OPDRACHT BEROEPSPRODUCT: SUBSET

INTRODUCTIE

Toen genetisch onderzoekster Marsha Falco in 1974 op zoek was naar een handige manier om genen en chromosomen in cellen te bestuderen, maakte ze kaartjes met daarop symbolen die bepaalde blokken informatie vertegenwoordigden. Vrienden wezen haar erop dat de kaartjes nog een functie zouden kunnen vervullen: het spelen van een spel. Zo werd het kaartspel Set geboren.

Jij gaat een iets vereenvoudigde versie van het spelletje programmeren in Processing. We noemen dit spelletje “SubSet”.

DE REGELS VAN SUBSET

De regels van het spelletje lijken erg op die van Set, en zijn als volgt:

Er zijn 27 unieke kaarten in het spel die op drie eigenschappen van elkaar kunnen verschillen:

- De hoeveelheid figuren: 1, 2 of 3
- De vorm van de figuren: driehoeken, rechthoeken of ellipsen
- De kleur van de figuren: rood, groen of blauw

Elke combinatie komt precies eenmaal voor. Zo is er dus bijvoorbeeld een kaart met 1 groene driehoek, een met 2 blauwe ellipsen en een met 3 rode rechthoeken.

Er zijn steeds 9 kaarten tegelijk te zien op het scherm en het is de taak van de speler om daarin “sets” te vinden. Een set bestaat altijd uit drie kaarten. Drie kaarten vormen alleen een set, als voor elk van de drie eigenschappen (aantal, vorm en kleur) afzonderlijk geldt dat de eigenschap ofwel op alle drie kaarten gelijk is, ofwel op elke kaart verschillend is.

VOORBEELDEN

Een paar voorbeelden van sets:

- 1 rode rechthoek / 2 rode rechthoeken / 3 rode rechthoeken (want aantal is steeds verschillend, kleur is steeds hetzelfde, vorm is steeds hetzelfde)
- 1 blauwe ellips / 2 groene driehoeken / 3 rode rechthoeken (want aantal is steeds verschillend, kleur is steeds verschillend en vorm is steeds verschillend)
- 1 blauwe ellips / 1 groene driehoek / 1 rode rechthoek (want aantal is steeds hetzelfde, kleur is steeds verschillend en vorm is steeds verschillend)
- 1 blauwe ellips / 1 groene ellips / 1 rode ellips (want aantal is steeds hetzelfde, kleur is steeds verschillend en vorm is steeds hetzelfde)

Een paar voorbeelden van combinaties die GEEN set zijn:

- 1 rode rechthoek / 2 rode rechthoeken / 3 blauwe rechthoeken (want kleur is niet steeds hetzelfde en ook niet steeds verschillend)
- 1 rode rechthoek / 1 rode cirkel / 2 rode rechthoeken (want aantal is niet steeds hetzelfde en ook niet steeds verschillend)

De speler ziet dus steeds 9 willekeurige kaarten op het scherm en kan zo’n kaart met een klik selecteren. Zodra de speler drie kaarten heeft geselecteerd, moet het programma nagaan of de kaarten samen een set vormen. Als dat niet zo is wordt de selectie opgeheven, maar gebeurt er verder niets. Als het wel een correct set is, verdwijnen de geselecteerde kaarten uit het spel en worden ze vervangen door nieuwe

kaarten. Het spel eindigt zodra er geen sets meer zijn. Dat kan betekenen dat de kaarten allemaal op zijn, maar het kan ook zijn dat er in de kaarten die op het scherm staan geen set meer te vinden is. Let op: zodra er geen nieuwe kaarten meer van de "stapel" gepakt kunnen worden, kan het nog wel zo zijn dat er nog sets te vinden zijn. Houd hier dus rekening mee.

VERPLICHTE FUNCTIONALITEIT

Je spel moet in elk geval de volgende functionaliteit bevatten:

- 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.

Uiteraard is het de bedoeling dat je de kaarten tekent in het tekenvenster en daar ook aangeeft welke kaarten geselecteerd zijn. Overige informatie (zoals aan het einde van het spelletje) mag je ook op het console weergeven.

TIPS

Het is waarschijnlijk een goed idee om een kaart te representeren als een string bestaande uit drie karakters: een voor elke eigenschapswaarde. De kaart met 1 rode ellips zou je dus kunnen representeren als "1re" en de kaart met 3 groene driehoeken als "3gd". Als je dat doet, kun je twee arrays van strings gebruiken. In de ene staan alle kaarten die nog in de "stapel" zitten, in de andere de kaarten die op het scherm staan. Van die tweede array kun je de positie van de kaart in de array overeen laten komen met de schermpositie. De kaart helemaal voorin de array (positie 0) wordt dan linksboven in het scherm getekend, en de kaart helemaal achterin de array (positie n-1) rechtsonder.

Het is waarschijnlijk ook handig om met een "schaduwadministratie" bij te houden waarin je bijhoudt hoeveel kaarten er nog in de stapel zitten. Je kunt een array namelijk niet laten inkrimpen, dus met "size" krijg je altijd 27 terug, ook als je eigenlijk al kaarten uit de rij "verwijderd" hebt. We raden je aan een integer bij te houden die in het begin de waarde 27 heeft, maar die steeds met 1 verlaagd wordt als je een kaart van de stapel "pakt". En als je dat "pakken" steeds vanaf het einde doet, hoef je alleen maar de teller te verlagen om de kaart uit de stapel te verwijderen.

Op een soortgelijke manier kun je ook bijhouden welke kaarten door de gebruikers zijn geselecteerd. Maak een int-array voor drie posities en houd met een aparte integer bij hoeveel kaarten er geselecteerd zijn. Zodra er drie kaarten geselecteerd zijn en de analyse heeft plaatsgevonden of het wel of geen set was, kun je de administratie-integer weer op 0 zetten en hoef je met de selectie-array niets te doen. Zodra er weer een kaart wordt aangeklikt, kun je de positie van die kaart gewoon opslaan en de teller met 1 verhogen.

Je programma zou o.a. de volgende variabelen en functies kunnen bevatten (je mag uiteraard ook voor een andere oplossing of naamgeving kiezen):

- String[] eigenschappen={"123","rgb","red"} – Bevat drie strings die elk een eigenschap representeren. In elke string zitten de mogelijke waarden voor die eigenschap.
- String[] genereerKaarten(String[] eigenschappen) – Bevat for-loops die over de eigenschappen itereren en op basis daarvan alle mogelijke kaarten samenstelt.
- String[] gedekteKaarten – Bevat alle kaarten die nog niet in het spel zijn. Is dus 27 elementen lang.
- int nGedekteKaarten – Geeft aan hoeveel gedekte kaarten er nog zijn. Zodra er kaarten van de “stapel” naar het scherm gaan, neemt dit aantal dus af.
- String[] openKaarten – Bevat de kaarten die op het scherm staan.
- int[] geselecteerdePosities – Hierin wordt bijgehouden welke van de kaarten in openKaarten is/zijn geselecteerd door de speler. Kan drie elementen bevatten.
- int nGeselecteerdePosities – Geeft aan hoeveel kaaren er geselecteerd zijn.
- String pakKaartVanStapel() – Kopieert de achterste kaart uit gedekteKaarten (dat is dus degene met volgnummer nGedekteKaarten-1) naar een open plaats in openKaarten en verlaagt nGedekteKaarten met 1.
- boolean isSet(String[] kandidaatset) – Geeft op basis van drie kaarten aan of het een set is.
- int nSetsOpTafel() – Zet een teller op 0 en kijkt voor elke combinatie van drie kaarten op tafel of het een set is (een for-loop voor de eerste kaart, met daarin een for-loop voor de tweede kaart, met daarin een for-loop voor de derde kaart). Zo ja, dan wordt de teller met 1 verhoogd. Het totaal wordt geretourneerd.
- int aangekliktePositie() – Geeft het positienummer van de kaart die zich op positie (mouseX, mouseY) bevindt. Als zich daar geen kaart bevindt, kun je bijvoorbeeld -1 laten retourneren.
- void tekenBord() – Steeds als er iets verandert aan het bord (er worden nieuwe kaarten getoond, er wordt een kaart geselecteerd) roep je deze functie aan. Je kunt het tekenen ook in de draw()-functie doen, maar dan wordt het bord heel vaak voor niets opnieuw getekend.
- void tekenKaart(String kaart,int bordpositie) – Tekent de kaart op basis van de stringrepresentatie (bijv. “1rr” voor de kaart met 1 rode rechthoek) en de positie op het bord. Uiteraard heb je ook functionaliteit nodig die de x- en y-positie behorende bij een bordpositie bepaalt. Het is aan te raden dat in een aparte functie te doen, zoals:
- int[] coordinatenBijBordpositie(int bordpositie) – Geeft een array (lengte 2) van integers terug met daarin de x- en y-coordinaat behorende bij de bordpositie. Je gebruikt uiteraard width en height om deze positie te helpen bepalen.

OVERIGE EISEN

Verder moet je uitwerking aan de volgende eisen voldoen:

- Er is een ontwerp beschikbaar dat voldoet aan de in de course besproken voorwaarden
- Je hebt je aan de course behandelde programmeerconventies gehouden, dus je hebt in elk geval zo veel mogelijk gebruik gemaakt van (herbruikbare) functies en variabelen en hebt je code goed gedocumenteerd
- Je hebt je programma uitgebreid getest: zowel de losse functies als het eindproduct met interface

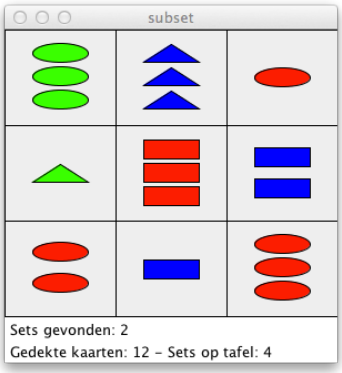
BONUSPUNTEN

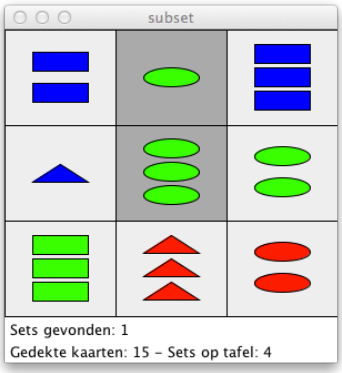
Bonus- of compensatiepunten zijn te verdienen met extra functionaliteit die je duidelijk moet markeren (zowel in het ontwerp als in je code). Hieronder staan enkele suggesties. Als je zelf een uitbreiding verzint, moet je hiervoor eerst toestemming vragen aan je docent.

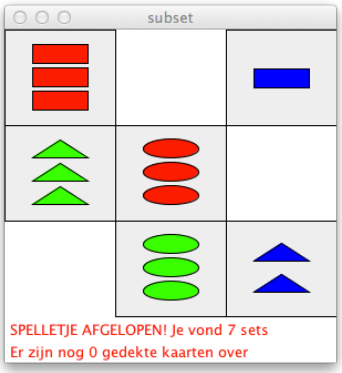
- Eenvoudig:
  - Het eindresultaat van het spelletje wordt niet in het console, maar in het tekenvenster aan de speler gepresenteerd. De speler hoeft dan dus helemaal niet naar het console te kijken.
  - De mogelijkheid om een geselecteerde kaart te de-selecteren door erop te klikken.
- Lastig:
  - Real-time informatie in beeld over hoeveel kaarten er nog in de “stapel” zitten en hoeveel sets er in beeld te vinden zijn.
  - Een knop “Geef me een hint” waarmee twee kaarten uit een van de op het scherm aanwezige sets gemarkeerd worden. Gebruik van deze knop zou tot puntenaftrek kunnen leiden.
  - De mogelijkheid om met twee spelers om de beurt drie kaarten te selecteren en zo echt te kunnen winnen of verliezen.
- Erg lastig:
  - Niet drie eigenschappen weergeven (aantal, vorm en kleur), maar vier zoals in het originele spel (verzin zelf de vierde eigenschap: het mag “vulling” zijn, maar iets anders mag ook). Het is in dat geval een goed idee om net zoals in het originele spel niet 9, maar 12 kaarten te tekenen.
- Heel erg lastig:
  - Een knop “Ik zie geen set” waarmee drie extra kaarten op het scherm komen. Gebruik van deze knop zou tot puntenaftrek kunnen leiden.
- Heel heel erg lastig:
  - Speel tegen elkaar (2 tot 6 spelers) in de klas over het netwerk.
  - Creëer een gezamenlijk bord
  - Definieer spelers die bij iedereen zichtbaar zijn, inclusief de scores
  - De selectie van de set moet naar alle spelers gecommuniceerd worden. Dus wie als eerst een goede set heeft geselecteerd moet dit gecommuniceerd krijgen naar de andere spelers en een routine moet bepalen wie de eerste is.
  - Scores van de verschillende spelers moet bijgehouden worden en ook voor iedereen zichtbaar zijn.
  - Een winnaar moet nadat de kaarten op zijn bepaald worden en zichtbaar worden in de schermen van de spelers.

SCREENSHOTS VAN VOORBEELD

Om een beetje een beeld te geven van hoe je eindproduct eruit zou kunnen zien, hier een paar screenshots uit een voorbeelduitwerking (let op: deze uitwerking bevat ook niet-verplichte functionaliteit).







OPDRACHT

1. Beschrijf de stappen die je programma moet doorlopen. Welke keuzes worden gemaakt en wat is het resultaat van deze keuzes. Maak een schets van het scherm(de schermen) die worden getoond bij deze stappen/keuzes. Welke gegevens moeten precies worden weggeschreven? Beschrijf welke tests het programma moet doorlopen en welke uitkomst.
2. Welke variabelen moet je programma vasthouden. Wat voor soort variabelen houd je vast. Welke functies zijn noodzakelijk of handig om te maken. Welke informatie hebben deze functies nodig om de taak te volbrengen (parameters). Welke waarden moeten tijdelijk worden vastgehouden door deze functies. En welke waarde levert de functie op (returnwaarde).
3. Realiseer het programma volgens het bij 2 gemaakte ontwerp.
4. Demonstreer het programma aan de hand van de bij 1 beschreven tests.

Alle casusopdrachten bestaan uit twee delen:

1. *Een ontwerp conform de richtlijnen die in de reader behandeld zijn*
2. *Naar aanleiding van je ontwerp realiseer je het programma in Processing-3*

Inleverinstructies:
Zowel het ontwerp en de realisatie lever je in via iSAS (zie toetsrooster iSAS voor de deadline)