Ultimate Subset

Door Sijmen Huizenga

Versie alpha 3

Inhoud

[Inleiding 3](#_Toc401217878)

[Set 4](#_Toc401217879)

[Hoe werkt de game? 4](#_Toc401217880)

[Eigenschappen? 4](#_Toc401217881)

[Wat is een set? 4](#_Toc401217882)

[Functionaliteit 5](#_Toc401217883)

[Menu 5](#_Toc401217884)

[Game 5](#_Toc401217885)

[Scorebord 6](#_Toc401217886)

[About 6](#_Toc401217887)

[Sneltoetsen 6](#_Toc401217888)

[Systeem 7](#_Toc401217889)

[Componenten 7](#_Toc401217890)

[Schermen 7](#_Toc401217891)

[Knoppen 7](#_Toc401217892)

[Game 8](#_Toc401217893)

[Testen 9](#_Toc401217894)

[Functies en Variable 10](#_Toc401217895)

[Bijlage 14](#_Toc401217896)

[Schermontwerp menu 14](#_Toc401217897)

[Schermontwerp game 15](#_Toc401217898)

[Schermontwerp scoreboard 16](#_Toc401217899)

[Schermontwerp about 17](#_Toc401217900)

# Inleiding

In dit rapport is een plan gemaakt voor het programma Ultimate Subset. Er zijn vier onderdelen. Ten eerste is er een uitleg over de game SET. Daarna is in het hoofdstuk analyse alle requirements beschreven. Hier is dus alle functionaliteit van het programma te vinden. Daarna is er een ontwerp met alle structuren, variabele en methode. Dit hoofdstuk is vooral van belang voor programmeurs. Als laatste is er een testplan waarin wordt beschreven hoe na ontwikkeling het programma de functionaliteit getest kan worden.

# Set

## Hoe werkt de game?

In beeld heb je een aantal kaarten. Dit aantal kaarten varieert tussen 9 en 12 afhankelijk of je het originele of een versimpelde verzie speelt. De bedoeling is om drie kaarten te selecteren die een ‘set’ vormen. Als deze kaarten een set vormen, worden deze kaarten van het bord verwijdert en komen er nieuwe kaarten uit de stapel.

De stapel bevat in het begin van het spel alle mogelijke kaarten(27 of 81). Er worden steeds een aantal op tafel gelegd. Vervolgens pakt de speler een set totdat er geen sets meer op tafel liggen of totdat de stapel leeg is. Elke keer als de speler een kaart pakt, wordt het puntenaantal met 1 verhoogd. Het aantal punten is de score van de speler. Dit gaat door totdat alle kaarten op zijn of totdat er geen set meer mogelijk is.

## Eigenschappen?

Elke kaart heeft een aantal eigenschappen. In de verkleinde versie zijn er 3 eigenschappen:

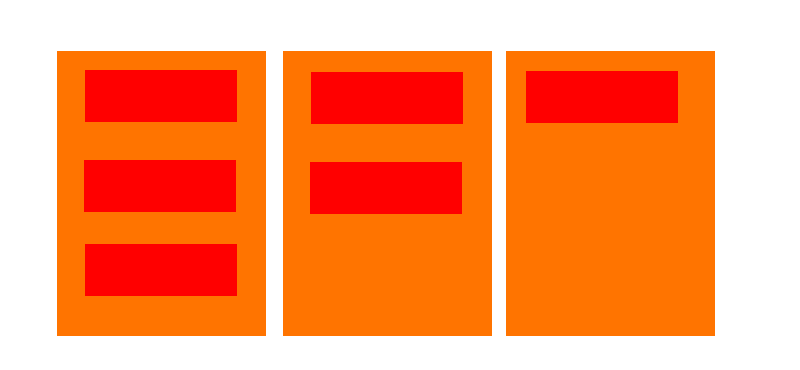
* De hoeveelheid (1, 2, 3)
* De vorm (rechthoek, driehoek, ovaal)
* De kleur van de vorm (rood, geel, blauw)

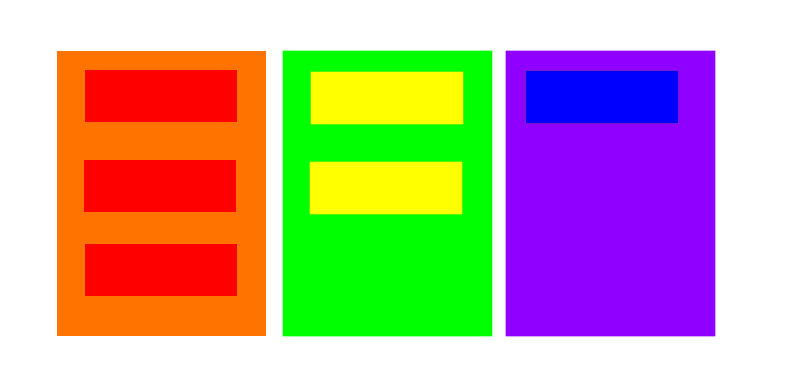
In de originele versie is er nog een extra eigenschap:

* De kleur van de kaart (oranje, groen, paars)

Alle mogelijke kaarten zijn online te vinden: [27 kaarten](https://github.com/SijmenHuizenga/SubSet/blob/master/afbeeldingen/27Kaarten.png) en [81 kaarten.](https://github.com/SijmenHuizenga/SubSet/blob/master/afbeeldingen/81Kaarten.png)

## Wat is een set?

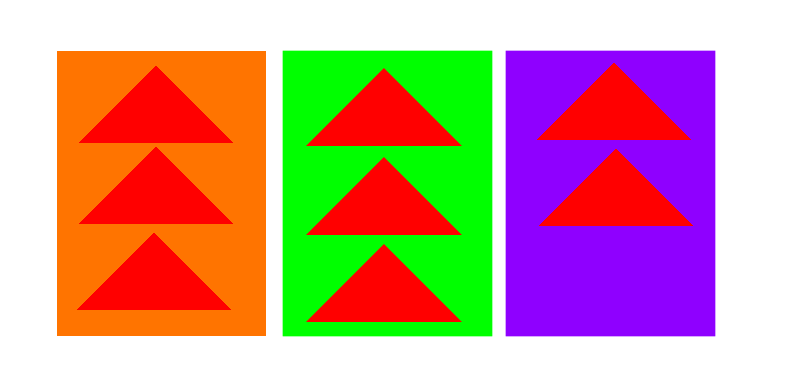
Een set heeft 3 kaarten. Een set kenmerkt zicht dat binnen elke eigenschappen alles verschillend is, of alles anders is. Hier zijn wat voorbeelden:

**voorbeeld 1**  
kleur: rood, rood, rood (overal zelfde)   
achtergrond kleur: oranje, oranje, oranje (overal zelfde)   
vorm: rechthoek, rechthoek, rechthoek (overal zelfde)   
hoeveelheid: 3, 2, 1 (overal anders)  
Dus dit is een valide set.

Voorbeeld 1

**voorbeeld 2**  
Kleur: rood, geel, blauw (overal anders)   
Achtergrond kleur: oranje, groen, paars (overal anders)   
Vorm: rechthoek, rechthoek, rechthoek (overal gelijk)   
Hoeveelheid: 3, 2, 1 (overal anders)   
Dus dit is een valide set.

Voorbeeld 2

**voorbeeld 3**  
kleur: rood, rood, rood (overal anders)  
achtergrond kleur: oranje, groen, paars (overal anders)  
vorm: driehoek, driehoek, driehoek (overal gelijk)  
hoeveelheid: 3, 3, 2 (*deels gelijk, deels anders*)  
Dus dit *niet* een valide set.

voorbeeld 3

# Functionaliteit

Zoals te zien is in de scherm ontwerpen, komen er vier schermen. Menu, score bord, about en game. In deze vier hoofdstukken leg ik uit wat er binnen deze schermpjes gebeurt. Daarna is er een opsomming te vinden van alle sneltoetsen.

## Menu

Het menu is het beginscherm van het programma. Vanuit hier kan de speler een aantal verschillende acties ondernemen. Ten eerste kan de speler via de twee witte knoppen het spel starten. Door op een van deze knoppen te drukken word het ‘Game’ scherm geopend. Bij ‘Start Easy Mode’ wordt de game op makkelijke modus gestart, en bij ‘Sart Original Mode’ wordt de game op originele moeilijkheidsgraad gestart.

Door op de knop ‘ScoreBoard’ te klikken wordt het score bord geopend. Scherm ‘Scoreboard’ wordt dan dus gestart. Door op de knop ‘About and Rules’ te klikken wordt het scherm ‘About’ weergegeven.

Als laatste is er nog de knop ‘Load saved Game’. Deze knop is alleen zichtbaar als er een opgeslagen game is. Door op deze knop te drukken wordt deze opgeslagen game gestart. Je komt dan uit in het Game scherm.

## Game

Het game scherm heeft een aantal onderdelen. Ten eerste zijn er statistieken. Deze zijn te vinden linksboven in het scherm. Deze statistieken geven bijvoorbeeld de huidige tijd, aantal kaarten op de stapel en de highscore weer.

Daarnaast heb je rechts het speelvlak. Hier liggen 9 of 14 kaarten, afhankelijk of je simpel of original speelt. Door met je muis te klikken en te slepen, kan je de kaarten van locatie veranderen. Op deze manier kan je de kaarten op een volgorde leggen die voor jou makkelijk is.

Als je denkt dat je een set hebt gevonden, dan kan je de kaarten een voor een aanklikken, en dan springen ze naar de vakken linksonder in het scherm. Als je een verkeerde kaart heb aangeklikt, kan je linksonder op de kaart klikken, en dan springt hij weer terug naar het scherm. Als er drie kaarten liggen, dan verschijnt er onder de drie hokjes een knop. Deze knop kan verschillende tekst hebben:  
Zijn de kaarten een set? Dan zegt de knop ‘Valid set: Hand In’.  
Zijn de kaarten geen set? Dan zegt de knop ‘Invalid set: Put back’  
In het eerste geval kan je de kaarten inleveren, en worden er drie nieuwe kaarten in het speelveld gelegd. Bij het tweede geval kan je op de knop drukken om de geselecteerde kaarten terug te leggen in het speelveld.

Verder zijn er nog vier knoppen die een actie uitvoren. Van boven naar beneden:  
‘**Save & Quit**’: Sla het spelletje op en sluit het spel af. Er kan altijd maar één spel zijn opgeslagen. Als er al een spel is opgeslagen wordt deze verwijdert.  
‘**Order Cards**’: Alle kaarten in het speelveld worden op volgorde gelegd zodat het speelveld weer overzichtelijk wordt.  
‘**Hint**’: Als de vakjes linksonder leeg zijn, dan worden er twee kaarten vanuit het speelveld gepakt en in de vakjes linksonder gelegd. Dit kost de speler één minuut. Als er al twee kaarten liggen wordt de derde er bij gezocht om de set af te maken. Dit kost se speler twee minuten.  
‘**Give Up**’: De speler geeft op. De game wordt afgesloten, de score wordt niet genoteerd sinds het spel niet is afgerond.

Op een gegeven moment is de stapel op en zijn alle set’s gevonden: Het spel is afgelopen. Er wordt een popup schermpje weergegeven met de mededeling dat het spel is afgelopen. Hier kan de speler zijn of haar naam invoeren voor het score bord. De score wordt samen met de naam van de speler genoteerd op het scorebord. Nu wordt het menu scherm weergegeven.

## Scorebord

Op het scorebord wordt de top vijf beste scores weergegeven. Er is een aparte ranking voor Easy en Original mode. Hoe dit er uit ziet is goed te zien in de schermontwerpen. Verder zijn er nog twee knoppen. De eerste genaamd ‘Clear Scores’ verwijdert alle score lijsten. De tweede knop ‘Back to Menu’ verwijst terug naar het menu scherm.

## About

Op deze pagina is te vinden hoe het spel werkt, hoe je het spel moet spelen, welke dingen van belang zijn enzovoort. Waarschijnlijk is één pagina niet groot genoeg, en moet er een scrollbar geplaatst worden zodat er gescrold kan worden om meer tekst weer te geven. De enige knop die hier aanwezig is de knop ‘Back to Menu’, deze knop verwijst naar het menu.

## Sneltoetsen

Er zijn een aantal sneltoetsen die een bepaalde actie uitvoeren. Hieronder staat een lijstje:

|  |  |
| --- | --- |
| Q | Stop het programma. Als je in het game scherm bent wordt je game eerst opgeslagen. |
| S | Screenshot. Maak een schermafbeelding. |

# Systeem

Je kan het programma verdelen in twee onderdelen: Components en Game. Met components bedoel ik alle knoppen, popup’s invoervakjes en verschillende schermen. Als er bijvoorbeeld een knop op het hoofdmenu staat ‘start game’, en je klikt op deze knop, dan wordt deze klik gehandeld door het components gedeelte van het programma. De game zijn de game mechanieken die het spel verzorgen. Zo houdt de game bijvoorbeeld de punten in de gaten.

## Componenten

Binnen het componenten gedeelte zijn er weer drie onderdelen: Het scherm gedeelte. Dit zorgt voor de achtergrond van de schermen.

### Schermen

Elk scherm(menu, game, scores, about) heeft een nummer. Door middel van dit nummer wordt bijgehouden welk scherm op dit moment wordt weergegeven. Om de schermen aan de nummer te linken, zijn er vier variabele die aangeven welk nummer bij welk scherm hoort. Dit zijn de vier variabele die beginnen met “SCREEN\_”.

Om aan te geven welk scherm nu wordt weergegeven wordt de variabele *selectedScreen* gebruikt.De standaard waarde van deze variabele is SCREEN\_MENU omdat het eerste scherm wat wordt weergegeven als het spel wordt opgestart het menu is.

Als dan de het scherm getekend moet worden, wordt naar de *selectedScreen* variabele gekeken en wordt deze gebruikt om de juiste draw functie aan te roepen(*drawMenu*, *drawGame*, *drawScores* en *drawAbout*). Dit wordt gedaan in de functie *drawScreen*().

### Knoppen

Naast schermen zijn er knoppen. Zoals te zien is in de schermontwerpen zijn er erg veel verschillende knoppen(ongeveer 12). Het is onmogelijk om voor elk van deze knoppen een aparte x, y, breedte en hoogte bij te houden. Daarom gebruik ik en 2D array genaamd *buttonData*. Elk array binnen dit array stelt een knop voor. Zoals te zien is in tabel 1, heeft elke locatie een eigen betekenis. Om deze locatie en betekenis makkelijk aan elkaar te linken zijn er de variabele die beginnen met “BUTTON\_” in het leven geroepen. Als je bijvoorbeeld de achtergrond kleur van de een knop wil weten, dan gebruik je deze code:  
*int kleur = knop[BUTTON\_BGCOLOR];*

Elke knop heeft een identificatie nummer, zodat ze makkelijk zijn terug te vinden. Er is hierbij een regel: alle identificatie nummers tussen de 100 en de 200 zijn gereserveerd voor kaart-knoppen(zie Game). Om makkelijk een knop te vinden, is er de functie *getButtonLocation(int id).* Deze zoekt naar de knop met het gegeven identificatie nummer en geeft een verwijzing naar de knop door middel van een int. Deze int stelt dan de locatie int het *buttonData* array voor.

Om te zorgen dat elke knop ook een tekst heeft, is er een String array genaamd *buttonText*. In dit array staan alle tekst gegevens van de knoppen opgeslagen. De locaties van buttonData en buttonText komen overeen. Dus als je de locatie van een button in *buttonData* hebt, heb je ook de locatie van de tekst in *buttonText*. Als in de tekst van een knop een sterretje (“\*”) staat, dan krijgt de knop twee sterretjes aan de linker en rechter kant van de knop. Zie schermontwerp scoreboard knop. Het karakter \* wordt niet getekent.

Op het moment dat ergens in het scherm geklikt wordt, worden bij alle knoppen die in het huidige scherm aanwezig zijn gekeken of er op die knop gedrukt wordt. Dit wordt gecheckt in de functie *handleMouseClick()*. Als er een knop is die ingedrukt wordt, dan wordt de functie *doButtonAction(int id)* aangeroepen. Deze functie heeft als argument een identificatie nummer. Aan de hand van dit nummer wordt de juiste actie uitgevoerd. Er zijn een heel aantal acties waaruit gekozen kan worden. Deze zijn weergegeven in tabel 2.

## Game

Het game gedeelte bevat alle informatie over het spel. Tijdens dit hoofdstuk ga ik er vanuit dat duidelijk is hoe het spel werkt, en wat de verschillen zijn tussen simple en original versie.

|  |  |  |
| --- | --- | --- |
| R | Red | De kleur van het figuur is rood. |
| B | Blue | De kleur van het figuur is blauw. |
| Y | Yellow | De kleur van het figuur is geel. |
| E | Ellipse | De vorm is een ovaal. |
| Q | Quad | De vorm is een vierkant. |
| T | Triangle | De vorm is een driehoek. |
| 1 | One | Er staat één vorm op het kaartje. |
| 2 | Two | Er staan twee vormen op het kaartje. |
| 3 | Three | Er staan drie vormen op het kaartje. |
| O | Orange | De kleur van de achtergrond is oranje. |
| G | Green | De kleur van de achtergrond is groen. |
| P | Purple | De kleur van de achtergrond is paars. |
| N | None | Er is geen achtergrondkleur. (simple version) |

Ten eerste zijn er kaarten. Om een kaart met zijn eigenschappen weer te geven wordt een String gebruikt. In deze analyse wordt een string die een kaart voorstelt een Kaart genoemd. Een Kaart bevat ten alle tijden 4 karakters. Elk karakter geeft een eigenschap aan. Hiernaast is in een tabel weergegeven wat de karakters betekenen. Een Kaart bevat dus één karakter uit het bovenste vak, een uit het tweede vak enzovoort. Een voorbeeld van een Kaart zou kunnen zijn “ROQ1” of “BPQ3”.

Om een stapel kaarten weer te geven wordt een String array gebruikt. Om een volledige set te genereren wordt de functie getCardStack(boolean simple) gebruikt. Deze functie geeft een String[] terug met 27 of 81 kaarten. Dit hangt af van de boolean die is ingegeven. Bij waar wordt een stapel gemaakt voor een simpele game, bij onwaar wordt een stapel gemaakt die gebruikt kan worden bij de originele game.

Wat misschien is opgevallen is dat bij simple mode, de vierde eigenschap altijd naar 4 wordt gezet. Deze 4 refereert naar de N van None. Dus bij simple mode is de achtergrond kleur altijd leeg. Het komt er dus op neer dat alle functies die gemaakt zijn voor 4 eigenschappen, ook bruikbaar zijn voor kaarten uit simple mode. Een voorbeeld is de functie die checkt of 3 kaarten een set zijn. Omdat de achtergrondkleur bij simple mode allemaal gelijk is, is in dat opzicht een set.

Om te zorgen dat de speler niet alle kaarten op volgorde voorgeschoteld krijgt, moet de stapel geschud worden. Dit gebeurt door de functie *shuffleArray*. Deze neemt als argument een String array, en geeft een geschud array terug. Om het array te schudden gebruik ik het “Fisher–Yates shuffle” algoritme. Dit algoritme is [hier](http://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle) te vinden.

Nu gaat de game gespeeld worden. Eerst moeten er 9 of 14 kaarten in het speelveld gedeponeerd worden. Het aantal kaarten wordt opgeslagen in de integer variabele

## Bestandtype

Er worden op twee plekken in het programma data opgeslagen. Hiervoor zijn ook twee bestandstype voor nodig.

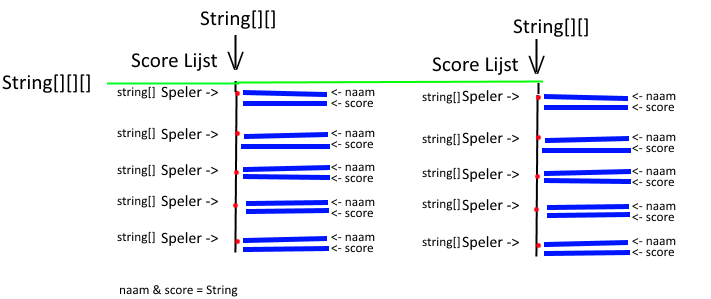
**.set**time: 3:40  
sets: 2  
wrongSets: 5  
cardsOnScreen: RON1;RON2;RON3;BQN1;RQN2;RQN3;PTY1;PTY2;PTY3cardsInStack: RON1;RON2;RON3;BQN1;RQN2;RQN3;PTY1;PTY2;PTY3

**.scores**  
Sijmen;2:40;Bram;2:50;David;3:00;Max;4:20;Ramon;5:50  
Sijmen;2:40;Bram;2:50;David;3:00;Max;4:20;Ramon;5:50

In een .scores bestand staan altijd twee regels. Elke regel bevat een even aantal stukjes. Deze stukjes zijn gescheiden door een puntkomma. Dit betekent dat de tekst die je opslaat nooit een puntkomma mag bevatten. Je kan steeds groepjes maken van twee stukjes. In het eerste stukje staat een naam, en in het tweede stukje staat een score. In het derde stukje staat weer een naam en in het vierde stukje weer een score.

Elke regel in het bestandje staat één score lijstje. Bij dit programma is de eerste regel de scores van de simpele versie, en de tweede regels is de scores van de originele versie.

## Scorebord

Het scorebord wordt opgeslagen in een drie dimensionaal array met tekst. In de afbeelding hieronder is te zien hoe dit array is ingedeeld.  


Om met het score bord te werken, zijn er een aantal functies. Ten eerste heb je de *initScoreBoard* functie. Deze moet aan het begin van het programma worden aangeroepen. Binnen deze functie wordt gekeken of het score bestand bestaat. Als het bestand bestaat wordt hij geladen, als hij niet bestaat wordt een nieuw scorebord aangemaakt.

Het laden van scores wordt gedaan door de functie *loadScoreBoard*. Deze heeft als argument een String die de bestandslocatie voorstelt. Deze functie geeft een drie dimensionaal String array terug met daarin alle scores.

Als tegenhanger van *loadScoreBoard* is er *saveScoreBoard*. Deze functie slaat de scores op in een bestand op de computer. Als argument heeft deze functie een String die de bestandslocatie voorstelt en een drie dimensionaal array met de scores die opgeslagen moeten worden.

# Testen

Om uit te vinden of het programma goed werkt, moet er getest worden. Hieronder zijn alle testcases weergegeven.

|  |  |  |
| --- | --- | --- |
| Naam | Invoer | Uitvoer |
| Knop | Er wordt een knop ingedrukt. | De juiste actie wordt uitgevoerd. |
| Random | Print een geschud array met kaarten. | Dit array moet alle kaarten bevatten en de volgorde moet onvoorspelbaar zijn. |
| Scorebord | Er is een spel afgelopen. | De score is te vinden in het scorebord |
| Score volgorde | Bekijk het scorebord | De eerste plaats op het score bord heeft een beter score dan de tweede plek. |
| Set teller | Start de game in simpel mode, tel het aantal mogelijke sets | Het aantal getelde sets is gelijk aan het getal ‘possible sets’ |
| Enzovoort… |  |  |
|  |  |  |

# Functies en Variable

final int SCREEN\_MENU = 0;  
final int SCREEN\_GAME = 1;  
final int SCREEN\_SCORES = 2;  
final int SCREEN\_ABOUT = 3;

final int BUTTON\_ID = 0;  
final int BUTTON\_SCREEN = 1;  
final int BUTTON\_X = 2;  
final int BUTTON\_Y = 3;  
final int BUTTON\_WIDTH = 4;  
final int BUTOTN\_HEIGHT = 5;  
final int BUTTON\_BGCOLOR = 6;  
final int BUTTON\_FGCOLOR = 7;

final int COL\_RED = color(228, 33, 31);  
final int COL\_BLUE = color(53, 76, 152);  
final int COL\_YELLOW = color(245, 230, 5);  
final int BG\_ORANGE = color(244, 150, 16);  
final int BG\_GREEN = color(79, 169, 45);  
final int BG\_PURPLE = color(107, 55, 138);  
final int BG\_NONE = color(200, 200, 200);

final char C\_COL\_RED = ‘R’;  
final char C\_COL\_BLUE = ‘B’;  
final char C\_COL\_YELLOW = ‘C’  
final char C\_SHAPE\_ELLIPSE = ‘E’;  
final char C\_SHAPE\_QUAD = ‘Q’;  
final char C\_SHAPE\_TRINAGLE = ‘T’;  
final char C\_AMOUNT\_1 = ‘1’;  
final char C\_AMOUNT\_2 = ‘2’;  
final char C\_AMOUNT\_3 = ‘3’;  
final char C\_BG\_ORANGE = ‘O’;  
final char C\_BG\_GREEN = ‘G’;  
final char C\_BG\_PURPLE = ‘P’;  
final char C\_BG\_NONE = ‘N’

int buttonAmount = 3;  
int backgroundColor = 0;  
int selectedScreen = SCREEN\_MENU;  
boolean forceScreenUpdate = true;

String[] cardStack = null;

String[][] buttonData = new buttonData[buttonAmount][8];

Tabel 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Screen | X | Y | Width | Height | Background Color | Font Color |
| 1 | SCREEN\_MENU | 100 | 300 | 300 | 200 | # FFFFFF | # 00 |
| 2 | SCREEN\_MENU | 500 | 300 | 300 | 200 | # FFFFFF | # 00 |
| 101 | SCREEN\_MENU | 200 | 600 | 500 | 150 | # A00000 | # FFFAFF |
| NULL |  |  |  |  |  |  |  |

Als een row binnen buttonData null is, dan moet deze row genegeerd worden.

String[] buttonText = new String[buttonAmount];

void drawScreen(){  
 swich(selectedScreen){  
 case: SCREEN\_MENU: drawMenu(); break;  
 case: SCREEN\_ GAME: drawGame(); break;  
 case: SCREEN\_ SCORES: drawScores(); break;  
 case: SCREEN\_ ABOUT: drawAbout(); break;  
 }  
}

int getButtonLocation(int id){  
 zoek binnen buttonData naar een button met een identificatienummer wat gelijk is aan id. Als deze gevonden is wordt de locatie terug gegeven. Als hij niet gevonden is wordt -1 terug gegeven.  
}

void mouseClicked()  
 for(int[] but : buttonData){  
 if(but[BUTTON\_SCREEN] == selectedScreen){  
 if(mouseX > but[BUTTON\_X] && mouseX < (but[BUTTON\_X]+but[BUTTON\_WIDTH])  
 && mouseY > but[BUTTON\_Y] && mouseY < (but[BUTTON\_Y] + but[BUTOTN\_HEIGHT])){  
 doButtonAction(but[BUTTON\_ID];  
 }  
 }  
}  
}

*doButtonAction(int buttonID){  
 swich(buttonID){  
 case 1: startGame(false); break;  
 case 2: startGame(true); break;  
 case 3: showScoreScreen();break;  
 case 4: showAboutScreen();break;  
 case 5: loadGame();break;  
 case 6: backToMenu();break; //deze staat er twee keer in omdat er twee verschillende   
 case 7: backToMenu();break; // knoppen zijn die deze actie uitvoeren.  
 case 8: clearScores();break;  
 case 9: saveAndQuit();break;  
 case 10: orderCards();break;  
 case 11: hint();break;  
 case 12: giveUp();break;  
 case 12: validInvalidSet();break;  
 }  
 if(buttonID >100 && buttonID <200)  
 cardClickedAction(buttonID);  
}*

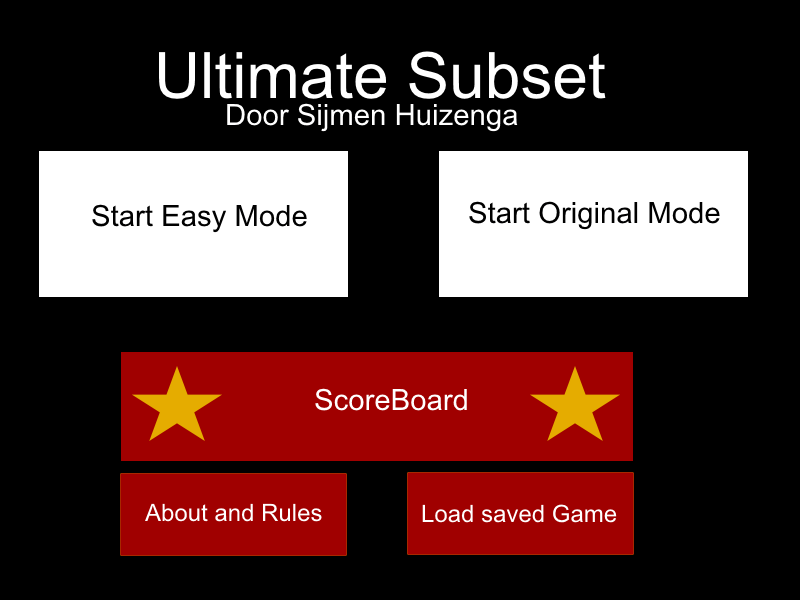
Tabel 2

|  |  |
| --- | --- |
| **Functienaam** | **Wat doet die?** |
| *startGame* | Start het spel. Een boolean argument geeft aan of subset of set gestart moet worden. |
| *showScoreScreen* | Laat het score bord zien. |
| *showAboutScreen* | Laat het about scherm zien. |
| *loadGame* | Laad een game vanuit de harde schijf. Daarna het gelade spel starten. |
| *backToMenu* | Laat het menu zien. |
| *clearScores* | Verwijder alle scores uit het score menu. |
| *saveAndQuit* | Sla het spel op en sluit het programma |
| *orderCards* | Organizeer de kaarten op het speelveld |
| *hint* | Geef een hint door twee kaarten te selecteren. |
| *giveUp* | Geef op en ga terug naar het menu |
| *validInvalidSet* | Doe een dubele actie. Zie hoofdstuk Functionaliteit paragraaf Game derde alinea. |
| *cardClickedAction* | Als een kaart geklikt is, moet er een speciale actie worden ondernomen. Deze actie krijg dus een aparte functie. Daarom moet ook als parameter het button id worden meegegeven. |

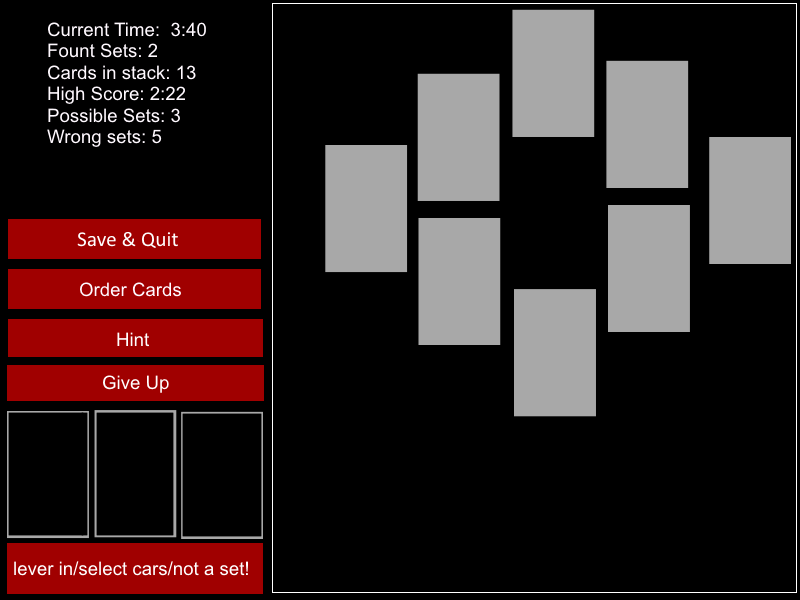
String[] getCardSet(boolean simple){  
String[] out = new String[simple ? 27 : 81];  
 int counter = 0;  
 for (int i = 1; i <=3; i++) {  
 for (int j = 1; j <=3; j++) {  
 for (int k = 1; k <=3; k++) {  
 if(simple){  
 out[counter] = makeCardString(i, j, k, 4);  
 counter++;  
 }else{  
 for(int l = 1; l <=3; l++){  
 out[counter] = makeCardString(i, j, k, l);  
 counter++;  
 }  
 }  
 }  
 }  
 }  
return out;  
}

# Bijlage

## Schermontwerp menu



## Schermontwerp game



## Schermontwerp scoreboard



## Schermontwerp about

