Data Dictionary

# Main Loop (mainloop())

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Item | Data Type | Format | Description | Example | Validation |
| White, black, red, yellow, green, blue | triple integer | (000, 000, 000) | An RGB value represent in red green blue | (255, 255, 255) | All numbers between 0 and 235 |
| Fly | Sprite Class | Sprite(x, y, vel, width, height, screen, shape, color) | An instance of the sprite class | Sprite(555, 315, 25, 80, 80, screen, ‘rect’, blue) | All values are correct type. x,y,vel,width,height = integer. Shape = string and color = rgb value and screen = pygame.display class |
| Gravity | gravity\_sim function call | gravity\_sim(desired-planet) | Determines the gravity felt by objects within the game. | gravity\_sim(‘earth’) | inputs a valid planet into the parameters as a string |
| vertAcc | float | NN.NN | vertical acceleration of player character | -91.8 |  |
| horAcc | float | NN.NN | horizontal acceleration of player character | 45.6 |  |
| key | integer | NN | keeps track of how many enemies have spawned | 3 | positive number |
| enemies | list | [Sprite, Sprite, …] | stores all the enemy sprites that have been created | [badGuy, badGuy, badGuy, badGuy,…] | all values are of class Sprite() |
| gr | GameRun class | GameRun(screen) | initiates game run as gr | GameRun(screen) |  |
| clock | pygame.time.clock class | pygame.time.Clock() | creates clock to limit frames per second | pygame.time.Clock() |  |
| loopRate | integer | NN | sets max speed of main loop | 60 | positive number |
| play | Boolean | True, False | determines whether to keep looping or not | True |  |
| flyXRange, bgXRange | array of floats | [NNN.NN, NNN.NN, …] | stores all the x values the character exists at | [173.0,174.0,175.0,176.0,177.0] | numbers are in ascending order |
| flyYRange, bgYRange | array of floats | [NNN.NN, NNN.NN, …] | stores all the y values the character exists at | [173.0,174.0,175.0,176.0,177.0] | numbers are in ascending order |
| playerScore | integer | NN | Stores players score as a number | 13 | positive number |
| spawn | boolean | 0 or 1 | returns 0 or 1 depending on if an enemy should spawn | 1 |  |
| bgX | integer | NNN | initial x value badGuy will receive | 74 | value between 0 and SCREENWIDTH |
| bgY | integer | NNN | initial y value badGuy will receive | -20 | value between 0 and -50 |
| bgWidth | integer | NNN | width badGuy will receive | 45 | value between 10 and 100 |
| bgHeight | integer | NNN | height bad guy will receive | 190 | value between 10 and 100 |
| badGuy | instance of Sprite class | Sprite(x, y, vel, width, height, screen, shape, color) | sets up badGuy as a sprite | Sprite(bgX, bgY, 5, bgWidth, bgHeight, screen, ‘rect’, (0,0,0)) | all parameters have appropriate inputs |
| collision | Boolean | True or False | checks if two objects are colliding and stores True if they are and False if they aren’t | True |  |
| direction | Boolean | none or -1 | stores whether to move the objects left or right, if it is -1 move left, 0 move right | -1 |  |
| currentScore | string | XX | playerScore as a string | “11” |  |
| playScore | iteration of render class | font.render(text\_to\_be\_displayed, anti aliasing t or f, color) | text ready to be blitted to screen | scoreFont.render(currentScore, True, (0,0,0)) | parameters are of correct format. Font = recognisable font, text\_to\_be\_displayed = string, anti aliasing = True or False, color = rgb value |
| textBox | pygame.rect | renderedTextToBeDisplayed.get\_rect() | rectangle the right size for text to fit in | playScore.get\_rect() | value it is called with is rendered text, see value above. |