StellarMap

Figure 1 : Système solaire (freepik, -)

Table des matières

2 Analyse préliminaire

2.1 Introduction

Ce projet est réalisé dans le cadre du au travail pratique individuel (TPI) qui s'effectue lors de la dernière année de CFC en informatique.

Ce travail s'effectue sur une période de réalisation de 90 heures, entre le 2 mai de 8h50 au 2 juin à 15h20.

Le sujet est une carte 3D interactive du système solaire, il a été choisi à la suite de la proposition de ce sujet par le candidat.

2.2 Objectifs

L'objectif de ce projet est de crée une carte interactive du système solaire sur lequel il sera possible de voir les 8 planètes et leurs lunes ainsi que le soleil. Il sera possible de tourner autour du soleil et d'observer les planètes sous un autre angle. Une description des planètes devra s'afficher lorsqu'un utilisateur clique sur celui-ci, de plus il sera possible d'accélérer la vitesse de déplacement des planètes.

Sept objectifs spécifiques sont à atteindre :

1. La carte s'affiche avec toutes les huit planètes.
2. L'utilisateur peut naviguer dans le système solaire.
3. Ergonomie et facilité d'utilisation du produit (Bastien et Scapin).
4. Les informations des différentes planètes s'affichent quand on clique dessus.
5. Le site est « responsive » et peut être utilisé depuis un smartphone ou une tablette.
6. L'utilisateur peut modifier la vitesse de déplacement des planètes.
7. Les angles de vue du système peuvent être déterminé par l'utilisateur.

Tout au long de mon travail je vais me conformer aux critères d'évaluation établis par le canton de Vaud (Schwab, 2018).

Cahier des charges

2.3 Planification initiale

La planification initiale se découpe en cinq sprints découpés sur cinq semaines.
Le détail de la planification est disponible à la section 9.1.

Ci-dessous vous trouverez la planification initiale sous forme de méthodologie Waterfall (Wikipedia, 22), j'ai choisi ce format car c'est cette forme convient le mieux pour la planification de projet demandée. Cependant le travail se déroulera en format agile comme le montre la deuxième image, l'analyse sera effectuée au fur et à mesure du projet.

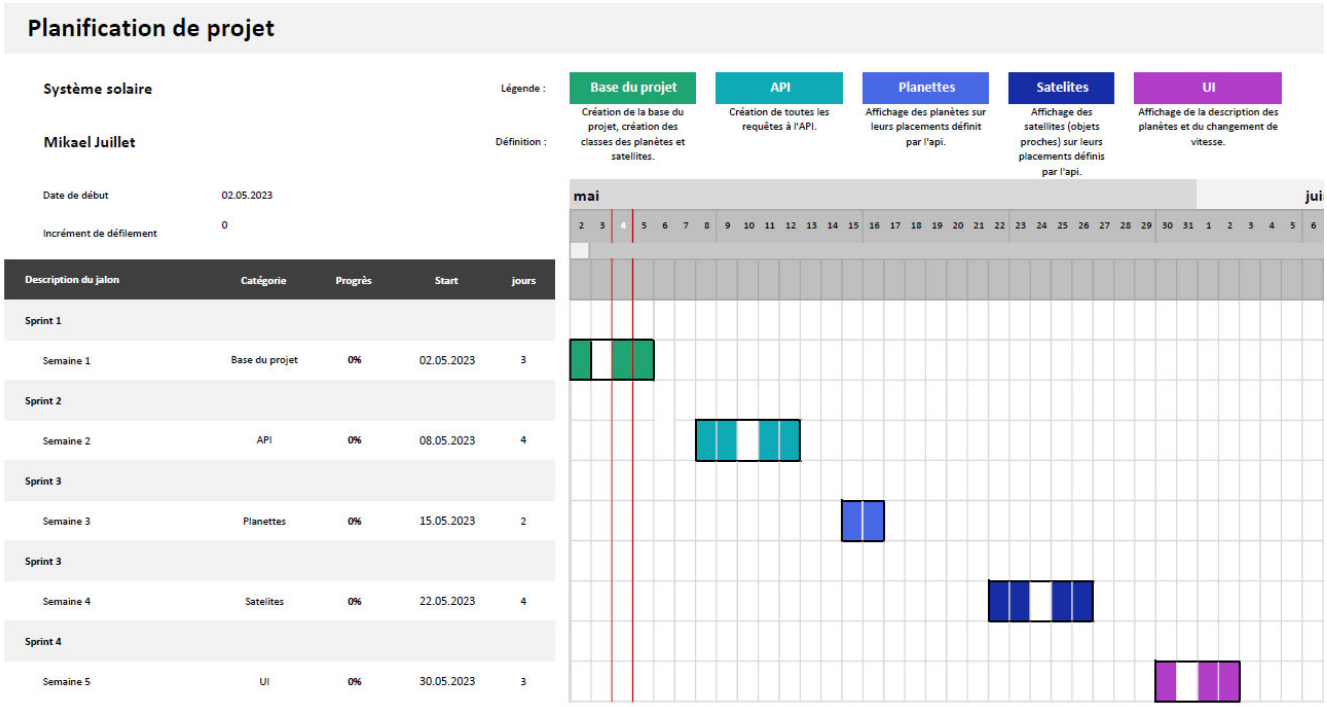
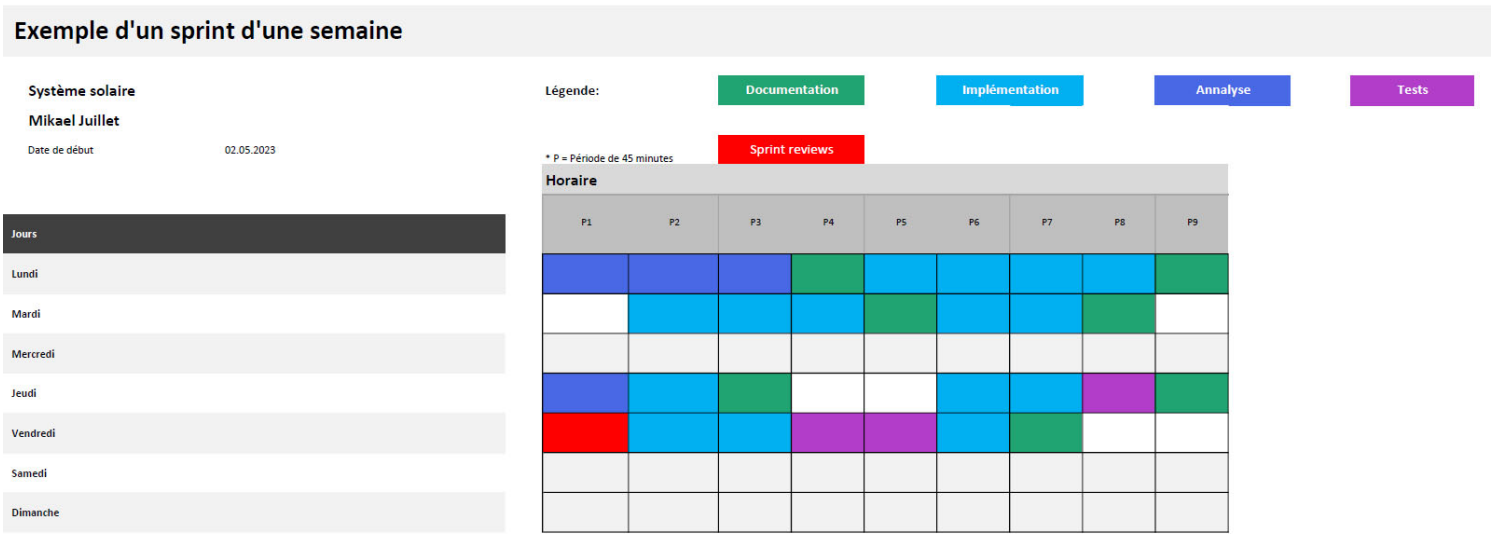


Figure 2 : Planification initiale du projet



3 Analyse / Conception

3.1 Concept

Ce site est conçu pour afficher un système solaire en 3D, le visiteur pourra tourner autour de la carte et ainsi voir les planètes sous d'autres angles, il aura aussi la possibilité d'accélérer le temps afin de voir le déplacement des planètes à des vitesses différentes.

Ce système se verra affiché les 8 planètes du système solaire, leurs lunes ainsi que les astéroïdes à proximité de la planète terre.

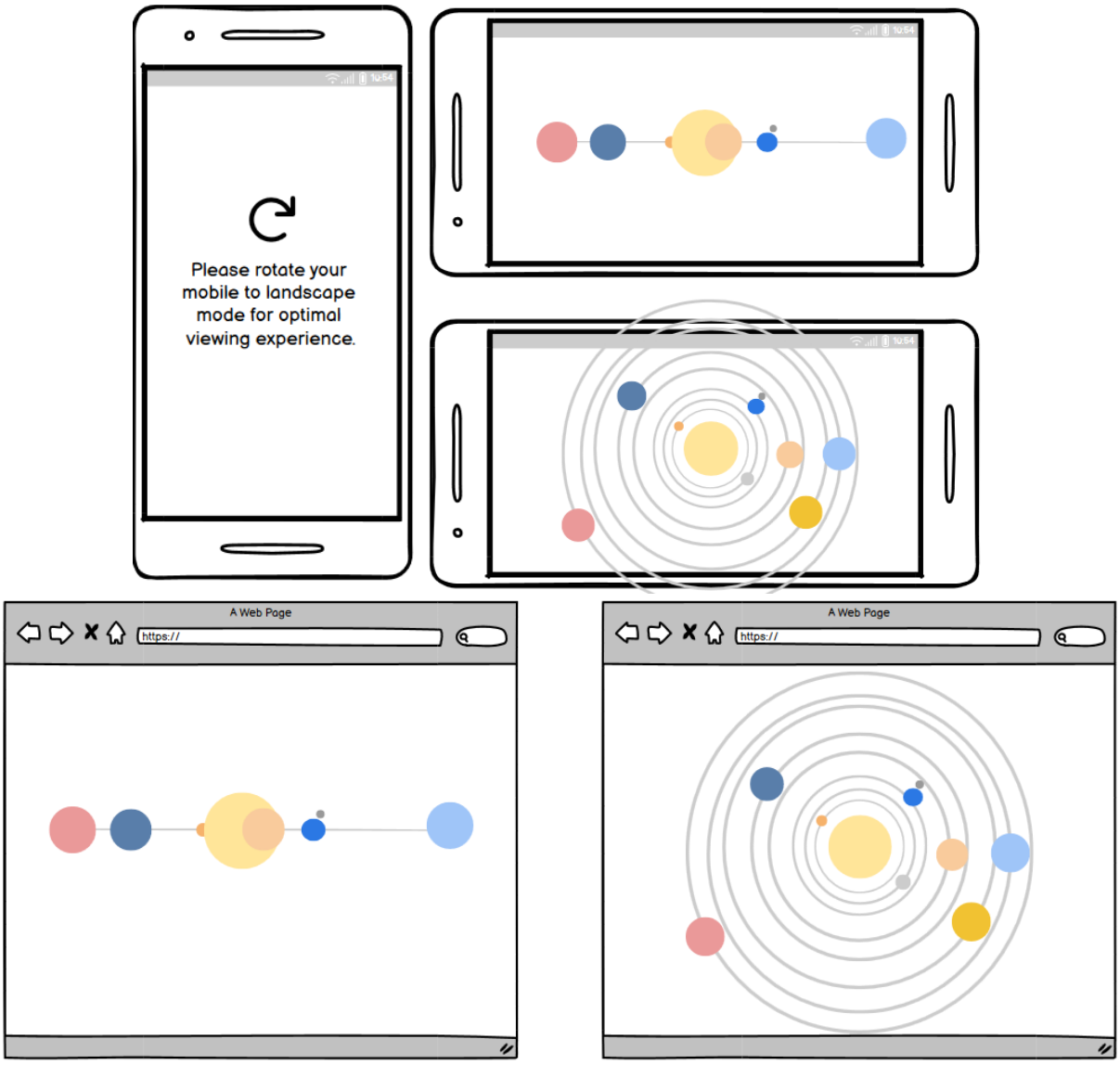
Le projet sera hébergé sur swisscenter, les liens des accès au projet sont les suivantes :

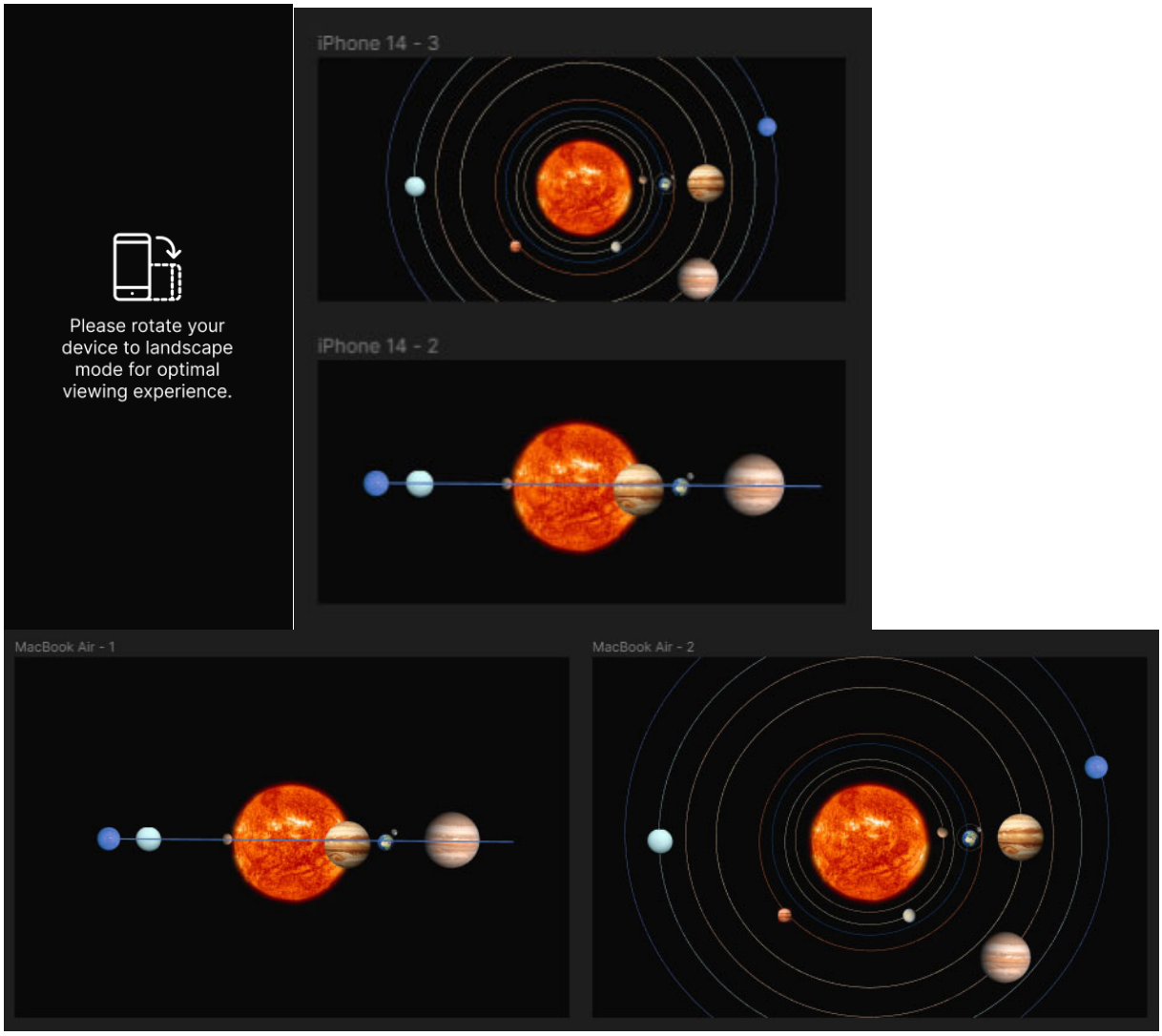
Code source : https://github.com/Juillet-Mikael/TPI
Planification du projet : https://icescrum.cpnv.ch/p/TPIJUILLET/#/project
Documentation :
https://github.com/Juillet-Mikael/TPI/blob/main/documents/documentation.docx
Journal de travail :
https://github.com/Juillet-Mikael/TPI/blob/main/documents/journaux.xlsm

3.1.1.1 Requêtes

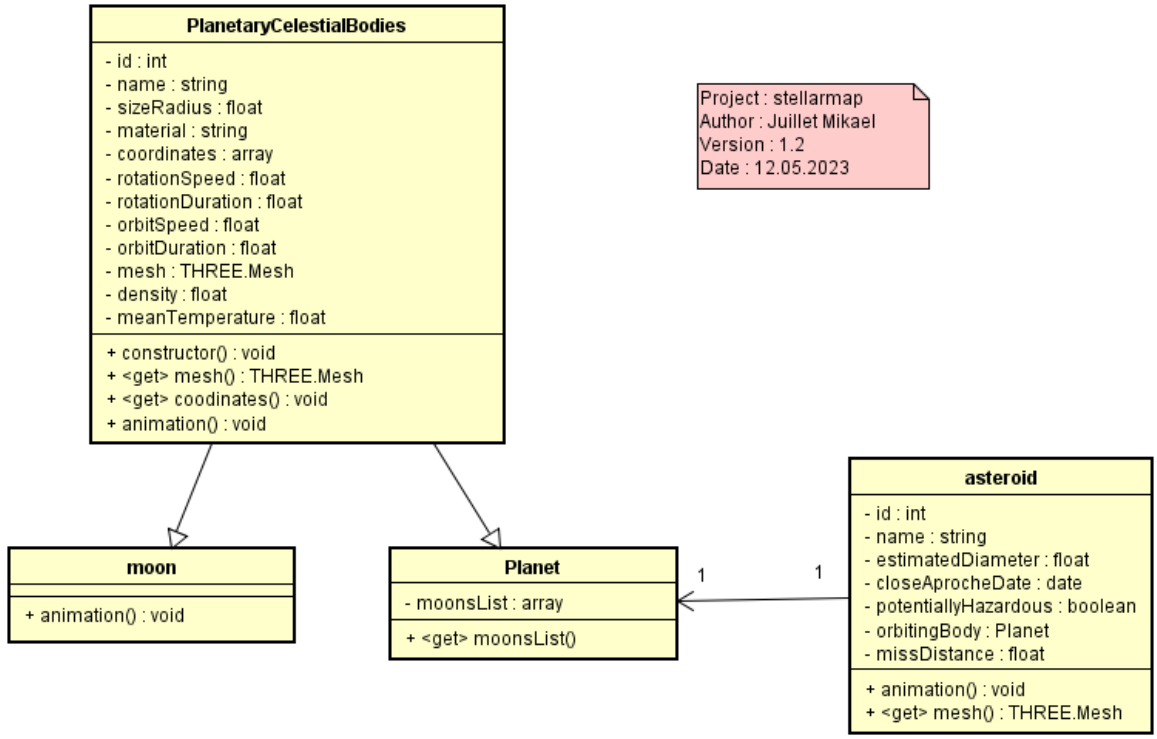
Il est prévu d'utiliser deux API de la Nasa, l'api « horizon view » permet de récupérer des informations précises sur les objets spatiaux dans notre système solaire. Horizon sera utilisé pour récupérer toutes les informations nécessaires au placement, et à la définition des planètes comme le volume, la densité, la position précise actuel, la température etc.
La deuxième API est « Near Earth Object » qui permet de récupérer la liste des objets proche de la terre à un temps donnée, elle sera utilisée pour placer approximativement les astéroïdes sur la carte car aucune donnée de placement précise ne peut être récupérer via cette api.

3.1.1.2 Maquettes




Please rotate your device to landscape mode for optimal viewing experience.
iPhone 14 - 3
iPhone 14 - 2
MacBook Air - 1
MacBook Air - 2

3.1.1.3 Diagramme de classe



3.1.1.4 Diagramme de séquences

3.2 Stratégie de test

Je vais essentiellement être effectuer manuellement pour la partie visuelle, j'ai choisi de faire cela car je ne dispose que de peu de temps pour crée mon projet de plus j'ai une très faible connaissance de la création de tests en Javascript.
Une exception sera faite pour ce qui est de la partie API en semaine 2, j'effectuerais des tests unitaires sur le filtrage des donnée et la réception des données.
J'effectuerai à chaque fin de sprint une série de tests dès l'implémentation de l'api en semaine numéro 2.

Les tests en rapport à la vue comme pour le déplacement des planètes seront des tests fonctionnels. Ces tests ont pour objectif de s'affurer du bon déroulement de l'affichage du système solaire.

3.3 Risques techniques

Les risques techniques sont mon manque de connaissance à l'utilisation de Three.js (three.js, 2023) et Ajax (W3schools, -), malgré de solides base acquises grâce à la préparation au TPI, j'ai tout encore besoin de beaucoup me référer aux documentations.

3.4 Planification

3.5 Dossier de conception

3.5.1 Logiciels / Framework utilisé :

Nom	Version	Utilisation
Visual Studio Code (visualstudio, 2023)	1.74.3	Editeur de code
Balsamiq Wireframe (balsamiq, 2023)	4.6.5	Wireframe
Figma (figma, 2023)	-	Mockup
HTML, CSS	Html 5, CSS 3	Mise en page
Vite.js (vitejs, 2023)	4.1.1	Frontend Tooling
Three.js (WebGL) (three.js, 2023)	0.152.2	Rendu 3D
Ajax (W3schools, -)	-	Requêtes
Moment (momentjs, 2023)	2.29.4	Gestion des dates
Vitest (vitest, 2021)	0.31.0	Tests

4 Réalisation

4.1 Dossier de réalisation

4.1.1 Requêtes API

Pour ce projet, il est nécessaire d'avoir deux API de la Nasa, la première s'appelle Horizon et sert à la récupération de données précises sur les objets céleste de notre système solaire. La deuxième API est Near Earth Objets et répertorie tous astéroides proches de la terre.

L'ensemble de mes requêtes s'effectuent sous forme de get et en https, Il est important d'utilisé le https car elle garantit l'intégrité des données envoyé en les chiffrant, il garantit l'identité du server et renforce la confiance des utilisateurs sur le site.

Il est possible de retrouver les liens des documentations des différentes API ci-dessous :

- **Horizon** : https://ssd-api.jpl.nasa.gov/doc/horizons.html
- **Near Earth Objects** (section : Asteroids - NeoWs) : https://api.nasa.gov/

L'api Near Earth Objet a besoin d'une clef API, c'est pourquoi j'ai choisi de crée un fichier .env qui contient ma clef API ainsi que les URL respectives des API.

```
VITE_API_URL_HORIZON=https://ssd.jpl.nasa.gov/api/horizons.api
VITE_API_URL_NEO=https://api.nasa.gov/neo/rest/v1/feed
VITE_API_KEY=APIKEY
```

Figure 3 : Contenu du fichier .env

Pour l'ensemble des mes API je forme mon Url via une fonction formatURL qui vas me permettre d'assembler mon url avec les paramètres demandé par l'api.

```
function formatURL(url, parameters) {

    var fullURL = url + "?";

    Object.keys(parameters).forEach( (key, index) => {
        fullURL = fullURL + key + "=" + parameters[key]

        if (index !== Object.keys(parameters).length - 1) {
            fullURL = fullURL + "&";
        }

    });

    return fullURL;
}
```

Figure 4 : Fonction de formating des URL de requêtes.

4.1.1.1 Horizon system

Url de requête : https://ssd.jpl.nasa.gov/api/horizons.api

Certains paramètres sont absolument obligatoires en voici la liste (Nasa, 2022) :

- **Format**, sera retourné en Json.
- **Command**, représente l'id de l'objet ciblé (ex. terre = 399).
- **Objet_data**, représente les données de l'objet tel que la période de rotation.
- **Make_Ephem**, représente les données de placement des planètes.
- **Start_time**, la date de départ des données en format années-mois-jour.
- **End_time**, la date de fin des données en format années-mois-jour.
- **Step_size**, la durée séparant les informations de placement de l'objet ciblé.

Lors ce que la requête est effectuée depuis le navigateur elle engendre une erreur corse. Pour résoudre ce problème, j'ai ajouté dans le header un Origin mais l'erreur ne se résout pas. (Braiam, 2017) (Simplified, 2021)
Je ne parviens pas à récupérer les données pourtant le lien est valide et retourne des données.

Erreur Corse :

> No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the request's mode to 'no-cors' to fetch the resource with CORS disabled.

Il est cependant possible de récupérer les données via Postman ce qui me fessait dire que l'api devait être ouverte.

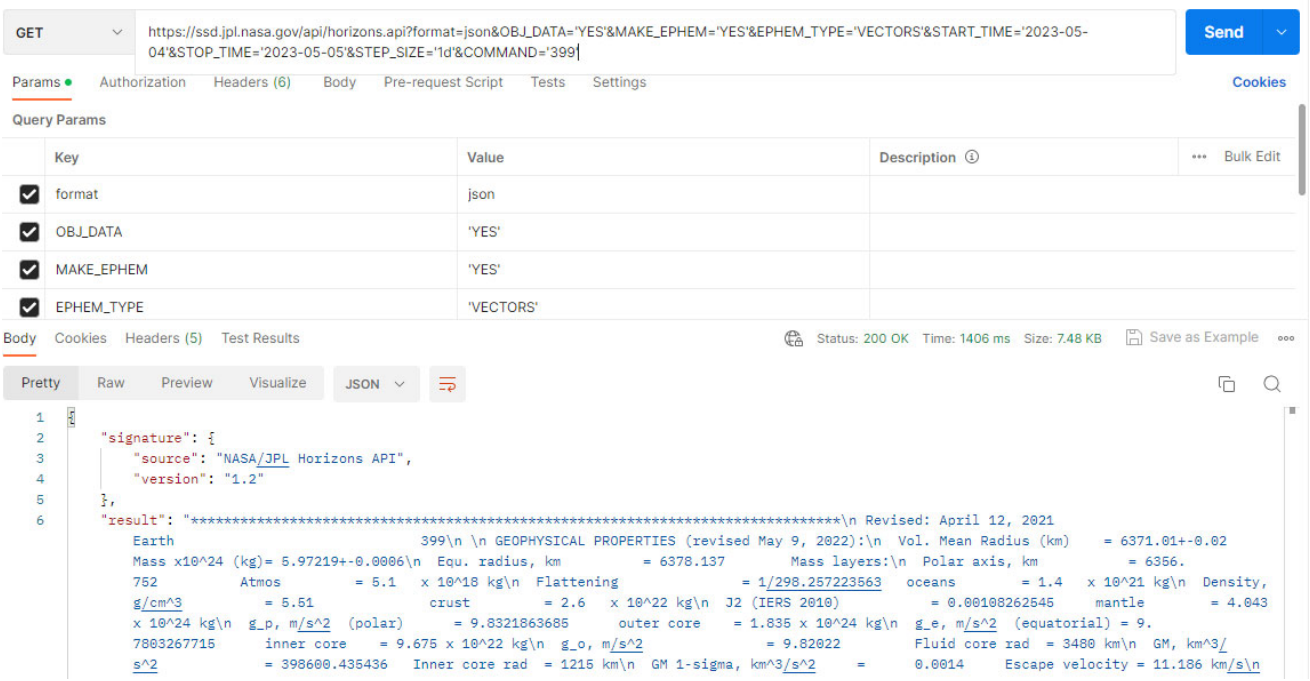


Figure 5: Données retournées via Postman.

Mais malgré un retour de requête avec un code d'état de 200 je n'avais toujours aucune donnée.

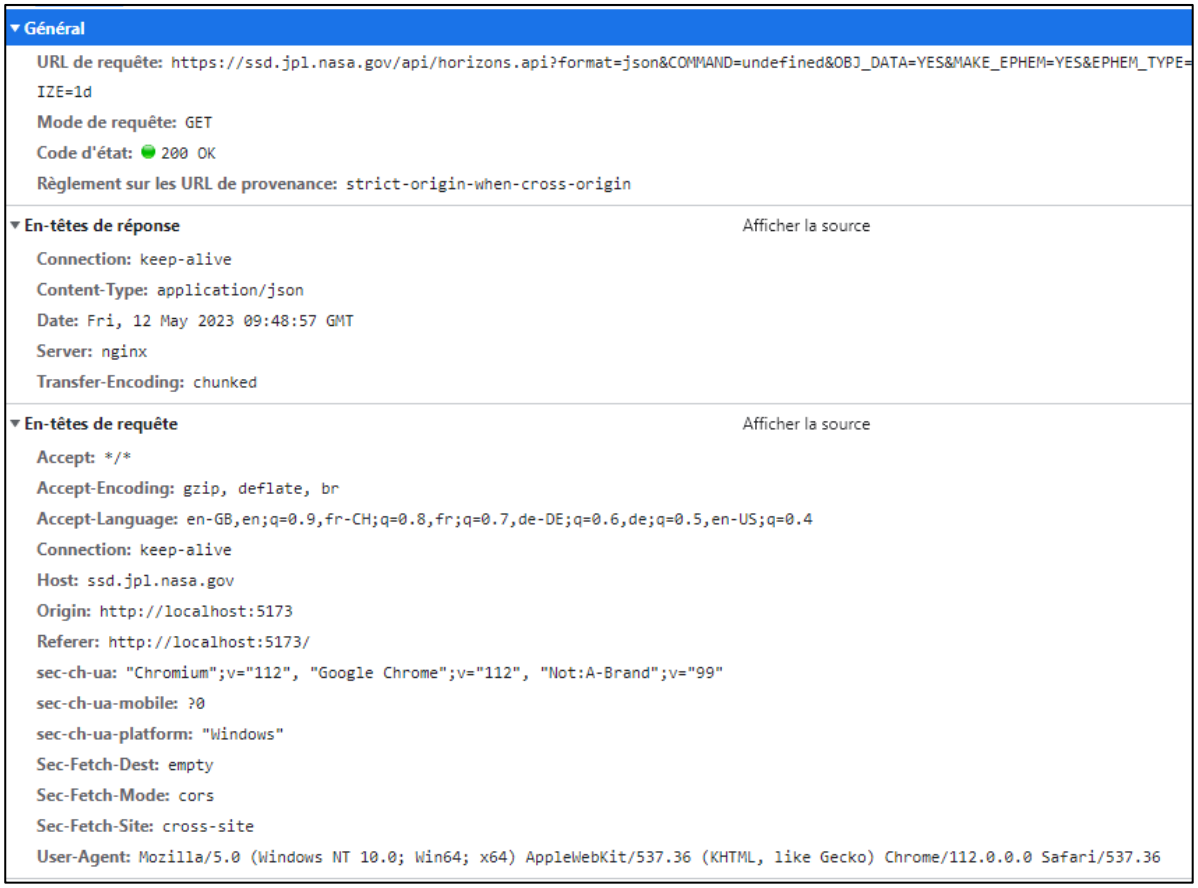


Figure 6 : Etat de la requête Horizon API sans données.

Figure 7 : Erreur d'affichage des données reçues.

Après en avoir discuté avec mon chef de projet Monsieur Benzonana, nous avons décidé de l'envoi d'un e-mail au support de la Nasa et de l'exportation les données dans un fichier json. Ainsi le moins de temps possible n'est perdu. J'ai créé une issue sur GitHub afin de maintenir le projet a jour.

La réponse de la Nasa a été expéditive (traduction du document original) :
« Les API de la NASA ne peuvent être intégrées à aucun site Web non-NASA. Donc, je crains que vous n'ayez besoin de trouver une autre méthode. C'est la politique de la NASA que nous sommes tenus de suivre. Vous pourrez peut-être appeler l'API à partir d'un script en dehors de votre serveur Web, puis déterminer comment connecter votre application Web à ce script. »

Suite à ce message nous avons gardé le document json pour la réalisation de ce projet.

4.1.1.1.1 Récupération des données

Les données reçues sont sous forme de String, je dois récupérer des valeurs précise dans ces tableaux, c'est pourquoi j'utilise des expression rationnelles (regex). Celle-ci vont me permettre de récupérer des valeurs en fonction d'une expression régulière dans mon fichier. (mozilla, 2023)

Voici un exemple des données reçues après suppression des espaces, les espaces ont été enlevé car ceux-ci posent des problèmes de correspondance des regex. Pour être plus précis les regex se fient à une expression régulière et prend donc en compte les espace pour éviter des problèmes de compatibilité entre deux valeurs, il est important d'enlever le maximum de valeurs pouvant interférer.


```
**********Revised:April12,2021Mercury199 dataFilter.js?t=1683886315807:19
PHYSICALDATA(updated2021-Apr-
12):Vol.MeanRadius(km)=2440+-1Density(gcm^-3)=5.427Massx10^23(kg)=3.302Volume(x10^10km^3)=6.085Siderealrot.period=58.6463dSid.rot.rate(rad/s)=0
.00000124001Meansolarday=175.9421dCoreradius(km)=~1600GeometricAlbedo=0.106Surfaceemissivity=0.77+-0.06GM(km^3/s^2)=22031.86855Equatorialradius
,Re=2440kmGM1-sigma(km^3/s^2)=Massratio(Sun/plnt)=6023682Mom.ofInertia=0.33Equ.gravitym/s^2=3.701Atmos.pressure(bar)=
<5x10^-15Max.angulardiam.=11.0"MeanTemperature(K)=440Visualmag.V(1,0)=-0.42Obliquitytoorbit[1]=2.11'+/-0.1'Hill'ssphererad.Rp=94.4Siderealorb.p
er.=0.2408467yMeanOrbitvel.km/s=47.362Siderealorb.per.=87.969257dEscapevel.km/s=4.435PerihelionAphelionMeanSolarConstant(W/m^2)1446262789126Max
imumPlanetaryIR(W/m^2)1270055008000MinimumPlanetaryIR(W/m^2)666**********Ephemeris/API_USERMonMay807:4
@:322023Pasadena,USA/Horizons**********Targetbodyname:Mercury(199)
{source:DE441}Centerbodyname:Earth(399){source:DE441}Center-
sitename:BODYCENTER**********Starttime:A.D.2023-May-
0400:00:00.0000TDBStoptime:A.D.2023-May-0500:00:00.0000TDBStep-
size:1440minutes**********Centergeodetic:0.0,0.0,0.0{E-
lon(deg),Lat(deg),Alt(km)}Centercylindric:0.0,0.0,0.0{E-
lon(deg),Dxy(km),Dz(km)}Centerradii:6378.137,6378.137,6356.752km{Equator_a,b,pole_c}Outputunits:KM-
SCalendarmode:MixedJulian/GregorianOutputtype:GEOMETRICcartesianstatesOutputformat:3(position,velocity,LT,range,range-
rate)EOPfile:eop.230504.p230727EOPcoverage:DATA-BASED1962-JAN-20TO2023-MAY-04.PREDICTS->2023-JUL-
26Referenceframe:EclipticofJ2000.0**********JDTDBXYZVXVYVZLTRGRR**********
**********$$SOE2460068.500000000=A.D.2023-May-
0400:00:00.0000TDBX=6.439857249172552E+07Y=5.346674700502940E+07Z=1.766200542283878E+05VX=5.932433425858534E+00VY=-9.068016481428440E+00VZ=-4.8
94541217796761E+00LT=2.791973394962744E+02RG=8.370125667464858E+07RR=-1.238470929543310E+002460069.500000000=A.D.2023-May-
0500:00:00.0000TDBX=6.496800946250510E+07Y=5.274906761607163E+07Z=-2.463364871165343E+05VX=7.230568569470446E+00VY=-7.535644675529745E+00VZ=-4.
893499593699156E+00LT=2.791468567349237E+02RG=8.368612232353663E+07RR=8.778380713596032E-
01$$EOE**********TIMEBarycentricDynamicalTime("TDB"orT_eph)outputwasreques
ted.Thiscontinuouscoordinatetimeisequivalenttotherelativisticpropertimeofaclockatrestinareferenceframeco-
movingwiththesolarsystembarycenterbutoutsidethesystem'sgravitywell.Itistheindependentvariableinthesolarsystemrelativisticequationsofmotion.TDBr
unsatauniformrateofoneSIsecondpersecondandisindependentofirregularitiesinEarth'srotation.CALENDARSYSTEMMixedcalendarmodewasactivesuchthatcalend
ardatesafterAD1582-Oct-15(ifany)areinthemodernGregoriansystem.Datespriorto1582-Oct-
5(ifany)areintheJuliancalendarsystem,whichisautomaticallyextendedfordatespriortoitsadoptionon45-Jan-
1BC.TheJuliancalendarisusefulformatchinghistoricaldates.TheGregoriancalendarmoreaccuratelycorrespondstotheEarth'sorbitalmotionandseasons.A"Greg
orian-
only"calendarmodeisavailableifsuchphysicaleventsaretheprimaryinterest.REFERENCEFRAMEANDCOORDINATESEclipticatthestandardreferenceepochReferencee
poch:J2000.0X-Yplane:adoptedEarthorbitalplaneatthereferenceepochNote:IAU76obliquityof84381.448arcsecondswrtICRFX-YplaneX-axis:ICRFZ-
axis:perpendiculartotheX-
Yplaneinthedirectional(+or-)senseofEarth'snorthpoleatthereferenceepoch.Symbolmeaning:JDTDBJulianDayNumber,BarycentricDynamicalTimeXX-
componentofpositionvector(km)YY-componentofpositionvector(km)ZZ-componentofpositionvector(km)VXX-componentofvelocityvector(km/sec)VYY-
componentofvelocityvector(km/sec)VZZ-componentofvelocityvector(km/sec)LTOne-waydown-legNewtonianlight-
time(sec)RGRange;distancefromcoordinatecenter(km)RRRange-
rate;radialvelocitywrtcoord.center(km/sec)ABERRATIONSANDCORRECTIONSGeometricstatevectorshaveNOcorrectionsoraberrationsapplied.Computationsby...
SolarSystemDynamicsGroup,HorizonsOn-LineEphemerisSystem4800OakGroveDrive,JetPropulsionLaboratoryPasadena,CA91109USAGeneralsite:https://ssd.jpl.
nasa.gov/Mailinglist:https://ssd.jpl.nasa.gov/email_list.ht…**********
```


Figure 8 : Données récupérée après suppression des espaces.

Pour formater le texte, il faut en premier définir l'expression régulière de l'élément que l'on souhaite exporter via des regex. Puis faire une reconnaissance dans le texte de l'élément spécifier par le regex et finalement récupérer la bonne valeur.
Pour crée les regex j'ai utilisé chatGPT qui a généré les regex en fonction des données que je luis ai transmis, étant donné que chaque planète a des données différentes il était plus rapide de demander à chatGPT de générer les regex en fonction des données voulues. (OpenAI, 2023)

Comme exemple le code ci-dessous qui cherche un texte « Vol. Mean Radius (km) = » qui est suivit d'une suite de chiffres.

Dans le regex nous avons plusieurs modifications possibles du texte :

1. [Mm] cherche la lettre M en majuscule ou minuscule.
2. [Rr] cherche la lettre R en majuscule ou minuscule.

Il existe une multitudes d'autres modifications possibles qui permettent de rechercher des données plus ou moins complexes.

```
const regexMeanRadius = /Vol\.[Mm]ean[Rr]adius\(km\)=([\d.]+)/
const matchMeanRadius = dataWithoutSpace.match(regexMeanRadius);
```

```
// Return data in object form
return {
  id : matchNameID[2],
  name : matchNameID[1],
  sizeRadius : matchMeanRadius[1],
  coordinate : {
    x : matchPlacment[1],
    y : matchPlacment[2],
    z : matchPlacment[3]
  },
  rotationSpeed : matchRotationRate[1],
  rotationDuration : matchRotationDays[2],
  orbitSpeed : matchOrbitalSpeed[2],
  orbitDuration : matchOrbitPeriod[2],
  oblliquity : matchObliquity[2],
  density : matchDensity[1],
  meanTemperature :  matchMeanTemp ? matchMeanTemp[2] : "Unknown"
};
```

Figure 9 : Exemple de l'utilisation de regex

Le système de requêtes est tout de même créé mais non utilisé :

```
export function GetHorizonSpecificBody(bodyId) {
    return new Promise(resolve => {
        const parameters = {};
        parameters.format = 'json';
        parameters.COMMAND = bodyId;
        parameters.OBJ_DATA = 'YES';
        parameters.MAKE_EPHEM = 'YES';
        parameters.EPHEM_TYPE = 'VECTORS';
        parameters.START_TIME = moment().subtract(1, 'day').format('YYYY-MM-DD');
        parameters.STOP_TIME = moment().format('YYYY-MM-DD');
        parameters.STEP_SIZE = '1d';

        fetch(formatURL(urlHorizon, parameters), {
            headers: {
                'Origin' : 'http://localhost:5173'
             }
        })
        .then(response => {
            resolve(response.json());
        })
        .catch(error => {
            console.error('Error:', error);
            resolve(error);
        });

    })
}
```

Figure 10 : Requête Horizon

4.1.1.2 Near Earth Objects

Url de requête : https://api.nasa.gov/neo/rest/v1/feed

La requête Near Earth Objects demande peu de paramètres, il suffit de passer dans l'url une date de début, une date de fin et une clé api donnée par la Nasa.

La fonction de requête ci-dessous montre qu'il est effectué un fetch de mon url formatée en passant url et les paramètres, puis la réponse est récupérée et retournée sous format json. Si une erreur apparait elle sera récupérée via le catch. (developer.mozilla.org, 2022)

```
export function GetNearEarthObjects(apiKey, startDate, endDate) {
    return new Promise(resolve => {
        const parameters = {};
        parameters.start_date = startDate;
        parameters.end_date = endDate;
        parameters.api_key = apiKey;

        fetch(formatURL(urlNeo, parameters))
        .then(response => {
            resolve(response.json())
        })
        .catch(function (err) {
            resolve("Something went wrong!", err);
        });

    })
}
```

Figure 11 : Fonction de création fetch des objets proches.

Voici les données récupérées :


```
{links: {…}, element_count: 25, near_earth_objects: {…}}
  element_count: 25
  links: {next: 'http://api.nasa.gov/neo/rest/v1/feed?start_date=20…&api_key=LTDBdyOvAiwdRywXsbe4dMf1eJrok44Ip5aZVe0F', previous: 'ht
  near_earth_objects:
    2023-05-11: Array(15)
      0:
        absolute_magnitude_h: 22.64
        close_approach_data: [{…}]
        estimated_diameter: {kilometers: {…}, meters: {…}, miles: {…}, feet: {…}}
        id: "2293726"
        is_potentially_hazardous_asteroid: false
        is_sentry_object: false
        links: {self: 'http://api.nasa.gov/neo/rest/v1/neo/2293726?api_key=LTDBdyOvAiwdRywXsbe4dMf1eJrok44Ip5aZVe0F'}
        name: "293726 (2007 RQ17)"
        nasa_jpl_url: "http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=2293726"
        neo_reference_id: "2293726"
        [[Prototype]]: Object
      1: {links: {…}, id: '2467460', neo_reference_id: '2467460', name: '467460 (2006 JF42)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/
      2: {links: {…}, id: '3092313', neo_reference_id: '3092313', name: '(2001 QN142)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.c
      3: {links: {…}, id: '3564040', neo_reference_id: '3564040', name: '(2011 HO5)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cgi
      4: {links: {…}, id: '3670297', neo_reference_id: '3670297', name: '(2014 JR2)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cgi
      5: {links: {…}, id: '3696301', neo_reference_id: '3696301', name: '(2014 WW4)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cgi
      6: {links: {…}, id: '3742055', neo_reference_id: '3742055', name: '(2016 CB31)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cg
      7: {links: {…}, id: '3841669', neo_reference_id: '3841669', name: '(2019 HF4)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cgi
      8: {links: {…}, id: '3878610', neo_reference_id: '3878610', name: '(2019 TQ4)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cgi
      9: {links: {…}, id: '3989253', neo_reference_id: '3989253', name: '(2020 BD11)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.cg
      10: {links: {…}, id: '54051138', neo_reference_id: '54051138', name: '(2020 QM)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.c
      11: {links: {…}, id: '54087420', neo_reference_id: '54087420', name: '(2020 VL)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.c
      12: {links: {…}, id: '54135432', neo_reference_id: '54135432', name: '(2021 GU3)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.
      13: {links: {…}, id: '54184284', neo_reference_id: '54184284', name: '(2021 PF7)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.
      14: {links: {…}, id: '54350904', neo_reference_id: '54350904', name: '(2023 FK2)', nasa_jpl_url: 'http://ssd.jpl.nasa.gov/sbdb.
      length: 15
      [[Prototype]]: Array(0)
    2023-05-12: (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
    [[Prototype]]: Object
  [[Prototype]]: Object
```


Figure 12 : Données récupérées via a l'api NEO.

4.2 Favicon

J'ai choisi comme favicon une image de soleil du site png art. (pngarts, -).

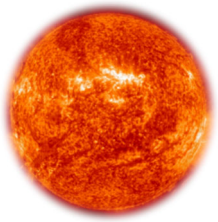


Figure 13 : Favicon

4.3 3D Affichages et animations

Ce projet utilise la librairie Three.js qui permet d'afficher sur des pages web du contenu en 3D.

Three.js est utilisé car il contient une énorme communauté du à sa grande popularité, il est facile à apprendre, il est compatible avec une large gamme de navigateurs et c'est une librairie que j'ai déjà expérimenté.

Threejs a besoin de 3 choses minimum :

- Une scène
- Une caméra
- Un moteur de rendu

La scène est une sorte de conteneur, elle permet de définir les éléments qui doivent être affiché ainsi que leurs placements. La caméra c'est elle qui va « filmer » ce que qu'il y a dans la scène et le moteur de rendu a pour objectif d'afficher la scène dans le navigateur via WebGL. (Bradley, 2022)

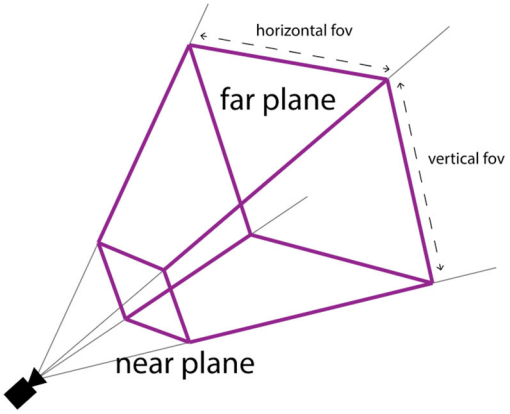


Figure 14 : Exemple de ce qu'est une scène. (Punkasem, 2017)

Il faut aussi ajouter une fonction d'animation, cette fonction crée une boucle qui fera afficher la scène via le moteur de recherche à chaque fois que l'écran est actualisé.

Quand un écran fait 60 images par seconde (frames per second ou FPS) c'est-à-dire qu'il y a 60 images qui sont affichées en une seconde

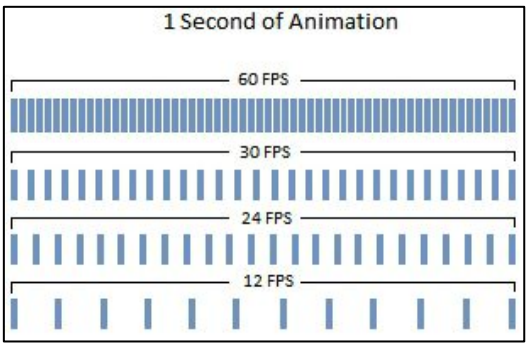


Figure 15 : Schéma d'animation par seconde.

Ce projet utilise programmation orientée objet (POO) qui réparti en plusieurs classes les éléments de mon projet.

- Renderer
  - Il est utilisé pour initialiser le rendu et géré l'animation.
- PlanetaryCelestialBody
  - Utiliser pour gérer les objets célestes planétaires.
- Planet
  - Hérite de PlanetaryCelestialBody.
- Moon
  - Hérite de PlanetaryCelestialBody.
  - Ajoute un paramètre orbitingBody qui définit planète autour de la quel elle orbite.
- Asteroid
  - Utilisé pour gérer les astéroïdes.

4.3.1 Classe Renderer

La classe Renderer est utilisé pour créer le rendu, il est défini dans celui-ci lors de la construction de la classe qu'il construit la scène, la caméra, la lumière, le renderer, la Skybox, le soleil et les planètes. Il prend comme paramètre un Canvas ainsi qu'une liste d'objets à ajouter dans la scène.

```
// Skybox creation
var skyboxGeometry = new THREE.SphereGeometry(-20000, 64, 16); // -20000 for facing interior
const skyboxTexture = new THREE.TextureLoader().load("./src/assets/images/Skybox.jpg");
const skyboxMaterial = new THREE.MeshBasicMaterial({ map: skyboxTexture });
const skybox = new THREE.Mesh( skyboxGeometry, skyboxMaterial );
this.#scene.add(skybox);
```

Figure 16 : Création de la skybox.

```
// Sun creation
var sunGeometry = new THREE.SphereGeometry(8, 64, 16);
const sunTexture = new THREE.TextureLoader().load("./src/assets/images/Sun.jpg");
const sunMaterial = new THREE.MeshBasicMaterial({ map: sunTexture });
this.#sun = new THREE.Mesh( sunGeometry, sunMaterial );
this.#sun.position.set(0,0,0);
this.#sun.rotation.x = 360;
this.#scene.add(this.#sun);
```

Figure 17 : Création du soleil.

```
// Create bodies
this.#bodiesList.forEach(bodies => {
    const bodiesSystem = bodies.planetarySystem;
    this.#scene.add(bodiesSystem);
    bodies.createOrbit();
});
```

Figure 18 : Création des planètes.

Il dispose d'une méthode animate qui se charge d'ajouter l'animation des objets, la rotation du soleil et de dessiner le rendu.

```
animate() {
    this.#bodiesList.forEach(bodies => {
        bodies.animation();
    });

    this.#sun.rotation.y += 0.00007292115;

    this.#renderer.render( this.#scene, this.#camera );
}
```

Figure 19 : Fonction d'animation du renderer

4.3.2 Classe PlanetaryCelestialBody

La classe PlanetaryCelestialBody est utilisée pour créer les objets célestes planétaires. Elle contient les méthodes de création de mesh, d'animation, de création d'orbite et de placement du système.
Elle prend pour paramètre :

- Un id
- Un nom
- Le radius de la taille
- Un fichier de texture
- Des coordonnées
- Une vitesse de rotation
- Une durée de rotation
- Une vitesse orbitale
- Une durée orbitale
- Une température moyenne

4.3.3 Classe Asteroid

4.3.4 Responsive

```
// Resize render when listen to resize
window.addEventListener('resize', () => {
    this.#camera.aspect = window.innerWidth / window.innerHeight;
    this.#camera.updateProjectionMatrix();
    this.#renderer.setSize(window.innerWidth, window.innerHeight);
    this.animate();
});
```

Figure 20 : Code permettant au redimensionnement du renderer

4.3.5 Mouvement utilisateur

```
// Add users controls
new OrbitControls( this.#camera, this.#renderer.domElement );
```

Figure 21 : Code permettant le mouvement de l'utilisateur

4.4 Corps céleste planétaire

Les corps célestes planétaires définissent tout corps en orbite autour du soleil.

4.4.1 Création

Une mesh est objets basés sur un maillage polygonal triangulaire.

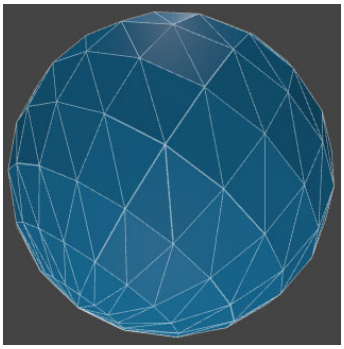


Figure 22 : Exemple de mesh sur une sphère

```
#createMesh() {
  const geometry = new THREE.SphereGeometry(this.sizeRadius / 1000, 64, 16 );
  const texture = new THREE.TextureLoader().load("./src/assets/images/" + this.textureFile);
  const material = new THREE.MeshBasicMaterial({ map: texture });
  const mesh =  new THREE.Mesh(geometry, material);
  mesh.position.set(parseFloat(this.coordinates.x) / 5000000, parseFloat(this.coordinates.y) / 5000000, 0);
  mesh.rotation.set(3, 3, 0)
  return mesh;
}
```

Figure 23 : Création physique des planètes

4.4.2 Animation

```
animation() {
  const deltaTime = this.#clock.getDelta();

  this.mesh.rotation.x = 190
  this.mesh.rotation.y += this.rotationSpeed * 10;

  const rayon = (Math.sqrt((Math.pow(this.coordinates.x, 2) /5000000 ) + (Math.pow(this.coordinates.y, 2)/5000000)))
  const distanceFactor = 2 * Math.PI * rayon; //km périmèter
  const timeFactor = this.orbitDuration * 365 * 24 * 60 * 60; //years => seconds

  const vitesse =  distanceFactor / timeFactor;
  this.#pointPivot.rotation.z += vitesse;

}
```

Figure 24 : animation des planètes

4.5 Planètes

Les planètes héritent de la classe corps céleste planétaire et n'a aucune modification.

4.6 Moon

4.7 Astéroïde

Les astéroïdes n'héritent d'aucunes classes car elles sont particulières et n'ont pas le même comportement que les lunes ou planètes.

```
constructor(id, name, estimatedDiameter, closeAprocheDate, potentiallyHazardous, orbitingBody, missDistance) {
  this.id = id;
  this.name = name;
  this.estimatedDiameter = estimatedDiameter;
  this.closeAprocheDate = closeAprocheDate;
  this.potentiallyHazardous = potentiallyHazardous;
  this.orbitingBody = orbitingBody;
  this.missDistance = missDistance;
}
```

Figure 25 : Constructeur de la classe Ateroid

4.8 Déploiement

Le déploiement s'effectue sur swisscenter, afin de parvenir à la connexion au serveur j'ai créé une clé ssh depuis la plateforme de swisscenter.

Site web : https://stellarmap.mycpnv.ch/

4.9 Répertoires

Code source : https://github.com/Juillet-Mikael/TPI
Planification du projet : https://icescrum.cpnv.ch/p/TPIJUILLET/#/project
Documentation se situe aussi dans un dossier nommé doc au sein du projet Git.

Architecture des documents :

- TPI
  - documents
    - journals
    - documentation
    - planification initiale
    - diagrams
      - diagrame de classe
      - diagrame de scéquence
    - instruction
  - src
    - model
    - view
    - controller
  -

4.10 Description des tests effectués

```
test('horizonAPIFilter() should return Jupiter data', () => {

  //Given
  const jupiter = Allplanets.planets[4].result;
  const jupiterExpected = {
      id : "599",
      name : "Jupiter",
      sizeRadius : "69911",
      material : "Jupiter.png",
      coordinate : {
        x : "7.900239544305509E+08",
        y : "3.977974646088730E+08",
        z : "-1.643804799922679E+07"
      },
      rotationSpeed : "0.00017585",
      rotationDuration : "9h55m29.71s",
      orbitSpeed : "13.0697",
      orbitDuration : "11.861982204",
      oblliquity : "3.13",
      density : "1.3262",
      meanTemperature : "Unknown"
  };

  //When
  const result = horizonAPIFilter(jupiter);

  //Then
  expect(result).toEqual(jupiterExpected);

})
```

```
 DEV  v0.31.0 C:/Users/Mikael.JUILLET/Desktop/TPI

 ✓ tests/nearEarthObjectsRequests.test.js (2) 1343ms
 ✓ tests/dataFilter.test.js (9)

 Test Files  2 passed (2)
      Tests  11 passed (11)
   Start at  14:23:34
   Duration  2.17s (transform 349ms, setup 0ms, collect 716ms, tests 1.35s, environment 0ms, prepare 275ms)


 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

4.11 Erreurs restantes

4.12 Liste des documents fournis

5 Conclusions

6 Bibliographie

balsamiq. (2023, - -). *balsamiq*. Récupéré sur balsamiq: https://balsamiq.com/

Bradley, S. (2022). *Scene, Camera and Renderer*. Récupéré sur sbcode.net: https://sbcode.net/threejs/scene-camera-renderer/#:~:text=The%20Renderer%20displays%20the%20scene,2D%20image%20for%20the%20Canvas.

Braiam. (2017, Mai 09). *No 'Access-Control-Allow-Origin' header is present on the requested resource—when trying to get data from a REST API*. Récupéré sur stackoverflow: https://stackoverflow.com/questions/43871637/no-access-control-allow-origin-header-is-present-on-the-requested-resource-whe

claude aubry. (21, mai 2018). https://claudeaubry.fr/post/2018/extraits-du-livre-scrum/. Paris, Paris, France.

cpnv.ch. (-, - -). *Icescrum*. Récupéré sur Icescrum.cpnv.ch: https://icescrum.cpnv.ch/#/

day.js. (2023, - -). *day.js*. Récupéré sur day.js: https://day.js.org/

developer.mozilla.org. (2022, Décembre 23). *Utiliser Fetch*. Récupéré sur developer.mozilla.org: https://developer.mozilla.org/fr/docs/Web/API/Fetch_API/Using_Fetch

figma. (2023, - -). *figma*. Récupéré sur figma: https://www.figma.com/

freepik. (-, - -). Vecteur gratuit système de système solaire classique avec deisgn plat. *Vecteur gratuit système de système solaire classique avec deisgn plat*. -, -, -. Récupéré sur https://fr.freepik.com/

kjpargeter. (s.d.). Free photo starry night sky. *Free photo starry night sky.* feepick, -. Récupéré sur https://www.freepik.com/free-photo/starry-night-sky_7061153.htm#query=stars&position=2&from_view=search&track=sph

momentjs. (2023, - -). *momentjs*. Récupéré sur momentjs: https://momentjs.com/

mozilla. (2023, Mai 05). *Regular expressions*. Récupéré sur developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions

Nasa. (2022, Septembre 1). *Horizons API*. Récupéré sur ssd-api: https://ssd-api.jpl.nasa.gov/doc/horizons.html

OpenAI. (2023, - -). *Introducing GPT-4, OpenAI's most advanced system*. Récupéré sur Introducing GPT-4, OpenAI's most advanced system: https://openai.com/

pinia. (2023, - -). *pinia*. Récupéré sur pinia: https://pinia.vuejs.org/

pngarts. (-, - -). *Images Transparentes de soleil*. Récupéré sur pngarts: https://www.pngarts.com/fr/explore/123391

Punkasem. (2017). *Learning Three.js*. Récupéré sur junethanaon.com: https://junethanaon.com/blog/blog-threejs.html

Simplified, W. D. (2021, Mai 22). *Apprenez CORS en 6 minutes*. Récupéré sur Youtube: https://www.youtube.com/watch?v=PNtFSVU-YTI

three.js. (2023, - -). *three.js*. Récupéré sur three.js: https://threejs.org/

visualstudio. (2023, - -). *visualstudio*. Récupéré sur visualstudio: https://code.visualstudio.com/

vitejs. (2023, - -). *vitejs*. Récupéré sur vite: https://vitejs.dev/

vitest. (2021, - -). *vitest*. Récupéré sur vitest: https://vitest.dev/

vuejs. (2023, - -). *vuejs*. Récupéré sur vuejs: https://vuejs.org/guide/components/provide-inject.html#prop-drilling

W3schools. (-, - -). *AJAX Introduction*. Récupéré sur W3schools: https://www.w3schools.com/js/js_ajax_intro.asp
Wikipedia. (22, Février 2023). https://fr.wikipedia.org/wiki/Mod%C3%A8le_en_cascade. -, -, -.
wrike. (-, - -). *wrike*. Récupéré sur wrike.com: https://www.wrike.com/main/

7 Table des illustrations

8 Lexique

M

mesh
Objets basés sur un maillage polygonal triangulaire. · 19
méthodologie Waterfall
Modèle en cascade qui consiste à la succession d'étape prédéfinies. · 5

R

regex
Regex ou expressions rationnelles sont des motifs de combinaisons de caractères au sein de chaînes d'un texte. · 14

S

scène
Définit les éléments qui doivent être affiché ainsi que leurs placements. · 17

U

Un moteur de rendu
Responsable de l'affichage graphique des objets 3D dans une scène. · 17

9 Annexes

9.1 Planification initiale

Description	Catégorie	Progrès	Début	Heures prévu
Sprint 1				
Diagramme de classes	Base du projet	0%	03.05.2023	0.75
Diagramme de séquence	Base du projet	0%	03.05.2023	0.75
Création de la classe planète	Base du projet	0%	03.05.2023	1.00
Création de la classe satellite	Base du projet	0%	03.05.2023	1.00
Ajout des opérations dans les classes	Base du projet	0%	05.05.2023	2.25
Ajout de vitejs	Base du projet	0%	05.05.2023	0.25
Création du fichier détenant les codes de planètes	Base du projet	0%	05.05.2023	0.50
Sprint 2				
Création d'un contrôleur	API	0%	08.05.2023	0.25
Création d'un modèle	API	0%	08.05.2023	0.25
Ajout d'un fichier .env	API	0%	08.05.2023	0.25
Récupération de la clef API	API	0%	08.05.2023	0.25
Requêtes de récupération des planètes	API	0%	08.05.2023	3.00
Requêtes de récupération des objets proches	API	0%	09.05.2023	3.00
Requêtes de récupération des images	API	0%	11.05.2023	2.25
Récupération des erreurs dans le contrôleur	API	0%	12.05.2023	0.75
Lien entre la récupération des données et les classes	API	0%	12.05.2023	0.75

Sprint 3				
Création de maquettes	Planettes	0%	14.05.2023	0.75
Ajout de three.js	Planettes	0%	14.05.2023	0.25
Création des planètes	Planettes	0%	14.05.2023	0.75
Placement des planètes	Planettes	0%	15.05.2023	0.50
Orbite sidérale	Planettes	0%	15.05.2023	0.50
Orbite autour du soleil	Planettes	0%	15.05.2023	0.75

Sprint 4				
Ajout du déplacement utilisateur	Satelites	0%	22.05.2023	2.25
Création des Satélites	Satelites	0%	23.05.2023	1.50
Ajout des lunes	Satelites	0%	25.05.2023	1.50
Placement sur la carte	Satelites	0%	25.05.2023	1.50
Ajout de l'orbite	Satelites	0%	26.05.2023	1.50
Orbite sidérale	Planettes	0%	26.05.2023	1.50

Sprint 5				
Création de maquettes	UI	0%	30.05.2023	0.75
Placement du canvas en arrière-plan	UI	0%	30.05.2023	0.75
Ajout de la desciption des planettes	UI	0%	01.06.2023	2.25
Ecoute d'un clique sur planètes	UI	0%	02.06.2023	0.75
Ajout de changement de vitesse	UI	0%	02.06.2023	1.50

Planification de projet

Système solaire

Mikael Juillet

Date de début 02.05.2023

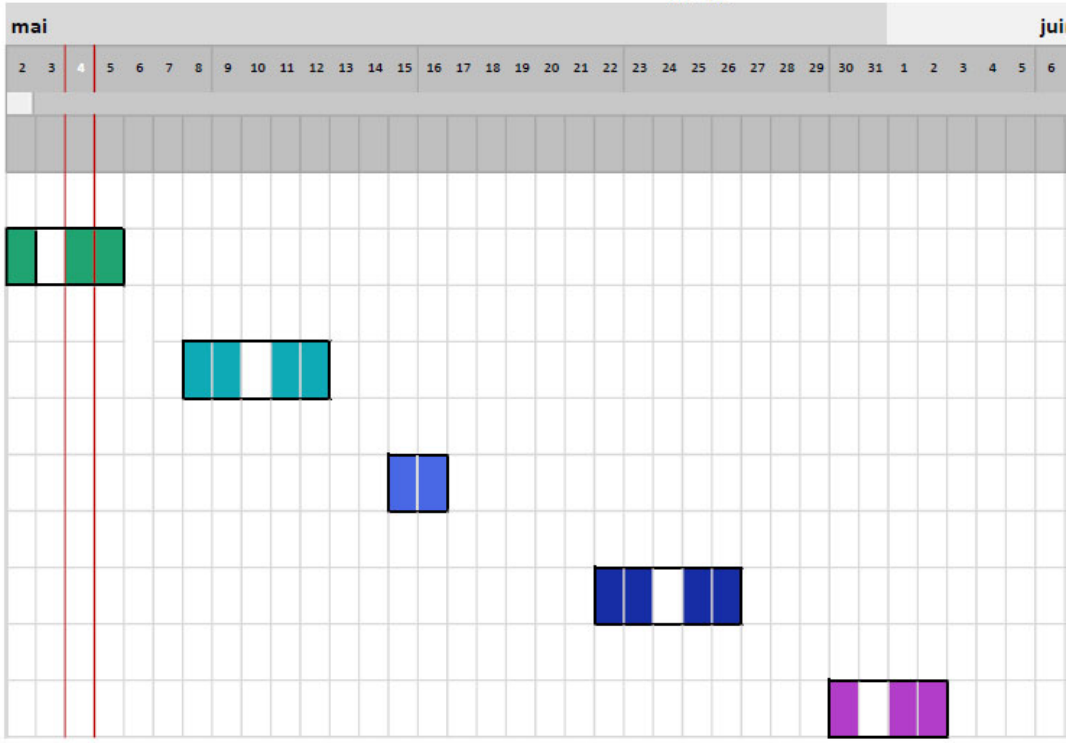
Incrément de défilement 0

Légende :

Définition :

Base du projet	API	Planettes	Satelites	UI
Création de la base du projet, création des classes des planètes et satellites.	Création de toutes les requêtes à l'API.	Affichage des planètes sur leurs placements définit par l'api.	Affichage des satellites (objets proches) sur leurs placements définis par l'api.	Affichage de la description des planètes et du changement de vitesse.

Description du jalon	Catégorie	Progrès	Start	jours
Sprint 1				
Semaine 1	Base du projet	0%	02.05.2023	3
Sprint 2				
Semaine 2	API	0%	08.05.2023	4
Sprint 3				
Semaine 3	Planettes	0%	15.05.2023	2
Sprint 3				
Semaine 4	Satelites	0%	22.05.2023	4
Sprint 4				
Semaine 5	UI	0%	30.05.2023	3



Exemple d'un sprint d'une semaine

Système solaire

Mikael Juillet

Date de début 02.05.2023

Légende: Documentation | Implémentation | Annalyse | Tests

Sprint reviews

* P = Période de 45 minutes

Jours
Lundi
Mardi
Mercredi
Jeudi
Vendredi
Samedi
Dimanche

Horaire

P1	P2	P3	P4	P5	P6	P7	P8	P9

Note : Il est important de prendre en compte que la disposition changera car il y a par exemple des semaines avec seulement deux jours mais dans ces deux jours il y aura de l'analyse et des tests même s'ils ne sont pas prévus. C'est un schéma approximatif.

9.2 Résumé du rapport du TPI / version succincte de la documentation

9.3 Journal de travail

9.4 Archives du projet

Media, … dans une fourre en plastique