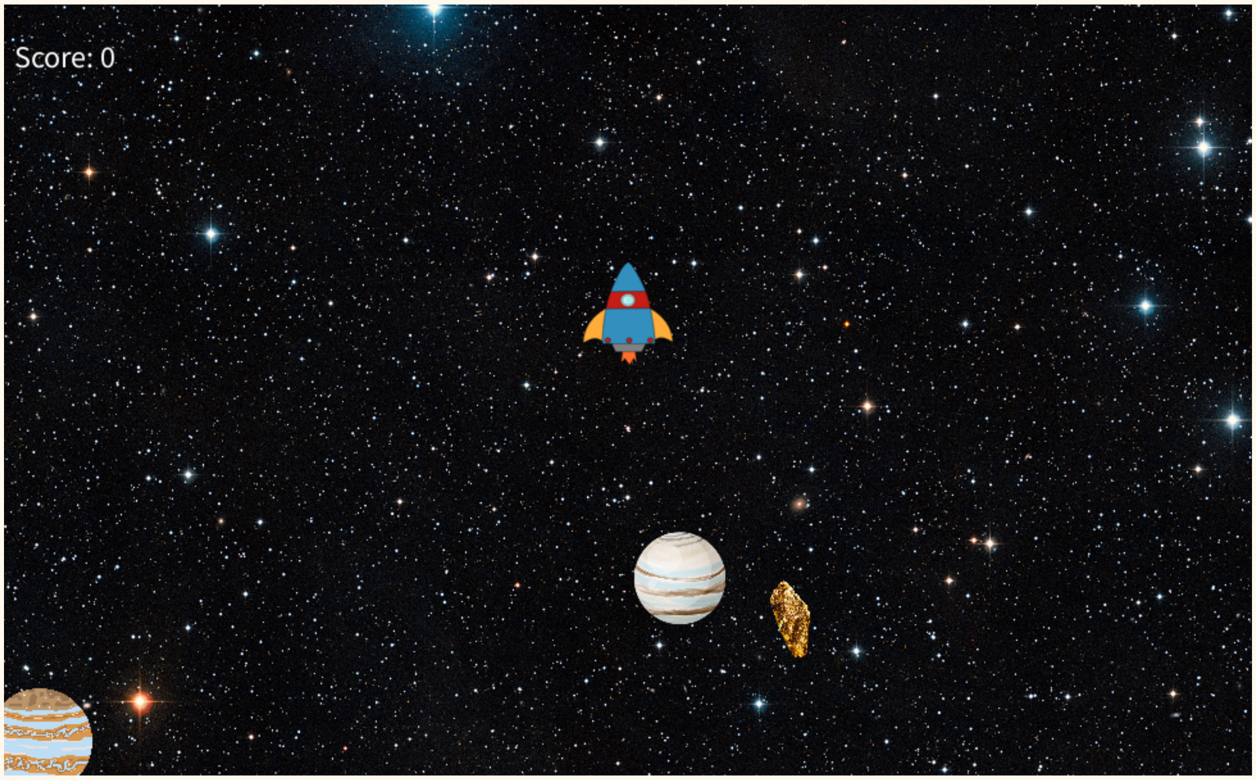
AP Computer Science A PERIOD 7
MS. Novillo

The Space Race
End Of Year Project 22-23

—

By Sophia Dasser



Description

The Space Race was inspired by the Flappy Bird mainly with the functionality of the user to only have the ability to go up and down. I complicated the Flappy Bird concept by incorporating moving planets that you can dodge as well as golden asteroids that you can collect to increase your score. The planets speed up as you go further into the game, and it gets harder to dodge them. Additionally, the number of golden asteroids decreases in generation (the frame rate at which the asteroid is generated decreases) which further increases the difficulty of the game. You can press the space key for the rocket to go upwards and if you die, press the R key to reset the game!

Functionalities:

The code is composed of three classes not including the main class. In order to understand the functionalities of this game we can walk through the functionality of each class:

1. **The "main" class**: This class sets up the game and generates new planets and coins as the game continues. Because the draw method is in this class, the main class also keeps up with the status of the play (whether alive or dead) and reacts accordingly, either continuing to call the loops in the draw method which generate the planets and allow the rocket to move or call the reset and dead() methods in order to allow the user to start from the beginning.
2. **The rocket class**: This class determines the position, speed, and y direction acceleration of the rocket. The y-direction acceleration is determined by the space key that the user presses. In the main class, within the keyPressed method, the rocket jump() method is called and this adds -10 to the y coordinate of the rocket. The update method within the rocket class also determines the gravity at which the rocket falls which is a vital part of flappy bird. Finally, the rocket class will also keep record of the rocket's position. If it is offscreen it will return a false boolean which will call the dead() method when called in the draw method.
3. **The planet class**: Arguably one of the hardest classes to code, this class determines the speed at which the planets generate, the speed at which they move, the size of the planet, their position, whether or not the rocket has hit the planet and in the original version of the game whether or not the rocket passed the planet (this would have given points to the user but I decided to make it harder). Also this class keeps record of the position of the planet and if it becomes "offScreen" the planet will be removed from the array, which effectively erases it from the screen. The planet is removed in the array when the isOffScreen method is called in the draw function.
4. **Last but not least the coin class**: This class was generally much easier to code after the snake game project and the planet class. This class simply keeps track of the number of coins "collectedBy" the rocket, hence the method name collectedBy and I decided to experiment with a second draw method which worked! The coins do not speed up or slow down in this game, however, they are generated less and less as the game progresses which is determined by the frame rate found in the main draw method.

Additional Information:

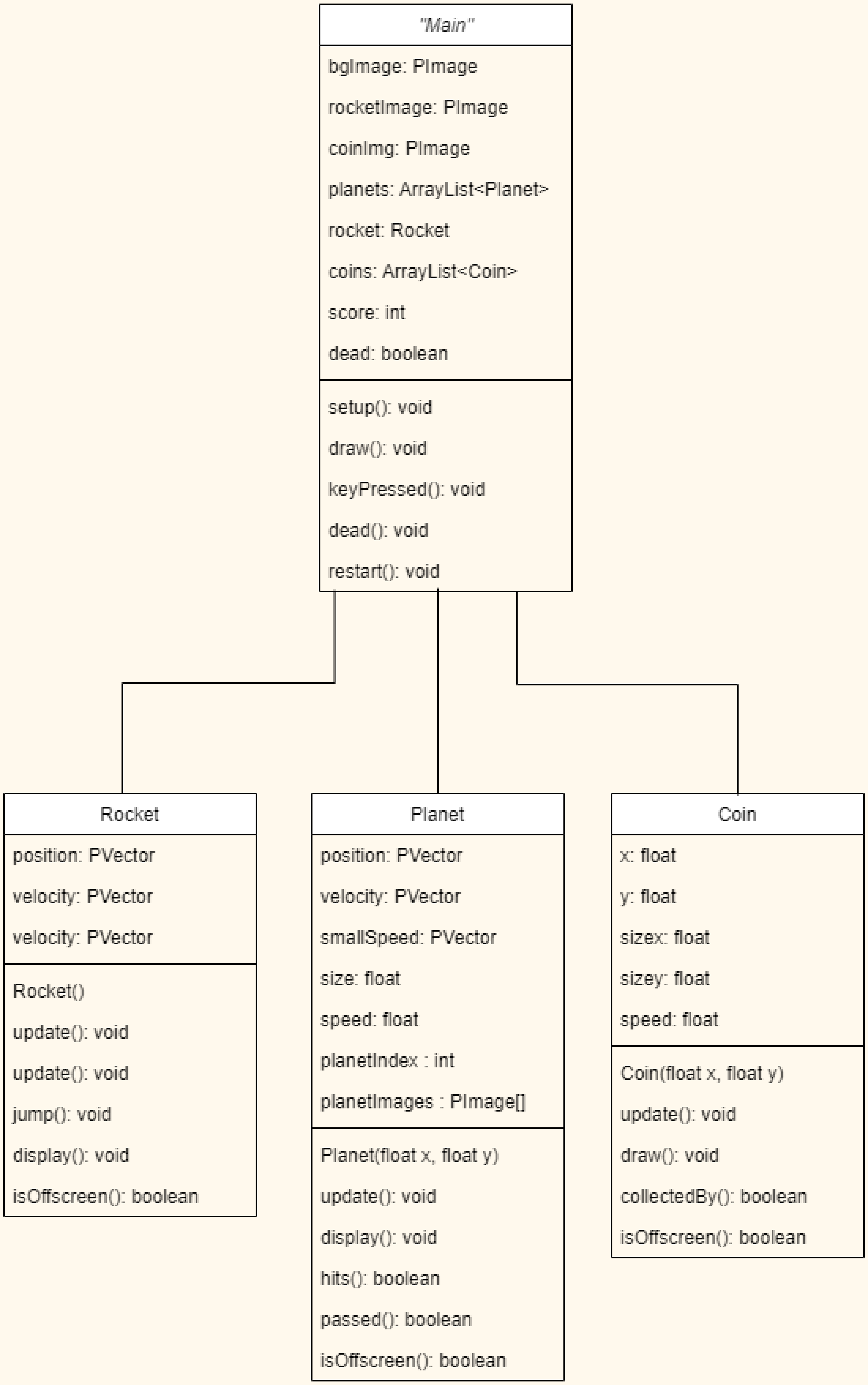
I would like to note that there are TWO versions of the game. You will find in the code, in the planet class, a commented out section that differentiates between the graphics of the planets and the simply drawn circles meant to represent the planets. You can comment out the graphics lines (which are denoted with comments) and comment the circle lines (which are also denoted by comments) to test out the game using planets that are PImages. However, I only advise you to do this if you are working on a desktop or a computer that can process high amounts of data. Otherwise, the game will lag severely due to the computer's lack of ability to process so many graphics at a time. The graphics will work on the school computer better but if you have a fast computer I recommend that you check them out as well!

How Does it Work?

The game works similarly to a flappy bird game:

1. Click on the "R" key to start the game
2. Start clicking on the space key immediately
3. The space key will allow the rocket to jump.
4. Avoid the planets and try to collect as many gold pieces as possible.
5. If the rocket goes off screen it will die and if the rocket touches a planet it will die.
6. Good luck, astronaut!

The UML will be found in the following page:


"Main"
bgImage: PImage
rocketImage: PImage
coinImg: PImage
planets: ArrayList<Planet>
rocket: Rocket
coins: ArrayList<Coin>
score: int
dead: boolean
setup(): void
draw(): void
keyPressed(): void
dead(): void
restart(): void
Rocket
position: PVector
velocity: PVector
velocity: PVector
Rocket()
update(): void
update(): void
jump(): void
display(): void
isOffscreen(): boolean
Planet
position: PVector
velocity: PVector
smallSpeed: PVector
size: float
speed: float
planetIndex : int
planetImages : PImage[]
Planet(float x, float y)
update(): void
display(): void
hits(): boolean
passed(): boolean
isOffscreen(): boolean
Coin
x: float
y: float
sizex: float
sizey: float
speed: float
Coin(float x, float y)
update(): void
draw(): void
collectedBy(): boolean
isOffscreen(): boolean