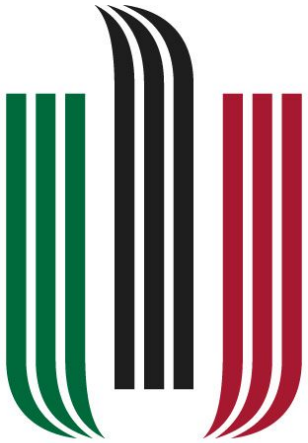


AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Ruch planet Układu Słonecznego

Autorzy: Borowy Szymon, Kołodziej Dominik, Kozak Marcin
Kierunek studiów: Informatyka
Opiekun projektu: mgr Grzegorz Bazior

Kraków, 2019

Wstęp

Opis pracy

Celem symulacji jest zobrazowanie ruchu planet w Układzie Słonecznym. Takie symulacje w zależności do stopnia skomplikowania i dokładności mogą służyć różnym celom. Ich podstawowym założeniem jest podawanie (pokazywanie) orientacyjnego położenia ciał względem siebie.
Bardziej dokładne symulacje, bazujące na setkach tysięcy pomiarów obiektów Układu Słonecznego mogą służyć nawet symulacji trajektorii wystrzelonych satelitów, sond, czy statków kosmicznych. Pozwalają one także zaobserwować takie zjawiska jak retrogradacja[1] (np: Marsa) czy libracja (np: Księżyca). Prostsze symulacje, ze względów na swoje niedokładności spowodowane m.in: brakiem dostępu do specjalistycznego sprzętu zarówno pomiarowego jak i obliczeniowego, czy nieidealnym oddaniem praw działających we Wszechświecie mogą być wykorzystywane jako nie zaawansowane materiały edukacyjne.
Naszym celem jest zbudowanie prostej symulacji Układu Słonecznego, opartej na podstawowych prawach i własnościach fizycznych i na podstawie bardziej zaawansowanych symulacji zbadać, jak dobrze nasza symulacja odwzorowuje rzeczywistość.

[1] https://pl.wikipedia.org/wiki/Retrogradacja_(astronomia)

Opis literatury

Przystępując do realizacji projektu, postanowiliśmy najpierw przeglądnąć dostępne symulacje i popatrzeć jakie rozwiązania zostały tam zastosowane. Znaleźliśmy również publikację naukową wyczerpującą problematykę naszej symulacji.

jsOrrery[2] [3]

Twórca tej symulacji chciał jak najdokładniej odwzorować Układ Słoneczny. Do opisania orbit planet użył zbiorów 6 liczb nazywanych elementami orbitalnymi[4]. Elementy orbitalne są parametrami które wylicza się dla planety na podstawie zasad dynamiki Newtona oraz prawa powszechnego ciążenia. Ruch ciała można sparametryzować na kilka sposobów, co sprawia że istnieje wiele zbiorów elementów orbitalnych, jednoznacznie identyfikujących orbitę danego ciała, równoważnych względem siebie,. W zależności od wyliczanych parametrów liczba elementów orbitalnych potrzebnych do jednoznacznego wyliczenia orbity waha się między 6-8 parametrów[5]. Mając parametry orbit twórca obliczył prędkość używając równania vis-viva, będącego wynikiem zasad zachowania energii mechanicznej. Był świadom że algorytmy użyte przez niego nie są idealne: *“The algorithm I use to calculate the motion of the bodies is not 100% accurate. It is good enough to view the motion of the planets over a short period of time, but if the scenario is kept playing for a long time, the positions become less and less accurate.”*[6]

The Sky Live[7]

Strona ta, której częścią jest symulacja 3D Układu Słonecznego, dostarcza rzeczywistych danych ponad 300 obiektów Układu Słonecznego. Dane do symulacji Układu Słonecznego pochodzą m.in z projektu Star Chart Dan’a Burtona[8]. Do symulacji orbit użyto javascriptowej biblioteki three.js, a do obliczeń pozycji planet biblioteki astronomy.js Don’a Cross’a. Dane o kącie obserwacji planet, fazach księżyca, położeniach komet, satelitów i innych obiektów pochodzą między innymi z danych zebranych przez misje Hipparcos oraz katalogów Tycho 2, PGC 2003 i PGC 2.3. Obrazy pochodzą m.in z teleskopu Samuela Oschina, czy z teleskopu Schmidta [9]

[2] https://medium.com/@m_vezina/building-jsorrery-a-javascript-webgl-solar-system-7330e71d64d5
[3] https://mgvez.github.io/jsorrery/
[4] https://marine.rutgers.edu/cool/education/class/paul/orbits.html#7
[5] https://en.wikipedia.org/wiki/Orbital_elements#Alternative_parametrizations
[6] https://medium.com/@m_vezina/building-jsorrery-a-javascript-webgl-solar-system-7330e71d64d5
[7] https://theskylive.com/
[8] http://www.midnightkite.com/index.aspx?AID=0&URL=StarChartFAQ
[9] https://theskylive.com/about#acknowledgements

Solar System Dynamics[10]

Publikacja naukowa wprowadzając czytelnika w zagadnienia związane z fizyką Układu Słonecznego. Znajdziemy tam proste prawa i zależności fizyczne dotyczące grawitacji jak i bardzo skomplikowane wzory, równania do opisu ruchów zachodzących w naszym układzie. Znaleźliśmy tam szczegółowo opisane problemy dotyczące między innymi:

- praw Keplera,
- problemu 2 i 3 ciał,
- kształtu orbit planet,
- wpływu ruchów księżyców na orbity okrążanych planet.

Jest to praca naukowa, nastawiona na przekazanie rzeczywistej wiedzy czytelnikowi. Podchodzi do problemu w sposób naukowy, obrazując niektóre prawa i zależności fizyczne w sposób, ciężki do przedstawienia w sposób programistyczny, a łatwe do zobrazowania w sposób matematyczno-fizyczny.
Jest to stosunkowo dobre źródło do poszerzenia swojej wiedzy w dziedzinie astronomii jak i do głębszego zrozumienia zasad i praw zastosowanych w symulacjach komputerowych.

[10] Solar System Dynamics / C.D. Murray, S.F. Dermott

Proponowany model

Wielkości w Układzie Słonecznym

Przystępując do realizacji wybranego modelu Układu Słonecznego natknęliśmy się na kilka problemów związanych ze sposobem implementacji i przedstawienia naszej symulacji.
Pierwszym problemem były wielkości w Układzie Słonecznym. Początkowo chcieliśmy precyzyjnie (na ile pozwalały na to dostępne dane) przedstawić wielkości planet i ich odległości od Słońca. Poniżej przedstawiam tabelkę z wybranymi danymi planet[11] Układu Słonecznego:

Nazwa Planety	**Średnica (równikowa w km)**	**Odległość od Słońca (km)**
Słońce	ok 1 392 000	nie dotyczy
Merkury	4 879	57 909 170
Wenus	12 104	108 208 926
Ziemia	12 756	149 597 887
Mars	6 805	227 936 637
Jowisz	142 984	778 412 027
Saturn	120 536	1 426 725 413
Uran	51 118	2 870 972 220
Neptun	49 528	4 498 252 900

Średnice i odległości planet od Słońca są wielkościami różnych rzędów:

1. Chcąc przedstawić odpowiednio planety: zakładając że Słońce będzie miało wielkość ok 500px, to najmniejsza pod względem średnicy planeta w Układzie Słonecznym- Merkury miałby wielkość ok 1,75px. Wielkość Merkurego nie byłaby największym problemem z tego powodu że przy takiej skali układu najdalej oddalona planeta od Słońca- Neptun znajdowałby się ok 1 615 700 px od Słońca. Powoduje to że przedstawienie Układu Słonecznego w pojedynczym widoku jest niemożliwe

[11] Wszystkie dane brane z tej tabeli: https://pl.wikipedia.org/wiki/Uk%C5%82ad_S%C5%82oneczny#Wi%C4%99ksze_cia%C5%82a_niebieskie

2. Chcąc dobrze odwzorować odległości od Słońca okazuje się że choć wszystkie planety “mieszczą się” na jednym ekranie, to są niemożliwe do zobaczenia ze względu na swoją wielkość (średnice ok 0,001px)

Dlatego zdecydowaliśmy się na swego rodzaju kompromis w przedstawieniu tych wielkości: stosunek średnic planet do siebie jest zachowany. Stosunki odległości planet od Słońca też są między sobą zachowane. Jednakże 1px w długości średnicy planet nie jest równy 1px w odległości planety od Słońca. Planety dalej są bardzo małe, przez co zdecydowaliśmy się dodać “ślady” planet które pomagają zlokalizować obiekt w układzie.

Obrazowanie wielkości w kosmosie jest znanym problemem, który twórcy różnych symulacji rozwiązali na różne sposoby. Np: TheSkyLive[12] zaniedbał wielkości planet (Słońce niewiele większe od np: Merkurego). 3D Solar System Simulator[13] również pominął tę kwestię, a twórca jsOrrery [14]napisał tak:

“I chose, though, to keep the planets to their real scale in regards to the size of their orbits, and to the Solar System as a whole. I wanted to keep a feeling of the vastness of the Solar System. Most visualisations that we see always have the planets heavily scaled up, otherwise we couldn’t see them, but I thought that it gave a false impression of what the real thing is like. I eventually added the possibility to scale the planets at will in the simulation, still I prefer it to be 1:1 scale.”[15]

Liczby wysokich rzędów są zwykle ciężkie do wyobrażenia, dlatego podajemy dwa źródła, które w całkiem przystępny sposób pokazują problem wielkości w Układzie Słonecznym:

http://www.phrenopolis.com/perspective/solarsystem/

http://joshworth.com/dev/pixelspace/pixelspace_solarsystem.html.

Eliptyczność orbit

Mimośród (ekscentryczność), to *“liczba zdefiniowana dla przecięcia stożkowego*[16] *,wyrażana za pomocą wielkiej półosi a i małej półosi b*”[17] wyraża się wzorem:

$$e = \sqrt{1 - (\frac{b}{a})2}$$

Im mniejszy mimośród, tym elipsa bardziej przypomina okrąg(mimośród dla okręgu wynosi 0)

[12] https://theskylive.com/

[13] http://solarsystem.appzend.net/?q=solarsimulator1

[14] https://mgvez.github.io/jsorrery/

[15] https://medium.com/@m_vezina/building-jsorrery-a-javascript-webgl-solar-system-7330e71d64d5#e67c

[16] okrąg/elipsa jest też przecięciem stożkowym

[17] Tłumaczenie z angielskiego. Dosłowny cytat dostępny:
http://mathworld.wolfram.com/Eccentricity.html

Nasz model zakłada także że trasy obiegów planet nie są elipsami (jak w rzeczywistości) tylko okręgami o wspólnym środku, w którym to leży Słońce. Założenie to, biorąc pod uwagę, że nasza symulacja jest dwuwymiarowa, nie jest bardzo dużym zaniedbaniem. Popatrzmy na mimosródy planet Układy Słonecznego:

Planeta	Mimośród
Merkury	0,2056[18]
Wenus	0,00677323[19]
Ziemia	0,01671123[20]
Mars	0,09341233[21]
Jowisz	0,04839266[22]
Saturn	0,05415060[23]
Uran	0,04716771[24]
Neptun	0,00858587[25]

Największy mimośród ma Merkury (0,2056). Różnica między jego apocentrum a perycentrum wynosi 0,1592 AU (ok 23 820 000km). Skala przyjęta przy odległości planet od Słońca jest następująca: 7,742px = 1 AU. Powoduje to, że różnica w orbicie Merkurego byłaby praktycznie niezauważalna na naszej symulacji.

[18] David R. Williams: Mercury Fact Sheet (ang.). W: Planetary Fact Sheets [on-line]. NASA Goddard Space Flight Center, 2016-07-13
[19] David R. Williams: Venus Fact Sheet (ang.). NASA, 2016-12-23
[20] E. Myles Standish, Williams, James C: Orbital Ephemerides of the Sun, Moon, and Planets (ang.). International Astronomical Union Commission 4: (Ephemerides)
[21] David R. Williams: Mars Fact Sheet. W: National Space Science Data Center. NASA, 2016-12-23
[22] Jupiter Fact Sheet (ang.). 2016-12-23
[23] Dr David R. Williams: Saturn Fact Sheet. NASA, 2016-12-23
[24] Ed Grayzeck: Uranus Fact Sheet (ang.). NASA, 22 grudnia 2015
[25] David R. Williams: Neptune Fact Sheet (ang.). W: Planetary Fact Sheets [on-line]. NASA, 2016-12-23.

Technologie i narzędzia

Technologie

Do zwizualizowania naszej symulacji wybraliśmy język skryptowy JavaScript, HTML 5 i CSS 3. Dzięki takiemu doborowi technologii zrozumienie kodu symulacji w celu przyszłych samodzielnych modyfikacji nie będzie stanowiło problemu. Kod naszej symulacji jest publiczny i gotowy do uruchomienia. Znajduje się w serwisie GitHub [26]. Głównym graficznym elementem symulacji jest obiekt canvas[27]. Dzięki swojej prostocie jest dobrym narzędziem do rysowania prostych obiektów 2D i manipulacji na nich. Wybranie JavaScriptu umożliwia uruchomienie symulacji na każdej platformie (przeglądarce internetowej) w niezmienionej formie. Dodatkowo JavaScript działa lokalnie w przeglądarce, więc symulacja do celów edukacyjnych nie wymaga żadnego serwera czy specjalnego dedykowanego oprogramowania. Aplikację zdecydowaliśmy się umieścić w serwisie hostingowym CBA, ponieważ oferuje on darmowy hosting stron internetowych[28].

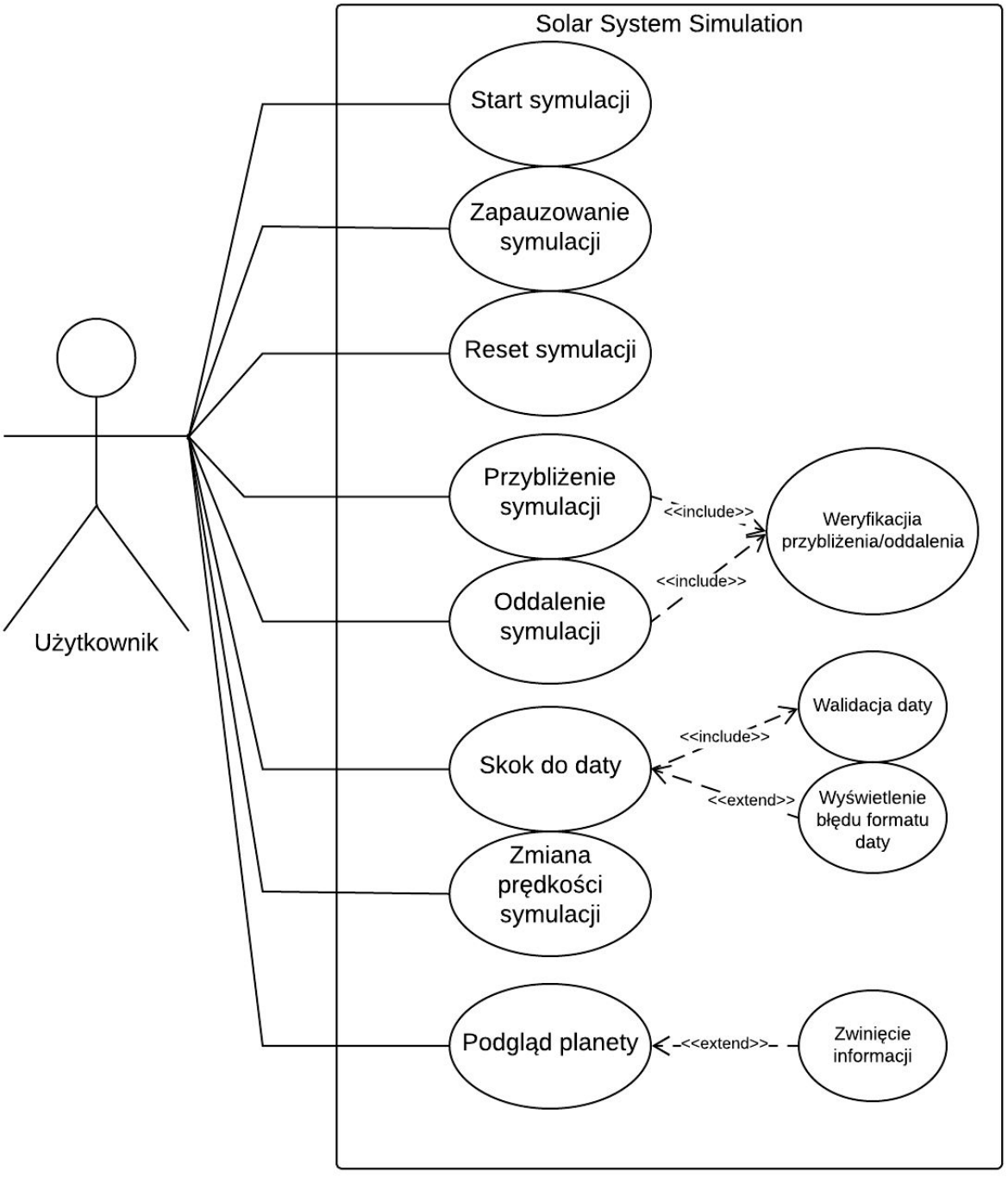
Narzędzia

Stworzenie symulacji wymagało odpowiednich narzędzi. Jako zespół zdecydowaliśmy się na bezpłatną[29] wersję oprogramowania WebStorm 2018.3.5 dostarczanego przez firmę JetBrains. Ten edytor pozwala na tworzenie plików HTML, CSS, JavaScript. Pozwala na wybranie przeglądarki do uruchamiania symulacji (z tych zainstalowanych).
Kod naszej aplikacji był bezpośrednio publikowany w opensourcowym serwisie GitHub z poziomu WebStorma (jest to praktyczna funkcja tego edytora). GitHub jest jednym z najbardziej popularnych systemów kontroli wersji.

[26] https://github.com/kolorowerowe/AGH_Solar_system_simulation
[27] https://www.w3schools.com/html/html5_canvas.asp
[28] http://www.sssimulation.cba.pl/
[29] https://www.jetbrains.com/student/

Zastosowane rozwiązania

User case diagram



Opracowanie własne przy pomocy strony Lucichart.com

Parametry obiektów

Ruch planet realizowany jest przez funkcję draw. Jeżeli stan symulacji(running) ma wartość true, realizowane jest przesuwanie planet.
Planety są obiektami, których kod źródłowy znajduje się w pliku objects.js. Wszystkie obiekty trafiły do zbiorczej tablicy objects, która ma nam ułatwić iteracją po wszystkich obiektach.
Przykładowy kod jednej z planet poniżej:

```
let earth = {
    "name": "Ziemia",
    "radius": 12756*radius_scale/2,
    "color": "#0000FF",
    "circle_time": 365.25,
    "spin_time": 24,
    "initx": distance_scale,
    "x": 0,
    "init_rotate": 110/360*2*3.14,
    "rotate": 0,
    "info": "<b>Planeta:</b> <span style=\"color: #0000FF;
font-size: 40px;\"> Earth </span>" + "<br />" +
            "<b>Aktualny obrót:</b> " + "<div id='obrot'></div>" +
"<br />" +
            "<b>Promień równikowy:</b> 6378.1366 ± 0.0001 km" +
"<br />" +
            "<b>Średni promień:</b> 6371.0084 ± 0.0001 km" + "<br
/>" +
            "<b>Masa:</b> 5.97237 ± 0.00028 x10<sup>24</sup> kg" +
"<br />" +
            "<b>Gęstość objętościowa:</b> 5.5136 ± 0.0003"+"
g/cm<sup>3</sup>" + "<br />" +
            "<b>Okres obrotu gwiazdowego:</b> 0.99726968 dni" +
"<br />" +
            "<b>Okres orbity gwiazdowej:</b> 1.0000174 lat" + "<br
/>" +
            "<b>Grawitacja równikowa:</b> 9.80 "+"
m/s<sup>2</sup>" + "<br />" +
            "<b>Prędkość ucieczki:</b> 11.19 km/s" + "<br />"+
            "<img src='src/img/earth.jpg' height='300' width='300'
>"
};
```

Pokrótce omówię niektóre pola.
Radius- wszystkie wartości (w tym przypadku 12756) liczbowe są średnicami branymi z prac David R. Williams Planetary Fact Sheets[30]. Człon /2 bierze się właśnie z faktu że wprowadzone zostały średnice a nie promienie. czynnik radius_scale jest parametrem który ulega zmianie przy przybliżaniu, oddalaniu układu za pomocą przycisków w lewym panelu.
circle_time- wartość kluczowa dla naszej symulacji. Jest to czas (w dniach ziemskich) obiegu planety wokół Słońca. Wykorzystywany później przy algorytmie obliczającym pozycję planet.
spin time- czas (w ziemskich godzinach) obrotu planety wokół własnej osi. Wartość niewykorzystywana na tym etapie symulacji. Ewentualne wykorzystanie podczas dalszego rozwoju aplikacji.
initx- jest to odległość (w jednostkach astronomicznych AU) od Słońca. Wartość potrzebna do resetu położenia planet układu
init_rotate- wartość kąta o jaki obrócona była planeta w dniu 14-04-2019 (dzień początkowy symulacji)
rotate- o jaki kąt (w radianach) przesunęła się planeta od położenia ***init_rotate***
info- informacje o planecie mające się wyświetlić po wysunięciu prawego panelu-kliknięciu na planetę

Algorytm obrotu planet

Kod algorytmu przesuwania planet w języku JS

```
for (let item of objects)
      {
        if(item.name !== "Sun" && item.name !== "Asteroid belt")
        {
            ctx.save();
            let przes =0;
            if (running === 1) {
                przes = (2 * Math.PI * gain / (item.circle_time *
100));
            }

          ctx.rotate(item.rotate+przes);
          circle(item.radius, item.color,item.x ,0);
          planet_arc(item);
```

[30] https://nssdc.gsfc.nasa.gov/planetary/factsheet/

```
        ctx.font = "25px Montserrat";
        ctx.fillText(item.name,item.x,0);
        item.rotate+=przes;
        ctx.restore();
      }
      else
      {
       circle(item.radius, item.color,item.x ,0);
      }

        if(item.rotate % item.circle_time === 0){item.rotate=0;}


    }
```

Wyjaśnienie:
Dla każdego obiektu w tablicy objects (tablica ciał w układzie Słonecznym), jeżeli nie jest to ani Słońce, ani Pas Asteroid zapisz stan canvasa. Następnie przystępujemy do operacji przesunięcia:

```
przes = (2 * Math.PI * gain / (item.circle_time * 100));
```

Przesunięcie jest uzależnione od 2 czynników- item.circle_time (czyli czas obiegu planety wokół Słońca), oraz częstotliwości odświeżania funkcji draw. Ta częstotliwość jest zaszyta pod stałą 100.
Następnie dokonujemy obrotu canvasa o kąt będący sumą tego o ile planeta przesunęła się do tej pory i wartości o jaką powinna się przesunąć w tej części czasu.
Następnie mamy wyświetlenie planety oraz ogona który pomaga w jej lokalizacji. Aktualizujemy pole init_rotate w danym obiekcie, a następnie przywracamy parametry canvasa za pomocą funkcji restore();
Aby uniknąć przepełnienia zmiennych przechowujących wartość kąta o jaki obróciła się planeta od położenia początkowego, wartość item.rotate jest zerowana za każdym razem jak będzie całkowitą wielokrotnością zmiennej circle_time będącą czasem obiegu planety wokół Słońca.

Algorytm skoku do daty

Skok do daty jest wykonywany w następujący sposób: funkcja jumpDate() sprawdza najpierw czy data wprowadzona przez użytkownika jest prawidłowa i jeżeli nie to wyświetlany jest stosowny komunikat. Jeśli dane przejdą pomyślnie weryfikację,

obliczana jest różnica w milisekundach między aktualną datą, a wprowadzoną. Następnie do aktualnego obrotu każdej z planet jest dodawana wartość o jaką muszą zostać przesunięte aby ich położenie odpowiadało położeniu w dacie podanej przez użytkownika. Poniżej prezentujemy kod z pliku main.js:

```
function jumpDate(){

    var boxDate=document.getElementById('skokData').value;

    var dates=boxDate.split('-');
    if ( dates.length !=3 || dates[0]<0 || dates[1]<1 || dates[1]>12 ||
dates[2]<1 || dates[2]>31){
        alert('Nieprawidłowy format daty');
        setDefaultDate();
        return;
    }
    let inputDate = new Date(dates[0], dates[1]-1, dates[2], 12, 0, 0,
0).getTime();
    let skok = (inputDate - date_mili)/(24*60*60*1000);

    time+=skok;
    date_mili+=skok*(24*60*60*1000);

    for(let item of objects)
    {
        if(item.name !== "Sun")
        {
            item.rotate+=(skok*100)*(2*Math.PI/(item.circle_time*100));
        }
    }
    draw();
    writeDate();

}
```

Informacje o planetach

W symulacji użytkownik ma możliwość podglądnięcia wybranych informacji o planetach układu Słonecznego. Planety na symulacji są stosunkowo małe, przez co kliknięcie na nie może być problematyczne. MIędzy innymi z tego powodu zdecydowaliśmy się na następujące rozwiązanie: wyświetlamy nazwy planet i “ogony” (ślady) planety w kolorze odpowiadającym kolorowi planety. Po kliknięciu w

którekolwiek miejsce na canvasie uruchamiana jest funkcja, która sprawdza czy kolor pixela na który kliknęliśmy pokrywa się z którymś z kolorów planet i jeżeli tak, to wyświetla informacje o planecie której kolor pokrywa się z kolorem kliknięgo pixela. Poniżej prezentujemy kod pochodzący z pliku main.js:

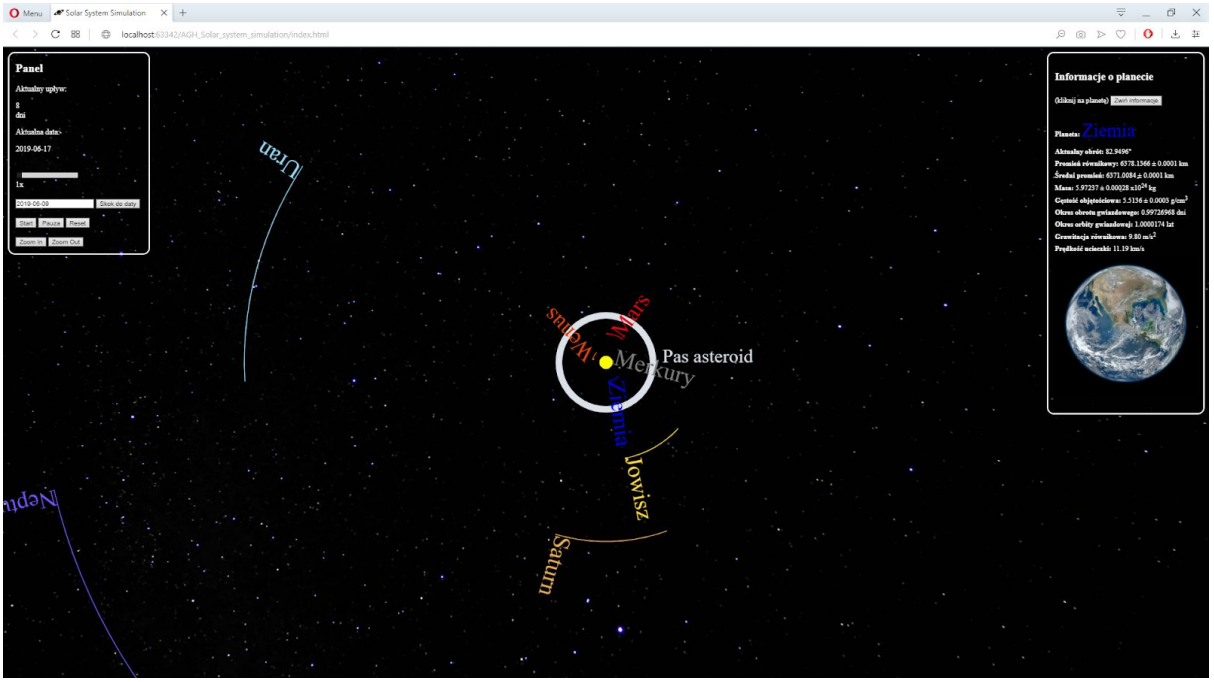
```
canvas.onclick = function(e) {
    console.log('canvas click');
    const mousePos = {
        x: e.clientX,
        y: e.clientY
    };
    //pixel pod kursorem
    const pixel = ctx.getImageData(mousePos.x, mousePos.y, 1, 1).data;

    //tworzymy HEX tego pixela
    const color =
(`#${rgbToHex(pixel[0])}${rgbToHex(pixel[1])}${rgbToHex(pixel[2])}`).toU
pperCase();


    objects.forEach(ob => {
        if (hasSameColor(color, ob)) {
            document.getElementById("info").innerHTML = ob.info;
            clicked_planet = ob;
            writeAngle();
            ever_clicked = true;
        }
    });
};
```

Przybliżanie/Oddalanie

Obie funkcje zoomin() i zoomout() to bardzo proste funkcje, które najpierw sprawdzają czy aby nie osiągnęliśmy maksymalnej wartości zmiennej scale (która odpowiada za zmianę wielkości układu) a następnie odpowiednio zmienia ten parametr.

Wynik symulacji

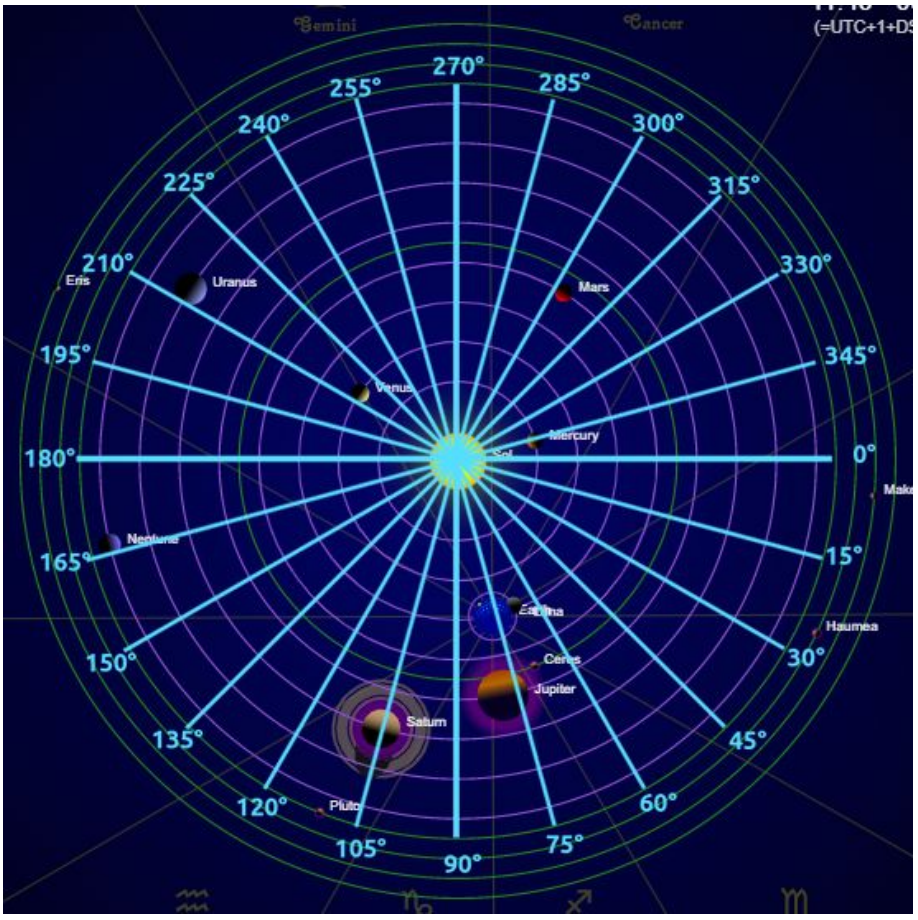
W panelu “Informacje o planecie” (po prawej stronie ekranu)

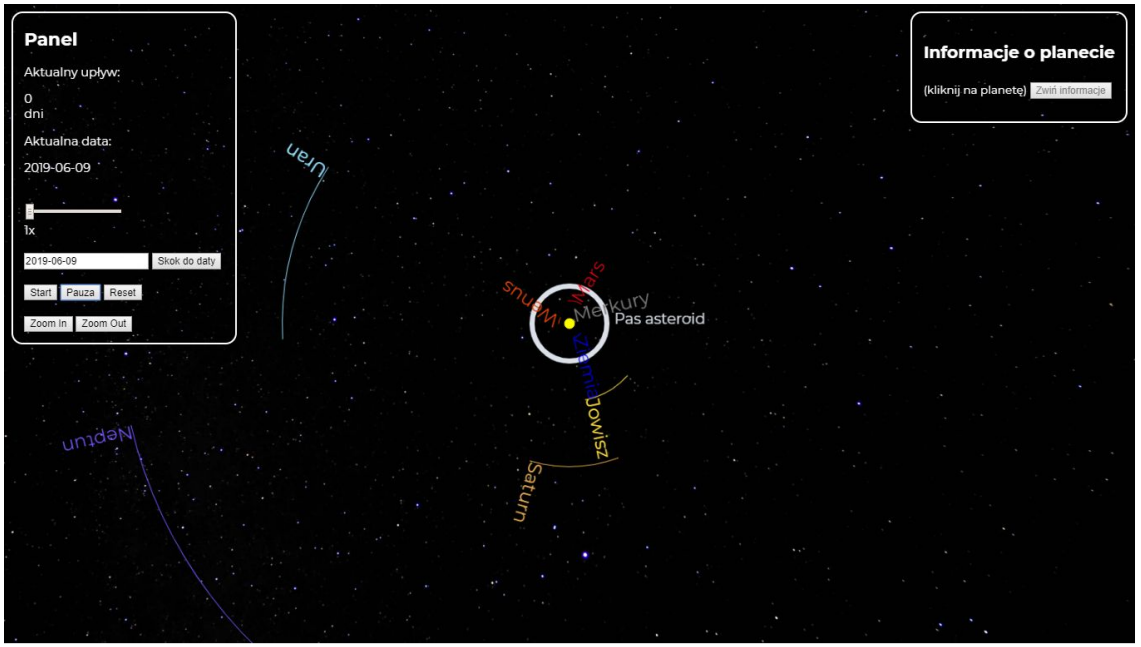


Dane liczbowe pochodzą ze strony[31]. Zdjęcia planet: Merkury[32], Wenus[33], Ziemia[34], Mars[35], Jowisz[36], Saturn[37], Uran[38], Neptun[39], Pas asteroid [40].

Naszą symulację będziemy porównywać z symulacją dostępną w internecie The Planets Today[41]. Dla lepszej czytelności odczytanych kątów na każdy obraz z symulacji nałożyliśmy siatkę z kątami.
Stan początkowy 9.06.2019:

[31] https://ssd.jpl.nasa.gov/?planet_phys_par
[32] https://solarsystem.nasa.gov/resources/771/colors-of-the-innermost-planet-view-1/?category=planets_mercury
[33] https://solarsystem.nasa.gov/resources/775/venus-computer-simulated-global-view-of-the-northern-hemisphere/?category=planets_venus
[34] https://solarsystem.nasa.gov/resources/581/earth-by-suomi-npp/?category=planets_earth
[35] https://solarsystem.nasa.gov/resources/948/hubbles-close-up-view-of-mars-dust-storm/?category=planets_mars
[36] https://solarsystem.nasa.gov/resources/2450/jupiter-marble/?category=planets_jupiter
[37] https://solarsystem.nasa.gov/resources/210/saturn-approaching-northern-summer/?category=planets_saturn
[38] https://solarsystem.nasa.gov/resources/599/uranus-as-seen-by-nasas-voyager-2/?category=planets_uranus
[39] https://solarsystem.nasa.gov/resources/611/neptune-full-disk-view/?category=planets_neptune
[40] https://ocs-pl.oktawave.com/v1/AUTH_2887234e-384a-4873-8bc5-405211db13a2/spidersweb/2017/09/nanosondy-1180x502.jpg
[41] https://www.theplanetstoday.com/index.html#

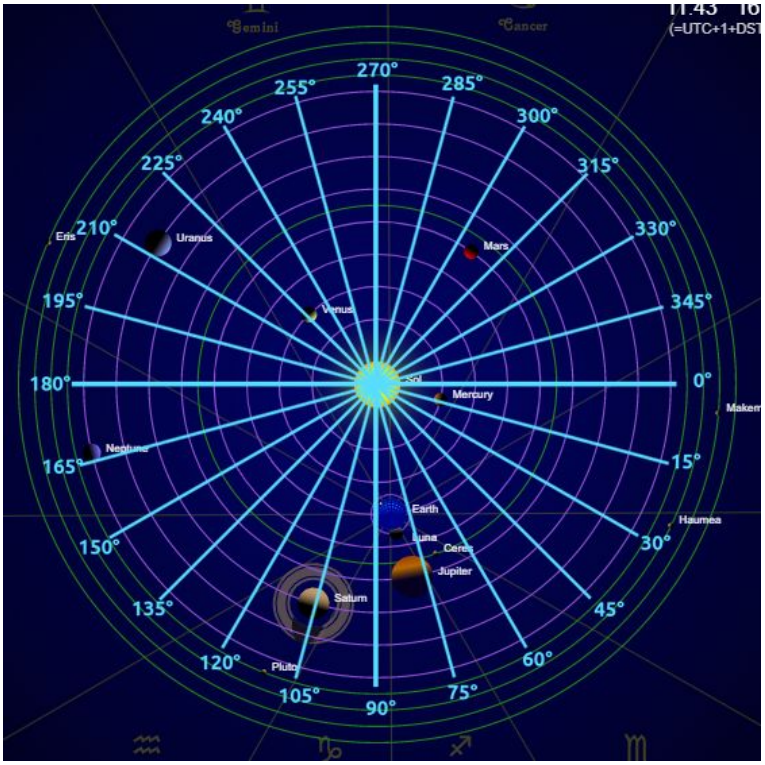


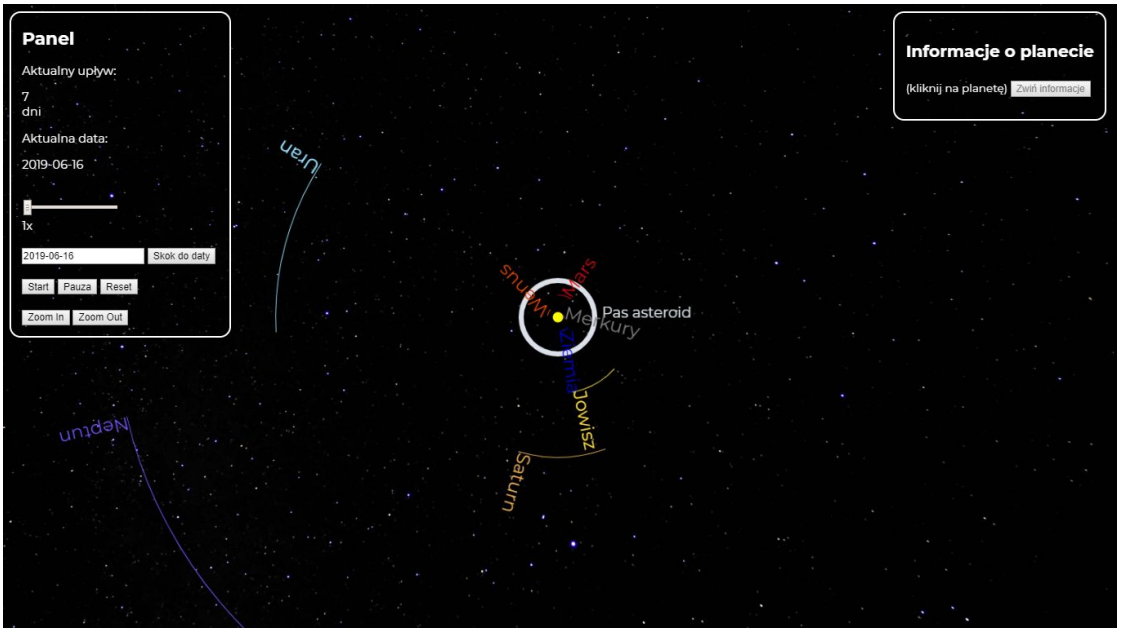


	Merkury	Wenus	Ziemia	Mars	Jowisz	Saturn	Uran	Neptun
The Planets Today	346	215	76	302	78	106	213	167
nasza	345,8246	214,891	75,9615	301,8469	77,9605	105,9463	212,892	166,9153

Przeprowadzamy symulacje dla 4 różnych odstępów czasowych. Kolejno dla: 1 tygodnia - 16.06.2019, 1 miesiąca - 9.07.2019,1 roku - 9.06.2020,10 lat - 9.06.2029

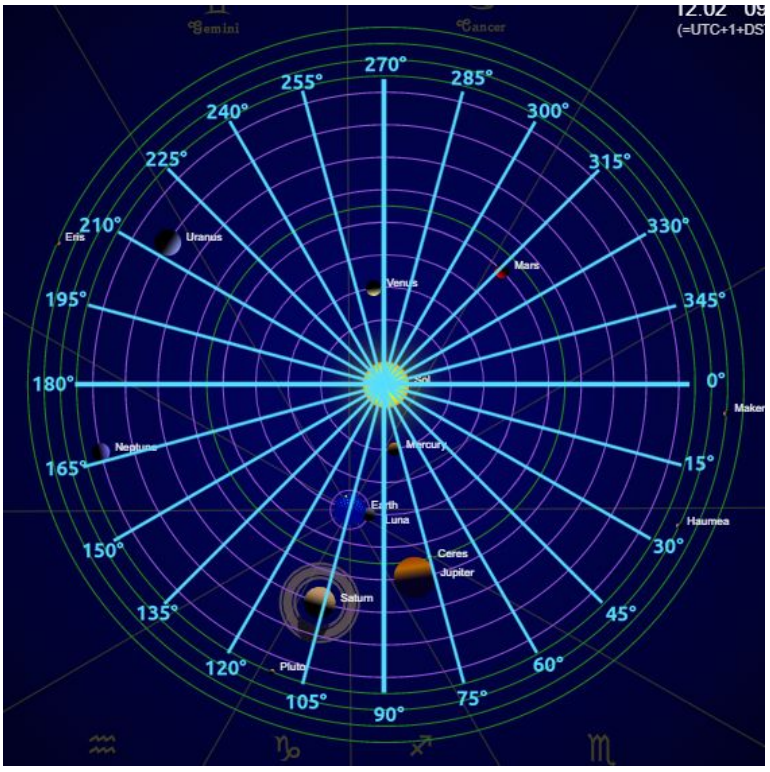
1 tygodnia - 16.06.2019

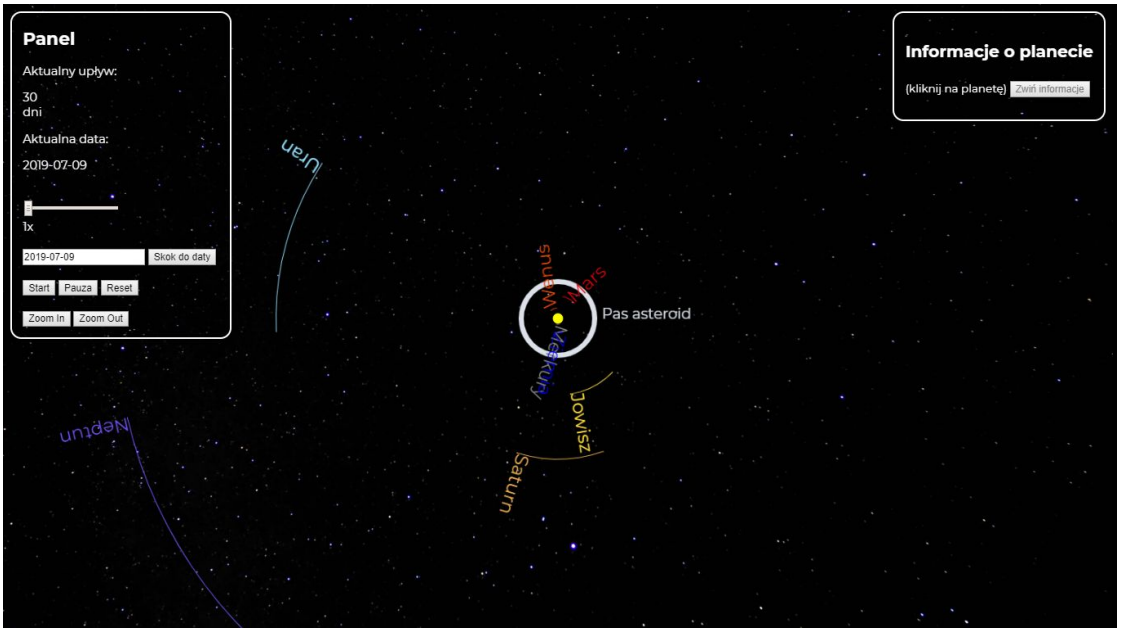




	Merkury	Wenus	Ziemia	Mars	Jowisz	Saturn	Uran	Neptun
The Planets Today	14	226	83	305	78	106	213	167
nasza	14,471	226,1059	82,8609	305,5152	78,542	106,1805	212,9741	166,9572

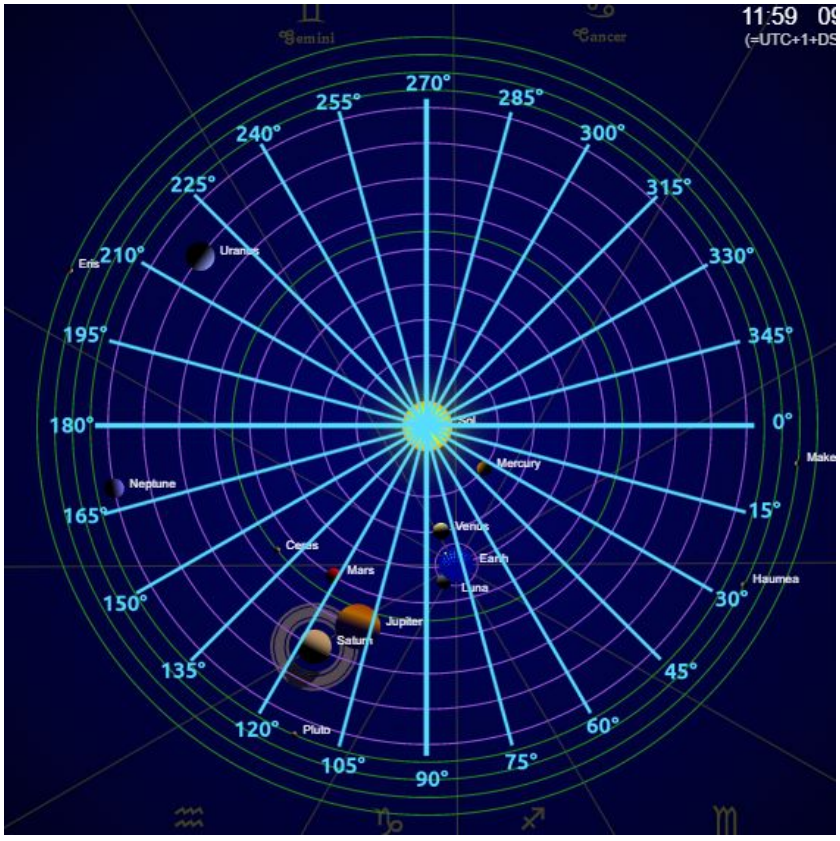
1 miesiąca - 9.07.2019

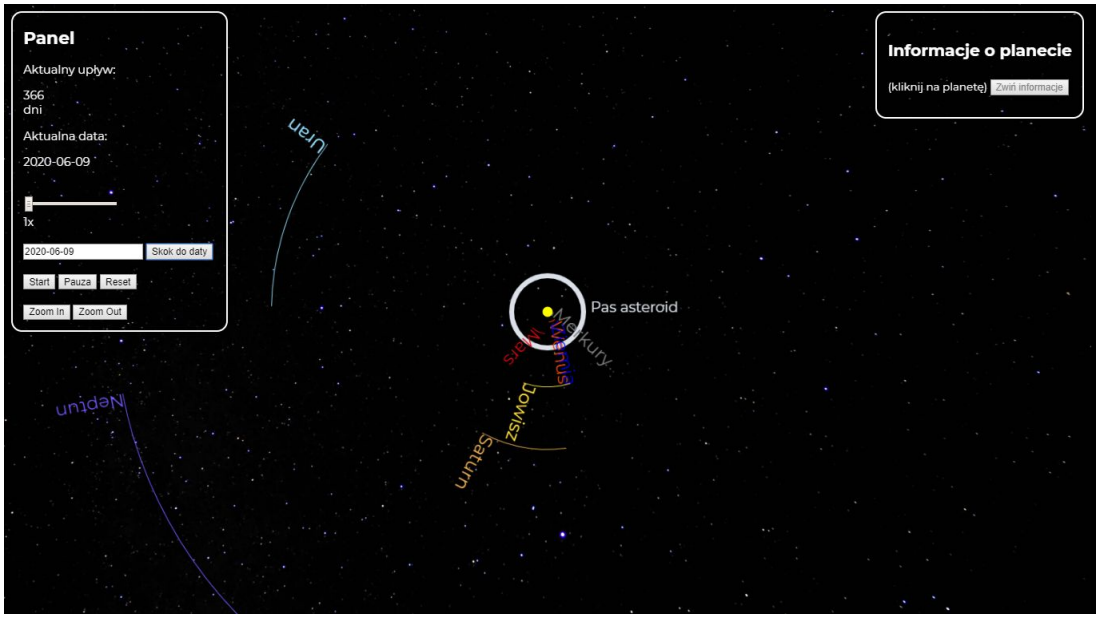




	Merkury	Wenus	Ziemia	Mars	Jowisz	Saturn	Uran	Neptun
The Planets Today	77	264	105	316	80	107	214	167
nasza	108,5951	262,9549	105,5303	317,5683	80,4528	106,9503	213,2437	167,0947

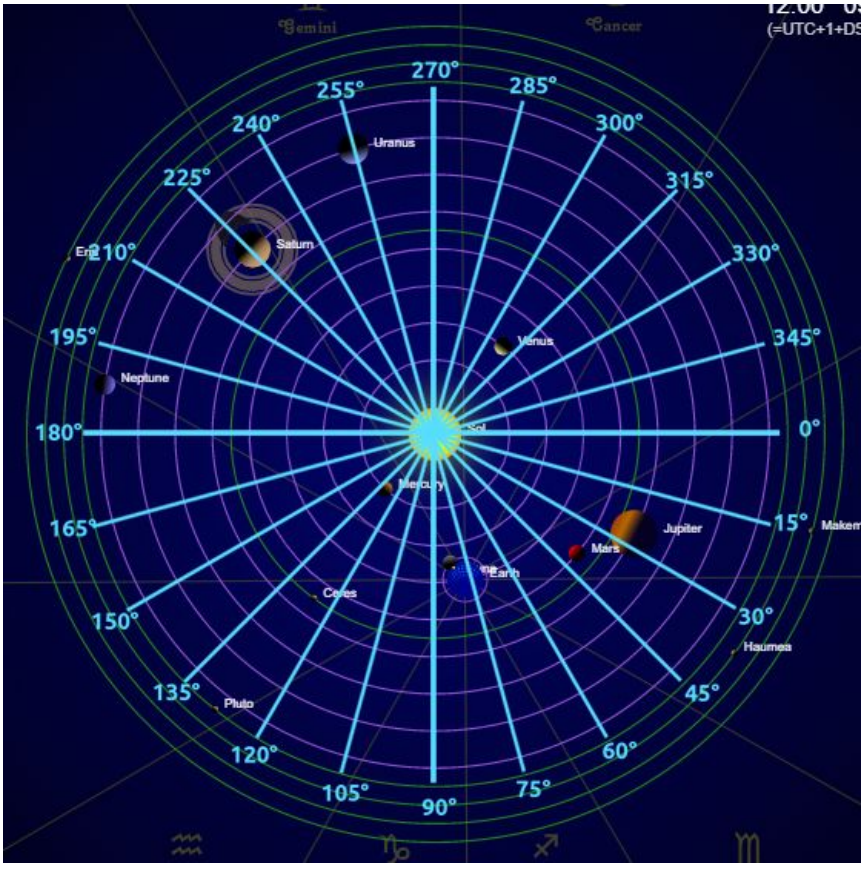
1 roku - 9.06.2020

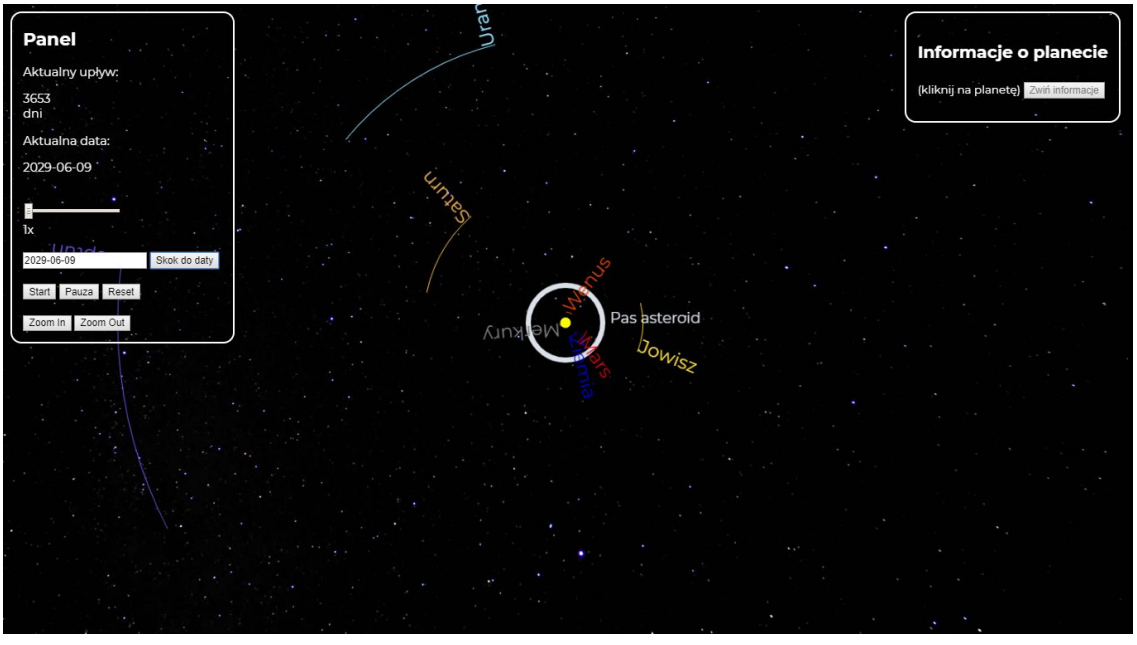




	Merkury	Wenus	Ziemia	Mars	Jowisz	Saturn	Uran	Neptun
The Planets Today	38	82	77	121	109	117	217	170
nasza	43,625	81,2702	76,7007	133,6485	108,3669	118,1959	217,1828	169,1032

10 lat - 9.06.2029





	Merkury	Wenus	Ziemia	Mars	Jowisz	Saturn	Uran	Neptun
The Planets Today	130	310	77	40	21	226	255	189
nasza	175,1818	307,4684	76,4543	56,1942	21,4437	228,2088	255,7181	188,752

Aby zobaczyć stopień rozregulowania symulacji mierzę odchylenie standardowe wg wzoru[42]:

$$\sigma = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

, gdzie :

- x_i - kąt planety według naszej symulacji
- y_i - kąt planety według symulacji The Planets Today
- n - liczba planet.

Odchylenie to można przyjąć jako miarę różnicy między naszą symulacją, a symulacją dostępną w internecie.

	Odchylenie standardowe	**Odchylenie standardowe liczone bez Merkurego i bez Marsa**
początek	0,3014134204	0,191429569
+1 tydzień	0,9201535524	0,599508565
+1 miesiąc	31,66815146	1,470342753
+1 rok	13,96123291	1,814069191
+10 lat	48,12490821	3,515625946

[42] https://pl.wikipedia.org/wiki/Odchylenie_standardowe

Wnioski

Dla skoku w przyszłość o długości jednego tygodnia rozregulowanie układu symulacji jest niewielkie. W skali kosmicznej jest to zbyt mały interwał czasowy żeby planety takie jak Jowisz, Saturn, Uran czy Neptun wykonały znaczny obrót wokół Słońca.
Dla jednego miesiąca wyniki naszych pomiarów są zbliżone do stanu faktycznego (z pominięciem Merkurego, którego przypadek zostanie objaśniony poniżej).
Przy różnicy jednego roku niedokładność symulacji Merkurego się utrzymuje, oraz dodatkowo możemy zaobserwować błędy w przewidywanym położeniu Marsa.
Przy interwale 10 lat największe różnice od stanów faktycznych nadal utrzymują Merkury i Mars.
Niedokładność symulacji konkretnie tych dwóch planet jest najprawdopodobniej spowodowana tym, że mimośród orbit planet zostały pominięte. Jeżeli popatrzymy na tabelę, zestawiającej mimosrody kolejnych planet, umieszczoną w tej pracy możemy zobaczyć, że Merkury i Mars są planetami którym mimośrody są największymi spośród badanych planet Układu Słonecznego. Kurs Merkurego, jako że ekscentryczność jego orbity jest największa, najwcześniej (już przy skoku o 1 miesiąc) zaczął odbiegać od faktycznego.
Jednakże, jeżeli popatrzymy na odchylenie standardowe nie uwzględniające pomiarów dla Merkurego i Marsa, widać że nie są one duże. Przy 10 latach odchylenie standardowe wynosi ok 4 stopnie. Biorąc pod uwagę, że eliptyczność orbit innych planet też została pominięta, należy się spodziewać że wraz z upływającym czasem symulacji różnice między położeniami kolejnych planet w naszej symulacji, a ich położeniem faktycznym będzie się zwiększać. Najwcześniej będą odpadały te planety których mimosródy są największe.
Na podstawie powyższych pomiarów, obserwacji i wniosków doszliśmy do konkluzji, że symulacja planet Układu Słonecznego zaproponowaną przez nas metodą jest stosunkowo dokładnym sposobem odwzorowania stanu rzeczywistego planet.
Problemem z którym nasza symulacja sobie nie radzi, są planety, których ekscentryczności orbit są za duże, mogły aby zostać pominięte bez utraty dokładności całego układu. Należy też wziąć pod uwagę, że wraz z rosnącym czasem symulacji kolejne planety będą “wypadały” ze swoich rzeczywistych położeń, co sprawi, że z czasem symulacja ulegnie całkowitemu rozregulowaniu.
Jak pokazaliśmy powyższymi obliczeniami, do momentu 09.06.2029 symulacja jest zadowalająco dokładna dla celów nie naukowych (edukacyjna obserwacja nieba, orientacyjne położenie planet) z wyłączeniem położenia Merkurego i Marsa jako planet o największym mimośrodzie.

Dalsze kroki

Jest wiele różnych sposobów na rozwinięcie tej symulacji. Opisana wyżej symulacja i jej algorytm (sekcja Proponowany model) zawierają uproszczenia, które w trakcie rozwoju należy zniwelować. Następny algorytm należałoby oprzeć o problem 2 i 3 ciał opisany w publikacji[43]. Proponowane rozwiązanie wymaga zbadania oddziaływań między planetami i wpływu tych oddziaływań na ruch obiektów niebieskich. Następną rzeczą do skorygowania są orbity planet z owalnych na elipsy. Można by w tym przypadku użyć metody wykorzystującej elementy orbitalne, ponieważ jest to stosunkowo dokładna metoda. Zaproponowane wyżej zmiany prowadzą do zmienienia podejścia prezentowania symulacji. Z płaskiego widoku 2D należałoby zrobić bardziej skomplikowany widok 3D ze swobodną kamerą umożliwiającą przybliżanie na dowolny obiekt i mającą zakres widzenia $360°$. Możnaby tutaj użyć biblioteki three.js oferującej widoki 3D. Zaproponowane zmiany powinny skutkować zwiększeniu atrakcyjności symulacji oraz znacznemu polepszeniu dokładności wyników.

[43] Solar System Dynamics / C.D. Murray, S.F. Dermott., rozdział 2-3