WEB で見て触る太陽系シミュレータ

深井友貴 [*1]

2021 年 9 月 29 日

[*1] 大阪工業大学 情報科学部情報メディア学科

目次

1　はじめに

1.1　概要

本研究では実際の太陽系を実スケール比で WEB 上で再現し、実際に触って動かすことができるシミュレータを作成した。このシミュレータは惑星の軌道を正確にするために、ルンゲクッタ法を用いて運動方程式を解いている。初期値は正確な値を用い、それぞれの惑星間で働く引力を加味して計算した。また、スケール比を理解できるように拡大・縮小が自由にできるように工夫したほか、マウスによる画面操作、注視点やカメラの視点変更、月日変更などの機能も持たせている。

1.2　研究目的

本研究では宇宙について知りたいもしくは興味がある人という人に向けて太陽系とはどんなものか日食月食とはどんなものか自分でシミュレータを操作し感じてもらうことを目的としてる。本シミュレータは太陽系表示ページ、軌道表示ページ、日食月食表示ページの 3 ページで構成されており、全てのページにおいて惑星ごとの間隔、惑星の大きさも実際のサイズ感などを理解してもらうために実スケール比で作成している。こうすることでシミュレータで太陽系や日食月食のような現象を忠実に再現できる。太陽系表示ページ、惑星軌道表示ページでは $250km$ を 1 ピクセルとして、日食月食表示ページでは影の表現の兼ね合いにより $1000km$ を 1 ピクセルとして表示している。利用者にはこのシミュレータを通して太陽系内惑星がそれぞれどれだけ離れているのか、惑星の公転の様子、太陽系内惑星の軌道が水平面に対してどれだけ傾いているのかなどを実際に経験することができる。日食月食ページで光や影の表現を用いて作成した理由は日食や月食が起こった際の地球にかかる影のかかり方であったり日食月食のような現象は毎回起きるわけではなくどういった条件で、どのような惑星の並び方でそれが起こるのか [1] これまで観測された、観測されると予測されている日食月食のシミュレーションを通して知ってもらうことを目的としている。また、シミュレータには描画間隔の変更ができるようにしてあるこれにより影がどう変化していくかというものを知ってもらうことも目的である。

1.3　文書構成

第 2 章では本研究に用いるルンゲクッタ法、初期条件の設定、運動方程式の解説。
第 3 章ではシミュレータ作成上の工夫。
第 4 章ではシミュレータの使い方の解説、シミュレータの他ソフトとの違いついて述べる。
第 5 章では本研究の結論と展望を述べる。

2　惑星の軌道の計算方法

この章では惑星間の万有引力を求める際に使った運動方程式、Runge-Kutta 法、初期条件の設定について解説する。

2.1　運動方程式

惑星間には万有引力が働いている。ある質量 M の惑星が質量 m の惑星から受ける万有引力は惑星間の距離を r とし、万有引力定数を G とおくと次式になる。

$$m\frac{d^2r}{dt^2} = -G\frac{Mm}{r^2} \tag{1}$$

本研究では長さとして km、時間として秒を単位にする。そのため $G = 6.672 \times 10^{-11}\,[\mathrm{m^3kg^{-1}s^{-2}}]$ [2] を㎞に変換し、$G = 6.672 \times 10^{-20}\,[\mathrm{㎞^3kg^{-1}s^{-2}}]$ を用いている。

しかし、(1) の式は 2 体間の万有引力である。本研究ではそれぞれの惑星の万有引力を計算する際太陽系内にある惑星それぞれから受ける万有引力を加味して計算する必要があるためある質量 M の惑星を地球とした時、地球にかかる万有引力の計算は次式になる。

$$M\frac{d^2r_M}{dt^2} = \sum_{n=1}^{N}(-G\frac{Mm_n}{{r_n}^2}) \tag{2}$$

r_M は太陽を原点とした時 M の位置であり、n は地球を除いた太陽系内惑星と月を示すラベルになっている。また、r_n は地球と n とラベルされた惑星間の距離である。本研究ではこの万有引力の向きを図 1 のように x,y,z 成分に分けてそれぞれ計算している。

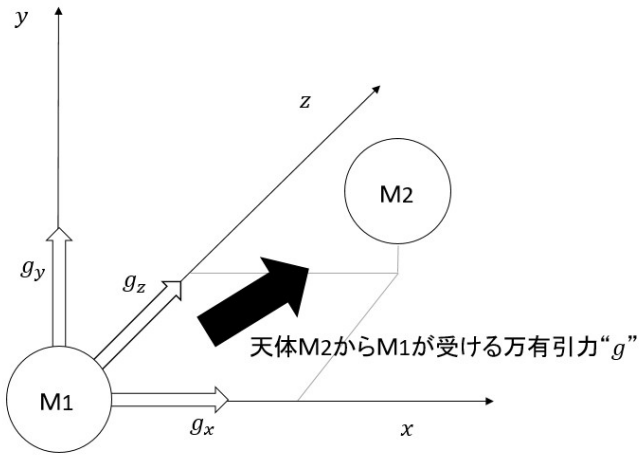


図 1　万有引力を x,y,z 軸方向に分けている

惑星間の万有引力を考える時、万有引力を受ける惑星と与える惑星との距離によって万有引力の値は変化するわけである。万有引力を与える惑星と受ける惑星の距離を惑星の位置座標から求める必要がある。例えば万有引力を受ける惑星を i とし与える惑星を j と置いた場合 (2) 式を x,y,z 成分に分けて書くと次のようになる。

$$\frac{d^2x_i}{dt^2} = -G\sum_{j=1}^{N}\frac{M_j(x_i - x_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{3}$$

$$\frac{d^2y_i}{dt^2} = -G\sum_{j=1}^{N}\frac{M_j(y_i - y_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{4}$$

$$\frac{d^2z_i}{dt^2} = -G\sum_{j=1}^{N}\frac{M_j(z_i - z_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{5}$$

これらを本研究では用いて惑星の位置を求めている。

2.2 Runge-Kutta 法

本研究では個々の惑星の位置を運動方程式を用いて導出する際 Runge-Kutta 法を用いて計算している。Runge-Kutta 法の中で最も簡易的なのが 1 次の Runge-Kutta 法である。ある $y' = f(x), y_n, x_n$ が与えられており、y_{n+1} を求めたい時、h を刻み幅、$f'(x_n, y_n)$ が点 y_n, x_n での傾きと置くと、y_{n+1} は $y_{n+1} = y_n + hf'(x_n, y_n)$ で表される。これはオイラー法と同じ方法である。

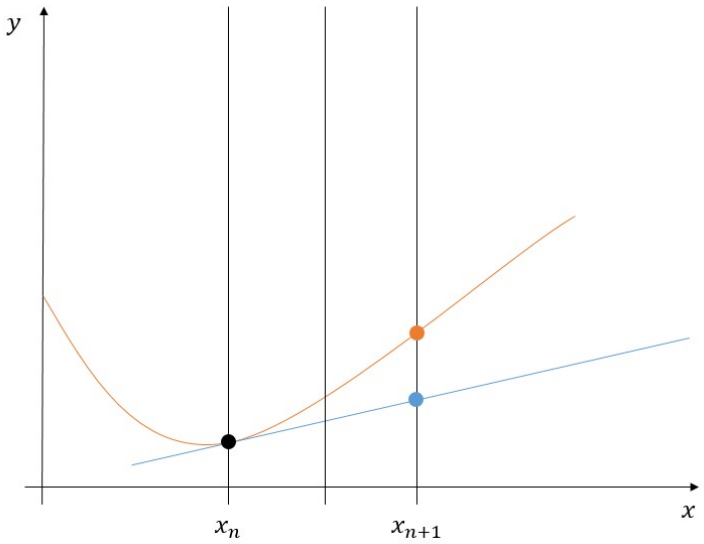


図 2　オイラー法の図:青線は傾き, オレンジ線は $y' = f(x)$ の傾きをもつグラフ

しかし、この計算方法は図 2 から分かるようにある点 (x_n, y_n) の点での傾き $f'(x_x, y_y)$ のまま h 進んだ次の点 (x_{n+1}, y_{n+1}) を求めるため精度が悪いわけである。そこで本研究では 4 次の Runge-Kutta 法を用いて計算をしている。

2.2.1　4 次の Runge-Kutta 法

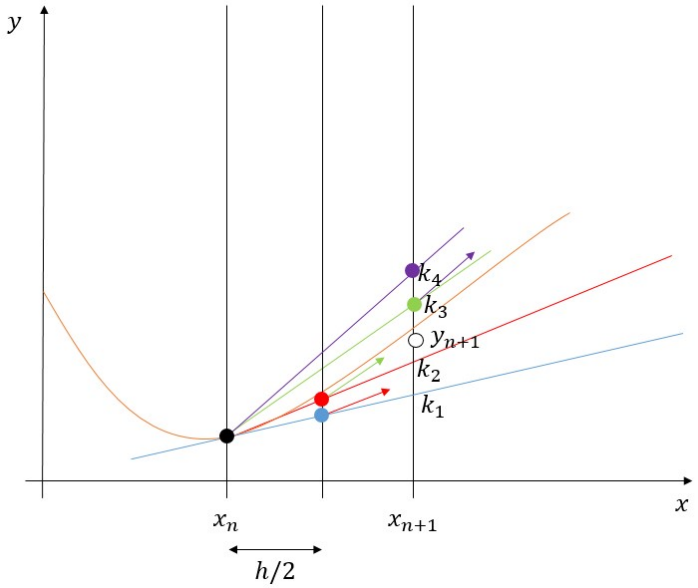


図 3　4 次の Runge-Kutta 法の図

オレンジ線は $y' = f(x)$ の傾きをもつグラフ、青線は黒点での傾き、赤線は青点での傾きで黒点から引いたもの、黄緑線は赤点での傾きで黒点から引いたもの、紫線は黄緑点での傾きで黒点から引いたものである。
4 次の Runge-Kutta 法は図 3 のように次の点を求める 1 ステップを 4 回に分けて予測し、その加重平均を取るというものである。k_1, k_2, k_3, k_4 はそれぞれの線と x_{n+1} の交点である。k_1, k_2, k_3, k_4 は次の式で与えられる。

$$k_1 = hf(x_n, y_n) \tag{6}$$

$$k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{7}$$

$$k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{8}$$

$$k_4 = hf(x_n+, y_n + k_3) \tag{9}$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{10}$$

さて、Runge-Kutta 法を用いて運動方程式を計算するわけだが Runge-Kutta 法は 1 階の微分方程式に用いる解法である。
しかし、実際に解く式は (3),(4),(5) から分かるように 2 階の微分方程式である。
その為 (3),(4),(5) をそれぞれ 1 階の微分方程式に分け 1 階の微分方程式を 2 回解くことで

2 階の微分方程式を解くことにする。
(3),(4),(5) を 1 階の微分方程式の連立に分解すると次のようになる。

$$\frac{dx_i}{dt} = v_{ix} \tag{11}$$

$$\frac{dv_{ix}}{dt} = -G\sum_{j=1}^{N} \frac{M_j(x_i - x_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{12}$$

$$\frac{dy_i}{dt} = v_{iy} \tag{13}$$

$$\frac{dv_{iy}}{dt} = -G\sum_{j=1}^{N} \frac{M_j(y_i - y_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{14}$$

$$\frac{dz_i}{dt} = v_{iz} \tag{15}$$

$$\frac{dv_{iz}}{dt} = -G\sum_{j=1}^{N} \frac{M_j(z_i - z_j)}{((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)^{3/2}} \tag{16}$$

これらの式をそれぞれ Runge-Kutta 法で解けば求めることができる。

2.3 初期条件

本研究では実際の動きに近いシミュレータの作成を目指しているため惑星それぞれの初期条件も実際の値を用いている。
以下に用いた初期条件の詳細な数値を記す。

記号	数値
G	$6.672 \times 10^{-20}\,[\mathrm{km^3 kg^{-1} s^{-2}}]$

表 1　万有引力定数

惑星名	質量 [kg]
太陽	1.9891×10^{30}
水星	3.301×10^{23}
金星	4.869×10^{24}
地球	5.972×10^{24}
火星	6.4171×10^{23}
木星	1.8986×10^{27}
土星	5.688×10^{26}
天王星	8.686×10^{25}
海王星	1.02413×10^{26}

表 2　惑星の質量

惑星名	x 座標 ([km])	y 座標 ([km])	z 座標 ([km])
太陽	0	0	0
水星	$-2.103286397832571 \times 10^{7}$	$-3.516637394080386 \times 10^{6}$	$-6.710193692959487 \times 10^{7}$
金星	$-1.074857931778700 \times 10^{8}$	$6.135028341736382 \times 10^{6}$	$-4.061819473582936 \times 10^{6}$
地球	$-2.519117146487857 \times 10^{7}$	$-2.480786407455802 \times 10^{7}$	$1.442326207767400 \times 10^{7}$
火星	$2.080148120849481 \times 10^{8}$	$-5.202972607572339 \times 10^{6}$	$-3.838308368167545 \times 10^{6}$
木星	$5.989289216342494 \times 10^{8}$	$-1.525670207775742 \times 10^{7}$	$4.384272878534235 \times 10^{8}$
土星	$9.587260943633295 \times 10^{8}$	$-5.524484791056234 \times 10^{7}$	$9.818699120519972 \times 10^{8}$
天王星	$2.158794456818467 \times 10^{9}$	$-3.564780312815702 \times 10^{7}$	$-2.055520449992805 \times 10^{9}$
海王星	$2.514873317825120 \times 10^{9}$	$1.901521508775187 \times 10^{7}$	$-3.739542713225744 \times 10^{9}$
月 (衛星)	$-2.550882170617361 \times 10^{7}$	$1.174795900236070 \times 10^{4}$	$1.439907376005011 \times 10^{8}$

表 3　惑星の位置

値は NASA の「HorizonSystem」[3] から 2000 年 1 月 1 日の値を引用している

惑星名	V_x([km/s])	V_y([km/s])	V_z([km/s])
太陽	0	0	0
水星	$3.863146297069179 \times 10^{1}$	-4.610175970896692	$-1.222519918902008 \times 10^{1}$
金星	2.867635714789456	$-7.738623575175794 \times 10^{-1}$	$-3.509456886099740 \times 10^{1}$
地球	$-2.786135740795796 \times 10^{1}$	$-2.419412502412592 \times 10^{-1}$	-5.142995022029005
火星	3.273479461672435	$2.770067921014707 \times 10^{-1}$	2.635905955106879
木星	-5.923461587315284	$-1.113297197816738 \times 10^{-1}$	$1.122781591210963 \times 10^{1}$
土星	-5.450409753233657	$-6.433857772279605 \times 10^{-2}$	6.803453118225864
天王星	6.616124463122162	$-2.848534607757223 \times 10^{-1}$	4.691831989492469
海王星	6.444278564186581	$-4.085691567564843 \times 10^{-1}$	3.140321200197810
月 (衛星)	$-2.730057034184695 \times 10^{1}$	$-2.435805832850177 \times 10^{-1}$	-5.942927282163541

表 4　惑星の初速度

値は NASA の「HorizonSystem」 [3] から 2000 年 1 月 1 日の値を引用している

3　シミュレータ作成上の工夫

3.1　制作の概略と目的

3.1.1　目的

このシミュレータは宇宙に少しでも興味がある人や関心のない人にとって興味を持ってもらうため、シミュレータ側の操作性や使いづらさソフトのインストールであったりライセンスの購入などのストレスなく気楽にシミュレーションを経験してもらうことを前提にしている。その為、どの媒体にも標準でインストールされている WEB ブラウザで制作している。

3.1.2　概略

まず、このシミュレータは WEB 上で操作、閲覧することを前提として制作しているため、使用言語は HTML,CSS,JavaScript を使用している。WEB 上で 3D 物体を作ろうとすると複雑なコードを書く必要があり、容量も多く重くなるため、宇宙空間にある惑星を複数再現する場合や光や影のなどのレンダリングが必要な処理がある場合時間がかかってしまうそこで、JavaScript については WEB 上で 3D コンテンツを簡易的なコードで楽に作ることができ処理を高速化することができる THREE.js というライブラリを用いている。このライブラリを用いることで WEB ブラウザ上に地球や太陽を表示でき惑星をそれぞれ回転させ惑星の公転や自転を再現ができ、太陽系の再現ができる。また、光や影を用いることで日食月食のような宇宙空間で起こる現象などを再現することができる。2 次元表示ではなく、3 次元空間を用いた理由はこのシミュレータは利用者に自身で触って動かしてもらうことが第一の目的であり、それを通して宇宙に興味を持ってもらうことがゴールであるため視覚的に楽しめる方が興味がわきやすいのではと考えたためである。

3.2　制作の工夫点

ここではシミュレータの工夫点について説明する。

3.2.1　全ページ共通の工夫点

■工夫点 1　本シミュレータは利用者に触ってもらうことが必須条件である。そのため、画面上に操作パネルを表示する必要があったがシミュレータ部分と操作パネル部分を同時に画面上に表示してしまうとシミュレータ部分が見づらくなってしまう上見栄えも悪いと感じた。そこで操作パネル部分を現在どのサイトでも使われているハンバーガーメニューという開閉可能なメニューにし、シミュレータ部分を全画面で表示できるようにした。こうすることでシミュレータの見栄えも良くなり、画面サイズが大きいことで見える範囲が広くなりシミュレーションもしやすくなるわけである。

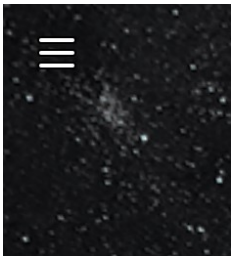


図 4　閉じているハンバーガーメニュー

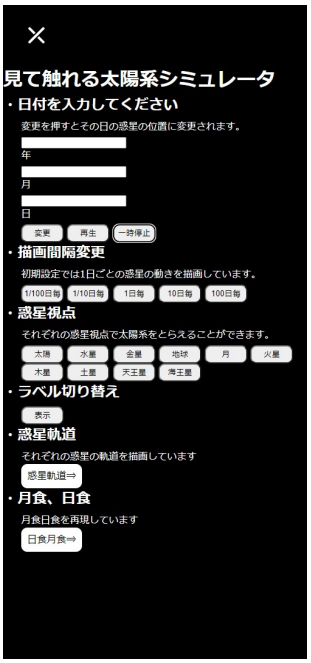


図 5　開いているハンバーガーメニュー

■工夫点 2　ブラウザは複数のウィンドウに分けたり画面サイズの変更が容易にできる特徴がある。シミュレータの利用者がウィンドウを分けて使用する場合も考え、ウィンドウサイズの変更がされてもそのサイズに自動でシミュレータのサイズも変更されるようにした。

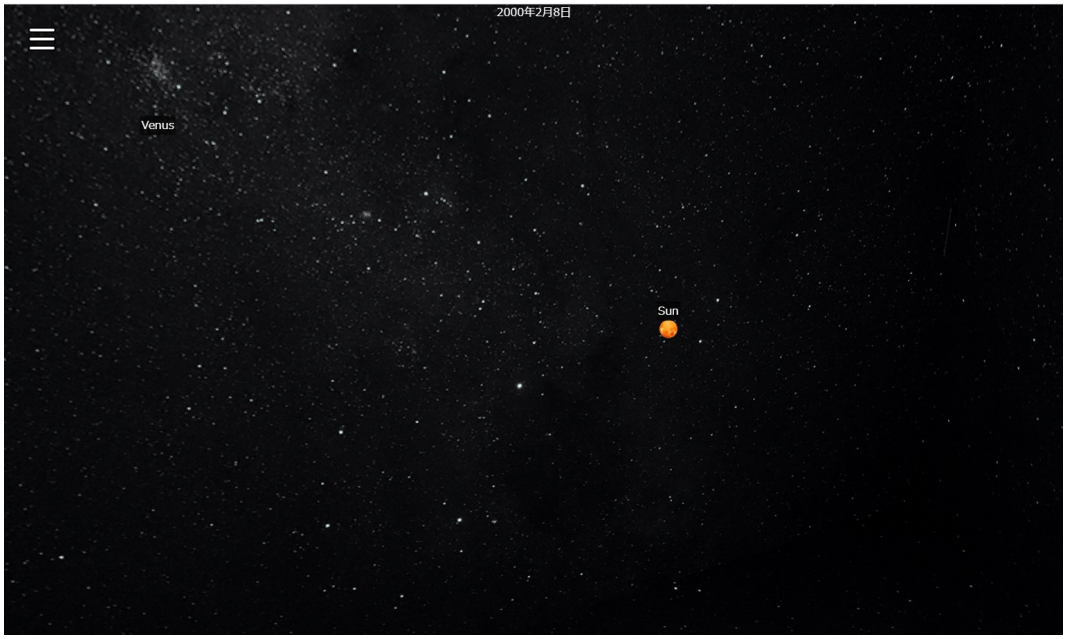


図 6　ウィンドウサイズのリサイズ設定がない場合

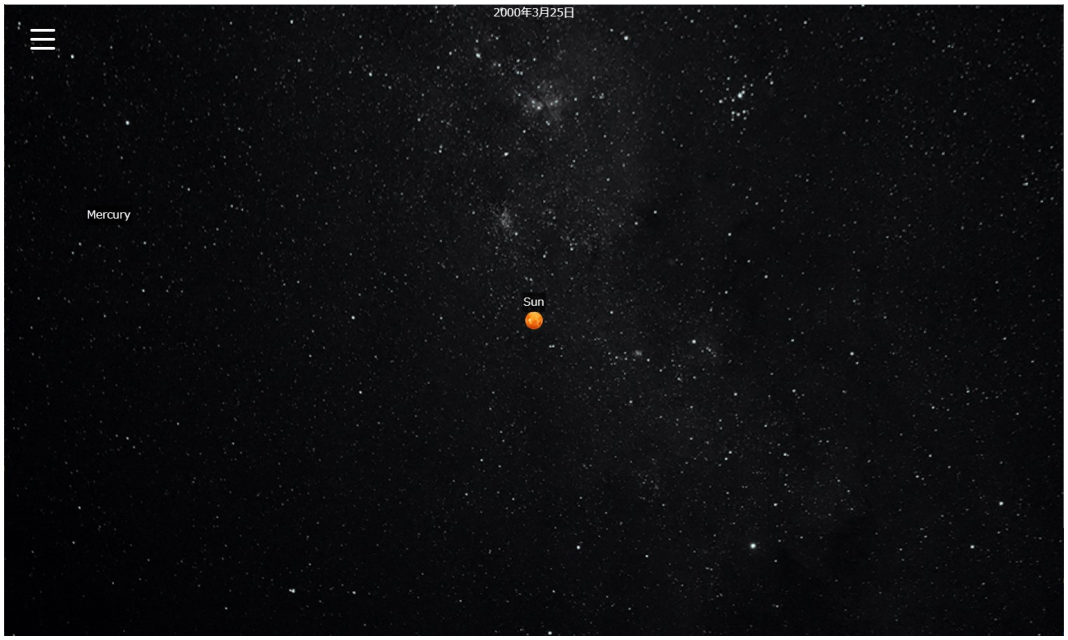


図 7　ウィンドウサイズのリサイズ設定がある場合

図 6 と図 7 を見比べてみると画面サイズのリサイズ機能が無いと画面中心に太陽が表示されていないのが図 7 から読み取れる。

画面のリサイズを受け取りその都度画面をリサイズする関数のコードを図 8 に示す。1 行目にあるコードがブラウザのウィンドウサイズが’rezize’ されたら関数’onResize’ を実行するコードである。関数’onResize’ については function 内に記述することでそれが実行される。画面がリサイズされたら画面の縦横サイズを調べそれを適応しカメラの縦横比を正し最後にアップデートするようになっている。

```
window.addEventListener('resize', onResize);

function onResize() {
    // サイズを取得
    const width = window.innerWidth;
    const height = window.innerHeight;
    // レンダラーのサイズを調整する
    renderer.setPixelRatio(window.devicePixelRatio);
    renderer.setSize(width, height);
    labelRenderer.setSize(width, height);
    // カメラのアスペクト比を正す
    camera.aspect = width / height;
    camera.updateProjectionMatrix();
}
```

図 8　リサイズのコード

■工夫点 3　惑星を表示する際少しでも宇宙を再現するために背景に星の画像を使用した。この背景画像は通常の WEB サイトで使われている背景の指定方法とは異なっている。立方体の中に太陽系を表示し、その立方体の内面に画像を張り付けている。’texturebg’ に背景画像を付けた立方体をロードし、それを’scene.background’ でシーンの背景に設定している。

```
const loaderbg = new THREE.CubeTextureLoader();
const texturebg = loaderbg.load([
      'img/bg2.png',
      'img/bg2.png',
      'img/bg2.png',
      'img/bg2.png',
      'img/bg2.png',
      'img/bg2.png',
    ]);
scene.background = texturebg;
```

図 9　背景画像の設定コード

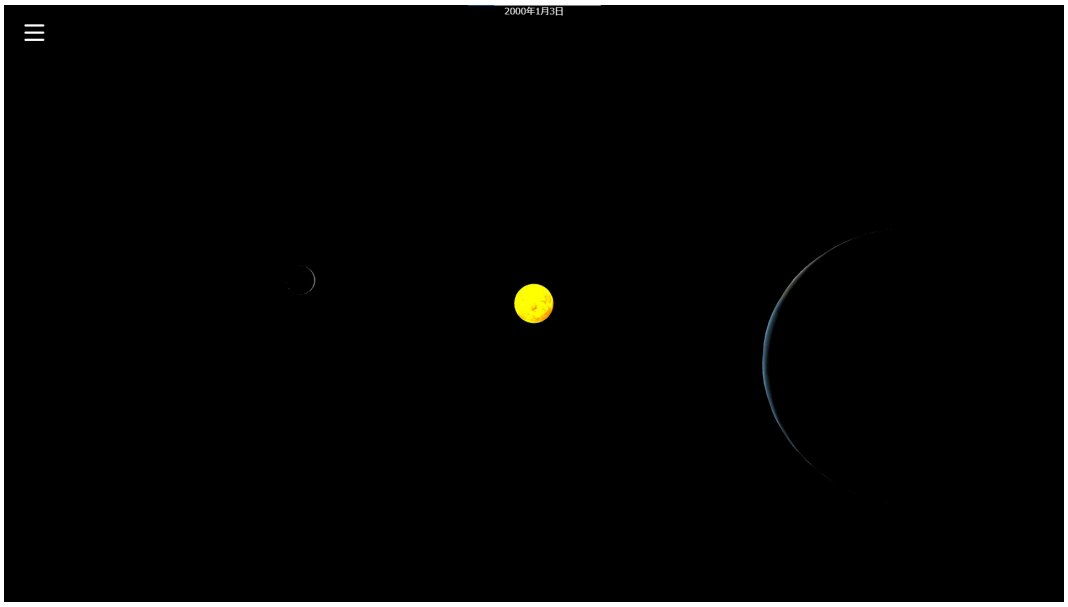


図 10　背景画像がない場合

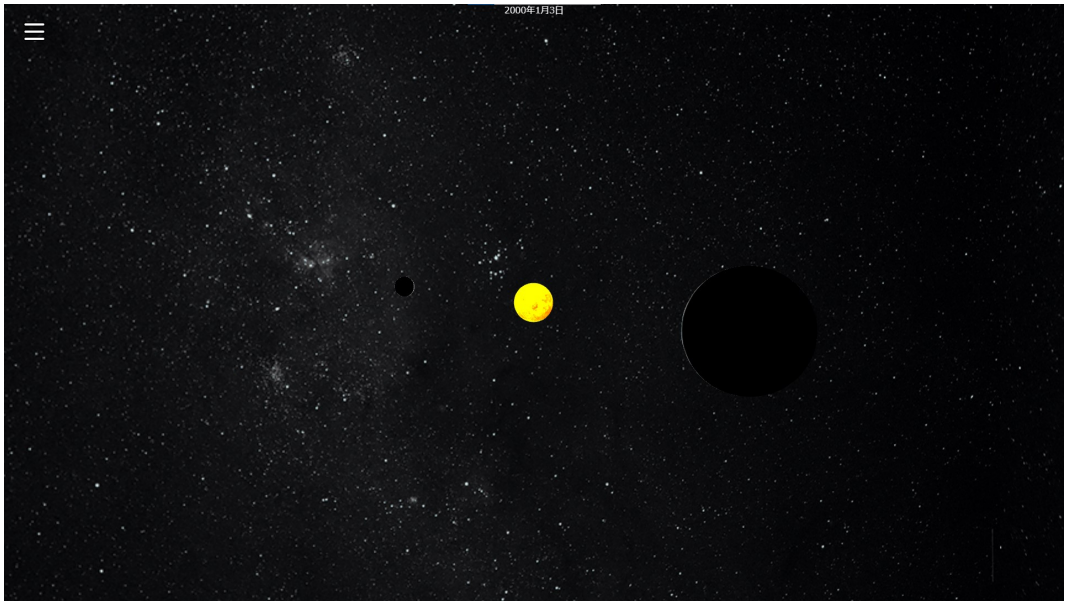


図 11　背景画像がある場合

こうすることでよりリアルに宇宙を表現できる。図 10 図 11 を見比べると背景画像がある方が宇宙がリアルに表現されているのが分かる。

■工夫点 4　太陽系ページで太陽系全体を表示しようとすると画面を縮小する必要があるため惑星の描画がされなくなってしまい、個々の惑星がどこにあるのかということが分かりにくくなっていた。THREE.js では 3 次元空間にある物体の位置座標を 2 次元に変換することができる。その機能を使い画面が縮小された場合でも物体の位置を 2 次元に変換後、惑星の名前をラベルとして表示した。こうすることで太陽系を全体で見た場合でも惑星がどこにあるのか分かり、大体の距離感を読み取ることができる。

図 12　ラベルがない場合

図 13　ラベルがある場合

図 12 と図 13 を見比べるとラベルがある方が分かりやすいのが読み取れる。

■**工夫点 5**　またラベルを常に表示している状態だと邪魔になる場合があると考えラベルの表示、非表示切り替えが出来るようにボタンをメニュー内に設置した。例えば図 11 のような画像を撮りたい場合ラベルが邪魔である。ラベルを非表示にすると見やすくなっていることが読み取れる。

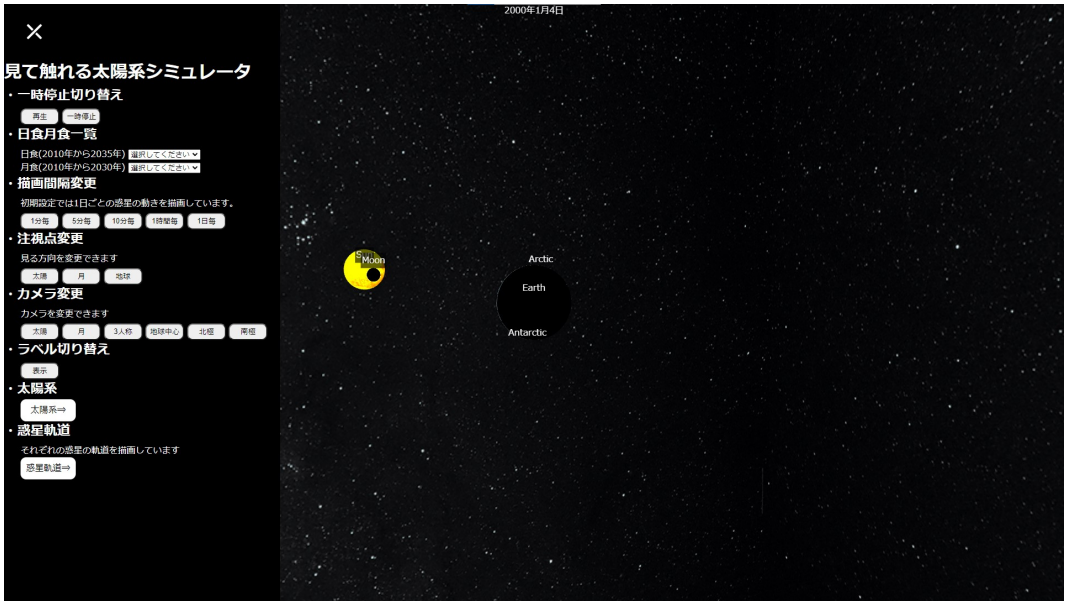


図 14　ラベル表示

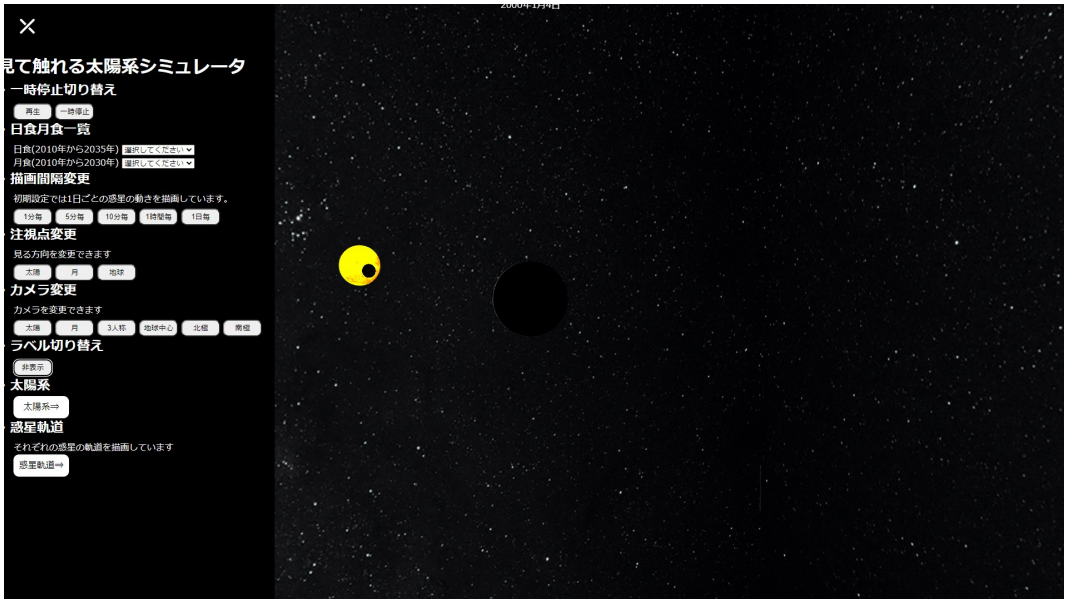


図 15　ラベル非表示

3.2.2　太陽系表示ページの工夫点

■工夫点 1　本研究では実際の太陽系を WEB ブラウザ上で再現することを目的としているため、惑星の大きさや惑星間の距離を実スケール比で再現している。ただ $1km$ を 1 ピクセルで表現しようとすると描画に時間がかかってしまうため、太陽系表示ページでは画面の 1 ピクセルを宇宙空間上での $250km$ と定義して個々の惑星を表示している。その際惑星間の距離と惑星の大きさの比も合わせる必要があるため太陽の大きさを 250 分の 1 にし、太陽に対して他の惑星の大きさを比で表すことで太陽系を再現している。

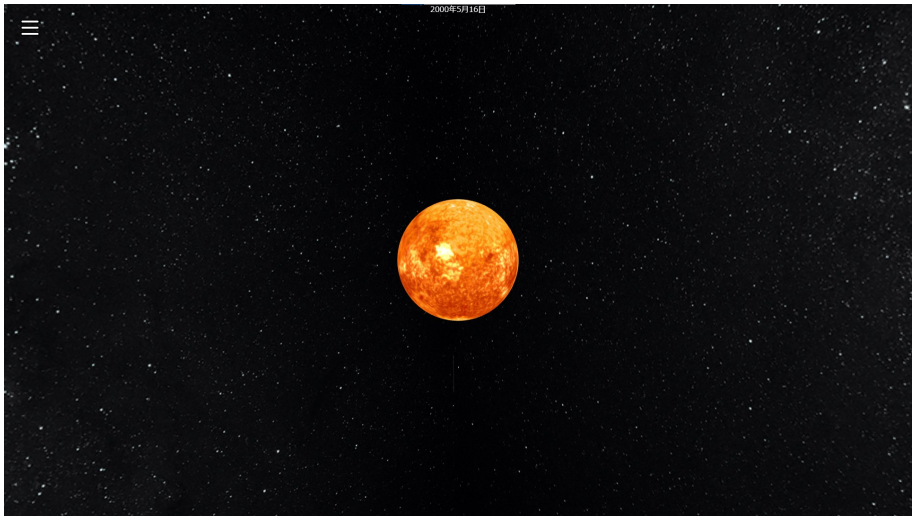


図 16　太陽の半径を $2784px$ としている

図 17　水星の半径は $(2784/1392000) * 4879.4px$ としている

■**工夫点** 2　太陽系では 1990 年から 2035 年までのシミュレーションができる。その際他の星を中心に太陽系全体を捉える事も出来るようにそれぞれの惑星視点をボタンで変更できるようにした。例として 2010 年 1 月 10 日の太陽系を表示した。

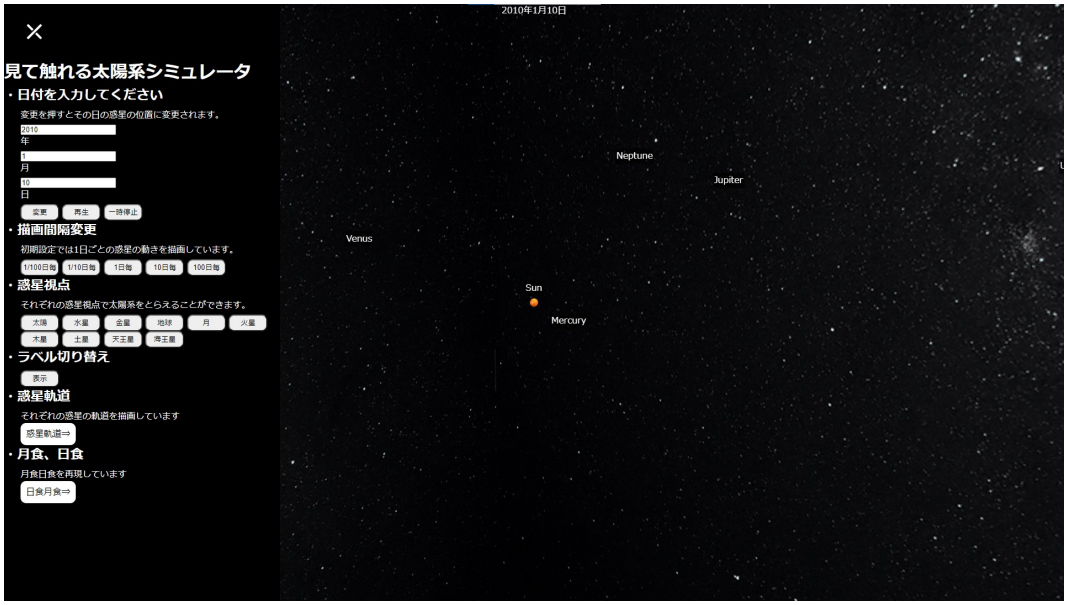


図 18　太陽系を太陽を中心として見た場合

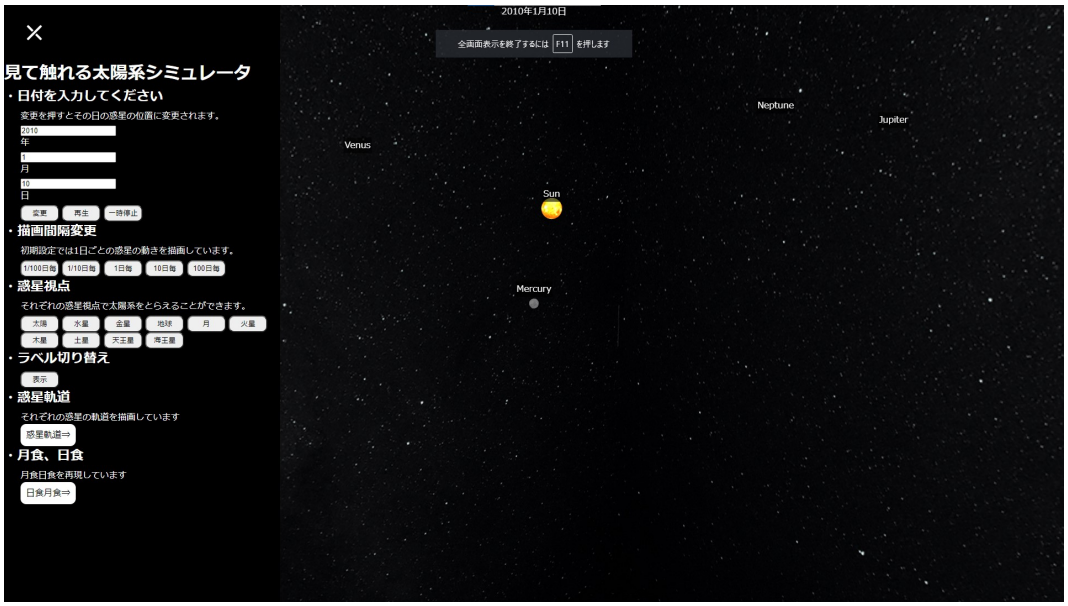


図 19　太陽系を水星を中心として見た場合

3.2.3　軌道表示ページの工夫点

■工夫点　惑星軌道表示ページではページの動作を軽くするために惑星の軌道を線で表示するだけにしている。ただ太陽だけ球体も表示し、惑星が太陽の周りをどう公転しているのか視覚的に分かりやすいようにした。また、図にあるように惑星ごとに軌道の色を変えて表示している。

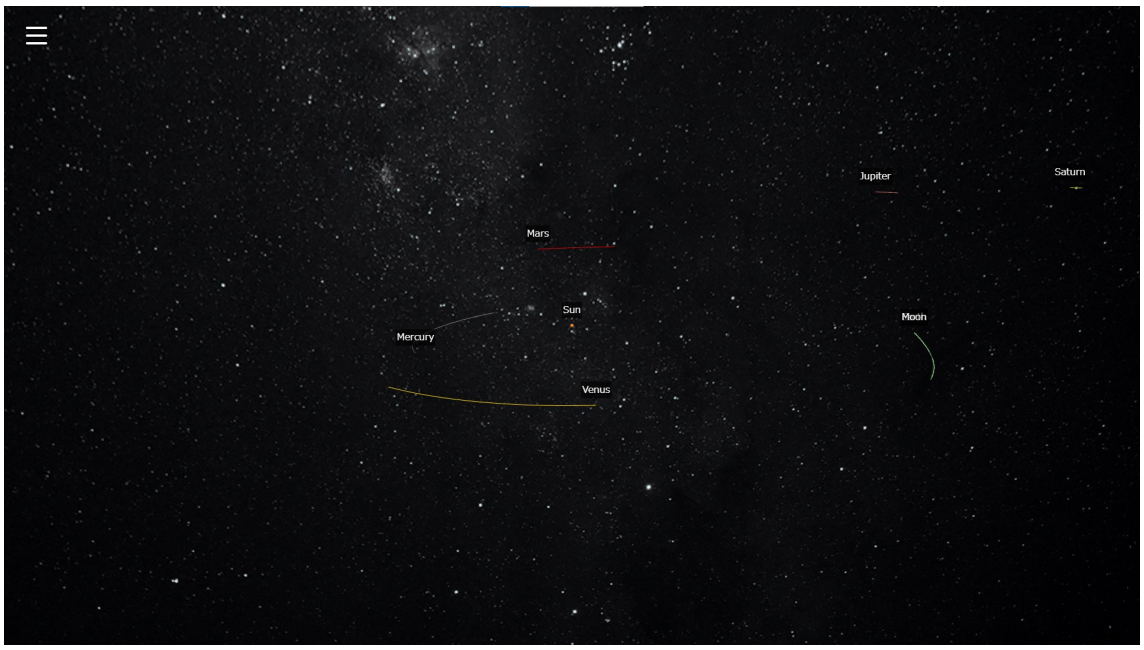


図 20　惑星軌道表示ページ

3.2.4　日食月食表示ページの工夫点

■工夫点　太陽のような恒星を再現する際 THREE.js ライブラリ内には自ら光を出す物体を作るコードはなく、光で再現する必要があった。そのため、影を作るための光とは別に太陽の周りに光を配置し、光の色を赤色に近くすることで太陽が自ら光っているように表現した。

```
const light2 = new THREE.RectAreaLight(0xffa500, 4, 20000, 20000);
```

図 21　使用した光のコード

引数は左から光の色、明るさの強度、ライトの横サイズ、ライトの縦サイズである。このシミュレータでは縦横 20000 の大きさのライトを使っており、光の色はオレンジ色にしている。

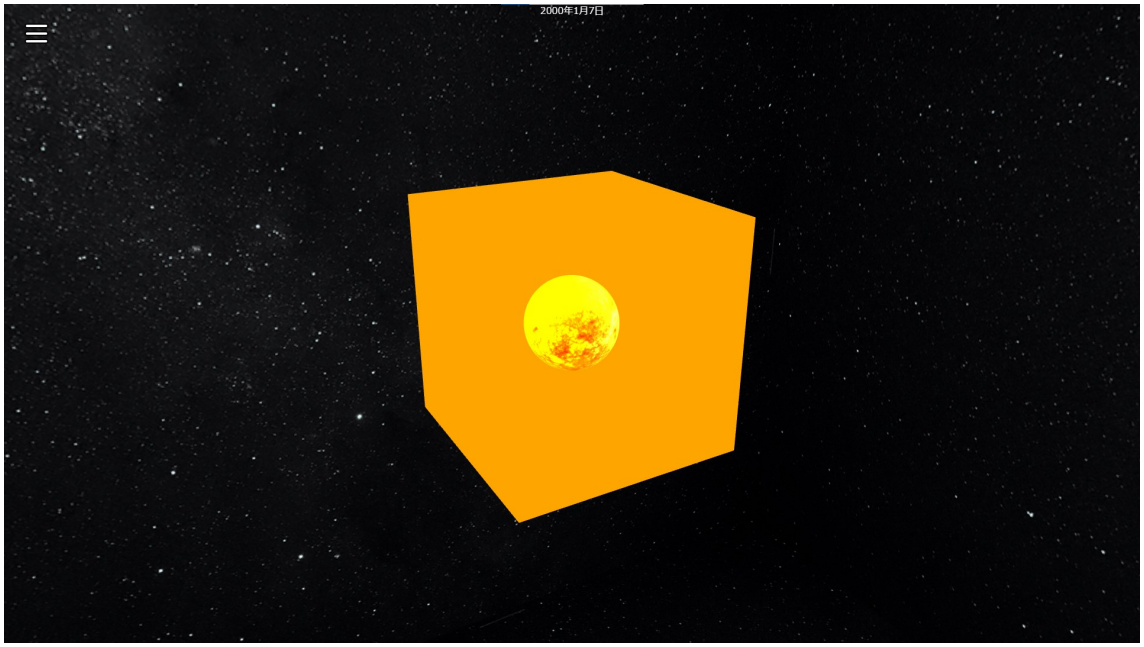


図 22　光の置き方

光は太陽の周りを囲むように置いた。こうすることで太陽だけ明るさが強くなり、自ら光っているように見えリアリティがあるのが次の画像から分かる。

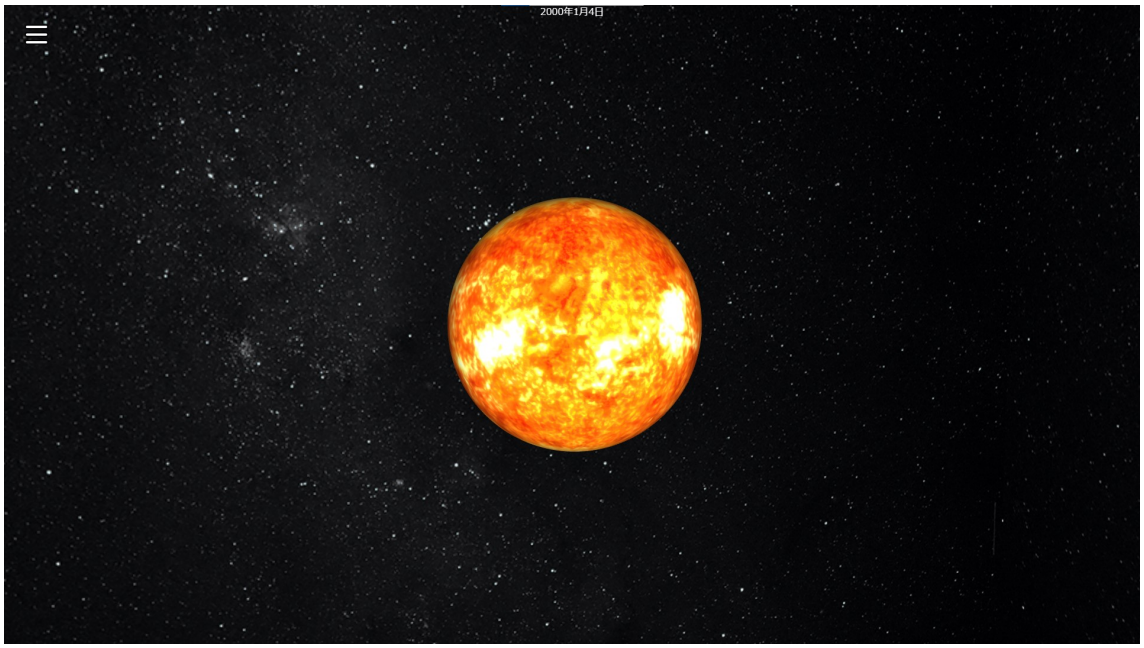


図 23　ライトを配置しない場合の太陽の見え方

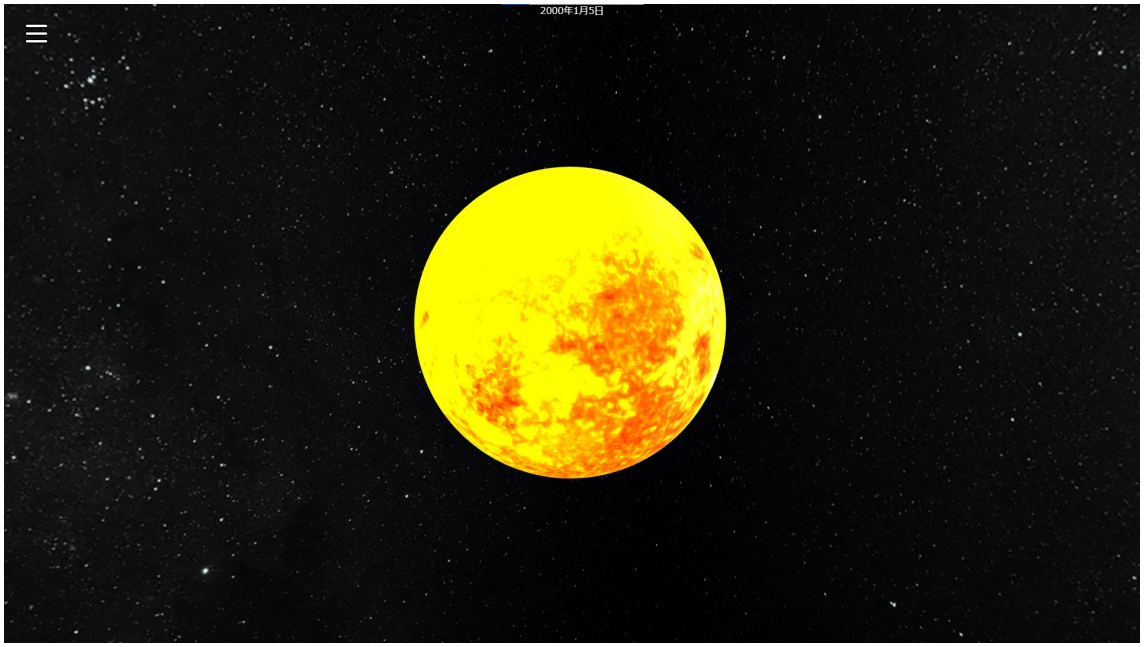


図 24　ライトを配置した場合の太陽の見え方

4　シミュレータの使い方と実例

4.1　シミュレータの構成と説明

このシミュレータは太陽系全体の画面、太陽系の軌道の画面、月食, 日食の画面の 3 部から構成されている。全てのページにおいて、マウススクロールで画面の拡大縮小、マウスによる画面の操作が出来るようになっている。

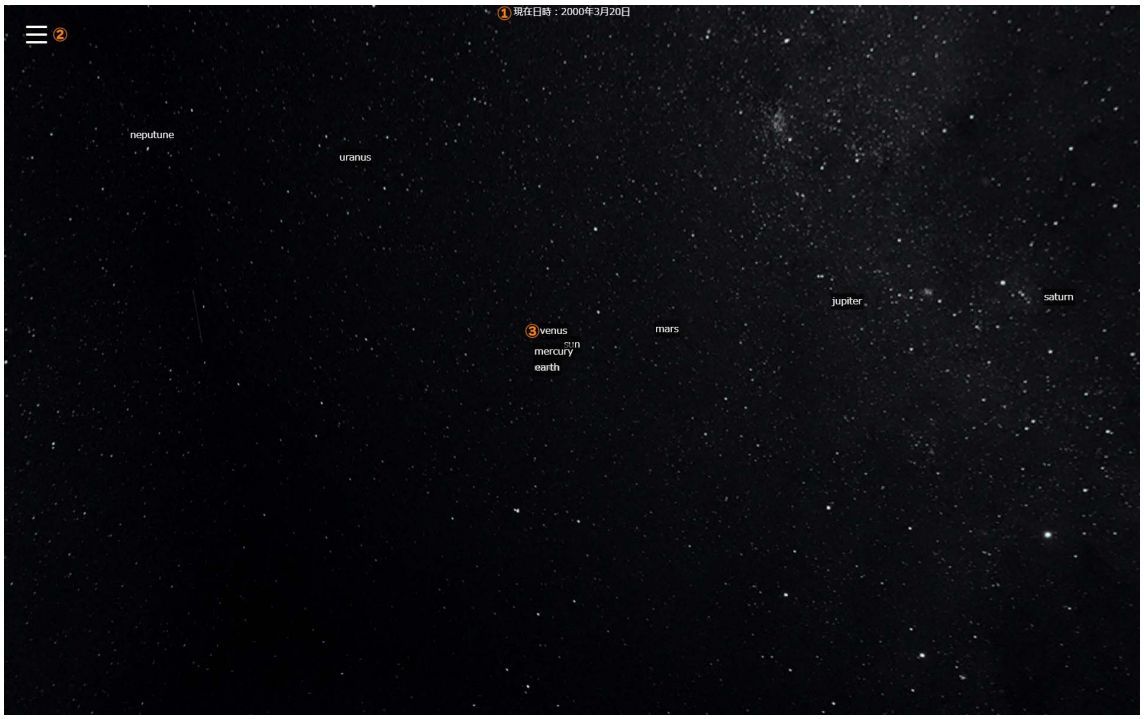


図 25　太陽系全体のページ

このページでは太陽系内の惑星を表示している。

①ここで表示されている日時の惑星の位置を表示している

②操作パネルメニューを開閉可能なメニューにし、開閉可能にすることで画面全体で太陽系を表示するようにした。

③惑星間の距離を実スケールで再現しているため、太陽系全体を移そうとすると惑星が表示されなくなってしまうため、どの惑星がなにかラベルをつけることで見えるようにしている。

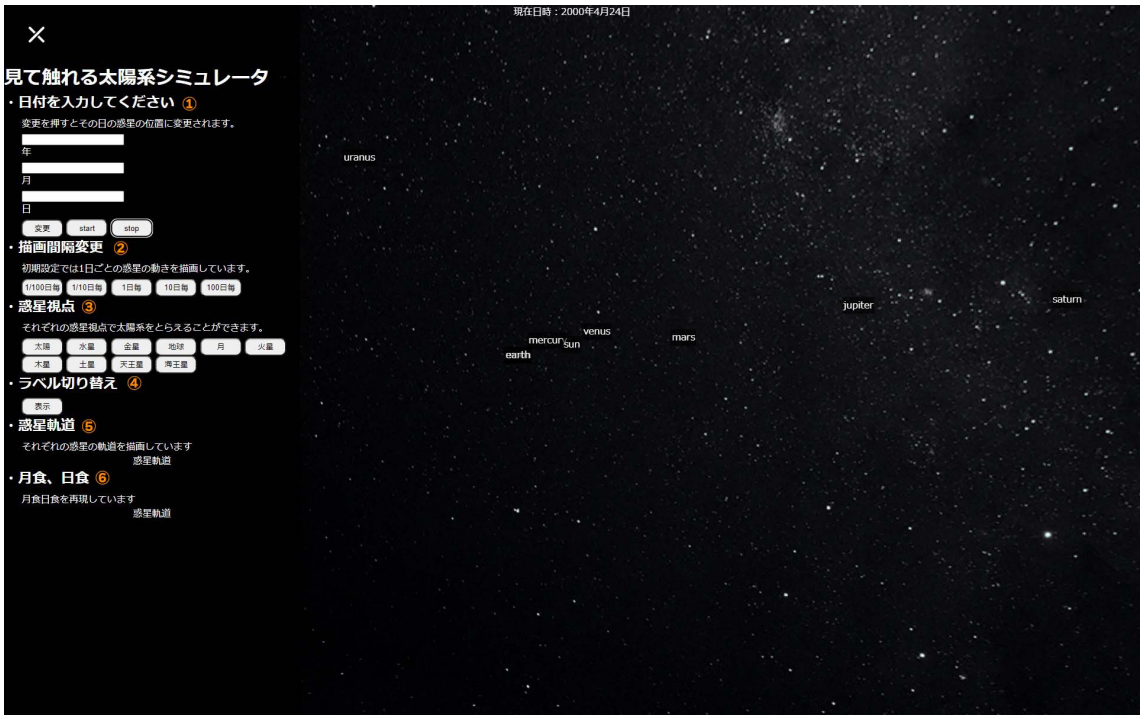


図 26　太陽系全体のページの操作パネル

①ここで日付を入力し、変更ボタンを押すことでその日の惑星の位置に変更することができる。また、start ボタンと stop ボタンはシミュレータを一時停止したり、進めたりできるようにした。

②ここでは描画間隔の変更ができる。デフォルトでは 1 日ごとの惑星の位置を表示しているが 100 日ごとに表示したり、1/100 日ごとに表示したりできるようにした。描画間隔が短いほど滑らかな描画になる。

③ここではそれぞれの惑星毎の視点に変更ができる。デフォルトでは太陽が画面の中心になっているが、注視点を変更するとその惑星での太陽系の動きを見ることができる。

④惑星の名前が邪魔である場合は消すことができるようにした。

⑤ここから太陽系の惑星軌道ページに飛べる。

⑥ここから月食日食ページに飛べる。

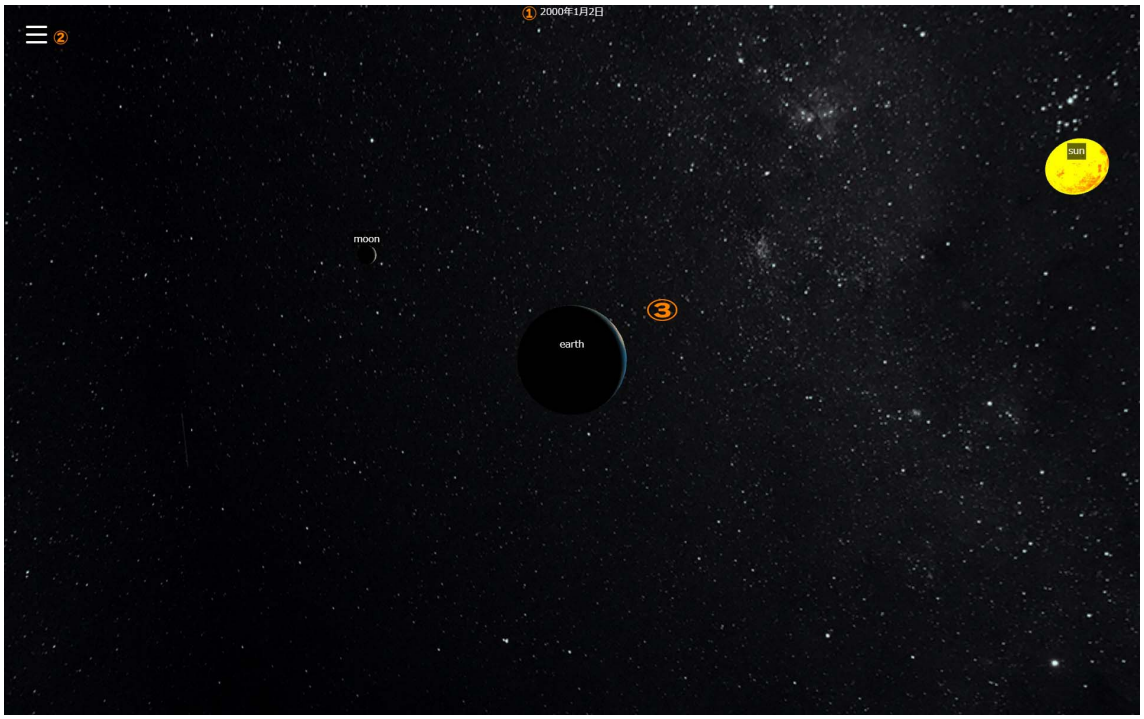


図 27　月食, 日食の画面

　日食月食はサイトの動作を軽くするために太陽系とは別のページで作成し、月食や日食の様子を確認できるページとしている。

①ここで表示されている日時の太陽、地球、月の位置を表示している

②操作パネルメニューを開閉可能なメニューにし、開閉可能にすることで画面全体で月食日食を表示するようにした。

③ここで太陽、地球、月を表示している。どの惑星が何かわかるようにラベルをつけた。

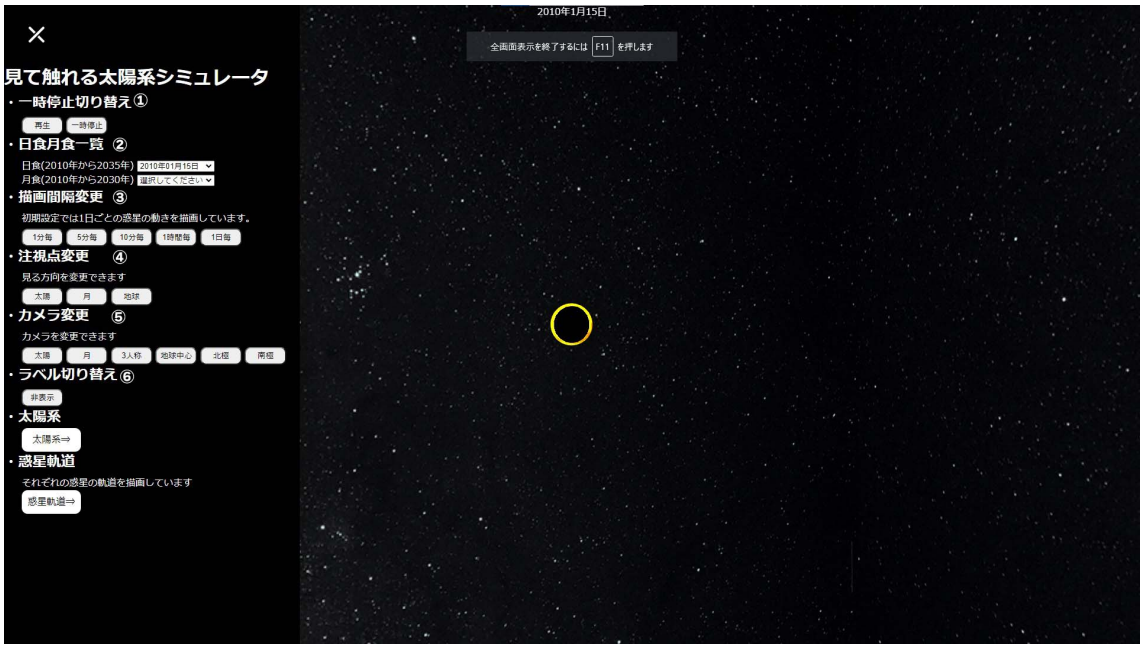


図 28　月食, 日食の画面

①ここでは一時停止と再生ボタンをおすことでシミュレーションを止めたりできるようにした。

②ここでは今まで観測された日食月食や予測されている日食月食の日時を選択すればその時の太陽、地球、月の位置が確認できるようにした。

③ここでは描画間隔の変更ができる。デフォルトでは 1 日ごとの惑星の位置を表示しているが 1 分ごとに表示したり、1 時間ごとに表示したりできるようにした。描画間隔が短いほど滑らかな描画になる。

④ここでは注視点を変更できるようにした。

⑤ここでは視点の位置を変更できる。例えば 3 人称視点ボタンを押すと惑星全体が見える位置に視点を変更することができ、北極, 南極ボタンを選択すると北極, 南極に視点が移動し観測することができる。

⑥惑星の名前が邪魔である場合は消すことができるようにした。

4.2　シミュレータの実行例

4.2.1　シミュレータの実行例:太陽系表示ページの例

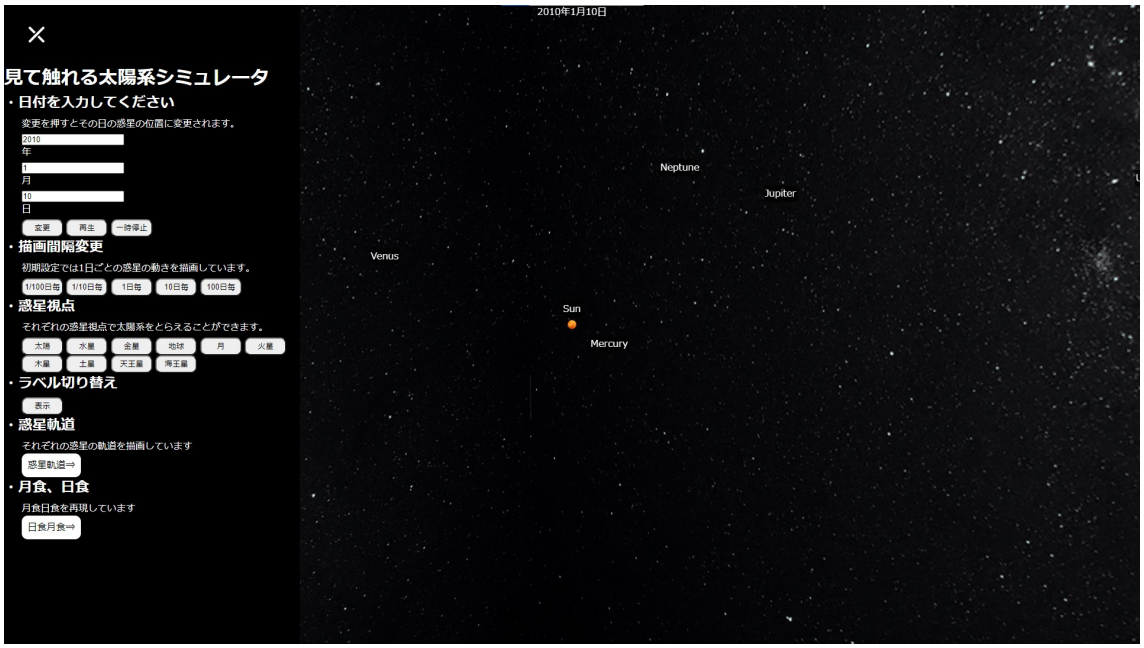


図 29　太陽系の実行例

例として 2010 年 1 月 10 日を入力すると、2010 年 1 月 1 日からシミュレータが始まりその日付になれば自動で止まるようにした。

4.2.2　シミュレータの実行例:日食の例

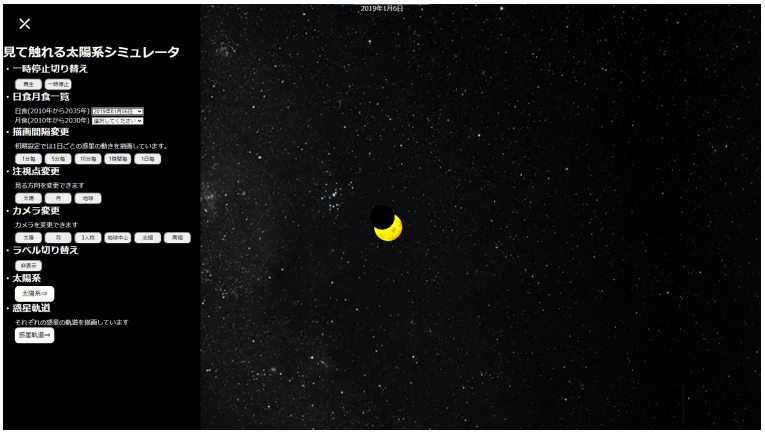


図 30　2019 年 1 月 6 日の日食の実行例

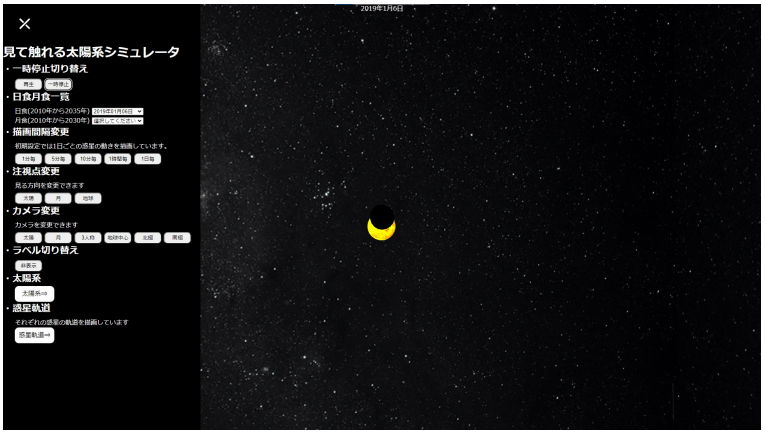


図 31　2019 年 1 月 6 日の日食の実行例

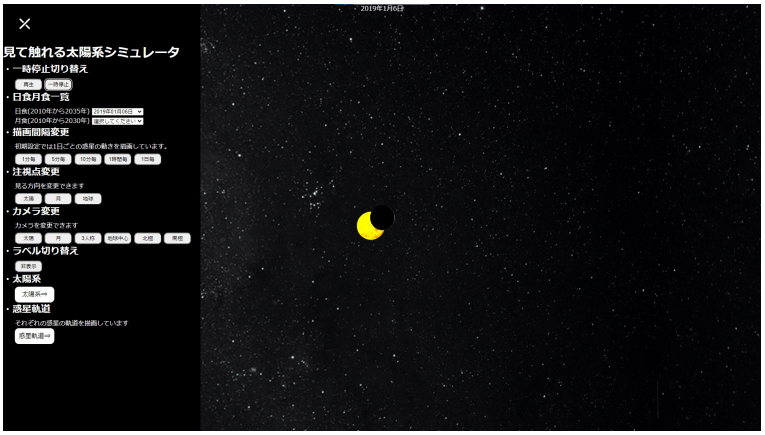


図 32　2019 年 1 月 6 日の日食の実行例

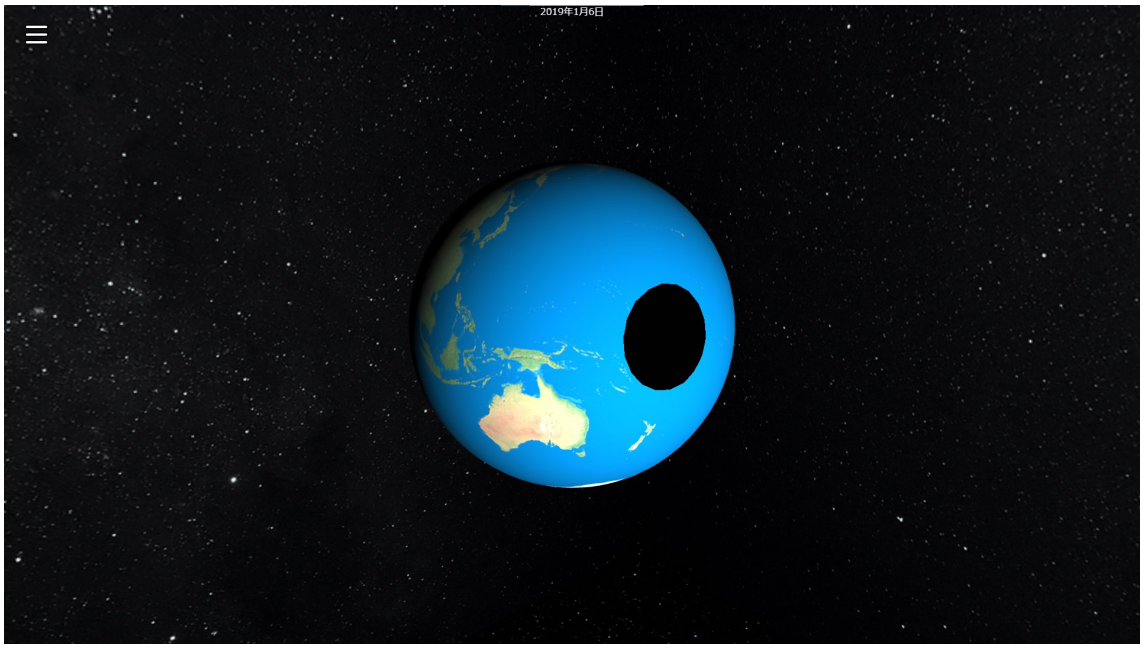
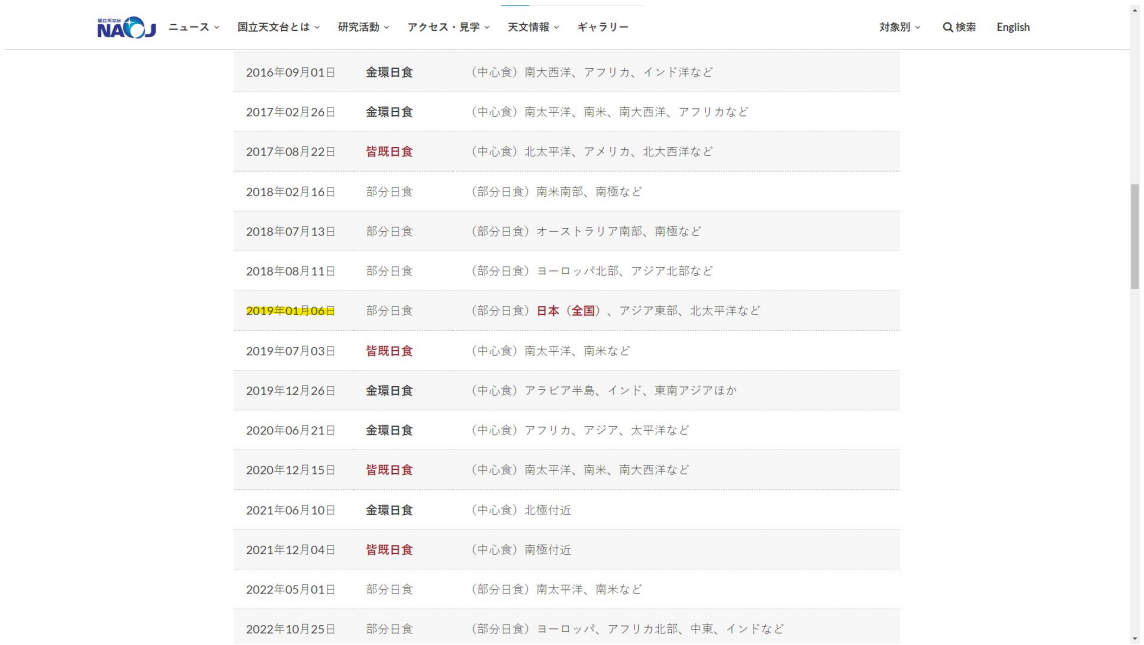


図 33　2019 年 1 月 6 日の日食の実行例



2016年09月01日	金環日食	（中心食）南大西洋、アフリカ、インド洋など
2017年02月26日	金環日食	（中心食）南太平洋、南米、南大西洋、アフリカなど
2017年08月22日	皆既日食	（中心食）北太平洋、アメリカ、北大西洋など
2018年02月16日	部分日食	（部分日食）南米南部、南極など
2018年07月13日	部分日食	（部分日食）オーストラリア南部、南極など
2018年08月11日	部分日食	（部分日食）ヨーロッパ北部、アジア北部など
2019年01月06日	部分日食	（部分日食）日本（全国）、アジア東部、北太平洋など
2019年07月03日	皆既日食	（中心食）南太平洋、南米など
2019年12月26日	金環日食	（中心食）アラビア半島、インド、東南アジアほか
2020年06月21日	金環日食	（中心食）アフリカ、アジア、太平洋など
2020年12月15日	皆既日食	（中心食）南太平洋、南米、南大西洋など
2021年06月10日	金環日食	（中心食）北極付近
2021年12月04日	皆既日食	（中心食）南極付近
2022年05月01日	部分日食	（部分日食）南太平洋、南米など
2022年10月25日	部分日食	（部分日食）ヨーロッパ、アフリカ北部、中東、インドなど

図 34　2019 年 1 月 6 日の日食の詳細

[4]

2019 年 1 月 6 日の日食を再現した例である。描画間隔を変更することで図 30〜図 32 のように影の変化が分かるようになっている。この時は部分日食で実際にそれが分かるようになっている。また、この時の日食は北太平洋あたりで見られることも図 11 から読み取れる。

■このシミュレータの対応日食の日付一覧　2010 年 01 月 15 日,2010 年 07 月 12 日,2011 年 01 月 04 日,2011 年 06 月 02 日,2011 年 07 月 01 日,2011 年 11 月 25 日,2012 年 05 月 21 日,2012 年 11 月 14 日,2013 年 05 月 10 日,2013 年 11 月 03 日, 2014 年 04 月 29 日,2014 年 10 月 24 日,2015 年 03 月 20 日,2015 年 09 月 13 日,2016 年 03 月 09 日,2016 年 09 月 01 日,2017 年 02 月 26 日,2017 年 08 月 22 日,2018 年 02 月 16 日,2018 年 07 月 13 日, 2018 年 08 月 11 日,2019 年 01 月 06 日,2019 年 07 月 03 日,2019 年 12 月 26 日,2020 年 06 月 21 日,2020 年 12 月 15 日,2021 年 06 月 10 日,2021 年 12 月 04 日,2022 年 05 月 01 日,2022 年 10 月 25 日,2023 年 04 月 20 日, 2023 年 10 月 15 日,2024 年 04 月 09 日,2024 年 10 月 03 日,2025 年 03 月 29 日,2025 年 09 月 22 日,2026 年 02 月 17 日,2026 年 08 月 13 日,2027 年 02 月 07 日,2027 年 08 月 02 日, 2028 年 01 月 27 日,2028 年 07 月 22 日,2029 年 01 月 15 日,2029 年 06 月 12 日,2029 年 07 月 12 日,2029 年 12 月 06 日,2030 年 06 月 01 日,2030 年 11 月 25 日,2035 年 9 月 2 日 [4]

4.2.3　シミュレータの実行例:月食の例

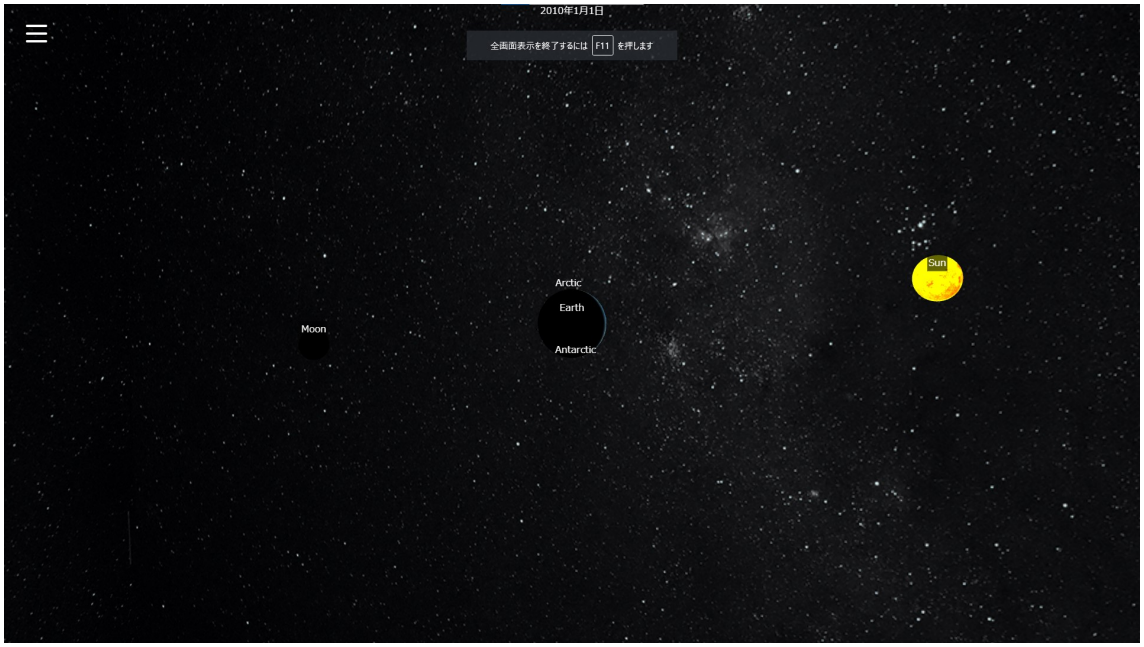


図 35　2010 年 01 月 1 日の月食の実行例

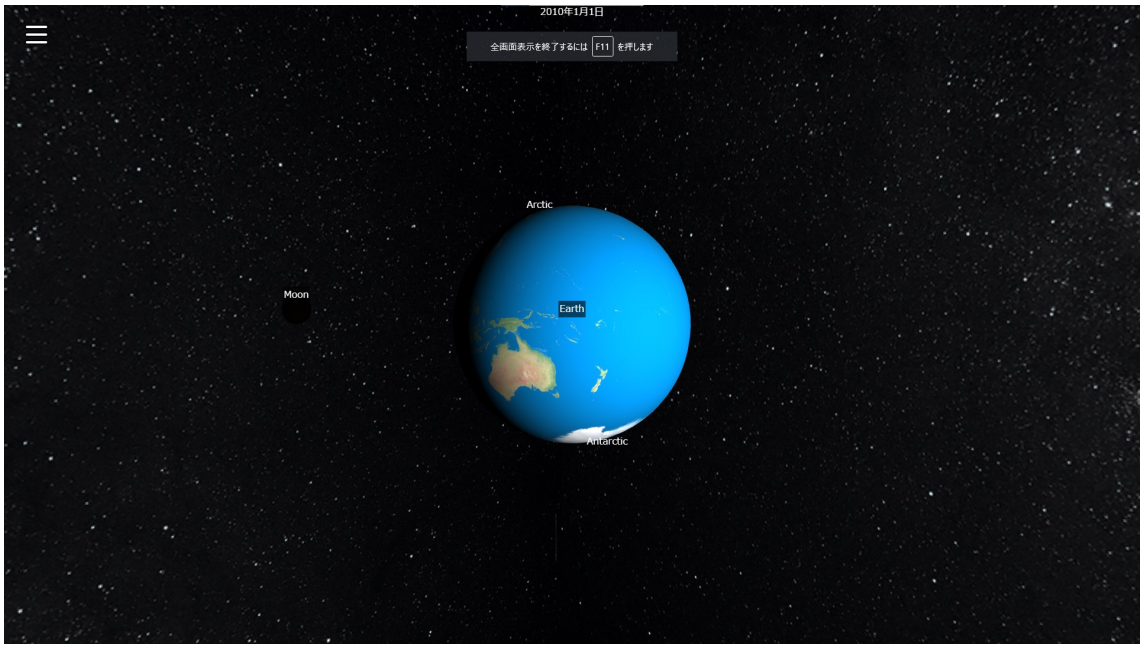


図 36　2010 年 01 月 1 日の月食の実行例

2010 年 01 月 01 日の月食を再現した例である。太陽と地球と月が一直線に並んでおり、地球の影に月が隠れていることが分かる。

■このシミュレータの対応月食日付一覧　2010 年 01 月 01 日, 2010 年 06 月 26 日,2010 年 12 月 21 日,2011 年 06 月 16 日,2011 年 12 月 10 日,2012 年 06 月 04 日,2013 年 04 月 26 日,2014 年 04 月 15 日,2014 年 10 月 08 日,2015 年 04 月 04 日, 2015 年 09 月 28 日,2017 年 08 月 08 日,2018 年 01 月 31 日,2018 年 07 月 28 日,2019 年 01 月 21 日,2019 年 07 月 17 日,2021 年 05 月 26 日,2021 年 11 月 19 日,2022 年 05 月 16 日,2022 年 11 月 08 日, 2023 年 10 月 29 日,2024 年 09 月 18 日,2025 年 03 月 14 日,2025 年 09 月 08 日,2026 年 03 月 03 日,2026 年 08 月 28 日,2028 年 01 月 12 日,2028 年 07 月 07 日,2029 年 01 月 01 日, 2029 年 06 月 26 日,2029 年 12 月 21 日,2030 年 06 月 16 日 [4]

4.3　他アプリケーションとの比較

今回ソフトのインストールやライセンスの購入などが必要のないブラウザで使用できるシミュレータを作成したが、宇宙のシミュレーションが出来る他ソフトとの比較をここで述べる。類似アプリとして Steam[5] という海外のアプリケーション販売サイト内で販売されている「SpaceEngine」[6] というソフトと比較してみる。

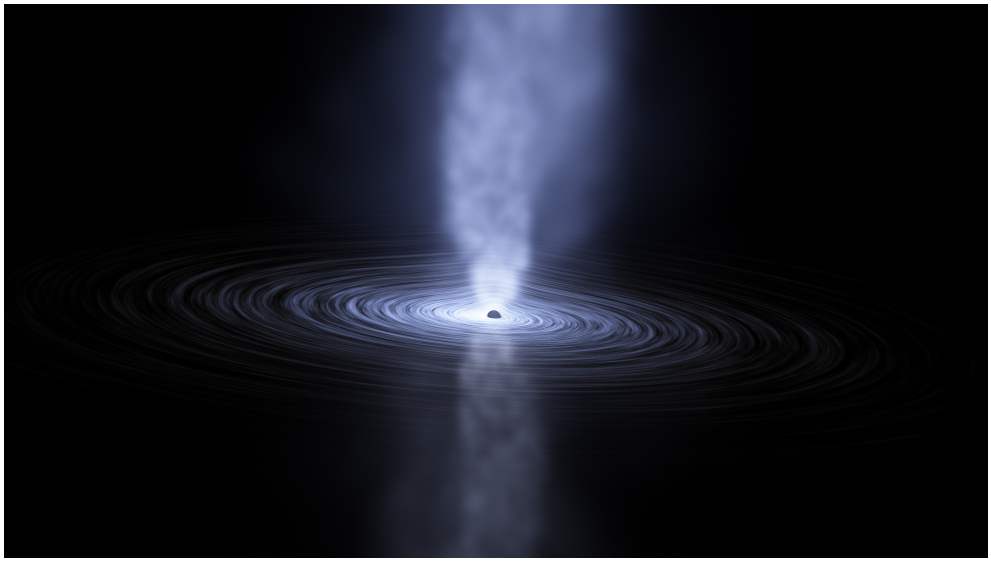


図 37　SpaceEngine 例

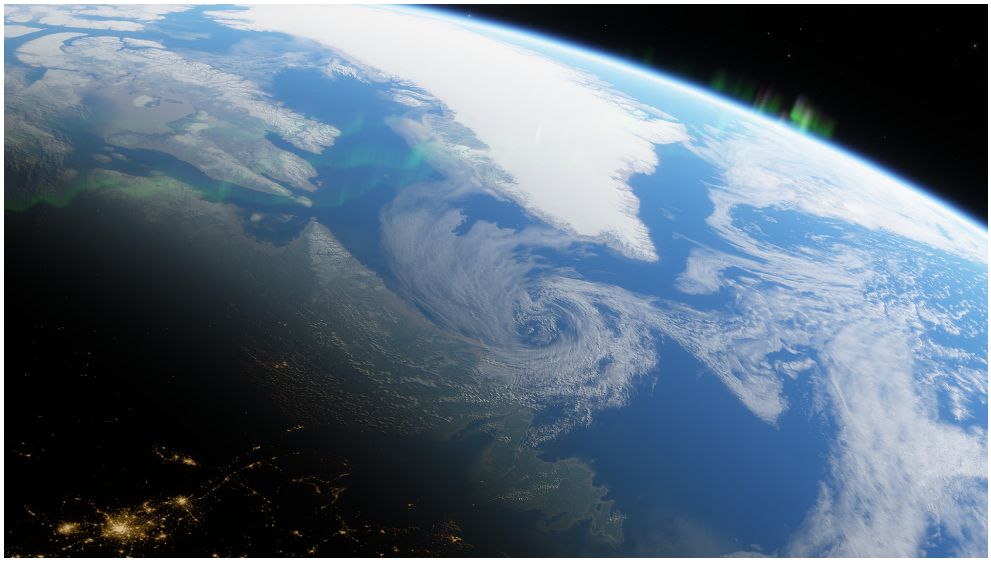


図 38　SpaceEngine 例

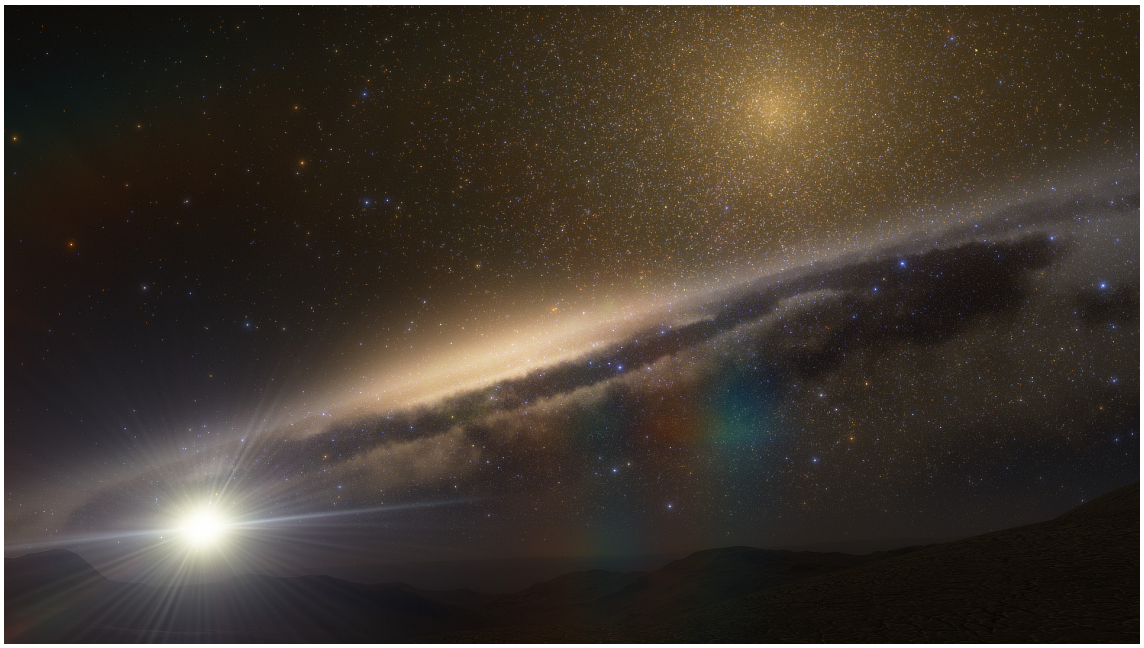


図 39　SpaceEngine 例

このように惑星の表示だけでなく星雲の表示もできる。例えば地球から見た太陽を表示してみると次のようになる。

図 40　SpaceEngine で地球から見た太陽を表示

光の再現までリアルにされている。

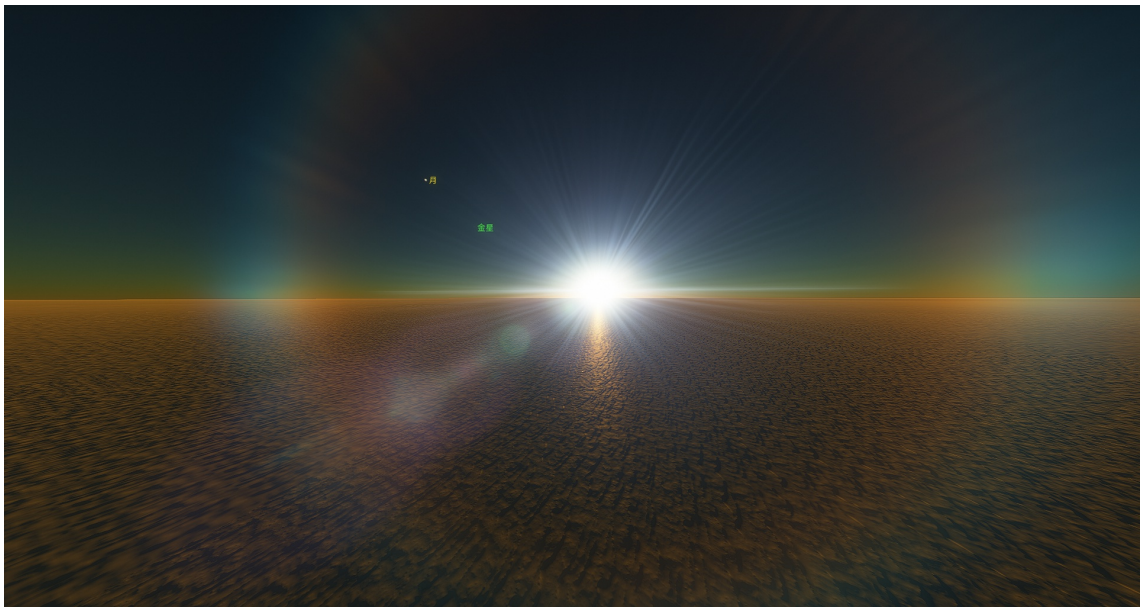


図 41　SpaceEngine で地表から見た太陽を表示

地球限定ではあるが地表に降りることができる。

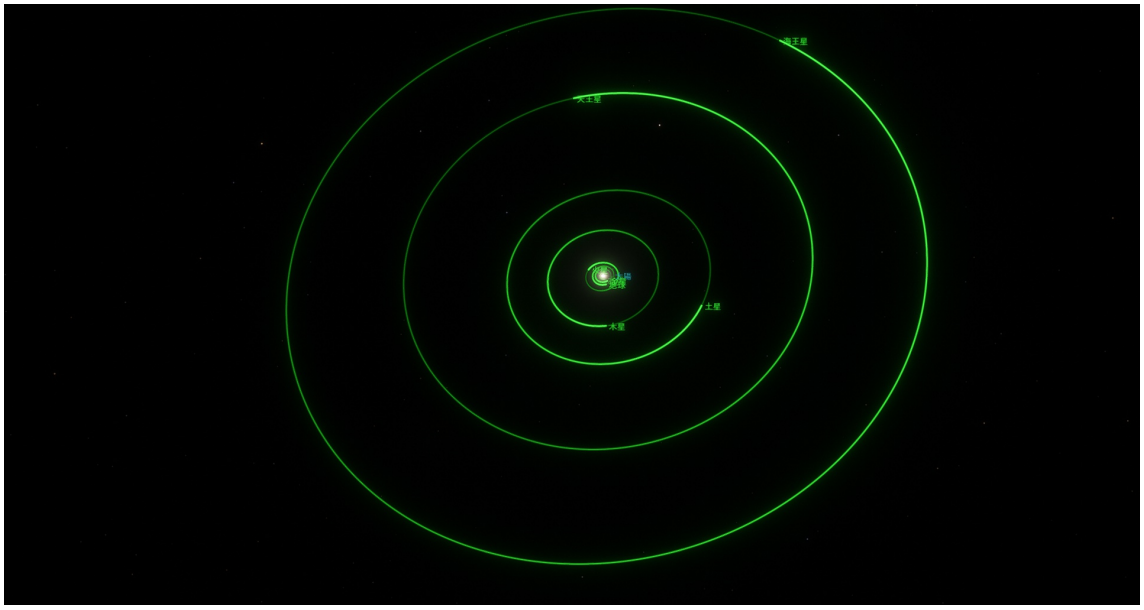


図 42　SpaceEngine で太陽系の表示

このように「SpaceEngine」は実際に宇宙に行きそこでシミュレーションをしているかのような体験ができる。

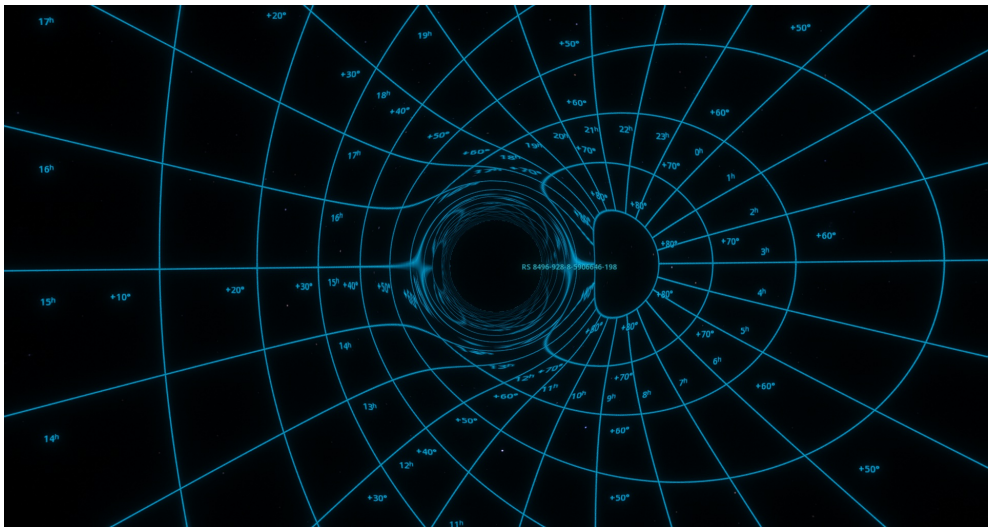


図 43　SpaceEngine でブラックホールの表示

図 44　SpaceEngine のメニュー

ただできることが多いこともあり、操作が複雑で直感操作はできない。メニューも一部であるが惑星の表示以外にも水星や小惑星、ブラックホールの表示もできる。このソフトは Steam で 2570 円で販売されている。このようにシミュレータ上で色んな事ができるソフトは有料で販売されており、またソフトの容量が大きくインストールに時間がかかってしまう。この点でこのシミュレータの方が無料で利用できる上インストールなど不要であるため優れている。

5　まとめ

本シミュレータで使用している Runge-Kutta 法は 4 次のものを用いているが、4 次であってもかなり厳密な再現ができるということが分かった。ただ計算量が多いということもあり、惑星の描画の早送りなどはできないという課題がある。このシミュレータの作成上で今後の課題点としては次が挙げられる。

- web で実際の宇宙空間を再現しているため、球などの物体や光、影を表示できる THREE.js という WEB 上で 3 次元コンピュータグラフィックスが使用できるライブラリを使っている。その為、サイトを動かし続けていると動作が重くなってしまう。
- インターフェースを改善し、見やすくする。
- JavaScript の規格により使用するブラウザによって対応していない部分がある為その改善。

これが大きな課題点である。

参考文献

[1]「Solar eclipse predictions」 J. Mottmann (1980)

[2]「NIST」Values of Fundamental Physical Constants (2021),https://physics.nist.gov/cuu/Constants/

[3]「Horizon System」 (2021),https://ssd.jpl.nasa.gov/horizons/app.html#/

[4]「 国 立 天 文 台:日 食 一 覧 」,https://www.nao.ac.jp/astro/basic/solar-eclipse-list.html

[5]「Steam」,https://store.steampowered.com/

[6]「Space Engine」,https://store.steampowered.com/app/314650/SpaceEngine/