Uniwersytet Wrocławski
Wydział Fizyki i Astronomii

Kierunek: Fizyka komputerowa

Igor Zygmunt Kucharski

Nr indeksu: 269724

Symulacja ruchu samolotu w jednorodnym polu grawitacyjnym z wizualizacją 3D

(Simulation of aircraft movement in a homogeneous gravitational field with 3D visualization)

Praca licencjacka

Opiekun pracy
dr hab. Maciej Matyka, prof. UWr

Wrocław, listopad 2019

Składam serdeczne podziękowania
prof. Maciejowi Matyce za
entuzjazm, wiarę, pomoc oraz
cierpliwość i wyrozumiałość okazane
mi podczas studiów i w trakcie
pisania tej pracy.

Spis treści

Symulacja ruchu samolotu w jednorodnym polu grawitacyjnym z wizualizacją 3D

Streszczenie

Niniejsza praca wraz z symulacją opisują ruch samolotu w wyidealizowanych warunkach fizycznych. Głównym celem było stworzenie symulacji wykorzystującej wzory empiryczne; wizualizacja wyników w czasie rzeczywistym oraz ocenienie czy uzyskany efekt jest zbliżony do ruchu prawdziwych odrzutowców.

W celu realizacji zagadnienia wykonano następujące kroki:

1. Stworzono w programie Blender model 3D wzorowany na myśliwcu F-16;
2. Wybrano i opisano elementy samolotu, które mają największy wpływ na jego ruch;
3. Zebrano i omówiono wzory empiryczne określające podstawowe siły działające na odrzutowiec;
4. Używając języka programowania Python i biblioteki Panda3D przygotowano symulację działania w/w sił na model 3D.

W podsumowaniu zawarto spostrzeżenia czy efekt ruchu modelu samolotu uzyskany na wizualizacji jest zbliżony do ruchu rzeczywistych maszyn powietrznych. Biorąc pod uwagę duże uproszczenia symulacji i jej wyidealizowane warunki można stwierdzić, że samolot porusza się w sposób wiarygodny i zbliżony do rzeczywistych maszyn. Wynika więc z tego, że zastosowane wzory empiryczne dobrze opisują ruch statków powietrznych w warunkach zbliżonych do warunków określonych w symulacji.

Słowa kluczowe: fizyka, samolot, symulacja, wizualizacja, Python, Panda3D

Simulation of aircraft movement in a homogeneous gravitational field with 3D visualization

Abstract

This paper and simulation are describing aircraft movement in perfect physical conditions. Main target was to create the simulation using empirical equations; displaying received movement in real time and compare it to the real jet motion.

Following steps has been made to achieve the goal:

1. 3D model based on F-16 jet has been made using Blender software;
2. Most important parts, according to their utility in airplane's movement have been chosen and described;
3. Elemental empirical equations of airplane forces of movement have been collected and described;
4. Simulation of forces, which are affecting 3D model has been prepared using Python programming language and Panda3D library.

In summary remarks if motion of visualized jet is close to real aircrafts movements have been included. Keeping in mind that simulation is very simplified and its conditions are set as perfect it could be said that model’s movements are very close to movements of real airplanes. It is follows that used empirical equations, with used environment parameters, are describing aircrafts motions in good approximation.

Key words: physics, aircraft, airplane, simulation, visualization, Python, Panda3D

I Wprowadzenie

Sztuka latania fascynowała ludzi od zawsze. Chęć oderwania stóp od ziemi i wzbicia się w przestworza powstawała w człowieku z różnych pobudek. Dla uciemiężonych i zniewolonych latanie niczym ptak mogło być symbolem wolności. Umysły marzycieli i poszukiwaczy przygód rozpalały wyobrażenia świata widzianego z góry. Wreszcie władcy oraz dowódcy wojskowi dostrzegali potencjał drzemiący w przewadze, którą można byłoby uzyskać, gdyby udało się poskromić siły natury i zaatakować przeciwnika za pomocą latających statków bądź żołnierzy przerzuconych górą na jego tyły.

Mimo oczywistych korzyści, które daje możliwość podróżowania w powietrzu, długo nie dysponowano technologią i wiedzą pozwalającą na budowę urządzeń, które by na to pozwalały. Wprawdzie możliwe było używanie pocisków miotanych za pomocą łuków, katapult, czy w późniejszym okresie za pomocą prochu, jednak wciąż nie było możliwe stworzenie maszyny zdolnej unieść człowieka i pozwalającej na manewrowanie nią.

Pewnym krokiem w przód było wynalezienie balonów, jednak ich wielkość, stosunkowo mała ładowność, trudności w manewrowaniu oraz niewielka rozwijana prędkość sprawiły, że ich zastosowanie, zwłaszcza militarne, było mocno ograniczone. Nie zmienia to faktu, że rozwój balonów doprowadził do powstania sterowców, służących jako duże, powietrzne statki pasażerskie.

Lotnictwo znane obecnie ma swoją genezę na początku XX wieku [1]. Zaczęły wówczas powstawać pierwsze samoloty napędzane silnikiem, które były zdolne do startu, kontrolowanego lotu i lądowania. Od tego czasu awionika rozwinęła się do tego stopnia, że możliwe stały się międzykontynentalne podróże samolotów pasażerskich zabierających na pokład kilkuset pasażerów; rakiety kosmiczne są w stanie wynosić w kosmos ludzi i satelity, a samoloty myśliwskie mogą operować z kilkukrotnymi prędkościami dźwięku.

Tak intensywny rozwój był możliwy dzięki ogromnej ilości badań i eksperymentów przeprowadzanych przez naukowców i inżynierów na całym świecie. Niemniej jednak, mimo że dalszy rozwój wymaga coraz bardziej zaawansowanych metod i coraz doskonalszych technologii, to podstawy fizyczne lotnictwa pozostają od początku jego istnienia takie same. Celem niniejszej pracy jest stworzenie symulacji i wizualizacji 3D ruchu samolotu w jednorodnym polu grawitacyjnym w oparciu o fundamentalne prawa fizyki.

II Cel, metodyka oraz zakres pracy

Głównym celem pracy jest stworzenie symulacji przedstawiającej ruch samolotu w jednorodnym polu grawitacyjnym wykorzystując zaimplementowane wzory empiryczne, określające siły oddziałujące na odrzutowiec oraz stwierdzenie, czy uzyskany w ten sposób ruch, prezentowany na wizualizacji, jest prawdopodobny i zbliżony do ruchu odrzutowców w rzeczywistości. Zakres pracy obejmuje przede wszystkim:

- Stworzenie modelu 3D samolotu użytego w wizualizacji;
- Zebranie podstawowych definicji używanych w awionice, które opisują elementy konstrukcyjne samolotu oraz określają siły i zjawiska występujące w trakcie lotu;
- Zebranie wzorów odzwierciedlających w/w siły;
- Zaimplementowanie wspomnianych wzorów do programu komputerowego oraz zastosowanie ich do obliczania w czasie rzeczywistym ruchu samolotu o określonych parametrach;
- Przypisanie wyników obliczeń do modelu 3D w celu ich wizualizacji.

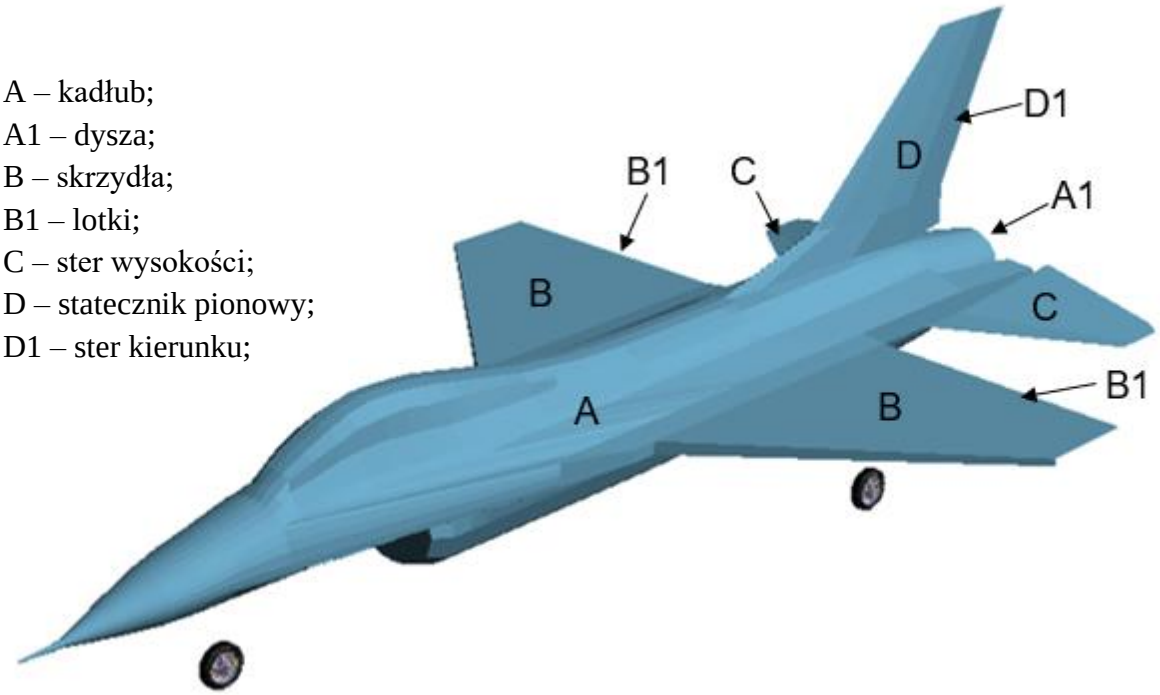
Narzędzia zastosowane w trakcie tworzeni niniejszej pracy to program komputerowy Blender 2.80 użyty do stworzenia modelu 3D samolotu oraz język programowania Python 3.7.4 wraz z biblioteką graficzną Panda3D 1.10.4.1, zastosowane do przeprowadzania obliczeń i wizualizowania ich wyniku jako ruchu trójwymiarowego modelu samolotu.

Parametry jak i współczynniki zastosowane w symulacji bazują na rzeczywistych parametrach samolotu myśliwskiego F-16. Również model 3D jest wzorowany na tej maszynie. Źródłem informacji są publikacje omawiające w/w myśliwiec, broszury oraz dane techniczne udostępniane przez producenta [2]. Ponadto korzystano z udostępnionej w Internecie bazy danych profili lotniczych, materiałów edukacyjnych udostępnionych przez NASA oraz wykładów [3] i publikacji naukowych traktujących o tematyce lotnictwa.

III Model 3D

Model 3D został przygotowany przez autora rozprawy w programie Blender 2.80. Został on stworzony na podstawie rzutów samolotu F-16 dostępnych w Internecie [4]. Na poniższej rycinie znajdują się również symbole rozmieszczone w miejscach, gdzie występują istotne z punktu widzenia tego opracowania, elementy konstrukcyjne, których opis znajduje się w kolejnym rozdziale.

A – kadłub;
A1 – dysza;
B – skrzydła;
B1 – lotki;
C – ster wysokości;
D – statecznik pionowy;
D1 – ster kierunku;



Ryc. 1: Uproszczony model 3D samolotu F-16 stworzony na potrzeby wizualizacji.
Źródło: Opracowanie własne.

IV Definicje

1. **Elementy konstrukcyjne samolotu.**

   A. **Kadłub (ang. body)** – główny element konstrukcyjny samolotu. Do niego przymocowane są wszystkie części nośne m.in. skrzydła. Wzdłuż jego osi działa siła pochodząca z silnika odrzutowego, z którego poprzez dyszę (A1) wyrzucane są z dużą prędkością gazy pozwalające na rozpędzanie samolotu [5].

   B. **Skrzydła (ang. wings)** – najważniejszy element nośny samolotu. To na ich powierzchniach powstaje siła nośna, umożliwiająca wznoszenie się statków powietrznych. W przypadku samolotu F-16 stosowane są skrzydła o profilu

NACA 64-206 [6]. Są rozmieszczone symetrycznie po obu stronach maszyny. W ich tylnej części znajdują się **lotki (ang. ailerons)** (B1), które są powierzchniami sterownymi, służącymi do przechylania samolotu na boki wzdłuż osi symetrii biegnącej poprzez środek masy od jego dzioba do ogona [5].

C. **Ster wysokości (ang. elevator)** – są to rozmieszczone symetrycznie względem kadłuba powierzchnie sterowne. W przypadku samolotu F-16 umieszczone są w tylnej części maszyny. Dzięki zmianie ich wychylenia możliwa jest zmiana kąta natarcia poruszającego się samolotu, a co za tym idzie zmiana wysokości. Obrót poruszającej się maszyny odbywa się wzdłuż horyzontalnie umieszczonej osi prostopadłej do kadłuba i przechodzącej przez środek masy samolotu [5].
D. **Statecznik pionowy (ang. vertical stabilizer)** – służy do stabilizowania lotu w osi pionowej. W jego tylnej części znajduje się **ster kierunku (ang. rudder)** (D1). Jego wychylanie służy do skręcania statkiem powietrznym. Zmiana kierunku odbywa się poprzez obrót samolotu względem osi pionowej przechodzącej prostopadle przez kadłub i środek ciężkości samolotu [5].

2. Podstawowe siły działające na samolot oraz ich reprezentacja w programie.

- **Ciężar ciała (ang. weight)** – jest to siła działająca na każde ciało obdarzone masą, znajdujące się w polu grawitacyjnym [7]. Jest równa iloczynowi całkowitej masy ciała i przyspieszenia grawitacyjnego. W rozpatrywanym przypadku jest to iloczyn masy samolotu i ziemskiego przyspieszenia grawitacyjnego. Zwrot wektora siły ciężkości jest skierowany prostopadle w stronę Ziemi.

$$\overrightarrow{F_g} = m\vec{g}$$

, gdzie:

$\overrightarrow{F_g}$ – ciężar ciała $[N]$;

m – masa ciała $[kg]$;

$\vec{g}$ – przyspieszenie grawitacyjne $\left[\frac{m}{s^2}\right]$;

Reprezentacja w programie:

Pierwszy fragment kodu zawiera początek definicji klasy App, która odzwierciedla środowisko symulacji. W niej zdefiniowane jest przyspieszenie grawitacyjne, które oddziałuje na wszystkie obiekty utworzone za pomocą biblioteki Bullet, których masa została określona. Biblioteka Bullet jest silnikiem fizycznym, wchodzącym w skład biblioteki Panda3D.

Drugi fragment przedstawia początek klasy Jet, wykorzystującej bibliotekę Bullet która pozwala na utworzenie obiektu określonego jako ciało sztywne obdarzone masą – w tym przypadku jest to samolot. W tym fragmencie jest również przedstawiony sposób powiązania biblioteki fizycznej i modelu 3D.

```
class App(ShowBase, BulletWorld):
    """Główna klasa aplikacji."""

    # inicjalizacja klasy;
    def __init__(self):
        # deklaracja, że klasa App dziedziczy z klasy ShowBase;
        ShowBase.__init__(self)
        # deklaracja, że klasa App dziedziczy z klasy BulletWorld;
        BulletWorld.__init__(self)

        # utworzenie środowiska fizycznego self.world za pomocą klasy BulletWorld;
        self.world = BulletWorld()
        # ustawienie wektora przyspieszenia grawitacyjnego w środowisku self.world
        #   na wartość 9,81 m/s^2 skierowaną pionowo w dół;
        self.world.setGravity(Vec3(0, 0, -9.81))
```

```python
class Jet():
    """Klasa definiująca model odrzutowca."""

    # Inicjalizacja klasy;
    def __init__(self, environment):
        # Zdefiniowanie środowiska, w tym przypadku utworzony obiekt
        #   wykorzystuje środowisko self.world z klasy App;
        self.environment = environment

        # Wywołanie funkcji zawierającej zbiór parametrów samolotu;
        self.jetParameters()

        # Wczytanie uprzednio stworzonego modelu 3D;
        self.np = loader.loadModel("F-16.bam")
        self.np.flattenStrong()
        self.mesh = BulletTriangleMesh()
        for geomNP in self.np.findAllMatches('**/+GeomNode'):
            geomNode = geomNP.node()
            ts = geomNP.getTransform(self.np)
            for geom in geomNode.getGeoms():
                self.mesh.addGeom(geom, ts)
        # Przypisanie wczytanego modelu do zmiennej self.shape;
        self.shape = BulletTriangleMeshShape(self.mesh, True)

        # Utworzenie punktu węzłowego self.jetNP o charakterystyce ciała sztywnego;
        self.jetNP = self.environment.render.attachNewNode(BulletRigidBodyNode('Jet'))
        self.jetNP.setPos(0, 0, 2.5)
        self.jetNP.setHpr(0, 0, 0)
        self.jetNP.setCollideMask(BitMask32.allOn())
        self.jetNP.setColor(Vec4(.35, .55, .65, 1))
        # Przypisanie wczytanego do zmiennej self.shape modelu
        #   do punktu węzłowego self.jetNP;
        self.jetNP.node().addShape(self.shape)
        # Przypisanie masy zapisanej w zmiennej self.mass
        #   do punktu self.jetNP, reprezentującego samolot;
        self.jetNP.node().setMass(self.mass)
```

- **Siła ciągu (ang. thrust)** – w omawianym przypadku jest wytwarzana przez silnik odrzutowy zainstalowany w samolocie. Jest siłą mechaniczną, a jej działanie opiera się na trzeciej zasadzie dynamiki Newtona. W wyniku pracy silnika odrzutowego poprzez dyszę wyrzucane są z dużą prędkością masy powietrza i spalin. Reakcją na powstałą w ten sposób siłę jest działające na samolot przyspieszenie, którego zwrot jest przeciwny do przyspieszenia wyrzucanych spalin i powietrza.

$$m_1 \cdot \overrightarrow{a_1} = -m_2 \cdot \overrightarrow{a_2}$$

, gdzie:

m_1 – masa samolotu $[kg]$;

$\overrightarrow{a_1}$ – przyspieszenie samolotu $\left[\frac{m}{s^2}\right]$;

m_2 – masa wyrzuconych spalin $[kg]$;

$\overrightarrow{a_2}$ – przyspieszenie wyrzuconych spalin $\left[\frac{m}{s^2}\right]$;

Reprezentacja w programie:

Poniższy kod zawiera fragment funkcji aktualizującej stan samolotu. Na podstawie macierzy transformacji punktu węzłowego samolotu możliwe jest przeliczanie wektora o kierunku biegnącym od środka do dzioba samolotu. Dzięki normalizacji tego wektora można utworzyć wektor siły ciągu o takim samym zwrocie, ale o wartości równej sile tworzonej przez silnik.

```
def update(self):
    """Funkcja aktualizująca stan samolotu"""

    # Zapisanie macierzy zawierającej położenie w przestrzeni
    #   i obroty punktu węzłowego self.jetNP;
    self.jetCenterMatrix = self.jetNP.getNetTransform().getMat()
    # Wektor ze zwrotem biegnącym od środka samolotu do początku jego dzioba;
    self.jetFrontVector = self.jetCenterMatrix.xformVec(self.jetFrontNP.getPos())
    # Powyższy wektor unormowany;
    self.jetFrontVectorNorm = Vec3(self.jetFrontVector.normalized())
    # Wektor o wartości siły ciągu silnika self.thrust o kierunku
    #   zgodnym z wekorem self.jetFrontVector
    self.jetThrustVector = self.jetFrontVectorNorm * self.thrust
    # Zadziałanie siłą reprezentowaną przez powyższy wektor
    #   na punkt węzłowy reprezentujący środek masy samolotu;
    self.jetNP.node().applyCentralForce(self.jetThrustVector)
```

- **Siła nośna (ang. lift)** [8] – dzięki niej samolot może się wznosić. Powstaje tylko, gdy ciało porusza się w względem ośrodka (cieczy lub gazu). Jest zależna przede wszystkim od prędkości poruszającego się obiektu i jego kształtu. W trakcie trwania ruchu samolotu skrzydło rozdziela strugę powietrza doprowadzając do powstawania różnic ciśnienia pod i nad skrzydłem. Ciśnienie pod skrzydłem jest większe niż nad, co doprowadza do powstania siły nośnej. W zależności od rodzaju samolotu skrzydła mają różne profile. Każdy profil ma wyznaczany eksperymentalnie współczynnik siły nośnej, który zmienia się wraz z kątem natarcia skrzydła. Siła nośna jest skierowana prostopadle do prędkości samolotu, zgodnie z siłą powstającą w wyniku różnicy ciśnienia nad i pod skrzydłem. Wzór [7] na siłę nośną jest następujący:

$$\overrightarrow{F_l} = 0{,}5 \cdot C_l \cdot r \cdot \vec{v}^2 \cdot A$$

, gdzie:

$\overrightarrow{F_l}$ – siła nośna $[N]$;

C_l – współczynnik siły nośnej;

r – gęstość powietrza $\left[\frac{kg}{m^3}\right]$;

$\vec{v}$ – prędkość względem ośrodka $\left[\frac{m}{s}\right]$;

A – powierzchnia skrzydeł $[m^2]$;

Reprezentacja w programie:

Siłę nośną obliczono dla każdej powierzchni z osobna wykorzystując następujące parametry:

- Gęstość powietrza [7] – 1,225 $\left[\frac{kg}{m^3}\right]$;
- Współczynnik siły nośnej został przybliżony na podstawie danych eksperymentalnych [9] następującym wzorem: $C_l = 1{,}9 \cdot \sin(2 \cdot \alpha + 0{,}2)$, gdzie: α – kąt natarcia (kąt pomiędzy płaszczyzną powierzchni nośnej,
  a wektorem prędkości) $[rad]$.

```
def liftCoeffCalculate(self, angle):
    """Funkcja służąca do obliczania współczynnika siły nośnej"""

    return round(1.9 * sin(2 * radians(angle) + 0.2), 2)
```

```
# Kąt natarcia w płaszczyźnie pionowej
self.angleOfAttackVertical = round(self.jetVelocityVectorNorm.signedAngleDeg(\
    self.jetFrontVectorNorm, self.jetRightWingEdgeVectorNorm),2)
# Kąt natarcia w płaszczyźnie poziomej
self.angleOAttackHorizontal = round(self.jetVelocityVectorNorm.signedAngleDeg(\
    self.jetFrontVectorNorm, -self.jetUpVectorNorm), 2)
```

- Skrzydła (powierzchnie statyczne) – każde po 8,85 m^2;

```
# Obliczanie współczynnika siły nośnej:
coeff = self.liftCoeffCalculate(self.angleOfAttackVertical)
# Obliczanie siły nośnej:
self.liftForceValue = 0.5 * self.environment.airDensity \
* self.wingsStaticArea * (self.jetVelocityValue)**2 * coeff
# Pomnożenie siły nośnej przez wektor jednostkowy o zwrocie do góry samolotu:
self.jetLiftVector = self.jetUpVectorNorm * self.liftForceValue
# Przyłożenie wektora siły nośnej skrzydeł do modelu samolotu:
self.jetNP.node().applyForce(self.jetLiftVector, self.jetWingsPoint)
```

- Statecznik pionowy (powierzchnia statyczna) – 4,20 m^2

  (kod w programie analogiczny jak dla skrzydeł);
- Ster wysokości (powierzchnia sterowna) – 11,80 m^2;

```
# Zdefiniowanie macierzy obrotu:
self.elevatorsRotationMatrix = Mat4()
# Zapisanie w macierzy zadanego obrotu powierzchni sterownej:
self.elevatorsRotationMatrix.setRotateMatNormaxis( \
    self.elevatorsAngle, self.jetRightWingEdgeVectorNorm)
# Utworzenie wektora skierowanego pod w/w kątem:
self.elevatorsAbsoluteVector = \
    self.elevatorsRotationMatrix.xformVec(self.jetFrontVectorNorm)
# Normalizacja w/w wektora:
self.elevatorsAbsoluteVectorNorm = \
    self.elevatorsAbsoluteVector.normalized()
# Obliczenie kąta pomiędzy w/w wektorem a wektorem prędkości:
self.elevatorsAbsoluteAngle = round(self.jetVelocityVectorNorm.signedAngleDeg( \
    self.elevatorsAbsoluteVectorNorm, self.jetRightWingEdgeVectorNorm),2)
# Obliczenie skierowanego w górę wektora prostopadłego do powierzchni sterownej:
self.elevatorsAbsoluteVectorOrto = \
    self.elevatorsRotationMatrix.xformVec(self.jetUpVectorNorm)
# Normalizacja w/w wektora:
self.elevatorsAbsoluteVectorOrtoNorm = \
    self.elevatorsAbsoluteVectorOrto.normalized()
# Obliczanie współczynnika siły nośnej:
coeff = self.liftCoeffCalculate(self.elevatorsAbsoluteAngle)
# Obliczanie siły nośnej:
self.elevatorsForceValue = 0.5 * self.environment.airDensity \
    * self.elevatorsArea * (self.jetVelocityValue)**2 * coeff
# Pomnożenie siły nośnej przez wektor jednostkowy o zwrocie do góry płaszczyzny:
self.elevatorsLiftVector = \
    self.elevatorsAbsoluteVectorOrtoNorm * self.elevatorsForceValue
# Przyłożenie wektora siły nośnej płaszczyzny sterownej do modelu samolotu:
self.jetNP.node().applyForce(self.elevatorsLiftVector, self.jetElevatorsPoint)
```

- Lotki na skrzydłach (powierzchnie sterowne) – każda po 1,15 m^2 *(kod w programie analogiczny jak dla steru wysokości)*;
- Ster kierunku (powierzchnia sterowna) – 1,40 m^2 *(kod w programie analogiczny jak dla steru wysokości)*;

- **Siła oporu (ang. drag)** [10] – powstaje w trakcie trwania ruchu samolotu względem powietrza. Zależy od prędkości i ma zwrot do niej skierowany przeciwnie. Opór jest wytwarzany przez każdą powierzchnię wystawioną na ruch powietrza. Można go podzielić na opór wynikający z kształtu i tzw. opór indukowany. W pierwszym przypadku wskutek przepływu powietrza wokół ciała powstają lokalne zmiany prędkości oraz ciśnienia, które oddziałując na powierzchnie powoduje powstawanie siły oporu. Opływowe ukształtowanie elementów konstrukcyjnych pozwala ograniczyć ten rodzaj oporu. Drugi rodzaj oporu jest związany z siłą nośną. Powstaje ze względu na to, że poziomy przepływ powietrza w pobliżu krańców skrzydeł jest zaburzony. Różnice ciśnienia nad i pod skrzydłem (które wytwarzają siłę nośną) nie są w stanie się wyrównać w sposób niegwałtowny, co doprowadza do powstawania wirów powietrza przy krawędziach. Powstawanie tego typu wirów z punktu widzenia statku powietrznego powoduje straty energii. Opór indukowany określa, ile energii jest rozpraszane, a co za tym idzie, o ile musi wzrosnąć siła ciągu silnika, aby te straty zrównoważyć. Całkowita siła oporu, używana w niniejszej pracy, zdefiniowana jest wzorem [7]:

$$\overrightarrow{F_d} = 0{,}5 \cdot C_d \cdot r \cdot \vec{v}^2 \cdot A$$

, gdzie:

$\overrightarrow{F_d}$ – siła oporu [N];
C_d – współczynnik siły oporu;
r – gęstość powietrza $\left[\frac{kg}{m^3}\right]$;
$\vec{v}$ – prędkość względem ośrodka $\left[\frac{m}{s}\right]$;
A – powierzchnia skrzydeł [m^2];

Reprezentacja w programie:

Do obliczenia siły oporu przyjęto te same parametry, co w przypadku obliczania siły nośnej.

- Współczynnik siły oporu przybliżono następująco, korzystając z danych dostępnych w internetowej bazie profili lotniczych [11]. $C_d = 0{,}001 \cdot \alpha^2 + 0{,}01$, gdzie: α – kąt natarcia (kąt pomiędzy płaszczyzną powierzchni nośnej, a wektorem prędkości) $[deg]$.

```
def dragCoeffCalculate(self, angle):
    """Funkcja służąca do obliczania współczynnika siły oporu"""

    return round(0.001 * angle**2 + 0.01, 2)
```

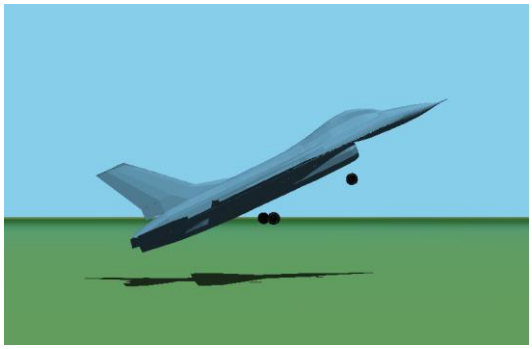
- Dla każdej z powierzchni nośnych zastosowano w programie analogiczne obliczenia siły oporu jak w przypadku skrzydeł:

```
# Obliczanie współczynnika siły oporu:
coeff = self.dragCoeffCalculate(self.angleOfAttackVertical)
# Obliczanie siły oporu:
self.wingsDragForceValue = 0.5 * self.environment.airDensity \
* self.wingsStaticArea * (self.jetVelocityValue)**2 * coeff
# Pomnożenie siły oporu przez wektor jednostkowy
# o zwrocie przeciwnym do prędkości samolotu:
self.jetDragVector = -self.jetVelocityVectorNorm * self.wingsDragForceValue
# Przyłożenie wektora siły oporu skrzydeł do modelu samolotu:
self.jetNP.node().applyForce(self.jetDragVector, self.jetWingsPoint)
```

V Opis symulacji.

Symulacja działa w sposób ciągły od momentu włączenia programu aż do jego wyłączenia przez użytkownika. Po uruchomieniu następuje inicjalizacja środowiska, terenu, samolotu itp. Gdy wszystkie obiekty zostaną wczytane rozpoczyna się wizualizacja przedstawiająca model 3D samolotu, który porusza się zgodnie z obliczonymi w programie siłami. Wizualizacja wyświetla się ze stałą częstotliwością 50 klatek / sekundę. Oznacza to, że wyświetlany obraz jest płynny, a obliczenia nie wpływają na jakość animacji. W trakcie trwania każdej klatki animacji program realizuje jednocześnie dwa zadania: wyświetla obraz wizualizujący wyniki obliczeń z klatki poprzedniej oraz wykonuje obliczenia, które będą wizualizowane w trakcie trwania klatki kolejnej.

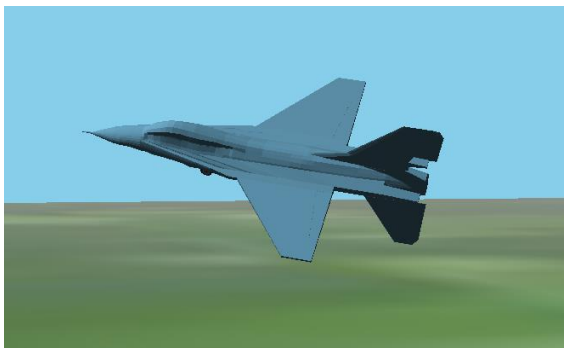
Po każdym rozpoczęciu symulacji samolot stoi na ziemi w tym samym miejscu. Każdy jego ruch jest inicjowany przez użytkownika. Początkowo jedyną akcją, mogącą wpłynąć na zmianę stanu jest zwiększenie siły ciągu silnika (klawisz: strzałka w górę, zmniejszenie siły: strzałka w dół). Samolot zacznie wówczas poruszać się do przodu z przyspieszeniem proporcjonalnym do zadanej siły (jej wartość wyświetlana jest w konsoli). Po puszczeniu klawisza siła ciągu silnika pozostaje taka, jaka została ustawiona. W przypadku braku dalszych akcji użytkownika samolot będzie przyspieszał, aż siły oporów zrównoważą siłę ciągu silnika – wówczas samolot będzie jechał po ziemi ze stałą prędkością, o ile zadana siła ciągu silnika była stosunkowo niska. Gdy jednak siła ciągu silnika będzie odpowiednio wysoka to samolot zacznie się delikatnie wznosić. Wynika to z faktu, że przy zerowym kącie ustawienia steru wysokości współczynnik siły nośnej ma określoną minimalną, dodatnią wartość, co powoduje powstawanie siły nośnej, a tym samym wznoszenie się samolotu do góry.



Ryc. 2: Wznoszenie się samolotu na symulacji.
Źródło: opracowanie własne.

Fot. 1: Wznoszenie się samolotu w rzeczywistości.
Źródło: https://farm2.static.flickr.com/
1123/5098585327_e179ca20cc.jpg
(wejście: 20 listopada 2019)

Aby zwiększyć kąt natarcia samolotu, a co za tym idzie szybkość wznoszenia należy wychylić ster wysokości do przodu (klawisz W, zmniejszenie kąta natarcia: klawisz S). Po puszczeniu klawisza każdy ster wraca do pozycji bazowej. Jeżeli wychylenie steru wysokości będzie duże, samolot mocno zwiększy swój kąt natarcia, a prędkość będzie stosunkowo niewysoka może dojść do tzw. przeciągnięcia (ang. stall), czyli spadku siły nośnej, a co za tym idzie utraty wysokości.



Ryc. 3: Obrót samolotu na symulacji.
Źródło: opracowanie własne.

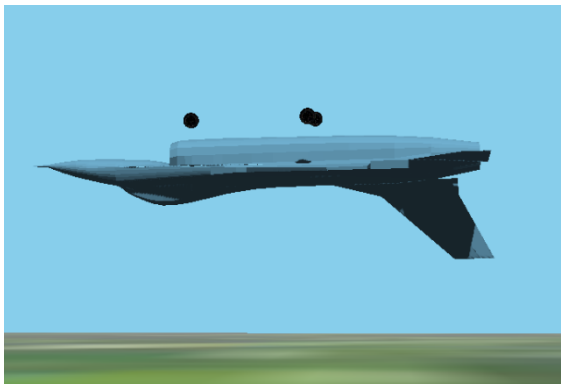
Fot. 2: Obrót samolotu w rzeczywistości.
Źródło: https://farm4.staticflickr.com/
3126/2812914132_bfa5a2ff17_b.jpg
(wejście: 20 listopada 2019)

Gdy samolot osiągnie odpowiednią wysokość użytkownik może przystąpić do obrotu samolotu na lewe skrzydło (klawisz A) lub na prawe (klawisz D) używając lotek. Tak jak w przypadku dużego kąta przy wznoszeniu również i w tej sytuacji nastąpi spadek siły nośnej, jeżeli skrzydło będzie pod kątem zbliżonym do prostego w stosunku do ziemi. Aby dokonać nawrotu należy w trakcie obracania samolotu na skrzydło wychylić również do przodu ster wysokości.

Ostatnim manewrem dostępnym w symulacji jest obracanie samolotu w płaszczyźnie równoległej do skrzydeł. Służy do tego ster kierunku (skręt w lewo: strzałka w lewo, skręt w prawo: strzałka w prawo).

Symulacja obsługuje kolizje samolotu z ziemią, jednak nie przewiduje uszkodzeń, więc co za tym idzie samolot w wizualizacji jest niezniszczalny. W przypadku zderzenia z ziemią dalszy lot jest wciąż możliwy.

Samolot można obserwować z kilku ujęć. Do ich zmian służą klawisze 0-5. Możliwe jest również wyświetlenie układu współrzędnych klawiszem F2. Pokazana jest również prosta o kierunku zgodnym z siłą wypadkową. Klawisz Tab przełącza układ współrzędnych pomiędzy układem ortogonalnym a układem względem samolotu. Klawisz F3 wyświetla debuger graficzny.



Ryc. 4: Lot „na plecach" na symulacji.
Źródło: opracowanie własne.

Fot. 3: Lot „na plecach" w rzeczywistości.
Źródło: https://www.airforcemedicine.af.mil/News/Photos/igphoto/2000640299/ (wejście: 20 listopada 2019)

VI Podsumowanie

Praca miała na celu przedstawienie całego procesu poruszania się samolotu w jednorodnym polu grawitacyjnym przy stałej gęstości powietrza z wykorzystaniem wcześniej przedstawionych wzorów oraz empiryczną weryfikację ich prawidłowego działania w porównaniu do samolotów odrzutowych obserwowanych w realnym świecie.

Biorąc pod uwagę duże, w stosunku do rzeczywistości, uproszczenia w symulowanej konstrukcji samolotu, wyidealizowane środowisko symulacji oraz nieskomplikowany matematycznie opis sił oddziałujących na poruszający się statek powietrzny można stwierdzić, że symulacja odzwierciedla w prawdopodobny sposób ruch wizualizowanego samolotu.

VII Literatura

[1] Internet, https://encyklopedia.pwn.pl/haslo/Lotnictwo-wazniejsze-wydarzenia-z-historii;446946.html (wejście: 9 września 2019);

[2] Internet, https://www.lockheedmartin.com/en-us/products/f-16.html, (wejście: 4 września 2019);

[3] Internet, http://www.dept.aoe.vt.edu/~mason/Mason_f/F16S04.pdf (wejście: 2 września 2019);

[4] Internet, https://www.the-blueprints.com/blueprints/modernplanes/general-dynamics/60961/view/general_dynamics_f_16a_fighting_falcon/ (wejście: 24 sierpnia 2019);

[5] Bielawski, R. (2015). *Wybrane zagadnienia z budowy statków powietrznych. Definicje, pojęcia i klasyfikacje.* Warszawa: Wydawnictwo Akademii Obrony Narodowej;

[6] Aslam, A., Nabil Sabhi Mohd Jafri, M. i Fadhli Zulkafli , M. (2017, Grudzień). Numerical study of military airfoils design for compressible flow. *ARPN Journal of Engineering and Applied Sciences*, strony 7129-7133;

[7] Stöcker, H. (2010). Nowoczesne Kompendium Fizyki. Warszawa: Wydawnictwo Naukowe PWN;

[8] Internet, https://wright.nasa.gov/airplane/lifteq.html (wejście: 4 września 2019);

[9] Nguyen, L., Ogburn, M., Gilbert, W., Kibler, K., Brown, P. i Deal, P. (1979). Simulator Study of Stall/Post-Stall Characteristics of a Fighter Airplane With Relaxed Longitudinal Static Stability. NASA;

[10] Internet, https://wright.nasa.gov/airplane/drageq.html (wejście: 4 września 2019);

[11] Internet, http://airfoiltools.com/airfoil/details?airfoil=naca64206-il (wejście: 4 września 2019).

VIII Spis rycin.

IX Spis fotografii.