MILESTONE #2

1. Opis zadania symulacji w formie słownej

Zadaniem symulacji jest przeprowadzenie startu oraz lotu rakiety wraz z uwzględnieniem wielkości wpływających. Sama rakieta będzie składać się z komponentów/elementów składowych – takich jak między innymi silnik czy zbiornik paliwa od których parametrów będzie zależał przebieg symulacji. Ponadto uwzględnione zostaną także zewnętrze czynniki, takie jak między innymi wiatr, które także będą miły wpływ na przebieg symulacji. Za zakończenie symulacji uznany będzie moment w którym rakieta osiągnie zadany pułap.

1. Analiza czasownikowo – rzeczownikowa

Potrzeby będzie obiekt – rakieta. Na rzecz którego zostanie przeprowadzona symulacja. Będzie on zawierał **obiekty składowe**, których atrybuty będą miały wpływ na przebieg symulacji. Chcemy, aby **umożliwiały one wyłuskanie** **ilości paliwa zatankowanego do rakiety a także ciągu silników**. W symulacji będzie istniał także obiekt odpowiadający za symulację czynników zewnętrznych. Szczególnym przypadkiem tego obiektu będzie **obiekt „wiatr”**, którego atrybuty także będą miały wpływ na przebieg symulacji. Chcemy, aby **umożliwiał on wyłuskanie** **kierunku oraz siły wiatru**. Potrzebny będzie **obiekt**, który **w oparciu o atrybuty wyżej wspominanych obiektów będzie w stanie wyznaczyć** **tor oraz prędkość lotu** rakiety. Potrzeby będzie także **obiekt**, **który sprawdzi czy rakieta osiągnęła** **zadany pułap** i zakończy symulację.

1. Diagram przypadków użycia

Wprowadzenie ilości paliwa zatankowanego do rakiety

Wprowadzenie wartości ciągu silników

Wprowadzenie kierunku oraz siły wiatru

Wprowadzenie wartości zamierzonego pułapu

System

Aktor

1. Karty CRC

**Classname: Rocket**

**Superclass: none**

**Subclass/es: none**

**Responsibilities:**

**Collaborations:**

**IEngine**

**ITank**

**Classname: Engine**

**Superclass: IEngine**

**Subclass/es: none**

**Responsibilities:**

**Gets thrust value**

**Collaborations:**

**Classname: Wind**

**Superclass: IWind**

**Subclass/es: none**

**Responsibilities:**

**Gets wind velocity**

**Gets wind direction**

**Collaborations:**

**Classname: Tank**

**Superclass: ITank**

**Subclass/es: none**

**Responsibilities:**

**Gets fuel capacity**

**Collaborations:**

**Classname: Calculate**

**Superclass: ICalculate**

**Subclass/es: none**

**Responsibilities:**

**Gets ceiling**

**Calculates current height**

**Checks ending condition**

**Collaborations:**

1. Diagramy klas

rocketSimulator

**C**

Rocket

- name: String

**I**

IEngine

- getThrustValue():void

**I**

ITank

- getFuelCapacity():void

**C**

Engine

- name: String

- model: String

- thrustValue: double

**C**

Tank

- name: String

- model: String

- fuelCapacity: double

+ Engine (String, String, double)

+ getThrustValue(): void

+ Tank (String, String, double)

+ getFuelCapacity(): void

+ Wind(double, String)

+ getVelocity():void

+ getDirection():void

+ getVelocity():void

+ getDirection():void

**I**

**A**

ExternalConditions

**C**

Wind

- velocity: double

- direction: String

IWind

rocketSimulator

rocketSimulator

ICalculate

**I**

+ getCeiling():void

+ calCurrentHeight():void

+ checkEndingCondition():boolean

Calculate

**C**

+Calculate (double, double)

+ getCeiling():void

+ calCurrentHeight():void

+ checkEndingCondition():boolean

- ceiling: double

- currentHeight: double