Ultimate Subset

# Functionaliteit

Zoals te zien is in de future screen shots, komen er vier schermen. Menu, Scoreboard, About, 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 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 future screenshot. 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 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: Component en Game. Met component bedoel ik alle knoppen, checkboxes, invoervakjes en verschillende schermpjes. Als er bijvoorbeeld een knop op het hoofdmenu staat ‘start game’, en je klikt op deze knop, dan wordt deze klik gehandeld door het component gedeelte van het programma. De game zijn de game mechanieken die het spel verzorgen. Zo houdt de game bijvoorbeeld de punten in de gaten.

## Component

Laten we bij het begin beginnen. 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:

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

Om aan te geven welk scherm nu wordt weergegeven gebruik je: **int selectedScreen** = SCREEN\_MENU;

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.  
**swich(selectedScreen){  
 case:** SCREEN\_MENU: drawMenu(); **break;  
 case:** SCREEN\_GAME: drawGame(); **break;  
 case:** SCREEN\_SCORES: drawScores(); **break;  
 case:** SCREEN\_ABOUT: drawAbout(); **break;  
}**

Naast schermen zijn er knoppen. Zoals te zien is in de ‘future screenshots’ 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:

Int buttonAmount = 3;  
**String[][] buttonData = new buttonData[**buttonAmount**][**8**];**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

Omdat het verwarrend kan zijn welke locatie in het array corresponderen met welke eigenschap, zijn er een aantal variabelen die dit aangeven:

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

Om handig met dit array te kunnen werken is er een functie in het leven geroepen die een button locatie te vinden aan de hand van ID nummer:

**int getButtonLocation(int id){  
 int i = 0;  
 for(int[] but :** buttonData**){  
 if(but[**BUTTON\_ID**] == id)  
 return but;  
 i++;  
 }  
 return -1;  
}**Zoals te zien is, wordt bij een invalide id -1 teruggegeven.

Om te zorgen dat elke knop ook een tekst heeft, moet er een String array komen:

**String[] buttonText = new String[**buttonAmount**];**

In dit array is de locatie gelijk aan de locatie in het buttonData array. Dus als je de locatie van een button in dat array hebt, heb je ook de locatie van de tekst in dit array. Voorbeeldje:

*buttonText[getButtonLocation(2)];*

Dit geeft de tekst op de knop met id nummer 2.

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. Als een knop ingedrukt wordt, wordt de juiste functie aangeroepen. Dit ziet er dan ongeveer zo uit: **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];  
 }  
 }  
}**

En dan nu de functie doButtonAction met als argument één integer die het id nummer aangeeft:  
**swich(id){  
 case:** 0: goToMenu(); **break;  
 case:** 1: startGame(false); **break;  
 case:** 2: startGame(true); **break;  
 case:** 3: showScoreBoard(); **break;  
 case:** 5: showAbout(); **break;  
 case:** 6: loadLastGame(); **break;  
 case:** 7: saveGame(); **break;  
}  
if(id >= 100){** onCardClicked**(id-100);  
}**

## 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. Hier is een schets van een functie die een array maakt met alle mogelijk kaarten:

**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;  
}**

Zoals hierboven te zien is bestaat er een functie makeCardString(int i, int j, int k, int l). Het enige wat deze functie doet is door middel van die vier nummers een string maken met de bijpassende karakters. Dus i=2 refereert naar B, i = 3 naar Yellow, j=3 T. Dit is gewoon een aantal swich statments die de getallen omzetten naar karakters.

Wat misschien ook 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 hebben we dus een String array die de stapel kaarten voorstelt.