

Technisch Ontwerp

Project Team Building Challenge



Klas: EHI1V.Sb

Team: 2

Inhoudsopgave

Inhoudsopgave	2
Ontwikkelpatform	3
Hardware.....	3
Software.....	3
Installatiehandleiding	3
Modellen	6
Basismodellen	6
Klassendiagram	6
Overige diagrammen.....	9
Sequentiediagram	9
Wireframe	10
Autopilot.....	13
Testplan	14
Acceptatiecriteria	15

Documenthistorie:

Versie	Datum	Gewijzigd door	Wijziging
0.1	26-04-2021	Emanuel, Kalli en Jochem	Modellen, Testplan
1.0	21-05-2021	Emanuel, Kalli en Jochem	Ontwikkelpatform, klassendiagram
1.1	04-06-2021	Emanuel, Kalli en Jochem	Aanpassingen voor sprint 2
1.2	29-06-2021	Emanuel, Kalli en Jochem	Aanpassingen voor sprint 3

Ontwikkelform

Beschrijf zo compleet mogelijk wat een ontwikkelaar aan hardware nodig heeft en aan software moet installeren om ontwikkeling aan het project te kunnen doen.

Hardware

Een volledig inzicht in de te gebruiken hardware om de applicatie te ontwikkelen. Maak eventueel een onderscheid in minimale eisen en geadviseerde eisen.

Categorie	Product
CPU	Pentium 2 266 MHz
RAM	4GB 1066 MHz DDR3

Software


Inventarisatie software die nodig is om de applicatie te creëren zoals:

- Programmeertalen,
- Tooling/ IDE,
- Bibliotheken/ modules/ softwarepakketten (inclusief gebruikte versienummers!),
- Operating System(s)
- Enz. Enz.

Categorie	Product
Besturingssysteem	Windows 7+, Linux, MacOS X 10.8.3+
Platform	Java SDK 11
IDE	IntelliJ IDEA
Framework	Java Swing

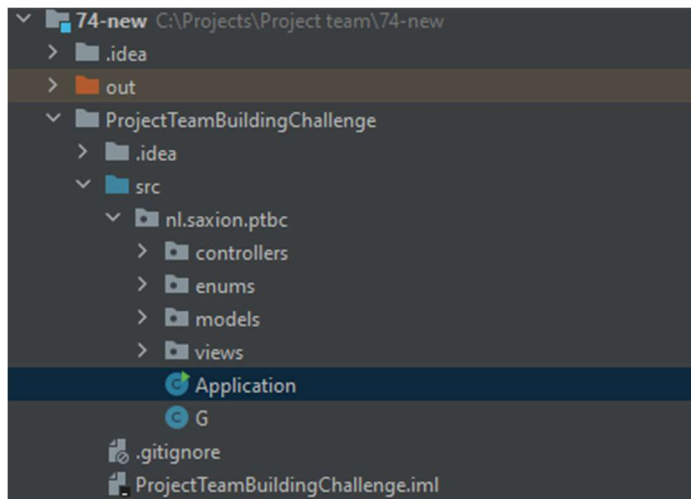
Installatiehandleiding

Om de applicatie werkend te krijgen moet er eerst een IDE geïnstalleerd worden. Voor de development is er gebruikt gemaakt van [intelliJ](#). Deze wordt ook aangeraden.

Dan moet de code gedownload worden via [deze](#) website. Klik dan op dit  icon om het te downloaden. De source code is nog ingepakt in een zip of tar bestand. Deze moet eerste uitgepakt worden.


Daarna kan de folder met de naam "74-master" geopend worden in uw gekozen IDE.

Om het programma werkend te krijgen moet eerst de applicatie opgestart worden. Zie afbeelding hier onder

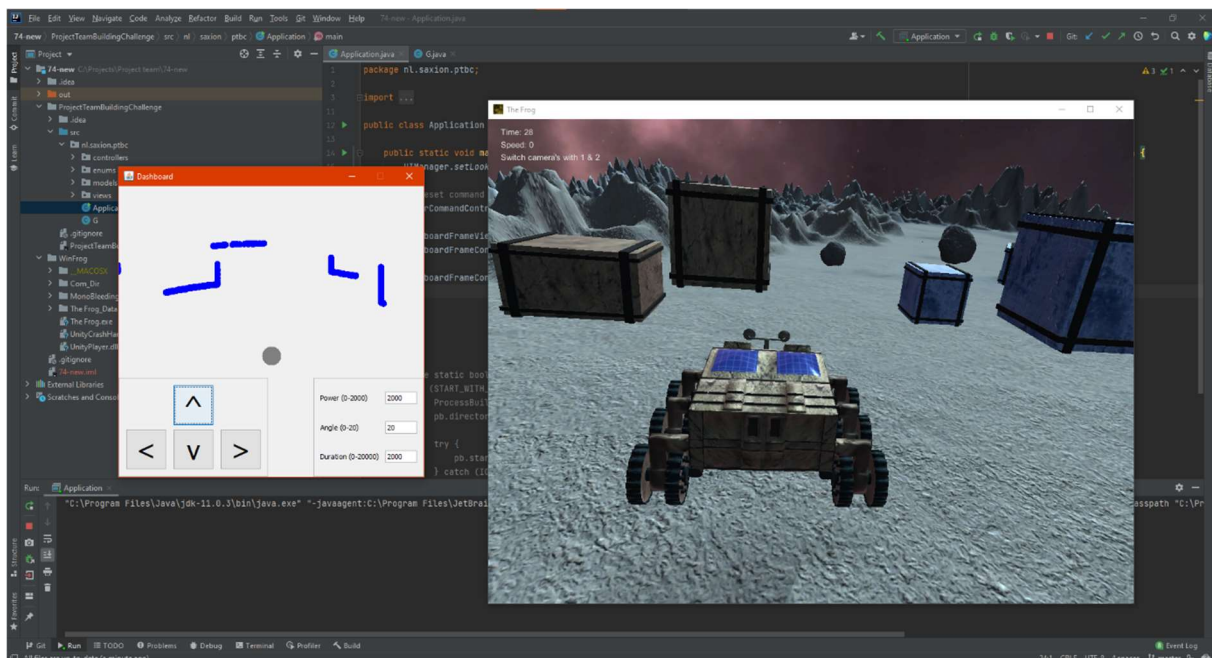


Open het Application bestand.



Dan klik op het dit  icoontje om het programma op te starten.

Hierna kan pas de The Frog-game opgestart worden. Deze bevindt zich in de WinFrog folder en heeft de naam "The Frog.exe".



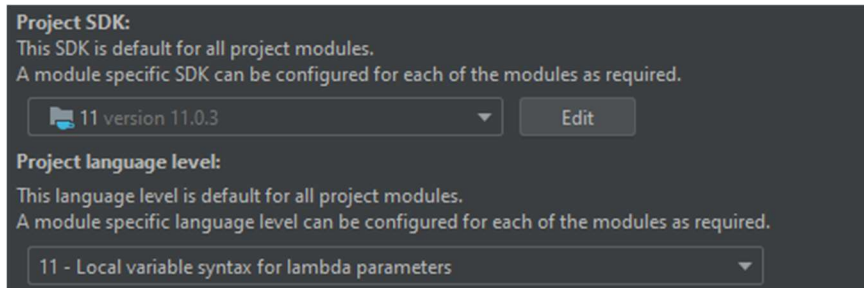
Dit ongeveer het eindresultaat

Zijn er problemen?

Vergeet niet eerste de applicatie op te starten voor dat the Frog-game is opgestart.

Allemaal errors?

Probeer eens op CTRL+ALT+SHIFT+S te drukken en dan te kijken of Project SDK en Project language level allebei op 11 staan.

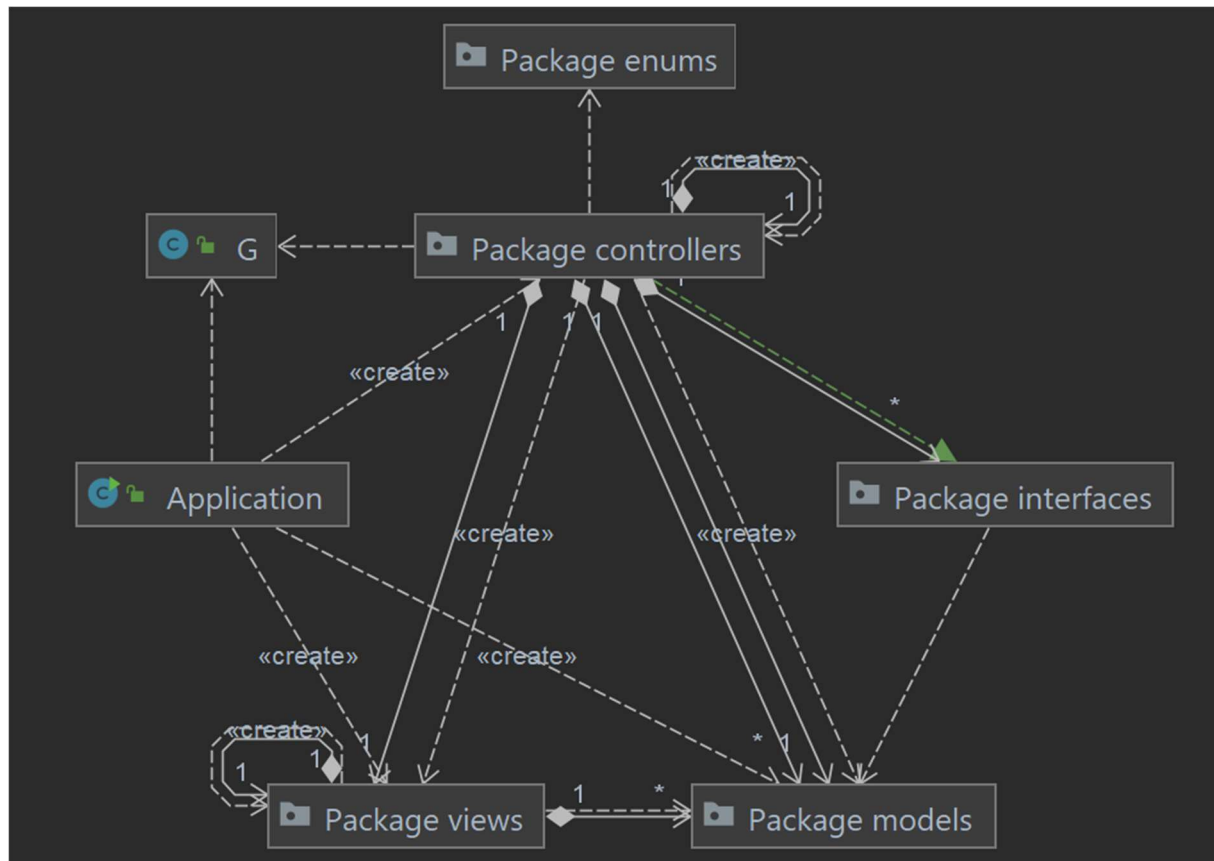


Modellen

Basismodellen

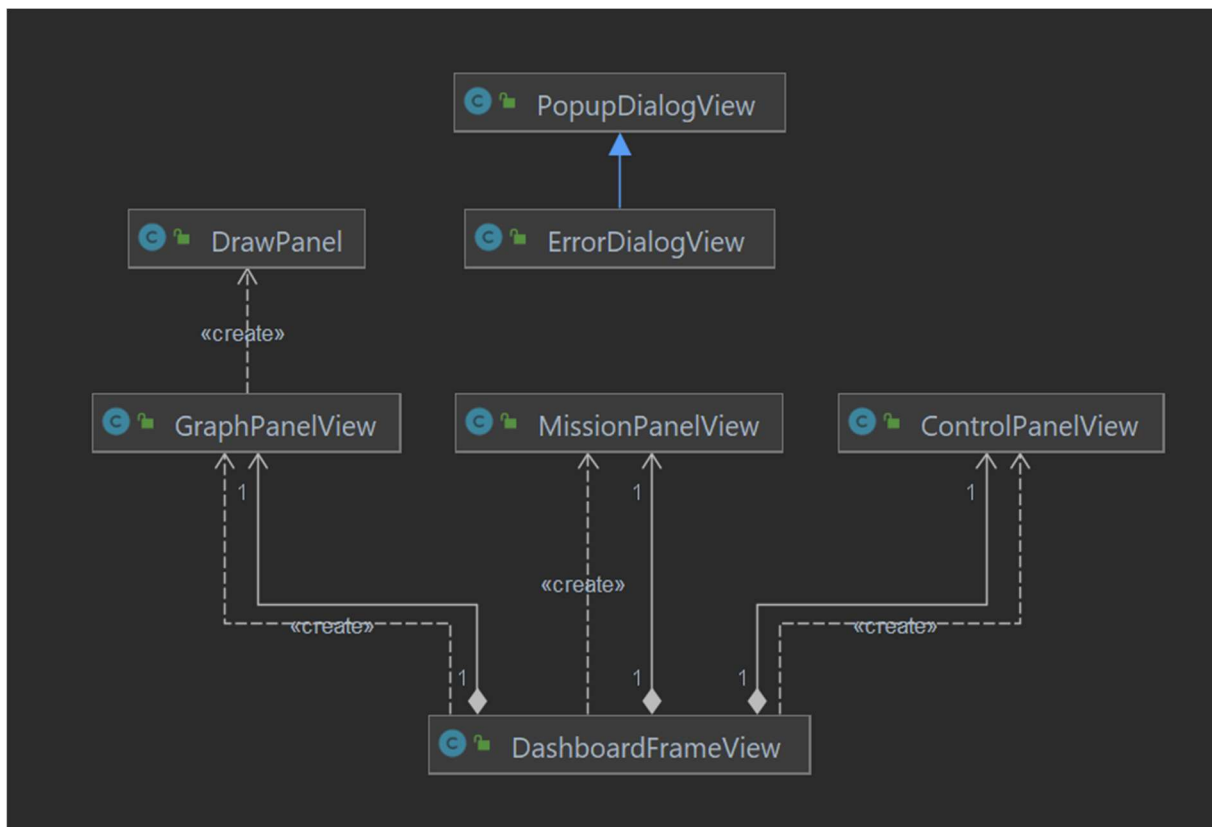
Klassendiagramm

Packages



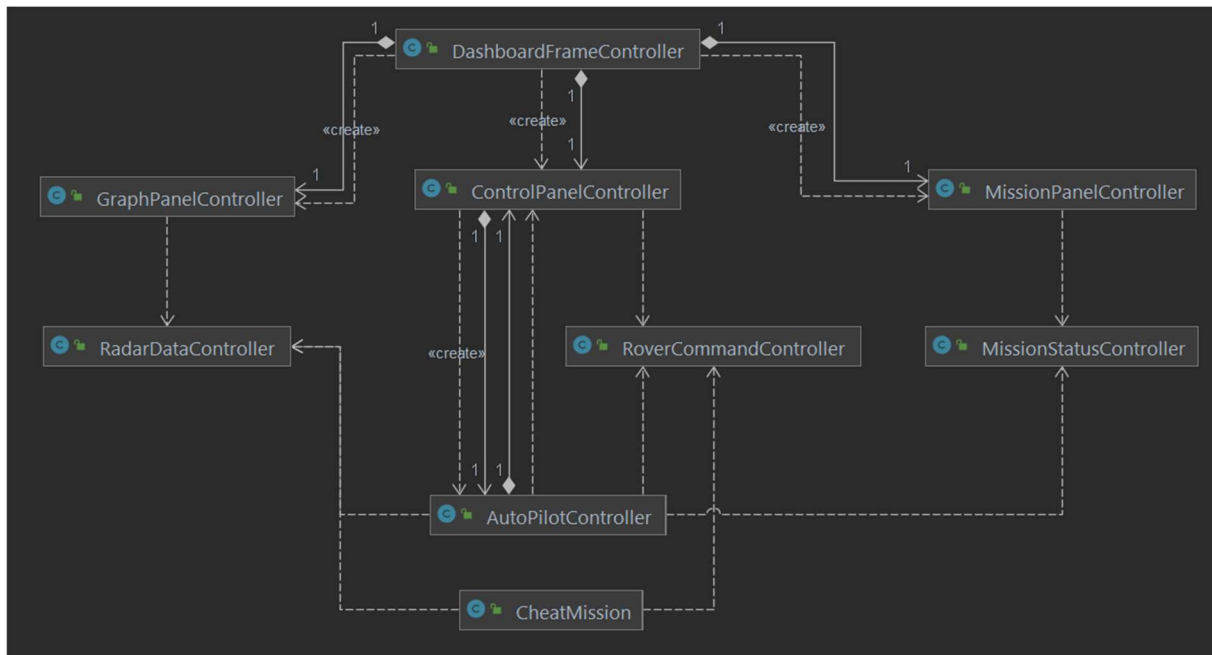
Dit diagram geeft de samenhang van de packages, de Application klas en de Global klas weer. De Global klas heeft variabelen die alle andere klassen kunnen gebruiken. Zoals te zien is, hebben de MVC packages een Pyramide relatie met elkaar. Waarbij de modellen soms een één op meer relatie zijn.

Views



De DashboardFrame is de hoofd window van de applicatie. Het heeft een GraphPanel, een MissionPanel en een ControlPanel die allemaal hun eigen data weergeven. De GraphPanel heeft ook een DrawPanel om de punten van de radar data te tekenen. Buiten dit hele systeem om zijn er ook 2 popup klassen. PopupDialog is een generieke popup die alle soorten berichten weer zou kunnen geven. ErrorDialog breidt zich uit op de popup klas en kan de applicatie sluiten als de error te erg is.

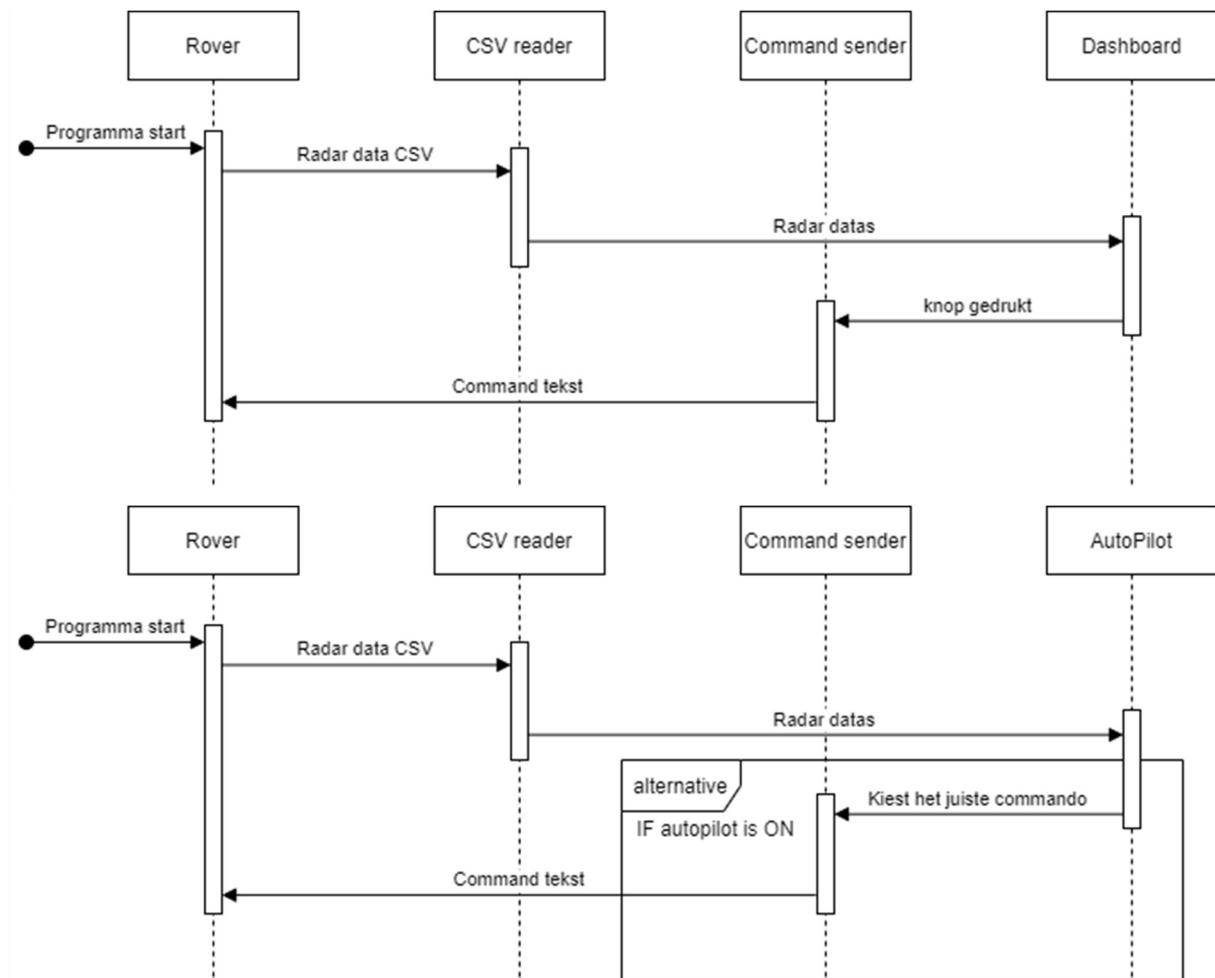
Controllers



De frame en panel controllers hebben dezelfde hiërarchie als de views. Dit is omdat ze deze views beheren en functioneel maken. Omdat alle 3 panels indirect verbonden zijn met de rover input/output files, heeft elke panel controller ook een utility klas die de rover data beheert of data naar de rover toe stuurt. Verder is er ook een AutoPilot die de rover zelfstandig beweegt en een CheatMission die help bij het testen en debuggen van de missie elementen.

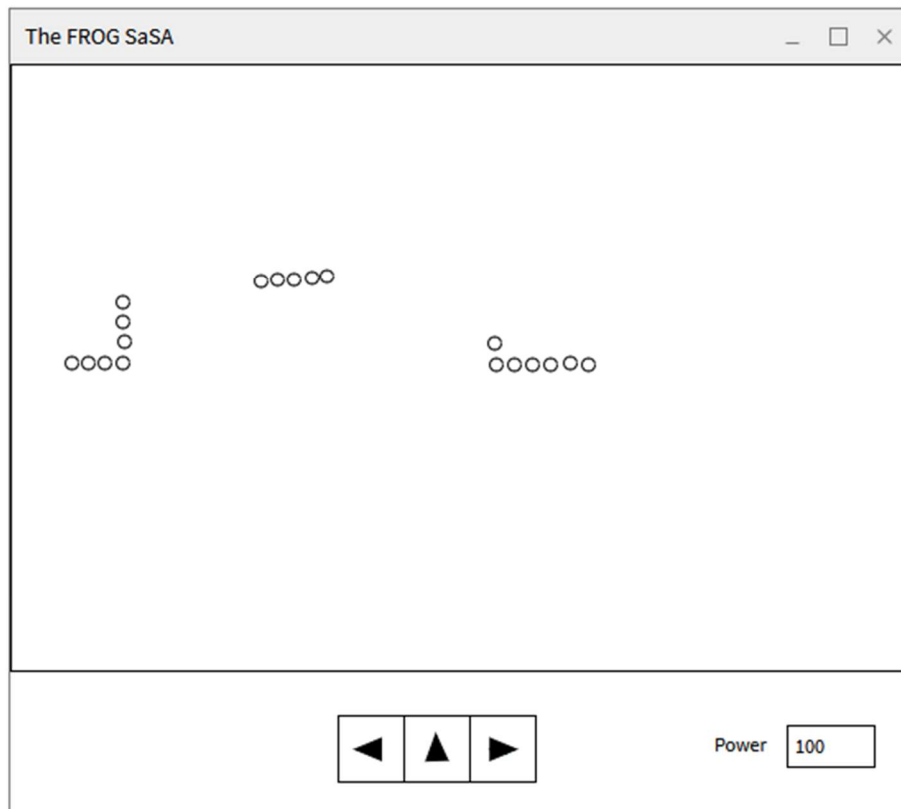
Overige diagrammen

Sequentiediagram

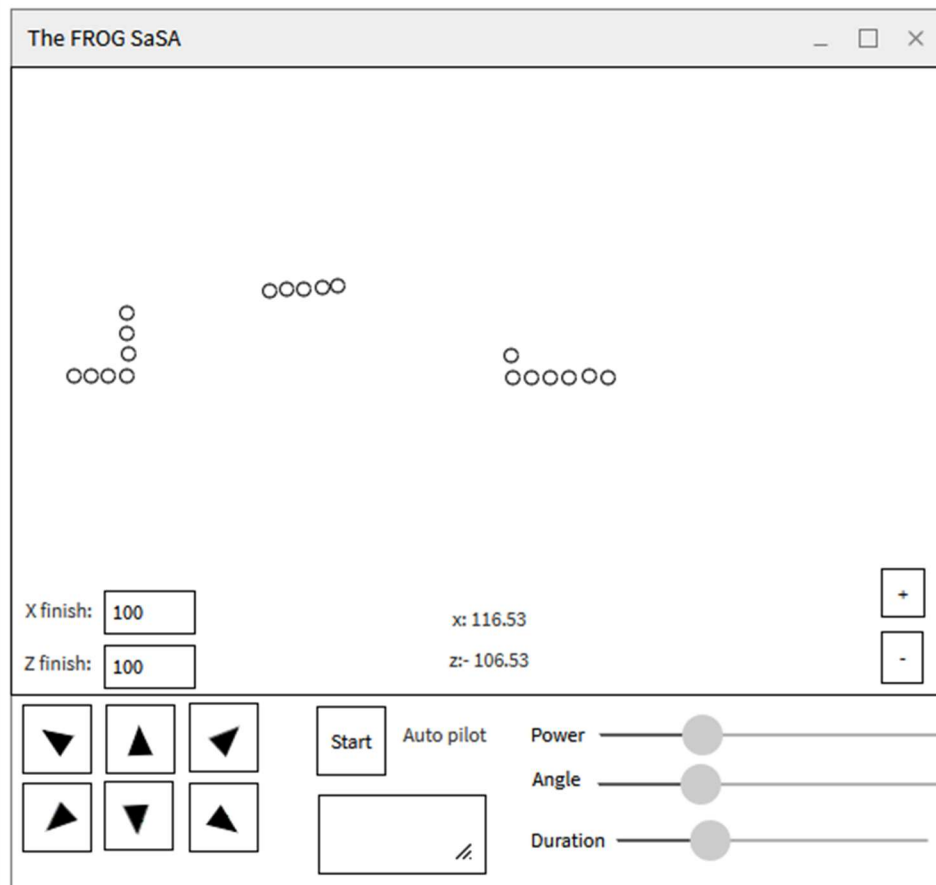


Deze Sequentiediagrammen moeten een goed beeld geven aan het proces cycle waar het programma door loopt. Ook de autopilot loopt dit proces door. Er wordt alleen niet letterlijk op een knop gedrukt.

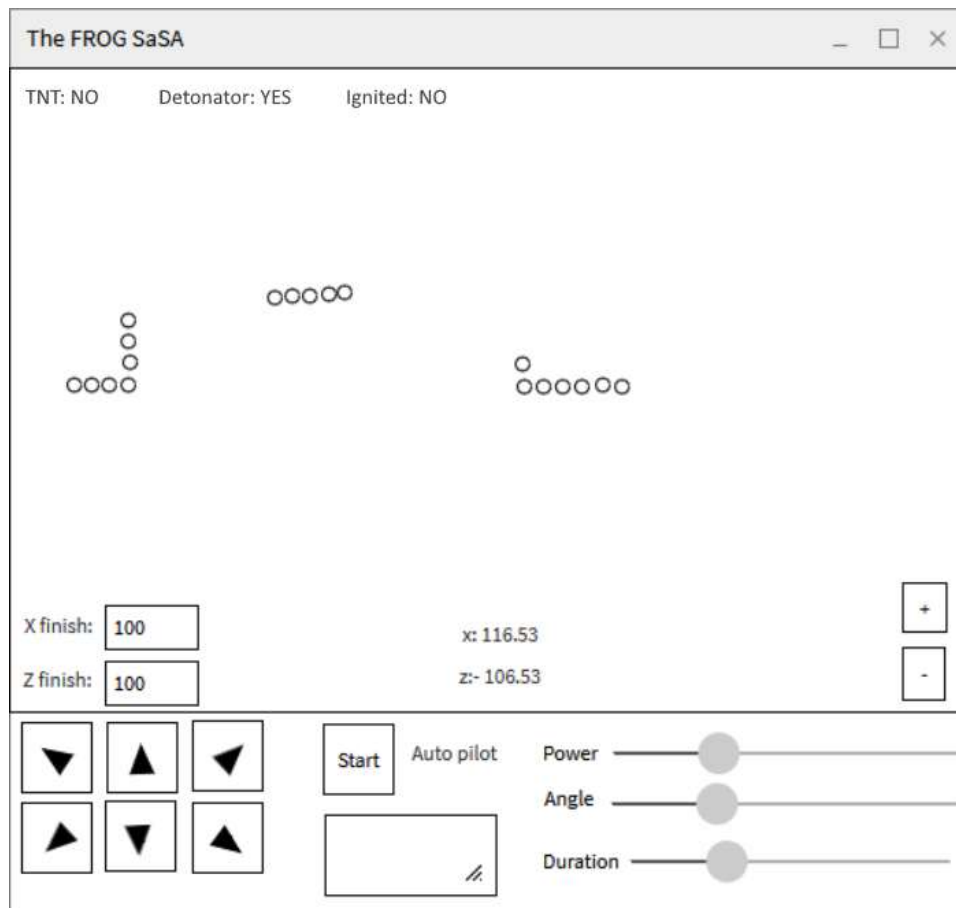
Wireframe



Dit is de wireframe van de applicatie zoals de applicatie er aan het eind van de eerste sprint uitzag. We wilde voor het maken van een Swing UI deze wireframe maken om te zien of de lay-out mooi/kloppend zou zijn.



Dit is de wireframe van de applicatie zoals hij er na de tweede sprint uit ziet. Met dus meer mogelijkheden om te sturen maar ook het starten en stoppen van de auto pilot die je hier ziet.

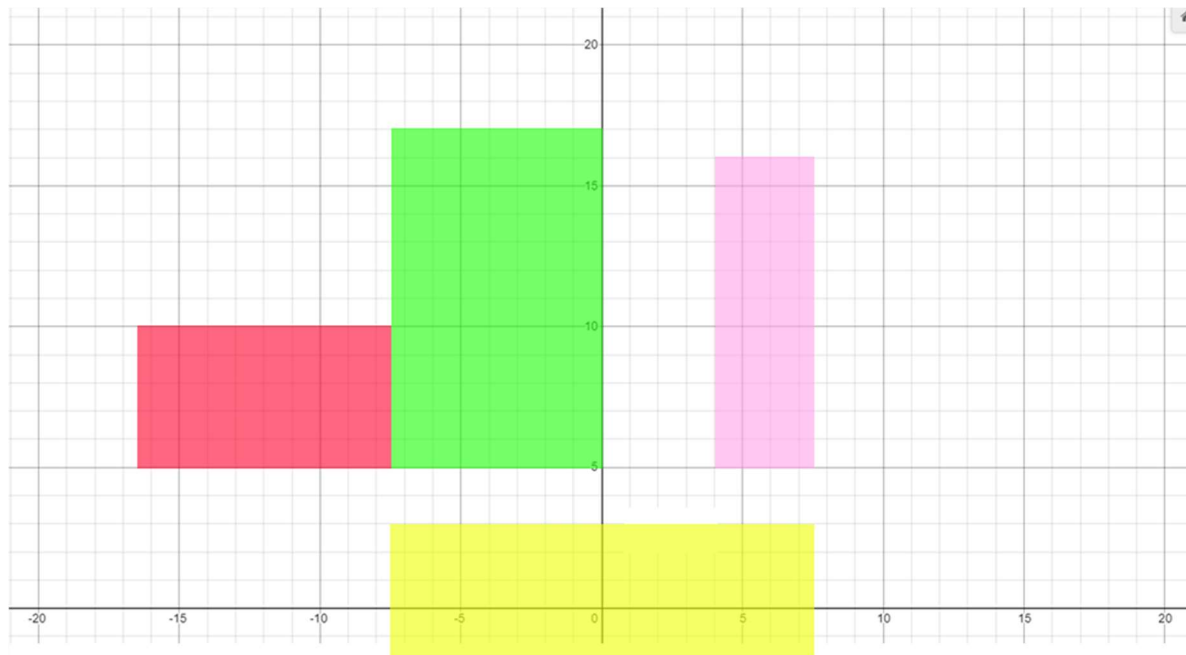


Er is nu een nieuw data te zien linksboven in het scherm. Deze data laat zien hoever de rover in zijn missie is. Als ignited op YES staat gaat de rover gecontroleerd naar achter rijden en wacht geduldig op de ontploffing. Daarna rijdt de rover weer voor uit en vervolgt zijn autopilot gedrag.

Autopilot

De autopilot neemt het sturen van de rover over. Het zoekt een linker muur op en laat de rover langs deze muur rijden.

Het kijkt naar verschillende zones binnen de radar data en beslist hiermee wat de volgende beweging word. Als er datapunten aanwezig zijn in een zone word de zone als actief gezien.



Hiërarchie	Zones	Beslissing
1	Geel	Achteruit
2	Groen	Rechts
3	Roze	Links
4	Rood	Rechtdoor
5	Geen	Links

Er zijn twee extra checks die problemen met het systeem oplossen.

Als de rover vast zit, word geen radar data gestuurd en word de rover ook niet om een nieuw commando gevraagd. Er is een timer die elk commando herstart wordt. Als er na zeven seconden geen nieuwe radar data beschikbaar is, gaat de rover achteruit.

In specifieke gevallen zou de rover rond een object kunnen blijven draaien. Als geel, groen en rood voor 20 commando's leeg zijn, gaat de rover naar rechts.

Testplan

In dit project gaat er gebruikt gemaakt worden van unit testen. Met Java is de JUnit Test Framework daar erg geschikt voor. Met deze framework kunnen er tests opgezet worden in een Java class. Zie afbeelding hier onder.

```
import junit.framework.*;

public class JavaTest extends TestCase {
    protected int value1, value2;

    // assigning the values
    protected void setUp(){
        value1 = 3;
        value2 = 3;
    }

    // test method to add two values
    public void testAdd(){
        double result = value1 + value2;
        assertTrue(result == 6);
    }
}
```

Alles wat direct in ons swing applicatie aan code wordt uitgevoerd kan hier getest worden. Denk aan bekeringen, algoritmes, data manipulatie.

Wel moeten er requirements getest worden wat buiten de applicatie vallen. Denk aan het gedrag van de rover. Ook zijn er visuele requirements in de applicatie zoals de manier van data weergave van de rover.

Deze testen worden in een testrapport gestopt. Dit is een overzicht van alle testen die er zijn gemaakt in het project. Ook laat het zien bij welke requirement de test hoort en of deze test is voltooit.

Acceptatiecriteria

Zorg er ook altijd voor dat het voor de Product Owner c.q. klant duidelijk is dat er voldaan is aan de (welke) acceptatiecriteria m.a.w. wat is er van de requirements opgeleverd (wat krijgt de klant voor zijn geld?).

Nr. Req.	Requirement	Acceptatietest	Ok
1	Als Sasa medewerker wil ik de Radar data op een dashboard te zien krijgen.	Blauwe punten worden weergegeven op de plek van botsing.	X
2	Als Sasa medewerker wil ik de maanrover op afstand kunnen besturen.	Als er op een knop wordt gedrukt is er feedback in de Unity game. Bijvoorbeeld, de rover rijdt naar voren.	X
3	Als Sasa medewerker wil ik dat de maanrover op zichzelf naar het voor af aangegeven punt kan komen.	Als De maanrover stil staat moet hij op de juiste coördinaten staan.	X
4	Als Sasa medewerker wil graag dat de rover de blockkade vernietigd en daarna het Saxion space center binnen rijdt.	Als De maanrover de TNT laadt ontploffen, een veilige afstand neemt en door de lange tunnel rijdt van het Saxion space center.	X