POLITECHNIKA KRAKOWSKA im. T. Kościuszki
Wydział Fizyki, Matematyki i Informatyki
Instytut Fizyki

Kierunek studiów: Nanotechnologia i Nanomateriały
Specjalność : Inżynier nanostruktur

STUDIA STACJONARNE

PRACA INŻYNIERSKA

Karol Marcinkowski

Orbita transferowa Hohmanna.

The Hohmann transfer orbit.

Promotor:
Dr. Radosław Kycia

Uzgodniona ocena :…….

……
podpis promotora i recenzenta

Kraków 2017

Spis treści:

1. Wprowadzenie

Chęć poznania przestrzeni kosmicznej nurtowała człowieka od zalania dziejów. Dopiero w XX wieku, ówczesna technologia, badania oraz obliczenia teoretyczne, dały cień nadziei, że człowiek kiedyś będzie mógł wylecieć poza granice swojej planety. To stymulowało ludzkość do opracowywania metod umożliwiających eksplorację przestrzeni kosmicznej.

1.1. Motywacja:

Jednym z bardziej znaczących odkryć dla dzisiejszej nauki była orbita transferowa opisana przez Waltera Hohmanna i Władimira Pietrowicza Wietczinkina w 1925r., było to 39 lat przed wysłaniem pierwszego człowieka do przestrzeni kosmicznej [1]. Manewr transferowy Hohmanna polega na przejściu z orbity kołowej na eliptyczną o dużo większym promieniu, dwa razy używając silników w statku kosmicznym [Rys.3]. Hobbystycznie interesuję się kosmosem oraz wszystkim co się z nim wiąże, więc temat mojej pracy bardzo mi się spodobał. W 1991 roku Japońska sonda Hiten zastała wystrzelona na orbitę Księżyca [2]. Celem misji było zbadanie możliwości wykonania manewrów w przestrzeni kosmicznej za pomocą asysty grawitacyjnej Księżyca i hamowania w atmosferze Ziemi. Po wykonaniu dziesięciu przejść obok Księżyca i dwóch manewrach hamowania w atmosferze Ziemi główna misja sondy została zakończona. Statek po wejściu na orbitę okołoksiężycową został rozbity o powierzchnię naszego naturalnego satelity gdyż w jego zbiornikach nie zostało wystarczająco dużo paliwa, aby móc go sprowadzić na Ziemie.

1.2. Metodyka:

W niniejszej pracy zawarte zostały tematy związane z wykorzystaniem symulacji komputerowej do badania orbit transferowych. Tego typu obliczenia bardzo często są podstawą dla naukowców do projektowania misji do ciał niebieskich, gdyż pozwalają do nich dotrzeć w bardzo ekonomiczny sposób. Niestety wiąże się to również z bardzo długim czasem oczekiwań na osiągnięcie celu przez statek kosmiczny. Moim celem będzie dokładne opisanie orbity transferowej dookoła Księżyca i zasymulowanie. W mojej pracy za bazowy przykład do opisu tej orbity posłużę się lotem statku Apollo

Część I- teoretyczna

2. Teoria grawitacji Newtona

2.1. Prawo grawitacji Newtona

Na gruncie klasycznej teorii Grawitacji sformułowanej przez Newtona, każde ciało we Wszechświecie przyciąga każde inne ciało siłą wprost proporcjonalną do iloczynu mas obu ciał, M i m, i odwrotnie proporcjonalnie do kwadratu odległości r między środkami obu ciał [3,4]. Siła ta działa wzdłuż prostej łączącej oba środki masy oraz jest ona zawsze przyciągająca.

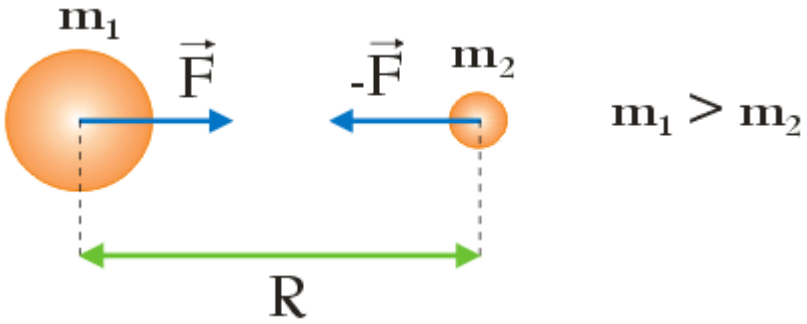
$$F^i = G\frac{Mm}{r^2}e^i,$$

G - stała grawitacji, G= 6.673×10-11 $N \cdot m^2 \cdot kg^{-2}$

x^i – wektor łączący środki mas obu ciał, a *r* jest długością tego wektora

e^i – wersor bazowy wzdłuż osi i

Na wszystkie ciała we wszechświecie oddziałuje siła grawitacji, niezależnie od ich wielkości (Ziemia, Księżyc, człowiek).



Rys. 1 Przyciąganie grawitacyjne dwóch ciał [5]

2.2. Natężenie pola grawitacyjnego

Natężeniem pola grawitacyjnego nazywamy stosunek siły **F** działającej na ciało do masy m tego ciała umieszczonego w polu. Masa ciała próbnego musi być mała, aby nie zaburzyła pola grawitacyjnego, które mierzymy tą masą. Wektor **g** określa wartość i kierunek natężenia pola grawitacyjnego.

$$g = \frac{\boldsymbol{F}}{m} = G\frac{M}{r^2}\hat{r},$$

Kiedy określamy ciężar ciała znajdującego się w pobliżu Ziemi (M będzie masą Ziemi, zaś r odległością środka masy ciała do środka masy Ziemi), to siłę działająca na masę w polu grawitacyjnym nazywamy ciężarem tego ciała **w**:

$$w = mg.$$

2.3. Grawitacyjna energia potencjalna i potencjał grawitacyjny.
Energią potencjalną ciała określa się przez jego położenie obliczając pracę, jaką należy wykonać, aby przenieść ciało z punktu, w którym aktualnie się znajduje do nieskończoności, gdzie siła grawitacji zanika.

Jednym z najważniejszych parametrów charakteryzujących pole grawitacyjne jest **potencjał grawitacyjny** opisujący zdolność pola do nadawania energii potencjalnej ciału, jakie się w nim znajdzie. Wielkość tą oznaczamy przez symbol $\boldsymbol{\gamma}$, definiujemy jako stosunek energii potencjalnej $\mathbf{U_g}$ jaką ma ciało umieszczone w polu grawitacyjnym do wartości tej masy:

$$\gamma = \frac{U_g}{m}.$$

3. Prędkości kosmiczne

3.1. Pierwszą prędkością kosmiczną nazywamy najmniejszą poziomą prędkość jaką należy nadać ciału względem przyciągającego je ciała niebieskiego. Dzięki tym specyficznym warunkom można wnioskować, że dla ciała niebieskiego o kształcie kuli, orbita będzie orbitą kołową o promieniu równym promieniowi planety.

$$\frac{mv^2}{R} = \frac{GMm}{R^2}$$

$$v^2 = \frac{GM}{R}$$

$$v_I = \sqrt{\frac{GM}{R}}$$

G – stała grawitacji;
M – masa ciała niebieskiego;
m – masa rozpędzanego ciała czyli satelity krążącego wokół ciała niebieskiego;
R – promień planety.

3.2. Drugą prędkością kosmiczną nazywamy prędkość jaką trzeba nadać obiektowi na powierzchni tego ciała niebieskiego, aby tor jego ruchu stał się krzywą otwartą. Obliczamy ją przyrównując energię obiektu znajdującego się na powierzchni ciała niebieskiego do energii ciała, które zatrzyma się w nieskończoności. Energia w nieskończoności wynosi 0 (ze względu na brak oddziaływań z polem grawitacyjnym i zerową prędkość), zatem na powierzchni sumaryczna energia też musi się równać 0.

$$E = -G\frac{Mm}{R} + \frac{mv^2}{2}$$

M – masa ciała niebieskiego;
m – masa wystrzeliwanego ciała;
v – prędkość początkowa;
R – promień ciała niebieskiego.

Stąd wynika:

$$v_{II} = \sqrt{\frac{2GM}{R}} = \sqrt{2}v_I$$

4. Projektowanie misji statków kosmicznych

Pierwsze realne koncepcje podróży kosmicznych przypisuje się Konstantinowi Ciołkowskiemu. W 1903 r. opublikował on swoja najsłynniejszą rozprawę pt. *„Eksploracja przestrzeni kosmicznej dzięki urządzeniom odrzutowym”*. Natomiast dopiero po 54 latach od udostępnienia publikacji przez Ciołkowskiego, Związek Radziecki wystrzelił na orbitę okołoziemską pierwszego sztucznego satelitę o nazwie Sputnik 1. Było to pierwszym dużym krokiem do podboju kosmosu co w późniejszych latach doprowadziło do wysłania pierwszego załogowego statku kosmicznego.

Przestrzeń kosmiczna obecnie definiowana jest jako obszar powyżej stu kilometrów nad powierzchnią Ziemi. Jeśli chcemy wystrzelić pocisk rakietowy w kosmos, potrzebujemy nadać mu zmianę prędkości dela-v, która przekłada się na impuls siły potrzebny aby wykonać manewr (np. zmiany trajektorii lotu). Prędkość ta jest o wiele mniejsza niż 2 prędkość kosmiczna.

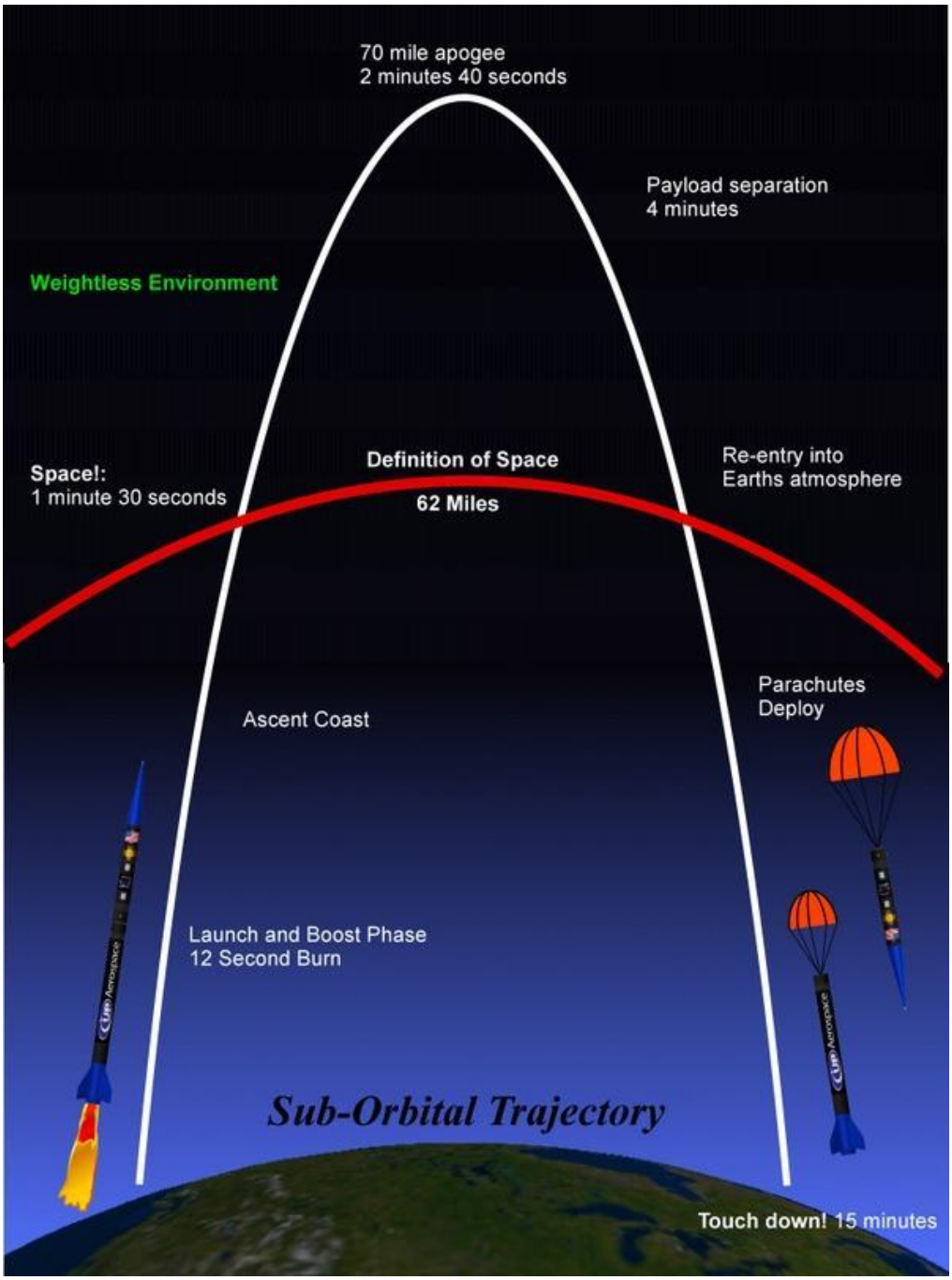
$$\Delta v = \int_t \frac{|\vec{T}|}{m} dt = \int_t |\vec{a}| dt$$

Gdzie:
T – chwilowa siła ciągu silnika
m – chwilowa masa
a – chwilowe przyśpieszenie

Rozpatrując przypadek, w którym siły zewnętrzne nie działają na nasz obiekt oraz jeśli ciąg działa we wskazanym kierunku, można zastosować uproszczenie

$$\Delta v = \int_t |\vec{a}| dt = |v_1 - v_0|$$

Loty w kosmos można podzielić na trzy główne typy: suborbitalny, orbitalny oraz bezpośrednie wyniesienie. O **suborbitalnym locie** kosmicznym mówi się wtedy, gdy statek dociera do przestrzeni kosmicznej ale nie zostaje umieszczony na orbicie. Trajektoria jego prowadzi z powrotem na Ziemię.



Rys. 2 Wizualizacja orbity lotu suborbitalnego [7]

Lot orbitalny występuje wtedy, gdy statek kosmiczny zostanie umieszczony na trajektorii w przestrzeni kosmicznej na której może pozostać. Aby taki lot mógł się odbyć wymagany jest pionowy start rakietowy, dzięki czemu jest w stanie pokonać opór atmosferyczny czy grawitację, aż do uzyskania docelowego pułapu i przyśpieszenia poziomego.

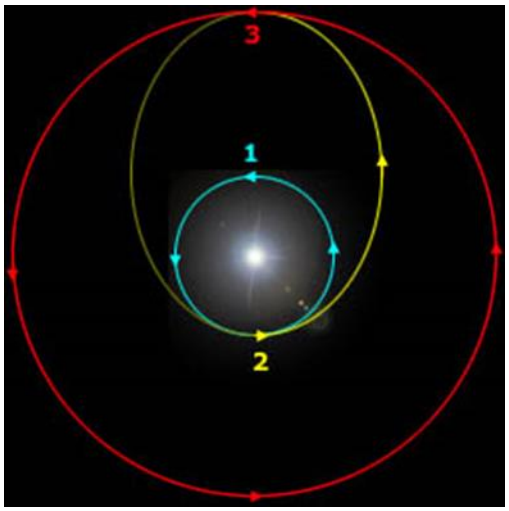
Bezpośrednie wyniesienie jest pomysłem opierającym się na ogromnych rakietach Nova czy Saturn C-8, których moc pozwalałaby wystrzelić statek wprost na księżyc bez uzyskiwania zamkniętej orbity oraz prędkości ucieczki. W czasie wstępnego planowania misji Apollo NASA rozważała zastosowanie bezpośredniego wyniesienia na Księżyc, jednak pomysł ten został porzucony ze względu na masę pojazdu. Ten pomysł nigdy się nie sprawdził, jedynie został przedstawiony jako koncepcja w filmach science-fiction np. *„Kierunek Księżyc”*- Irvinga Pichela.

5. Orbita transferowa Hohmanna

Niemiecki naukowiec Walter Hohmann opublikował w 1925 roku założenia manewru transferowego[*Walter Hohmann (1925). Die Erreichbarkeit der Himmelskörper. Verlag Oldenbourg in München. ISBN 3-486-23106-5*]. Manewr ten polega na ekonomicznej zmianie orbity kołowej statku kosmicznego na wyższą lub niższą, poprzez dwukrotne użycie silników. Jednak główną ideą pomysłu Hohmanna jest możliwość przetransferowania statku z okolicy jednego ciała w pobliże drugiego. Statek ten porusza się tam w nieskończoność, chyba, że odpali silniki, wówczas wyląduje. Jeśli nie odpali bądź nie miał by takiej możliwości (np. w przypadku awarii), to powróci do punktu startu. Stąd wynika bezpieczeństwo tej orbity.[8]

Podwyższenie orbity następuje wtedy, gdy statek przechodzi z orbity kołowej do orbity w kształcie połówki elipsy, która jest styczna zarówno do opuszczanej orbity jak i orbity docelowej [Rys. 3]. Przejście między orbitami inicjowane jest poprzez pierwsze odpalenie silnika, czego skutkiem jest podwyższenie aktualnej orbity.

Gdy statek kosmiczny osiąga docelową wysokość, następuje drugie odpalenie silnika, którego celem jest dostosowanie prędkości do prędkości orbitalnej wymaganej przez orbitę docelową. Istnieje możliwość aby statek przez pewien czas krążył po orbicie transferowej Hohmanna.



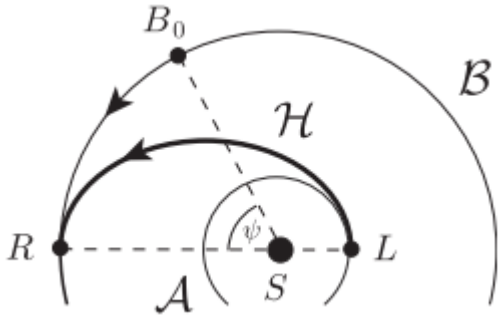
Rys. 3 Wizualizacja manewru transferowego Hohmanna [8]

Wykorzystując manewr transferowy Hohmanna możemy równiej sprowadzić statek kosmiczny z wyżej orbity kołowej na niższą. Jedyną różnicą jest kierunek ciągu impulsów, muszą być one skierowane przeciwnie do kierunku lotu statku. Obniżenie orbity oraz spowolnienie lotu jest skutkiem pierwszego impulsu. Natomiast drugi impuls powoduje dostosowanie prędkości statku do prędkości jaką musi przyjąć nasz obiekt na nowej niższej orbicie.

Niezależnie czy podwyższamy czy obniżamy orbitę w wyżej opisanych manewrach wykorzystuje się silniki o dużym ciągu w celu redukcji zużycia paliwa. Gdybyśmy wykorzystali silniki o małym ciągu nasz manewr mógłby zostać wykonany tylko w przybliżeniu.

Podczas analizy tego manewru należy założyć, że czas działania impulsu jest krótki, w którym silniki rakietowe wypalają paliwo, ponieważ dostarczają one impulsu do pojazdu kosmicznego w wyniku gwałtownej zmiany prędkości.

Dwa ciała niebieskie poruszają się po orbitach kołowych A i B. Statek jest zobowiązany do oddalenia od jednego z ciał i spotkanie z drugim w pewnym punkcie swojej orbity.



Rys. 4 Model orbity transferowej [8]

Na rysunku obok można zaobserwować następujące punkty:
- S – pierwsza planeta,
- L – statek kosmiczny,
- R – druga planeta,
- H – połowa orbity Hohmanna,
- A – orbita wstępna statku,
- B – orbita planety drugiej,
- φ - różnica w kątach obrotu satelity i ciała docelowego

Pierwszy etap manewru to krótkotrwale uruchomienie silników w momencie, gdy pojazd kosmiczny znajduje się w punkcie L wstępnej orbity kołowej o promieniu R_0 (odległość między S a L). Dysza silnika skierowana jest przeciwnie do kierunku lotu i przyrost prędkości ΔV_A skierowany jest zgodnie z wektorem prędkości początkowej V_0 czyli stycznie do okręgu. Również nowa prędkość V_A jest prostopadła do promienia wodzącego pojazdu, gdyż jest sumą jednakowo skierowanych V_0 i ΔV_A.

ΔV może zostać określona przez wielokrotne stosowanie następującego równania, które mówi, że całkowita energia jest sumą energii kinetycznej i energii potencjalnej.

$$\mathcal{E} = \frac{v^2}{2} - \frac{\mu}{r^2}$$

gdzie:
$\mathcal{E}$ - jest całkowitą energią przypadającą na jednostkę masy obiektu lub „określoną energią”
v - to prędkość obiektu w bieżącym położeniu
μ - to GM centralnego ciała tzn. stała grawitacyjna Newtona pomnożona przez masę
r - to bieżąca odległość od centralnego ciała.

Kluczem jest to, że całkowita energia obiektu jest stałą ruchu na orbicie, jeśli silniki nie są używane.
Wykorzystuje również fakt, że orbity są elipsami, a równanie, które określa, stałą ruchu apsyd orbity tzn. promienie najbliższych i najdalszych punktów na orbicie $\boldsymbol{r_1}$ i $\boldsymbol{r_2}$ [7]

$$\mathcal{E} = \frac{-\mu_s}{r_1 + r_2}$$

Dla tego problemu definiujemy:
$\boldsymbol{\mu_S}$ - GM słońca,
$\boldsymbol{\mu_1}$ - GM ciała wylotu,
$\boldsymbol{\mu_2}$ - GM ciała przylotu,
$\boldsymbol{r_1}$ - promień orbity ciała wylotu krążącego wokół słońca, zakłada się, że orbita jest kołowa,
$\boldsymbol{r_2}$- promień orbity ciała przylotu krążącego wokół słońca, zakłada się, że jest kołowa.

Orbita Transferu Hohmana ma absydy $\boldsymbol{r_1}$i $\boldsymbol{r_2}$**.** Więc jego specyficzną energią jest:

$$\mathcal{E}_H = -\frac{\mu_S}{r_1 + r_2},$$

Następnie obliczam prędkość orbity r_1:

$$\mathcal{E}_H = -\frac{\mu_S}{r_1 + r_2} = \frac{v^2{}_{H1}}{2} - \frac{\mu_S}{r_1},$$

Co daje:

$$v_{H1} = \sqrt{\frac{2\mu_S r_2}{r_1(r_1 + r_2)}},$$

Dla kołowej orbity ciała wylotu, krążącego wokół Słońca mamy dla jego prędkości wokół słońca:

$$-\frac{\mu_S}{2r_1} = \frac{v_1^2}{2} - \frac{\mu_S}{r_1},$$

Lub

$$v_1 = \sqrt{\frac{\mu_S}{r_1}}.$$

Prędkość orbity transferu Hohmana w perycentrum jest w tym samym kierunku co prędkość ciała wylatującego i są one w takim samym promieniu od słońca, więc zamiana prędkości od wylotu kołowej orbity do orbity transferu Hohmana jest tylko różnicą (zostanie to później podniesione do potęgi, więc znak przed tą różnicą nie ma znaczenia).

$$v_{T1} = v_{H1} - v_1 = \sqrt{\frac{2\mu_S r_2}{r_1(r_1 + r_2)}} - \sqrt{\frac{\mu_S}{r_1}},$$

Jest to zmiana prędkości z orbity ciała wylotu do orbity transferu Hohmana, ale bez jego obecności. Aby wykonać manewr z kołowej orbity wokół ciała użyjemy łatanego przybliżenia stożkowego [8], ponieważ potrzebujemy prędkości wyjścia z ciała równej do prędkości transferu. Tu płaszczyzna orbity krążącej wokół ciała jest ważna, ponieważ musi się ona ustawić w jednej linii z kierunkiem prędkości transferu do zminimalizowania $\triangle V$. W tym przypadku jest to wspólna płaszczyzna orbity obu ciał.

Określona energia obiektu wyjściowego znajdującego się wystarczająco daleko od ciała to równanie, w którym drugi warunek dąży do zera, gdy odległość zmierza ku nieskończoności.

$$\mathcal{E}_{escape} = \frac{v_\infty^2}{2}$$

Teraz, aby uzyskać zmianę prędkości wyjścia i włączenie do transferu Hohmana bierzemy różnicę między prędkością energii wyjścia w promieniu orbity sondy wokół ciała wyjścia a prędkością orbitalnej sondy:

$$\frac{v_{T1}^2}{2} = \frac{v_{escape}^2}{2} - \frac{\mu_1}{a_1}$$

$$v_{escape} = \sqrt{v_{T1}^2 + \frac{2\mu_1}{a_1}}$$

$$v_{orbit} = \sqrt{\frac{\mu_1}{a_1}}$$

$$v_{inject} = V_{escape} - v_{orbit} = \sqrt{v_{T1}^2 + \frac{2\mu_1}{a_1}} - \sqrt{\frac{\mu_1}{a_1}}$$

$$v_{inject} = \sqrt{\left(\sqrt{\frac{2\mu_2 r_2}{r_1(r_1 + r_2)}} - \sqrt{\frac{\mu_S}{r_1}}\right)^2 + \frac{2\mu_2}{a_2}} - \sqrt{\frac{\mu_1}{a_1}}$$

To jest $\triangle V$ wymagana przez statek by poruszać się bezpośrednio z orbity kołowej wokół ciała wylotu do orbity transferowej Hohmana pojedynczym momentalnym manewrem.
Możemy powtórzyć to wszystko, aby wstawić w orbitę przylotu, by uzyskać:

$$v_{insert} = \sqrt{\left(\sqrt{\frac{2\mu_2 r_2}{r_1(r_1+r_2)}} - \sqrt{\frac{\mu_S}{r_1}}\right)^2 + \frac{2\mu_2}{a_2}} - \sqrt{\frac{\mu_2}{a_2}},$$

Całkowita ΔV jest więc sumą dwóch manewrów:

$$\Delta V = V_{escape} - v_{orbit} = \sqrt{v_{T1}^2 + \frac{2\mu_1}{a_1}} - \sqrt{\frac{\mu_1}{a_1}} + \sqrt{\left(\sqrt{\frac{2\mu_2 r_2}{r_1(r_1+r_2)}} - \sqrt{\frac{\mu_S}{r_1}}\right)^2 + \frac{2\mu_2}{a_2}} - \sqrt{\frac{\mu_2}{a_2}},$$

lub nieco uproszone:

$$\Delta V = \sqrt{\frac{v_s}{r_1}\left(\sqrt{\frac{2r_2}{r_1+r_2}} - 1\right)^2 + \frac{2\mu_1}{a_1}} - \sqrt{\frac{\mu_1}{a_1}} + \sqrt{\frac{\mu_S}{r_2}\left(\sqrt{\frac{2r_1}{r_1+r_2}} - 1\right)^2 + \frac{2\mu_2}{a_2}} - \sqrt{\frac{\mu_2}{a_2}},$$

Okres ruchu po elipsie można otrzymać z trzeciego prawa Keplera, zaś połowa jego daje nam czas transferu Hohmanna.

$$T_t = \frac{(2\pi)}{2}\sqrt{\frac{a_t^3}{\mu}} = \frac{\pi}{2}\sqrt{\frac{(r_1+r_2)^3}{2\mu}}.$$

Kątowe ustawienie α (w radianach) w momencie startu między obiektem źródłowym a docelowym można obliczyć z następującego wzoru:

$$\alpha = \pi - \omega_2 t_H = \left(1 - \frac{1}{2\sqrt{2}}\sqrt{\left(\frac{r_1}{r_2}+1\right)^3}\right).$$

Jest to różnica w kątach obrotu satelity i ciała docelowego.

6. Opis języka programowania Python

6.1. Wstęp.

Język ten został wydany w 1991r. przez Guio van Rossuma i jest zarządzany przez organizację non-profit Python Software Foundation. Istnieje wiele interpreterów oraz kompilatorów tego języka programowania. Język ten jest zazwyczaj interpretowany jednak można go skompilować, lecz robić się to tylko w specyficznych sytuacjach, np. abu kod działał szybciej.

Python jest językiem programowania o ogólnym przeznaczeniu. Jego filozofia zaprojektowania kładzie nacisk na czytelność kodu oraz produktywność [9-11]. Posiada minimalną, ale zarazem wiele wszechstronnych bibliotek, w tym interfejs programowania aplikacyjnego. Jezyk Pythona z racji, że jest bardzo minimalistyczny definiuje wbudowane obiekty takie jak listy i słowniki [11-13]. Python obsługuje równie programowanie zorientowane obiektowo, programowanie proceduralne i programowanie funkcyjne [14-16].

6.2. Skrypt i wyświetlanie tekstu

Przez skrypt rozumiemy plik tekstowy o rozszerzeniu **.py**. Interpretery Pythona zostały wydane dla wszystkich znaczących systemów operacyjnych – Windows, Linux, MacOS. Zintegrowanym środowiskiem programistycznym, które ułatwia programowanie jest edytor Spyder [17].

Aby wyświetlić tekst w Pythonie należy skorzystać z formuły:

```
print ”Witaj świecie”
```

Tekst między cudzysłowami jest dowolny. W Pythonie napisy zamykane są poprzez pojedynczy lub podwójny cudzysłów. Jeśli tekst ograniczymy z obu stron potrójnymi cudzysłowami to tekst może zajmować kilka wierszy.

Napis możemy przypisać do dowolnej zmiennej np. ”**a**” a następnie skorzystać z operatora indeksowania by wyświetlić określony fragment napisu. Operator indeksowania ma następującą postać:

```
nazwa_zmiennej[od:do:co_ile]
```

gdzie `od:do` oznacza numery porządkowe w napisie.

Do łączenia napisów służy znak ”**+**”:

```
a = ‘witaj swiecie’
b = ‘w 2017 roku’
print a + b
```

Gdy wywołamy napis wyświetli się nam „Witaj swiecie w 2017 roku”

6.3. Liczby

Operacje matematyczne w języku Python są intuicyjne i proste np. 2/5, 2+5, 2*5, 2%5 itd.. Znak % oznacza dzielenie modulo tzn. zwróci resztę z dzielenia. Liczby zmiennoprzecinkowe zapisuje się w Pythonie w następujący sposób: 2.0, 5.0, 10.5 itd..

Wynik naszych obliczeń można zapisać jako połączenie liczby i tekstu np.

```
a = 2.0
b = 5.0
wynik = a/b
poczatek = ‘wynik dzielenia wynosi:’
koniec = ‘co bylo spodziewane’
print ”%s %f %s” %(początek, wynik, koniec)
```

`%s, %f, %s` są odpowiednio – napis, liczba całkowita i liczba zmiennoprzecinkowa. Znaki te wstawione w napis będą zamienione wartościami zmiennych podanych na końcu wiersza.

6.4. Instrukcje warunkowe

Do wykonywania testów „jeśli” służy składnia:

```
if warunek:
     instrukcje
else:
     instrukcje
```

Szczególną uwagę należy zwrócić na wcięcia, gdyż są one obowiązkowe.

Jako warunki podajemy zdarzenia zwracające wartość prawda/fałsz, np.:

== - jest równe

!= - nie jest równe

<> - mniejsze, większe niż

Napisanie w kodzie, że jest a = b jest przypisaniem, natomiast a == b jest porównaniem. Pomijając jeden znak zmieniamy całkowicie sens naszego kodu.

6.5. Pętle

W języku Python mamy też kilka rodzajów pętli. Jednak z najbardziej podstawowych i najpopularniejszy to **while**. Pętla ma postać ogólną:

```
while WARUNEK:
          ZDARZENIA_DLA_PĘTLI
```

Kolejną pętla w Pythonie jest funkcja **range**, która tworzy listę wartości całkowitych od zera do podanej wartości. Możemy też podać trzeci parametr określający przyrost. Dla przykładu:

```
for i in rangle (10):
     print i

print ”Petla 2:”
for i in rangle(3, 5):
     print i

print ”Petla 3:”
for I in rangle(10, 100, 10):
     print i
```

6.6. Funkcje

W Pythonie funkcję definiujemy za pomocą instrukcji **def**. Przykładowo:

```
def nazwa_funkcji(parametr1, parametr2, parametr3)
     #kod funkcji
     return zmienna

print nazwa_funkcji (”Poczatek”)
```

Funkcja return zwraca podane wyrażenie (zmienna, obiekt, itp.) Zmienne w definicji funkcji mogą mieć przypisane wartości domyślne. Natomiast zmienne utworzone wewnątrz funkcji nie są dostępne poza nią, lecz można je zdefiniować jako zmienne globalne za pomocą operatora global.

7. Opis bibliotek Visual w Pythonie

7.1. Informacje ogólne

Vpython jest biblioteką Pythona umożliwiającą prostą i efektywną wizualizację [18,19]. Biblioteka powstała w środowisku akademickim, ale obecnie jest używana również w zastosowaniach przemysłowych. Vpython pozwala użytkownikowi tworzyć obiekty takie jak kule, czy stożki w przestrzeni trójwymiarowej oraz wyświetlać je w oknie. Wszystko to sprawia, iż jest niezwykle łatwo stworzyć prostą wizualizację pozwalając jednocześnie programiście na skupieniu się bardziej na aspekcie obliczeniowym pisanego programu. Prostota VPythona zrobiła z niego narzędzie do ilustrowania prostej fizyki, szczególnie w środowisku szkolnym/edukacyjnym.

7.2. Zastosowanie użytkowe

VPython jest prostym narzędziem do renderowania obiektów trójwymiarowych i prostych wykresów [19]. Jego głównym przeznaczeniem jest edukacja, ale jest stosowany również w placówkach handlowych i naukowych.

7.3. Podstawy Vpython

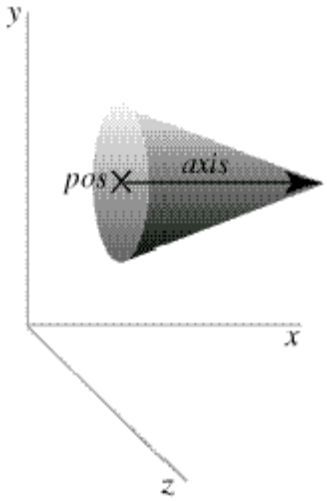
Vpython to biblioteka do wyświetlania grafiki 3D. Została napisana w celu szybkiego tworzenia wizualizacji w środowisku akademickim, natomiast później zaczęto używać jej w przemyśle [19].

Vpython posiada okno wyświetlające, gdzie pokazywane są obrazy 3D. Centrum znajduje się w punkcie (0,0,0), oś – `x` biegnie od lewej do prawej; oś – `y` biegnie od dołu do góry; oś – `z` biegnie w stronę ekranu. Poza oknem wyświetlającym jest równiej okno tak zwane wychodzące gdzie wyświetlany jest tekst. Vpython używa wektorów do pozycjonowania obiektów. Po zdefiniowaniu wektorów (np. a= vector(1,2,3)) możemy wykonywać na nich proste operacje.

W Vpythonie różnorodne obiekty są dostępne do zobrazowania np. cylinder, kula, pudełko, pierścień czy stożek. Tym obiektom można

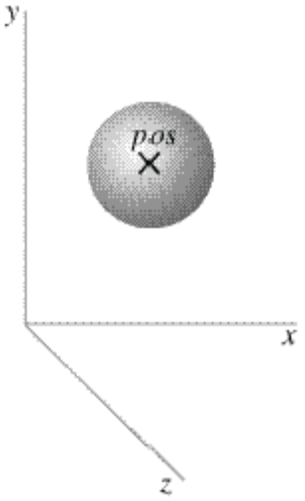
nadawać wiele atrybutów choćby: pozycje, kolor, widzialność bądź niewidzialność. Kolory można opisywać słowami typu: green, blue, yellow ale równiej jako liczby z przedziału 0.0 a 1.0.

Opis stożka składa się z pozycji, osi oraz jednej wartości skalarnej jaką jest promień szerokiego końca stożka.



Rys.5 Cechy stożka w Vpython [19]

Kula posiada wyjątkowo mało cech które ją opisują, mianowicie pozycję, która jest centrum sfery oraz promień tej sfery.



Rys.6 Cechy kuli w VPhyton [19]

7.4. Ruch i rotacje

Większość obiektów, której jesteśmy w stanie stworzyć w Vphythonie jest w stanie się obracać. Aby taką czynność udało się wykonać należy zastosować następujące transformacje do obiektu:
- rotacja kąta w radianach
-przeciwne do wskazówek zegarka wokół linii wyznaczonej przez origin
-przez domyślne obroty wokół własnej pozycji bądź osi obiektu.

W celu wykonania rotacji przez nasz obiekt używa się następującej formuły:

```
obj.rotate(angle=pi/4, axis=axis, origin=pos)
```

Aby nasza animacja zachowywała się w tej sam sposób na komputerach z różną szybkością procesora stosuje się limit klatek animacji. Inaczej mówiąc narzuca się programowi aby wyświetlał określoną ilość naszych klatek na sekundę (funkcja `rate`) [19].
W kolejnej części zostaną opisane wyniki badań.

Część II- badania własne

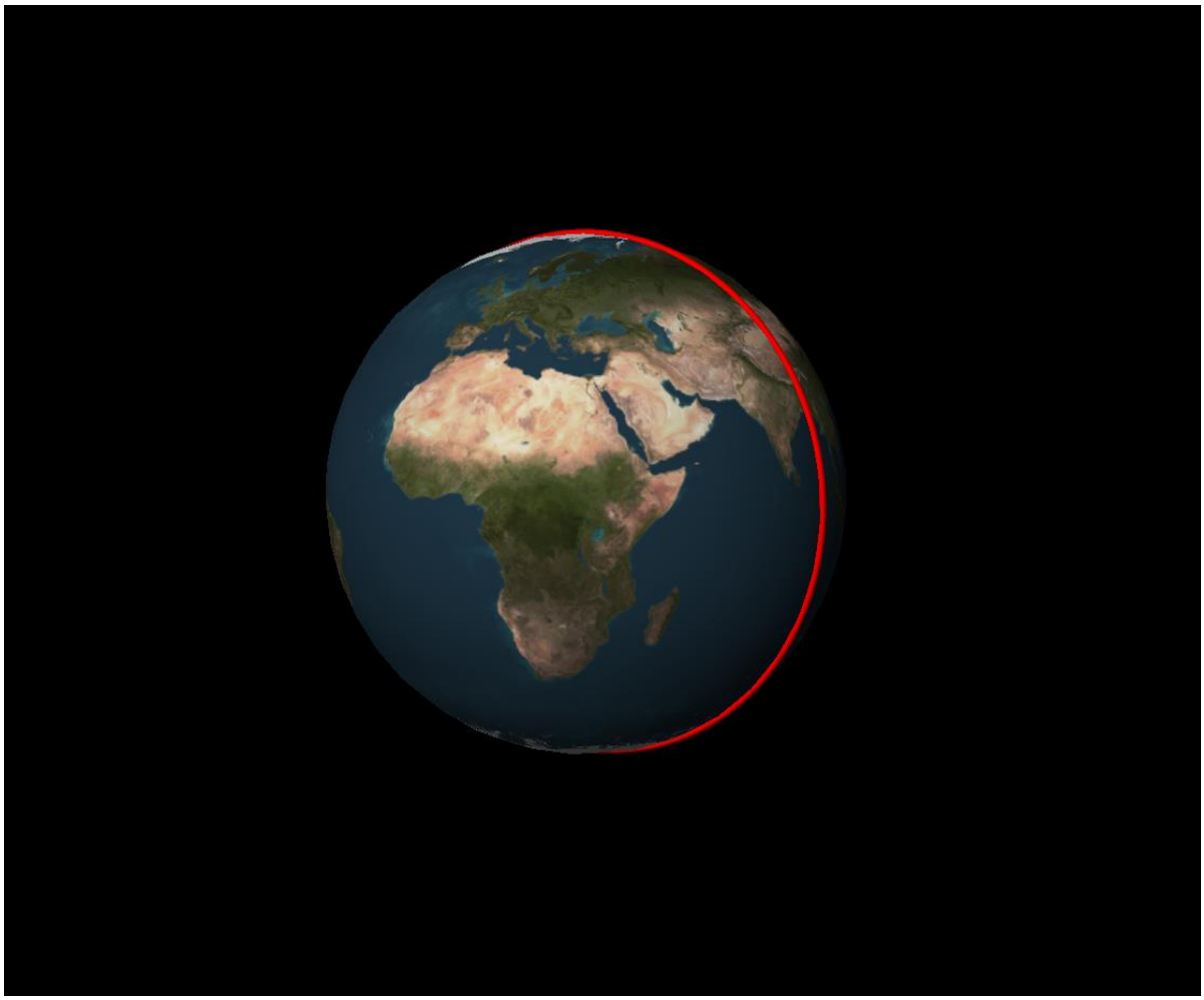
8. Opis programu i wyniki

Symulacje poniżej zostały wykonane na podstawie wskazówek zawartych w [20].

8.1. Wstęp:

Po zapoznaniu teoretycznym orbity Hohmanna postanowiłem podzielić swoją pracę na 4 etapy. W pierwszym etapie napisałem program, który obrazuje orbitę statku Apollo dookoła Ziemi.
Program [Appendix *„Program 1”*] zacząłem od zaimportowania bibliotek Visual oraz matematycznych. Zdefiniowałem również stałą grawitacji G oraz okres czasowy dt. Następnie ustaliłem rozmiar sceny oraz tło. W następnym kroku zdeklarowałem następujące zmienne dla Ziemi: kształt, pozycje, kolor, masę, położenie oraz promień. Kolejno opisałem zmienne dla statku Apollo. Następnie stworzyłem ścieżkę lotu Apollo dookoła Ziemi. W kolejnym kroku opisałem wszystkie pozostałe parametry Ziemi i statku: odchylenie od osi X oraz Y czy granice obramowania. Następny krok już był trudniejszy gdyż musiałem stworzyć pętle, która opisuje ruch Apollo 13 dookoła Ziemi. W tym celu wykorzystałem pętle `while` odpowiednio najpierw obliczając wektorowo zmianę momentu Apollo, a następnie aktualną pozycję statku.

Na załączonym niżej obrazku została zaznaczona orbita statku Apollo 13 kolorem czerwonym, która to znajduje się 173km nad powierzchnią Ziemi. Jest to wystarczająca odległość aby móc orbitować dookoła naszej planety. Statek znajdował się na wysokości 173km, posiadał masę równą 31300kg.

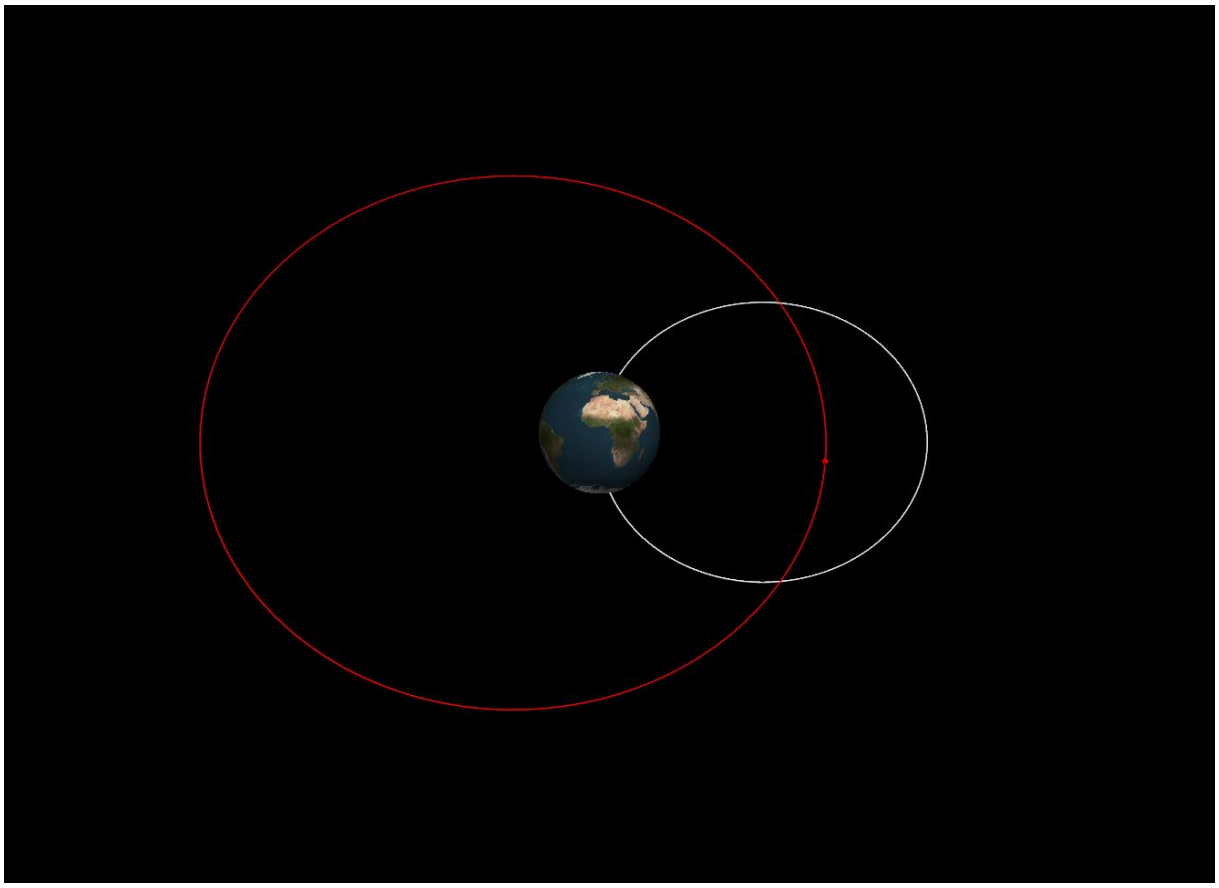


Rys. 7 Orbita statku Apollo 13 dookoła Ziemi

8.2. Drugim krokiem było stworzenie programu, w którym Ziemia oraz statek orbitują wokół wspólnego środka masy [Appendix *„Program 2”*]. Do programu należało dopisać kilka rzeczy. Zacząłem od stworzenia ścieżki orbity Ziemi. Następnie zdeklarowałem moment pędu następująco, moment pędu Ziemi = - moment pędu Apollo aby zachowany został bilans pędów układu izolowanego Ziemia - statek. W pętli wykonywałem te same obliczenia co w programie pierwszym z tym, że musiały one być wykonywane dla Apollo jak i Ziemi jednocześnie. Z tego powodu dodałem kod bardzo podobny jak dla Apollo tylko z parametrami dla Ziemi.

Jak można zaobserwować na niżej zamieszczonym obrazku orbita Ziemi zaznaczona jest kolorem białym, a statku kolorem czerwonym (wielkość statku jest mocno przesadzona, aby można było zaobserwować krążenie

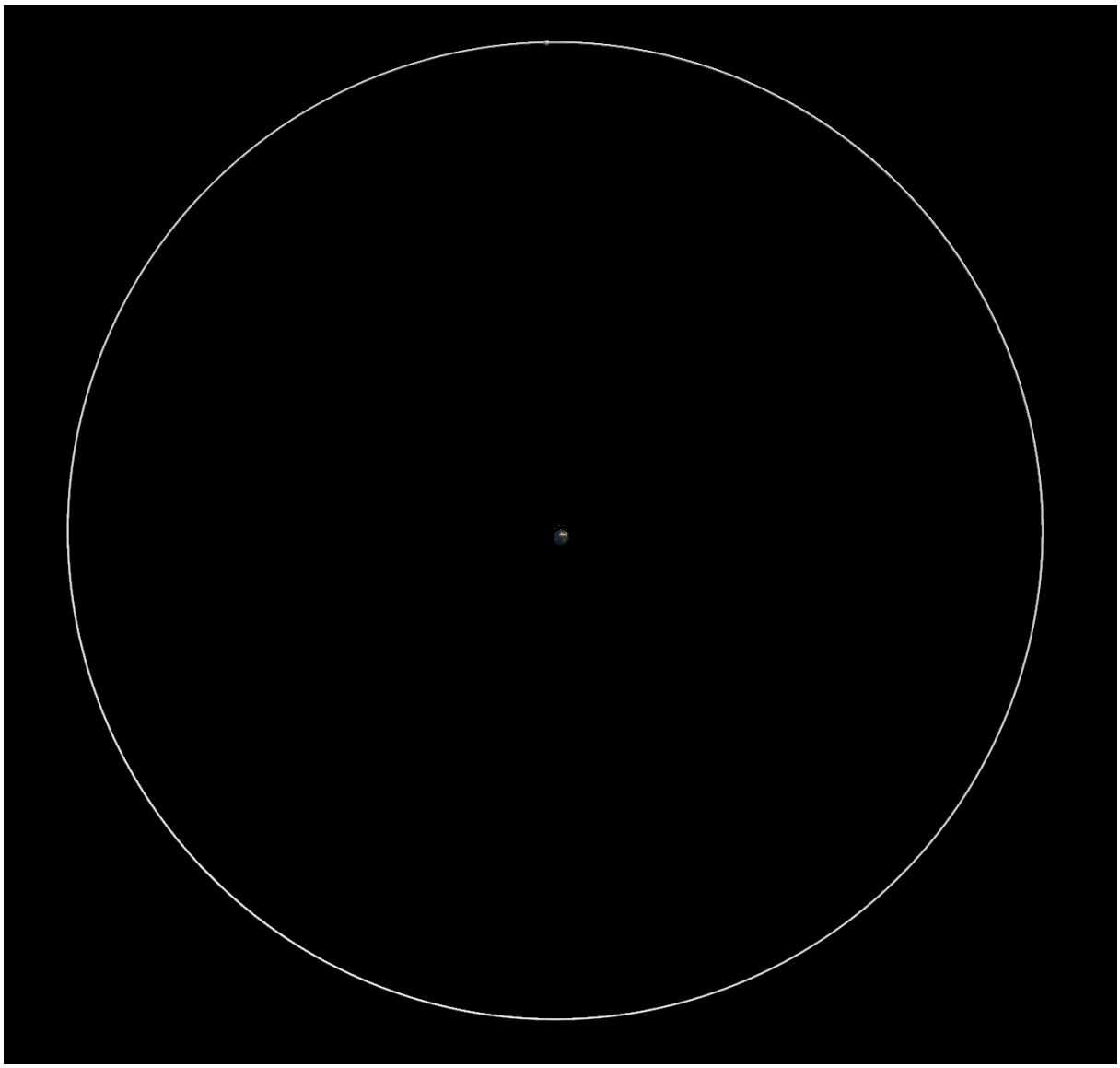
wokół wspólnego środka masy, w rzeczywistości wynosi 31300kg). Widać wspólny środek masy tak więc symulacja zakończyła się powodzeniem.



Rys. 8 Orbita Ziemi i Księżyca wokół wspólnego środka masy

8.3. Trzecim krokiem było dodanie do programu naszego naturalnego satelity [Appendix *„Program 3”*].
Modernizację programu zacząłem od zdeklarowania zmiennych (pozycja, kolor, masa, promień, orbitę) dla Księżyca. Następnie ustalić odchylenie od osi X oraz Y oraz resztę parametrów, które musiałem ustalić podczas opisu Ziemi oraz statku w programie pierwszym. W pętli obliczenia dla Apollo z programu drugiego zamieniłem na obliczenia dla Księżyca odpowiednio zmieniając kod. Obliczenia dla Ziemi pozostały niezmienne.

W tym programie nie symulowałem jeszcze lotu Apollo żeby nie popełnić po drodze błędu. Efekt orbity księżyca dookoła Ziemi można zauważyć na rysunku poniżej.

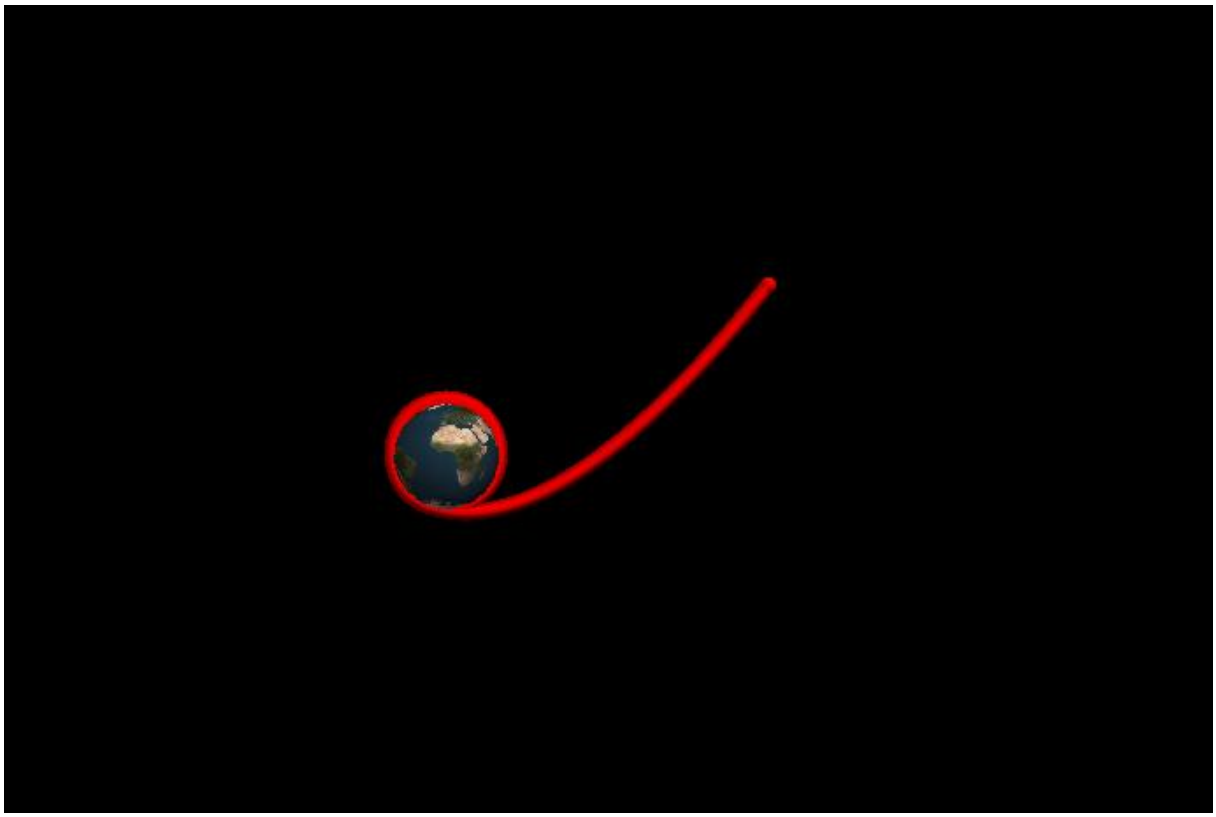


Rys. 9 Orbita Księżyca dookoła Ziemi

8.4. Czwartym i ostatnim krokiem w mojej pracy było dodanie statki orbitującego na orbicie Hohmanna [Appendix *„Program 4”*].
W tym programie już musiałem wprowadzić szereg zmian w kodzie. Zacząłem od ponownego zdefiniowania momentu pędu. Z racji iż mamy trzy obiekty a nie dwa jak to było w poprzednich przypadkach moment ten wygląda następująco, moment pędu Ziemi = - (moment pędu Księżyca + moment pędu Apollo). Następnie przed pętlą wprowadziłem szereg równań na orbitę Hohmanna opisujących promienie, zmianę prędkości, kąty alfa oraz omega. Następnie w pętli musiałem opisać

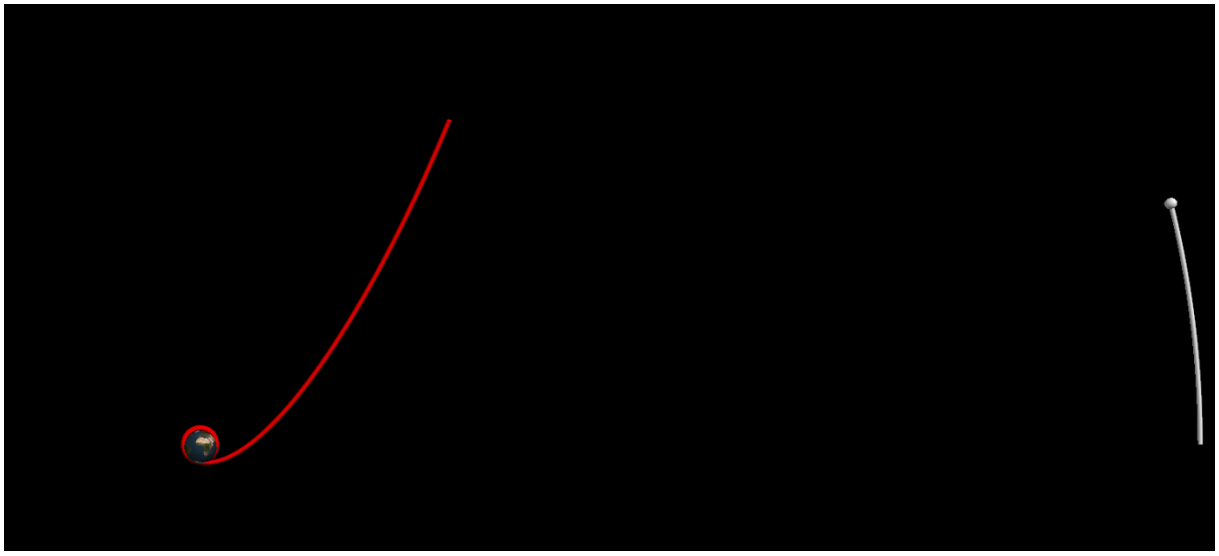
dystans dla trzech przypadków: Ziemia – Księżyc, Księżyc – Apollo, Ziemia – Apollo. Potem zdefiniować siły działające na nasze trzy obiekty. W kolejnym kroku poczyniłem obliczenia dotyczące aktualnej pozycji dla Księżyca, Ziemi oraz Apollo. Kończąc już pętle obliczyłem jeszcze kąt między księżycem a Apollo.

Opis tego programu podzieliłem na cztery ważne etapy.
W pierwszym etapie obserwujemy statek Apollo który po kilku orbitach dookoła Ziemi zostaje wystrzelony na orbitę Hohmanna. Program wylicza idealny kąt aby założenia orbity transferowej były spełnione a następnie kieruje nasz statek w stronę księżyca. Moment opuszczenia orbity okołoziemskiej to moment w którym nasz statek odpala silniki i następuje impuls pozwalający na zmianę typu orbity.



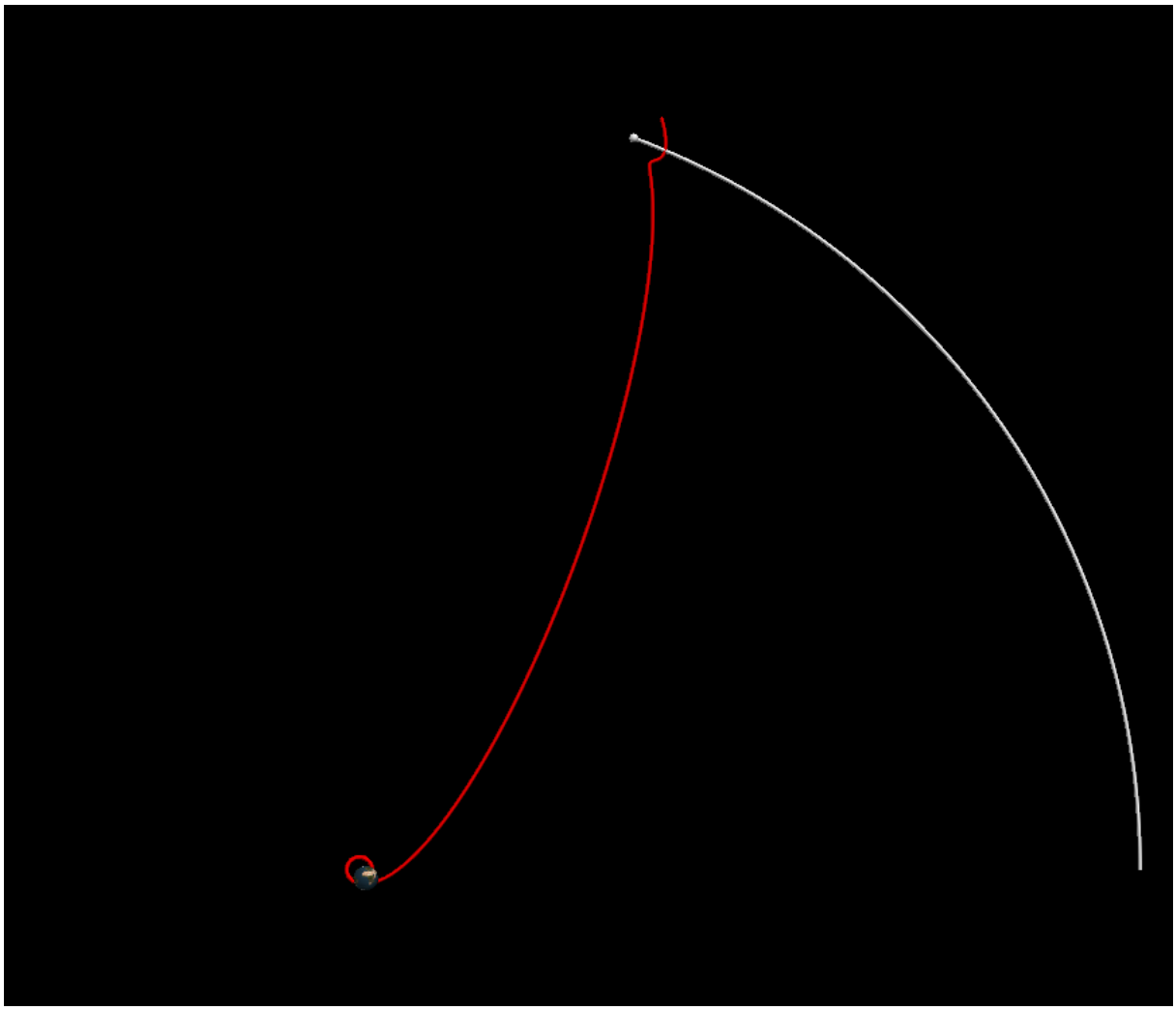
Rys. 10 Apollo 13 opuszczające orbitę Ziemi

Drugim etapem było pokazanie lotu statku kosmicznego w okresie pomiędzy wystrzeleniem z orbity dookoła ziemskiej a dotarciem do naszego naturalnego satelity. Jak widać pęd nadany przez wystrzelenie z orbity okołoziemskiej powoduje duże przyśpieszenie statku w początkowej fazie drogi do Księżyca natomiast w późniejszym etapie nasz statek zwalnia coraz bardziej.



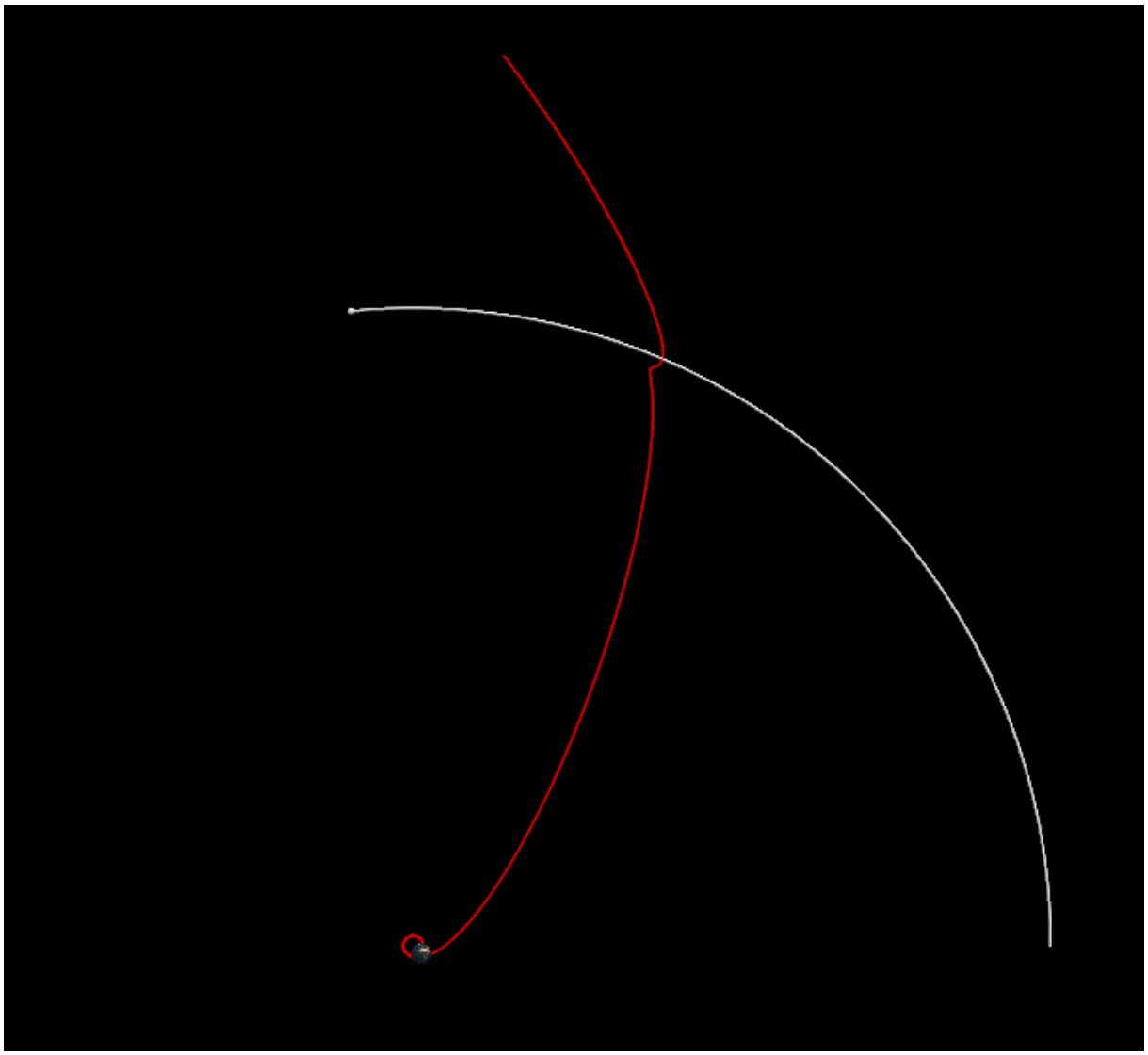
Rys. 11 Apollo 13 będące na orbicie Hohmanna

Trzecim etapem było ukazanie naszego statku w pobliżu Księżyca. Jak widać na Rys. 12, Apollo 13 zakrzywia swoją orbitę w punkcie, w którym mija się z Księżycem. Dzieje się ponieważ Księżyc oddziałuje na statek. Gdyby jednak oddziaływania nie było to nasz statek okrążył by naszego naturalnego satelitę a następnie wrócił bezpiecznie na Ziemię. Oczywiście idealna orbita transferowa Hohmanna nie jest osiąga ze względu na oddziaływanie Księżyca, jednak naukowcom udaje się uzyskać zmodyfikowaną orbitę transferową, lecz wiąże się to z symulacjami wprowadzającymi korekty, które uwzględniają oddziaływanie Księżyca, a korekty orbity wykonywane są na bieżąco podczas lotu statku bądź sondy. Opis teoretyczny nie uwzględnia wszystkich sił i oddziaływań dlatego moje wyniki mają postać jak na załączonym niżej obrazku.



Rys. 12 Apollo 13 mija orbitę Księżyca

Ostatnim czwartym etapem było pokazanie na Rys. 13, co się stało z naszym statkiem kosmiczny po przecięciu z orbitą Księżyca. Apollo 13 korzystał z asysty grawitacyjnej naszego naturalnego satelity po czym można zaobserwować, że kąt dalszej jego orbity różni się znacząco od kąta początkowego zaraz po wystrzeleniu z Ziemi. Jest to manewr zwany asystą grawitacyjną, który jest wykorzystywany do rozpędzenia statków w celu podróży do oddalonych ciał niebieskich przy minimalnym wydatku paliwa.



Rys. 13 Statek Apollo 13 po przejściu przez orbitę Księżyca

9. Podsumowanie

Projekt inżynierski opisywany w tej pracy miał za zadanie obliczenia teoretyczne oraz wizualizację orbity transferowej Hohmanna, wykorzystując do tego środowisko programowania Python. Obliczenia teoretyczne dla orbity Hohmanna opierają się o kilka podstawowych praw fizycznych. Tworzenie wizualizacji zacząłem od nauki języka Python. Zastosowanie bibliotek Visual umożliwiło zaobserwowanie ruchu statku kosmicznego po ów orbicie. W symulacji udało się uzyskać orbitę, którą możemy nazwać transferową. W trakcie projektowania orbity skupiłem się głównie na uproszczeniu kodu oraz poprawie jego czytelności. W opracowaniach naukowych rozważana jest idealna orbita Hohmanna, na która nie bierze pod uwagę oddziaływania Księżyca. W swojej pracy uwzględniłem te siły przez co orbita różni się od tych prezentowanych w literaturze. Symulacja została przygotowana zgodnie z prawami dynamiki Newtona, a jej wyniki pokazują jak wyglądał lot na Księżyc i dodatkowo, jaka jest różnica pomiędzy idealną orbitą Hohmanna i konieczości wprowadzenia korekt do tej orbity.

Bibliografia:

1. https://pl.wikipedia.org/wiki/Manewr_transferowy_Hohmanna [2016r.]
2. https://pl.wikipedia.org/wiki/Hiten [2017r.]
3. https://pl.wikipedia.org/wiki/Principia_mathematica [2017r.]
4. http://www.if.pw.edu.pl/~anadam/WykLadyFO/FoWWW_06.html [2016r.]
5. http://efizyka.net.pl/prawo-powszechnego-ciazenia_6749 [2017r.]
6. https://pl.wikipedia.org/wiki/Pr%C4%99dko%C5%9B%C4%87_kosmiczna#Pierwsza_pr.C4.99dko.C5.9B.C4.87_kosmiczna [2016r.]
7. http://bcs.whfreeman.com/webpub/Ektron/Tipler%20Modern%20Physics%206e/Classical%20Concept%20Review/Chapter_4_CCR_13_Energy_of_a_Particle_in_an_Elliptical_Orbit.pdf [2017r.]
8. http://space.stackexchange.com/questions/1380/how-to-calculate-delta-v-required-for-a-planet-to-planet-hohmann-transfer [2017r.]
9. http://www.bestchinanews.com/Explore/1991.html [2016r.]
10. Programming 3 apollo mission

11. http://www.python.rk.edu.pl/w/p/wprowadzenie-do-pythona/ [2017r.]
12. https://pl.wikipedia.org/wiki/Python [2016r.]
13. https://pl.python.org/docs/ref/ref.html [2016r.]
14. https://pl.wikipedia.org/wiki/Programowanie_obiektowe [2017r.]
15. https://pl.wikipedia.org/wiki/Programowanie_proceduralne [2017r.]
16. https://pl.wikipedia.org/wiki/Programowanie_funkcyjne [2017r.]
17. https://pypi.python.org/pypi/spyder [2017r.]
18. https://en.wikipedia.org/wiki/VPython [2016r.]
19. COMP2720: Automating Tools for New Media-Introduction to Visual Python
20. https://www.google.pl/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=0ahUKEwjw98ef6trRAhXLGCwKHag2CkgQFggdMAA&url=http%3A%2F%2Fnuclear.ucdavis.edu%2F~bhaag%2FvPython%2FVPython%2520Implementation%2FCourse%2520Materials%2FVisual%2520Python%2520Activity%25203%2520Computational%2520Modeling.doc&usg=AFQjCNGtB-6HsUSeQEY2qkwURtJFj0Q3gA&bvm=bv.144686652,d.bGg [2017r.]

Appendix:

Program 1

```
from visual import *
import math

G=6.67e-11
dt=4

scene = display(title = 'Earth and Apollo 13',width = 800, height = 600)
scene.background = (0,0,0)

#####################################################
#   Declaration of Variables                        #
#####################################################

#Declaring variables for earth
earth= sphere(pos=(0,0,0), material=materials.earth)
earth.mass=5.97e24  #mass of the earth
earth.p = vector(0,0,0)*earth.mass
#radius of the earth

#Declaring variables for apollo
apollo=sphere(pos=(173.0+earth.radius,0,0),radius = 100e5,color=color.red)
apollo.mass = 31.3e3 #mass of apollocraft
print( "v_orbit = ", math.sqrt(G*earth.mass/mag(apollo.pos) ) )

apollo.p=vector(0, math.sqrt(G * earth.mass / mag(apollo.pos)
),0)*apollo.mass
apollo.radius= 100e3

#Creating the trails for graphing the orbital path of Apollo
apollo.orbit=curve(pos=[apollo.pos],color=apollo.color,radius = 1e5)


#Labelling Earth and Apollo
earth.name = label(pos = earth.pos,text='Earth',xoffset = 10, yoffset = 10,
space = earth.radius, height = 10, border = 6)
apollo.name = label(pos = apollo.pos,text='Apollo13',xoffset = 10, yoffset
= 10, space = apollo.radius, height = 10, border = 6)

#####################################################
#   Loop for moving the Apollo 13 around earth      #
#####################################################

print("start loop")

while (1==1):
rate(1000)
# Calculating the change in p of apollo
dist = apollo.pos - earth.pos
force = 6.67e-11 * apollo.mass * earth.mass * dist / mag(dist)**3
apollo.p = apollo.p - force*dt
apollo.orbit.append(pos=apollo.pos)
#Calculating the updated position of the Apollo
apollo.pos = apollo.pos + apollo.p/apollo.mass * dt
apollo.orbit.append(pos=apollo.pos)
apollo.name.pos = apollo.pos
```

Program 2

```
from visual import *
import math

G=6.67e-11
dt=10

scene = display(title = 'Earth and Apollo 13',width = 800, height =
600)

##################################################
#   Declaration of Variables                     #
##################################################

#dorbit = 173.0
dorbit = 17300000.0  #testing

#Declaring variables for earth
earth= sphere(pos=(0,0,0), material=materials.earth)
earth.mass=5.97e24  #mass of the earth
earth.radius = 6378.5e3  #radius of the earth
earth.p = vector(0,0,0)*earth.mass


#Creating the trails for graphing the orbital path of Apollo
earth.orbit=curve(pos=[earth.pos],color=earth.color,radius = 1e5)

#Declaring variables for apollo
apollo=sphere(pos=(dorbit+earth.radius,0,0),radius =
100e3,color=color.red)
#apollo.mass = 31.3e3 #mass of apollocraft
apollo.mass = 31.3e23 #mass of apollocraft
print( "v_orbit = ", math.sqrt(G*earth.mass/mag(apollo.pos) ) )

apollo.p=vector(0, math.sqrt(G * earth.mass / mag(apollo.pos)
),0)*apollo.mass
apollo.radius= 100e3

#Creating the trails for graphing the orbital path of Apollo
apollo.orbit=curve(pos=[apollo.pos],color=apollo.color,radius = 1e5)


#Labelling Earth and Apollo
earth.name = label(pos = earth.pos,text='Earth',xoffset = 10,
yoffset = 10, space = earth.radius, height = 10, border = 6)
apollo.name = label(pos = apollo.pos,text='Apollo13',xoffset = 10,
yoffset = 10, space = apollo.radius, height = 10, border = 6)


##################################################
#   Loop for moving the Apollo 13 around earth     #
##################################################

#momentum balacne
```

```
earth.p = - apollo.p

print("start loop")

while (1==1):
    rate(1000)
    # Calculating the change in p of apollo
    dist = apollo.pos - earth.pos
    force = 6.67e-11 * apollo.mass * earth.mass * dist /
mag(dist)**3

    apollo.p = apollo.p - force*dt
    apollo.orbit.append(pos=apollo.pos)
    #Calculating the updated position of the Apollo
    apollo.pos = apollo.pos + apollo.p/apollo.mass * dt
    apollo.orbit.append(pos=apollo.pos)
    apollo.name.pos = apollo.pos

    earth.p = earth.p + force*dt
    earth.orbit.append(pos=earth.pos)
    #Calculating the updated position of the Earth
    earth.pos = earth.pos + earth.p/earth.mass * dt
    earth.orbit.append(pos=earth.pos)
    earth.name.pos = earth.pos
```

Program 3

```
from visual import *
import math

G=6.67e-11
dt=10

scene = display(title = 'Earth and Apollo 13',width = 800, height =
600)

##################################################
#   Declaration of Variables                     #
##################################################

#dorbit = 173.0
dorbit = 17300000.0  #testing

#Declaring variables for earth
earth= sphere(pos=(0,0,0), material=materials.earth)
earth.mass=5.97e24  #mass of the earth
earth.radius = 6378.5e3  #radius of the earth
earth.p = vector(0,0,0)*earth.mass


#Creating the trails for graphing the orbital path of
Apollomaterial=materials.earth
earth.orbit=curve(pos=[earth.pos],color=earth.color,radius = 1e5)

#Declaring variables for apollo
apollo=sphere(pos=(dorbit+earth.radius,0,0),radius =
100e3,color=color.black)
#apollo.mass = 31.3e3 #mass of apollocraft
apollo.mass = 31.3e23 #mass of apollocraft
print( "v_orbit = ", math.sqrt(G*earth.mass/mag(apollo.pos) ) )

apollo.p=vector(0, math.sqrt(G * earth.mass / mag(apollo.pos)
),0)*apollo.mass
apollo.radius= 100e3

#Creating the trails for graphing the orbital path of Apollo
apollo.orbit=curve(pos=[apollo.pos],color=apollo.color,radius = 1e5)


#Declaring variables for Moon
moon=sphere(pos=(384e6,0,0),color=color.white)
moon.mass=7.3480e22
moon.radius=2.238e6
moon.p=vector(0,math.sqrt(G*earth.mass/mag(moon.pos) ),0) *
moon.mass
moon.orbit=curve(pos=[moon.pos],color=moon.color,radius = 1e6)



#Labelling Earth and Apollo
```

```
earth.name = label(pos = earth.pos,text='Earth',xoffset = 10,
yoffset = 10, space = earth.radius, height = 10, border = 6)
apollo.name = label(pos = apollo.pos,text='Apollo13',xoffset = 10,
yoffset = 10, space = apollo.radius, height = 10, border = 6)
moon.name = label(pos = moon.pos,text='Moon',xoffset = 10, yoffset =
10, space = apollo.radius, height = 10, border = 6)


#####################################################
#   Loop for moving the Apollo 13 around earth      #
#####################################################

#momentum balacne
earth.p = - moon.p

print("start loop")

while (1==1):
    rate(10000)
    # Calculating the change in p of apollo
    dist = moon.pos - earth.pos
    force = 6.67e-11 * moon.mass * earth.mass * dist / mag(dist)**3

    moon.p = moon.p - force*dt
    moon.orbit.append(pos=moon.pos)
    #Calculating the updated position of the Moon
    moon.pos = moon.pos + moon.p/moon.mass * dt
    moon.orbit.append(pos=moon.pos)
    moon.name.pos = moon.pos

    earth.p = earth.p + force*dt
    earth.orbit.append(pos=earth.pos)
    #Calculating the updated position of the Earth
    earth.pos = earth.pos + earth.p/earth.mass * dt
    earth.orbit.append(pos=earth.pos)
    earth.name.pos = earth.pos
```

Program 4

```
from visual import *
import math

G=6.67e-11
dt=3

scene = display(title = 'Earth and Apollo 13',width = 2800, height =
1600)

#apollo orbit above sea level
dorbit = 173.0


#Declaring variables for earth
earth= sphere(pos=(0,0,0), material=materials.earth)
earth.mass=5.97e24  #mass of the earth
earth.radius = 6378.5e3  #radius of the earth
earth.p = vector(0,0,0)*earth.mass


#Creating the trails for graphing the orbital path of Apollo
earth.orbit=curve(pos=[earth.pos],color=earth.color,radius = 1e5)

#Declaring variables for apollo
apollo=sphere(pos=(dorbit+earth.radius,0,0),radius =
100e4,color=color.red)
apollo.mass = 31.3e3 #mass of apollocraft
#apollo.mass = 31.3e23 #mass of apollocraft TESTING

apollo.p=vector(0, math.sqrt(G * earth.mass / mag(apollo.pos)
),0)*apollo.mass
apollo.radius= 100e4

#Creating the trails for graphing the orbital path of Apollo
apollo.orbit=curve(pos=[apollo.pos],color=apollo.color,radius = 1e5)


#Declaring variables for Moon
moon=sphere(pos=(384e6,0,0),color=color.white)
moon.mass=7.3480e22
moon.radius=2.238e6
moon.p=vector(0,math.sqrt(G*earth.mass/mag(moon.pos) ),0) *
moon.mass
moon.orbit=curve(pos=[moon.pos],color=moon.color,radius = 1e6)


#Labelling Earth and Apollo
earth.name = label(pos = earth.pos,text='Earth',xoffset = 10,
yoffset = 10, space = earth.radius, height = 10, border = 6)
apollo.name = label(pos = apollo.pos,text='Apollo13',xoffset = 10,
yoffset = 10, space = apollo.radius, height = 10, border = 6)
moon.name = label(pos = moon.pos,text='Moon',xoffset = 10, yoffset =
10, space = apollo.radius, height = 10, border = 6)
```

```
#####################################################
#   Loop for moving the Apollo 13 around earth      #
#####################################################

#momentum balacne
earth.p = - (moon.p + apollo.p)



#Hohmann orbit
r1 = earth.radius + dorbit
r2 = mag(moon.pos)

dv1 = math.sqrt(G * earth.mass / r1) * (math.sqrt(2.0*r2/(r1+r2))-1)
#dv1 = 3190

print("dv1 = ", dv1)

tH = math.pi * math.sqrt( ((r1+r2)**3)/(8.0*G*earth.mass) )
omega2 = math.sqrt( G*earth.mass / r2 ** 3 )
alpha = math.pi - omega2 * tH

print("alpha = ", alpha)

fired = False

epsilon = 0.0001


print("start loop")

t = 0

while (1==1):
    rate(10000)
    # Calculating forces
    distME = moon.pos - earth.pos
    forceME = 6.67e-11 * moon.mass * earth.mass * distME /
mag(distME)**3

    distMA = moon.pos - apollo.pos
    forceMA = 6.67e-11 * moon.mass * apollo.mass * distMA /
mag(distMA)**3

    distEA = earth.pos - apollo.pos
    forceEA = 6.67e-11 * earth.mass * apollo.mass * distEA /
mag(distEA)**3

    forceE = -forceME + forceEA
    forceM = forceME + forceMA
    forceA = -forceMA - forceEA


    moon.p = moon.p - forceM*dt
```

```
    moon.orbit.append(pos=moon.pos)
    #Calculating the updated position of the Moon
    moon.pos = moon.pos + moon.p/moon.mass * dt
    moon.orbit.append(pos=moon.pos)
    moon.name.pos = moon.pos

    earth.p = earth.p - forceE*dt
    earth.orbit.append(pos=earth.pos)
    #Calculating the updated position of the Earth
    earth.pos = earth.pos + earth.p/earth.mass * dt
    earth.orbit.append(pos=earth.pos)
    earth.name.pos = earth.pos


    apollo.p = apollo.p - forceA*dt
    apollo.orbit.append(pos=apollo.pos)
    #Calculating the updated position of the Apollo
    apollo.pos = apollo.pos + apollo.p/apollo.mass * dt
    apollo.orbit.append(pos=apollo.pos)
    apollo.name.pos = apollo.pos

    #angle of moon and apollo
    angle = math.acos( dot(apollo.p, moon.p) /
(mag(apollo.p)*mag(moon.p)) )


    if( fired == False and t > 20.0 ):
        if( abs(angle - alpha)  < epsilon ):
            apollo.p = apollo.p + dv1 * apollo.mass * apollo.p /
mag(apollo.p)
            fired = True
            print("FIRED")
            print("angle = ", angle)
            print("alpha = ", alpha)
            print("diff = ", abs( angle - alpha ))


    t = t + dt
```