Subset

Functioneel en technisch ontwerp

Inhaltsverzeichnis

[Opdrachtomschrijving 2](#_Toc4059271)

[Flow diagram 3](#_Toc4059272)

[Schermontwerpen 4](#_Toc4059273)

[Start 4](#_Toc4059274)

[Punt voor de speler 4](#_Toc4059275)

[Een kaart selecteren 5](#_Toc4059276)

[Software benodigheden 6](#_Toc4059277)

[Classes, Variables, Events, Methodes 6](#_Toc4059278)

[Programma Flow 8](#_Toc4059279)

[Testplan 11](#_Toc4059280)

# Opdrachtomschrijving

De opdracht is het maken van een kaartspel genaamd SubSet. Dat is een spel waar 27 kaarten in totaal. Bepaald door 3 factoren.

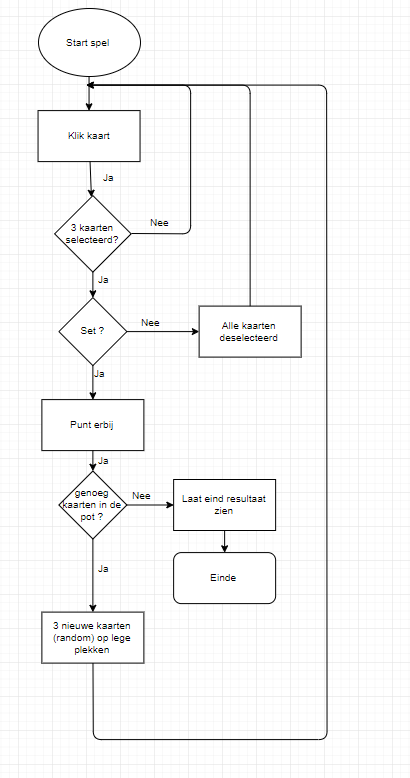
1. Kleuren -> rood, blauw en groen
2. Figuren -> rechthoek circel en een driehoek
3. Aantal -> 1,2 en 3

Van elke kaart is er 1. Op het veld liggen altijd 9 kaarten waarmee de bedoeling is dat je een set maakt. Een set maak je door van elke factoren alles hetzelfde hebt of alles anders. Let erop dat alle factoren die eisen hebben dus bijvoorbeeld 1 rode driehoek 2 blauwe circles en 2 groene vierkanten is geen set dit omdat er 2 keer 2 aantal worden gebruikt en dus 1 van die 2 een 3 zijn moet of de kaart met een rode driehoek had 2 driehoeken moet en hebben.

De volgende minimale eisen ten aanzien van het spel gelden.

* Alle kaarten moeten aangemaakt worden. Je zou dit kunnen doen door een array van 27 strings te maken, waarin elke kaart wordt gerepresenteerd door drie karakters: een voor de waarde van elke eigenschap. Kaart “1rd” is dus de kaart met 1 rode driehoek, “2ge” is de kaart met 2 groene ellipsen, en “3br” is de kaart met drie blauwe rechthoeken.
* De kaarten moeten geschud worden. Processing heeft geen standaardmethode om de elementen in een array op willekeurige volgorde te zetten. Wat je zou kunnen doen is een groot aantal maal twee willekeurige elementen uit de array laten verwisselen.
* De kaarten moeten netjes op het beeld getekend worden. Het ligt voor de hand om een raster van drie bij drie kaarten te tekenen, maar als je iets leukers weet mag dat natuurlijk ook.
* De figuren op de kaarten worden netjes op de kaart getekend. Dus mooi uitgelijnd en op uniforme afstand.
* Een kaart mag maar eenmaal in het spel gebruikt worden. Je moet dus bijhouden welke kaarten er nog in de “stapel” zitten en welke op het scherm staan. Bijhouden welke kaarten weg zijn hoeft niet (mag wel), want die informatie heb je eigenlijk niet nodig.
* De kaarten moeten geselecteerd kunnen worden door de speler en dat moet zichtbaar zijn voor de speler. Je kunt de geselecteerde kaarten bijvoorbeeld een andere achtergrondkleur geven. Let erop dat de gebruiker niet tweemaal dezelfde kaart kiest binnen een selectie.
* Het moet steeds bekend zijn hoeveel mogelijke sets er op tafel liggen. Dat is nodig om te kunnen bepalen of het spelletjes is afgelopen. Als er immers geen sets meer zijn, is het spel afgelopen. Let op: het kan best zo zijn dat een kaart in meerdere mogelijke sets past. Deze sets mag je gewoon allemaal tellen.
* Je moet bijhouden hoeveel sets de speler heeft gevonden en dit aan het einde aan de speler presenteren. Op deze manier kan de speler zien hoe goed (of slecht) hij het gedaan heeft.

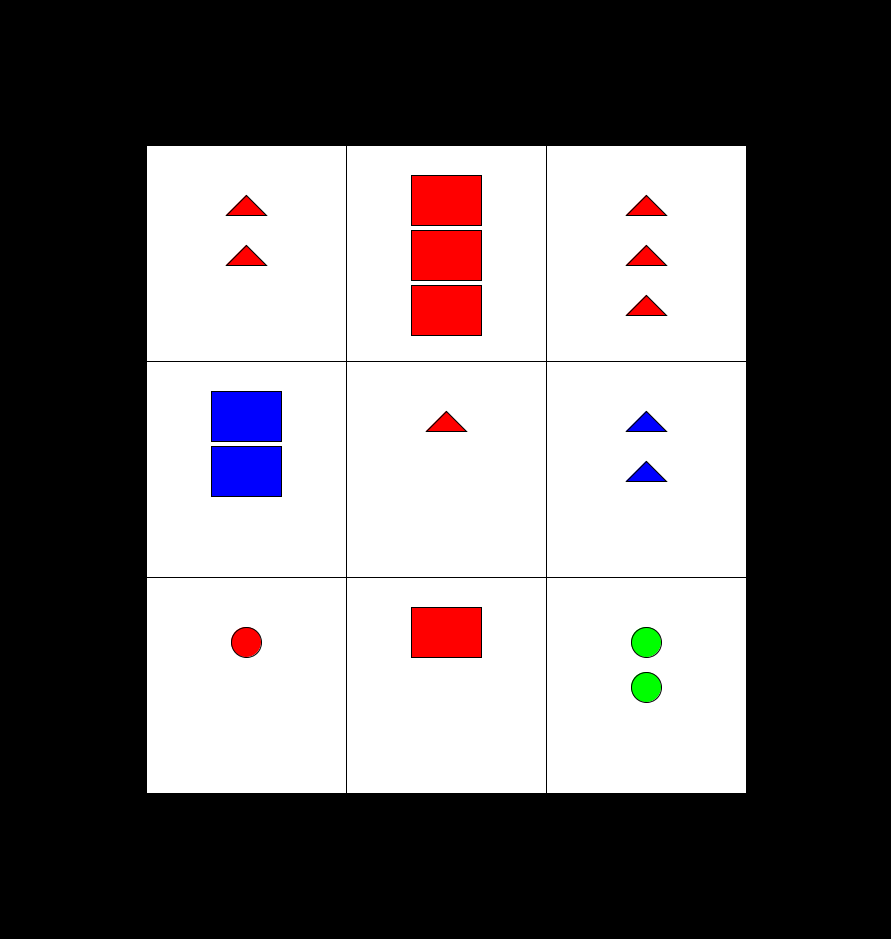
# Flow diagram



# Schermontwerpen

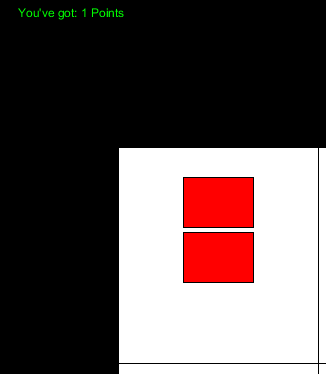
## Start

Hieronder word zichtbaar hoe het er ongeveer uit moet zien wanneer je het spel speelt.



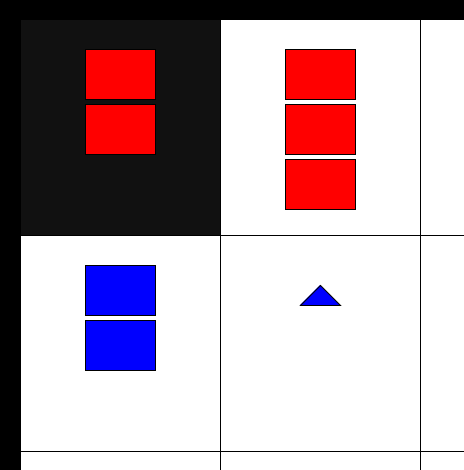
## Punt voor de speler

Wanneer de speler een set heeft gevonden word dat getoont d.m.v. leesbare tekst waar de speler altijd kann zien hoeveel punten hij heeft. (wanneer de speler een verkeerde set speelt word er een punt afgetrokken)



## Een kaart selecteren

Wanneer de speler op een kaart klikt moet dat zichtbaar zijn voor de speler dat hij een kaart heeft geselecteerd. Dit wordt gedaan door de geselecteerde kaart(en) een donkere achtergrond te geven.



# 

# Software benodigheden

## Classes, Variables, Events, Methodes

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Type** | **Name** | **why/what** |
| Kaart | Class | Kaart | aangezien er meerdere kaarten zijn met ieder hun eigen eigenschappen heb ik besloten een class aan te maken  waardoor ik dan de kaart (object) 1 voor 1 veranderen/oproepen of specifieke functies kann laten uitvoeren. |
|  | Variable int | id | Uit ervaring heb ik geleerdt dat een object altijd een unique id moet krijgen. Dit is hier eigenlijk niet nodig  maar principieel bouw ik het er in |
|  | Variable int | plaatsVeld | Dit is de plaats op het veld waardoor ik kann bepalen welke kaart het is |
|  | Variable int | aantal | Dit is dhet aantal figuren op de kaart. |
|  | Variable int | breedteKaart | Dit is de breedte van de kaart. |
|  | Variable int | hoogteKaart | Dit is de hoogte van de kaart. |
|  | Variable String | kleur | Dit is de kleur dat de kaart figuren krijgt |
|  | Variable String | figuur | Dit is het figuur dat de kaart krijgt |
|  | Variable Boolean | selected | Dit is een boolean die een kaart een geselecteerde prop geeft |
|  | Function | berekenPlaatsX | Berekent de plaats vanaf de x positie |
|  | Function | berekenPlaatsY | Berekent de plaats vanaf de y positie |
|  | Function | tekenKaart | tekent de kaart background vanuit zijn propertie selected |
|  | Function | toggleSelected | die zorgt ervoor dat wanneer de kaart is geselecteed wordt gedeselecteerd en andersom |
|  | Function | tekenSymbole | tekent de figuren van de kaart |
|  |  |  |  |
| Global | Variable int | margin | ik wilde het spel niet de complete breedte van het scherm geven. Aangezien ik deze int vaker nodig heb  maak ik daarvan een global int en will ik die liever niet elke keer doorgeven. |
|  | Variable int | aantalKaarten | Deze integer is om vooraf te bepalen hoeveel kaarten er worden gespeeld. |
|  | Variable int | Player1Punt | Deze integer is gemaakt om de punten van player 1 bij te houden. |
|  | Variable int | Player2Punt | Deze integer is gemaakt om de punten van player 2 bij te houden. |
|  | Object array | kaartenOpVeld | In deze objecten array zijn alle kaart opjecten opgeslagen om deze in het spel te kunnen gebruiken. |
|  | Variable boolean | restart | Dete boolean maakt het op een moment mogelijk om het spel te restarten. |
|  | String array | figuren |  |
|  | String array | kleur | Deze zijn nodig voor wanneer het spel begint. Deze maken de mogelijke kaarten |
|  | String array | aantallen |  |
|  |  |  |  |
| Function |  | startGame | deze functie zorgt ervoor dat alle functies die nodig zijn om het spel te starten worden uitgevoerd in 1 functie. |
|  | Variable int | breedte | Deze int wordt gebruikt om een breedte van een kaart te berekenen. |
|  | Variable int | hoogte | Deze int wordt gebruikt om een hoogte van een kaart te berekenen. |
|  | Stringlist | mogKaa | Deze String array wordt gebruikt om de kaarten eigenlijk te shuffelen en zo altijd een random kaart wordt getrokken |
|  |  |  |  |
| Function |  | tekenKaarten | Deze functie zorgt ervoor dat de kaarten worden getekent vanuit de stringarray |
|  | Variable int | marginLeft | door de plaats te berekenen met berekenPlaatsX en berekenPlaatsY kan ik de kaart tekenen |
|  | Variable int | marginTop |  |
|  |  |  |  |
| Function |  | tekenSymbolen | |
|  | Variable int | marginLeft | door de plaats te berekenen met berekenPlaatsX en berekenPlaatsY kan ik de kaart tekenen |
|  | Variable int | marginTop |  |
|  | Varialbe kleur | kleur | deze wordt bepaald wanneer het object in de constructor wordt gemaakt. |
|  |  |  |  |
| Function |  | checkAmountOfSets | Checkt hoeveel sets er zijn te zien |
|  | Variable int | amountOfPoss | deze variabe wordt geretured uit de functie met de waarde hoeveel sets je kann maken |
|  |  |  |  |
| Function |  | checkHetzelfdeOfAnders | Checked of de 3 kaarten die zijn geselecteerd een set zijn |
|  | Object Kaart | Kaart1 |  |
|  |  | Kaart2 | is een Object van de class Kaart |
|  |  | Kaart3 |  |
|  | Variable String | kleur | wordt gezet door de eerste kaart eigenschap kleur |
|  | Variable String | figuur | wordt gezet door de eerste kaart eigenschap figuur |
|  | Variable int | aantal | wordt gezet door de eerste kaart eigenschap aantal |
|  | Variable boolean | kleurCheck |  |
|  | Variable boolean | figuurCheck | Na de loop wordt er gekeken of ze aan een set voldoen |
|  | Variable boolean | aantalCheck |  |
|  |  |  |  |
| Function |  | setText | Handelt alle text die op het scherm wordt getoont af |
|  | Variable int | poss | krijgt de waarden van de functie checkAmountOfSets |
|  |  |  |  |
| Function |  | checkAmountOfSets | Is de functie die alle functie beinhoud waar wordt gechecked of het een set is |
|  | Variable int | arrayIndex | Deze int wordt gebruikt om de Objecten array te vullen |
|  | Variable Boolean | isSet | Krijgt door een functie een false of true als waarde waardoor er wat gebeurd |
|  |  |  |  |
| Function |  | checkSelected | Kijkt hoeveel kaarten er zijn geselecteerd |
|  |  |  |  |
| keyEvents |  |  |  |
| Function |  | keyPressed | In deze functie wordt de mogelijkheid gemaakt om met de toets r aan de top 3 nieuwe kaarten te krijgen.  En wanneer er geen kaarten meer zijn de mogelijkheid hebt met s de game volledig te restarten |
|  |  |  |  |
| mouseEvents | |  |  |
| Function |  | mousePressed | Daarin wordt gekeken waar er gedrukt is vanuit daar wordt er berekent of het een kaart is en zo ja welke  kaart het en voert daaruit functies uit |

# 

# Programma Flow

int margin = 150;

int aantalKaarten = 9;

int player1Punt;

int player2Punt;

int turn = 2;

Kaart[] kaartenOpVeld = new Kaart[aantalKaarten];

StringList mogKaa;

boolean restart;

String[] figuren = {"r", "d", "e"};

String[] kleur = {"r", "g", "b"};

String[] aantallen = {"1", "2", "3"};

void setup() {

size(900, 950);

startGame();

}

**void draw() {**

}

**void startGame() {**

restart = false;

int breedte = (width - ( margin \* 2 )) / 3;

int hoogte = (height - ( margin \* 2 )) / 3;

background(0);

fill(0, 255, 0);

mogKaa = new StringList();

for (int fi = 0; fi < figuren.length; fi++) {

for (int ki = 0; ki < kleur.length; ki++) {

for (int ai = 0; ai < aantallen.length; ai++) {

mogKaa.append(aantallen[ai]+figuren[fi]+kleur[ki]);

}

}

}

for (int i = 0; i < aantalKaarten; i++) {

kaartenOpVeld[i] = new Kaart(mogKaa, i, i, breedte, hoogte);

}

tekenKaarten();

tekenSymbolen();

checkAmountOfSets();

fill(0, 255, 0);

setText(player1Punt, player2Punt);

}

**void tekenKaarten() {**

for (int i = 0; i < aantalKaarten; i++) {

kaartenOpVeld[i].tekenKaart(margin);

}

}

**void tekenSymbolen() {**

for (int i = 0; i < aantalKaarten; i++) {

kaartenOpVeld[i].tekenSymbole(margin);

}

}

**int checkAmountOfSets() {**

int amountOfPoss = 0;

for (int kaartOpVeld = 0; kaartOpVeld < aantalKaarten; kaartOpVeld++) {

for (int tweedeKaart = kaartOpVeld+1; tweedeKaart < aantalKaarten; tweedeKaart++) {

for (int derdeKaart = tweedeKaart+1; derdeKaart < aantalKaarten; derdeKaart++) {

boolean test = checkHetzelfdeOfAnders(kaartenOpVeld[kaartOpVeld], kaartenOpVeld[tweedeKaart], kaartenOpVeld[derdeKaart] );

if (test == true) {

amountOfPoss++;

}

}

}

}

return amountOfPoss;

}

**boolean checkHetzelfdeOfAnders(Kaart Kaart1, Kaart Kaart2, Kaart Kaart3) {**

String kleur = Kaart1.kleur;

String figuur = Kaart1.figuur;

int aantal = Kaart1.aantal;

boolean kleurCheck = false;

boolean figuurCheck = false;

boolean aantalCheck = false;

if (kleur.equals(Kaart2.kleur) && kleur.equals(Kaart3.kleur) && Kaart2.kleur.equals(Kaart3.kleur) || !kleur.equals(Kaart2.kleur) && !kleur.equals(Kaart3.kleur) && !Kaart2.kleur.equals(Kaart3.kleur) ) {

kleurCheck = true;

}

if (figuur.equals(Kaart2.figuur) && figuur.equals(Kaart3.figuur) && Kaart2.figuur.equals(Kaart3.figuur) || !figuur.equals(Kaart2.figuur) && !figuur.equals(Kaart3.figuur) && !Kaart2.figuur.equals(Kaart3.figuur) ) {

figuurCheck = true;

}

if (aantal == Kaart2.aantal && aantal == Kaart3.aantal && Kaart2.aantal == Kaart3.aantal || aantal != Kaart2.aantal && aantal != Kaart3.aantal && Kaart2.aantal != Kaart3.aantal ) {

aantalCheck = true;

}

if (kleurCheck == true && figuurCheck == true && aantalCheck == true ) {

return true;

} else {

return false;

}

}

**void setText(int player1Punt, int player2Punt) {**

int amountLeft = mogKaa.size();

int poss = checkAmountOfSets();

if (turn % 2 == 0) {

push();

fill(255);

ellipse(30,15,5,5);

pop();

} else {

push();

fill(255);

ellipse(30,30,5,5);

pop();

}

text("Player 1 have got: " + player1Punt + " Points", 50, 20);

text("Player 2 have got: " + player2Punt + " Points", 50, 35);

fill(0, 255, 0);

text("There are " + poss + " possibilities", 50, 50);

text("There are " + amountLeft + " cards left", 50, 65);

text("Press 'R' to get some new cards at the top", width-250, 20);

turn++;

}

**void checkSet(int breedte, int hoogte) {**

Kaart[] selected = new Kaart[3];

int arrayIndex = 0;

for (int i = 0; i < aantalKaarten; i++) {

if (kaartenOpVeld[i].selected) {

selected[arrayIndex] = kaartenOpVeld[i];

arrayIndex++;

}

}

boolean isSet = checkHetzelfdeOfAnders(selected[0], selected[1], selected[2]);

if (isSet) {

if (turn % 2 == 0) {

player2Punt++ ;

} else {

player1Punt++ ;

}

background(0);

fill(0, 255, 0);

if (mogKaa.size() <= 2) {

startGame();

}

for (int i = 0; i < aantalKaarten; i++) {

if (kaartenOpVeld[i].selected) {

kaartenOpVeld[i] = new Kaart(mogKaa, i, i, breedte, hoogte);

}

}

setText(player1Punt, player2Punt);

} else {

for (int i = 0; i < aantalKaarten; i++) {

if (kaartenOpVeld[i].selected) {

kaartenOpVeld[i].toggleSelected();

}

}

if (turn % 2 == 0) {

player2Punt-- ;

} else {

player1Punt-- ;

}

background(0);

fill(255, 0, 0);

setText(player1Punt, player2Punt);

}

}

**int checkSelected() {**

int isChecked = 0;

for (int i = 0; i < aantalKaarten; i++) {

if (kaartenOpVeld[i].selected) {

isChecked++;

}

}

return isChecked;

}

# Testplan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Volgnr** | **Situatie** | **Actie** | **Verwacht Resultaat** | **Correct Ja/Nee** | **Opmerkingen** |
| 1 | Start Spel | Starten van het spel | scherm met 9 random gegenereerde kaarten |  |  |
| 2 | Spel gestart | Druk op een kaart | Kaart krijgt een donkere achtergrond |  |  |
| 3 | 2 kaarten geselecteerd | druk op een NIET geselecteerde kaart | Check of het een set is. (wanneer een set krijgt de speler een pluspunt) |  |  |
| 4 | speelt het spel | druk op een geselecteerde kaart | Kaart krijgt weer de standaart achtergrond |  |  |
| 5 | Wanneer er geen kaarten meer in de pot zijn | speelt een set | genereerd weer opnieuw de kaarten |  |  |
| 6 | Gebruiker denkt dat er geen set meer op het veld is | Druk op r | worden 3 nieuwe kaarten aan de bovenkant neergelegd |  |  |
| 7 | 2 kaarten geselecteerd | Druk op een kaart (geen set) | Alle kaarten worden gedeselecteerd en speler krijg een punt aftrek |  |  |
| 8 | geen kaarten meer voor voor een random generate aan de top | Text komt dat de gebruiker een spel kann resetten door op S te drukken | game restart punten weer op 0 |  |  |
| 9 | Speler heeft met de S toets een spel gereset | Speler drukt op S | Gebeurd niks |  |  |

**Gerealiseerd door: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Getest door: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Datum: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**