

Analyse Zeeslag

Versie 2

Informatica Communicatie Academie | Nijmegen

1 november 2019

Structured Program Development
Docent : Mark Achten

Analyse geschreven door:
Lauren Meenhorst SN632206
Klas 1^E

Inhoudsopgave

Inleiding.....	3
functioneel ontwerp	4
Beschrijving van de functionaliteit	4
Optionele functionaliteit.....	4
MoSCoW methode	5
Schermontwerpen en beschrijving van de functionele werking van alle schermen....	7
loop	7
Startscherm	8
Spelscherm	9
Eindscherm	13
Technisch ontwerp.....	14
1. init_Zeeslag	14
2. Classes	15
Definiëren	15
initialiseer functies	16
3. opstellingen.....	17
4. events	17
Variabelen	17
Functies.....	17
5. startScherm	19
Variabelen	19
Functies.....	19
5. spelScherm	21
Variabelen	21
Functies.....	21
eindScherm	23
functies	23
Test plan en uitvoering	24
startScherm	24
spelScherm	24
eindScherm	24
Conclusie	25

Inleiding

In dit analyse document denk ik na over hoe ik het spel zeeslag ga maken.

Ik doe dit in drie hoofdstukken: het functioneel ontwerp, het technisch ontwerp en het testplan met uitvoering. Dit doe ik om mij te ondersteunen bij het maken van het spel.

functioneel ontwerp

Beschrijving van de functionaliteit

Dit systeem is een versie van zeeslag. Het doel van het spel is om te ontdekken waar de schepen van de tegenstander zich bevinden. De speler die dit als eerste doet wint het spel.

In het startscherm geven spelers de gewenste roostergrootte aan. Spelers hebben de keuze uit een rooster van 10x10, 15x15 of 20x20 vakken.

Tevens geven spelers hier aan hoe veel afweersystemen zij willen gebruiken. Het aantal mogelijke afweersystemen begint bij 0 en eindigt bij 20.

Indien gewenst kunnen de spelers hier hun naam ingeven. Hierna wordt in het programma met deze namen naar hen verwezen.

Indien er geen naam wordt ingevoerd wordt er uitgegaan van de default namen speler 1 en speler 2.

Beide spelers krijgen een armada. Deze bevat 1 slagschip van 4 cellen lang, 2 kruisers van 3 cellen, 3 torpedojagers van 2 cellen en 4 onderzeeërs van 1 cel.

De opstellingen van de armada's worden willekeurig door de computer bepaald. De computer kies uit 1 van 3 mogelijke opstellingen en zal dit blijven doen totdat beide spelers een verschillende opstelling toegewezen hebben gekregen.

Hierna verdeelt de computer het door spelers gekozen aantal mijnen en afweersystemen willekeurig over de lege cellen.

Wanneer alle schepen, mijnen en afweersystemen zijn geplaatst bepaalt de computer welke speler mag beginnen.

Per beurt zijn er een aantal mogelijkheden deze worden later beschreven onder het kopje algoritmen.

Als er een winnaar bekend is eindigt het spel met een felicitatie scherm met de optie om opnieuw te beginnen. Wanneer de spelers hiervoor kiezen beginnen zij weer bij het startscherm.

Optionele functionaliteit

Indien mogelijk zal het opslaan en laden van een spelsituatie worden geïmplementeerd.

MoSCoW methode

<i>algemeen</i>	must	should	could	Would
Spel wordt met 2 spelers gespeeld.	x			
Spel kan met (>2) spelers worden gespeeld.				x
Speler kan kiezen om tegen de computer te spelen.			x	
Er zijn 3 schermen: <ul style="list-style-type: none"> Startscherm Spelscherm Eindscherm 	x			
Spel kan worden opgeslagen en kan bij opnieuw opstarten weer ingeladen worden.			x	
Alle spelelementen zijn volledig schaalbaar t.o.v. het grafische scherm van processing.			x	
Er worden animaties gebruikt voor spel elementen.				x
Spel heeft levels. Als het spel is gewonnen gaan spelers door naar een volgend level.				x

<i>Startscherm</i>	must	should	could	Would
Speler kan kiezen voor een roostergrote van 10x10, 15x15 of 20x20.	x			
Speler kan kiezen voor min 0 en max 20 afweersystemen.	x			
Speler kan een naam opgeven. De naam wordt onthouden en in het spel gebruikt.		x		
Er is een startknop aanwezig.	x			
Er is een knop aanwezig die spelers de mogelijkheid biedt verder te gaan met een opgeslagen spel.			x	

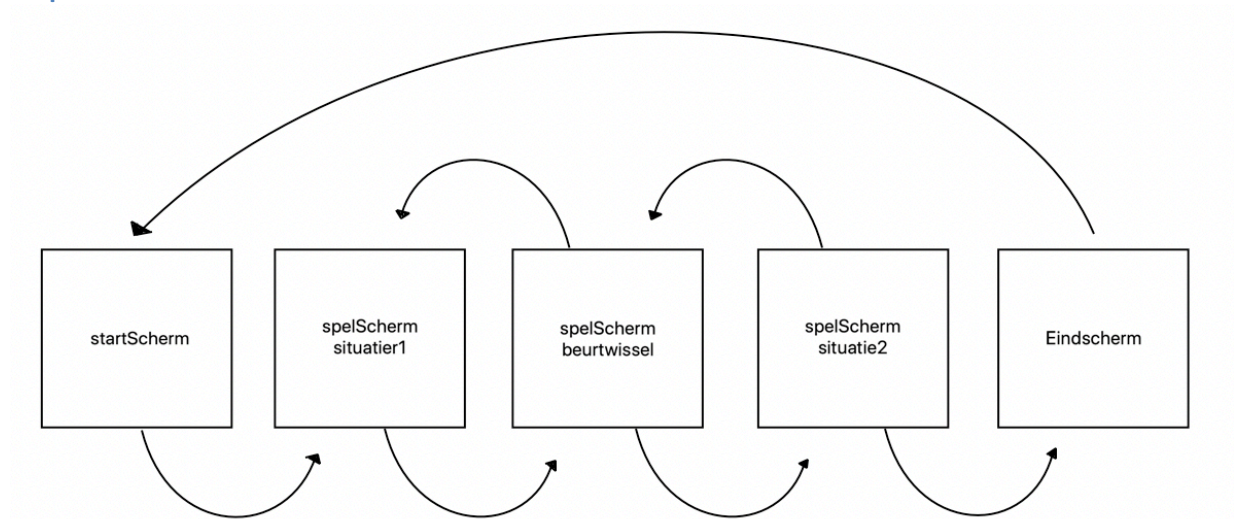
<i>Spelscherm</i>	must	should	could	Would
Opstelling word willekeurig gekozen uit een set van vooraf bepaalde opstellingen.	x			
Schepen worden op volledig willekeurige posities geplaatst.		x		
Speler kan schepen zelf in zijn rooster plaatsen				x
Mijnen en afweer systemen worden op willekeurige posities geplaatst. Dit gebeurt na het plaatsen van de schepen.	x			
Het aantal mijnen is gelijk aan het aantal afweersystemen.	x			
Schepen, mijnen en afweersystemen kunnen nooit in het zelfde blok staan.	x			
Speler met de eerste beurt wordt willekeurig gekozen.		x		
Elke speler heeft zijn eigen speelveld.	x			
Speler X kan alleen in zijn eigen speelveld klikken.	x			
Bij elke rij en kolom staat een getal dat aangeeft hoeveel vakken bezet zijn door scheepsonderdelen in die rij of kolom.	x			
Tijdens het spel wordt constant aangegeven wie de Actieve speler is, wat zijn score is en wat de speeltijd is	x			

Voor spelers wordt het aantal kliks bijgehouden (alleen wanneer een cel nog niet is behandeld, en de klik binnen het juiste rooster valt). Spelers hebben een beperkt aantal pogingen. Als het spel eindigt voor alle schepen gezonken zijn is de speler met de hoogste score de winnaar.			x	
Er is een knop aanwezig die spelers de mogelijkheid biedt hun spel op te slaan waarna het spel wordt beëindigd.		x		
Er is een knop aanwezig die spelers de mogelijkheid biedt hun spel op te slaan waarna het spel wordt beëindigd.			x	

<i>Eindscherm</i>	must	should	could	Would
De winnaar wordt bekend gemaakt (naam of speler1/2)	x			
Er is een mogelijkheid om opnieuw te spelen.	x			

Schermontwerpen en beschrijving van de functionele werking van alle schermen.

loop



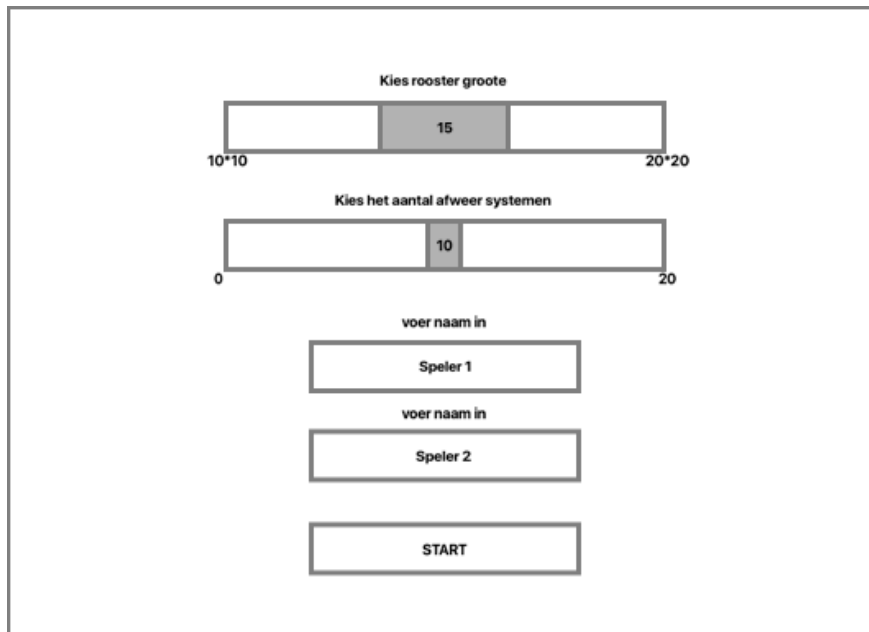
Bovenstaand plaatje verduidelijkt de flow van het programma.

Wanneer het programma start komen we in het startscherm. Na het event (mousePressed in knop) komen we in het spelscherm van speler1 (Dit kan ook speler 2 zijn als de computer speler 2 laat beginnen).

We wisselen tussen spelscherm speler1, beurtwissel en speler2 tot er een winnaar bekend is of totdat het maximaal aantal beurten is bereikt.

Als de winnaar bekend is komen we in het eindscherm. Indien de spelers nog een spel willen spelen komen we weer in het startscherm anders eindigt het spel.

Startscherm



The diagram illustrates the layout of a start screen. At the top, a slider is labeled 'Kies rooster groote' with values '10*10' and '20*20'; the value '15' is currently selected. Below this, another slider is labeled 'Kies het aantal afweer systemen' with values '0' and '20'; the value '10' is currently selected. Underneath the sliders are two text input fields, each preceded by the label 'voer naam in'. The first input field contains the text 'Speler 1' and the second contains 'Speler 2'. At the bottom of the form is a rectangular button labeled 'START'.

De afbeelding hierboven is een eerste ontwerp voor het startscherm. Er is nog geen rekening gehouden met kleuren en fonts omdat dit niet noodzakelijk is bij het beschrijven van de functionaliteit van het scherm.

In dit scherm kiezen spelers hun roostergrootte en aantal afweersystemen en impliciet ook de mijnen. Optioneel geven zij hier ook hun naam mee aan het programma.

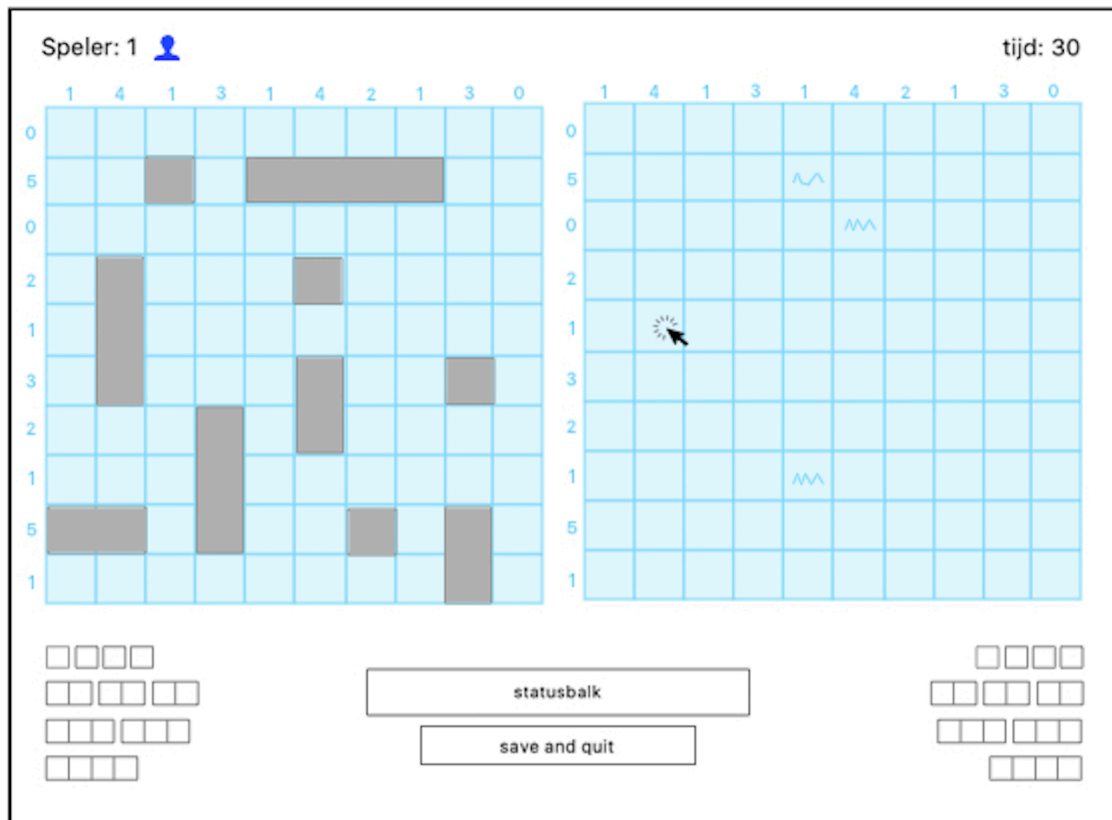
Er zijn 3 mogelijkheden voor roostergrootte. De speler sleept het slider blok naar 1 van de 3 opties. De waarde van deze optie wordt in een variabele opgeslagen.

Het aantal afweersystemen wordt wederom bepaald met een slider. De positie van de slider wordt opgeslagen in een variabele en vervolgens gebruikt bij het plaatsen van X aantal mijnen en afweersystemen.

De startknop verandert de schermstatus naar spelscherm.

Optioneel komt in dit scherm een button die spelers de mogelijkheid geeft een opgeslagen spel te laden. Indien de speler hiervoor kiest worden de aangeven waarden voor roostergrootte en afweersystemen overschreven.

Spelscherm



Dit wireframe is een eerste ontwerp van spelscherm situatie1.

Links boven in het scherm wordt aangegeven welke speler aan de beurt is. Rechts boven wordt de speel tijd afgebeeld.

Het linker rooster laat de schepen van de speler 1 zien. Wanneer speler 1 in dit veld klikt gebeurt er niets. Het rechter rooster is het rooster van speler 2. Eigenlijk is dit een ander (leeg) rooster maar wanneer speler 1 in het afgebeelde rooster klikt controleert het programma de inhoud van de geklikte cel in het rooster van speler 2.

Naast elk rooster staan cijfers. Deze cijfers geven aan hoeveel blokjes worden gevuld door schepen in rij onder of de kolom naast het cijfer.

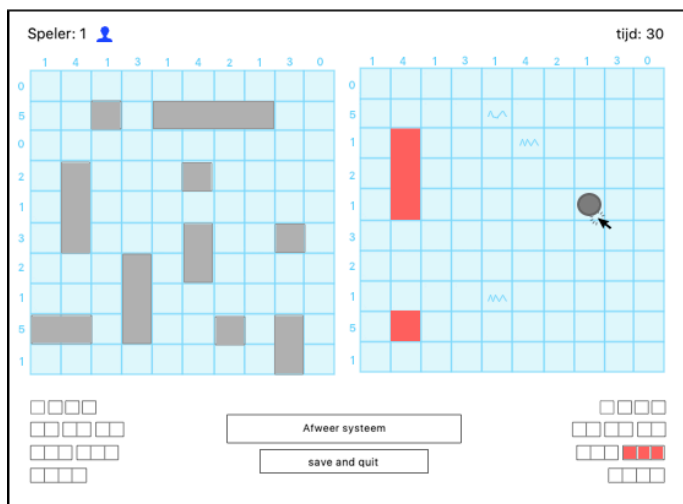
Optioneel wordt onder de roosters wordt de status van het spel weergegeven. Hier komt bij elke klik te staan of de speler mis, raak, op een afweersysteem of op een mijn heeft geklikt.

Optioneel wordt er afgebeeld hoe veel punten de speler heeft en hoeveel beurten er nog zijn tot einde spel.

Optioneel is er een legenda aanwezig die aangeeft wat de kleuren in het veld betekenen.

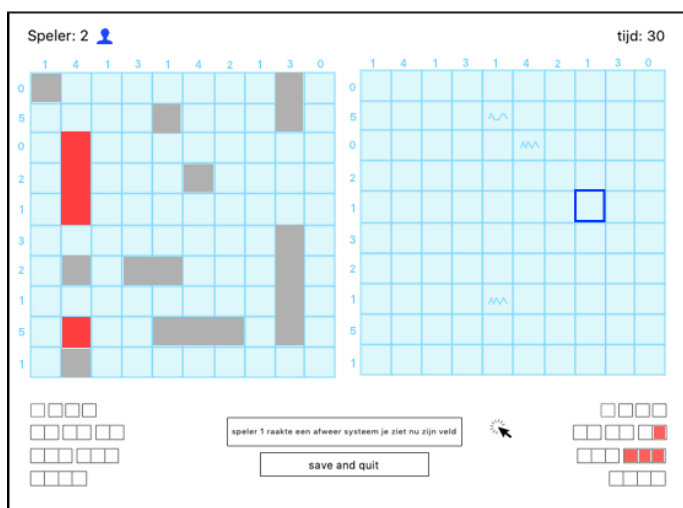
Optioneel staan onder elk rooster blokjes die aangeven hoeveel schepen er zijn. Wanneer een schip is gezonken kleurt het schip.

Optioneel komt er een knop onder de statusbalk waar spelers op kunnen klikken om hun spel op te slaan en later verder te spelen.



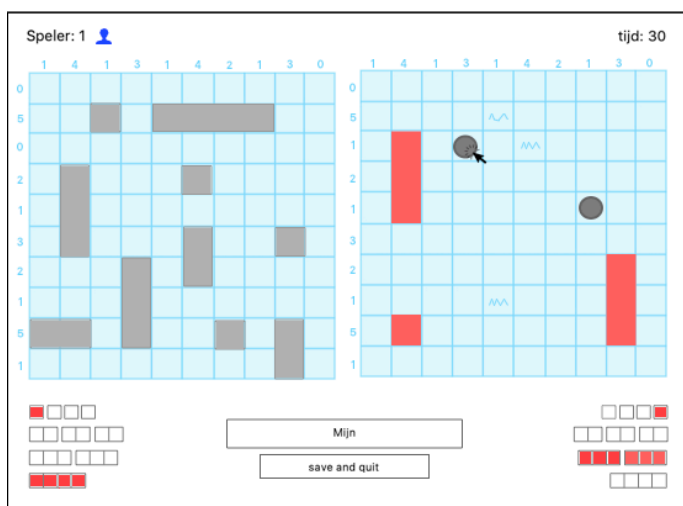
Dit wireframe toont het scherm van speler 1 wanneer hij een afweer systeem raakt.

De beurt van de speler gaat voorbij. De cel waar de speler op heeft geklikt komt bij speler 2 permanent bloot te liggen.



Dit wireframe toont het scherm van speler 2 nadat speler 1 een afweersysteem heeft geraakt. Het vak waar speler 1 op heeft geklikt komt bloot te liggen. Dit wordt ook in de statusbalk aangegeven.

In dit geval is het vak leeg.



Dit wireframe toont het scherm van speler 1 wanneer hij een mijn raakt.

De beurt van de speler gaat voorbij.

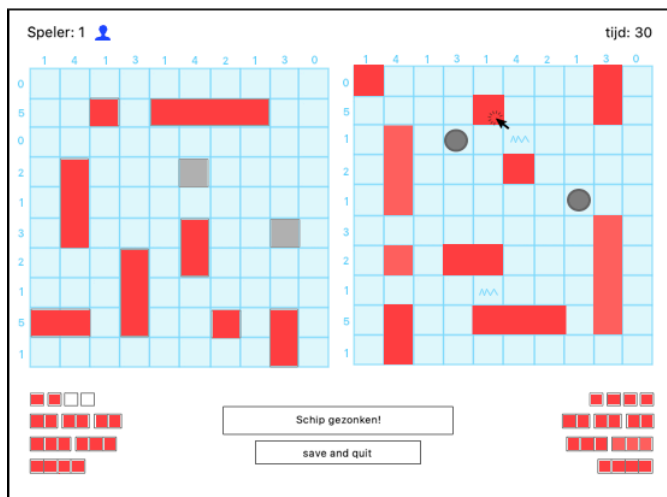


Dit wireframe toont het scherm dat de spelers zien wanneer de computer moet worden doorgegeven.

Het scherm verschijnt wanneer de beurt van speler 1 of 2 is afgelopen.

Het scherm blijft een x aantal seconden staan om spelers de tijd te geven het scherm door te geven.

Optioneel is dit scherm opgemaakt met een plaatje.



Dit wireframe toont speler 1 die de winnende zet doet.

Na dit scherm zal speler 2 geen beurt meer krijgen.

Spelers krijgen het eindscherm te zien.

Eindscherm



Bovenstaand wireframe toont het scherm dat de spelers zien wanneer de winnaar bekend is.

Het scherm toont de winnaar en geeft de spelers de mogelijkheid opnieuw te spelen.

Deze knop brengt hen terug naar het startscherm.

Optioneel is dit scherm mooi opgemaakt met een plaatje.

Technisch ontwerp

Overzicht van programma per tab.

1. init_Zeeslag

```
// spelers
final float SPELER1 = 0;
final float SPELER2 = 1;
int actieveSpeler = bepaalEersteSpeler();
int inactieveSpeler = 0;

// schermwaarden
final int STARTSCHEM = 0;
final int SPELSCHERM = 1;
final int EINDSCHEM = 2;
int spelStatus = STARTSCHEM;

// roosters
final int AANTALROOSTERS = 4;
int roosterGrootte, rijen, kolommen, aantal Verdedigingsmechanismen;
float roosterRuimte, celZijde;
int marge = 40;

// tijd
float stoptime, wait;
boolean beurtWissel = false;

// beurten
int maxAantalBeurten = 0;
```

Dit tabblad bevat 3 functies:

1. void beurtwissel()
2. void setup()
3. void draw()

Functie 1; Wisselt de waarde van actieve en inactieve speler, hoogt het aantal beurten op initialiseert de stoptijd en zet de boolean beurtwissel op true.

```
void beurtWissel() {
    int hulpVariabeleSpeler = actieveSpeler;
    spelers[actieveSpeler].aantalBeurten ++;
    actieveSpeler = inactieveSpeler;
    inactieveSpeler = hulpVariabeleSpeler;
    stoptime = millis() + wait;
    beurtWissel = true;
}
```

Functies 2 en 3 bevatten een switch die de functie aanroepen op basis van spelStatus.

2. Classes

Definiëren

```
class Speler {
    String naam;
    int totaalScore;
    int opstellingRooster;
    int situatieRooster;

    Speler (String n, int t, int o, int s) { //constructor
        naam = n;
        totaalScore = t;
        opstellingRooster = o;
        situatieRooster = s;
    }
}

class Cel { // Deze klasse wordt gebruikt voor de cellen in de roosters
    int status; // Cel status bijvoorbeeld leeg of een boot
    int kleur;
    float xPos;
    float yPos;
    boolean behandeld;

    Cel (int s, int k, float x, float y, boolean b) { //constructor
        status = s;
        kleur = k;
        xPos = x;
        yPos = y;
        behandeld = b;
    }
}

class Boot { // Deze klasse wordt gebruikt voor de armada
    String type;
    int lengte;
    int score;
    int kleur;
    int waarde;
    int aantal;

    Boot (String t, int l, int s, int k, int w, int a) { //constructor
        type = t;
        lengte = l;
        score = s;
        kleur = k;
        waarde = w;
        aantal = a;
    }
}

class Verdediging { // Deze klasse bevat de verdedigingsobjecten
    String type;
    int lengte;
    int score;
    int waarde;
    int aantal;

    Verdediging (String t, int l, int s, int w, int a) { //constructor
        type = t;
    }
}
```

```

    lengte = l;
    score = s;
    waarde = w;

```

```

// maak arrays van type class
Speler [] spelers;
Cel [][] roosters;
Boot [] boten;
Verdediging [] verdedigingsMechanismen;

```

initialiseer functies

```

void initSpelers() {
    spelers = new Speler[2]; // Aantal spelers
    spelers[0] = new Speler("Speler 1", 200, 2, 0);
    spelers[1] = new Speler("Speler 2", 200, 3, 1);
}

```

```

void initRoosters() {
    roosters = new Cel [AANTALROOSTERS][rijen][kolommen];
    for ( int rooster = 0; rooster < AANTALROOSTERS; rooster ++ ) {
        for ( int rij = 0; rij < rijen; rij++) {
            for ( int kolom = 0; kolom < kolommen; kolom++) {
                roosters[rooster][rij][kolom]=new Cel( 0, 0, 0, 0, false);
            }
        }
    }
}

```

```

void initBoten() {
    boten = new Boot[4]; // Aantal typen boten
    boten[0] = new Boot("Slagschip", 4, 100, #9CACC6, 1, 1);
    boten[1] = new Boot("Kruiser", 3, 75, #8894AA, 2, 2);
    boten[2] = new Boot("Torpedojager", 2, 50, #7B879B, 3, 3);
    boten[3] = new Boot("Onderzeeer", 1, 25, #677081, 4, 4);
}

```

```

void initVerdediging() {
    verdedigingsMechanismen = new Verdediging [2]; // Aantal typen verdedigingsMechanismen
    verdedigingsMechanismen[0] = new Verdediging("Afweer systeem", 1, -100, 5,
aantalVerdedigingsMechanismen);
    verdedigingsMechanismen[1] = new Verdediging("Mijn", 1, -50, 6, aantalVerdedigingsMechanismen);
}

```


3. opstellingen

```
1. void vulSpelopstellingRoosters()
    als roosterGroote = 11
        vul roosters [2] [3] en [4]
    als roosterGroote = 16
        vul roosters [5] [6] en [7]
    als roosterGroote = 21
        vul roosters [8] [9] en [10]
```

4. events

Variabelen

```
// CONSTANTEN VOOR score
final int MIS = 10;
final int RAAK = 11;
final int RAAKAFWEER = 12;
final int RAAKMIJN = 13;
```

Functies

```
1. void keyTyped
    keyPressedInStartScherf();
2. void mouseDragged()
    switch( spelStatus )
        case 0:
            mouseDraggedInStartScherf();
            break;

3. void mousePressed()
    switch( spelStatus )
        case 0:
            mousePressedInStartScherf();
            break;
        case 1:
            mousePressedInSpelScherf();
            break;
        case 2:
            mousePressedInScoreScherf();
            break;

4. void mousePressedInSituatieRooster(int situatieRoosterActieveSpeler, int
    opstellingRoosterActieveSpeler, int opstellingRoosterInactieveSpeler, int
    situatieRoosterInctieveSpeler)
    voor stopTijd is kleiner dan millis()
        voor alle rijen
            voor alle kolommen
                als klik valt binnen situatieRooster actieve speler
                    als cel is LEEG en cel behandeld is false
                        verander waarde cel in status rooster
                        verander kleur cel in status rooster
                        maak cel behandeld true
                        startTijd = millis();
                        beurtWissel();

                    anders als cel is boten[0].waarde en cel behandeld is
                    false
                        verander waarde cel in status rooster
                        verander kleur cel in status rooster
                        maak cel behandeld true
```

```

        hoog score op met boten[0].score;
        raakteller + 1

    anders als cel is boten[1].waarde en cel behandeld is
    false
        verander waarde cel in status rooster
        verander kleur cel in status rooster
        maak cel behandeld true
        hoog score op met boten[1].score;
        raakteller + 1

    anders als cel is boten[2].waarde en cel behandeld is
    false
        verander waarde cel in status rooster
        verander kleur cel in status rooster
        maak cel behandeld true
        hoog score op met boten[2].score;
        raakteller + 1

    anders als cel is boten[3].waarde en cel behandeld is
    false
        verander waarde cel in status rooster
        verander kleur cel in status rooster
        maak cel behandeld true
        hoog score op met boten[3].score;
        raakteller + 1
    als (spelers[actieveSpeler].aantalBeurten >=
    maxAantalBeurten en spelers[inactieveSpeler].aantalBeurten
    >= maxAantalBeurten) Of (spelers[actieveSpeler].raakTeller
    == AANTALBOOTDELEN of
    spelers[inactieveSpeler].raakTeller == AANTALBOOTDELEN)
        eindspel()

```

5. startScherM

Variabelen

```
int xSliders, breedteSliders, hoogteSliders;  
int yRoosterSlider, breedteRoosterSliderBlok;  
int yAfweerSlider, breedteAfweerSliderBlok;  
int startScherMarge = 65;  
int positieRoosterSlider = 1;  
int positieAfweerSlider = 10;  
int kleurKnop, startKnopX, startKnopY, startKnopBreedte, startKnopHoogte;  
int inputSpelerX, inputSpeler1Y, inputSpeler2Y, inputSpelerBreedte, inputSpelerHoogte;  
boolean klikOpInputSpeler1, klikOpInputSpeler2;
```

Funcities

1. void initialiseerStartScherMVariabelen()
2. boolean bepaalOfKlikOpKnop(int knopX, int knopY, int knopBreedte, int knopHoogte)
als muisX en muisY vallen binnen knop;
klikOpKnop = true
Return klikOpKnop
3. int bepaalPositieSlider (int ySlider, int breedteSliderBlok, int initPositieSlider, int
maxPositieSlider)
als muisX en muisY vallen binnen slider
positieSlider = (mouseX – xSliders) / breedteSliderBlok
return positieSlider;
4. void tekenSlider(int ySlider, int breedteSliderBlok, int positieSlider, String
sliderOpdrachtVoorSpeler, int tekstVoorPositieAanduiding, String tekstMin, String tekstMax)
//tekent een slider
5. void tekenKnop(int kleurKnop, int x, int y, int breedte, int hoogte, String tekst)
6. void tekenInvoerVelden()
//tekent invoer velden
7. int bepaalRoosterGroote(int positieRoosterSlider)
als positieRoosterSlider = 0
roosterGroote = 11
als positieRoosterSlider = 1
roosterGroote = 16
als positieRoosterSlider = 2
roosterGroote = 21
8. void keyPressedInStartScherM()
als klikOpInputSpeler1 = true
maakNickNameSpeler(0)
anders klikOpInputSpeler2 = true
maakNickNameSpeler(1)
9. void maakNickNameSpeler(int speler)
als key = BACKSPACE
als spelers[speler].naam.length() > 0
spelers[speler].naam = spelers[speler].naam -1
anders als key == SHIFT
println("Shift");
anders als key == '\n'
anders
spelers[speler].naam = spelers[speler].naam + key;

```

10. void mouseDraggedInStartScherM()
    positieRoosterSlider = bepaalPositieSlider (yRoosterSlider, breedteRoosterSliderBlok,
    positieRoosterSlider, 2);
    roosterGrootte = bepaalRoosterGrootte(positieRoosterSlider);
    positieAfweerSlider = bepaalPositieSlider (yAfweerSlider, breedteAfweerSliderBlok,
    positieAfweerSlider, 20);
    aantalVerdedigingsMechanismen = positieAfweerSlider;

11. void mousePressedInStartScherM()
    klikOpInputSpeler1 = bepaalOfKlikOpKnop(inputSpelerX, inputSpeler1Y,
    inputSpelerBreedte, inputSpelerHoogte)
    klikOpInputSpeler2 = bepaalOfKlikOpKnop(inputSpelerX, inputSpeler2Y,
    inputSpelerBreedte, inputSpelerHoogte)
    boolean klikOpStartKnop = bepaalOfKlikOpKnop(startKnopX, startKnopY,
    startKnopBreedte, startKnopHoogte);

    als klikOpStartKnop == true
        startSpel();

12. void tekenStartScherM()
    tekenSlider // rooster
    tekenSlider // afweer
    tekenInvoerVelden
    tekenKnop //startknop.

13. void startSpel()
    // roosterGrootte = 11;
    rijen = roosterGrootte; //roosterGrootte;
    kolommen = roosterGrootte; //roosterGrootte;
    roosterRuimte = (width - (3 * marge)) / 2;
    celZijde = roosterRuimte / rijen;
    maxAantalBeurten = round(((roosterGrootte - 1) * (roosterGrootte - 1))/2);
    println(maxAantalBeurten);

    //init classes
    initBoten();
    initVerdediging();
    initRoosters();
    actieveSpeler = floor(random(spelers.length));

    als actieveSpeler = 0
        inactieveSpeler = 1;
    anders
        inactieveSpeler = 0;

    //init rooster aspecten
    positionerenRoosters();
    vulSpelopstellingRoosters();
    switch(roosterGrootte)
        case 11;
            geef spelers een random opstelling rooster
        case 16:
            geef spelers een random opstelling rooster
        case 21:
            geef spelers een random opstelling rooster
    plaatsVerdediging();
    bepaalHintNummers();

    spelStatus = 1

```

5. spelScherm

Variabelen

```
final int AANTALBOTEN = 10;
final int LEEG = 0;
final int SLAGSCHIP = 1;
final int KRUISER = 2;
final int TORPEDOJAGER = 3;
final int ONDERZEEER = 4;
final int AFWEER = 5;
final int MIJN = 6;
final int AANTALBOOTDELEN = 20;
```

Functies

1. void mousePressedInSpelScherm()
2. void tekenSpelScherm()
 - als beurtWissel
 - als stoptijd > millis()
 - teken elementen incl NotificatieBeurtwissel
 - anders
 - teken elementen excl NotificatieBeurtwissel
 - anders
 - teken elementen excl NotificatieBeurtwissel
3. void tekenLegenda(int legendaX, float legendaY, int zijde, int aantalBlokjes)
 - voor (int blokje = 0; blokje < aantalBlokjes; blokje ++)
 - switch voor kleur en tekst
 - teken blokje
4. void tekenStatusBalk
5. void tekenNotificatieBeurtwissel
6. void positionerenRoosters
 - voor alle roosters
 - voor alle rijen
 - voor alle kolommen
 - als rooster < 2
 - plaats rooster rechts
 - anders
 - plaats rooster links
7. void plaatsAfweerSystemen(int opstellingsRooster)
 - teller = 0
 - zolang teller is niet gelijk aan aantalAfweersystemen
 - kies willekeurige rij
 - kies willekeurige kolom
 - als rooster [opstellingsRooster][willekeurige rij][willekeurige kolom] is leeg
 - rooster [opstellingsRooster][willekeurige rij][willekeurige kolom] =
 - afweer
 - teller ++
8. void plaatsMijnen(int opstellingsRooster)
 - teller = 0
 - zolang teller is niet gelijk aan aantalAfweersystemen
 - kies willekeurige rij
 - kies willekeurige kolom
 - als rooster [opstellingsRooster][willekeurige rij][willekeurige kolom] is leeg
 - rooster [opstellingsRooster][willekeurige rij][willekeurige kolom] =
 - afweer
 - teller ++

```

9. void plaatsVerdediging()
    voor alle spelers
        plaats afweersystemen
        plaats mijnen

10. void bepaalHintNummers()
    voor alle roosters
        voor alle rijen
            voor alle kolommen
                als cel inhoud > leeg en cel inhoud < afweer
                    roosters[rooster][0][kolom].status ++

    voor alle roosters
        voor alle rijen
            voor alle kolommen
                als cel inhoud > leeg en cel inhoud < afweer
                    roosters[rooster][rij][0].status ++

11. void kleurCellen(int rooster, int rij, int kolom)
    als rij = 0 of kolom = 0
        kleur cel is kleur roosters
    anders als roosters [rooster] [rij] [kolom] == leeg
        kleur cel is kleur roosters
    anders als roosters [rooster] [rij] [kolom] == boten[0].waarde
        kleur cel is boten[0].kleur
    anders als roosters [rooster] [rij] [kolom] == boten[1].waarde
        kleur cel is boten[1].kleur
    anders als roosters [rooster] [rij] [kolom] == boten[2].waarde
        kleur cel is boten[2].kleur
    anders als roosters [rooster] [rij] [kolom] == boten[3].waarde
        kleur cel is boten[3].kleur
    anders als roosters [rooster] [rij] [kolom] == verdedigingsMechanismen[0].waarde
        kleur cel is zwart
    anders als roosters [rooster] [rij] [kolom] == verdedigingsMechanismen[1].waarde
        kleur cel is blauw

12. void tekenRooster( int rooster)
    voor elke rij
        voor elke kolom
            teken rooster
            als rij = 0 of kolom = 0
                als rooster = spelers[0].situatieRooster
                    tekst in cel is
                    roosters[spelers[1].opstellingRooster][rij][kolom].status
                anders als rooster = spelers[1].situatieRooster
                    tekst in cel is
                    roosters[spelers[0].opstellingRooster][rij][kolom].status
            anders
                tekst in cel is roosters[rooster][rij][kolom].status

13. void eindSpel()
    spelStatus = 2

```

eindScherm

functies

1. void mousePressedInScoreScherm()
 boolean klikOpKnop = bepaalOfKlikOpKnop
 als klikOpKnop = true
 herstartSpel
2. void tekenScoreScherm()
 tekst winnaar
 knop herstart
3. int bepaalWinnaar()
 voor alle spelers
 als spelers[speler].totaalScore > hoogsteScore
 hoogsteScore = spelers[speler].totaalScore;
 winnaar = speler;
4. void herstartSpel()
 spelStatus = STARTSCHEM;
 setup();
 plaats sliders op default positie
 geef sliders de default waarde

Test plan en uitvoering

startScherm

Testcase	Actie	Verwacht resultaat	Geslaagd
1	Slider op positie 1 zetten	Een spelbord van 11 bij 11 cellen	Ja
2	Slider op positie 2 zetten	Een spelbord van 16 bij 16 cellen	Ja
3	Slider op positie 3 zetten	Een spelbord van 21 bij 21 cellen	Ja
4	Invoeren naam "Lauren"	Naam speler wordt Lauren	Ja
5	Invoeren hele lange naam "Ruud Jan Erik Bob Kees Luuk van Der Linden"	Foutmelding voor speler naam mag niet zo lang zijn	Nee

spelScherm

Testcase	Actie	Verwacht resultaat	Geslaagd
1	Klik in opstelling rooster	Er gebeurt niets	Ja
2	Klik in statusbalk	Er gebeurt niets	Ja
3	Klik in legenda	Er gebeurt niets	Ja
4	Klik in opstelling rooster	Er gebeurt niets	Ja
5	Klik in situatie rooster MIS	Beurten speler - 1 Laat beurt wissel indicatie zien Beurt wissel	Ja
6	Klik in situatie rooster RAAK	Beurten speler - 1 Punten speler + <code>booten[].score</code>	ja
7	Klik in situatie rooster AFWEER	Beurten speler - 1 Punten speler + <code>verdedegingsMechanismen[0].score</code> Laat cel zien in situatie rooster andere speler Beurtwissel	ja
8	Klik in situatie rooster MIJN	Beurten speler - 1 Punten speler + <code>verdedegingsMechanismen[1].score</code> Laat beurt wissel indicatie zien Beurtwissel	Ja
9	Klik in hint nummers	Er gebeurt niets	Ja

eindScherm

Testcase	Actie	Verwacht resultaat	Geslaagd
1	Klik in scherm	Er gebeurt niets	Ja
2	Klik op herstartKnop	Spel herstart	Ja

Conclusie

Mijn versie van zeeslag is het eerste complexe programma wat ik heb gebouwd. Ik vond het erg leuk om na te denken over wat het spel moest doen en hoe het eruit moest komen te zien.

Ik vond het erg lastig om een strakke planning te maken en niet te lang te blijven hangen in details. Hierdoor heb ik veel meer tijd besteed aan de opdracht dan nodig en ben ik niet helemaal tevreden met het uiteindelijke resultaat.

Ik ben echter wel heel trots op de groei die ik in 10 weken heb kunnen maken. In week één hadden wij onze eerste ervaring met processing. Dit vond ik leuk en spannend, ik wilde graag alles weten. Nu drie maanden later is het mij gelukt een werkend spel te maken. Daar is nog wel het een en ander op aan te merken maar het werkt wel!

Ik moet het product nu opleveren maar ik zou er graag in mijn eigen tijd aan door werken om er helemaal tevreden over te zijn. Indien mogelijk zou ik het er dan graag nog even met Mark willen bespreken.