



../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.35.25%

The ape, Donkey Kong, has kidnapped Mario’s girlfriend, Pauline, and is holding her captive on a construction site. The user will be playing as Mario who is trying to climb the structure in order to rescue Pauline. Be careful though because Donkey Kong will be trying to stop you by rolling barrels to hit you. Mario will need to climb to the highest platform where Pauline is kept in order to rescue her. You get 100 points for jumping over a barrel, and 250 points for saving Pauline. If you rescue her, Donkey Kong will kidnap her again, and he will be rolling more barrels at you. Rescue her five times and you will defeat Donkey Kong because he will be so exhausted from climbing and throwing barrels. You can also win the game if you get over 999999 points because his mind will literally be blown your barrel jumping skills! You only have three lives, so good luck!

../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.35.32%

DURING GAME PLAY:

LEFT ARROW Mario moves left

RIGHT ARROW Mario moves right

UP ARROW Mario moves up ladders

DOWN ARROW Mario moves down ladders

SPACE BAR Mario jumps

GAME OVER/YOU WIN MENU:

UP ARROW Selects top option

DOWN ARROW Selects bottom option

RETURN KEY Enters your option response

ANYTIME:

ESCAPE KEY Quit the game interface

../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.35.39%

The player’s score and the high score will be shown on the interface at the top of the screen. The score under “1UP” is the player’s score and the score under “HIGH SCORE” is the high score.

All the players and their scores will also be outputted on the console in rank order after you quit the game.

../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.35.46%

INPUT:

* Asks for input to confirm you have read the instructions
* User’s name so that the score has a name associated to it
* Keyboard keys are inputted to see which way Mario goes and which option is selected in the menus

OUTPUT:

* Images are outputted on to the pygame interface
* The ranks of all the players and their scores are outputted on the console at the end of the game

CALCULATIONS:

* Add points to your score
* Add/subtract to the x coordinate and/or y coordinate every time Mario, the barrels, confetti, or Donkey Kong moves
* Add/subtract Mario and the barrels’ y coordinates so that he moves on an angle with the beam platform
* Keep track of how many lives Mario has left, minus one every time he gets hit
* Increase difficulty of the game by making the range of numbers smaller, making it more likely for Donkey Kong to throw more barrels
* Make each number in the score the same distance from each other
* Add/minus images’ x and y coordinates to see if it hit a boundary, platform, barrel, ladder
* Finds out the difference in digits between the score and the score placeholder, which can hold 6 digits
* Each time you output a name, score, and rank on the console, add one to the rank to find the rank of the next person

IF STATEMENTS:

* Checks if Mario or a barrel has hit a boundary, ladder
* Checks if Mario’s x values and y values hit a barrel’s x values and y values
* Checks if a stage in the game has happened yet
* If a number is the same as a score’s digit
* If the score is the same as another person’s score
* Checks which image is being outputted and/or where it is being outputted and/or what the image is doing
* Checks when you win the game/level by looking at the level number, score and x value of Mario
* Checks when a keyboard key is pressed

LOOPS:

* Keeps asking user for name in while loop if the name has already been used
* While the game hasn’t been quitted, keep running the game on the screen
* For loop to go through lists to check the values with the number 0 to 9 for the scores
* For loop used to print all the names, ranks, and scores
* For loop to go through all the names and scores to see which score is associated to which name
* For loop to output multiple of the image, like barrels
* For loop to change values for the all barrels or confetti
* For loop to append all the confetti values
* For loop to go through all the boundaries/ladders/inclines with the coordinates of Mario and the barrels
* For loop to check all the barrels with the boundaries/ladders/inclines

FUNCTIONS:

* Used to output images/circles onto the screen
* Incline, collisions, and boundaries for Mario and the barrels to adjust and check coordinates
* Outputs the ranks, names, and scores on the console
* Finds the high score and name of user

LISTS:

* Store boundaries for Mario and the barrels
* Store the x values for where the platform’s beam inclines
* Stores the range of x values, top and bottom y value, if you can go up all the way, if you can go down all the way for the ladders where Mario can move up and down
* Stores x values and top and bottom y value of the ladder for where the barrels can go down a ladders
* Store all x values, y values, image, direction, if they are falling, how long they have fallen for, if it can move left and if it can move right for the barrels
* Store the image sequence for the falling platforms, white numbers, blue numbers, and rolling barrels
* Store the scores, and the names when the dictionary is separated to sort the scores and then to find the associated name
* Stores the confetti’s x value, y values, radius, falling speed, and colour

DICTIONARIES:

* Keeps the name and the associated score together

PYGAME:

* Use pygame to output all of the images on to the screen
* The screen filled black
* Time delays were used to stop the program for a few seconds so that the user could see what was being displayed for longer
* Time delays were also used to make the animations for Mario and Donkey Kong climbing smoother because if the amount of time between each redraw screen was too short, the switching between the two images to make the effect that they were moving would be ruined
* Draws multiple circles of different sizes, at different x and y coordinates for confetti

../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.35.55%

Title screen image found here:

<http://www.arcade-museum.com/game_detail.php?game_id=7610>

All intro backgrounds images and Donkey Kong climbing without Pauline found here:

<http://www.spriters-resource.com/arcade/dk/sheet/46025/>

Start screen and all the numbers found here:

<http://www.spriters-resource.com/arcade/dk/sheet/46052/>

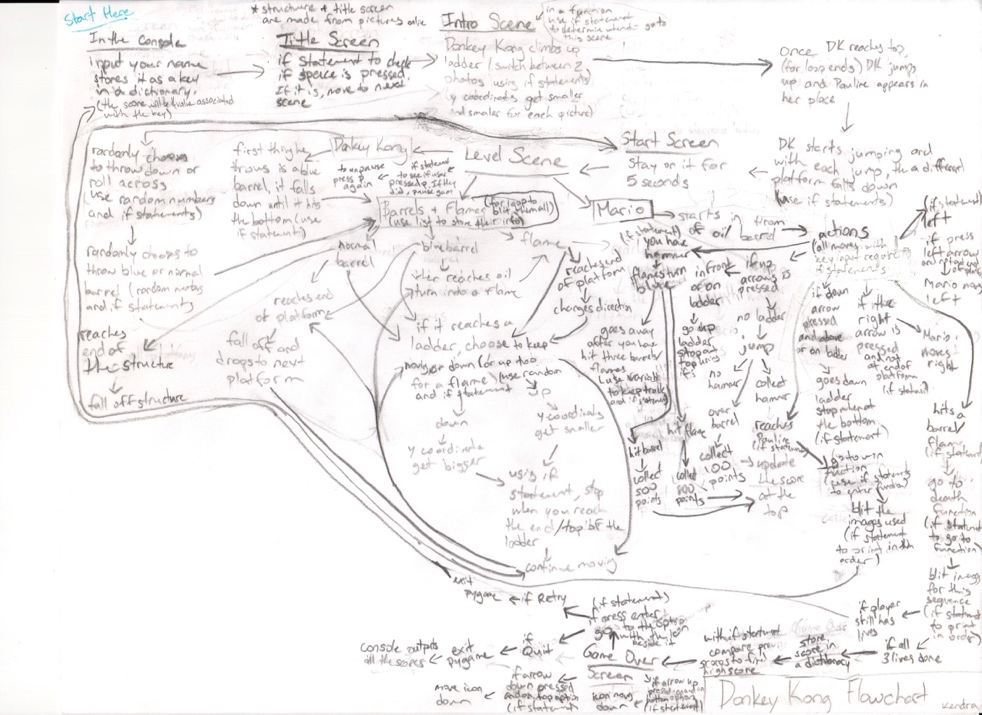
Level image, logo on user’s manual, and all Donkey Kong (except for him climbing without Pauline), Mario, Pauline, barrel, heart, icon images found here:

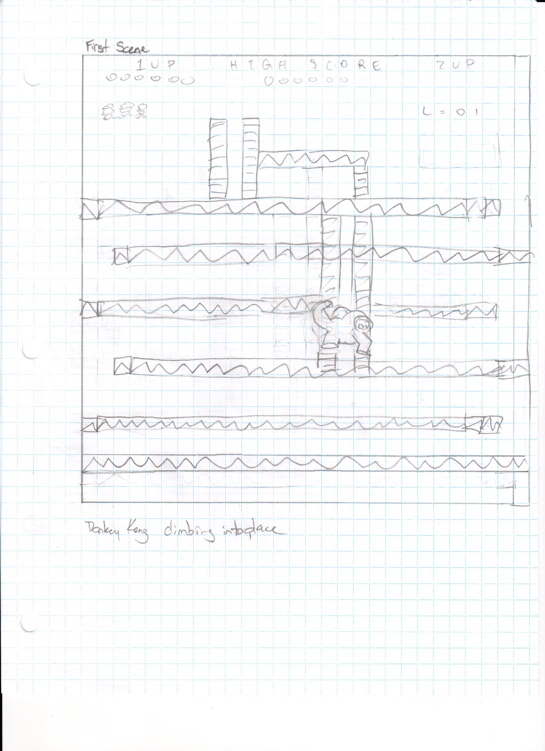
<http://www.classicgaming.cc/classics/donkey-kong/graphics>

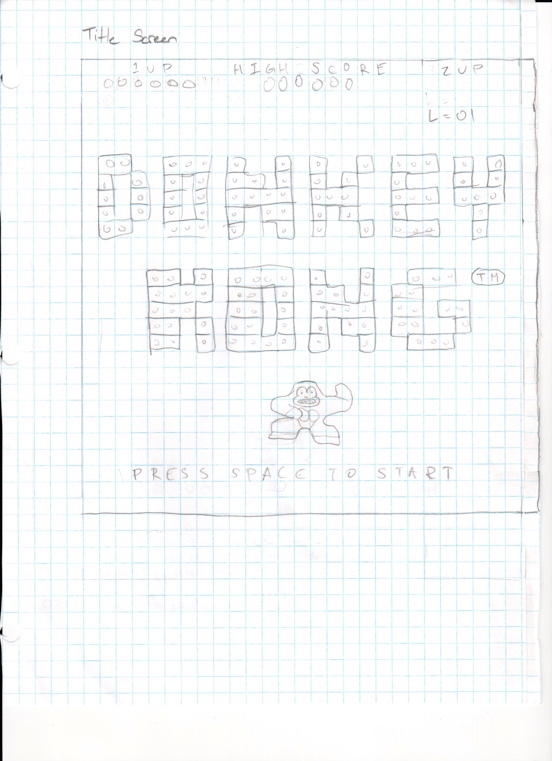
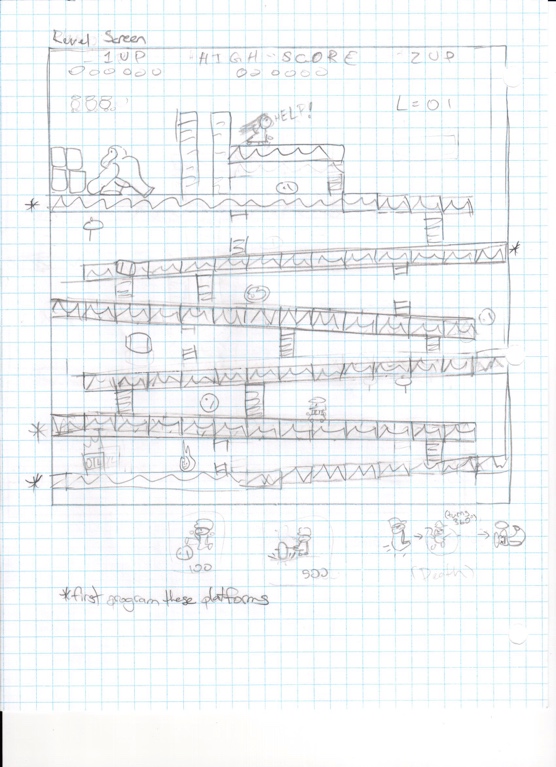
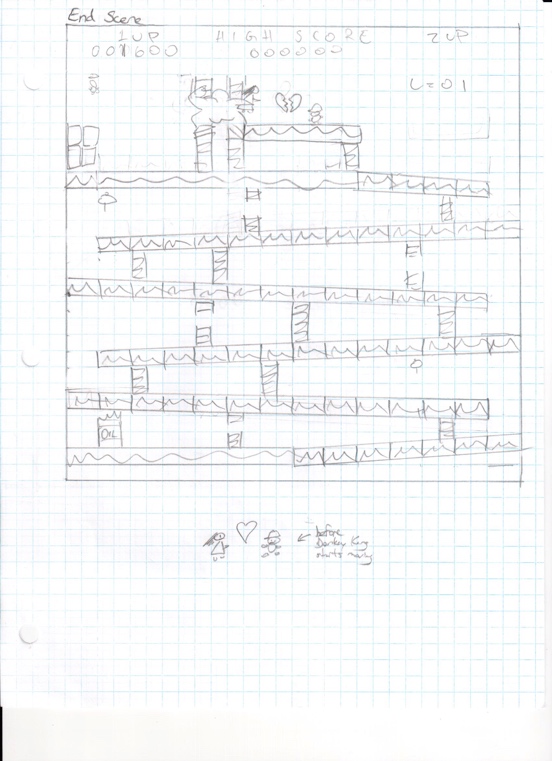
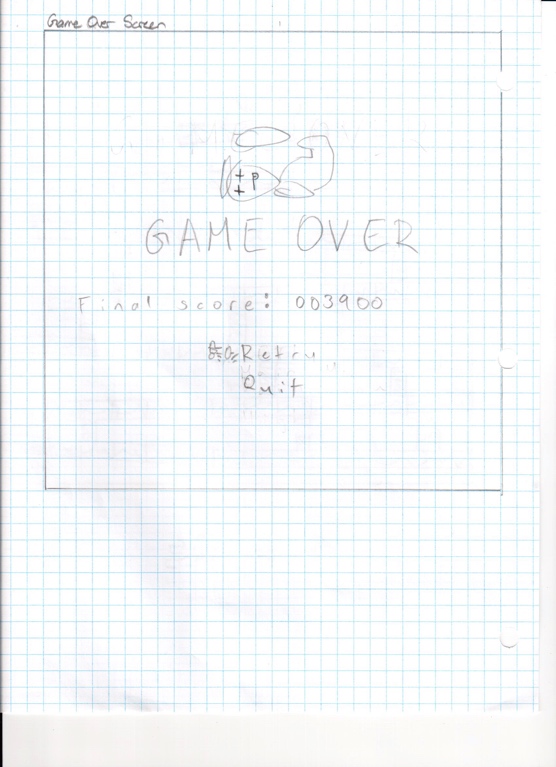
Win and end screen image found here:

<http://www.classicgaming.cc/classics/donkey-kong/play-donkey-kong>

../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.36.00%



../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.36.08%



../../../../Desktop/Screen%20Shot%202016-05-26%20at%205.36.14%

|  |  |  |
| --- | --- | --- |
| Functions | Expected | Actual |
| instructions | Prints the instructions in the console. Hit return key to continue program. | Prints the instructions in the console. Hit return key to continue program. |
| getName | Asks for your name. If the name has already been used, a while loop will keep looping to ask you for another name.  Then, return the name. | Asks for your name. If the name has already been used, a while loop will keep looping to ask you for another name.  Then, return the name. |
| highScore | Adds your name and score on to the leaderboard dictionary. Separates the score values and then sorts them form least to greatest. Finds the highest score by finding score at the last index in the scores list. | Adds your name and score on to the leaderboard dictionary. Separates the score values and then sorts them form least to greatest. Finds the highest score by finding score at the last index in the scores list. |
| outputLeaderboard | Separates the keys and values in the dictionary. Use a for loop to check all the score, and a nested for loop inside to check all the names. If the name has not already been appended to the list of sorted names, check if the score matches the value at the leaderboard with the key of the name the for loop is on. If the score does match, append the name into a sorted names list. Print LEADERBOARD and a row of stars on the line below that. Then using a for loop that starts the last index of the sorted scores list and goes to index 0, use an if statement to check if i is greater than the length of leaderboard minus one. If it is True, have another if statement inside to check if scores[i] is the same as scores[i+1]. If it is, add one to the rank and print out the rank, “|”, names[i], “:”, scores[i]. At the end, print the message, “THANKS FOR PLAYING! :)” | Separates the keys and values in the dictionary. Use a for loop to check all the score, and a nested for loop inside to check all the names. If the name has not already been appended to the list of sorted names, check if the score matches the value at the leaderboard with the key of the name the for loop is on. If the score does match, append the name into a sorted names list. Print LEADERBOARD and a row of stars on the line below that. Then using a for loop that starts the last index of the sorted scores list and goes to index 0, use an if statement to check if i is greater than the length of leaderboard minus one. If it is True, have another if statement inside to check if scores[i] is the same as scores[i+1]. If it is, add one to the rank and print out the rank, “|”, names[i], “:”, scores[i]. At the end, print the message, “THANKS FOR PLAYING! :)” |
| collide | A for loop goes through every barrel’s x and y coordinates and checks if the range of the barrel’s x and y values is in the range of Mario’s x and y values. If it is, then return hit as True. | A for loop goes through every barrel’s x and y coordinates and checks if the range of the barrel’s x and y values is in the range of Mario’s x and y values. If it is, then return hit as True. |
| ladderCheck | Set upLadder, downLadder, and moveSides as False, False, and True, respectively. Use a for loop to go through all the ladder coordinates. If Mario is within range to climb a ladder, set downLadder, upLadder, and moveSides to True, True, False, respectively. Other if statements are nested in to check if mario’s Y coordinate is at the top of the ladder or the bottom of the ladder. If marioY equals to ladderY1[i], then Mario is at the top of the ladder and upLadder is False. Nested inside that if is another if that checks if fullLadderUp[i] is True. If it is, then move Sides is True. Similar process happens when marioY equals to ladderY[2], which means he is at the bottom of the ladder, except this time, downLadder will be set to False, and if fullLadderDown[i] is True. moveSides will also be True.  After checking if Mario is within the range of the ladder, if upLadder or downLadder is True, break out of the loop. At the end of the program return upLadder, downLadder, moveSides. | Set upLadder, downLadder, and moveSides as False, False, and True, respectively. Use a for loop to go through all the ladder coordinates. If Mario is within range to climb a ladder, set downLadder, upLadder, and moveSides to True, True, False, respectively. Other if statements are nested in to check if mario’s Y coordinate is at the top of the ladder or the bottom of the ladder. If marioY equals to ladderY1[i], then Mario is at the top of the ladder and upLadder is False. Nested inside that if is another if that checks if fullLadderUp[i] is True. If it is, then move Sides is True. Similar process happens when marioY equals to ladderY[2], except this time, downLadder will be set to False, and if fullLadderDown[i] is True, then moveSides is also True.  After checking if Mario is within the range of the ladder, if upLadder or downLadder is True, break out of the loop. At the end of the program return upLadder, downLadder, moveSides. |
| incline | Get y, x, direction, and objectt as paramters. If y is between 720 and 657, then startNum is 6, endNum is len(platInclineX)-1, and move is 3. If y is between 638 and 553 or between 353 and 438, then startNum is 0, endNum is len(platInclineX)-2, and move is -3. If y is between 541 and 456 or between 341 and 256, then startNum is 1, endNum is len(platInclineX)-1, and move is 3. If y is between 245 and 149, then startNum is 8, endNum is len(platInclineX)-2, and move is -3. Otherwise, startNum is 0, endNum is 0, and move is 3. Then in a for loop that has the range of startNum to endNum, check if x is equal to platInclineX[i]. If it is and jumpLeft or jumpRight is True and objectt is “mario”, then increase inclineCount up by one. Otherwise, check if the direction is “right”, then y=y-move, if direction is “left”, then y=y+move. When the loop is done running, check if jumpLeft or jumpRight is True and if objectt is “mario”. If it is all True, return move, otherwise, return y. | Get y, x, direction, and objectt as paramters. If y is between 720 and 657, then startNum is 6, endNum is len(platInclineX)-1, and move is 3. If y is between 638 and 553 or between 353 and 438, then startNum is 0, endNum is len(platInclineX)-2, and move is -3. If y is between 541 and 456 or between 341 and 256, then startNum is 1, endNum is len(platInclineX)-1, and move is 3. If y is between 245 and 149, then startNum is 8, endNum is len(platInclineX)-2, and move is -3. Otherwise, startNum is 0, endNum is 0, and move is 3. Then in a for loop that has the range of startNum to endNum, check if x is equal to platInclineX[i]. If it is and jumpLeft or jumpRight is True and objectt is “mario”, then increase inclineCount up by one. Otherwise, check if the direction is “right”, then y=y-move, if direction is “left”, then y=y+move. When the loop is done running, check if jumpLeft or jumpRight is True and if objectt is “mario”. If it is all True, return move, otherwise, return y. |
| boundaries | Get the x and y variables from the parameter. Set both left and right equal to True. If x is between 96 and 105, inclusive, then go through a for loop of all the values in leftBoundariesY. If y is between leftBoundaires[i] and leftBoundaries[i]-49, then left is false. Then if x wasn’t in the range before, check if x is between 660 and 669. If it is, a for loop will go through all of the rightBoundairesY values. If y is between rightBoundariesY[i] and rightBoundariesY[i]-49, then right is false. At the end of the function, return left and right. | Get the x and y variables from the parameter. Set both left and right equal to True. If x is between 96 and 105, inclusive, then go through a for loop of all the values in leftBoundariesY. If y is between leftBoundaires[i] and leftBoundaries[i]-49, then left is false. Then if x wasn’t in the range before, check if x is between 660 and 669. If it is, a for loop will go through all of the rightBoundairesY values. If y is between rightBoundariesY[i] and rightBoundariesY[i]-49, then right is false. At the end of the function, return left and right. |
| introScene | If dkClimb is less than or equal to 390, output on the screen, the withLadder image at (48, 0). If statement nested inside will then check if dkClimb modulus 30 is 0, output the Donkey Kong image dkUp2 at (350, 660-dkClimb). Otherwise, dkUp2 at (370, 660-dkClimb). If dkClimb is between 390 and 580, including 580, output platform0 image at (55, 9) and dkUp2 at (350, 660-dkClimb). If climbDone is true, then output platforms[platNum] at the coordinates (platformsX[platNum], platformsY[platNum]) and output on the screen, dkForward at the coordinates (dkJumpX, dkJumpY). Also call pauline function and give the argument paulineStill. | If dkClimb is less than or equal to 390, output on the screen, the withLadder image at (48, 0). If statement nested inside will then check if dkClimb modulus 30 is 0, output the Donkey Kong image dkUp2 at (350, 660-dkClimb). Otherwise, dkUp2 at (370, 660-dkClimb). If dkClimb is between 390 and 580, including 580, output platform0 image at (55, 9) and dkUp2 at (350, 660-dkClimb). If climbDone is true, then output platforms[platNum] at the coordinates (platformsX[platNum], platformsY[platNum]) and output on the screen, dkForward at the coordinates (dkJumpX, dkJumpY). Also call pauline function and give the arguement paulineStill. |
| startScreen | Outputs the “How high can you get?” image on the screen at (48, 0). | Outputs the “How high can you get?” image on the screen at (48, 0). |
| background | Outputs the level background at (31, -14) and the barrel stack at (60, 188) on the screen. | Outputs the level background at (31, -14) and the barrel stack at (60, 188) on the screen. |
| dk | Outputs a Donkey Kong image at (130, 176) on the screen. | Outputs a Donkey Kong image at (130, 176) on the screen. |
| mario | Outputs a Mario image at the coordinates of two global variables, marioX and marioY, on the screen. | Outputs a Mario image at the coordinates of two global variables, marioX and marioY, on the screen. |
| pauline | Outputs a Pauline image given by the parameter at (335, 133) on the screen | Outputs a Pauline image given by the parameter at (335, 133) on the screen |
| barrel | A for loop outputs all the barrels on the screen. Each barrel has a different image located at a x and y coordinates which are stored in lists. The range of the for loop depends on how many values are int. | A for loop outputs all the barrels on the screen. Each barrel has a different image located at a x and y coordinates which are stored in lists. The range of the for loop depends on how many barrels there are. |
| marioLives | A for loop outputs all the life images. The range is from 0 to the value stored in the variable lives. Each life image will be 20 pixels away from the other life image. | A for loop outputs all the life images. The range is from 0 to the value stored in the variable lives. Each life image will be 20 pixels away from the other life image. |
| levelNumber | A for loop will go through all the numbers from 0 to the length of the list blueNumbers. If levelNum divided by 10 equals to the number the for loop is on (i.e. i), output the image blueNumbers[i] at the coordinates (611, 86). If levelNum%10 is equal to I then output the image blueNumbers[i] at (635, 86). | A for loop will go through all the numbers from 0 to the length of the list blueNumbers. If levelNum divided by 10 equals to the number the for loop is on (i.e. i), output the image blueNumbers[i] at the coordinates (611, 86). If levelNum%10 is equal to I then output the image blueNumbers[i] at (635, 86). |
| playersScores | Make a new temporary variable called tempScore to hold the player’s score or high score as a string, depending on which score was given in the parameter. scoreX and scoreY are also parameters. To find the number of zeros needed to be printed in front of the score, do 6 minus the length of the string. Use a for loop to output all the zeros, adding 24 to the x value the first number is at each time. The y value is always the same. Then in a nested for loop, go through each character in tempScore, and go through the digits 0 to 9. When the the tempScore[i] character is converted into an integer and it is equal to the digit the nested for loop is on, j, output the whiteNumbers[j[ to the screen and then add 24 to score X. | Make a new temporary variable called tempScore to hold the player’s score or high score as a string, depending on which score was given in the parameter. scoreX and scoreY are also parameters. To find the number of zeros needed to be printed in front of the score, do 6 minus the length of the string. Use a for loop to output all the zeros, adding 24 to the x value the first number is at each time. The y value is always the same. Then in a nested for loop, go through each character in tempScore, and go through the digits 0 to 9. When the the tempScore[i] character is converted into an integer and it is equal to the digit the nested for loop is on, j, output the whiteNumbers[j[ to the screen and then add 24 to score X. |
| win | Output the background and Mario looking left at the coordinates (440, 150) on the screen. If dkClimb variable is less than or equal to 30, output Pauline not saying help with the pauline function, and a full heart. Else, output a broken heart. Then, if winGame is false, and dkClimb modulus 30 is zero, output the first climbing image of Donkey Kong, with an x of 240 minus a variable that corrects the x so that Donkey Kong climbs up smoothly, and a y value that is 160 minus dkClimb)  If dkClimb modulus 30 is not zero, do the same thing in the above, except output a different Donkey Kong image. If winGame was True, just call the dk function. | Output the background and Mario looking left at the coordinates (440, 150) on the screen. If dkClimb variable is less than 30, output Pauline not saying help with the pauline function, and a full heart. Else, output a broken heart. Then, if winGame is false, and dkClimb modulus 30 is zero, output the first climbing image of Donkey Kong, with an x of 240 minus a variable that corrects the x so that Donkey Kong climbs up smoothly, and a y value that is 160 minus dkClimb)  If dkClimb modulus 30 is not zero, do the same thing in the above, except output a different Donkey Kong image. If winGame was True, just call the dk function. |
| end | Outputs the end screen, which is different depending on which image was given in the parameter. If the option variable is equal to “bottom”, the select icon will be on the bottom choice at the coordinates (270, 640). Otherwise, the icon will be outputted beside the top choice at the coordinates (270, 640). | Outputs the end screen, which is different depending on which image was given in the parameter. If the option variable is equal to “bottom”, the select icon will be on the bottom choice at the coordinates (270, 640). Otherwise, the icon will be outputted beside the top choice at the coordinates (270, 640). |
| confetti | Draws 200 circles on the screen, accessing the x and y coordinates, radius, and colour of each circle in 5 different lists. (confettiX, confettiY, confettiRadius, confettiSpeed, confettiColour) | Draws 200 circles on the screen, accessing the x and y coordinates, radius, and colour of each circle in 5 different lists. (confettiX, confettiY, confettiRadius, confettiSpeed, confettiColour) |
| redrawScreen | Fills the screen black. Check if gameDone is True. If it is, call end function with the argument gameOverScreen, playerScores function with the arguments score, 388, and 387, and playerScore fucntion again only this time with the arguments highestScore, 485, and 445. If gameDone was False, check if winGame is True. If it is, check if winGameSceneOutput is True. If it is call the win function, marioLives function, levelNumber function, and the playersScore function twice (the first one with the agruements score, 388, and 387, and the second one with the arguments highestScore, 485, and 445). If winGameSceneOutput is false, then, check if winGameSceneDone is True. If it is True, then call the end function with the argument winScreen, the confetti function, the playersScore function with the arguments score, 388, and 387, and the playersScore function with the arguments highestScore, 485, and 445. If winGame is False and gameDone is False, then first check if pressed is False. If it is then output the title screen at (54, 18). If pressed is True, check if introDone is False too. If both are satisfied, then call the introScene function and the marioLives function. If those two conditions aren’t both satisfied, then check if introDone is True and if gameStart is False. If they are both satisfied, call the startScreen and marioLives function. Also, set both startOutput and startDone to True. If those two aren’t both satisfied, check if gameStart is True and if winLevel is False. If they are satisfied, call the background, dk, mario, pauline with the argument paulineHelp, and marioLives function. Then check if scoreWin is False because if it is, then also call the barrel function. If both gameStart is True and winLevel is False are not satisfied, check if deathScene is True. If it is, call background, mario, dk, pauline with the argument paulineHelp, and marioLives function. If deathScene is also False, check if winLevel is True. If it is, then call the win and marioLives function.  Then, out of this string of ifs, call the levelNumber function, and the playersScores function twice (the first one with the arguments score, 88, and 40, and the second one with the arguments highestScore, 327, and 40). At the end of the whole function, and not in any if statements, this function updates the pygame display. | Fills the screen black. Check if gameDone is True. If it is, call end function with the argument gameOverScreen, playerScores function with the arguments score, 388, and 387, and playerScore fucntion again only this time with the arguments highestScore, 485, and 445. If gameDone was False, check if winGame is True. If it is, check if winGameSceneOutput is True. If it is call the win function, marioLives function, levelNumber function, and the playersScore function twice (the first one with the agruements score, 388, and 387, and the second one with the arguments highestScore, 485, and 445). If winGameSceneOutput is false, then, check if winGameSceneDone is True. If it is True, then call the end function with the argument winScreen, the confetti function, the playersScore function with the arguments score, 388, and 387, and the playersScore function with the arguments highestScore, 485, and 445. If winGame is False and gameDone is False, then first check if pressed is False. If it is then output the title screen at (54, 18). If pressed is True, check if introDone is False too. If both are satisfied, then call the introScene function and the marioLives function. If those two conditions aren’t both satisfied, then check if introDone is True and if gameStart is False. If they are both satisfied, call the startScreen and marioLives function. Also, set both startOutput and startDone to True. If those two aren’t both satisfied, check if gameStart is True and if winLevel is False. If they are satisfied, call the background, dk, mario, pauline with the argument paulineHelp, and marioLives function. Then check if scoreWin is False because if it is, then also call the barrel function. If both gameStart is True and winLevel is False are not satisfied, check if deathScene is True. If it is, call background, mario, dk, pauline with the argument paulineHelp, and marioLives function. If deathScene is also False, check if winLevel is True. If it is, then call the win and marioLives function.  Then, out of this string of ifs, call the levelNumber function, and the playersScores function twice (the first one with the arguments score, 88, and 40, and the second one with the arguments highestScore, 327, and 40). At the end of the whole function, and not in any if statements, this function updates the pygame display. |