Period 2

William Haratsch

Group Name: William Haratsch

Project Title: Space Lander

Final Document

Project Description: In this project, the player controls a moon lander. The objective is to land the moon lander on a moon surface in the specified target area. If the player lands too fast, or in the wrong orientation, or hits the surface of the moon, then the lander will be destroyed and the player loses the game. Otherwise, the player wins a point. The goal is to go on a winning streak, without destroying the lander.

Functionalities:

Classes:

- Asteroid class. This class holds various instance fields belonging to the Asteroid objects, including float asterSize, which holds the value for the width of the displayed asteroid. The class also holds the Asteroid() constructor, which sets instance fields, including position and speed vectors, to random values within a specified range. Lastly, the class holds the updateAsteroid() method, which updates the position of the asteroid, in addition to checking whether the asteroid is out of bounds of the display. The purpose of this class is to be able to create multiple Asteroid objects that can be displayed on the screen.

- Bullet class. This class holds instance vectors, which are assigned in the Bullet() constructor within the class. The class also holds the updateBullet() method, which changes the position of the bullet every frame, checks if it's out of bounds, and checks for collision with an asteroid object. The purpose of this class is to be able to create multiple bullet objects, such as to allow the player to shoot multiple bullets at asteroids in order to destroy them.
- Lander class. This class holds instance vectors, such as lander position (pos), lander speed (speed), and gravity, which are assigned in the Lander() constructor within the class. The class also holds the updateLander() method, which updates the vectors of the lander object every frame, thus changing the position of the lander on the screen. This method also checks if the lander is out of bounds, and sets the "exploding" boolean to true to allow for the explosion animation to play whenever the lander hits the predefined "ground" on-screen. The purpose of this class is to keep the methods and instance fields pertaining to the Lander object in one, organized location.

Methods:

- Setup() method. This method sets the size of the display, instantiates the asteroid objects, instantiates the arrayList of bullets, instantiates the lander object, sets the values of numerous global variables such as explosionIndex, calls methods loadResources() and createArrays() which load images and sets the values of the keys[] and explosion[] arrays, and lastly sets the frame rate to 60 frames per second. This method can be described as the "initiation" method, in that it sets all the default starting values needed to get the rest of the program running.

- Draw() method. This method allows the images to be seen on the display by calling the drawEnvironment() method. It also calls the checkKeys(), drawScore(), player.updateLander(), asteroidUpdater(), and collider() methods. The method also assigns a new bullet object for each bullet in the bullet arrayList, and subsequently calls the bullet.updateBullet() method for that bullet object. Additionally, the draw() method continuously prints the coordinates of the lander's position in the command console. Lastly, the method checks if the exploding sequence has been initiated, and if it has, the method calls the exploderLander() method, which initiates the explosion animation for when the lander crashes on the surface.
- drawEnvironment() method. This method draws the environment, including the lunar surface and the lunar lander. It ensures that the position of the lunar lander on the screen is updated every frame. It also implements the illumination of the landing zone with blinking yellow lights, and adds a lander shadow which changes shape and location depending on the position of the lander. Lastly, the method both calls the drawAsteroids() method, which spreads out the respawning of asteroid objects over a greater period of time, and displays the asteroid on the screen, updating its position every frame.
- updateLander() method. This method updates the values of the lander every frame, namely its acceleration, speed, and corresponding position. The method also checks if the lander has landed in one of the two designated landing zones, and updates the player score if it is. The method checks if the lander touched the "ground," and if it does, it calls the reset() method, but not before setting the

"exploding" boolean to true, thus setting the condition for the explosion animation in the exploderLander() method called in draw().

- reset() method. This method returns the lunar lander to its starting position, and resets basic variables such as position, speed, and acceleration.
- keyPressed(), keyReleased(), and checkKeys() methods. These methods take in user input through the arrow buttons. With these methods, the user can use the arrow buttons to move the lander left, right, and up, in addition to pressing the "enter" key to call the reset() method.
- createArrays() method. This method sets a size for the keys[] array, in addition to setting the value of each PImage in the explosion[] array with the use of a double for-loop.
- drawAsteroids() method. This method ensures that each asteroid is being displayed at intervals of 3 seconds by setting a boolean to "true". This is to ensure that all asteroids won't all respawn on the screen at once, because that would be overwhelming for the player.
- asteroidUpdater() method. This method is a continuation of the drawAsteroids() method, and if the boolean for the corresponding asteroid object returns true, then the updateAsteroid() method for that asteroid object will be called.
- collider() method. This method checks if an asteroid collides with the player-controlled lander. If the asteroid has, then the reset() method is called.
- exploderLander() method. This method checks if the explosionIndex global integer is less than the length of the arrayList of PImage explosion objects. If so, the frame of the explosion updates to the next PImage in the arrayList. The

purpose of this method is to show the explosion, frame by frame, at the former position of the lunar lander at the time of its crash.

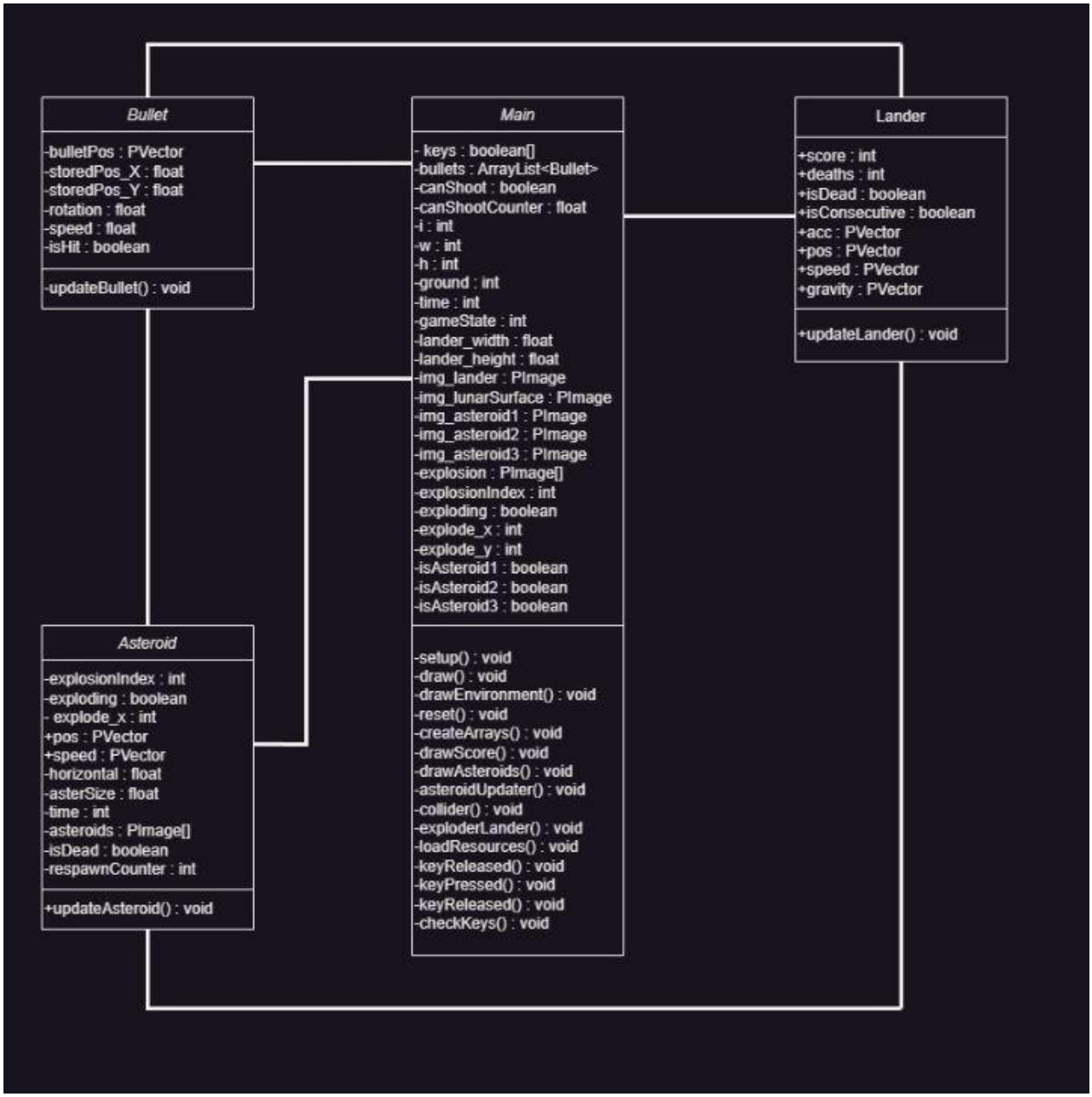
- updateAsteroid() method. This method updates the position and speed vectors of the asteroid objects. It also checks if the asteroid hit the in-game “ground level”. If so, its instance variables are reset.
- updateBullet() method. This method updates the position of each bullet, and removes a bullet if the bullet is out of bounds on the screen or if the bullet hits an asteroid.
- loadResources() method. This method loads the downloaded images in the Processing file, and assigns them to a PImage.

Cool features:

- Illumination of the landing zone. The landing zone is marked by an ellipse, which blinks yellow to grab the player’s attention. This landing zone illumination is implemented in the drawEnvironment() method. When a player lands their lunar lander on the landing zone, the reset() method is called.
- Implementation of a shadow for the lunar lander. This shadow’s position changes with the position of the lunar lander, and gets larger if the lander is farther away from the “ground,” and “smaller” if the lander is closer to the ground. The shadow also gets fainter the further away the lander is from the ground.
- Implementation of the score-keeping functionality. The player’s total score and total deaths are displayed as text in the top-right-hand corner of the screen. Total score denotes the number of consecutive successful landings in the landing zone conducted by the player, and resets after an unsuccessful landing attempt.

- Implementation of multiple landing zones to choose from for landing on the lunar surface.
- Asteroids. Asteroids fall into the lunar landscape from outer space, and if an asteroid collides with the player's lander, the player's lander is destroyed, and the reset() method is called. This is implemented in the Asteroid class and in the collision() method of the "main" file.
- Shooting mechanics. With a mouse click to shoot, and mouse cursor to aim, the player can shoot bullets from their lander in order to protect their lander from collisions with asteroids. When a bullet collides with an asteroid, both the bullet and the asteroid will disappear. This is implemented in the Bullet class, and makes use of an arrayList of bullet objects.
- Wrap-around. When the moon lander hits the screen's boundary, it wraps around, such that its position is now on the opposite side of the screen compared to where it once was. This is implemented in the updateLander() method of the lander class.
- Explosion animation. When the lunar lander reaches the ground position, a series of 64 images is played, frame by frame, which animates an explosion at the point of impact on the ground. This is done with the use of the "explosion" array of PImage objects, the explosionIndex, exploding, explode_x, and explode_y variables. If the "exploding" boolean returns true, then the exploderLander() method is called, which does the actual image display.

UML Diagram:



How to Play: Press the left, right, and up arrow keys to move the lunar lander left, right, and up. Shoot and destroy asteroids by clicking your mouse, and aim with the mouse cursor. Press “enter” to restart the game. Remember: the objective of the game is to gently land the lander in one of the landing zones enclosed by a flashing yellow ring.