Gimnazija Franca Miklošiča Ljutomer
Prešernova 34, Ljutomer

Vpliv planetov na Zemljino orbito in temperaturo na Zemlji

RAZISKOVALNA NALOGA
Področje: FIZIKA IN ASTRONOMIJA

Avtor:
Maj Pintarič
Mentor:
Sandi Šalamon

Leto izdelave:
2022/2023
Ljutomer, marec 2023

KAZALO VSEBINE

KAZALO TABEL

KAZALO SLIK

POVZETEK

V raziskovalni nalogi sem želel ugotoviti, kakšen vpliv imajo ostali planeti na Zemljino orbito in temperaturo na Zemlji. V ta namen sem v programskem jeziku Python izdelal sedem simulacij Osončja. V vsakem programu sem meril, kako odsotnost določenih planetov vpliva na Zemljino orbito, orbitalno hitrost, obhodni čas in efektivno temperaturo. Primerjal sem simulacijo z vsemi planeti in simulacijo z zgolj Zemljo, da bi ugotovil, kako se Zemljina orbita spremeni brez ostalih planetov. Zemljina orbita se je v določenih parametrih povečala, v določenih pa pomanjšala. Primerjal sem tudi simulacijo z zgornjimi planeti in simulacijo s spodnjima planetoma, da bi ugotovil, kateri planeti imajo večji vpliv na Zemljino orbito. Ker imata spodnja planeta večjo kotno hitrost od zgornjih, sta imela večji vpliv na y koordinato Zemljine orbite. Zunanji planeti so imeli zaradi manjše kotne hitrosti večji vpliv na x koordinato Zemjine orbite. Ugotavljal sem, kateri planet lahko najbolj spremeni Zemljino orbito. S primerjanjem sil med planeti in Zemljo sem ugotovil, da je to Jupiter. Zanimal me je še vpliv planetov na obhodni čas okoli Sonca in površinsko temperaturo na Zemlji. Rezultati so pokazali, da se kljub odsotnosti večine planetov Zemljina orbita in njena površinska temperatura spremenita za zelo majhne vrednosti, največ za 0,0018 °C.

ABSTRACT

In this research, I wanted to discover how other planets affect the Earth's orbit and its temperature. Therefore, I created seven computer simulations of the Solar System in Python. In each simulation, I measured how the absence of certain planets would affect the Earth's orbit, orbital velocity, orbital period and effective temperature. I compared the two simulations – the one with all planets and the one with just the Earth – in order to determine how the Earth's orbit changes without the other planets. Earth's orbit increased in certain parameters and decreased in others. In addition, I compared the simulation to establish which planets have greater influence on the Earth's orbit. Because the inferior planets have a higher angular velocity than the superior planets, they have a greater influence on the y-coordinate of Earth's orbit. The superior planets had a greater influence on the x-coordinate of Earth's orbit because of their lower angular velocity. Collectively, I was trying to determine which planet can change Earth's orbit the most hence I did a comparison of the forces between other planets and Earth – the results showed that the greater influencer is Jupiter itself. I was also interested in the influence of the planets on the orbital period around the Sun and the surface temperature of the Earth. The results showed that, despite the absence of most planets, the Earth's orbit and surface temperature change by very small amounts, up to a maximum of 0. 0018 °C.

ZAHVALA

Za pomoč pri nastajanju raziskovalne naloge se zahvaljujem mentorju iz Gimnazije Franca Miklošiča Ljutomer, Sandiju Šalamonu, profesorju fizike. Še posebej bi se zahvalil Juriju Šumaku in Janu Casarju za pomoč pri programiranju ter profesorici Katji Peršak Hajdinjak za lektoriranje besedila.

UPORABLJENI SIMBOLI IN OZNAKE

Latinski simboli

Tabela 1: Latinski simboli

Oznaka	Ime količine	Enota
$\overrightarrow{\boldsymbol{F}}_{\mathbf{1,2}}$	vektor sile	[N]
$\overrightarrow{\boldsymbol{v}}$	povprečna hitrost	[m/s^2]
$\boldsymbol{D_p}$	razdalja med Soncem in planetom	[m]
$\boldsymbol{F_g}$	gravitacijska sila	[N]
$\boldsymbol{I_p}$	dolgovalovno sevanje, ki ga odda planet	[W/m^2]
$\boldsymbol{P_{ave}}$	povprečna količina energije na planetu	[W]
$\boldsymbol{P_o}$	skupna energija krogle orbite planeta	[W]
$\boldsymbol{P_p}$	skupna energija, ki pade na planet	[W]
$\boldsymbol{P_s}$	skupna energija na površini Sonca	[W]
$\boldsymbol{T_a}$	temperatura atmosfere	[K]
$\boldsymbol{T_p}$	površinska temperatura planeta	[K]
$\boldsymbol{T_s}$	površinska temperatura Sonca	[K]
$\boldsymbol{T_z}$	površinska temperatura Zemlje	[K]
$\boldsymbol{a_r}$	radialni pospešek	[m/s^2]
$\boldsymbol{j}^*$	gostota izsevanega svetlobnega toka zvezde	[W/m^2]
$\boldsymbol{j_{ave}}$	povprečna gostota svetlobnega toka na planetu	[W/m^2]
$\boldsymbol{j_o}$	gostota svetlobnega toka valovanja, ki se odbije od planeta	[W/m^2]
$\boldsymbol{j_p}$	gostota svetlobnega toka krogle orbite planeta	[W/m^2]
$\boldsymbol{m_1}$	masa delca 1	[kg]
$\boldsymbol{m_2}$	masa delca 2	[kg]
$\boldsymbol{r_{,12}}$	razlika pozicij	[m]
$\hat{\boldsymbol{r}}$	enotski vektor	[-]
$\boldsymbol{r_1}$	pozicija delca 1	[m]
$\boldsymbol{r_p}$	polmer planeta	[m]
$\boldsymbol{r_s}$	polmer Sonca	[m]
$\boldsymbol{r_2}$	pozicija delca 2	[m]
$\boldsymbol{t_0}$	začetni čas	[s]
$\boldsymbol{t_1}$	časovni interval 1	[s]
$\boldsymbol{t_o}$	obhodni čas	[s]
$\boldsymbol{A}$	velika polos	[m]
$\boldsymbol{B}$	mala polos	[m]
$\boldsymbol{D}$	razdalja med Zemljo in Soncem	[m]
$\boldsymbol{F}$	sila	[N]
$\boldsymbol{G}$	gravitacijska konstanta	[N·m^2/kg^2]

$\boldsymbol{I}\uparrow$	dolgovalovno sevanje, ki ga atmosfera odda v vesolje	[W/m^2]
$\boldsymbol{I}\downarrow$	dolgovalovno sevanje, ki ga atmosfera odda proti planetu	[W/m^2]
$\boldsymbol{M}$	masa Sonca	[kg]
$\boldsymbol{P}$	povprečna moč	[W]
$\boldsymbol{R}$	razdalja med delcema	[m]
$\boldsymbol{S}$	površina	[m^2]
$\boldsymbol{a}$	albedo	[-]
$\boldsymbol{d}$	diferencial	[-]
$\boldsymbol{j}$	jakost valovanja	[W/m^2]
$\boldsymbol{m}$	masa planeta	[kg]
$\boldsymbol{o}$	obseg elipse	[m]
$\boldsymbol{p}$	gibalna količina	[kg ·m/s]
$\boldsymbol{t}$	časovni interval	[s]
$\boldsymbol{v}$	obhodna hitrost	[m/s]
$\boldsymbol{x}$	vrednost Zemljine pozicije na x osi	[m]
$\boldsymbol{y}$	vrednost Zemljine pozicije na y osi	[m]

Grški simboli

Tabela 2: Grški simboli

Oznaka	Ime količine	Enota
$\boldsymbol{\varepsilon_a}$	absorpcijska sposobnost	[-]
$\boldsymbol{\varepsilon_i}$, $\boldsymbol{\varepsilon}$	izsevnost	[-]
$\boldsymbol{\pi}$	pi	[-]
$\boldsymbol{\sigma}$	Stefanova konstanta	[W·K^4/m^2]

Oznake, uporabljene v programu Python

Tabela 3: Oznake, uporabljene v programu Python

Oznaka	Ime količine
`gforce`	gravitacijska sila
`G`	gravitacijska konstanta
`r_vec`	razlika med pozicijama
`planet, planet3`	Zemlja
`planet1`	Merkur
`planet2`	Venera
`planet4`	Mars
`planet5`	Jupiter
`planet6`	Saturn
`planet7`	Uran
`planet8`	Neptun
`p1, p2`	vesoljsko telo

`.pos`	pozicija
`r_mag`	absolutna vrednost vektorja `r_vec`
`r_hat`	enotski vektor
`force_mag`	absolutna vrednost gravitacijske sile
`.mass`	masa telesa
`force_vec`	vektor gravitacijske sile
`star`	Sonce
`t`	čas
`dt`	časovni korak
`.force`	sila na telo
`.momentum`	gibalna količina telesa
`maxpos_x`	maksimalna pozicija na x osi
`maxpos_y`	maksimalna pozicija na y osi
`current_y`	trenutna pozicija na y osi
`previous_y`	pozicija na y osi v prejšnjem časovnem intervalu
`i`	vsota števila podatkov neke vrednosti
`total_D`	seštevek razdalj
`total_v`	seštevek hitrosti
`D`	razdalja med Soncem in Zemljo
`v`	obhodna hitrost
`o`	obseg orbite
`a`	zemljin albedo
`ε`	izsevnost atmosfere
`Ts`	površinska temperatura Sonca
`Te`	površinska temperatura Zemlje

1. UVOD

Zemlja že več kot 4,5 milijarde let kroži okoli Sonca. Njeno premikanje omogoča gravitacijska sila, ki deluje med vsemi delci, ki imajo neko maso. Sila med Zemljo in Soncem je dovolj velika, da je lahko Zemlja vzpostavila stabilno orbito na dolžini 150.000.000 km.

Za premikanje Zemlje ni odgovorna le sila med Zemljo in Soncem, temveč so odgovorne tudi sile vseh delcev v vesolju. Mnogi delci so zelo majhni in oddaljeni, zato je sila med njimi in Zemljo zanemarljiva. Planeti v našem Osončju pa so dovolj blizu in imajo dovolj veliko maso, da rahlo spremenijo Zemljino kroženje okoli Sonca. Planeti se med seboj razlikujejo v masi in oddaljenosti od Zemlje, zato se tudi njihove sile na Zemljo razlikujejo.

S pomočjo razumevanja teorije in programiranja sedmih različnih simulacij Osončja sem hotel ugotoviti, kako določeni planeti spremenijo Zemljino orbito in njeno površinsko temperaturo. Program sem pisal v programskem jeziku Python. Uporabljal sem integrirano razvojno okolje Pycharm in knjižnico Vpython, ki mi je omogočila ustvarjanje predmetov v 3D-prostoru. Da bi določil, kako planeti vplivajo na Zemljo, sem meril sedem spremenljivk. Te spremenljivke so bile: velika polos, mala polos, povprečna razdalja, povprečna hitrost, obseg orbite, dolžina leta in efektivna temperatura. Vseh sedem programov se nahaja v prilogi.

1.1. NAMEN NALOGE

Namen naloge je ugotoviti, kakšne bi bile lastnosti Zemljinega kroženja in njena površinska temperatura, če bi:

- iz Osončja izbrisali vse ostale planete,
- iz Osončja izbrisali zgornje planeta,
- iz Osončja izbrisali spodnja planete,
- iz Osončja izbrisali planet, ki najbolj vpliva na Zemljino orbito,
- planet, ki najbolj vpliva na Zemljino orbito, postavili v točko v orbiti, v kateri je najbolj oddaljen od Zemlje.

1.2. RAZISKOVALNO VPRAŠANJE

Raziskoval bom, kako se spreminja Zemljina orbita in površinska temperatura v odsotnosti različnih skupin planetov. Zanima me, kako se spremeni Zemljina orbita v odsotnosti vseh planetov, ali zgornji planeti bolj vplivajo na Zemljo, kot spodnja, in kateri planet najbolj vpliva na Zemljino orbito. Pri tem me še zanima, kako se spremeni Zemljina orbita v odsotnosti tega planeta, ter kako, ko je ta planet najbolj oddaljen od Zemlje. Ključno vprašanje, na katerega želim najti odgovor, je, kako velike so spremembe v Zemljini orbiti ob odsotnosti določenih planetov.

2. TEORETIČNO OZADJE

2.1. GIBANJE PLANETOV OKOLI SONCA

2.1.1. Gravitacijska sila

Sila je kakršen koli pritisk ali vlečenje predmeta. Premikanje planetov omogoča gravitacijska sila, ki jo lahko opišemo po Newtonovem splošnem gravitacijskem zakonu. Ta pravi, da vsak delec z maso v vesolju privlači vse druge delce s silo, ki je sorazmerna s produktom njunih mas in obratno sorazmerna s kvadratom razdalje med njima. Ta sila deluje vzdolž črte, ki povezuje oba delca.

$$F_g = G\frac{m_1 m_2}{R^2},$$

(1)

pri čemer je F_g gravitacijska sila med dvema telesoma, m_1 in m_2 sta masi delcev, R razdalja med njima in G gravitacijska konstanta ($G = 6{,}67 \cdot 10^{-11}\,\frac{\mathrm{Nm^2}}{\mathrm{kg^2}}$). V primeru planeta, ki kroži okoli Sonca, velja:

$$F_g = G\frac{Mm}{R^2},$$

(2)

kjer je M masa Sonca in m masa planeta. Če dano enačimo z II. Newtonovim zakonom $F = ma_r$, lahko zapišemo, da je:

$$G\frac{Mm}{R^2} = ma_r,$$

(3)

pri čemer je a_r radialni pospešek, ki je enak razmerju med kvadratom obhodne hitrosti (v) in razdaljo med njima (R), zato lahko trdimo:

$$G\frac{Mm}{R^2} = m\frac{v^2}{R}.$$

(4)

Iz te enačbe lahko izrazimo obhodno hitrost in dobimo:

$$v = \sqrt{\frac{GM}{R}}.$$

(5)

Gibalna količina telesa je definirana kot zmnožek njegove mase in hitrosti:

$$p = mv.$$

(6)

2.1.2. Gravitacija kot vektor

Newtonov splošni gravitacijski zakon lahko zapišemo v vektorski obliki, ki upošteva smer gravitacijske sile in njeno velikost:

$$\vec{F}_{1,2} = -G\frac{m_1 m_2}{|r_{,12}|^2}\hat{r},$$

(7)

kjer je $\vec{F}_{1,2}$ sila, s katero delec 2 deluje na delec 1, $|r_{1,2}| = r_1 - r_2$ razlika med pozicijama delcev, $\hat{r}$ pa enotski vektor od delca 2 na delec 1, ali:

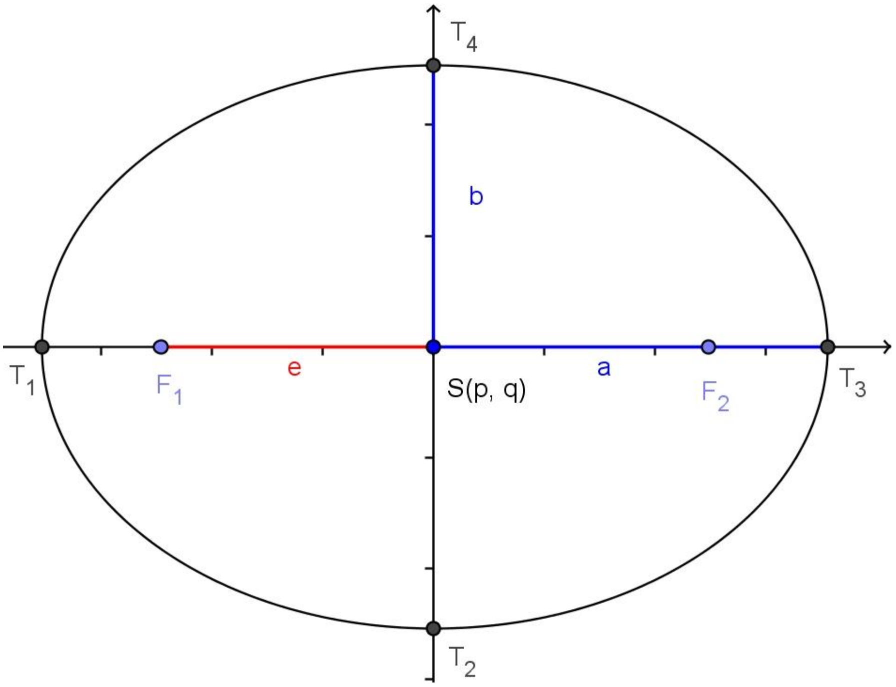
$$\hat{r} = \frac{r_1 - r_2}{|r_1 - r_2|}.$$

(8)

Sila ima negativen predznak, saj je privlačna [2].

2.2. OBSEG ORBITE

I. Keplerjev zakon pravi, da je orbita vsakega planeta elipsa s Soncem v enem izmed gorišč [1]. Na sliki 1 je prikazana elipsa z veliko polosjo A in malo polosjo B.



Slika 1: Elipsa

Pot, ki jo planet opiše na enem obhodu okoli Sonca, je enaka obsegu njegove elipsaste orbite. Obseg elipse (o) izračunamo z [3]:

$$o = \pi(A + B)\left(1 + \frac{3\frac{(A-B)^2}{(A+B)^2}}{10 + \sqrt{4 - 3\frac{(A-B)^2}{(A+B)^2}}}\right).$$

(9)

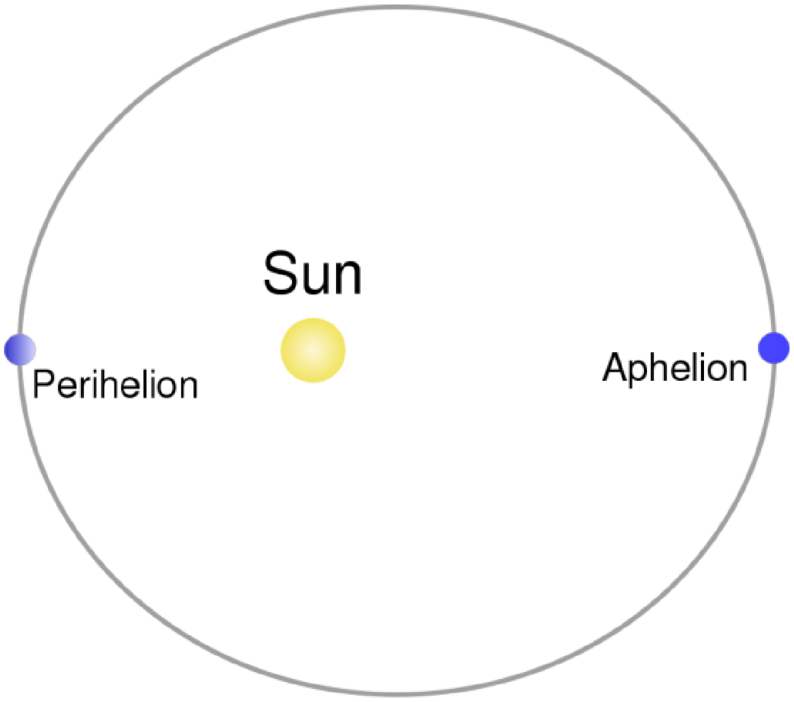
2.3. OBHODNI ČAS

Obhodni čas (t_o) je časovni interval enega obhoda. Pri enakomernem kroženju je enak kvocientu med prepotovano potjo in obhodno hitrostjo [4]:

$$t_o = \frac{o}{v} = \frac{2\pi R}{v}.$$

(10)

Planeti se okoli Sonca ne gibljejo enakomerno, saj je njihova orbita eliptična. Iz I. Keplerjevega zakona lahko izpeljemo, da se orbitalna hitrost spreminja. Planet ima največjo hitrost, ko je v Soncu najbližji točki, periheliju, in najmanjšo, ko je v Soncu najbolj oddaljeni točki, afeliju.



Slika 2: Perihelij in afelij

Pri neenakomernem kroženju je obhodni čas enak razmerju med prepotovano potjo in povprečno hitrostjo. Povprečno hitrost ($\underset{v}{\rightarrow}$) lahko zapišemo kot integral hitrosti [5]:

$$\underset{v}{\rightarrow} = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} v(t)\, dt.$$

(11)

2.4. EFEKTIVNA TEMPERATURA

2.4.1. Stefan-Boltzmannov zakon

Jakost valovanja (j) je enaka povprečni moči (P), ki jo valovanje prenese skozi neko površino (S) skozi prostor, torej [6]:

$$j = \frac{P}{S}. \tag{12}$$

Valovanje, ki ga oddajajo zvezde, je elektromagnetno valovanje, ki je sestavljeno iz valovanj različnih valovnih dolžin. Infrardeče valovanje je odgovorno za segrevanje planeta [1], vidna svetloba pa nam omogoča videnje barv. Barva zvezd je odvisna od njene temperature, temperatura pa je sorazmerna z jakostjo valovanja ali gostoto svetlobnega toka. To opisuje Stefan-Boltzmannov zakon [7]:

$$j^* = \varepsilon_i \sigma {T_s}^4, \tag{13}$$

kjer je j^* gostota izsevanega svetlobnega toka zvezde, ε_i izsevnost zvezde, σ Stefanova konstanta $(\sigma = 5{,}67 \cdot 10^{-8} \frac{\mathrm{WK^4}}{\mathrm{m^2}})$ in T_s površinska temperatura zvezde, v tem primeru Sonca. Zvezde so skoraj popolna črna telesa, zato je Sončeva izsevnost enaka 1, torej:

$$j^* = \sigma {T_s}^4. \tag{14}$$

Zvezda z dvakrat večjo temperaturo izseva šestnajstkrat več energije [1].

2.4.2. Gostota svetlobnega toka na površini krogle

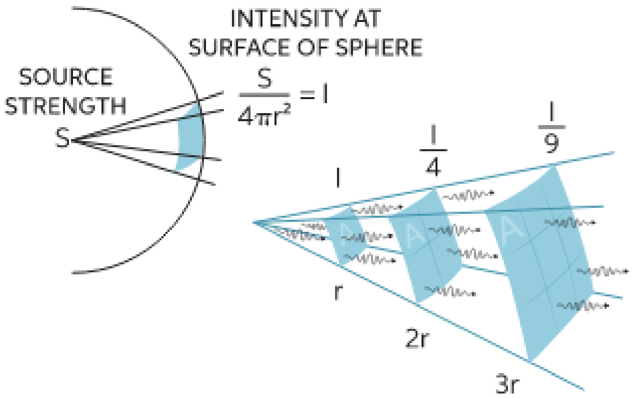
Povprečna moč sevanja je v primeru zvezde njen izsev. Po Stefan-Boltzmannovem zakonu je izsev ali skupna energija enaka produktu gostote svetlobnega toka in površine krogle. Površina krogle z radijem r je $4\pi r^2$, torej lahko zapišemo:

$$P = j^* S = j^* 4\pi r_s^2. \tag{15}$$

Če je r_s polmer Sonca, je skupna energija na površini Sonca (P_s) enaka:

$$P_s = j^* 4\pi r_s^2. \tag{16}$$

Pri oddajanju sevanja s te kroglaste površine se sevanje razprši po vedno večjih kroglastih površinah, zato se gostota svetlobnega toka manjša, kot je shematično prikazano na sliki 3. Na sliki je gostota svetlobnega toka označena z I.



Slika 3: Spreminjanje gostote svetlobnega toka z oddaljenostjo

Ko energija, ki jo oddaja Sonce, doseže orbito planeta (P_o, o – orbita), se razprši v sfero s polmerom, enakim razdalji med Soncem in planetom (D_p):

$$P_o = j_p 4\pi D_p^2. \quad (17)$$

Gostota svetlobnega toka na kateremkoli mestu na tej površini (j_p) je manjša od tiste na površini Sonca. Vendar je skupna energija, ki se razprši po tej veliki površini, enaka skupni energiji, ki je na površini Sonca, zato ju lahko enačimo:

$$P_s = P_o, \quad (18)$$

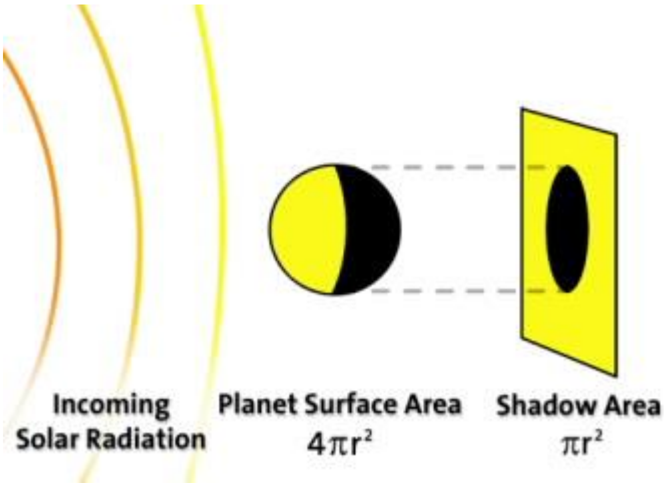
$$j^* 4\pi r_s^2 = j_p 4\pi D_p^2. \quad (19)$$

Če izrazimo j_p, dobimo:

$$j_p = \frac{j^* r_s^2}{D_p^2}. \quad (20)$$

2.4.3. Gostota svetlobnega toka na površju planeta

Ko sončno sevanje doseže planet, ne pade na vsa območja planeta pod enakim kotom. V bližini ekvatorja pada neposredno, v bližini polov pa bolj poševno. Povprečno količino energije, ki vstopa v planetovo ozračje, lahko določimo s pomočjo spodnje slike.



Slika 4: Povprečna količina energije, ki pada na planet

Skupna količina sevanja, ki pade na planet, je enaka količini, ki jo planet lahko prestreže. S tem planet odda namišljeno krožno senco, kot je prikazano na sliki. Planet lahko prestreže le toliko energije, kolikšen delež površine planeta je ploščina sence. Ploščina sence je enaka ploščini kroga s polmerom enakem polmeru planeta (r_p), torej πr_p^2. Potemtakem je skupna količina sevanja, ki pade na planet (P_p):

$$P_p = j_p \pi r_p^2. \tag{21}$$

Povprečna količina sevanja na površju planeta (P_{ave}, ave – average) je enaka produktu povprečne gostote svetlobnega toka na površju planeta (j_{ave}) in površini planeta ($4\pi r_p^2$). P_p in P_{ave} sta enaki, zato lahko enačimo:

$$P_p = P_{ave}, \tag{22}$$

$$j_p \pi r_p^2 = j_{ave} 4\pi r_p^2. \tag{23}$$

Izpostavimo j_{ave}:

$$j_{ave} = \frac{j_p}{4}, \tag{24}$$

torej je za vsak planet povprečna gostota svetlobnega toka na površju planeta (j_{ave}) štirikrat manjša kot gostota svetlobnega toka na planetovi namišljeni senci (j_p). To je možno razbrati iz slike 4. Površina krogle je večja od površine kroga, zato se mora enaka količina energije razporediti po večji površini [8].

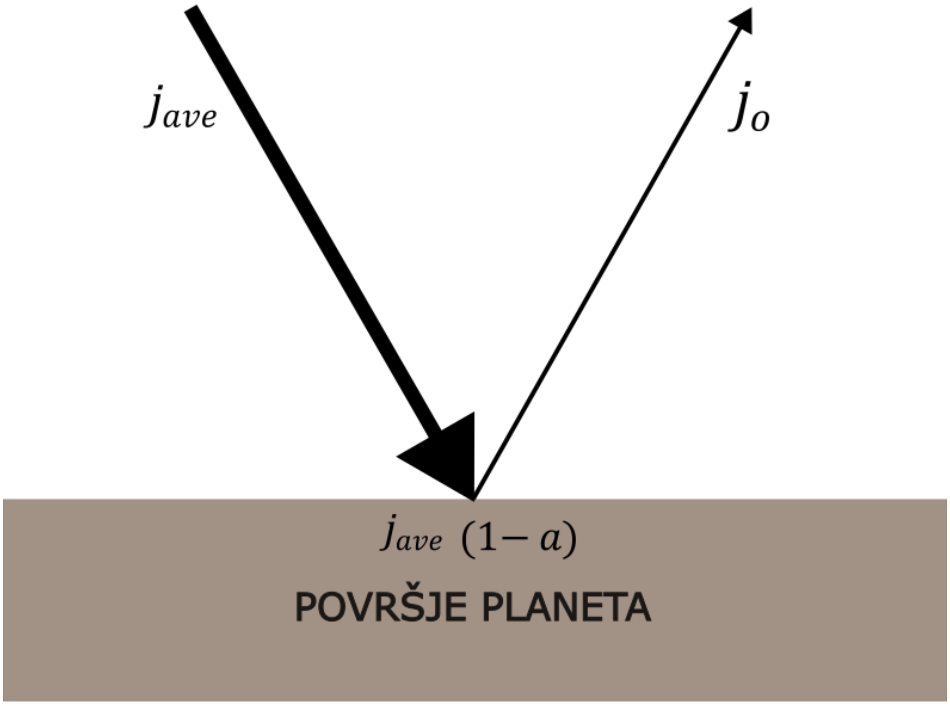
Do konca izpeljimo j_{ave}:

$$j_{ave} = \frac{j_p}{4} = \frac{\frac{j^* r_S^2}{D_p^2}}{4} = \frac{\sigma {T_S}^4 r_S^2}{4D_p^2}.$$

(25)

2.4.4. Planet brez atmosfere

Atmosfera planeta prispeva k segrevanju planeta zaradi učinka tople grede. V primeru planeta brez atmosfere se vso sončno sevanje, ki ga planet ne vpije, odbije od njegovega površja. Ker tak planet nima atmosfere, ki bi sevanje odbila nazaj proti površini planeta, je gostota svetlobnega toka na površju enaka $j_{ave}(1-a)$, kjer je a albedo ali kvocient odbojnosti.



Slika 5: Energijska shema na planetu brez atmosfere

Zakon ohranitve energije pravi, da se skupna količina energije skozi čas v izoliranem sistemu ne spreminja. Zato se mora odbiti vsa energija, ki ni bila vpita na površju planeta. Gostota svetlobnega toka, ki se odbije od površja planeta (j_o, o – odbita), je enaka razliki povprečne gostote svetlobnega toka in gostoti svetlobnega toka, ki jo planet vpije [9]:

$$j_o = j_{ave} - j_{ave}(1-a),$$

(26)

$$j_o = j_{ave}a.$$

(27)

Površinsko temperaturo planeta (T_p) dobimo iz Stefan-Boltzmannovega zakona [1]:

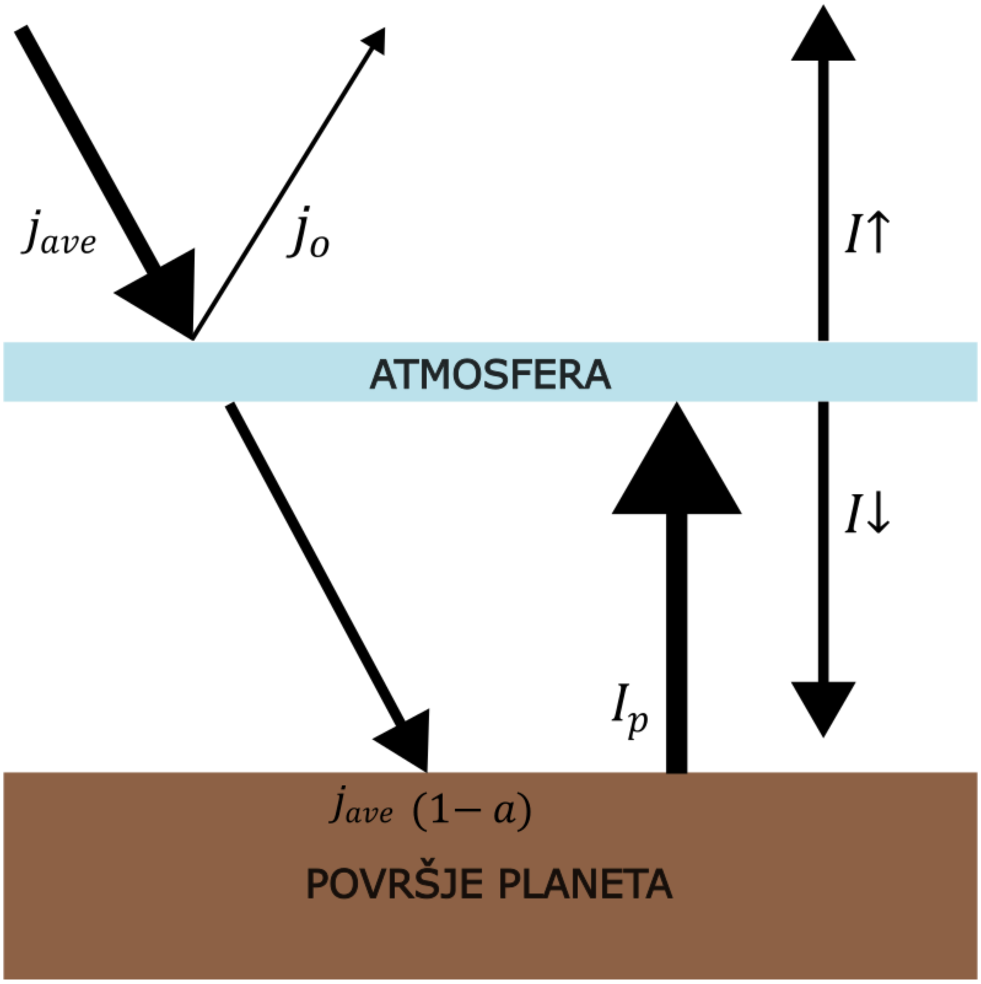
$$j^* = \varepsilon_i \sigma {T_p}^4, \tag{28}$$

$$j_{ave}(1-a) = \varepsilon_i \sigma {T_p}^4, \tag{29}$$

$$T_p = \sqrt[4]{\frac{j_{ave}(1-a)}{\varepsilon_i \sigma}}. \tag{30}$$

2.4.5. Planet z atmosfero, ki vpije vso infrardeče sevanje

V tem primeru predpostavimo, da le atmosfera odbije kratkovalovno sevanje, torej je albedo v tem primeru albedo atmosfere. Predpostavimo tudi, da le površje planeta vpije kratkovalovno sevanje in nič atmosfera. Pomagamo si s sliko 6.



Slika 6: Energijska shema na planetu z atmosfero, ki sprejme vso infrardeče sevanje

Da planet ostane v energijskem ravnovesju, mora vso energijo, ki jo vpije iz kratkovalovnega sevanja ($j_{ave}(1-a)$), tudi oddati [10]. Molekule na površju planeta vpijejo kratkovalovno sevanje in ga oddajo v obliki toplote. Površje planeta toploto odda v obliki dolgovalovnega (infrardečega) sevanja (I_p, I – infrardeče), ki se širi do atmosfere. [11]. Atmosfera vpije vso planetovo dolgovalovno sevanje, in da ostane v energijskem ravnotežju, jo mora vso oddati. Pol te energije odda nazaj proti površju planeta ($I\downarrow$) in pol v vesolje ($I\uparrow$). Zapišemo lahko:

$$\frac{I_p}{2} = I\downarrow, \tag{31}$$

$$\frac{I_p}{2} = I\uparrow. \tag{32}$$

Iz tega sledi:

$$I\downarrow = I\uparrow. \tag{33}$$

Procesu širjenja absorbirane toplote nazaj proti planetu ($I\downarrow$) pravimo učinek tople grede.

Ker mora celoten sistem obdržati energijsko ravnotežje, je energija, ki jo atmosfera odda nazaj v vesolje, enaka energiji kratkovalovnega sevanja, ki jo planet vpije:

$$I\uparrow = j_{ave}(1-a). \tag{34}$$

$I\uparrow$ in $I\downarrow$ sta enaki, zato sledi:

$$I\downarrow = j_{ave}(1-a). \tag{35}$$

Površje planeta mora oddati dovolj energije, da izenači energijo, ki jo dobi iz kratkovalovnega sevanja ($j_{ave}(1-a)$) in dolgovalovnega sevanja ($I\downarrow$):

$$I_p = j_{ave}(1-a) + I\downarrow. \tag{36}$$

Ker sta ti količini enaki, mora površje planeta oddati:

$$I_p = 2j_{ave}(1-a). \tag{37}$$

Temperatura na planetu je posledica kratkovalovnega sevanja, ki preide skozi atmosfero, in dolgovalovnega sevanja, ki ga odda atmosfera nazaj proti planetu. Po Stefan-Boltzmannovem zakonu zapišemo:

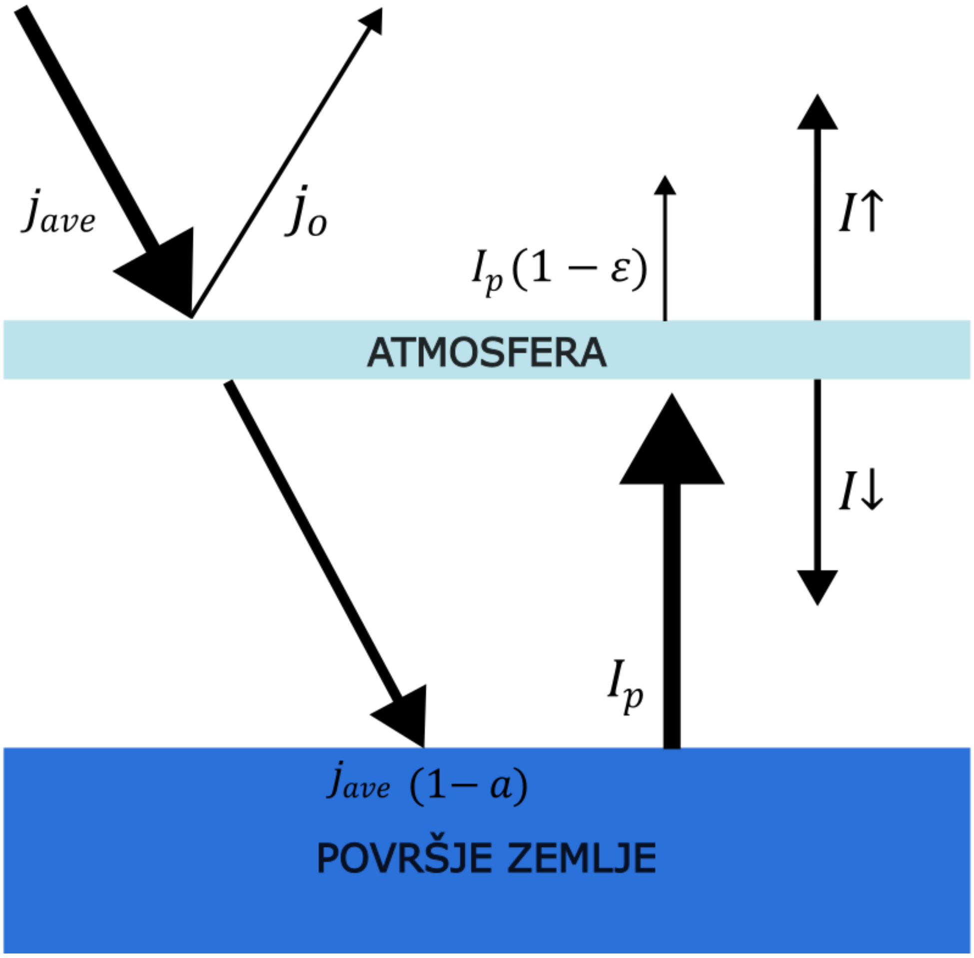
$$T_p = \sqrt[4]{\frac{I_p}{\varepsilon_i \sigma}}. \tag{38}$$

Ker planeti v infrardečem spektru sevajo kot črna telesa, je njihova izsevnost enaka 1 [10]. Sledi:

$$T_p = \sqrt[4]{\frac{2 j_{ave}(1-a)}{\sigma}}. \tag{39}$$

2.4.6. Zemljina atmosfera

Zemljina energijska shema deluje po podobnih načelih kot energijska shema planeta iz poglavja 2.4.5. Zemljino energijsko shemo predstavlja spodnja slika 7.



Slika 7: Zemljina energijska shema

Energijska shema Zemlje se od tiste v poglavju 2.4.5. razlikuje v absorpcijski sposobnosti atmosfere (ε_a). Zemljina atmosfera ni popolno črno telo, zato ne vpije vsega infrardečega valovanja, ki jo obseva. Vpije le 78 %. Ker mora atmosfera oddati enako količino energije, kot je prejme, velja Kirchhoffov zakon o toplotnem sevanju. Ta pravi, da je za vse valovne dolžine absorpcijska sposobnost ozračja (ε_a) enaka izsevnosti (ε_i):

$$\varepsilon_a = \varepsilon_i = \varepsilon. \tag{40}$$

To pomeni, da mora atmosfera oddati 78 % energije, ki je prejme od Zemlje (I_p). Ostalih 22 % odda v vesolje ($I_p(1-\varepsilon)$).

Zemljin energijski sistem mora biti v ravnotežju, zato mora tudi v tem primeru držati zakon o ohranitvi energije. Vpita energija kratkovalovnega sevanja ($j_{ave}(1-a)$) je enaka energiji, ki izstopi iz atmosfere:

$$j_{ave}(1-a) = I_p(1-\varepsilon) + I \uparrow. \tag{41}$$

Posledično mora biti tudi energija, ki jo odda Zemljino površje, enaka vsoti energije kratkovalovnega sevanja ($j_{ave}(1-a)$) in odbitega dolgovalovnega sevanja ($I \downarrow$):

$$I_p = j_{ave}(1-a) + I \downarrow \tag{42}$$

Energijsko ravnovesje izpeljemo tako, da iz enačb (41) in (42) izpostavimo $j_{ave}(1-a)$ in ju enačimo:

$$I_p - I \downarrow = I_p(1-\varepsilon) + I \uparrow. \tag{43}$$

Ker sta vrednosti $I \downarrow$ in $I \uparrow$ enaki, lahko zapišemo:

$$I_p = I_p(1-\varepsilon) + 2I \downarrow. \tag{44}$$

Dano preoblikujemo:

$$I_p - I_p(1-\varepsilon) = 2I \downarrow, \tag{45}$$

$$I_p(1-(1-\varepsilon)) = 2I \downarrow \tag{46}$$

in dobimo:

$$I_p\varepsilon = 2I \downarrow. \tag{47}$$

Po Stefan-Boltzmannovem zakonu lahko izpeljemo vrednosti gostot infrardečega toka. Ker planeti sevajo kot črna telesa, je izsevnost Zemlje enaka 1. Gostota infrardečega toka z Zemljinega površja je enaka:

$$I_p = \sigma T_z^4, \tag{48}$$

pri čemer je T_z efektivna temperatura Zemlje. Zemljina atmosfera ni popolno črno telo, zato je gostota infrardečega toka iz atmosfere enaka [12]:

$$I \downarrow = \varepsilon \sigma T_a^4, \tag{49}$$

kjer je T_a temperatura atmosfere. I_p in $I \downarrow$ iz enačb (48) in (49) vstavimo v enačbo (47) in dobimo:

$$\varepsilon \sigma T_z^4 = 2 \varepsilon \sigma T_a^4. \tag{50}$$

Izrazimo T_a:

$$T_z^4 = 2 T_a^4, \tag{51}$$

$$T_a = \frac{T_z}{\sqrt[4]{2}}. \tag{52}$$

Cilj je izraziti T_z, zato moramo v enačbo (41) najprej vstaviti enačbi (48) in (49). Dobimo:

$$j_{ave}(1-a) = \sigma T_z^4 (1-\varepsilon) + \varepsilon \sigma T_a^4, \tag{53}$$

nato zamenjamo T_a z $\frac{T_z}{\sqrt[4]{2}}$:

$$j_{ave}(1-a) = \sigma T_z^4 (1-\varepsilon) + \varepsilon \sigma \frac{T_z^4}{2}, \tag{54}$$

preoblikujemo in izrazimo T_z:

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\sigma\left(1-\frac{\varepsilon}{2}\right)}}. \tag{55}$$

Postopek preoblikovanja enačbe (54) v (55) se nahaja v prilogi. Za konec vstavimo še izpeljano vrednost j_{ave} iz enačbe (25):

$$T_z = \sqrt[4]{\frac{\frac{\sigma {T_s}^4 r_s^2}{4D_p^2}(1-a)}{\sigma(1-\frac{\varepsilon}{2})}},$$

(56)

ter preoblikujemo:

$$T_z = \sqrt[4]{\frac{{T_s}^4 r_s^2(1-a)}{4D_p^2(1-\frac{\varepsilon}{2})}}.$$

(57)

3. EKSPERIMENTALNI DEL

Eksperimentalni del je sestal iz programiranja simulacij in vpeljevanja enačb iz teoretičnega dela. Zasnovo za simulacije smo jemali iz videov [2, 13, 14]. Namen prve simulacije je zgolj ponazoriti gibanje nekega planeta okoli Sonca, zato smo izračunane vrednosti v njej zaokroževali. Posledično je prav tako nismo vključili med rezultate. V ostalih šestih simulacijah smo želeli dobiti točne rezultate, zato smo uporabili točne vrednosti. Za simulacije smo uporabljali naslednje podatke:

Tabela 4: Uporabljeni podatki v simulacijah

Oznaka	**Ime količine**	**Vrednost in vir podatka**	**Enota**
-	pi	3,14159265359	[-]
`a`	albedo Zemlje	0,306 [15]	[-]
`G`	gravitacijska konstanta	$6,67430 \cdot 10^{-11}$ [16]	[N·m^2/kg^2]
`planet.mass` `planet3.mass`	masa Zemlje	$5,972 \cdot 10^{24}$ [17]	[kg]
`planet.pos` `planet3.pos`	velika polos Zemljine orbite	149597887000 [18]	[m]
`planet.radius` `planet3.radius`	*polmer Zemlje	6371000 [17]	[m]
`planet1.mass`	masa Merkurja	$3,301 \cdot 10^{23}$ [17]	[kg]
`planet1.pos`	razdalja med Soncem in Merkurjem v afeliju	69816900000 [19]	[m]
`planet1.radius`	*polmer Merkurja	2439400 [17]	[m]
`planet2.mass`	masa Venere	$4,867 \cdot 10^{24}$ [17]	[kg]
`planet2.pos`	razdalja med Soncem in Venero v afeliju	108939000000 [19]	[m]
`planet2.radius`	*polmer Venere	6051800 [17]	[m]
`planet4.mass`	masa Marsa	$6,4171 \cdot 10^{23}$ [17]	[kg]
`planet4.pos`	razdalja med Soncem in Marsom v periheliju	206650000000 [19]	[m]
`planet4.radius`	*polmer Marsa	3389000 [17]	[m]
`planet5.mass`	masa Jupitra	$1,8982 \cdot 10^{27}$ [17]	[kg]
`planet5.pos`	razdalja med Soncem in Jupitrom v 1) periheliju in 2) afeliju	1) 740595000000 [19]; 2) 816363000000 [19]	[m]
`planet5.radius`	*polmer Jupitra	69911000 [17]	[m]
`planet6.mass`	masa Saturna	$5,6834 \cdot 10^{26}$ [17]	[kg]
`planet6.pos`	razdalja med Soncem in Saturnom v periheliju	$1,35255 \cdot 10^{12}$ [19]	[m]
`planet6.radius`	*polmer Saturna	58232000 [17]	[m]
`planet7.mass`	masa Urana	$8,6810 \cdot 10^{25}$ [17]	[kg]

`planet7.pos`	razdalja med Soncem in Uranom v periheliju	$2{,}9415 \cdot 10^{12}$ [19]	[m]
`planet7.radius`	*polmer Urana	25362000 [17]	[m]
`planet8.mass`	masa Neptuna	$1{,}02413 \cdot 10^{26}$ [17]	[kg]
`planet8.pos`	razdalja med Soncem in Neptunom v periheliju	$4{,}4736 \cdot 10^{12}$ [19]	[m]
`planet8.radius`	*polmer Neptuna	24622000 [17]	[m]
`star.mass`	masa Sonca	$1{,}9885 \cdot 10^{30}$ [20]	[kg]
`star.radius`	polmer Sonca	696340000 [21]	[m]
`Ts`	površinska temperatura Sonca	5772 [20]	[K]
`ε`	izsevnost Zemljine atmosfere	0,78 [22]	[-]

*Polmeri planetov so pomnoženi z 10^3, saj se drugače planeti v simulaciji ne opazijo. To ne vpliva na fizikalne zakone, saj je masa teles točkasta in se nahaja v njihovem središču.

Po vrsti si sledijo:
- simulacija Osončja z Zemljo in Soncem,
- simulacija Osončja s spodnjima planetoma in Zemljo,
- simulacija Osončja z vsemi planeti,
- simulacija Osončja z zgornjimi planeti in Zemljo,
- simulacija Osončja brez Jupitra in
- simulacija Osončja z Jupitrom na drugi strani.

Vsi programi se nahajajo v prilogi.

3.1. PROGRAM

Vseh sedem programov smo izdelali v integriranem razvojnem okolju Pycharm in knjižnici Vpython, ki nam je omogočila ustvarjanje predmetov v 3D-prostoru. Osnova programa je temeljila na Newtonovem splošnem gravitacijskem zakonu iz enačb (7) in (8). Premikanje planetov smo omogočili z Eulerjevo metodo. Določili smo časovni korak (dt) in posodobili vrednost gibalne količine:

$$p_2 = p_1 + \frac{dp}{dt}\,dt, \tag{58}$$

kjer je p_1 prvotna gibalna količina telesa in p_2 posodobljena gibalna količina. Gibalno količino lahko zamenjamo z njeno definicijo iz enačbe (6) in dobimo:

$$\frac{dp}{dt} = \frac{dmv}{dt} = m\frac{dv}{dt} = ma = F, \tag{59}$$

torej je odvod gibalne količine nekega telesa enak sili, ki deluje na telo:

$$p_2 = p_1 + F\,dt.$$

(60)

Posodobili smo še pozicije teles:

$$r_2 = r_1 + \frac{dr}{dt}\,dt,$$

(61)

kjer je r_1 prvotna pozicija in r_2 posodobljena pozicija. Odvod pozicije lahko zapišemo kot:

$$\frac{dr}{dt} = v = \frac{p}{m}.$$

(62)

$\frac{dr}{dt}$ zamenjamo z $\frac{p}{m}$ in dobimo:

$$r_2 = r_1 + \frac{p}{m}\,dt.$$

(63)

Za simulaciji 6 in 7 smo morali ugotoviti, kateri planet najbolj vpliva na Zemljino orbito. Izračunali smo sile med Zemljo in vsakim planetom.

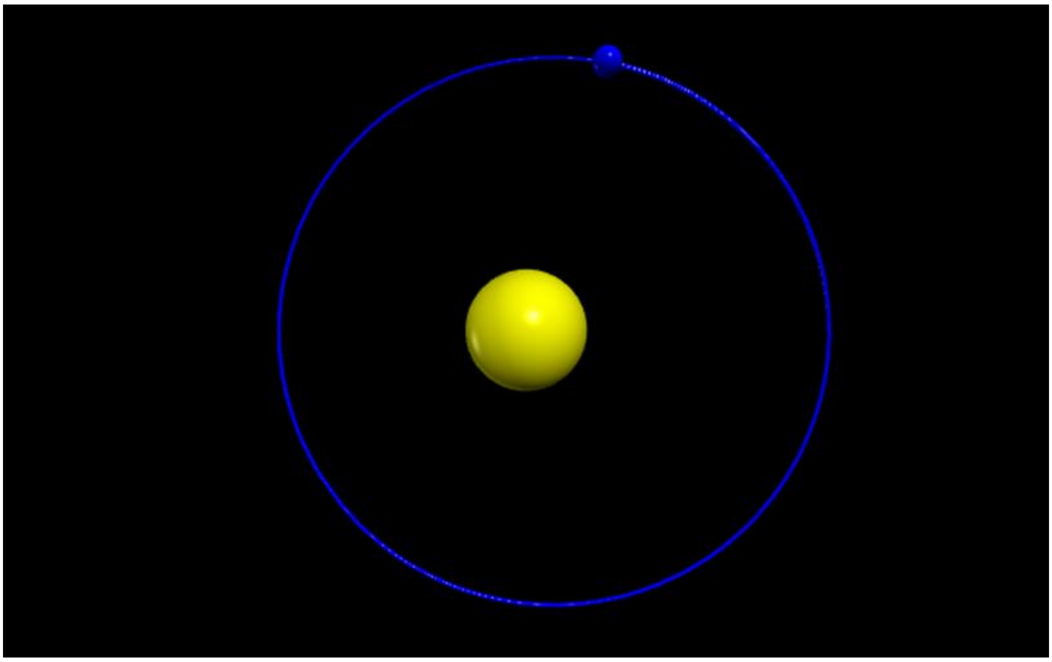
Uporabili smo enačbo (1) in podatke iz tabele 4. Razdaljo med spodnjima planetoma in Zemljo smo zapisali kot razliko med veliko polosjo Zemljine orbite in razdaljo v njunem afeliju. Razdaljo med zgornjimi planeti in Zemljo smo zapisali kot razliko med razdaljo v njihovem periheliju in veliko polosjo Zemljine orbite. Izračunane sile prikazuje tabela 9.

Tabela 5: Izračunane sile med Zemljo in ostalimi planeti

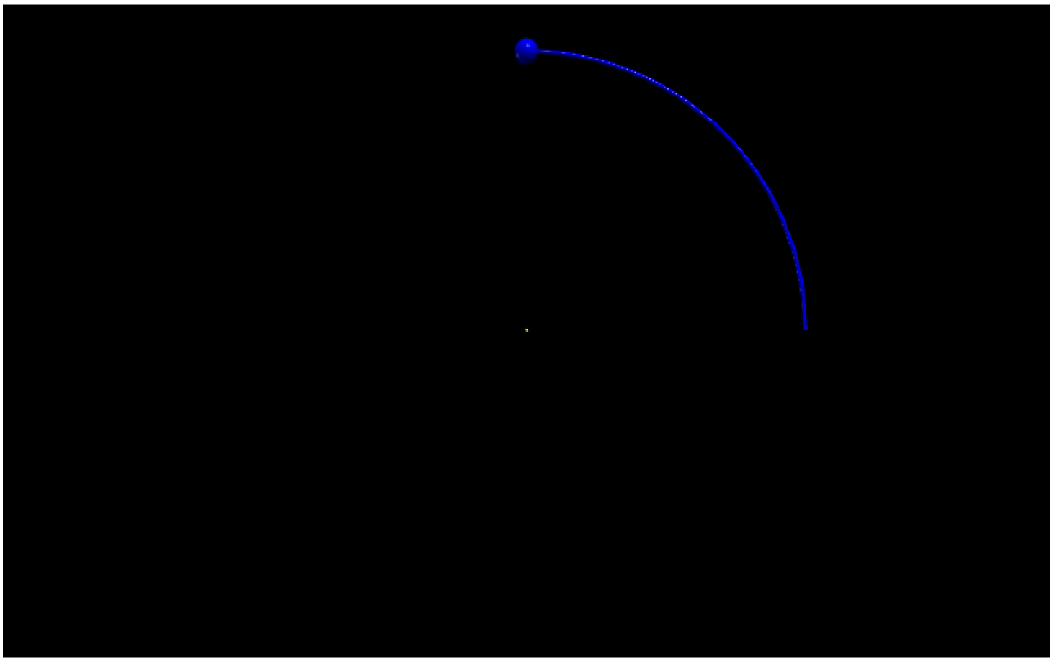
Ime planeta	**Spodnji/zgornji planet**	**Gravitacijska sila [$\cdot 10^{15}$ N]**
Merkur	spodnji	20,672
Venera	spodnji	1173,481
Mars	zgornji	78,581
Jupiter	zgornji	2166,191
Saturn	zgornji	156,544
Uran	zgornji	4,439
Neptun	zgornji	2,183

Največja sila je med Zemljo in Jupitrom.

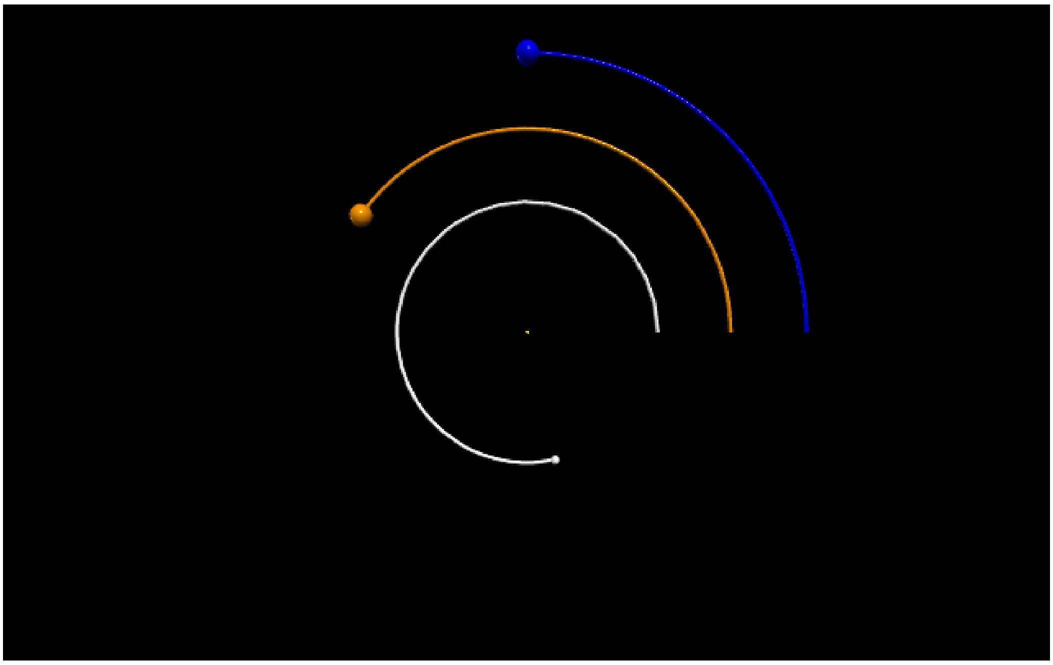
Ko smo zagnali program, so se nam izrisale simulacije z različnim številom krogel – nebesnih teles.



Slika 8: Simulacija poljubnega planeta in poljubne zvezde

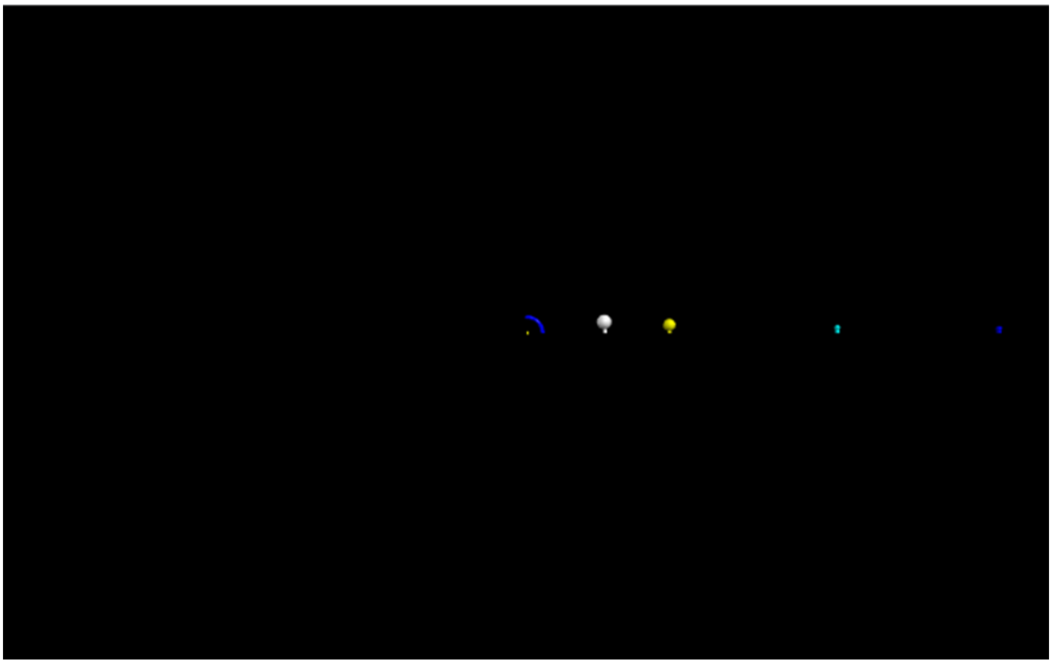


Slika 9: Simulacija Osončja z Zemljo in s Soncem

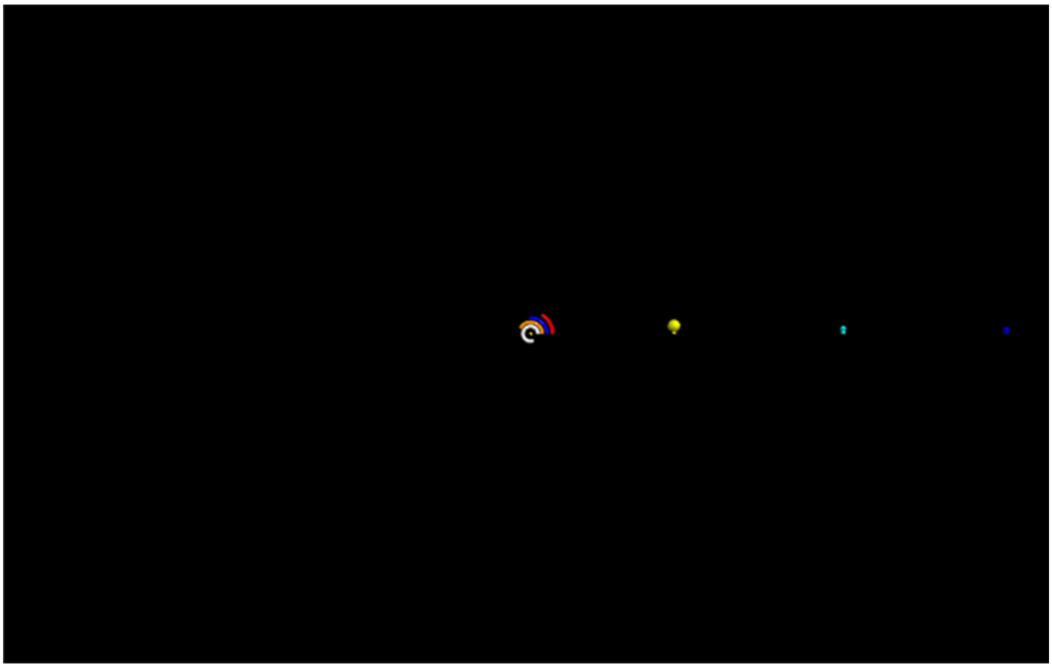


Slika 10: Simulacija Osončja s spodnjima planetoma in z Zemljo

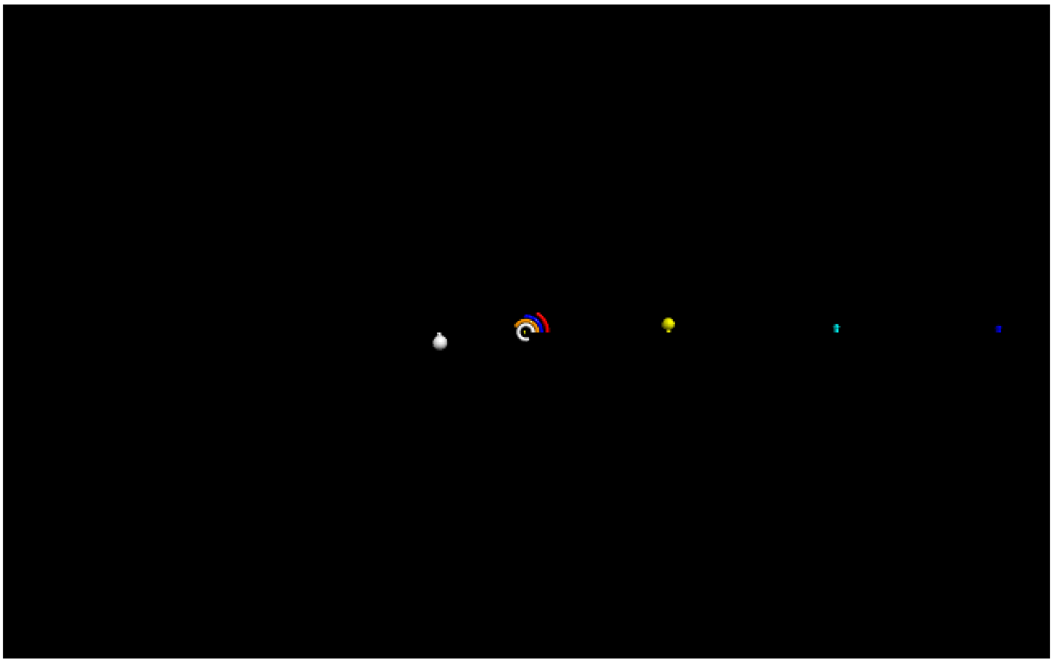
Slika 11: Simulacija Osončja z vsemi planeti



Slika 12: Simulacija Osončja z zgornjimi planeti in Zemljo



Slika 13: Simulacija Osončja brez Jupitra



Slika 14: Simulacija Osončja z Jupitrom na drugi strani

3.2. MERJENJE SPREMENLJIVK

Ob simulacijah smo merili vrednost sedmih spremenljivk, da bi določili vpliv določenih planetov na Zemljino orbito in površinsko temperaturo. Merili smo:

- veliko polos Zemljine orbite,
- malo polos Zemljine orbite,
- povprečno razdaljo med Zemljo in Soncem,
- povprečno orbitalno hitrost Zemlje,
- obseg Zemljine orbite,
- dolžino leta in
- efektivno temperaturo.

Vsako telo je v simulacijah imelo svojo x in y koordinato v koordinatnem sistemu. Ko je bila Zemlja v poziciji, v kateri je imela maksimalno vrednost na x koordinati, je bila od Sonca oddaljena za toliko, kolikor je vrednost velike polosi Zemljine orbite. Ko pa je dosegla svojo maksimalno y vrednost, je bila od Sonca oddaljena za toliko, kolikor je vrednost male polosi Zemljine orbite.

Razdaljo med Zemljo in Soncem (D) smo definirali s pomočjo Pitagorovega izreka:

$$D = \sqrt{x^2 + y^2},$$

(64)

kjer je x vrednost Zemljine pozicije na x osi, y pa vrednost Zemljine pozicije na y osi. Povprečno razdaljo med Zemljo in Soncem smo dobili tako, da smo sešteli vrednosti D v vsakem časovnem intervalu in ta seštevek delili s številom podatkov spremenljivke D.

Hitrost smo izračunali s pomočjo enačbe (5). Povprečno hitrost smo dobili tako, da smo sešteli vse vrednosti v v vsakem časovnem intervalu in ta seštevek delili s številom podatkov spremenljivke v.

Obseg orbite smo izračunali z enačbo (9), dolžino leta ali obhodni čas z enačbo (10), efektivno temperaturo pa z enačbo (57).

Podatke za določene simulacije smo zbrali v tabelah.

Tabela 6: Vrednost spremenljivk v simulaciji Osončja z Zemljo in s Soncem

Merjena količina	Vrednost	Enota
velika polos	149597885517,41223	[m]
mala polos	149590248503,66827	[m]
povprečna razdalja	149593105932,90155	[m]
povprečna hitrost	29785,841269087694	[m/s]
obseg orbite	939927244035,033	[m]
dolžina leta	365,2335164356086	[dan]
efektivna temperatura	14,439890445310198	[°C]

Tabela 7: Vrednost spremenljivk v simulaciji Osončja s spodnjima planetoma in z Zemljo

Merjena količina	Vrednost	Enota
velika polos	149597885517,36224	[m]
mala polos	149588813978,43918	[m]
povprečna razdalja	149592265234,72498	[m]
povprečna hitrost	29785,92496749779	[m/s]
obseg orbite	939922737403,8765	[m]
dolžina leta	365,23073896445203	[dan]
efektivna temperatura	14,440698561814202	[°C]

Tabela 8: Vrednost spremenljivk v simulaciji Osončja z vsemi planeti

Merjena količina	Vrednost	Enota
velika polos	149597885517,46304	[m]
mala polos	149589279467,3238	[m]
povprečna razdalja	149593812919,183	[m]
povprečna hitrost	29785,770884067948	[m/s]
obseg orbite	939924199759,0519	[m]
dolžina leta	365,2331965611465	[dan]
efektivna temperatura	14,439210863904066	[°C]

Tabela 9: Vrednost spremenljivk v simulaciji Osončja z zgornjimi planeti in Zemljo

Merjena količina	Vrednost	Enota
velika polos	149597885517,50977	[m]
mala polos	149590652485,8224	[m]
povprečna razdalja	149594599544,94357	[m]
povprečna hitrost	29785,692570570813	[m/s]
obseg orbite	939928513166,9371	[m]
dolžina leta	365,235832940713	[dan]
efektivna temperatura	14,438454735742312	[°C]

Tabela 10: Vrednost spremenljivk v simulaciji Osončja brez Jupitra

Merjena količina	Vrednost	Enota
velika polos	149597885517,37238	[m]
mala polos	149588888052,43835	[m]
povprečna razdalja	149592422780,03467	[m]
povprečna hitrost	29785,90928254045	[m/s]
obseg orbite	939922970110,7263	[m]
dolžina leta	365,2310217154266	[dan]
efektivna temperatura	14,440547121731981	[°C]

Tabela 11: Vrednost spremenljivk v simulaciji Osončja z Jupitrom na drugi strani

Merjena količina	Vrednost	Enota
velika polos	149597885517,33844	[m]
mala polos	149588814241,05194	[m]
povprečna razdalja	149591890312,3195	[m]
povprečna hitrost	29785,962294283363	[m/s]
obseg orbite	939922738228,8115	[m]
dolžina leta	365,2302815898847	[dan]
efektivna temperatura	14,441058956113352	[°C]

4. REZULTATI IN UGOTOVITVE

Primerjavo rezultatov prikazuje tabela 12.

Tabela 12: Primerjava rezultatov

	Velika polos $[\cdot 10^{-6}]$	Mala polos $[\cdot 10^{-6}]$	Povprečna razdalja $[\cdot 10^{-6}]$	Povprečna hitrost $[\cdot 10^{-6}]$	Obseg orbite $[\cdot 10^{-6}]$	Dolžina leta $[\cdot 10^{-6}]$	Efektivna temperatura $[\cdot 10^{-6}]$
2	-0,000000339643838041205375 66	6,477979892146454028156181 9	-4,726039584484315144098030 96	2,3630417362690486490114234	3,2388526456499315488943941	0,875808839699617377971028 3	47,0649963171363471651762255
3	-0,000000673806315185071873	-3,111779709599317484157073 02	-10,345912226036551313600088 31	5,1730549610994752074419137	-1,55582245437969621839731808	-6,728842606886515107719716 01	103,03180167934447950467345 11
4	0	0	0	0	0	0	0
5	0,000000312370725283714371 3	9,178588890121109289060887 5	5,258411061391750250717234 4	-2,62922512362736770359492864	4,5891018513043236590549103	7,218345953551961449788792 6	-52,36630788765685203313576822
6	-0,000000606024608478954523 87	-2,616597170892185843274331 99	-9,292758311341478062650119 31	4,6464626697315960645565368	-1,30824201134008298432210964	-5,954677012870850424004359 48	92,54368819112205508843430 54
7	-0,000000832899472937102731 9	-3,110024150905972985583059 48	-12,852181691088218787020801 31	6,4262300331257510293856852	-1,55494479318083414973443065	-7,981123537635446644308451 19	127,99121965217383293046641 7

Vrednosti iz simulacij 2, 3, 5, 6 in 7 v tabeli predstavljajo za kakšen delež se njihova dejanska vrednost razlikuje od vrednosti iz 4. simulacije. Vrednosti z negativnim predznakom so za tolikšno vrednost manjše od vrednosti iz 4. simulacije in so pobarvane rdeče. Vrednosti, ki so se povečale, so pobarvane zeleno. V rezultatih ni podatkov za simulacijo 1, saj je njen namen le prikazati sistem poljubnega planeta in poljubne zvezde.

Vpliv zgornjih planetov se opazi v simulaciji 3, saj so tam ti planeti odsotni. Vpliv spodnjih planetov se opazi v simulaciji 5, saj tam ti manjkajo. Vsi rezultati so pomnoženi z 10^6, da so rezultati bolj berljivi. Dejanski deleži so torej zelo manjši – za 10^6, kar prikazuje tabela 13.

Tabela 13: Dejanske vrednosti rezultatov

	Velika polos	Mala polos	Povprečna razdalja	Povprečna hitrost	Obseg orbite	Dolžina leta	Efektivna temperatura
2	-0,00000000000033964383804120537566	0,0000064779798921464540281561819	-0,00000472603958448431514409803096	0,0000023630417362690486490114234	0,0000032388526456499315488943941	0,0000008758088396996173779710283	0,0000470649963171363471651762255
3	-0,000000000000673806315185071873	-0,00000311177970959931748415707302	-0,00001034591222603655131360008831	0,0000051730549610994752074419137	-0,00000155582245437969621839731808	-0,00000672884260688651510771971601	0,0001030318016793444795046734511
4	0	0	0	0	0	0	0
5	0,0000000000003123707252837143713	0,0000091785888901211092890608875	0,00000525841106139175025071 72344	-0,00000262922512362736770359492864	0,0000045891018513043236590549103	0,0000072183459535519614497887926	-0,000052366307887656852033135 76822
6	-0,000000000000606024608478954 52387	-0,00000261659717089218584327433199	-0,00000929275831134147806265011931	0,0000046464626697315960645565368	-0,00000130824201134008298432210964	-0,00000595467701287085042400435948	0,0000925436881911220550884343054
7	-0,000000000000832899472937102 7319	-0,00000311002415090597298558305948	-0,00001285218169108821878702080131	0,0000064262300331257510293856852	-0,00000155494479318083414973443065	-0,00000798112353763544664430845119	0,0001279912196521738329304664 17

Na podlagi rezultatov lahko odgovorimo na raziskovalno vprašanje.

Če primerjamo vrednosti med simulacijo 2 – z zgolj Zemljo, in 4 – z vsemi planeti, lahko opazimo, kako se je Zemljina orbita spremenila brez ostalih planetov. Opazimo, da so se nekatere spremenljivke spremenile tako, da so nakazovale na daljšo orbito, nekatere pa na krajšo. Velika polos se je pomanjšala, mala pa povečala, kar nakazuje na spremembo v ekscentričnosti. Povprečna razdalja se je zmanjšala, kar nakazuje, da je bila Zemlja v povprečju bližje Soncu. Po drugi strani pa sta se povečala obseg orbite in dolžina leta, kar pomeni, da je Zemlja prepotovala daljšo pot in jo je potovala dlje časa. Povprečna razdalja in efektivna temperatura sta se povečali, kar nakazuje na manjšo razdaljo med Zemljo in Soncem, saj sta ti količini obratno sorazmerni z razdaljo. Ker nekatere spremenljivke nakazujejo na daljšo orbito, nekatere pa na krajšo, tu ne pridemo do enovitega odgovora.

Ko primerjamo rezultate simulacij 3 – s spodnjima planetoma, in 5 – z zgornjimi planeti, razvidimo, da se orbita v simulaciji 5 v vseh parametrih poveča. To je predvidljivo, saj so ti planeti bolj oddaljeni od Sonca, kot Zemlja. Če pa primerjamo velikosti sprememb spremenljivk, pa opazimo razlike. Brez spodnjih planetov, torej v simulaciji 5, se bolj spremeni mala polos, obseg orbite in dolžina leta. Brez zgornjih planetov se bolj spremeni velika polos, povprečna razdalja, povprečna hitrost in efektivna temperatura. Ker obe skupini planetov različno vplivata na različne parametre Zemljine orbite, ne moremo določiti, katera skupina bolj vpliva na Zemljino orbito. Razlog za različne rezultate se nahaja v kotni hitrosti planetov. Spodnja planeta imata večjo orbitalno hitrost, zato bolj vplivata na Zemljino y vrednost, kot zgornji planeti z manjšo kotno hitrostjo. Odgovor na vprašanje »Ali Zgornji planeti bolj vplivajo na Zemljino orbito kot spodnja planeta?« ni enovit, ampak odvisen od naše interpretacije kaj je »večji vpliv na orbito«.

S pomočjo Newtonovega splošnega gravitacijskega zakona sem dokazal, da, ko je najbližje Zemlji, Jupiter deluje na Zemljo z največjo silo.

Vpliv planeta, ki najbolj vpliva na Zemljo, sem meril v simulaciji 6. Iz primerjave med simulacijo 3 – brez zgornjih planetov in simulacijo 6 opazimo, da so rezultati zelo primerljivi. To nakazuje, da Jupiter prispeva večino gravitacijskega vpliva zgornjim planetom. Gravitacijska sila ostalih planetov na Zemljo je torej neprimerljivo manjša od sile med Jupitrom in Zemljo. Kot pričakovano sta se Zemljini orbiti v simulacijama brez Jupitra in z Jupitrom v najbolj oddaljeni točki zmanjšali.

Rezultati nakazujejo, da se Zemljina orbita ob odsotnosti mnogih planetov spremeni za zelo majhen delež. Če te deleže pretvorimo v dejanske vrednosti, opazimo, da se najbolj spremeni efektivna temperatura v simulaciji 7, in še to le za 0,0018 °C. Iz tega bi lahko sklepali, da se življenje na Zemlji kljub odsotnosti mnogih planetov praktično ne bi spremenilo.

5. ZAKLJUČEK

Iz rezultatov lahko razberemo, da se ob odsotnosti mnogih planetov Zemljina orbita in površinska temperatura skoraj ne spremenita. Kljub temu igrajo vsi planeti v našem Osončju pomembno vlogo. Vsi planeti pripomorejo k vzdrževanju ravnotežja. Čeprav se na prvi pogled zdi njihova naloga nepomembna, se spremembe ob njihovi odsotnosti čez milijone let naberejo. Zato lahko trdimo, da bi ob odsotnosti enega samega planeta pred 4,5 milijarde let naše Osončje danes izgledalo popolnoma drugače. Kdo ve, ali bi se življenje na Zemlji razvilo, kot se je, če en sam planet ne bi nikoli obstajal? Jupiter zaradi svoje velike mase poleg Zemlje privlači tudi asteroide, komete in ostala manjša vesoljska telesa, ki bi drugače lahko trčila v Zemljo.

V času raziskovanja sem obogatil svoje znanje s področja astronomije in programiranja tako v teoriji kot tudi v praktičnem delu. Tema, ki sem jo raziskoval, je dokaj neraziskana. Z univerzitetnim znanjem iz fizike in astronomije bi se zagotovo lahko globlje poglobil v temo, zato se veselim nadaljnjega raziskovanja.

6. VIRI IN LITERATURA

1. Giancoli, D. C. (2014). *Physics: Principles with applications.* (7. izdaja). Pearson Education, Inc.
2. Let's Code Physics. (17. 9. 2018). *Let's Build a Solar System 1 Force Function* [Video]. https://www.youtube.com/watch?v=4ycpvtIio-o&list=LL&index=25&t=127s
3. Raman, M. (31. 10. 2022). *Ellipse Circumference Calculator*. https://www.omnicalculator.com/math/ellipse-circumference Pridobljeno: 20. 12. 2022.
4. Mohorič, A. in Babič, V. (2012). *FIZIKA 1, učbenik za fiziko v 1. letniku gimnazij in štiriletnih strokovnih šol*. Mladinska knjiga.
5. *Average value of a function. (b. d.).* https://www.whitman.edu/mathematics/calculus_online/section09.04.html
6. Elert, G. (b. d.). *Intensity - The Physics Hypertextbook.* https://physics.info/intensity/
7. Avsec, F. in Prosen M. (2006). *Astronomija*. DMFA – založništvo.
8. *Energy from the Sun - American Chemical Society*. (b. d.). https://www.acs.org/climatescience/energybalance/energyfromsun.html
9. *Predicted Planetary Temperatures - American Chemical Society. (b .d.).* https://www.acs.org/climatescience/energybalance/predictedplanetarytemperatures.html
10. *How do we calculate the Earth's effective temperature?* (10. 10. 2001). https://atmos.washington.edu/academics/classes/2001Q4/211/notes_greenhouse.html
11. National Aeronautics and Space Administration, Science Mission Directorate. (2010). *The Earth's Radiation Budget*. https://science.nasa.gov/ems/13_radiationbudget
12. Marshall, J. in Plumb, R.A. (2007). *Atmosphere, Ocean and Climate Dynamics, Chapter 2, The global energy balance.* *http://www.geo.utexas.edu/courses/387H/Lectures/chap2.pdf*
13. Let's Code Physics. (1. 1. 2018). *Euler-Cromer Method for Beginners 1. The ECM Reverses the Derivative* [Video]. https://www.youtube.com/watch?v=rPkJtpVJwSw&list=PLdCdV2GBGyXOOutOEKggaZo1rCHtUYh-A&index=2
14. Let's Code Physics. (24. 9. 2018). *Let's Build a Solar System 2 Multiple Planets* [Video]. https://www.youtube.com/watch?v=OTJWGvibBfk&list=LL&index=24
15. Williams, David R. (26. 7. 2018). *Earth Fact Sheet*. NASA/Goddard Space Flight Center. https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html
16. NIST. *The NIST Reference on Constants, Units, and Uncertainty.* (20. 5. 2019) https://physics.nist.gov/cgi-bin/cuu/Value?bg
17. Jet Propulsion Laboratory. (11. 8. 2020). *Planetary Physical Parameters.* *https://ssd.jpl.nasa.gov/planets/phys_par.html*
18. *Earth - McGill School Of Computer Science* (b. d.). https://www.cs.mcgill.ca/~rwest/wikispeedia/wpcd/wp/e/Earth.htm

19. Yeomans, Donald K. (4. 2. 2021). *HORIZONS Web-Interface for Venus (Major Body=2).* https://ssd.jpl.nasa.gov/horizons/app.html#/
20. NASA Goddard Space Flight Center. (12. 8. 2013). *Sun Fact Sheet.* https://nssdc.gsfc.nasa.gov/planetary/factsheet/sunfact.html
21. Emilio, M. idr. (2012). *Measuring the Solar Radius from Space during the 2003 and 2006 Mercury Transits, The Astrophysical Journal. https://ui.adsabs.harvard.edu/abs/2012ApJ...750..135E/abstract*
22. *A Single-Layer Atmosphere Model - American Chemical Society* (b. d.). https://www.acs.org/climatescience/atmosphericwarming/singlelayermodel.html

6.1. VIRI SLIK

1. https://si.openprof.com/wb/elipsa?ch=108
2. https://www.differencebetween.com/difference-between-perihelion-and-aphelion/
3. https://www.chegg.com/learn/physics/introduction-to-physics/inverse-square-law
4. https://www.acs.org/climatescience/energybalance/energyfromsun.html
5. Lasten vir
6. Lasten vir
7. Lasten vir
8. Posnetek zaslona
9. Posnetek zaslona
10. Posnetek zaslona
11. Posnetek zaslona
12. Posnetek zaslona
13. Posnetek zaslona
14. Posnetek zaslona

7. PRILOGA

Preoblikovana enačba (54)

$$j_{ave}(1-a) = \sigma T_z^4(1-\varepsilon) + \varepsilon\sigma\frac{T_z^4}{2}$$

$$j_{ave}(1-a) = T_z^4(\sigma(1-\varepsilon) + \frac{\varepsilon\sigma}{2})$$

$$T_z^4 = \frac{j_{ave}(1-a)}{\sigma(1-\varepsilon) + \frac{\varepsilon\sigma}{2}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\frac{2\sigma(1-\varepsilon)}{2} + \frac{\varepsilon\sigma}{2}}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\frac{2\sigma(1-\varepsilon) + \varepsilon\sigma}{2}}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\frac{\sigma((2-2\varepsilon) + \varepsilon)}{2}}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\frac{\sigma(2-\varepsilon)}{2}}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\frac{\sigma(2-\varepsilon)}{2}}}$$

$$T_z = \sqrt[4]{\frac{j_{ave}(1-a)}{\sigma(1-\frac{\varepsilon}{2})}}$$

Tabele:

Tabela 5: Izračunane sile med Zemljo in ostalimi planeti

Ime planeta	Spodnji/zgornji planet	Gravitacijska sila [$\cdot 10^{15}$ N]
Merkur	spodnji	20,672
Venera	spodnji	1173,481
Mars	zgornji	78,581
Jupiter	zgornji	2166,191
Saturn	zgornji	156,544
Uran	zgornji	4,439
Neptun	zgornji	2,183

Tabela 6: Vrednost spremenljivk v simulaciji Osončja z Zemljo in s Soncem

Merjena količina	Vrednost	Enota
velika polos	149597885517,41223	[m]
mala polos	149590248503,66827	[m]
povprečna razdalja	149593105932,90155	[m]
povprečna hitrost	29785,841269087694	[m/s]
obseg orbite	939927244035,033	[m]
dolžina leta	365,2335164356086	[dan]
efektivna temperatura	14,439890445310198	[°C]

Tabela 7: Vrednost spremenljivk v simulaciji Osončja s spodnjima planetoma in z Zemljo

Merjena količina	Vrednost	Enota
velika polos	149597885517,36224	[m]
mala polos	149588813978,43918	[m]
povprečna razdalja	149592265234,72498	[m]
povprečna hitrost	29785,92496749779	[m/s]
obseg orbite	939922737403,8765	[m]
dolžina leta	365,23073896445203	[dan]
efektivna temperatura	14,440698561814202	[°C]

Tabela 8: Vrednost spremenljivk v simulaciji Osončja z vsemi planeti

Merjena količina	Vrednost	Enota
velika polos	149597885517,46304	[m]
mala polos	149589279467,3238	[m]
povprečna razdalja	149593812919,183	[m]
povprečna hitrost	29785,770884067948	[m/s]
obseg orbite	939924199759,0519	[m]
dolžina leta	365,2331965611465	[dan]
efektivna temperatura	14,439210863904066	[°C]

Tabela 9: Vrednost spremenljivk v simulaciji Osončja z zgornjimi planeti in Zemljo

Merjena količina	**Vrednost**	**Enota**
velika polos	149597885517,50977	[m]
mala polos	149590652485,8224	[m]
povprečna razdalja	149594599544,94357	[m]
povprečna hitrost	29785,692570570813	[m/s]
obseg orbite	939928513166,9371	[m]
dolžina leta	365.235832940713	[dan]
efektivna temperatura	14,438454735742312	[°C]

Tabela 10: Vrednost spremenljivk v simulaciji Osončja brez Jupitra

Merjena količina	**Vrednost**	**Enota**
velika polos	149597885517,37238	[m]
mala polos	149588888052,43835	[m]
povprečna razdalja	149592422780,03467	[m]
povprečna hitrost	29785,90928254045	[m/s]
obseg orbite	939922970110,7263	[m]
dolžina leta	365,2310217154266	[dan]
efektivna temperatura	14,440547121731981	[°C]

Tabela 11: Vrednost spremenljivk v simulaciji Osončja z Jupitrom na drugi strani

Merjena količina	**Vrednost**	**Enota**
velika polos	149597885517,33844	[m]
mala polos	149588814241,05194	[m]
povprečna razdalja	149591890312,3195	[m]
povprečna hitrost	29785,962294283363	[m/s]
obseg orbite	939922738228,8115	[m]
dolžina leta	365,2302815898847	[dan]
efektivna temperatura	14,441058956113352	[°C]

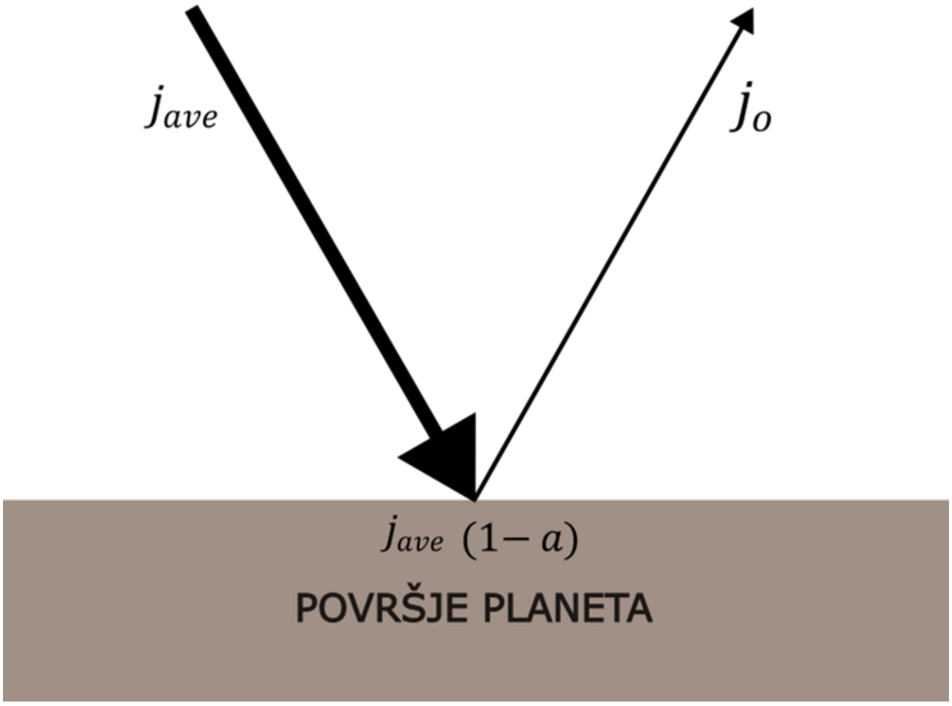
Tabela 12: Primerjava rezultatov

	Velika polos $[\cdot 10^{-6}]$	Mala polos $[\cdot 10^{-6}]$	Povprečna razdalja $[\cdot 10^{-6}]$	Povprečna hitrost $[\cdot 10^{-6}]$	Obseg orbite $[\cdot 10^{-6}]$	Dolžina leta $[\cdot 10^{-6}]$	Efektivna temperatura $[\cdot 10^{-6}]$
2	-0,00000033964383804120537566	6,4779798921464540281561819	-4,72603958448431514409803096	2,3630417362690486490114234	3,2388526456499315488943941	0,8758088396996173779710283	47,0649963171363471651762255
3	-0,000000673806315185071873	-3,11177970959931748415707302	-10,34591222603655131360008831	5,1730549610994752074419137	-1,55582245437969621839731808	-6,72884260688651510771971601	103,0318016793444795046734511
4	0	0	0	0	0	0	0
5	0,0000003123707252837143713	9,1785888901211092890608875	5,2584110613917502507172344	-2,62922512362736770359492864	4,5891018513043236590549103	7,2183459535519614497887926	-52,36630788765685203313576822
6	-0,00000060602460847895452387	-2,61659717089218584327433199	-9,29275831134147806265011931	4,6464626697315960645565368	-1,30824201134008298432210964	-5,95467701287085042400435948	92,5436881911220550884343054
7	-0,0000008328994729371027319	-3,11002415090597298558305948	-12,85218169108821878702080131	6,4262300331257510293856852	-1,55494479318083414973443065	-7,98112353763544664430845119	127,991219652173832930466417

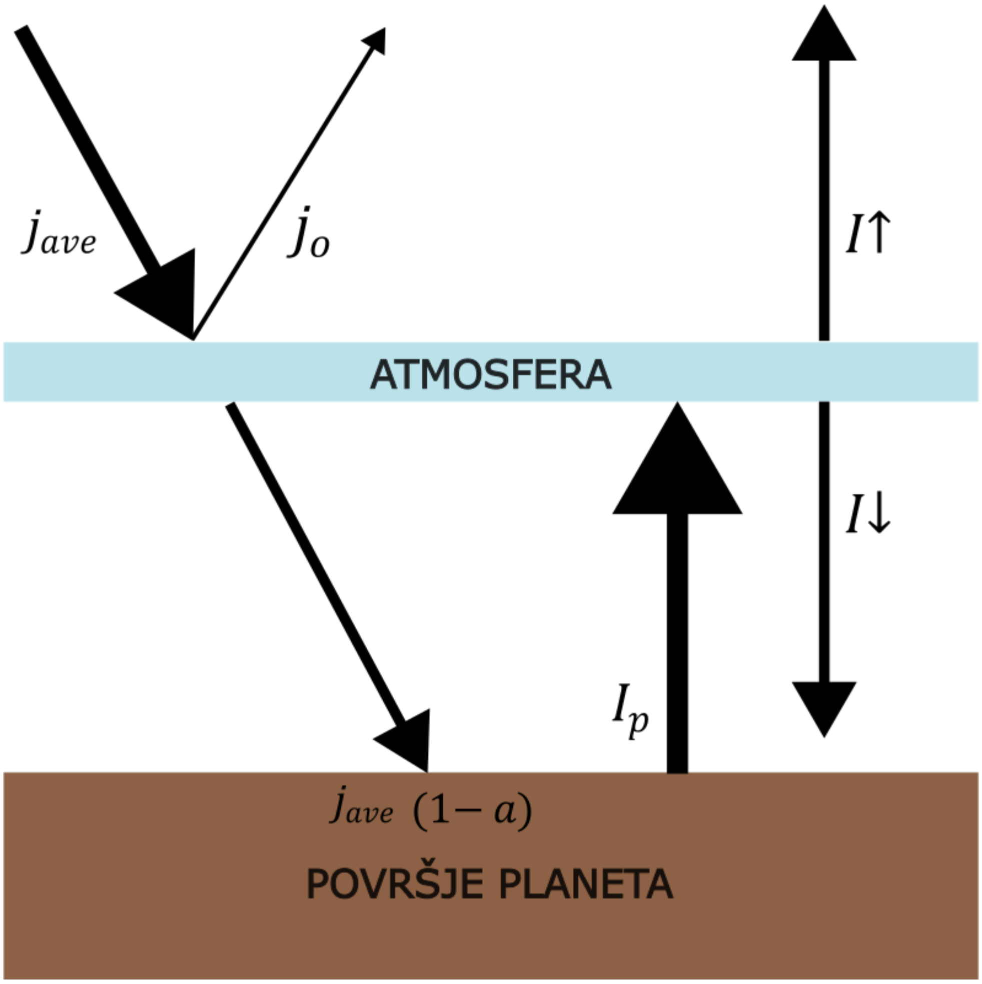
Tabela 13: Dejanske vrednosti rezultatov

	Velika polos	Mala polos	Povprečna razdalja	Povprečna hitrost	Obseg orbite	Dolžina leta	Efektivna temperatura
2	-0,00000000000033964383804120537566	0,0000064779798921464540281561819	-0,00000472603958448431514409803096	0,0000023630417362690486490114234	0,0000032388526456499315488943941	0,0000008758088396996173779710283	0,0000470649963171363471651762255
3	-0,000000000000673806315185071873	-0,00000311177970959931748415707302	-0,00001034591222603655131360008831	0,0000051730549610994752074419137	-0,00000155582245437969621839731808	-0,00000672884260688651510771971601	0,0001030318016793444795046734511
4	0	0	0	0	0	0	0
5	0,0000000000003123707252837143713	0,0000091785888901211092890608875	0,0000052584110613917502507172344	-0,00000262922512362736770359492864	0,0000045891018513043236590549103	0,0000072183459535519614497887926	-0,00005236630788765685203313576822
6	-0,00000000000060602460847895452387	-0,00000261659717089218584327433199	-0,00000929275831134147806265011931	0,0000046464626697315960645565368	-0,00000130824201134008298432210964	-0,00000595467701287085042400435948	0,0000925436881911220550884343054
7	-0,0000000000008328994729371027319	-0,00000311002415090597298558305948	-0,00001285218169108821878702080131	0,0000064262300331257510293856852	-0,00000155494479318083414973443065	-0,00000798112353763544664430845119	0,000127991219652173832930466417

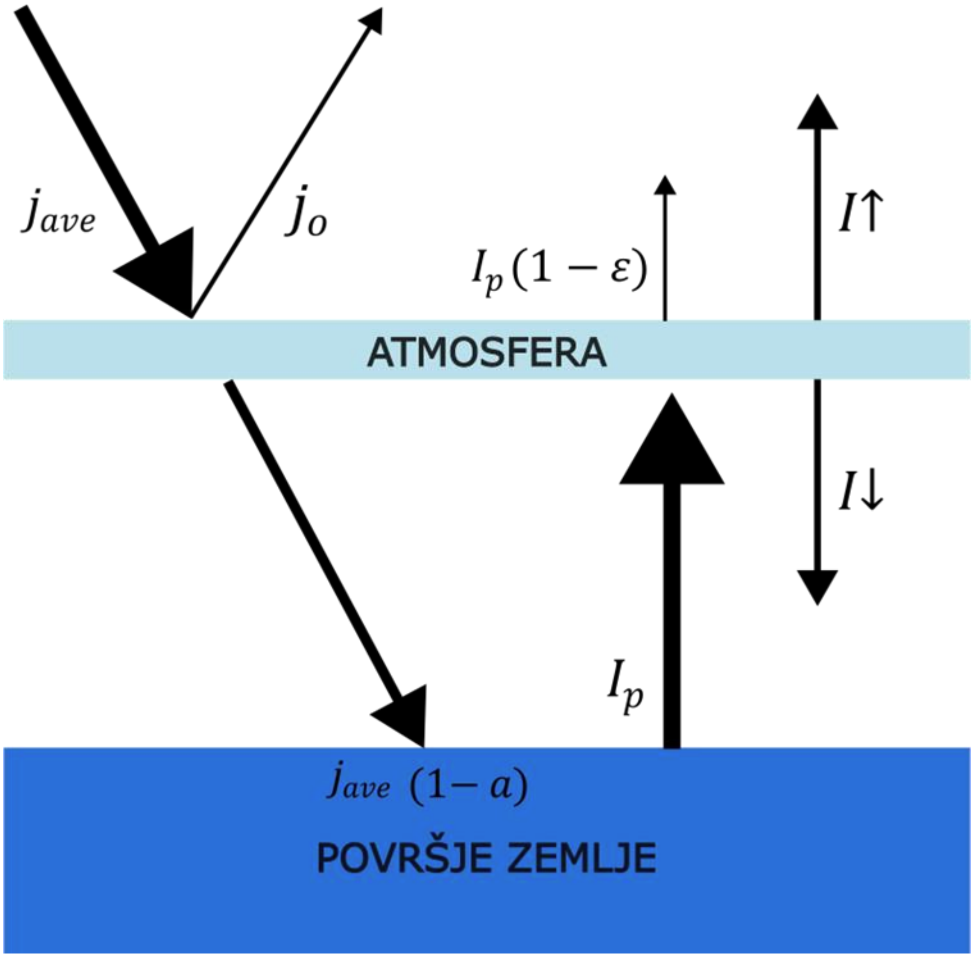
Slike:



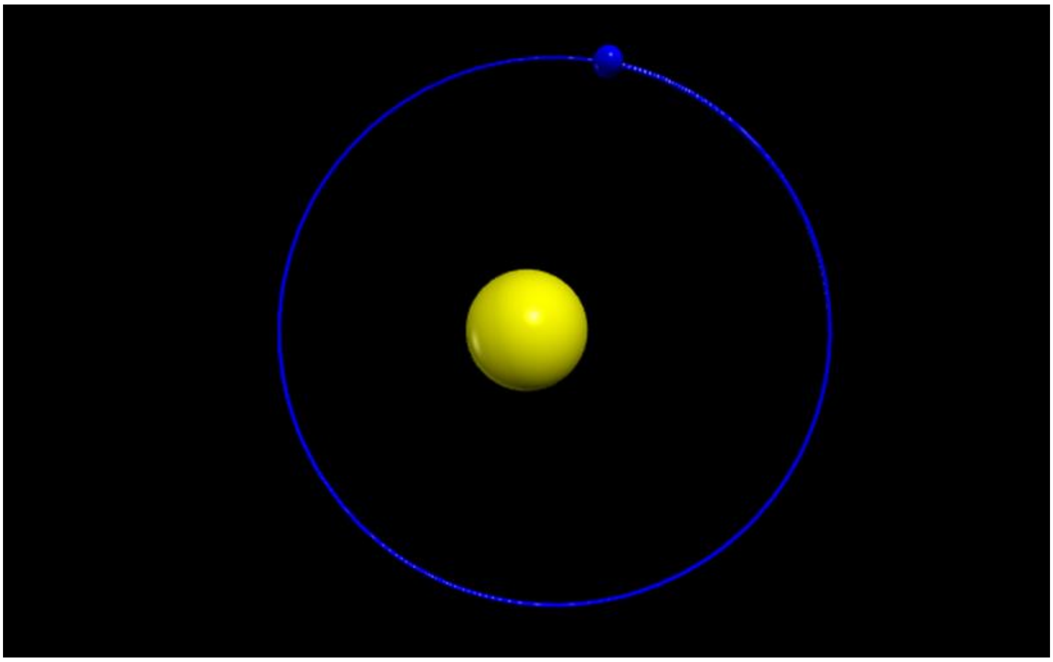
Slika 5: Energijska shema na planetu brez atmosfere



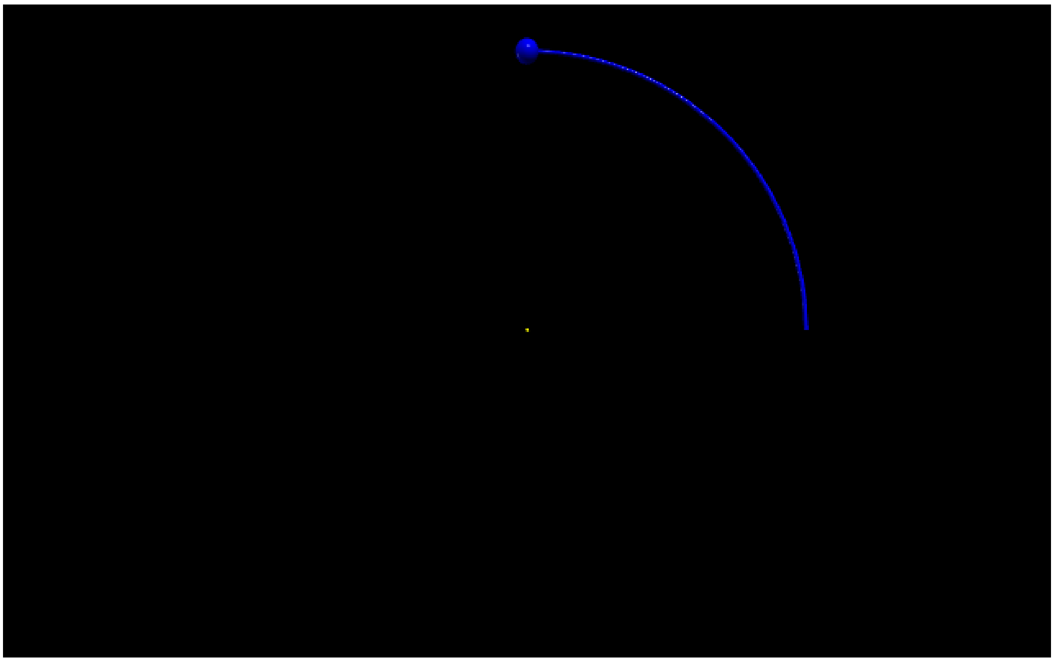
Slika 6: Energijska shema na planetu z atmosfero, ki sprejme vso infrardeče sevanje



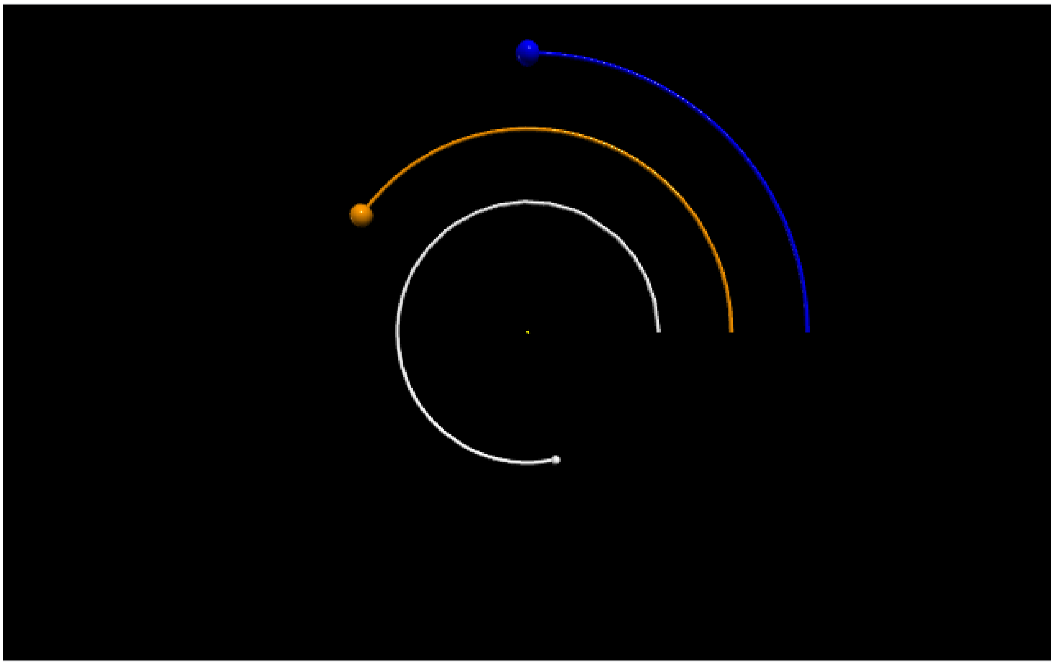
Slika 7: Zemljina energijska shema



Slika 8: Simulacija poljubnega planeta in poljubne zvezde

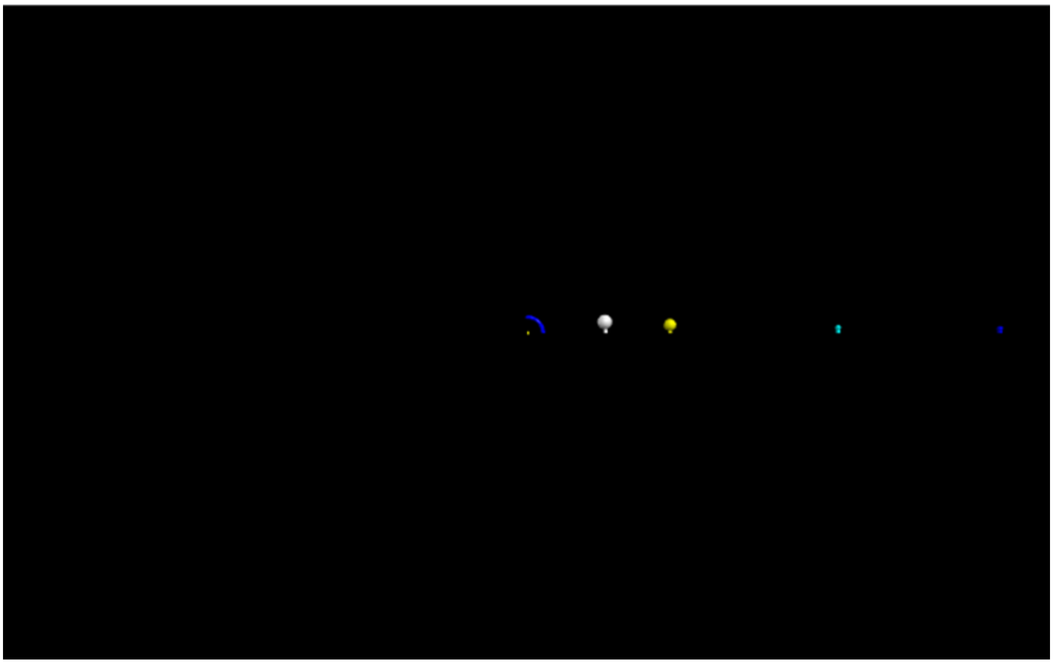


Slika 9: Simulacija Osončja z Zemljo in s Soncem

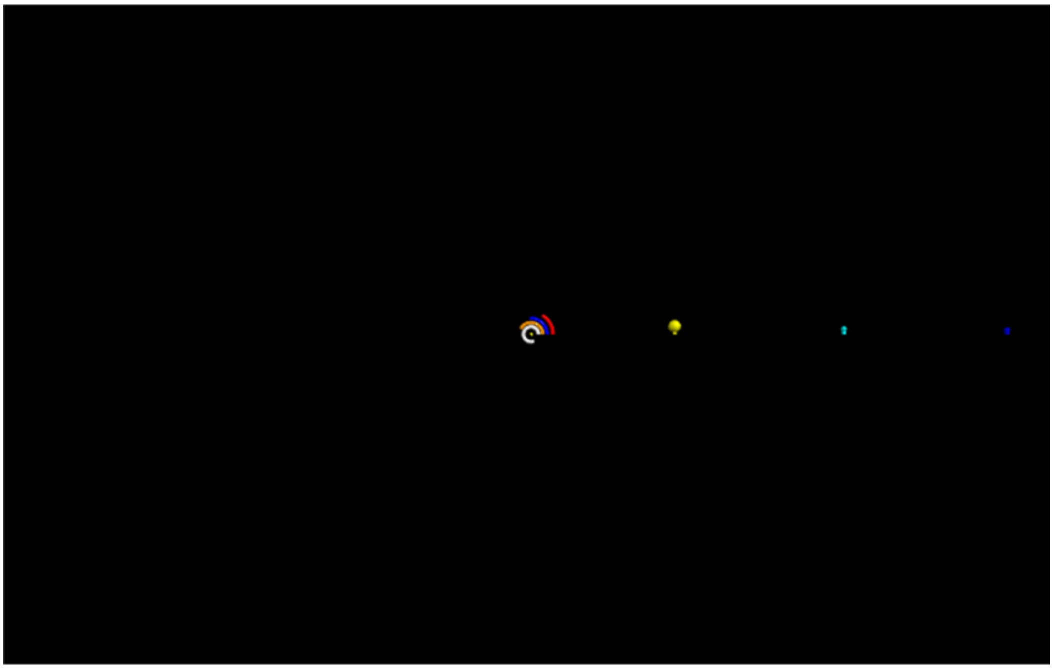


Slika 10: Simulacija Osončja s spodnjima planetoma in z Zemljo

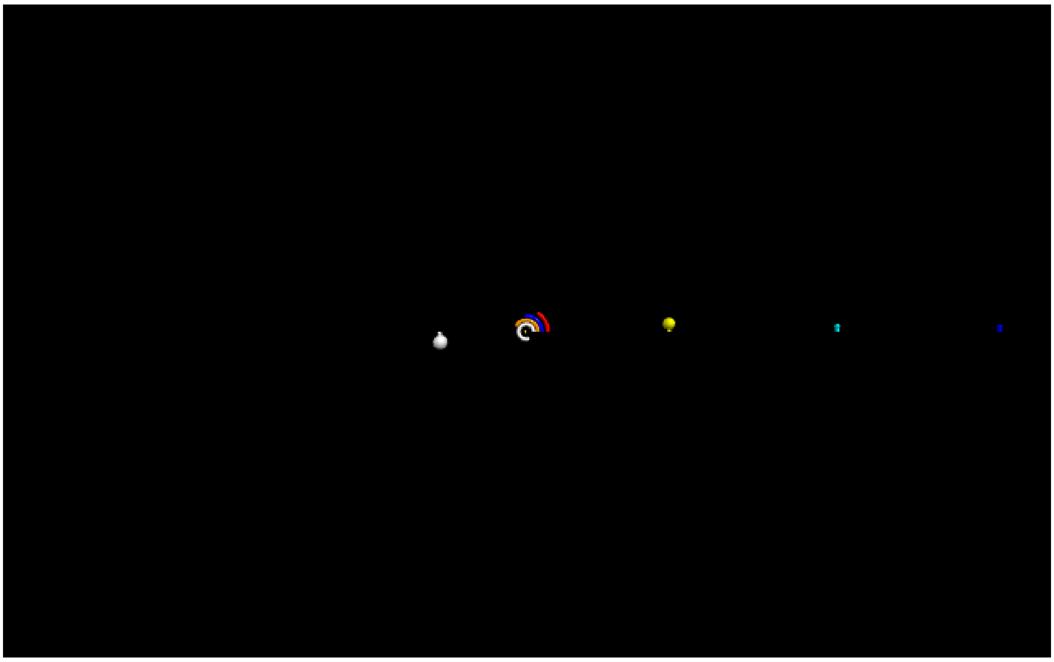
Slika 11: Simulacija Osončja z vsemi planeti



Slika 12: Simulacija Osončja z zgornjimi planeti in Zemljo



Slika 13: Simulacija Osončja brez Jupitra



Slika 14: Simulacija Osončja z Jupitrom na drugi strani

Program simulacije poljubnega planeta in poljubne zvezde

```
import vpython
from vpython import *

e_graph = gcurve(color=color.blue)

def gforce(p1, p2):
    G = 1
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
```

```
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat

    return force_vec


star = sphere(pos=vector(0, 0, 0), radius=0.2, color=color.yellow,
              mass=1000, momentum=vector(0, 0, 0), make_trail=True)

planet = sphere(pos=vector(1, 0, 0), radius=0.05, color=color.blue,
                mass=1, momentum=vector(0, 30, 0), make_trail=True)

t = 0
dt = 0.0001
while (True):
    rate(500)

    star.force = gforce(star, planet)
    planet.force = gforce(planet, star)

    star.momentum = star.momentum + star.force * dt
    planet.momentum = planet.momentum + planet.force * dt

    star.pos = star.pos + star.momentum / star.mass * dt
    planet.pos = planet.pos + planet.momentum / planet.mass * dt

    t = t + dt
```

Program simulacije Osončja z Zemljo in s Soncem

```
from vpython import *
from vpython import sphere

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430 / 10 ** 11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat

    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885 * 10 ** 30, momentum=vector(0, 0, 0),
make_trail=True)

planet: sphere = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
color=color.blue,
                        mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10
** 21) * (G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)

maxpos_x = 1
```

```
maxpos_y = 1

current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500

while True:
    current_y = planet.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet)
        planet.force = gforce(planet, star)

        star.momentum = star.momentum + star.force * dt
        planet.momentum = planet.momentum + planet.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet.pos = planet.pos + planet.momentum / planet.mass * dt

        t = t + dt

        D = (planet.pos.x ** 2 + planet.pos.y ** 2) ** 0.5
        i = i + 1

        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x-maxpos_y)**2 / (maxpos_x+maxpos_y)**2
        o = 3.14159265359*(maxpos_x+maxpos_y)*(1+(3*h)/(10+(4-3*h)**0.5))

        if maxpos_x < planet.pos.x:
            maxpos_x = planet.pos.x

        if maxpos_y < planet.pos.y:
            maxpos_y = planet.pos.y

        previous_y = current_y

    else:
        break

avg_D = total_D / i

avg_v = total_v / i

t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772
```

```
Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x ,"m")

print("mala polos:", maxpos_y ,"m")

print("Povprečna razdalja:", avg_D ,"m")

print("Povprečna hitrost:", avg_v ,"m/s")

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```

Program simulacije Osončja s spodnjima planetoma in z Zemljo

```
import vpython
from vpython import *

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430e-11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat
    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885*10**30, momentum=vector(0, 0, 0),
make_trail=True)

planet1 = sphere(pos=vector(69816900000, 0, 0), radius=2439400000,
color=color.white,
                 mass=3301 * 10 ** 20, momentum=vector(0, (3301 * 10 **
20)*(G * star.mass / 69816900000) ** 0.5, 0), make_trail=True)

planet2 = sphere(pos=vector(108939000000, 0, 0), radius=6051800000,
color=color.orange,
                 mass=4867 * 10 ** 21, momentum=vector(0, (4867 * 10 **
21)*(G * star.mass / 108939000000) ** 0.5, 0), make_trail=True)

planet3 = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
color=color.blue,
                 mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10 **
21)*(G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)


maxpos_x = 1
maxpos_y = 1
```

```
current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500
while True:
    current_y = planet3.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet1) + gforce(star, planet2) +
gforce(star, planet3)

        planet1.force = gforce(planet1, star) + gforce(planet1, planet2) +
gforce(planet1, planet3)

        planet2.force = gforce(planet2, star) + gforce(planet2, planet1) +
gforce(planet2, planet3)

        planet3.force = gforce(planet3, star) + gforce(planet3, planet1) +
gforce(planet3, planet2)

        star.momentum = star.momentum + star.force * dt
        planet1.momentum = planet1.momentum + planet1.force * dt
        planet2.momentum = planet2.momentum + planet2.force * dt
        planet3.momentum = planet3.momentum + planet3.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet1.pos = planet1.pos + planet1.momentum / planet1.mass * dt
        planet2.pos = planet2.pos + planet2.momentum / planet2.mass * dt
        planet3.pos = planet3.pos + planet3.momentum / planet3.mass * dt

        t = t + dt

        D = (planet3.pos.x ** 2 + planet3.pos.y ** 2) ** 0.5
        i = i + 1

        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x - maxpos_y) ** 2 / (maxpos_x + maxpos_y) ** 2
        o = 3.14159265359 * (maxpos_x + maxpos_y) * (1 + (3 * h) / (10 + (4
- 3 * h) ** 0.5))

        if maxpos_x < planet3.pos.x:
            maxpos_x = planet3.pos.x

        if maxpos_y < planet3.pos.y:
            maxpos_y = planet3.pos.y

        previous_y = current_y
```

```
    else:
        break

avg_D = total_D / i

avg_v = total_v / i

t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772

Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x)

print("mala polos:", maxpos_y)

print("Povprečna razdalja:", avg_D)

print("Povprečna hitrost:", avg_v)

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```

Program simulacije Osončja z vsemi planeti

```
import vpython
from vpython import *

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430e-11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat
    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885*10**30, momentum=vector(0, 0, 0),
make_trail=True)

planet1 = sphere(pos=vector(69816900000, 0, 0), radius=2439400000,
color=color.white,
                 mass=3301 * 10 ** 20, momentum=vector(0, (3301 * 10 **
20)*(G * star.mass / 69816900000) ** 0.5, 0), make_trail=True)

planet2 = sphere(pos=vector(108939000000, 0, 0), radius=6051800000,
```

```
color=color.orange,
                 mass=4867 * 10 ** 21, momentum=vector(0, 4867 * 10 **
21*(G * star.mass / 108939000000) ** 0.5, 0), make_trail=True)

planet3 = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
color=color.blue,
                 mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10 **
21)*(G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)

planet4 = sphere(pos=vector(206650000000, 0, 0), radius=3389000000,
color=color.red,
                 mass=64171 * 10 ** 19, momentum=vector(0, (64171 * 10 **
19)*(G * star.mass / 206650000000) ** 0.5, 0), make_trail=True)

planet5 = sphere(pos=vector(740595000000, 0, 0), radius=69911000000,
color=color.white,
                 mass=1.8982 * 10 ** 27, momentum=vector(0, (1.8982 * 10 **
27)*(G * star.mass / 740595000000) ** 0.5, 0), make_trail=True)

planet6 = sphere(pos=vector(1352.55 * 10 ** 9, 0, 0), radius=58232000000,
color=color.yellow,
                 mass=5.6834 * 10 ** 26, momentum=vector(0, (5.6834 * 10 **
26)*(G * star.mass / (1352.55 * 10 ** 9)) ** 0.5, 0), make_trail=True)

planet7 = sphere(pos=vector(2.9415 * 10 ** 12, 0, 0), radius=25362000000,
color=color.cyan,
                 mass=8.6810 * 10 ** 25, momentum=vector(0, (8.6810 * 10 **
25)*(G * star.mass / (2.9415 * 10 ** 12)) ** 0.5, 0), make_trail=True)

planet8 = sphere(pos=vector(4.4736 * 10 ** 12, 0, 0), radius=24622000000,
color=color.blue,
                 mass=1.02413 * 10 ** 26, momentum=vector(0, (1.02413 * 10
** 26)*(G * star.mass / (4.4736 * 10 ** 12)) ** 0.5, 0), make_trail=True)

maxpos_x = 1
maxpos_y = 1

current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500
while True:
    current_y = planet3.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet1) + gforce(star, planet2) +
gforce(star, planet3) + gforce(star, planet4) + gforce(star, planet5) +
gforce(star, planet6) + gforce(star, planet7) + gforce(star, planet8)

        planet1.force = gforce(planet1, star) + gforce(planet1, planet2) +
gforce(planet1, planet3) + gforce(planet1, planet4) + gforce(planet1,
planet5) + gforce(planet1, planet6) + gforce(planet1, planet7) +
gforce(planet1, planet8)
```

```
        planet2.force = gforce(planet2, star) + gforce(planet2, planet1) +
gforce(planet2, planet3) + gforce(planet2, planet4) + gforce(planet2,
planet5) + gforce(planet2, planet6) + gforce(planet2, planet7) +
gforce(planet2, planet8)

        planet3.force = gforce(planet3, star) + gforce(planet3, planet1) +
gforce(planet3, planet2) + gforce(planet3, planet4) + gforce(planet3,
planet5) + gforce(planet3, planet6) + gforce(planet3, planet7) +
gforce(planet3, planet8)

        planet4.force = gforce(planet4, star) + gforce(planet4, planet1) +
gforce(planet4, planet2) + gforce(planet4, planet3) + gforce(planet4,
planet5) + gforce(planet4, planet6) + gforce(planet4, planet7) +
gforce(planet4, planet8)

        planet5.force = gforce(planet5, star) + gforce(planet5, planet1) +
gforce(planet5, planet2) + gforce(planet5, planet3) + gforce(planet5,
planet4) + gforce(planet5, planet6) + gforce(planet5, planet7) +
gforce(planet5, planet8)

        planet6.force = gforce(planet6, star) + gforce(planet6, planet1) +
gforce(planet6, planet2) + gforce(planet6, planet3) + gforce(planet6,
planet4) + gforce(planet6, planet5) + gforce(planet6, planet7) +
gforce(planet6, planet8)

        planet7.force = gforce(planet7, star) + gforce(planet7, planet1) +
gforce(planet7, planet2) + gforce(planet7, planet3) + gforce(planet7,
planet4) + gforce(planet7, planet5) + gforce(planet7, planet6) +
gforce(planet7, planet8)

        planet8.force = gforce(planet8, star) + gforce(planet8, planet1) +
gforce(planet8, planet2) + gforce(planet8, planet3) + gforce(planet8,
planet4) + gforce(planet8, planet5) + gforce(planet8, planet6) +
gforce(planet8, planet7)

        star.momentum = star.momentum + star.force * dt
        planet1.momentum = planet1.momentum + planet1.force * dt
        planet2.momentum = planet2.momentum + planet2.force * dt
        planet3.momentum = planet3.momentum + planet3.force * dt
        planet4.momentum = planet4.momentum + planet4.force * dt
        planet5.momentum = planet5.momentum + planet5.force * dt
        planet6.momentum = planet6.momentum + planet6.force * dt
        planet7.momentum = planet7.momentum + planet7.force * dt
        planet8.momentum = planet8.momentum + planet8.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet1.pos = planet1.pos + planet1.momentum / planet1.mass * dt
        planet2.pos = planet2.pos + planet2.momentum / planet2.mass * dt
        planet3.pos = planet3.pos + planet3.momentum / planet3.mass * dt
        planet4.pos = planet4.pos + planet4.momentum / planet4.mass * dt
        planet5.pos = planet5.pos + planet5.momentum / planet5.mass * dt
        planet6.pos = planet6.pos + planet6.momentum / planet6.mass * dt
        planet7.pos = planet7.pos + planet7.momentum / planet7.mass * dt
        planet8.pos = planet8.pos + planet8.momentum / planet8.mass * dt

        t = t + dt

        D = (planet3.pos.x ** 2 + planet3.pos.y ** 2) ** 0.5
        i = i + 1
```

```
        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x - maxpos_y) ** 2 / (maxpos_x + maxpos_y) ** 2
        o = 3.14159265359 * (maxpos_x + maxpos_y) * (1 + (3 * h) / (10 + (4
- 3 * h) ** 0.5))

        if maxpos_x < planet3.pos.x:
            maxpos_x = planet3.pos.x

        if maxpos_y < planet3.pos.y:
            maxpos_y = planet3.pos.y

        previous_y = current_y

    else:
        break

avg_D = total_D / i

avg_v = total_v / i

t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772

Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x)

print("mala polos:", maxpos_y)

print("Povprečna razdalja:", avg_D)

print("Povprečna hitrost:", avg_v)

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```

Program simulacije Osončja z zgornjimi planeti in Zemljo

```
import vpython
from vpython import *

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430e-11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
```

```
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat
    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885*10**30, momentum=vector(0, 0, 0),
make_trail=True)

planet3 = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
color=color.blue,
                 mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10 **
21)*(G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)

planet5 = sphere(pos=vector(740595000000, 0, 0), radius=69911000000,
color=color.white,
                 mass=1.8982 * 10 ** 27, momentum=vector(0, (1.8982 * 10 **
27)*(G * star.mass / 740595000000) ** 0.5, 0), make_trail=True)

planet6 = sphere(pos=vector(1352.55 * 10 ** 9, 0, 0), radius=58232000000,
color=color.yellow,
                 mass=5.6834 * 10 ** 26, momentum=vector(0, (5.6834 * 10 **
26)*(G * star.mass / (1352.55 * 10 ** 9)) ** 0.5, 0), make_trail=True)

planet7 = sphere(pos=vector(2.9415 * 10 ** 12, 0, 0), radius=25362000000,
color=color.cyan,
                 mass=8.6810 * 10 ** 25, momentum=vector(0, (8.6810 * 10 **
25)*(G * star.mass / (2.9415 * 10 ** 12)) ** 0.5, 0), make_trail=True)

planet8 = sphere(pos=vector(4.4736 * 10 ** 12, 0, 0), radius=24622000000,
color=color.blue,
                 mass=1.02413 * 10 ** 26, momentum=vector(0, (1.02413 * 10
** 26)*(G * star.mass / (4.4736 * 10 ** 12)) ** 0.5, 0), make_trail=True)

maxpos_x = 1
maxpos_y = 1

current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500
while True:
    current_y = planet3.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet3) + gforce(star, planet5) +
gforce(star, planet6) + gforce(star, planet7) + gforce(star, planet8)
        planet3.force = gforce(planet3, star) + gforce(planet3, planet5) +
gforce(planet3, planet6) + gforce(planet3, planet7) + gforce(planet3,
planet8)
```

```
        planet5.force = gforce(planet5, star) + gforce(planet5, planet3) + gforce(planet5, planet6) + gforce(planet5, planet7) + gforce(planet5, planet8)

        planet6.force = gforce(planet6, star) + gforce(planet6, planet3) + gforce(planet6, planet5) + gforce(planet6, planet7) + gforce(planet6, planet8)

        planet7.force = gforce(planet7, star) + gforce(planet7, planet3) + gforce(planet7, planet5) + gforce(planet7, planet6) + gforce(planet7, planet8)

        planet8.force = gforce(planet8, star) + gforce(planet8, planet3) + gforce(planet8, planet5) + gforce(planet8, planet6) + gforce(planet8, planet7)

        star.momentum = star.momentum + star.force * dt
        planet3.momentum = planet3.momentum + planet3.force * dt
        planet5.momentum = planet5.momentum + planet5.force * dt
        planet6.momentum = planet6.momentum + planet6.force * dt
        planet7.momentum = planet7.momentum + planet7.force * dt
        planet8.momentum = planet8.momentum + planet8.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet3.pos = planet3.pos + planet3.momentum / planet3.mass * dt
        planet5.pos = planet5.pos + planet5.momentum / planet5.mass * dt
        planet6.pos = planet6.pos + planet6.momentum / planet6.mass * dt
        planet7.pos = planet7.pos + planet7.momentum / planet7.mass * dt
        planet8.pos = planet8.pos + planet8.momentum / planet8.mass * dt

        t = t + dt

        D = (planet3.pos.x ** 2 + planet3.pos.y ** 2) ** 0.5
        i = i + 1

        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x - maxpos_y) ** 2 / (maxpos_x + maxpos_y) ** 2
        o = 3.14159265359 * (maxpos_x + maxpos_y) * (1 + (3 * h) / (10 + (4 - 3 * h) ** 0.5))

        if maxpos_x < planet3.pos.x:
            maxpos_x = planet3.pos.x

        if maxpos_y < planet3.pos.y:
            maxpos_y = planet3.pos.y

        previous_y = current_y

    else:
        break

avg_D = total_D / i

avg_v = total_v / i
```

```
t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772

Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x)

print("mala polos:", maxpos_y)

print("Povprečna razdalja:", avg_D)

print("Povprečna hitrost:", avg_v)

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```

Program simulacije Osončja brez planeta, ki najbolj vpliva na Zemljino orbito

```
#GlowScript 2.7 VPython

import vpython
from vpython import *

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430e-11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat
    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885*10**30, momentum=vector(0, 0, 0),
make_trail=True)

planet1 = sphere(pos=vector(69816900000, 0, 0), radius=2439400000,
color=color.white,
                 mass=3301 * 10 ** 20, momentum=vector(0, (3301 * 10 **
20)*(G * star.mass / 69816900000) ** 0.5, 0), make_trail=True)

planet2 = sphere(pos=vector(108939000000, 0, 0), radius=6051800000,
color=color.orange,
                 mass=4867 * 10 ** 21, momentum=vector(0, 4867 * 10 **
21*(G * star.mass / 108939000000) ** 0.5, 0), make_trail=True)

planet3 = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
```

```
color=color.blue,
                 mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10 **
21)*(G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)

planet4 = sphere(pos=vector(206650000000, 0, 0), radius=3389000000,
color=color.red,
                 mass=64171 * 10 ** 19, momentum=vector(0, (64171 * 10 **
19)*(G * star.mass / 206650000000) ** 0.5, 0), make_trail=True)

planet6 = sphere(pos=vector(1352.55 * 10 ** 9, 0, 0), radius=58232000000,
color=color.yellow,
                 mass=5.6834 * 10 ** 26, momentum=vector(0, (5.6834 * 10 **
26)*(G * star.mass / (1352.55 * 10 ** 9)) ** 0.5, 0), make_trail=True)

planet7 = sphere(pos=vector(2.9415 * 10 ** 12, 0, 0), radius=25362000000,
color=color.cyan,
                 mass=8.6810 * 10 ** 25, momentum=vector(0, (8.6810 * 10 **
25)*(G * star.mass / (2.9415 * 10 ** 12)) ** 0.5, 0), make_trail=True)

planet8 = sphere(pos=vector(4.4736 * 10 ** 12, 0, 0), radius=24622000000,
color=color.blue,
                 mass=1.02413 * 10 ** 26, momentum=vector(0, (1.02413 * 10
** 26)*(G * star.mass / (4.4736 * 10 ** 12)) ** 0.5, 0), make_trail=True)

maxpos_x = 1
maxpos_y = 1

current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500
while True:
    current_y = planet3.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet1) + gforce(star, planet2) +
gforce(star, planet3) + gforce(star, planet4) + gforce(star, planet6) +
gforce(star, planet7) + gforce(star, planet8)

        planet1.force = gforce(planet1, star) + gforce(planet1, planet2) +
gforce(planet1, planet3) + gforce(planet1, planet4) + gforce(planet1,
planet6) + gforce(planet1, planet7) + gforce(planet1, planet8)

        planet2.force = gforce(planet2, star) + gforce(planet2, planet1) +
gforce(planet2, planet3) + gforce(planet2, planet4) + gforce(planet2,
planet6) + gforce(planet2, planet7) + gforce(planet2, planet8)

        planet3.force = gforce(planet3, star) + gforce(planet3, planet1) +
gforce(planet3, planet2) + gforce(planet3, planet4) + gforce(planet3,
planet6) + gforce(planet3, planet7) + gforce(planet3, planet8)

        planet4.force = gforce(planet4, star) + gforce(planet4, planet1) +
gforce(planet4, planet2) + gforce(planet4, planet3) + gforce(planet4,
planet6) + gforce(planet4, planet7) + gforce(planet4, planet8)
```

```
        planet6.force = gforce(planet6, star) + gforce(planet6, planet1) +
gforce(planet6, planet2) + gforce(planet6, planet3) + gforce(planet6,
planet4) + gforce(planet6, planet7) + gforce(planet6, planet8)

        planet7.force = gforce(planet7, star) + gforce(planet7, planet1) +
gforce(planet7, planet2) + gforce(planet7, planet3) + gforce(planet7,
planet4) + gforce(planet7, planet6) + gforce(planet7, planet8)

        planet8.force = gforce(planet8, star) + gforce(planet8, planet1) +
gforce(planet8, planet2) + gforce(planet8, planet3) + gforce(planet8,
planet4) + gforce(planet8, planet6) + gforce(planet8, planet7)

        star.momentum = star.momentum + star.force * dt
        planet1.momentum = planet1.momentum + planet1.force * dt
        planet2.momentum = planet2.momentum + planet2.force * dt
        planet3.momentum = planet3.momentum + planet3.force * dt
        planet4.momentum = planet4.momentum + planet4.force * dt
        planet6.momentum = planet6.momentum + planet6.force * dt
        planet7.momentum = planet7.momentum + planet7.force * dt
        planet8.momentum = planet8.momentum + planet8.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet1.pos = planet1.pos + planet1.momentum / planet1.mass * dt
        planet2.pos = planet2.pos + planet2.momentum / planet2.mass * dt
        planet3.pos = planet3.pos + planet3.momentum / planet3.mass * dt
        planet4.pos = planet4.pos + planet4.momentum / planet4.mass * dt
        planet6.pos = planet6.pos + planet6.momentum / planet6.mass * dt
        planet7.pos = planet7.pos + planet7.momentum / planet7.mass * dt
        planet8.pos = planet8.pos + planet8.momentum / planet8.mass * dt

        t = t + dt

        D = (planet3.pos.x ** 2 + planet3.pos.y ** 2) ** 0.5
        i = i + 1

        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x - maxpos_y) ** 2 / (maxpos_x + maxpos_y) ** 2
        o = 3.14159265359 * (maxpos_x + maxpos_y) * (1 + (3 * h) / (10 + (4
- 3 * h) ** 0.5))

        if maxpos_x < planet3.pos.x:
            maxpos_x = planet3.pos.x

        if maxpos_y < planet3.pos.y:
            maxpos_y = planet3.pos.y

        previous_y = current_y

    else:
        break

avg_D = total_D / i

avg_v = total_v / i
```

```
t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772

Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x)

print("mala polos:", maxpos_y)

print("Povprečna razdalja:", avg_D)

print("Povprečna hitrost:", avg_v)

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```

Program simulacije Osončja z Jupitrom na drugi strani

```
import vpython
from vpython import *

e_graph = gcurve(color=color.blue)


def gforce(p1, p2):
    G = 6.67430e-11
    r_vec = p1.pos - p2.pos
    r_mag = mag(r_vec)
    r_hat = r_vec / r_mag
    force_mag = G * p1.mass * p2.mass / r_mag ** 2
    force_vec = -force_mag * r_hat
    return force_vec


G = 6.67430 / 10 ** 11

star = sphere(pos=vector(0, 0, 0), radius=696340000, color=color.yellow,
              mass=1.9885*10**30, momentum=vector(0, 0, 0),
make_trail=True)

planet1 = sphere(pos=vector(69816900000, 0, 0), radius=2439400000,
color=color.white,
                 mass=3301 * 10 ** 20, momentum=vector(0, (3301 * 10 **
20)*(G * star.mass / 69816900000) ** 0.5, 0), make_trail=True)

planet2 = sphere(pos=vector(108939000000, 0, 0), radius=6051800000,
color=color.orange,
                 mass=4867 * 10 ** 21, momentum=vector(0, 4867 * 10 **
21*(G * star.mass / 108939000000) ** 0.5, 0), make_trail=True)

planet3 = sphere(pos=vector(149597887000, 0, 0), radius=6371000000,
color=color.blue,
                 mass=5972 * 10 ** 21, momentum=vector(0, (5972 * 10 **
```

```
21)*(G * star.mass / 149597887000) ** 0.5, 0), make_trail=True)

planet4 = sphere(pos=vector(206650000000, 0, 0), radius=3389000000,
color=color.red,
                 mass=64171 * 10 ** 19, momentum=vector(0, (64171 * 10 **
19)*(G * star.mass / 206650000000) ** 0.5, 0), make_trail=True)

planet5 = sphere(pos=vector(-816363000000, 0, 0), radius=69911000000,
color=color.white,
                 mass=1.8982 * 10 ** 27, momentum=vector(0, -(1.8982 * 10
** 27)*(G * star.mass / 816363000000) ** 0.5, 0), make_trail=True)

planet6 = sphere(pos=vector(1352.55 * 10 ** 9, 0, 0), radius=58232000000,
color=color.yellow,
                 mass=5.6834 * 10 ** 26, momentum=vector(0, (5.6834 * 10 **
26)*(G * star.mass / (1352.55 * 10 ** 9)) ** 0.5, 0), make_trail=True)

planet7 = sphere(pos=vector(2.9415 * 10 ** 12, 0, 0), radius=25362000000,
color=color.cyan,
                 mass=8.6810 * 10 ** 25, momentum=vector(0, (8.6810 * 10 **
25)*(G * star.mass / (2.9415 * 10 ** 12)) ** 0.5, 0), make_trail=True)

planet8 = sphere(pos=vector(4.4736 * 10 ** 12, 0, 0), radius=24622000000,
color=color.blue,
                 mass=1.02413 * 10 ** 26, momentum=vector(0, (1.02413 * 10
** 26)*(G * star.mass / (4.4736 * 10 ** 12)) ** 0.5, 0), make_trail=True)

maxpos_x = 1
maxpos_y = 1

current_y = 1
previous_y = -1

i = 0
total_D = 1
total_v = 1

t = 0
dt = 500
while True:
    current_y = planet3.pos.y
    if current_y > previous_y:

        rate(10000)

        star.force = gforce(star, planet1) + gforce(star, planet2) +
gforce(star, planet3) + gforce(star, planet4) + gforce(star, planet5) +
gforce(star, planet6) + gforce(star, planet7) + gforce(star, planet8)

        planet1.force = gforce(planet1, star) + gforce(planet1, planet2) +
gforce(planet1, planet3) + gforce(planet1, planet4) + gforce(planet1,
planet5) + gforce(planet1, planet6) + gforce(planet1, planet7) +
gforce(planet1, planet8)

        planet2.force = gforce(planet2, star) + gforce(planet2, planet1) +
gforce(planet2, planet3) + gforce(planet2, planet4) + gforce(planet2,
planet5) + gforce(planet2, planet6) + gforce(planet2, planet7) +
gforce(planet2, planet8)

        planet3.force = gforce(planet3, star) + gforce(planet3, planet1) +
gforce(planet3, planet2) + gforce(planet3, planet4) + gforce(planet3,
```

```
planet5) + gforce(planet3, planet6) + gforce(planet3, planet7) +
gforce(planet3, planet8)

        planet4.force = gforce(planet4, star) + gforce(planet4, planet1) +
gforce(planet4, planet2) + gforce(planet4, planet3) + gforce(planet4,
planet5) + gforce(planet4, planet6) + gforce(planet4, planet7) +
gforce(planet4, planet8)

        planet5.force = gforce(planet5, star) + gforce(planet5, planet1) +
gforce(planet5, planet2) + gforce(planet5, planet3) + gforce(planet5,
planet4) + gforce(planet5, planet6) + gforce(planet5, planet7) +
gforce(planet5, planet8)

        planet6.force = gforce(planet6, star) + gforce(planet6, planet1) +
gforce(planet6, planet2) + gforce(planet6, planet3) + gforce(planet6,
planet4) + gforce(planet6, planet5) + gforce(planet6, planet7) +
gforce(planet6, planet8)

        planet7.force = gforce(planet7, star) + gforce(planet7, planet1) +
gforce(planet7, planet2) + gforce(planet7, planet3) + gforce(planet7,
planet4) + gforce(planet7, planet5) + gforce(planet7, planet6) +
gforce(planet7, planet8)

        planet8.force = gforce(planet8, star) + gforce(planet8, planet1) +
gforce(planet8, planet2) + gforce(planet8, planet3) + gforce(planet8,
planet4) + gforce(planet8, planet5) + gforce(planet8, planet6) +
gforce(planet8, planet7)

        star.momentum = star.momentum + star.force * dt
        planet1.momentum = planet1.momentum + planet1.force * dt
        planet2.momentum = planet2.momentum + planet2.force * dt
        planet3.momentum = planet3.momentum + planet3.force * dt
        planet4.momentum = planet4.momentum + planet4.force * dt
        planet5.momentum = planet5.momentum + planet5.force * dt
        planet6.momentum = planet6.momentum + planet6.force * dt
        planet7.momentum = planet7.momentum + planet7.force * dt
        planet8.momentum = planet8.momentum + planet8.force * dt

        star.pos = star.pos + star.momentum / star.mass * dt
        planet1.pos = planet1.pos + planet1.momentum / planet1.mass * dt
        planet2.pos = planet2.pos + planet2.momentum / planet2.mass * dt
        planet3.pos = planet3.pos + planet3.momentum / planet3.mass * dt
        planet4.pos = planet4.pos + planet4.momentum / planet4.mass * dt
        planet5.pos = planet5.pos + planet5.momentum / planet5.mass * dt
        planet6.pos = planet6.pos + planet6.momentum / planet6.mass * dt
        planet7.pos = planet7.pos + planet7.momentum / planet7.mass * dt
        planet8.pos = planet8.pos + planet8.momentum / planet8.mass * dt

        t = t + dt

        D = (planet3.pos.x ** 2 + planet3.pos.y ** 2) ** 0.5
        i = i + 1

        total_D = total_D + D

        v = (G * star.mass / D) ** 0.5

        total_v = total_v + v

        h = (maxpos_x - maxpos_y) ** 2 / (maxpos_x + maxpos_y) ** 2
        o = 3.14159265359 * (maxpos_x + maxpos_y) * (1 + (3 * h) / (10 + (4
```

```
- 3 * h) ** 0.5))

        if maxpos_x < planet3.pos.x:
            maxpos_x = planet3.pos.x

        if maxpos_y < planet3.pos.y:
            maxpos_y = planet3.pos.y

        previous_y = current_y

    else:
        break

avg_D = total_D / i

avg_v = total_v / i

t = o/(avg_v * 60*60*24)

a = 0.306
ε = 0.78
Ts = 5772

Te = ((Ts**4 * star.radius**2 * (1-a)) / (4 * avg_D**2 * (1-(ε/2))))**0.25
- 273.15

print("velika polos:", maxpos_x)

print("mala polos:", maxpos_y)

print("Povprečna razdalja:", avg_D)

print("Povprečna hitrost:", avg_v)

print("Obseg orbite:", o ,"m")

print("Dolžina leta:", t ,"dni")

print("Efektivna temperatura:", Te ,"°C")
```