Rishi Varma –Project

1. Variables – I used variables to change speed, width, and placement of images, movement of background and more. I defined variable x\_change as 5 and x (the x –coordinate of the spaceship) as “x += x\_change” so then every time the loop runs, ‘x’ increments by x\_change to move the spaceship. The width variable defines the width of the fuel bar which is just a rectangle. But I can decrease/increase the size of the fuel bar by increasing or decreasing the width variable. Also this ‘width’ variable allows me to say that if width is 0 then game over and if it is a certain amount then says that the tank is filled. I can move the background by saying x1(x-coordinate of background img) -= 7 so that the background will move back 7 pixels each time the statement executes instead of importing a new file to the screen I can reuse that one file using variables.

If Statements-I used if statements many times to perform many simple and complex tasks. Simple one is if fuelBar’s width is 0 then ends the game. If the spaceship’s coordinates are between the asteroid’s coordinates then ends the game. Other types or elifs as well like to determine which direction to move, if sInput is \*\* elif \*\* elif \*\* and whichever number it is, that way I can move in. It can be used like a Java’s switch statement in short. A bit more complex one was ‘if x+w > mouse[0] > x and y+h > mouse[1] > y:’ which checks if the cursor is hovering over the button.

For loops- I have used for loops for one very important purpose. “for event in pygame.event.get():” which help receive and read all the events that take place like mouse movement, mouse click, key press, key release and etc. Without this loop I would be able to use mouse to select the buttons and regardless of how many times I press the red x, the window would have closed since the program would not be able to read the quit event. With for loops I was able to operate the game with a keyboard and use the mouse to click on buttons and get a response.

While loops - My program runs on while loops. Without while loops, there is no game. The main game loop ‘while not gameExit:’ continuously runs and in one execution, it reads input form the parallel port, moves the spaceship accordingly, moves the asteroid, moves the background, reads for events to know whether to quit or pause or continue and also to math work like increasing speed and displaying score and also change the fuel bar size. But running all this once is not enough. You need to run this hundreds of thousands of times on a healthy computer to get smooth graphics and animations. The while loop that performs this function is known as the main loop or the game loop. Also other functions like pause screen and intro screen require while loops because until the button is not clicked, you display the certain objects and after you click the button, you break the loop and go to next statement.

2.Buttons-First place I used buttons was as fuel bar filler. When you press the button, the current flows from wire 15 to ground and the game loop that is reading from the port gets an input of 11 or whatever and so that executes the code to fill up the tank. Another place buttons were used was to move the spaceship. There were 4 buttons and depending on which button was pressed, a certain wire’ s current passed to the ground returning a number to the program and depending on the number the spaceship would move either left right up or down.

LEDs – There were 2 LEDs used in this project. The red led lights up when the asteroid will crash into the spaceship if it does not move meaning that the spaceship is in the asteroid’s path. The yellow LED lights up when the asteroid is not in your path and the spaceship is not in the asteroid’s path signaling that you are safe for now until a another asteroid comes.

Buzzer – A single buzzer is used for 2 purposes in this game. The first is that it beeps so the user knows that his fuel is low and needs to refill it. The second way is that a long continuous beep is given off when the spaceship is in the asteroid’ s path. So along with the red light, the buzzer also goes off until the spaceship move out of the way.

5.The game was designed very well except the fact that it will only run better on computers with bigger RAMs and better CPUs. In older computers it will show a blinking spaceship and in a beefy PC it will show a solid fast spaceship. The rule of the game is that no cheating. Also other rules are that you need to tilt your controller in the direction you want it to move in and avoid contact with the asteroids. Also, you don’t want to run out of fuel or else you will lose. Another rule is that you can’t put too much fuel or the game slows down. So when the sign reads fuel filled, release the blue button. There are 2 main objectives in the game, stay out of contact with the asteroids and don’t run out of fuel and of course there is the original objective which is to get the highest score possible. Some features are that this is a the future so the spaceship and asteroid both have a semi shield on them so if they come in contact then you lose. Also the spaceship is very techy so if you go too low then no worried because it will teleport to the top and vice versa. Unfortunately the spaceship loves fuel so it eats lots of fuel therefore so refilling is required often. The game can be paused by the ‘p’ key. The game displays the score right at the top so it is easy to track your score. Also, the speed of the asteroid increments by one every time the score reaches a multiple of 5 and when the score gets to 50, its speed will increase rapidly. When your fuel is low the game slows down and tells you that the fuel is low. Also when the fuel tank is full, the game slows down until you stop filling the fuel and use some of it and also displays the message saying that the tank is full. It has buttons to select play, resume, or exit.