CS1527 Object-Oriented Programming 2017-2018 Mini-Project 2: Mars Lander

Created by: Hasan Ahmedov

ID: 51768277

1. Mars Lander

This mini project was all about creating an updated version of one of the first arcade games to be based on a real space mission and using pseudo-realistic physics. It is called Lunar Lander by Atari and was launched back in 1979.

Mars Lander game was designed and built using the Python pygame package which is an Open Source library for making multimedia applications and games. It was pretty challenging and required to bring all the knowledge and skills one has acquired during the CS1527 course. However, it was all fun and worth the effort kind of experience which I believe motivated me and vastly expanded my knowledge in the field of Pygame and Python.

2. Description

The player begins with 3 “lives” (Figure 1), with the lander positioned at the top of the screen with random vertical and horizontal velocities. It also has 500 kg of fuel being reduced by 5 every time the main engine thrust is fired (“space” key).

Figure 1. *Game start.*

Whenever the lander goes beyond the right or the left of the screen it wraps onto the opposite side of it. Besides, the Mars lander is not allowed to fly off the top of the screen.

At the top left hand corner of the game window the user is provided with instrument panel which displays the changing time, altitude, fuel reserves, velocity (x, y), damage sustained and lives left. The player's main goal is to successfully land on one of the three landing pads placed on different locations on the Martian surface. Proper land adds up 50 points to the total game score which the user should aim to increase as much as possible. Furthermore, "You Have Landed Successfully! +50 pts" message (Figure 2) is shown on the screen while the program is paused and waits for a key press.

Figure 2. *Successful landing.*

Each time the lander starts with 0% damage which instantly jumps to 100% in case the lander reaches the bottom of the screen or a hard landing occurs. As seen in Figure 3, his results in a “You Have Crashed!” message being displayed on the game screen and costs the player 1 life and the game pauses till a key is pressed. Moreover, the Mars lander’s velocity (y) is incremented by a small amount every game cycle in order to introduce the acceleration due to gravity

Figure 3. *Crash message.*

effect.

There are random control failures which occur for a duration of 2 secs preventing the player from using one randomly chosen control feature of the lander. The user is notified by the “*ALERT*” message (Figure 4) displayed at

Figure 4. *Alert message and meteor storm.*

the bottom left hand side of the instrument panel.

The program also demonstrates 5 fixed position obstacle sprites which cause 10% damage to the lander when hit which destroys them (makes them invisible). It also displays random meteor storms consisting of 5 to 10 meteors every one of which dealing 25% damage to the Mars lander. If the lander sustains 100% damage (meteors and/or obstacles), all the controls are disabled (resulting in a crash).

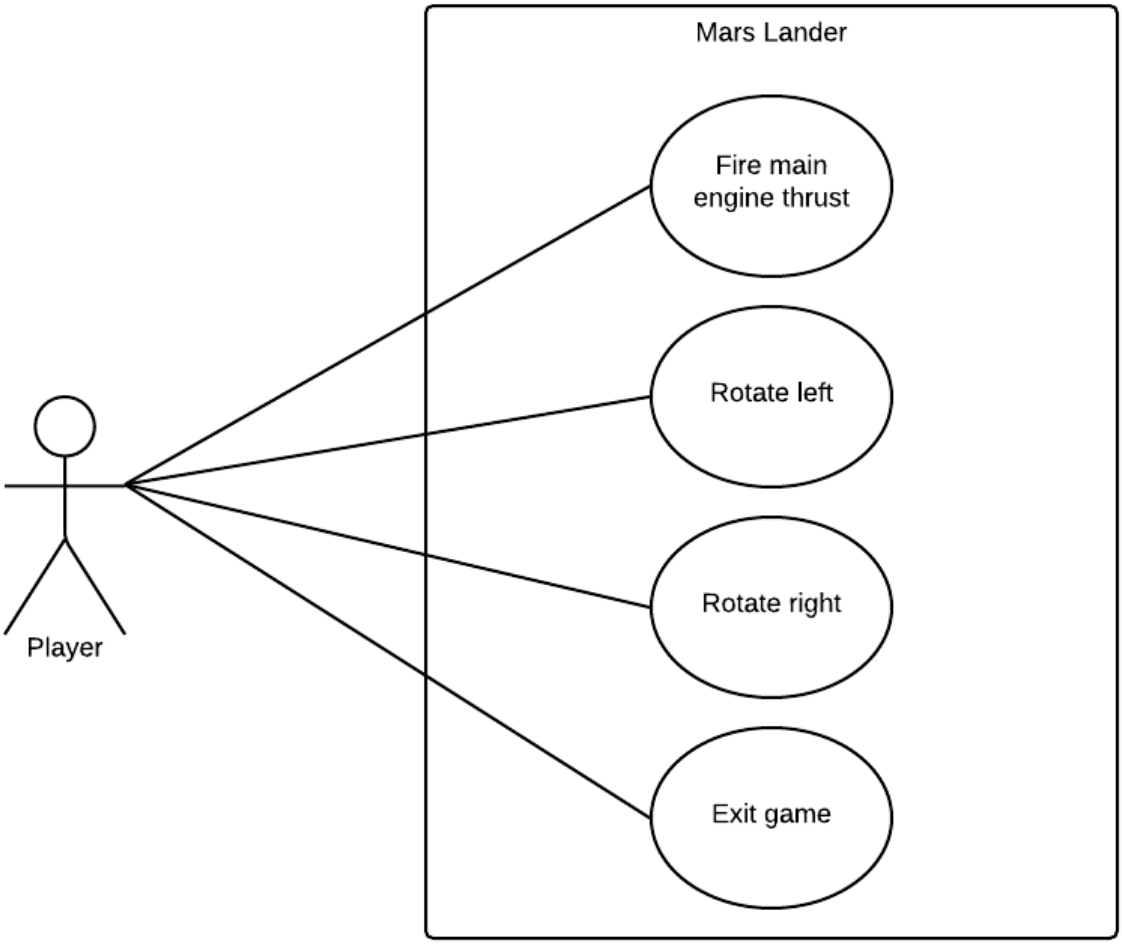
If one wishes to exit the game they can do so by pressing the “Esc” key or by clicking the red “X” button.

3. Testing

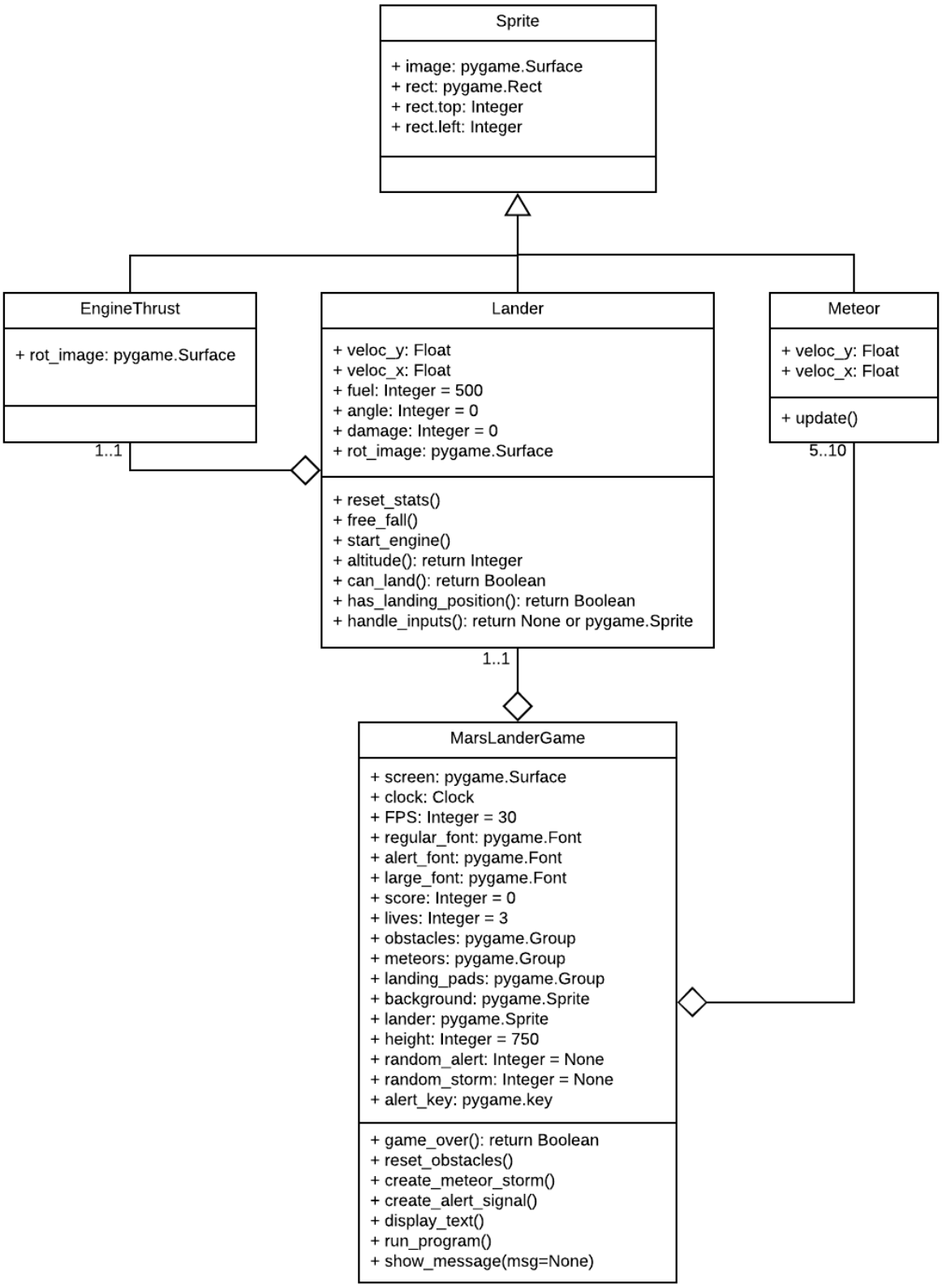
The aim of testing is to make one understand more about the program or application under test. For this project I used the Bottom-Up testing approach to ensure it’s doing what I intend it to do. I ran the program every time I implemented a new feature in the code to analyze the output. It was the clearest and easiest way for me to check for errors. One of the main advantages of this approach is that it offers easier observation of test results. Besides, it does, indeed, have some disadvantages but at the end of the day it helped me reach my goal and understand how the Pygame package behaves.

4. UML

- UML Use Case Diagram:



UML Class Diagram:



References: https://www.pygame.org/docs/, https://stackoverflow.com/