Verslag Softwareproject 2025 Team 2

SolarSim

De Laet Renaud
Vanhoutte Seppe
Ponomariov Maxim
Bossuyt Mathis
Inghelbrecht Thomas

Ongenae Veerle en Brouns Leen

Inhoudstafel

Inleiding

Het project SolarSim richt zich op het simuleren van een zonnestelsel, waarbij de zwaartekracht tussen de hemellichamen zo nauwkeurig mogelijk wordt berekend via het Barnes-Hut algoritme. Dit algoritme zorgt voor een efficiënte modellering, wat heel belangrijk is voor een realistische simulatie met meerdere hemellichamen.

Via de applicatie zullen de gebruikers zelf planeten kunnen toevoegen, en meegeven welke snelheid en massa die hebben. Daarnaast zullen geconfigureerde zonnestelsels opgeslagen, ingeladen en gedeeld kunnen worden via tekstbestanden. Verder zal de applicatie ook andere dingen bijhouden, zoals de mogelijkheid van leven op een planeet en botsingen.

Het doel van dit project is om een gebruiksvriendelijke en aantrekkelijke simulatie te ontwikkelen die niet enkel de zwaartekracht tussen hemellichamen realistisch simuleert, maar ook ruimte laat voor verdere uitbreidingen en verfijningen zoals kometen, meerdere sterren…

Structuur

Code

De code wordt gestructureerd in verschillende klassen waarbij sommige van elkaar zullen erven en anderen volledig los zullen staan van elkaar. Hier kunnen natuurlijk nog kleine veranderingen aan komen tijdens het programmeren.

Om te beginnen zal er een klasse ‘Zonnestelsel’ zijn. Deze klasse zal alles updaten (methode “update”) wanneer een planeet wordt toegevoegd. De klasse zal ook geregeld updates uitvoeren in verband met de baan van de planeten (N-body problem algoritme). De klasse “Zonnestelsel” zal de belangrijke methoden “toevoegen” en “botsingcheck” hebben. Deze zullen er voor instaan om enerzijds planeten te kunnen toevoegen en te checken als er een botsing plaatsvindt. Indien er een botsing plaatsvindt zal er doorverwezen worden naar een andere methode "botsing", deze zal bepalen hoe de botsing wordt afgehandeld.

Daarnaast is er ook de klasse die meer gericht is op het opslaan van het zonnestelsel en het inladen van het zonnestelsel. Zo is er de klasse “opslaan” die alles zal afhandelen op dit vlak met behulp van de methoden “opslaantekst”, “opslaanafbeelding” en “inladen” (hier zal je het tekstbestand kunnen inladen dat eerder is opgeslaan).

Ook op vlak van de UI komt er een klasse, namelijk “UImanager”. Deze zal verschillende methoden hebben op vlak van gebruiker gerichte toepassingen. Zo is er de methode “kleurmodus”, hier zal je dan de kleurmodus kunnen aanpassen, deze methode kan ook gebruikt worden om rekening te houden met kleurenblindheid. De methode “slider” zal dan weer de functie hebben om ervoor te zorgen dat de gebruiker de snelheid van de tijd kan bepalen (en dus tijd rapper of trager te laten verdergaan, tot zelfs pauzeren van de tijd). Als laatste grote methode komt er dan nog “VisueleUitleg”. Deze methode opent een nieuwe tab waarin het gebruikte algoritme visueel wordt uitgelegd.. Dit deel van de webpagina zal iets minder interactief zijn en ook iets minder methoden nodig hebben.

Tot slot is er dan nog alles met betrekking tot de ruimte objecten, zoals kometen, de ster, planeten. Hiervoor zullen we 1 abstracte klasse “RuimteObject” gebruiken waar verschillende ruimte objecten dan van zullen kunnen erven. Zo kunnen er, aan de hand van hoeveel tijd er over is, gemakkelijk extra soorten ruimteobjecten toegevoegd worden. Deze klasse zal de volgende eigenschappen bijhouden: snelheid, positie, afbeelding. De klasse die er zeker komt is de klasse "planeet", deze zal dan de extra eigenschappen: “toestand”, “leven” bijhouden. De klasse zal dan ook de methoden ‘leven check’ en ‘toestandcheck’ gebruiken. Beide methoden zullen controleren welke toestand de planeet kan hebben en als leven op de planeet mogelijk is. Ook de klasse "ster" komt er zeker, het enige dat deze klasse zal bijhouden is de grootte en gewicht van de ster zodat we kunnen checken dat geen enkele planeet groter en/of zwaarder wordt dan de ster.

De methode “reset” zorgt ervoor dat het huidige zonnestelsel reset en er opnieuw kan worden begonnen

Gebruiker

Bij het inladen van de website zal de gebruiker de mogelijkheid krijgen om een bestaand planetenstelsel in te laden of om zelf een te maken. Als de gebruiker ervoor kiest om zelf een stelsel te maken kan dit aan de hand van knoppen zoals toevoegen, pauzeren enz… Deze knoppen zullen allemaal vrij intuïtief verlopen.

Planeten toevoegen zal gebeuren via een pop-up waar dan verschillende waarden zullen kunnen worden ingesteld. Als de gebruiker niets invult zal de website standaardwaarden genereren. Na het invullen van deze waarden zal de gebruiker kunnen aantonen op het stelsel waar de planeet moet worden ingeladen.

Indien er meerdere planeten worden toegevoegd wordt de kans op een botsing ook steeds groter. Wanneer een botsing zich voordoet zal 1 van de botsende planeten verdwijnen. De andere planeet zal veranderen van toestand. Zo kan de planeet bijvoorbeeld van ijs planeet veranderen in een rotsplaneet, of kan het aantal manen veranderen.

De gebruiker zal ook de optie hebben om met een slider de snelheid van het stelsel aan te passen, tot de optie om te pauzeren.

Ook de optie om het stelsel op te slaan zal aanwezig zijn. De gebruiker krijgt dan de optie om het stelsel op te slaan als afbeelding, tekstbestand of beide. Als ervoor gekozen wordt om als tekstbestand op te slaan dan kan dit bestand later gebruikt worden om het weer in te laden of om het door te sturen naar andere mensen.

In de instellingen zal de gebruiker ook verschillende visuele aanpassingen kunnen doen. Zo kan je een optie kiezen voor kleurenblindheid of de kleuren aanpassen naar je persoonlijke voorkeur.

De gebruiker kan specifieke ruimteobjecten selecteren en meer in detail bekijken, zoals snelheid, baan en of er leven mogelijk is op een planeet. Dit zou ook kunnen aan de hand van een lijst die onder het stelsel zal komen te staan. Ook planeten verwijderen zal vanaf deze lijst mogelijk zijn.

Barnes-Hut algoritme

Voor het oplossen van het n-body probleem wordt gebruikgemaakt van de Barnes-Hut simulatie, een benaderingsalgoritme dat een boomstructuur gebruikt om de krachten tussen de lichamen efficiënt te berekenen. Aangezien de simulatie in twee dimensies plaatsvindt, wordt een quadtree gebruikt. Elke knoop in deze boom bevat een deel van de ruimte, waarbij de wortel de volledige ruimte omvat. De ruimte wordt recursief in vier kwadranten verdeeld totdat elke bladknoop maximaal één lichaam bevat.

In elke interne knoop van de boom worden het massamiddelpunt en de totale massa van de deeltjes berekend. Om de kracht op een deeltje te bepalen, wordt de boom doorlopen vanaf de wortel, waarbij een acceptatiecriterium wordt gebruikt die bepaalt of een interne knoop als één massapunt kan worden beschouwd, of dat die knoop verder doorlopen moet worden. Dit acceptatiecriterium hangt af van:

- De grootte van de cel (s)
- de afstand tussen het deeltje en het massamiddelpunt van de cel (*d*)
- een zelf in te stellen parameter (*θ*)

Een interne knoop wordt als één massapunt beschouwd indien $s/d < \theta$. De exacte waarde van *θ* zal pas bekend zijn na het testen van de simulatie.

Zodra de krachten op elk deeltje zijn berekend via de gravitatiewet

$F = (G * m1 * m2)/r^2$ $G = 6,67 * 10^{-11}$

worden de versnellingen bepaald met behulp van de 2e wet van Newton.

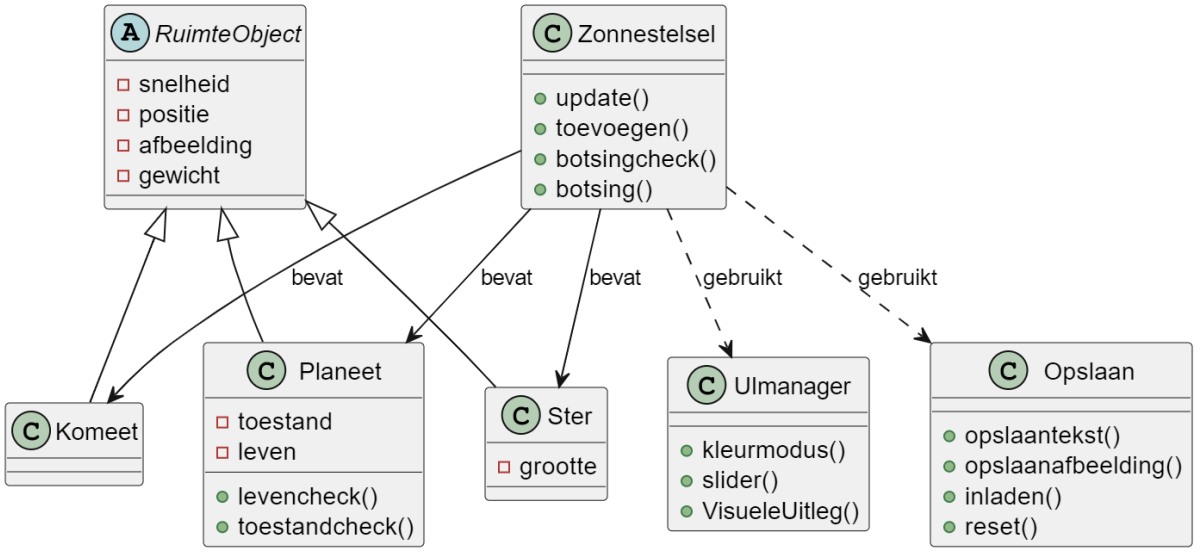
$F = m * a$

Dit proces wordt herhaald bij elke tijdstap van de simulatie.

Diagrammen

Er werden 3 UML-diagrammen gemaakt om de structuur en de werking van bepaalde methoden van SolarSim te verduidelijken. Ten eerste werd er een klassendiagram gemaakt om de algemene structuur van de code in kaart te brengen. Vervolgens werd er een sequentiediagram gemaakt om de werking van de methode ‘toevoegen’ uit te leggen en als laatste werd eenzelfde type diagram gemaakt maar dan voor de methode “reset”. Deze methoden werden gekozen omwille van hun grote belang binnen het project en een vrij complexe manier van fungeren.

Klassendiagram



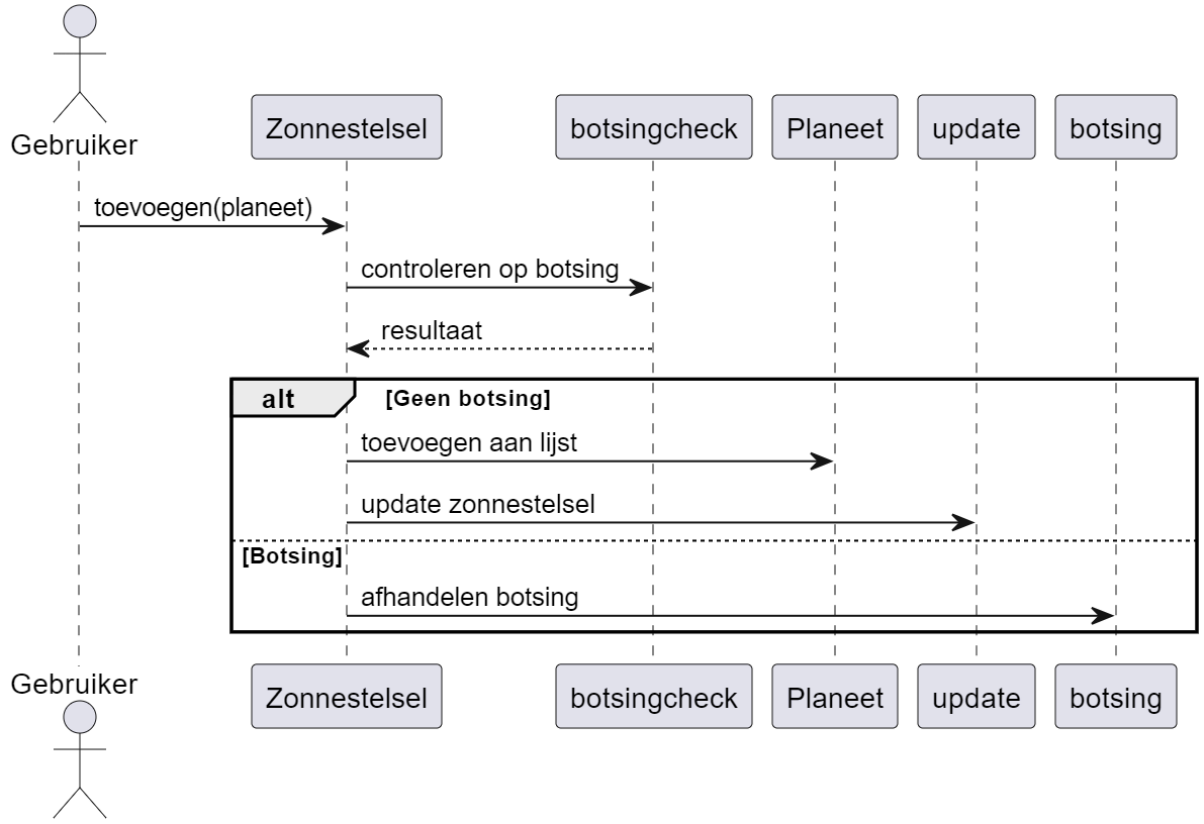
Figuur 1: Klassendiagram bouw zonnestelsel

Het klassendiagram op Figuur 1 geeft een overzicht van de structuur van het systeem en hoe de verschillende klassen met elkaar interageren. De kern van het systeem wordt gevormd door de klasse ‘Zonnestelsel’. Deze is verantwoordelijk voor het beheer van kometen, sterren en planeten. Zoals te zien op het diagram is de klasse ‘RuimteObject’ een abstracte klasse waarvan er specifieke type objecten afgeleid kunnen worden zoals ‘Komeet’, ‘Planeet’ en ‘Ster’. Hiernaast zijn er ook nog 2 klassen waarvan het 'Zonnestelsel' gebruik maakt in ‘Opslaan’ en ‘UImanager’.
Dit diagram toont duidelijk de structuur van de code en geeft aan de gebruiker een beeld van hoe er ‘achter de schermen’ te werk gegaan wordt.

Sequentiediagram: toevoegen()

In Figuur 2 staat het sequentiediagram van de methode “toevoegen” toont aan hoe de methode in zijn werk gaat om een planeet aan het zonnestelsel toe te voegen. Dit proces begint wanneer de gebruiker een planeet toevoegt aan het systeem. Vervolgens roept de klasse ‘Zonnestelsel’ de methode ‘botsingcheck’ aan om te verifiëren of de nieuwe planeet geen bestaande objecten in de ruimte zal raken. Dit resulteert in twee mogelijke scenario’s:

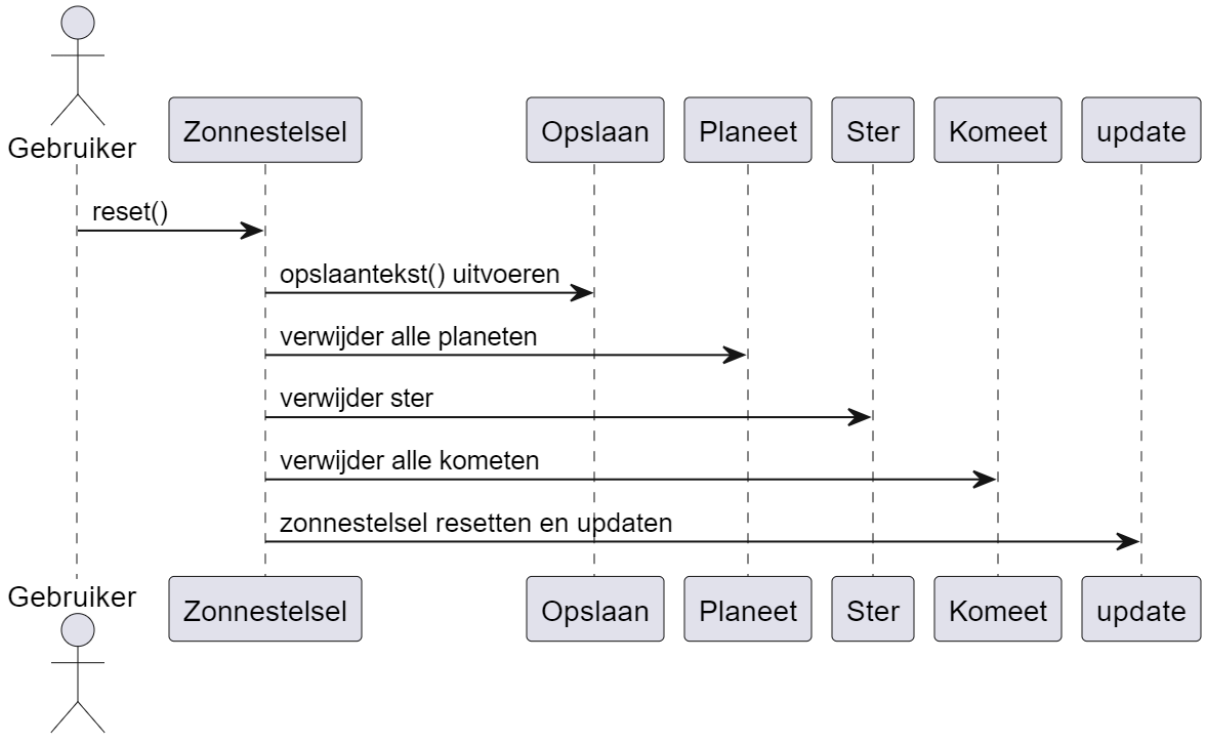
ofwel vindt er geen botsing plaats na het toevoegen van een nieuwe planeet en wordt de planeet gewoon toegevoegd aan de lijst van reeds bestaande planeten. Ofwel is er wel een botsing en dan wordt de methode ‘botsing’ aangeroepen om de gevolgen van de botsing af te handelen. Dit sequentiediagram toont duidelijk welke stappen genomen worden als een planeet wordt toegevoegd en welke verschillende situaties kunnen optreden (wel of geen botsing).



Figuur 2: sequentiediagram toevoegen planeet

Sequentiediagram: reset()

Het tweede sequentiediagram (Figuur 3) toont de uitvoering van de methode ‘reset’. Het doel van deze methode is om het zonnestelsel volledig te herinitialiseren. Na het oproepen van ‘reset’ wordt eerst de methode 'opslaan Tekst' opgeroepen zodat de huidige vorm van het zonnestelsel opgeslagen wordt. Vervolgens worden alle planeten, sterren en eventueel kometen uit het zonnestelsel verwijderd. Tot slot wordt “update” opgeroepen zodat het zonnestelsel vernieuwd wordt en wijzigingen kunnen doorgevoerd worden.
In tegenstelling tot 'toevoegen' zijn er in dit sequentiediagram geen verschillende mogelijke situaties. De nadruk ligt hier op de chronologische volgorde waarin methoden en acties worden uitgevoerd.



Figuur 3: sequentiediagram 'resetten' zonnestelsel

Toegankelijkheid

Kleurenblindheid

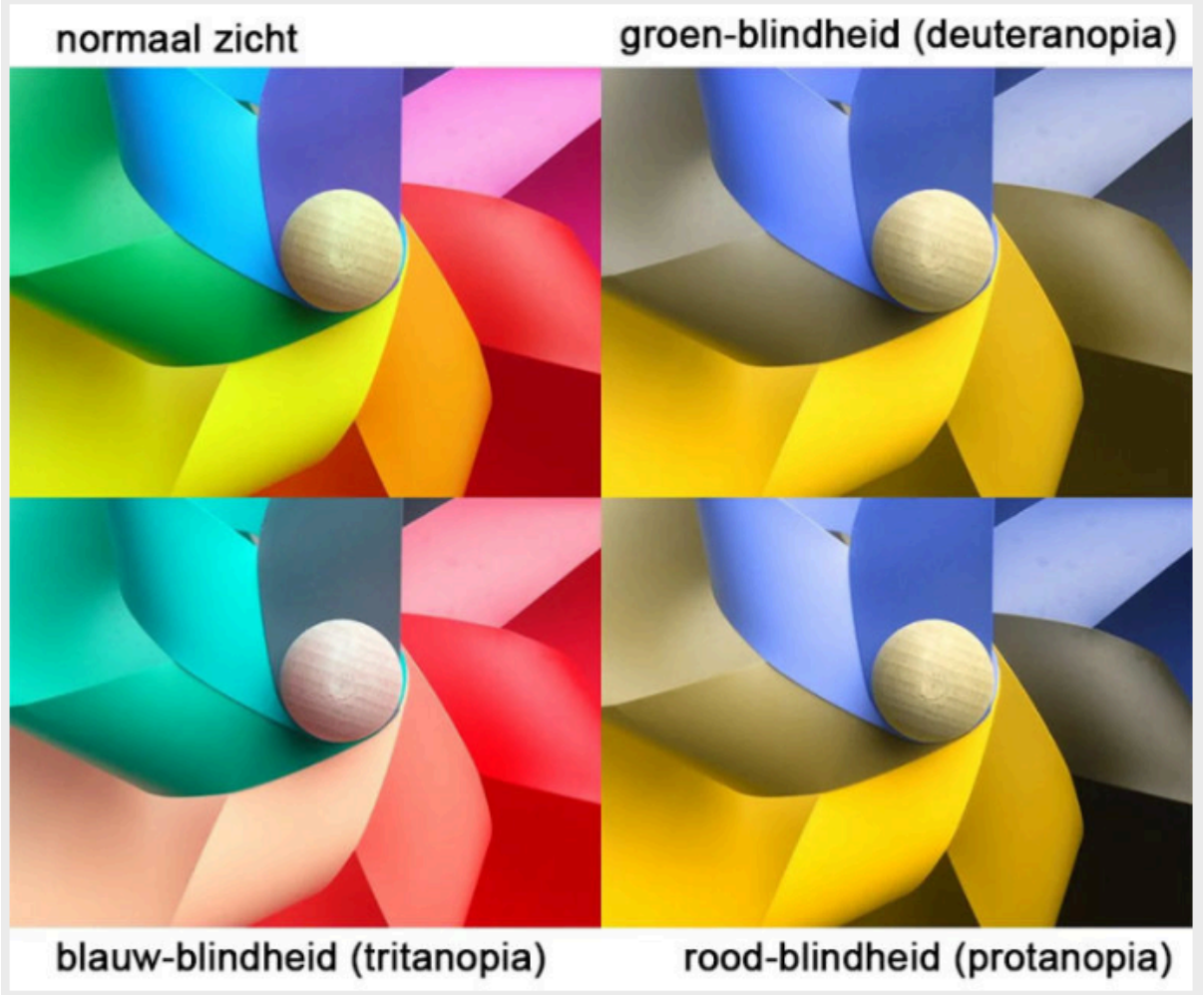
Om rekening te houden met accessibility wordt ervoor gekozen om kleurenblindheid toe te passen op onze applicatie. Kleurenblindheid zal opgedeeld worden in 3 (meest voorkomende) verschillende soorten: Protanopie (rood-groen blindheid), Deuteranopie (groen blindheid) en Tritanopie (blauw-geel blindheid). Deze verschillende soorten zijn te zien in *Figuur 1*. Dit zal vervolgens geïmplementeerd worden door middel van verschillende soorten knoppen waarbij elke knop één van deze 3 zal moeten voorstellen. Bij het klikken op één van deze knoppen zal dan heel de pagina moeten veranderen naar de geselecteerde kleurenblindheid.
Voor elk van deze 3 soorten zal een andere oplossing voorzien moeten zijn van hoe de pagina zal moeten veranderen. Een goede oplossing hierbij is om te spelen met contrasten en helderheid door gebruik te maken van CSS (filters) en dat voor elk van deze 3 een andere waarde wordt toegekend. Een andere oplossing die gebruikt kan worden is om bepaalde kleuren te verwisselen met een ander soort kleur. Dus bijvoorbeeld voor protanopie zullen mensen de kleur rood minder goed kunnen onderscheiden, dus zal dit opgelost kunnen worden door alle rode elementen in de pagina te vervangen door een andere kleur (bijvoorbeeld blauw of geel). Hierbij zal ook een reset-knop voorzien zijn om weer naar de standaardversie te gaan van de applicatie.

Indien de applicatie nog verder uitgebreid zal worden, dan kan er een LocalStorage toegevoegd worden die je laatst gekozen instelling opslaat en daarna zal weergeven als je de pagina opnieuw zal herladen. Vervolgens kunnen er ook meerdere soorten kleurenblindheden toegevoegd worden die minder voorkomen zoals Achromatopsie (volledige kleurenblindheid), zodat een breder publiek de applicatie zal kunnen gebruiken.

Keuze technologie

In deze applicatie is ervoor gekozen om HTML+CSS en Javascript te gebruiken. Dit werd gekozen omwille van de prestaties. Javascript laadt sneller dan Angular omdat er geen extra framework nodig is. Hierbij is Javascript makkelijker om te gebruiken en te begrijpen. Angular wordt meestal gebruikt bij veel grotere projecten met heel veel componenten, waarbij dat hier niet het geval is.

Het gebruik van een server zal dan ook waarschijnlijk niet toegepast worden op de applicatie aangezien dit niet echt nodig is. Daarbij heeft ook niet iedereen in het team de volledige kennis daarover. Alle functionaliteit zal dus lokaal verwerkt worden. Dit zal dus ook onnodige complexiteit vermijden. Met deze aanpak blijft de applicatie snel, eenvoudig en uitbreidbaar. Indien er tijd over is zal er dan pas een server gebruikt worden om profielen aan te kunnen maken.



Figuur 4: Illustratie soorten kleurenblindheden

Uitbreidingen

Profiel

De eerste uitbreiding die mogelijk is, is het aanmaken van een profiel. De gebruiker zal dan een profiel kunnen aanmaken waarop het zelfgemaakte zonnestelsel kan worden opgeslagen en gedeeld kan worden met andere profielen.

Inzoomen

Inzoomen op planeten mogelijk maken

Komeet

Een extra ruimte object toevoegen, dit zorgt voor extra mogelijkheden bij het bouwen van het zonnestelsel. Er zouden ook nog andere soorten objecten kunnen toegevoegd worden, maar daar zijn nog geen concrete plannen voor.

Verwijder-modus

Een modus waarin je planeten kan aanduiden, die daarna dan verwijderd worden. Zo moet er niet steeds opnieuw begonnen worden na het maken van een fout.

Internationalisering

Applicatie maken zodat hij werkt in meerdere verschillende talen.

Toegankelijkheid E-Readers

De webapplicatie toegankelijk maken zodat deze door gebruikers met E-Readers op een eenvoudige, conflictloze manier kan worden bezocht.

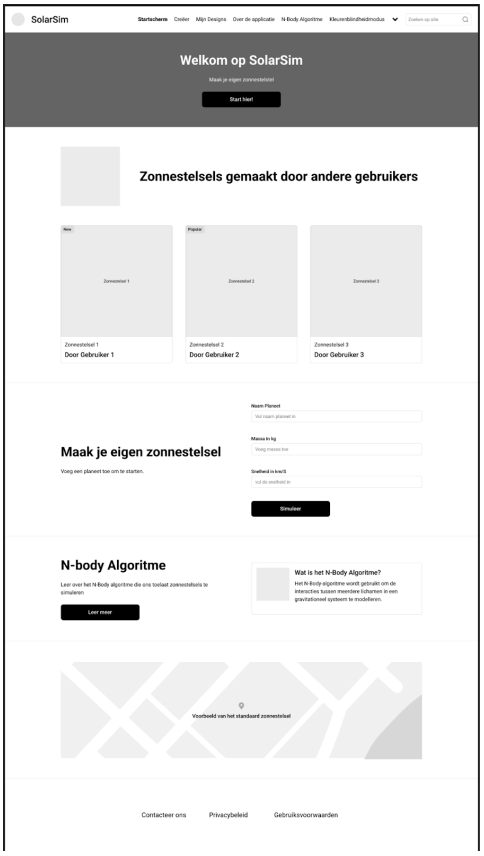
Schetsen van app functionaliteit

Hieronder wordt een voorbeeld geschetst van hoe de SolarSim webapplicatie er ongeveer zal uitzien. De webapplicatie wordt per deelpagina overlopen.

Startscherm

Bij het betreden van de webapplicatie komt de gebruiker op deze pagina uit. De pagina toont onmiddellijk enkele onderverdelingen:

- De header van de pagina met links naar de verschillende pagina's die de webapplicatie omvat alsook een dropdown menu om de webpagina toegankelijk te maken voor gebruikers die moeite hebben om bepaalde kleuren waar te nemen, dit werd al aangehaald in het hoofdstuk over toegankelijkheid. Helemaal links bevindt zich de naam en het logo van de webpagina en tot slot een zoekbalk rechts in de header.
- De meeste info bevindt zich in de body van de webapplicatie. Bovenaan in het startscherm wordt er meteen voorgesteld om de gebruiker een stelsel te laten simuleren. Daaronder bevinden zich enkele zonnestelsels die al door andere gebruikers werden gemaakt en ook door de gebruiker opnieuw kunnen gesimuleerd worden. Het derde deel bestaat opnieuw uit een uitnodiging om een nieuw zonnestelsel te creëren. Het laatste deel van de body bevat dan een link die uitleg geeft over het N-Body algoritme dat de webapplicatie gebruikt om de planeten en hemellichamen te beschrijven.
- Tot slot hebben we de footer, deze bevat slechts drie zaken: Een manier om de eigenaars van de applicatie te contacteren en dan nog het privacybeleid en de gebruiksvoorwaarden van de applicatie.

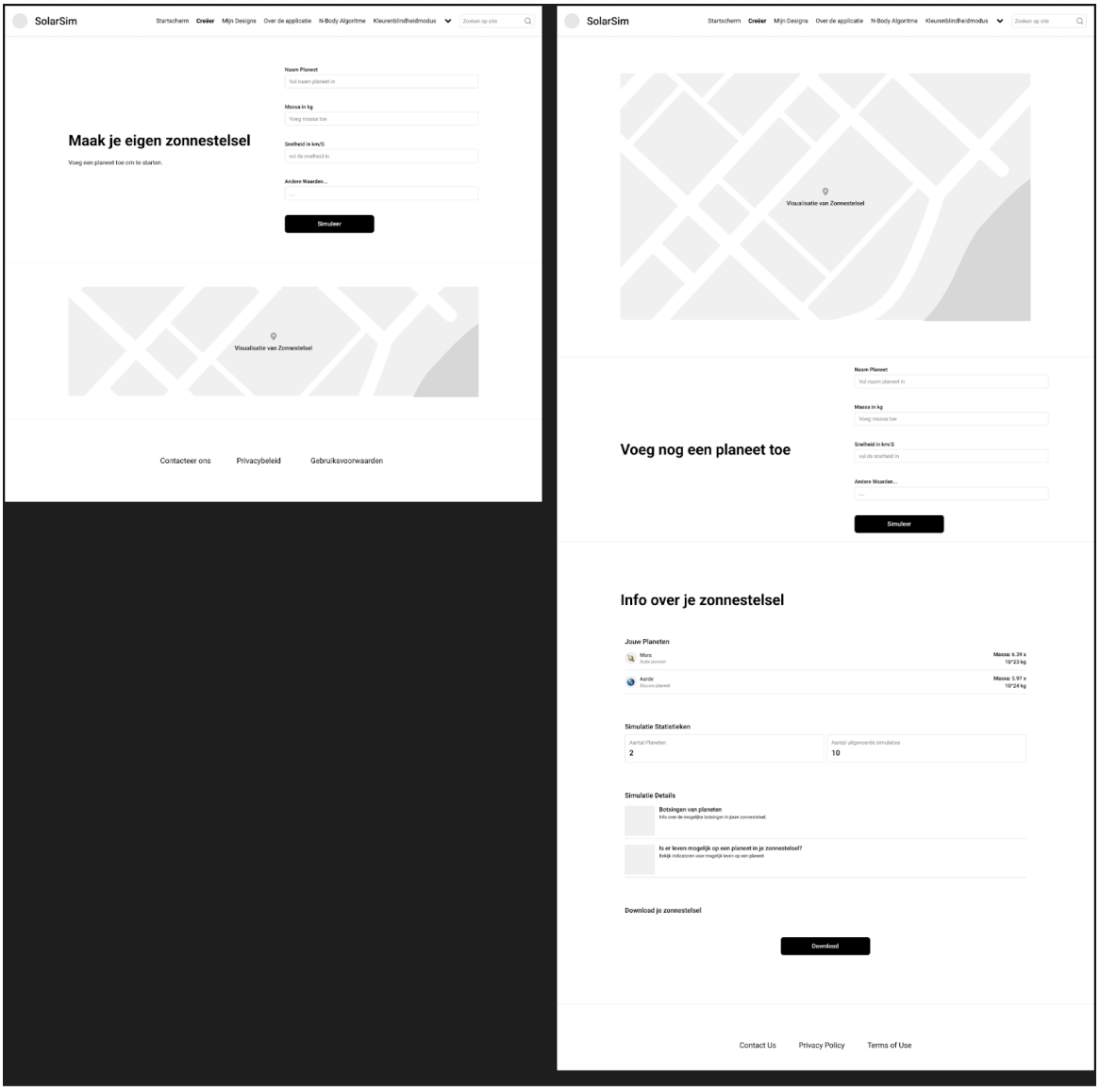


Figuur 5: Schets startscherm SolarSim webapplicatie

Creëer

Net zoals het startscherm bevat de Creëer-pagina dezelfde inhoud als het startscherm.
De inhoud van deze pagina is afhankelijk van of de gebruiker al gestart is met de creatie van het zonnestelsel. Indien de gebruiker dat nog niet gedaan heeft, bevat deze pagina enkel de mogelijkheid om een eerste planeet toe te voegen. Als de gebruiker een eerste planeet heeft toegevoegd krijgt, wordt er onmiddellijk een simulatie getoond aan de gebruiker zoals rechts te zien in de figuur. Daarnaast krijgt de gebruiker ook de mogelijkheid om nog planeten toe te voegen aan zijn stelsel. De gebruiker vindt daarnaast ook info over de planeten, statistieken en details over zijn volledige simulatie.

De gebruiker heeft daarnaast ook de mogelijkheid om zijn zonnestelsel te downloaden. Er wordt dan een .csv bestand aangemaakt die alle waarden bevat die nodig zijn om het zonnestelsel aan te maken. Bij een volgend bezoek aan de webapplicatie kan dit bestand dan upgeloadt worden om hetzelfde zonnestelsel terug te krijgen.

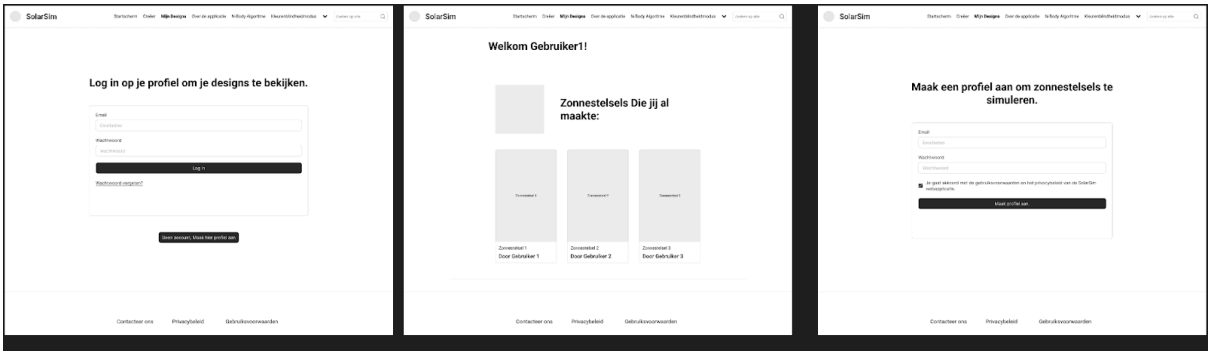


Figuur 6: Schets creëer pagina SolarSim webapplicatie

Mijn designs

Dit tabblad bevat de gemaakte designs van de gebruiker, zoals aangehaald in het hoofdstuk over de mogelijke uitbreidingen wordt aangehaald dat de gebruiker een profiel kan aanmaken. links in de figuur vinden we de inlogpagina, hier kan er ingelogd worden aan de hand van zijn e mailadres en wachtwoord, indien de gebruiker nog niet beschikt over een profiel, kan er via de knop onderaan de pagina een account aangemaakt worden.

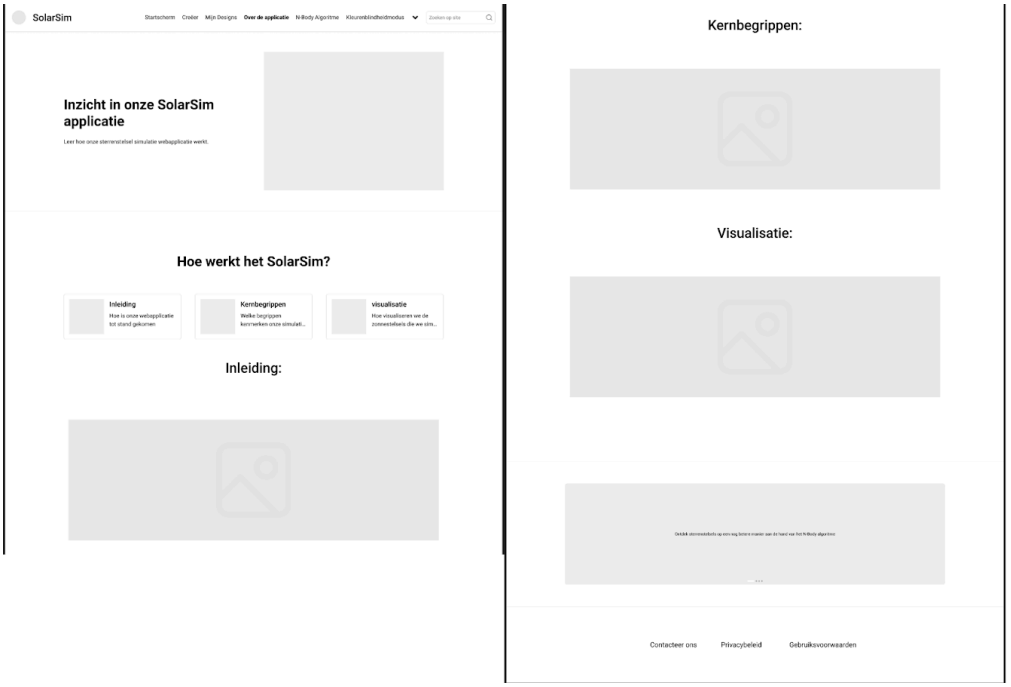
Als de gebruiker eenmaal ingelogd is, wordt de gebruiker begroet door middel van zijn gebruikersnaam en vindt hij ook zijn eerder gemaakte stelsels terug.



Figuur 7: Schets mijn designs pagina SolarSim webapplicatie

Over de applicatie

De pagina genaamd ‘over de applicatie’ bevat uitleg over hoe de SolarSim webapplicatie is opgebouwd. Het bevat over de gebruikte algoritmen en andere technologieën.

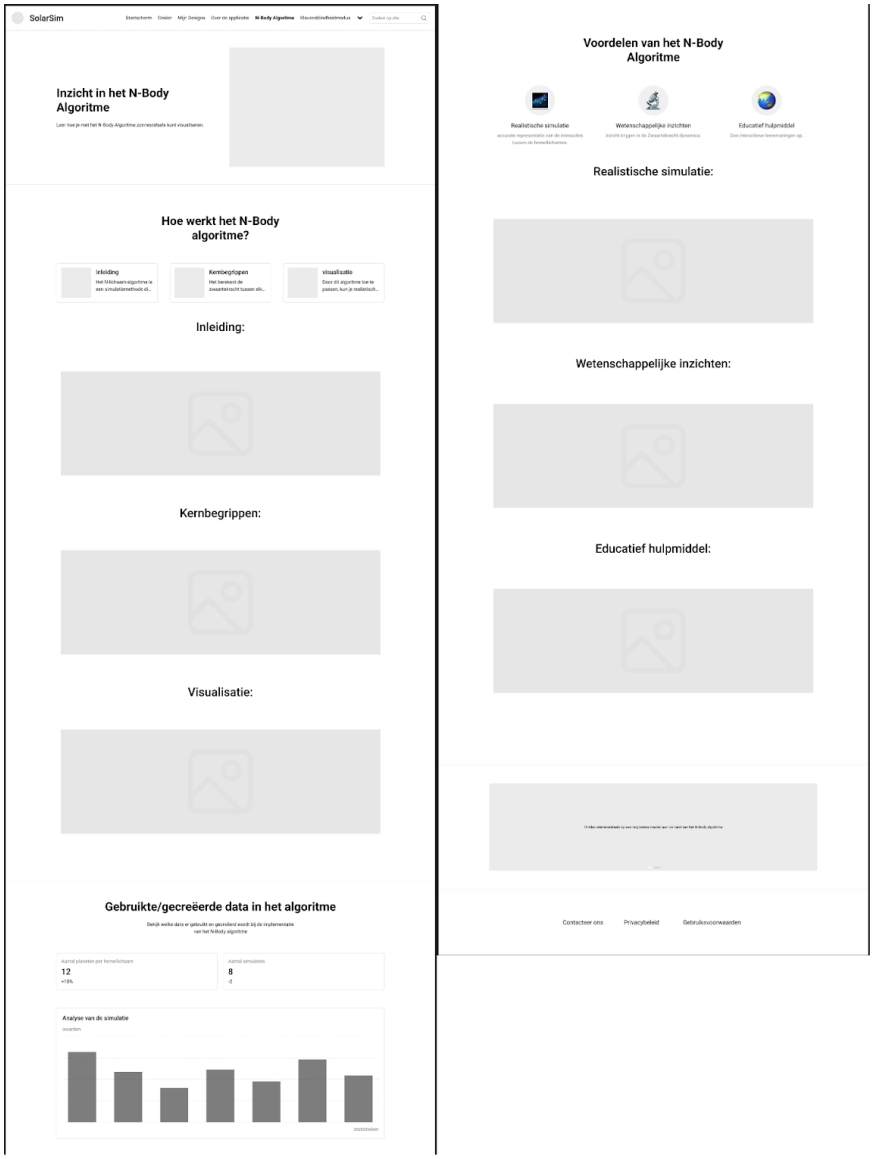


Figuur 8: Schets over de applicatie pagina SolarSim webapplicatie

N-body Algoritme

De pagina N-Body algoritme bevat zoals de naam van de pagina al verklapt info over het N-body algoritme theoretisch werkt en hoe dit op de webapplicatie is toegepast om realistische simulaties van zonnestelsels te verkrijgen.

Hoe het algoritme is toegepast wordt stuk voor stuk uitgelegd, daarnaast worden ook de nodige en gecreëerde data toegelicht en welke voordelen dit algoritme met zich meebrengt.



Figuur 9: Schets N-Body algoritme pagina SolarSim webapplicatie

Bijlagen

Userstories

NMBS vertragingen

Titel Bezoeker ziet kaart van België met trein netwerk/spoor
Als developer
Wil ik dat de bezoeker van de site een beeld krijgt van hoe de NMBS in elkaar zit en welke problemen ze hebben
Zodat ze hun reis in de toekomst beter kunnen plannen

Vereisten

- Complex algoritme ()
- Accessibility (bijvoorbeeld: kleurenblindmode)
- Heatmap van vertragingen moet ook voor kleurenblinden leesbaar zijn
- Inzoomen map moet mogelijk zijn om spoorwegen/connecties zo in detail te bekijken.
- Data over NMBS en treinen, vertragingen, ...

Scenario's

Scenario 1: standaard view map + vertraging
Stel dat er vertraging is
Als de gebruiker vertraging bekijkt op de map
Dan worden de vertragingen getoond als heatmap

Scenario 2: standaard view map + geen vertraging
Stel dat er geen vertraging is
Als de gebruiker vertraging bekijkt op de map
Dan wordt er een melding getoond met boodschap "momenteel geen vertraging" en blijft de map met connecties en verbindingen onveranderd.

Scenario 3: station bekijken
Stel dat er informatie is over station
Als de gebruiker informatie opvraagt over station
Dan worden statistieken weergegeven (bijvoorbeeld gemiddelde vertraging, gemiddeld trajectlengte, langste traject mogelijk…), ook worden eventuele routes weergegeven

Scenario 4: station bekijken zonder informatie
Stel dat er geen informatie is over station
Als de gebruiker informatie opvraagt over station
Dan verschijnt een melding: “Sorry, er is geen informatie beschikbaar voor dit station.**”** En wordt de gebruiker doorverwezen naar de NMBS om daar eventueel informatie op te zoeken over het station.

Scenario 5: route opvragen
Stel dat er een route mogelijk is zonder overstappen
Als de gebruiker een begin- en eindstation opgeeft
Dan wordt er visueel voorgesteld op de map hoe de route zal lopen, hoeveel het zal kosten en om welke uren er een trein is en wanneer je waar zal toekomen

Scenario 6: route opvragen
Stel dat er geen route mogelijk is zonder overstappen
Als de gebruiker een begin- en eindstation opgeeft
Dan wordt er visueel voorgesteld op de kaart hoe de route zal lopen, hoeveel het zal kosten, om welke uren er een trein is en wanneer je waar zal toekomen en ook hoeveel tijd je hebt om over te stappen en op welke uren het efficiëntst is om je route te maken

Zonnestelsel

Titel planeten toevoegen
Als developer
Wil ik dat bezoekers planeten kunnen toevoegen aan het zonnestelsel en bestaande zonnestelsels kunnen inladen
Zodat een (on)bestaand zonnestelsel gesimuleerd kan worden en er geleerd kan worden over de werking ervan (on)gekende zonnestelsels

Vereisten

- Complex algoritme (n-body problem)
- Accessibility (voorbeelden: dyslexie, kleurenblindheid of autisme)
- Verschillende soorten planeten
- Botsingen moeten kunnen gebeuren
- Snelheid is afhankelijk afstand met de zon
- Bestaande zonnestelsels kunnen inladen
- Duidelijke en eenvoudige visualisatie van de gecreëerde zonnestelsels.

Scenario’s

Scenario 1: standaard view map bestaand zonnestelsel
Stel dat zonnestelsel bestaat
Als bezoeker bestaand zonnestelsel inlaadt
Dan wordt het bestaande zonnestelsel ingeladen met accurate gegevens. De bezoeker kan extra informatie vinden op de website, als hij op een planeet klikt krijgt hij extra informatie over de planeet.

Scenario 2: creatie zonnestelsel
Stel dat planeet toevoegen geen botsingen veroorzaakt
Als de gebruiker planeet, komeet … toevoegt
Dan wordt deze ingeladen met ofwel standaardwaarden bepaalt door programma ofwel wordt het ingeladen met waarden meegegeven door gebruiker

Scenario 3: creatie zonnestelsel
Stel dat planeet toevoegen botsingen veroorzaakt
Als de gebruiker planeet, komeet … toevoegt die op dezelfde baan ligt als iets anders
Dan zal bij botsing een van de 2 objecten verwijderd worden (zal duidelijk gemaakt worden aan de hand van een animatie) de andere zal afhankelijk van het type botsingen veranderen van type of niet (zo kan een planeet bijvoorbeeld veranderen van vorm door inslag van een kleine komeet of kan de temperatuur aanpassen na verloop van tijd)

Scenario 4: planeet met slim leven
Stel dat planeet slim leven heeft
Als de gebruiker klikt op deze planeet
Dan wordt het bevolkingsaantal weergegeven, bij botsing met een andere planeet kan dit krimpen. Dus de parameters van zo'n 'slimme' planeet kunnen veranderen naargelang de omstandigheden in het gesimuleerde zonnestelsel veranderen

Scenario 5: planeet onleefbaar
Stel dat planeet geen leven heeft
Als de gebruiker klikt op deze planeet
Dan wordt er weergegeven dat er geen leven kan zijn op deze planeet (kan bijvoorbeeld wel veranderen na een botsing met andere planeet)

Testen

Uitgewerkt

```
function CheckLife(temperatuur, typePlaneet) {
    const minTemp : number  = -50;
    const maxTemp : number  = 50;
    const typeAllowed : string[]  = ["Terrestrisch", "Rotsachtig","Water"];

    return temperatuur >= minTemp && temperatuur <= maxTemp && typeAllowed.includes(typePlaneet);
}

module.exports = CheckLife;

const CheckLife = require("./CheckLife");

test("Leven mogelijk op een aardse planeet binnen de juiste temperatuur", () : void  => {
    expect(CheckLife(20, "Terrestrisch")).toBe(true);
});

test("Te koude temperatuur maakt leven onmogelijk", () : void  => {
    expect(CheckLife(-100, "Terrestrisch")).toBe(false);
});

test("Te warme temperatuur maakt leven onmogelijk", () : void  => {
    expect(CheckLife(100, "Terrestrisch")).toBe(false);
});

test("Leven onmogelijk op een gasplaneet", () : void  => {
    expect(CheckLife(20, "Gasreus")).toBe(false);
});

test("Leven mogelijk op een rotsachtige planeet bij lage temperatuur", () : void  => {
    expect(CheckLife(-50, "Rotsachtig")).toBe(true);
});

test("Leven onmogelijk op een verkeerd type planeet", () : void  => {
    expect(CheckLife(20, "IJs")).toBe(false);
});
```

```
function checkAantalPlaneten(aantalPlaneten, maxAantal : number  = 20) {
    return aantalPlaneten >= 0 && aantalPlaneten <= maxAantal;
}

module.exports = checkAantalPlaneten;

const checkAantalPlaneten = require("./checkAantalPlaneten");

test("Aantal planeten klopt binnen de limiet", () : void  => {
    expect(checkAantalPlaneten(5)).toBe(true);
});

test("Precies het maximum aantal planeten is toegestaan", () : void  => {
    expect(checkAantalPlaneten(20)).toBe(true);
});

test("Meer dan het maximum aantal planeten is niet toegestaan", () : void  => {
    expect(checkAantalPlaneten(25)).toBe(false);
});

test("Een negatief aantal planeten is niet toegestaan", () : void  => {
    expect(checkAantalPlaneten(-1)).toBe(false);
});

test("Geen planeten (0) is toegestaan", () : void  => {
    expect(checkAantalPlaneten(0)).toBe(true);
});
```

Niet uitgewerkt

Volgende testen kunnen ook nog geïmplementeerd worden:

1. Is er een botsing?
2. Wordt de afbeelding correct weergegeven? (Is de afbeelding beschikbaar)
3. Kan het stelsel juist opgeslagen worden?
4. Verandert de toestand van de planeet op de juiste manier?
5. Wordt de info over planeten correct weergegeven (met andere woorden, kan de info worden ingeladen?)

Bronnen

Wikipedia contributors. (2024, 16 december). *Barnes–Hut simulation*. Wikipedia. https://en.wikipedia.org/wiki/Barnes%E2%80%93Hut_simulation

National Eye Institute. (z.d.). *Types of Color Vision Deficiency | National Eye Institute*. https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-vision-deficiency

Wikipedia contributors. (2025, 28 februari). *N-body problem*. https://en.wikipedia.org/wiki/N-body_problem#History

NMBS. (z.d.). Public data – mobility service providers. Belgiantrain. Geraadpleegd op 4 maart 2025, van https://www.belgiantrain.be/nl/3rd-party-services/mobility-service-providers/public-data

API Console — Infrabel - Open Data. (z.d.). Geraadpleegd op 4 maart 2025 https://opendata.infrabel.be/api/v1/console/datasets/1.0/search/

*© 2024Flash3000 Productionswebdesign & grafische vormgeving
Fuutstraat 23
2421XR Nieuwkoop
Tel: 0172 534564. (2024, April 26). Hoe zien kleurenblinden een website? © 2024Flash3000 Productionswebdesign & Grafische Vormgeving*

Fuutstraat 23

2421XR Nieuwkoop

Tel: 0172 534564.

https://flash3000.nl/blog/hoe-zien-kleurenblinden-een-website/