de gekozen puzzel opgelost heeft geeft de applicatie dit aan door middel van het woord Solved, dat over de puzzel heen verschijnt.



(a) Menu (b) Tutorial

Figuur 2: Spelen van applicatie

2.2 Productverantwoording

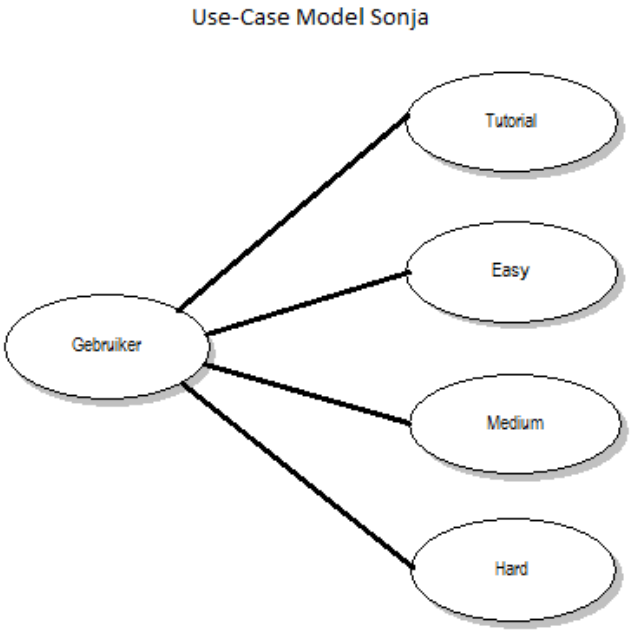
Er zijn niet veel applicaties beschikbaar in de Play Store die dit spel aanbieden. GraphiLogic was een van de weinige applicaties die we konden vinden. Een van de nadelen van deze applicatie was dat de zoom functie niet lekker werkte, het schokte teveel en was niet fijn om te gebruiken. Ook zag de opmaak van de gehele applicatie er niet uit, zwart met wit en felle kleuren. Ook waren de advertenties op de achtergrond zeer storend tijdens het spelen. Dit zagen we bij andere applicaties ook terug, het zoomen ging slecht of de applicatie zag er niet uit. En als het wel goed in elkaar zat dan koste de applicatie geld of maakte het gebruik van advertenties. Sommige applicaties, waaronder GraphiLogic, stelden de gebruiker in staat om zelf puzzels te maken en eventueel te verzenden naar vrienden, bij Sonja hebben we daar nadrukkelijk niet voor gekozen. Deze puzzels zijn namelijk zeer moeilijk te maken, een foutje is al snel gemaakt en de kans is groot dat de puzzel geen unieke oplossing heeft. Ook heeft Sonja geen fancy kleuren of opvallende advertenties, dit zorgt ervoor dat niks de gebruiker afleidt van het doel van de applicatie; de gebruiker in staat stellen om een Japanse beeldzoeker op te lossen. Ook met het andere knelpunt hebben we bij Sonja rekening gehouden. Om geen rare schokkerige of niet intutieve zoomfuncties in de applicatie te krijgen, hebben we er uiteindelijk maar voor gekozen om het hele zoom gedeelte achterwege te laten. Door het speelveld afhankelijk te maken van de schermgrootte van de mobiele telefoon of tablet waarop de applicatie gebruikt wordt, hebben we ervoor gezorgd dat het spel er op kleine maar ook op grote schermen goed uit ziet. Alle puzzels die Sonja aanbiedt zijn uitvoerig getest op

het feit of het voor de gebruiker nog wel goed te zien was en of ze wel op een unieke manier oplosbaar zijn. Het laatste grote voordeel is dat Sonja helemaal gratis is.

2.3 Specificaties

2.3.1 Functionele eigenschappen

Sonja is ontwikkeld met de focus op functionaliteit, vandaar dat de applicatie niet veel andere opties bied op het spelen van japanse beeldpuzzels na. Dit kan je duidelijk terug zien in ons Use Case Model, hier zitten maar vier delen in; Tutorial, Easy, Medium en Hard puzzels. Ter verduidelijking hebben we de Use Case "Medium"helemaal uitgewerkt. Deze Use Case is representatief voor bijna elke andere Use Case.



Use Case:	Medium Puzzel
Beschrijving	In deze Use Case wordt beschreven hoe de gebruiker met een puzzel om kan gaan
Actoren	De persoon die de applicatie Sonja gebruikt, hierna gebruiker genoemd.
Trigger	De gebruiker selecteert "Medium" in het hoofdmenu.
Basic Course of Events	1. Het systeem laat het speelveld zien en de bijbehorende cijfers bij elke kolom en rij
	2. De gebruiker tapt op een vakje
	3.1 Als het vakje wit is dan laat het systeem nu op die plek een zwart vakje zien
	3.2 Als het vakje zwart is dan laat het systeem nu op die plek een grijs vakje zien
	3.3 Als het vakje grijs is dan laat het systeem nu op die plek een wit vakje zien
	4. Het systeem laat in het groen "Solved" zien als de laatste actie de oplossing maakt
Alternate paths	A1. De gebruiker tapt op een zwart vakje terwijl "Solved" in beeld staat
	A2. Het systeem haalt "Solved" weg
Exception paths	E1. De gebruiker klikt op de menu knop
	E2. Systeem geeft het hoofdmenu weer
Assumpties	De gebruiker heeft een android smartphone die geschikt is voor de Sonja applicatie
Precondities	-
Postcondities	De gebruiker heeft een actie ondernomen in het speelscherm

De uitgewerkte Use Case voor het geval dat de gebruiker “Medium”selecteerd in het hoofdmenu

2.3.2 Niet-functionele eigenschappen

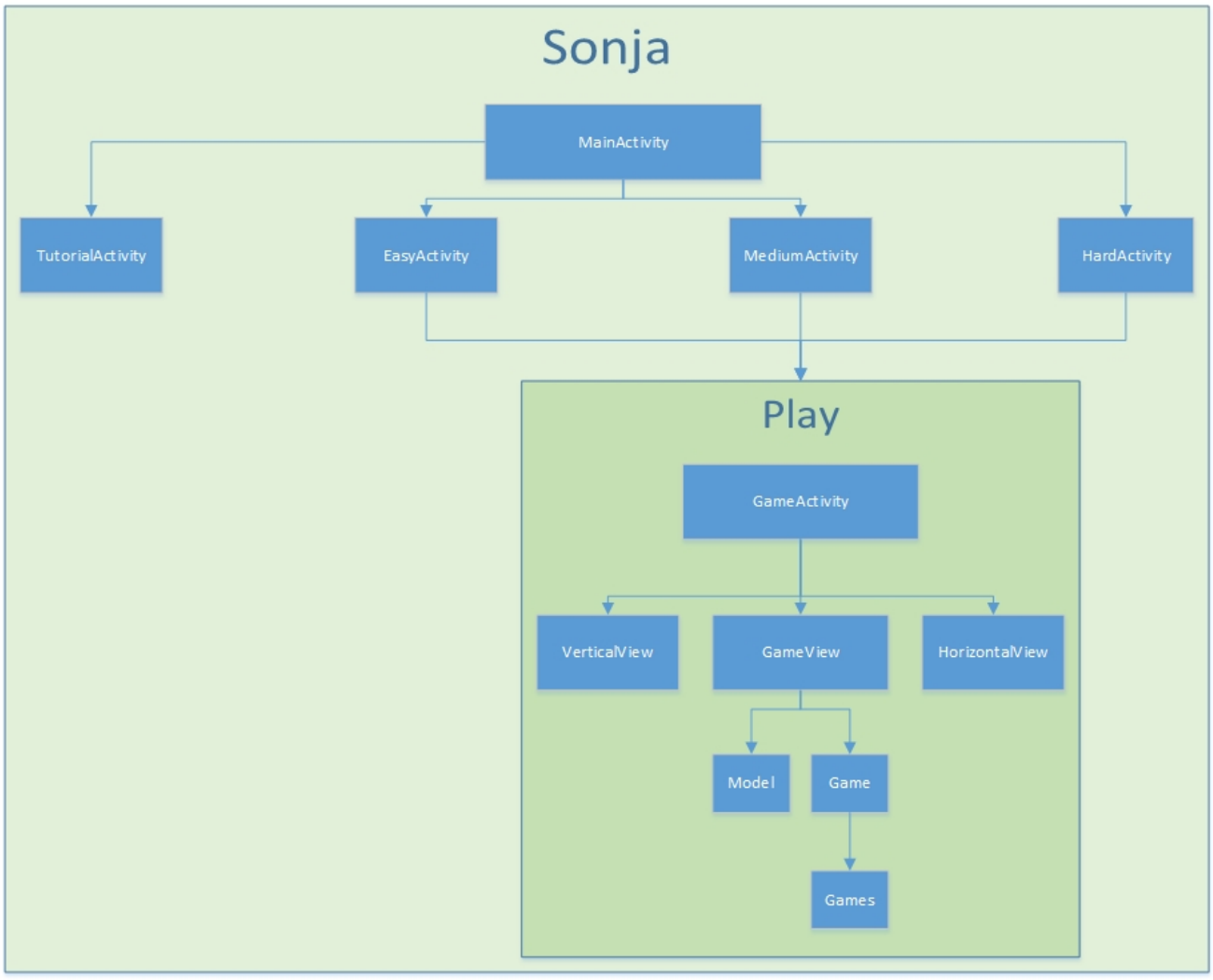
- Beschikbaarheid:
  Doordat Sonja geen internet, gps of bluetooth connectie nodig heeft is de applicatie 24/7 beschikbaar voor de gebruiker. Dit heeft ook als bijkomend voordeel dat Sonja erg batterijvriendelijk is.

- Error tolerantie:
  Sonja is zo geprogrammeerd dat het vrijwel onmogelijk is om de applicatie te laten crashen. De gebruiker hoeft dus niet bang te zijn voor onbegrijpelijke foutmeldingen of dat het zijn of haar voortgang in een puzzel opeens kwijt raakt.

3 Ontwerp

3.1 Globaal ontwerp

Het globale ontwerp van onze applicatie kan het best worden weergegeven in een diagram, wat laat zien welke classes invloed op elkaar hebben en door welke andere classes ze gebruikt worden. Hieronder staat zo'n diagram van onze applicatie.



Hiërarchie van de classes

We hebben voor een simpele en duidelijke navigatie voor het menu gekozen, hierdoor was het vrij eenvoudig om dit te implementeren. Zoals in het diagram zichtbaar is kan je in het hoofdmenu van de applicatie vier keuzes maken: Tutorial, Easy, Medium of hard.

- Tutorial:
  De Tutorial is een losstaand object van onze applicatie en geeft zoals de naam al zegt een uitleg over wat je moet doen om een puzzel op te lossen.
- Moeilijkheidsgraad:
  Dan heb je nog de drie andere knoppen. Dit zijn moeilijkheidsgraden voor het spelen van het spel. Binnen al deze moeilijkheidsgraden kan je weer kiezen tussen verschillende puzzels. Als je een puzzel gekozen hebt kan je beginnen met spelen, hetgeen wat je geleerd hebt in de tutorial.

Door op deze wijze de gebruiker door het menu te laten lopen, weet de gebruiker precies waar hij/zij zich bevindt en is het voor iedereen te begrijpen.

3.2 Detailontwerp

Hoe zit het nu werkelijk in elkaar. Zoals gezegd, kan een gebruiker eigenlijk twee keuzes maken. Of hij/zij leest de tutorial of hij/zij kiest een moeilijkheidsgraad.

Tutorial:

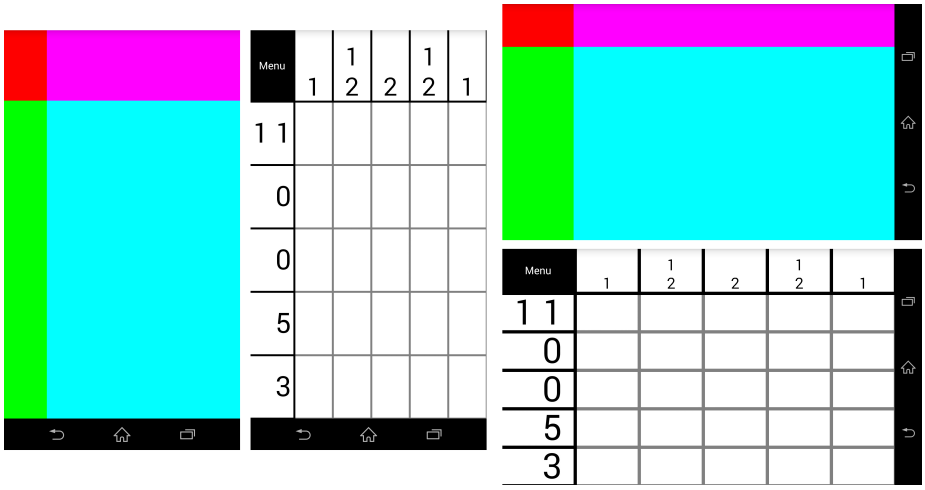
De tutorial is een activity die vrij gemakkelijk in elkaar zit. Het opent een "ScrollView" waarin, door middel van tekst en plaatjes, wordt uitgelegd wat de gebruiker geacht is te doen en hoe. Het gemakkelijke aan zo'n "ScrollView" is dat de tekst automatisch wordt uitgelijnd aan de grootte van het scherm van de telefoon. Hierdoor is het op elke telefoon leesbaar en kan je simpelweg door steeds verder naar beneden te scrollen de gehele tutorial lezen.

Moeilijkheidsgraad:

Deze activities zitten zelf erg eenvoudig in elkaar. Na elke moeilijkheidsgraad wordt een activity gestart waarin het aantal verschillende puzzels dat deze moeilijkheidsgraad bevat wordt getoond. De buttons waarop gedrukt kan worden om een bepaalde puzzel te starten bevatten allemaal een "Tag" die aangeeft welke knop het is. Hierdoor kunnen wij in de activity "Game", die hierdoor wordt aangeroepen, bepalen welke puzzel nu geladen moet worden. Dan kan je je nog afvragen waarom wij voor elke moeilijkheidsgraad een activity hebben gemaakt. Het antwoord daarop is vrij simpel. Ten eerste staan alle puzzel van een moeilijkheidsgraad bij elkaar en is er nu niet een scherm waarop alle puzzel van verschillende moeilijkheidsgraden door elkaar staan. Ten tweede geeft dit ons

nog extra informatie. We kunnen in de activity Game namelijk aan de hand van een gegeven integer kijken, vanuit welke moeilijkheidsgraad de aanroep kwam. Op basis van deze 2 gegevens kunnen we uit een dubbele array halen welke moeilijkheidsgraad het is en van die moeilijkheidsgraad de gekozen puzzel opvragen.

Als dit allemaal bekend is kan de puzzel gerepresenteerd gaan worden.



Plaatjes van de 4 linear layouts

Zoals in de afbeelding hierboven weergegeven is, zijn er vier velden binnen het scherm van een puzzel. Vanuit drie van die vlakken worden “Views” aangeroepen. Het gaat dan om een HorizontalView welke in het groene vlak wordt aangeroepen, een VerticalView welke in het roze vlak wordt aangeroepen en tot slot wordt in het blauwe vlak de GameView aangeroepen.

- HorizontalView:
  Wat deze “View” doet is naast het speelveld horizontaal de benodigde getallen neerzetten. Doordat we weten welke game gerepresenteerd gaat worden in het speelveld, weten we ook hoe groot deze puzzel is. Op basis van deze informatie kunnen we precies bereken hoe hoog een vlakje van de puzzel wordt. Als we dit weten kunnen we gaan bepalen hoe groot onze tekst maximaal mag zijn om nog binnen deze vlakjes te vallen. We gebruiken hiervoor een toString methode in “Game” die de langste string oplevert die er in moet komen. Met deze string bepalen we de maximale tekstgrootte. Dan worden door middel van één enkele for-loop de vlakjes getekend en wordt gelijk de tekst erin gezet. Deze tekst is rechts uitgelijnd binnen vlakjes, dan krijg je het resultaat zoals in de afbeelding weergegeven.

- VerticalView:
  Bij deze "View" tekenen we de tekst niet horizontaal, maar zoals de naam al aangeeft verticaal. Toch ook weer niet geheel verticaal, want de tekst komt hier onder elkaar te staan zoals te zien is in de afbeelding. Hier wilden we precies hetzelfde principe toepassen als bij de "HorizontalView", helaas ging deze vlieger niet op, want een canvas kan niet overweg met "endlines". De strings die we gebruikten voor deze verticale representatie bevatte deze "endlines" om de tekst netjes onder elkaar te krijgen. Na veel geprobeerd te hebben is dit niet gelukt en hebben we hier in plaats van een toString methode gewoon een array terug gegeven en de getallen handmatig onder elkaar gezet. Dit hebben we gedaan door de maximale tekstgrootte te gebruiken om het teken van de getallen steeds met dat grootte verschil te laten gebeuren.

- GameView:
  Hier komt het daadwerkelijke "bord" van het spel. Hier berekenen we weer op basis van hoe groot de puzzel moet worden, 5 bij 5, 10 bij 10 of 15 bij 15, de grootte van de hokjes. Daarna schrijven we deze hokjes weg op basis van een matrix die bevat welke kleur deze hokjes moeten worden. Doordat de matrix van het spel al geïnitaliseerd is voordat GameView aangeroepen wordt kan je tijdens het spelen je telefoon naar "Landscape Mode" draaien en weer terug zonder dat je weer opnieuw moet beginnen. De methode Gameview roept weer twee andere classes aan: het Model en Games.

  - Games:
    Hierin staan, zoals de naam al aangeeft, alle puzzels en hun huidige toestand opgeslagen. Deze functie wordt door GameView aangeroepen om voor het huidige spel een "Game" aan te maken. Deze wordt weer gebruikt door het Model.
  - Model:
    Het Model krijgt een bepaalde "Game" mee en bepaalt bij elke wijziging of dat de puzzel al opgelost is. Dit wordt dan als een boolean waarde terug gegeven. Als deze waarde "true" is dan tekent "GameView" het woord "Solved" over de puzzel heen zodat je weet dat de puzzel correct is gemaakt.

3.3 Ontwerpverantwoording

Voordat we begonnen met het ontwikkelen van de applicatie hadden we veel problemen met de vraag hoe we de puzzels moesten gaan weergeven. We moesten eigenlijk drie vlakken hebben die een samenhang hadden, maar we moesten ze wel apart kunnen benaderen. Toen hebben we nagedacht over het gebruik van een canvas voor het hele spel maar dit bracht een hoop moeilijkheden met zich mee. We konden toen wel alles aan elkaar linken, maar dan zouden sommige dingen onnodig vaak opnieuw getekend worden, wat slecht is voor de prestaties van de applicatie. Toen bedachten we dat we dit tekenen nu over drie canvassen konden verdelen, want dan zouden we van dat probleem af zijn. Toen was het nog de vraag hoe we dit konden doen. Hiervoor hebben we bij de studentassistenten aan de bel getrokken. Maurice gaf toen aan dat we het best vier linear layouts konden gebruiken. Deze maken automatisch een soort matrix voor je waarin we onze items dan konden tekenen. Dit hebben we geprobeerd maar dit is niet gelukt. We konden niet achterhalen hoe je de gegenereerde matrix nu kon gebruiken. Zelfs na informatie gekregen te hebben van Maurice lukte het niet. We zijn toen weer terug gestapt naar het originele idee met de canvassen, maar dan verdeeld over vier linear layouts.

Dit was toen een redelijk goede oplossing. Toen bedachten we dat we, aangezien we toch geen gebruik maken van de matrix die deze linear layouts maken, net zo goed relative layouts konden gebruiken. Toen kregen we het tweede probleem; hoe gaan we schalen op basis van de grootte van de telefoon. We konden hier niks op verzinnen en toen kwam Maurice met het idee om “weights” te gebruiken. Hiermee kan je aangeven hoe belangrijk iets is en dus hoe groot het zou moeten worden. Helaas kan dit niet bij relative layouts en toen zijn we weer terug gaan naar de linear layouts. Toen hadden we op elke telefoon een goede verhouding tussen waar de tekst moest komen en waar het speelveld moest komen. Een leuke bijkomstigheid hiervan is dat het ook werkt voor “Landscape mode” (zoals hierboven in de plaatjes te zien is). Toen konden we eindelijk gaan beginnen met het implementeren van de applicatie, ten minste dat dachten we.

Het volgende probleem kwam alweer om de hoek kijken. Hoe gaan we de puzzels omzetten naar bruikbare code. Toen bedachten we dat als we twee arrays zouden gebruiken om de cijfers voor het oplossen op te slaan dan zouden we aan de hand daarvan wel kunnen bereken wat de oplossing zou moeten zijn. Helaas was dit niet zo gemakkelijk als we dachten en moesten we een andere oplossing verzinnen. Het erbij opslaan van de oplossing zou dit oplossen maar dan zou je een drie dubbele array moeten gebruiken om alles te linken. Dit zou enorm veel complexiteit met zich mee brengen en hier zijn we dus ook vanaf gestapt. Toen kwam Niek, een studiegenoot, met het idee om net als in C++ een “struct” te gebruiken. We hebben toen de klasse “Game” gemaakt en hier de drie arrays in opgeslagen. Deze “Game’s” hebben we dan weer opgeslagen in een “LinkedList”.

Na het oplossing van deze twee moeilijkheden, ging het verder ontwikkelen van de applicatie eigenlijk redelijk gemakkelijk. We liepen niet echt tegen hele grote problemen meer aan. Het enige wat nog op ons pad kwam was het tekenen van een string met endlines in een canvas, dit hebben we uiteindelijk ook opgelost zoals bij VerticalView beschreven staat.

Hiermee hebben we denk ik een redelijk goed ontwerp van de applicatie gemaakt, wat niet enorm ingewikkelde code met zich mee brengt. Helaas kregen we achteraf van Maurice te horen dat het met één array gekund had. Hij zei: "*jullie heb nu gedacht vanuit het begin van de puzzel. Maar als jullie vanuit de oplossing gedacht hadden, zoals je een doolhof vaak ook makkelijk op kan lossen, had je kunnen bedenken dat als je de oplossing opslaat, je de rest relatief gemakkelijk had kunnen berekenen*". Dit zou inderdaad gekund hebben en zou dataopslag gespaard hebben, maar voor het aantal puzzels dat wij er nu in hebben staan zou het niet opwegen tegen het werk om hiervoor een algoritme te schrijven. Wel zou het meer gericht zijn op uitbreiding als we dit gedaan hadden. Hiermee heb je denk ik ook gelijk de rede waarom deze vraag: "*Geef aan waarom jullie ontwerp een goed ontwerp is.*" eigenlijk niet goed te beantwoorden valt, want je kan altijd wel code blijven optimaliseren, maar hoe zinnig is het voor hetgeen waar je mee bezig bent. Dus wanneer heb je een goed ontwerp?

4 Reflectie

4.1 Tevreden over het eindresultaat

Wij zijn tevreden over het eindresultaat van onze applicatie. De applicatie werkt. Alles wat we wilden dat de applicatie in ieder geval zou kunnen, doet het. Het menu is er met de tutorial, waarvan we ook blij zijn dat we deze hebben toegevoegd, en met de verschillende levels van puzzels. Per level is er een menu met daarin keuze uit een aantal puzzels die men kan maken. Het oplossen van de puzzel gaat goed. De hokjes kleuren bij één keer klikken zwart, bij twee keer klikken grijs en bij drie keer klikken weer wit zoals we graag wilden. De representatie van de puzzel ziet er netjes en duidelijk uit. Als de puzzel is opgelost komt er in beeld te staan dat de puzzel is opgelost. De applicatie crasht niet en heeft geen overbodige knoppen/functies die niet werken. En dit allemaal binnen de tijd die we ervoor gekregen hebben.

4.2 Verbeterpunten en mogelijkheden voor verdere ontwikkeling

Natuurlijk kan onze applicatie nog verbetert worden en er zijn zeker mogelijkheden voor verdere ontwikkeling. Het design is nu erg simpel. Wij hebben hier bewust voor gekozen, want functionaliteit had voor ons een hogere prioriteit dan een mooi design. Maar er is hier dus nog wel ruimte voor verbetering. Er kunnen mooiere knoppen worden gemaakt en er zou wat kleur aan de menu's toegevoegd kunnen worden. Ook kunnen we in plaats van het standaard android figuurtje een mooi plaatje maken voor het icoontje van de applicatie dat op een telefoon te zien is voor je de applicatie opstart. De applicatie kan ook altijd uitgebreid worden door meer puzzels toe te voegen. Wat we al als userfeedback hebben gekregen van de ervaren puzzelaar is dat het handig zou zijn als er om de vijf rijen en kolommen in de puzzel een dikkere zwarte lijn zou staan. Dit maakt het puzzelen namelijk een stuk makkelijker.

4.3 Wat we hebben geleerd

We hebben erg veel geleerd over android, maar dit was ook niet zo moeilijk, want we hadden er alle drie nog geen verstand van vóór we aan dit project begonnen. We hebben niet alleen geleerd om met android te werken, maar ook dat het ontwikkelen van een applicatie veel tijd kost en niet makkelijk is. Door hulp te vragen van medestudenten, van de student assistenten en door veel op te zoeken op internet hebben we verschillende problemen opgelost. Hoe we dit precies hebben gedaan, bijvoorbeeld hoe we uiteindelijk de puzzels representeren door middel van code, staat allemaal uitgelegd in de sectie over het ontwerp. Over samenwerking hebben we niet zo veel geleerd, want de samenwerking ging goed. We hebben taken duidelijk verdeeld en vaak samen gezeten om alle opdrachten te maken en de applicatie te ontwikkelen. We hebben slim gewerkt en gekozen voor een applicatie die niet al te moeilijk was, maar wel van redelijk niveau voor ons en die we van heel simpel tot heel uitgebreid zouden kunnen maken naar gelang we tijd zouden hebben.

4.4 Conclusie

Zonder voorkennis en met de relatief korte tijd die we hadden om deze applicatie te ontwikkelen mogen we zeer tevreden zijn met het eindresultaat. Er zijn natuurlijk nog genoeg mogelijkheden om de applicatie te verbeteren en uit te breiden, maar onze applicatie is zeker niet slecht. De applicatie werkt en is duidelijk en gebruikers zijn enthousiast. In de toekomst zouden we een project als dit waarschijnlijk op eenzelfde manier aanpakken, want het project is goed gegaan.