The adventure of making a:Dventure

Jonathan Moerman Mathis Sackers
Jan Potma Tom Nijholt

26 Juni 2015

2 VOORWOORD

Dit document is bedoeld om een terugblik te geven op het ontwikkelproces en het uiteindelijke ontwerp van ‘a:Dventure’, de app gemaakt voor het vak Research & Development 1, door de groep `free(lunch);`. Met dit document hopen we duidelijk te maken hoe onze app tot stand gekomen is. We zullen dit doen door eerst een inleiding te geven waar we de basale functies van a:Dventure zullen beschrijven. Dit volgen we op met de productverantwoording waarin we verantwoorden waarom er behoefte is aan onze app, waarna we dieper in gaan op de eigenschappen van de app in de specificatie. Daarna komt het technische aspect in het ontwerp gedeelte, waar we ingaan de structuur van de code. Tot slot is er een reflectie, waar de problemen die tijdens het ontwikkelproces onstonden besproken worden en onze ervaringen bij het algemene ontwikkelproces worden uitgelicht.

3 BESCHRIJVING

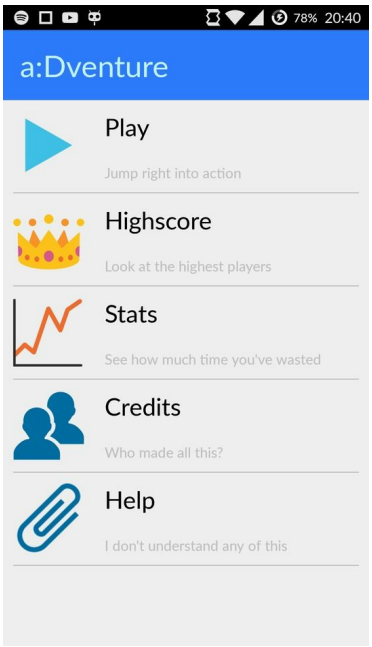
3.1 Inleiding

a:Dventure is een tactische rogue-like met emoji illustraties. Het design is gemaakt naar ontwerp van chat-applicaties zoals WhatsApp. De invoer van de gebruiker wordt gedaan door eenvoudige touch-input.
De gebruiker probeert een zo hoog mogelijk puntenaantal te verkrijgen door computergestuurde vijanden te verslaan en zelf niet geraakt te worden, en beweegt in een raster over een beperkt speelveld (het level). Afwisselend zijn de speler en de vijanden aan de beurt. Wanneer alle vijanden in een speelveld verslagen zijn gaat de speler naar een volgend level. Zowel de speler als de verschillende types vijanden hebben unieke bewegingsmogelijkheden.
Naarmate de speler het spel meer speelt leert hij of zij het spel en de mechaniek ervan beter kennen, waardoor hij of zij beter in het spel wordt en tactischer gaat spelen.

3.1.1 Rondleiding

Het eerste scherm dat de gebruiker te zien krijgt wanneer de app is opgestart is het hoofdmenu. Hiervandaan zijn alle andere schermen te bereiken. In het hoofdmenu heeft men de keuze uit 5 opties:

- ***Play***. Hier kan men het spel spelen.
- ***Highscore***. Hier is de top 10 te bekijken.
- ***Stats***. De statistieken van de gebruiker zijn hier te zien.
- ***Credits***. Hier kan men zien wie er verantwoordelijk zijn voor het maken van deze app.
- ***Help***. Hier kan men een simpele uitleg vinden over het spel, in het bijzonder de bewegingsmogelijkheden.



Het hoofdmenu

Play
Zodra de gebruiker op ‘Play’ drukt wordt onmiddellijk een level gegenereerd.

Het initiële Play venster

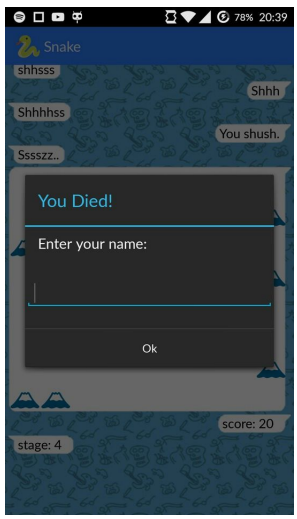
Portaal verschijnt

Andere vijanden

Het doel is om de vijanden te verslaan, waarna er een portaal verschijnt naar een volgend level met meer en/of moeilijkere vijanden. De speler is de kat en beweegt op orthogonale wijze door op het scherm te tikken, waarbij de locatie waar de speler tikt de richting bepaalt. De speler heeft verder ook nog de keuze op dezelfde plek te blijven staan, door op het midden van het scherm te tikken. Elk type vijand heeft zijn eigen icoon en bewegingsmogelijkheden. Zowel de speler als de vijanden kunnen niet op de muren landen (aangegeven door een kleine berg). Doordat de levels en vijanden willekeurig gegenereerd worden is elk level weer een verrassing.
Een vijand wordt verslagen door op het veld waar deze zich bevind; de vijand is dan van het level verwijderd en punten worden toegekend aan de speler. Op dezelfde wijze kan de speler worden verslagen door een vijand.

Meer vijanden

Touch feedback



Game over!

Er zijn in totaal 5 verschillende vijanden:

- De draak (3 punten): beweegt alleen orthogonaal waarbij hij één enkele stap uitvoert
- De rat (1 punt): beweegt alleen diagonaal waarbij hij één enkele stap uitvoert
- De slang (2 punten): wisselt elke beurt af tussen een orthogonale of diagonale beweging van 1 stap. De tong van de slang geeft aan in welke richting hij de volgende beurt zal bewegen, zodat de speler (nadat hij of zij geleerd heeft er op te letten) weet wat de mogelijkheden zijn.
- De wolf (2 punten): beweegt net zoals de draak, tenzij de speler binnen twee vakjes staat. In dat geval springt de wolf 2 stappen in orthogonale richting, waarbij hij over de speler heen kan springen als deze er direct naast staat.
- Het paard(3 punten): maakt paardensprongen zoals in schaken.

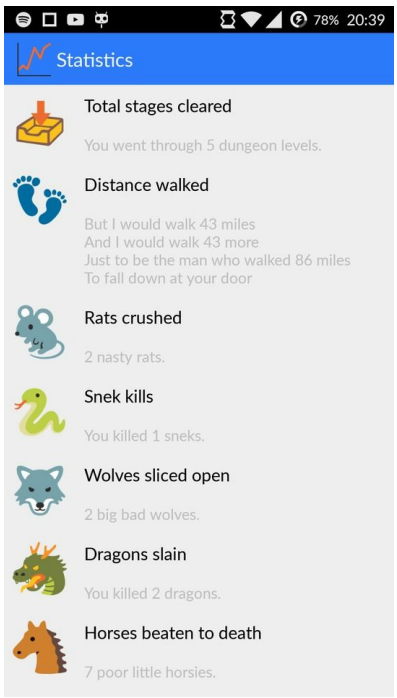
Het spel gaat net zolang door als de speler niet wordt verslagen door een vijand, waarna er een pop-up verschijnt voor het bevestigen van je score door het invullen van een naam voor de high scores. Zodra deze bevestigd is wordt de speler weer teruggebracht naar het hoofdmenu.

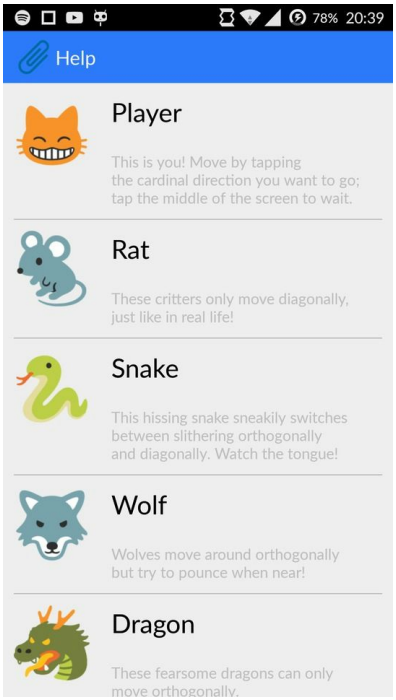
Highscore

Als de gebruiker op ´Highscore´ tikt worden de hoogste 10 scores getoond op het scherm. Samen met de scores worden de namen van spelers, een klein stukje tekst en de datum waarop de score was behaald weergegeven.

De top 10 scores

Stats
Als de gebruiker op ´Stats´ tikt worden statistieken over interacties in het spel getoond. Deze statistieken worden tijdens het spelen opgeteld bij de al bestaande statistieken en opgeslagen.



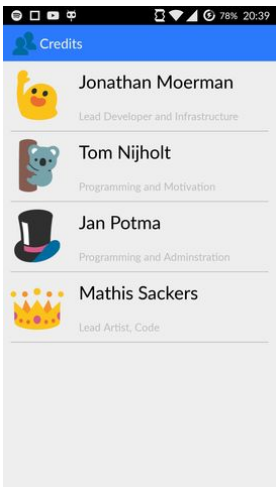


3.2

3.3 Het Statistics venster Het Help venster

Help
Ook is een kleine gids toegevoegd die kan helpen het spel beter te begrijpen.
Hierin wordt uitgelegd hoe de speler de kat kan besturen en wordt van alle vijanden het bewegingsgedrag beschreven.

Credits
Er is ook een kleine ode aan de ontwikkelaars aanwezig in de app.
Hier staan de namen van iedereen die heeft gewerkt aan de app, een leuk icoontje en een kleine indicatie van de rol in het maken van de app.

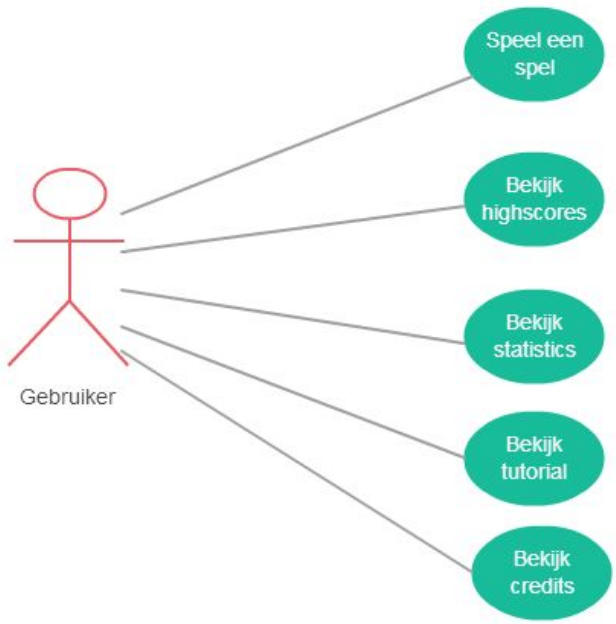


3.4 Productverantwoording

Er zijn legio andere rogue-likes te vinden in de app store, ondanks het feit dat het een ondergerepresenteerd genre is. Waar het zich in onderscheidt is de simpliciteit, toegankelijkheid en afstemming voor android.
Alternatieven zijn vaak veel complexer dan onze app in een poging de beleving van andere platforms zoals PC te emuleren, Ze hebben scrollende kaarten en heel veel items. De

focus van deze apps is vaak ontdekking en niet de puzzelachtige gevechten die we in onze app hebben. De combinatie van rogue-like met schaak-achtige gevechten is uniek.
Bepaalde apps (ook inspiratiebronnen voor ons) komen in de buurt: The Nightmare Cooperative gebruikt tactische bewegingen maar heeft lange sessies en complexiteit door meerdere karakters en items, zoals in de PC versie. Hoplite probeert ook zeer tactische gevechten, maar heeft onhandige invoer en keuzesystemen.
Het voordeel van onze app is dat de lengte van speelsessies zo kort als een minuut kan zijn, omdat de levels niet zo groot zijn en ze dus snel opgelost kunnen worden. Dit is essentieel voor succes bij een androidapplicatie.
De willekeurige generatie zorgt steeds voor verrassing en nieuwigheid; het leren en begrijpen van de mechaniek van het spel zorgt ervoor dat de gebruiker zichzelf steeds probeert te verbeteren - beide zorgen ervoor dat de gebruiker steeds terug wil komen.
De high scores en steeds moeilijker wordende levels zorgen ervoor dat er competitie is met anderen en de gebruiker blijft spelen.

3.5 Specificaties



Use case model

Uitgewerkte use case 'Speel een spel'
Deze use case is gekozen omdat deze ons het meest relevante leek.

Use Case	Speel een spel
Description	In deze use case wordt het spelen van een spel beschreven
Trigger	1. Gebruiker opent de app

	2. Gebruiker drukt op ‘Play’
Basic Course of Events	1. De app genereert een willekeurig level gebaseerd op de hoeveelheid gehaalde levels in deze sessie. 2. De gebruiker klikt op een van de 5 richtingsknoppen en beweegt hiermee de kat. 2.1. Als er op de plek waar de kat op gaat staan eerder een enemy stond worden er punten, die bepaald worden aan het verslagen type enemy, toegekend aan de speler. 2.1.1. Als alle enemies zijn verslagen in een level, dan genereert de app een portaal. 2.2. De speler komt op een portaal terecht. Terug naar event 1, maar nu met een verhoogd aantal levels verslagen. 2.3. De speler heeft een beweging gedaan die geen enemy verslaat. 3. De app laat de resterende enemies bewegen. 3.1. Een van de enemies verslaat de kat, er wordt gekeken naar de behaalde score. 3.1.1. Indien deze hoger is dan 0 wordt er om een naam gevraagd aan de gebruiker. 3.1.1.1. De gebruiker voert een naam in. 3.1.1.1.1. De app slaat de score op in een bestand en voegt deze hiermee toe aan de high scores mits het in de top 10 terechtkomt. Vervolgens stuurt de app de gebruiker terug naar het hoofdmenu. 3.1.2. Indien de score 0 is wordt de gebruiker teruggestuurd naar het hoofdmenu. 3.2. Geen enkele enemy verslaat de kat. Terug naar event 2.
Precondition	-
Postcondition	De gebruiker heeft een spel gespeeld.

4 ONTWERP

4.1 Globaal ontwerp

Doordat we hebben gewerkt volgens het MVC-principe zijn onze klassen op te delen in klassen die op de achtergrond berekeningen uitvoeren, klassen die het spel weergeven aan de speler en klassen waarin de huidige toestand van het level/achtergrondproces bijhouden. Dit zorgt voor een mooie scheiding tussen klassen en houd duidelijk wat elke klasse nou precies doet.

De klassen zijn als volgt onderverdeeld:
Model: Level, Score, StatisticsStorage, Corner, Direction, CellEntities(en diens onderliggende klassen).
View: CreditsActivity, GameActivity, HelpActivity, HighScoreActivity, MainMenuActivity, StatisticsActivity.
Controller: Controller, EnemyController, LevelGenerator.

4.2 Detailontwerp

4.2.1 Activities:

Deze klassen implementeren onCreate, onCreateOptionsMenu en onOptionsItemSelected van de super klasse Activity. Van deze methodes doet alleen onCreate iets wezenlijks, deze methode wordt namelijk gebruikt als constructor van deze klassen.

CreditsActivity
Deze activity geeft onze namen en onze functie binnen dit project weer.

- Geen extra methodes.

GameActivity
Deze activity geeft het spel weer, bevat de controller en beheert de highscores en statistieken.

Publieke methodes:

- **public void setHighScore ():** Deze methode kijkt naar de gehaalde score en als deze in de top 10 van scores valt wordt deze daaraan toegevoegd.
- **public void setStatistics ():** Voegt de opgeslagen statistieken van de huidige speelsessie toe aan de opsomming van alle statistieken
- **public void scorePopup ():** Zorgt voor een pop-up die vraagt om de naam van de speler zodat deze kan worden toegevoegd aan de behaalde score.

HelpActivity
Deze activity geeft onze namen en onze functie binnen dit project weer.

- Geen extra methodes.

HighScoreActivity
Deze activity geeft de behaalde high scores weer.

- Geen extra methodes.

MainMenuActivity
Deze activity geeft het hoofdmenu weer.

Publieke methodes:

- **public void start(View view):** Deze methode zorgt er voor dat een nieuwe GameActivity weergegeven wordt.
- **public void gotoHighscore(View view):** Deze methode zorgt er voor dat een nieuwe HighScoreActivity weergegeven wordt.
- **public void gotoStatistics(View view):** Deze methode zorgt er voor dat een nieuwe StatisticsActivity weergegeven wordt.
- **public void gotoCredits(View view):** Deze methode zorgt er voor dat een nieuwe CreditsActivity weergegeven wordt.

- **public void gotoHelp(View view):** Deze methode zorgt er voor dat een nieuwe HelpActivity weergegeven wordt.

StatisticsActivity
Deze activity geeft de statistieken weer.
- Geen extra methodes.

4.2.2 Views:

GameView
Deze acitivity geeft het level weer in een chat-achtige setting.

Publieke methodes:
- **public void setScore(int score):** Deze methode verandert de weergegeven score.
- **public void setStage(int stage):** Deze methode verandert het getal in het tekstwolkje dat aangeeft welk level op dit moment gespeeld wordt.
- **public void setLevel(Level level):** Deze methode verandert het weer te geven level.
- **public int[] getCoordinates():** Deze methode geeft de coördinaten van de linkerbovenhoek en rechterbenedenhoek van het weergegeven level weer.
- **public void setChat(int chat):** Deze methode verandert de weergegeven conversatie naar een van de andere *voorgeprogrammeerde* conversaties.

TouchOverlay
Deze klasse wordt gebruikt om aan te geven wat de aktie is die uitgevoerd zal worden als de gebruiker een gedeelte van het scherm aanraakt en loslaat.

Publieke methodes:
- **public void clear ():** Deze methode zorgt ervoor dat deze TouchOverlay niets weer geeft.
- **public void drawTouchArea (Corner corner, int[] coordinates, float cSize):** Deze methode geeft aan welk gedeelte van het spelbord wordt aangeraakt, en welke kant de speler dus op zal bewegen. (cSize wordt gebruikt om de groote van het centrum vlak aan te geven.)

4.2.3 Model:

Score
Deze klasse wordt gebruikt om een van de highscores in op te slaan. Deze klasse implementeerd Comparable<Score>.

Publieke methodes:
- **public Score(String name, String date, int num):** Dit is de constructor van deze klasse met als parameters de data die in deze klasse opgeslagen moet worden.
- **public int compareTo(Score sc):** Deze methode vergelijkt deze Score met een andere Score. Deze functie wordt gebruikt om

- **public String getScoreText():** Deze methode geeft de string representatie van deze Score.

StatisticsStorage
Deze klasse wordt gebruikt om statistieken in op te slaan.

Publieke attributen:
- **public int dragonsKilled:** Hoeveel draken de speler verslagen heeft.
- **public int snakesKilled:** Hoeveel slangen de speler verslagen heeft.
- **public int ratsKilled:** Hoeveel ratten de speler verslagen heeft.
- **public int wolvesKilled:** Hoeveel wolven de speler verslagen heeft.
- **public int horsesKilled:** Hoeveel paarden de speler verslagen heeft.
- **public int gamesLost:** Hoeveel keer de speler een spel verloren heeft.
- **public int stagesCleared:** Hoeveel levels de speler weten te voltooien.
- **public int distanceWalked:** Hoeveel stappen de speler heeft gezet.

Publieke methodes:
- **public StatisticsStorage (int snakes, int rats, int wolves, int horses, int lost, int stagesClear, int distance, int dragons):** Dit is de constructor van deze klasse met als parameters de data die in deze klasse opgeslagen moet worden.
- **public void addStatistics(StatisticsStorage add):** telt de statistieken van een andere StatisticsStorage bij deze StatisticsStorage op.
- **public String toString():** Deze methode geeft de string representatie van deze StatisticsStorage.

Corner
Deze enumeratie wordt gebruikt om het aangeraakte deel van het scherm door te geven.

Direction
Deze enumeratie wordt gebruikt om de richting die de speler in wil gaan door te geven.

Level
Deze klasse bevat alle informatie van het spelbord.
- **public Level(int xSize, int ySize, int difficulty):** Dit is de constructor van deze klasse. Als parameters worden de dimensies en de moeilijkheidsgraad doorgegeven.
- **public int getXSize():** Deze methode geeft de horizontale grootte van dit level.
- **public int getYSize():** Deze methode geeft de verticale grootte van dit level.
- **public int getDifficulty():** Deze methode geeft moeilijkheidsgraad van dit level.
- **public void setDifficulty(int newDifficulty):** Deze methode verandert de toegekende moeilijkheidsgraad van dit level.
- **public boolean gameover():** Deze methode geeft aan of de speler verslagen is.
- **public boolean levelCleared():** Deze methode geeft of alle tegenstanders verslagen zijn.
- **public boolean isNextToPlayer(int x, int y):** Deze methode geeft aan of een positie zich direct naast de speler bevind.

- **public boolean isFree(int x, int y):** Deze methode geeft aan of een bepaalde positie een geldige, vrije positie op het spelboord is.
- **public boolean isValid(int x, int y):** Deze methode geeft aan of een bepaalde positie een geldige positie op het spelboord is.
- **public boolean isEnemy(int x, int y):** Deze methode geeft aan of een bepaalde positie op het spelboord door een tegenstander bezet word.
- **public boolean isWall(int x, int y):** Deze methode geeft aan of op een bepaalde positie een muur staat.
- **public boolean isPortal(int x, int y):** Deze methode geeft aan of op een bepaalde positie een muur staat.
- **public Player getPlayer():** Deze methode geeft de Player op dit spelbord, als deze er is.
- **public CellEntity getEntity(int x, int y):** Deze methode geeft, als er een CellEntity staat, de CellEntity die op de gegeven positie staat.
- **public int getEnemyCount():** Deze methode geeft het aantal tegenstanders dat zich op het spelbord bevind.
- **public Enemy getEnemy(int index):** Deze methode geeft een tegenstander uit de interne List die zich op die index bevind. <- is dit goed Nederlands?
- **public void killPlayer():** Deze methode verwijdert de speler van het spelbord.
- **public void clearCell(int x, int y):** Deze maakt een vak op het spelbord vrij.
- **public void moveEntity(int oldX, int oldY, int newX, int newY):** Deze verplaatst een CellEntity binnen het spelbord.
- **public void removeEnemy(Enemy enemy):** Deze methode verwijdert een tegenstander uit de interne List.
- **public boolean addEnemy(Enemy enemy):** Deze methode voegt, indien er plek is, een tegenstander toe aan het spelbord.
- **public boolean addPlayer(Player player):** Deze methode voegt de speler aan het spelbord toe.
- **public boolean addWall(Wall wall):** Deze methode plaatst een muur op het spelbord.
- **public boolean addPortal(Portal portal):** Deze methode plaatst een portaal naar het volgende level op het spelbord.
- **public Portal getPortal():** Deze methode geeft het portaal naar het volgende level, mits er een op het spelbord staat.

CellEntity

Een interface voor alle objecten die op het spelbord kunnen staan.
Behalve de methodes wordt ook de klasse zelf als data gebruikt in het spel (instance of).

- **Drawable getImage(Context context):** Deze methode geeft de afbeelding voor op het spelbord.
- **int getX():** Deze methode geeft de horizontale positie op het spelbord.
- **int getY():** Deze methode geeft de verticale positie op het spelbord.
- **void setCoordinates (int xPos, int yPos):** Deze methode verandert de positie van deze CellEntity op het spelbord.

Wall

Implementeert CellEntity. Is een onbewegelijk obstakel voor de speler.

- **public Wall (int xPos, int yPos):** Dit is de constructor van deze klasse met als parameters de positie op het spelbord waar dit moet spawnen.

Player
In deze klasse wordt de locatie van de speler opgeslagen.
Implementeert CellEntity.
public Player (int xPos, int yPos): Dit is de constructor van deze klasse met als parameters de positie op het spelbord waar dit moet spawnen.

Portal
In deze klasse wordt de locatie van het Portaal opgeslagen.
Implementeert CellEntity.
public Portal (int xPos, int yPos): Dit is de constructor van deze klasse met als parameters de positie op het spelbord waar dit moet spawnen.

Enemy
In deze klasse wordt de locatie van een tegenstander opgeslagen, het geeft alle mogelijke volgende locaties voor deze tegenstander en geeft aan hoeveel punten de speler krijgt voor het uitschakelen van deze tegenstander.
Deze abstracte klasse komt voor in de volgende vormen: Dragon, Rat, Snake, Wolf, Horse.
Implementeert CellEntity.

- **public Enemy (int xPos, int yPos):** Dit is de constructor van deze klasse met als parameters de positie op het spelbord waar dit moet spawnen.
- **public int getPoints():** Deze methode geeft het aantal punten dat een enemy waard is.
- **public abstract LinkedList<int[]> getMoves (int xPlyr, int yPlyr):** Deze methode geeft een LinkedList met locaties waar een enemy naartoe kan gaan vanaf zijn huidige positie.
- **private int[] calcdelta(int x, int y, int xPlyr, int yPlyr):** Deze methode geeft de afstand tussen de player en de enemy.

4.2.4 Klassen die de klassen uit het model beheersen:

Controller
Deze klasse beheert het spel.
Implementeert View.OnTouchListener en Serializable.

- **public Controller (GameActivity ga):** Constructor van de controller, als parameter wordt de gameactivity meegegeven.
- **public boolean onTouch(View v, MotionEvent event):** Deze methode wordt aangeroepen als de gebruiker het scherm aanraakt, als de gebruiker een richting selecteert en vervolgens loslaat, bewegen de speler en tegenstanders.
- **public void addKills(Enemy enemy):** Deze methode werkt de tegenstander statistieken van deze sessie bij.

- **public Level getLevel():** Deze methode geeft het level dat doort deze Controller wordt beheert.
- **public void setLevel(Level level):** Deze methode verandert het te besturen level.
- **public int getScore():** Deze methode geeft de tot nu toe behaalde score.

LevelGenerator
Deze klasse wordt gebruikt om de levels mee te genereren.

- **public Level genLevel (int difficulty):** Genereert een level met de gegeven moeilijkheidsgraad.
- **private Enemy chooseEnemyLVL1 (int x, int y):** Kiest een enemy om te spawnen in level 1.
- **private Enemy chooseEnemyLVL2 (int x, int y):** Kiest een enemy om te spawnen in level 2.
- **private Enemy chooseEnemyLVL3 (int x, int y):** Kiest een enemy om te spawnen in level 3.
- **private int[] getRandomFreeSpace (Level level):** Geeft een willekeurige vrije plek in het gegeven level.
- **private boolean checkClosedSpaces (Level level, int addedX, int addedY):** Controleert dat er geen ontoegankelijke plek is in het level als er een muur wordt toegevoegd bij de gegeven coordinaten addedX en addedY.
- **private int breadthFirstExplore (Level level, int x, int y, int addedX, int addedY):** voert een breadth-first search uit om het aantal vrije plekken te berekenen vanaf lokatie (x, y).

EnemyController
Deze klasse beheert de bewegingen van de enemies.

- **public EnemyController (Level level):** De constructor voor de EnemyController met als parameter het level waarin enemies moeten bewegen.
- **public void setLevel (Level level):** Deze methode verandert het te besturen level.
- **public void moveEnemies ():** Beweegt alle enemies in het level.
- **public void sortMoves (List<int[]> moves):** Sorteert de gegeven lijst van moves.

4.3 Ontwerpverantwoording

We passen het MVC-principe, wat de standaard in de industrie is. Bepaalde klassen zijn bedoeld om de toestand op te slaan, andere voor het weergeven van de huidige toestand, en nog andere voor het aanpassen van de toestand. Dit zorgt voor een duidelijke structuur zonder complexe rollen, die ook makkelijk aanpasbaar is.
Door het gebruik van activities werkt de native terugknop naar behoren.

4.3.1 Technisch detail: levelgeneratie

Een goede, eerlijke level generatie is erg van belang voor een rogue-like, maar ook heel moeilijk om ideaal te implementeren. Wij hebben geprobeerd een paar van de grootste valkuilen de omzeilen.

Ten eerste moet elk vakje bereikbaar zijn - de speler mag niet ingesloten worden door muren. Om dit te bereiken hebben we een aangepaste breadth-first search geïmplementeerd die alle vakjes binnen bereik af gaat, om te controleren dat de speler overal kan komen.
Ten tweede moeten de levels gevarieerd zijn, en (in ons geval) ten derde steeds moeilijker worden. Hiervoor hebben we de reeds gebruikte punten gebruikt. Een level heeft een bepaalde moeilijkheidsgraad, die steeds met 2 omhoog gaat. Dit bepaalt de grootte van het level: de eerste paar levels zijn klein om te wennen en leren, de latere zijn groter en bevatten meer vijanden. Het puntentotaal van de vijanden in een level is gelijk aan de moeilijkheidsgraad van het level: zo kan een level van difficulty 4 een paard en een rat bevatten, maar ook twee slangen.
Ten vierde moeten de levels eerlijk zijn. Dit is heel moeilijk, idealiter wordt gecontroleerd of het level haalbaar is. Echter, dit is een enorm probleem, te vergelijken met de computeranalyse van schaak. Dit duurt in de praktijk veel te lang om fijn te spelen te zijn. Wij hebben gekozen voor een simpele oplossing, die misschien niet compleet is maar wel snel: er mag nooit een vijand naast de speler spawnen. Op deze manier heeft de speler vrijwel altijd een aantal veilige startbewegingen, waar als het goed is minstens een weer een veilige beweging oplevert. Deze methode is niet perfect, maar het overgrote deel van de generatie is eerlijk.

4.3.2 Ontwerpbeslissing: bewegingsmogelijkheden speler

De speler moet tegen een enorme overmacht van vijanden spelen, en op de een of andere manier toch een goede kans maken te winnen. Omdat we met onze gekozen gevechtsmethode (naar voorbeeld van schaken) geen levens of health konden gebruiken moest de speler een logistiek voordeel hebben. Dit heeft voor veel overleg gezorgd: een extra beurt voor de speler, extra grote en verre bewegingsmogelijkheden waren slechts een paar van de suggesties. Een van de ideeën was de mogelijkheid voor de speler om stil te staan. Dit werd aanvankelijk afgedaan omdat het te weinig impact zou hebben. Tijdens het testen van ons concept op papier en op de computer (voordat er een werkend prototype was) hebben we dit toch uitgeprobeerd, voornamelijk omdat het heel simpel te testen was. Toen bleek dat het toch heel veel mogelijkheden gaf om slimmer te bewegen dan de vijanden, die altijd moeten bewegen. Ook loste dit het probleem op dat ontstond wanneer er alleen nog maar een draak diagonaal ten opzichte van de speler was: dan had de speler geen kans. In combinatie met niet al te snuggere vijanden heeft deze keuze ervoor gezorgd dat de speler toch erg goede kans maakt te winnen.

5 REFLECTIE

5.1 Wat ging er goed?

Wij zijn heel tevreden met het resultaat. Het concept van de app was heel goed. De app is een goede luchtige game voor veel verschillende tijdsduren. Het bewegen van de speler werkt perfect en is intuïtief. De statistieken waren goed te implementeren en maken de hele

app iets interessanter en meer compleet. Door de willekeurige generatie van levels is iedere playthrough interessant en de speler is gemotiveerd vaker te spelen om zichzelf te verbeteren en meer punten te krijgen.

5.2 Wat ging er minder goed?

Alhoewel de app in het geheel wel een mooie algemene ervaring biedt, zijn er wel een aantal dingen die beter kunnen of ontbreken. De highscore lijst wordt b.v. alleen lokaal opgeslaan en is dus niet echt interessant om met vrienden te spelen. De game-screen is optisch ook nog niet perfect. Aan de onderkant van het scherm is nog ruimte om bijvoorbeeld meer tekstballonen te plaatsen. De tekst in de tekstballonen zelf is ook (afhankelijk van de grote van het scherm) een beetje raar. Soms is de level generatie ook niet compleet eerlijk en spawnen tegenstanders en muren op een manier, dat de speler in ieder geval dood gaat.
Het grootste probleem is waarschijnlijk dat er geen animaties zijn. Voor een nieuwe gebruiker is het niet meteen duidelijk, dat dit spel turn-based is, omdat de tegenstanders tegelijk met de speler bewegen. Een ander probleem voor nieuwe gebruikers is, dat er geen tutorial is. Er is wel een hulp, maar die is alleen tekst en geeft alleen informatie over de verschillende tegenstanders en niet hoe je speelt. Het helpt wel een beetje dat de eerste levels kleiner zijn en dus niet zo complex.

5.3 Conclusie

We zijn tevreden met het resultaat, hoewel we het jammer vinden dat we geen tijd meer hadden om animaties of een tutorial toe te voegen. Dit kwam vooral doordat we veel tijd hebben gestoken in de level generatie.
Wij hebben veel geleerd van het maken van deze app. Doordat Mathis de enige in onze groep was die al met Android gewerkt heeft en Jonathan de enige was die al wist hoe GitHub werkt moesten de andere groepsleden een beetje tijd steken in het leren van Android en het werken met GitHub. Maar door gebruik van Google en de documentatie van Android zijn alle problemen redelijk snel opgelost geweest.