Mini-Project:

Mars Lander

OBJECT ORIENTED PRORAMMING (OOP)

Student: Alexander Stefanov

**Introduction:**

The aim of this report is to provide a brief description of an application representing a Mars landing simulation created using python, to give an overview of the functions and features of the game and to describe how it was coded.

**Infrastructure:**

The game is created in Python 3 using the PyGame environment and the tools it provides. A set of different libraries (random, time, math) are used to help design the full functionality of the game. We use resources given to us to implement the outlook of our application (background, scoreboard, rover, pads, obstacles and meteors).

**Controls:**

Left Rotation Key – Left key

Right Rotation Key – Right key

Engine Key – Spacebar

Exit Key – Esc

New Game – Any key

**Design and Features:**

The game has a scoreboard consisting of time played, fuel left, damage taken, altitude, x and y velocity, score and lives left. The environment permits left and right movement, firing the thrust and going up, going through the sides, landing on a landing pad. The environment prohibits going through the top or landing on the surface of Mars.



The scoreboard changes throughout playing the game. With every press of the spacebar, the engine starts, executing the thrust, the fuel is used and the volume lowers itself until it runs out. When the fuel runs out, all of the controls are blocked and the rover is unable to move so it crashes down. The score evaluation consists of how many times you have won the game. Every win (perfect execution of a landing on a pad) will add 50 points to your score. Every loss (landing on the surface, landing on the side of the pad, landing in the wrong position in the pad) will take one of your lives.

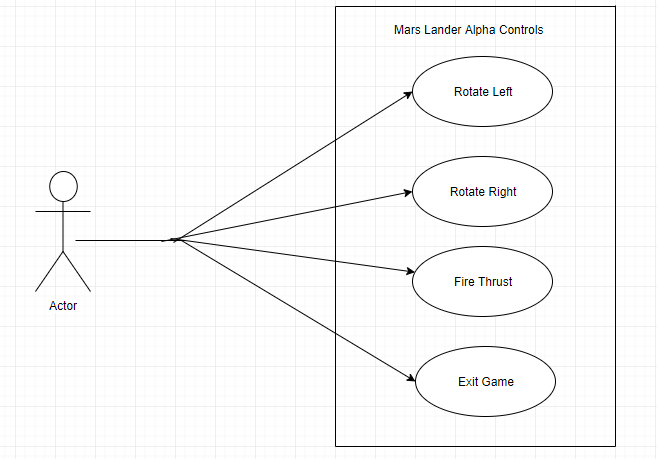
****

The game has obstacles and random meteor storms. If they hit you, they will cause damage from 10 points up to 25 points. When you reach 100 damage, your rover will stop reacting as it is no longer able to operate. It will fall down and crash. Game will restart if you have lives left.

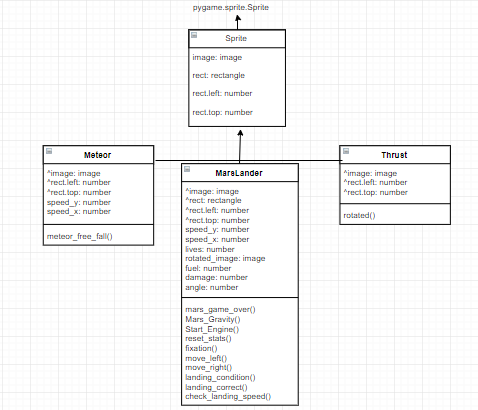
There are 3 randomly inserted landing pads. There are 6 fixed obstacles in the environment. There are 7 randomly inserted meteors that will form a random storm.



**Use Cases:**

****

**UML:**

****

**References:**

PyGame Documentation - <https://www.pygame.org/docs/>

Helpful articles - <http://renesd.blogspot.co.uk/2017/03/pixel-perfect-collision-detection-in.html>

**Testing:**

**The method I used when testing my application was the bottom up approach quote often. I also used print statements when I was not sure if my code that was related to time and clock-ticks was working.**