Ishika Prashar SID:3034618276

Shivani Sharma SID:3035293379

README : Snowfall

Introduction: Our game, Snowfall, is designed around relieving stress to students during the weeks before and after final season by creating a game in which the user must catch the falling snow in accordance to the welcoming of winter and the approaching holiday festivities.

Features

1. Our game snowfall has various types of snow with distinct colors, each color representing a certain value. The “red snow” is bad and takes away one of your lives. There is also a “navy” snow which is rare and adds a life and 5 points to score. The rest of the regular snow each add one point to your score if caught. This feature was implemented utilizing lists for the different snow types which was randomized to fall down from the tree at random speeds.
2. The purpose of Snowfall is to catch the good snow in your basket. However, if the bad/red colored snow is caught in your basket, you lose a life and score points. To make this basket we used python’s turtle and made two custom blocks to define the constraints of how far left or right the turtle can move. Then, we implemented this turtle to move with left or right arrow keys. To check if snowball goes into the basket, distance was used.

Description of lists and local variables in our program

We created a function, snowball_types(), found in line 94 of the code file right after draw_board implementation, which contains a local variable list of the different color of normal snowballs in our game and by importing the random module, we shuffled the list and returned its first color in order to enhance the unpredictability of which color snowballs will drop from the tree. In order to use this function, we called for it in our for-loop for the normal snowballs as we called for snowball_types() in the snowball.color argument. By calling for snowball_types() in the nested for-loop, it causes the local variable to be called for more than once through its specified range.

We also implemented lists for each of the different kinds of snowballs in our game as we set each of the snow types to an empty list. Afterwards, we defined the characteristics of one of the snow types within each of the lists and append them to the empty list of snow types through the range. Score and lives are also global variables used in implementing point system for all colored snowballs. They are found right after the basket left right implementation in line 190 and 191. These global variables are used inside for statements for snowballs to add one to score everytime a good snowball is caught. They are also used for bad snowballs to decrease life by one and score by five. Same goes for special snowball but this snowball adds a life and adds ten to score. Score and lives global variable is also used by turtle to write out the updated score and life on screen. X coordinate positions for basket left right functions were also local as each coordinate looked at a different position in the different functions. These can be found from lines 165 to 180.

Reference Table

Block/function name	Domain	Range	Behavior
basket_right	n/a [created to use with .onkey to get basket to move right only up to a certain limit]	n/a (sets x values)	Used to move the turtle to the right side of the screen without having the turtle go off the screen.
basket_left	n/a [created to use with .onkey to get basket to move left only up to a certain limit]	n/a (sets x values)	Used to move the turtle to the left side of the screen without having the turtle go off the screen.

snowball_types	n/a (just created to help randomize the snow colors as they fall)	String	Takes in a list of snowball color types and returns a random color which is later used to drop a random colored snowball from the screen.
recursive_tree	Numbers	n/a (draws using turtle)	Recursive tree placed as background enhancement and used to give a place for the snow to begin falling from
Timer	number	n/a(turtle draws and writes instructions on screen)	Timer function that takes in amount of seconds starting screen should stay on the screen. It commands turtle to change screen color, and writes the details of the game on the screen so the player knows what to do.

-We used Python version 3.7.3 to code our project.
-We imported turtle, random, and time to our program
-To begin the game, please type “python3 snowfall_shivani_ishika.py” into terminal
-wait for turtle to finish drawing tree, border, etc and you know game begins once the snowballs start falling
-Use left and right arrow keys to move the basket and catch snowballs
-Also, please expand window screen fully to enjoy the full game experience
-The snowball start in a line but once you begin catching those they become randomized over the tree branches
-Do not catch the red snowball, they are bad
-The dark navy snowball adds an extra life and plus ten score points but are harder to catch and come less often as well

Side note: This project ended up being a lot more complex than we could have ever expected. Coding in Python for the first time was very difficult. We had a third feature in our proposal which was to use a board in order to find coordinates of tree branches and have snowballs fall from there. We were able to fully implement the board’s list and turtle drawing. The border you see in the game is actually a board with one row and one column. However, implementing this to work was way too out of scope for us. We used different functions for turtle like .position() to set list elements as coordinates of the tree. Then, have the snowball start at index of certain coordinate in board list. When we couldn’t implement this, we even tried filling the board with a color each time the user caught the red/bad snowball. This way, after the user caught a certain number of bad snowballs, they would no longer be able to play as the screen would completely turn black. This also backfired as the turtle had a difficult time filling in only certain parts of the board based on the if condition that bad snowball was caught. All in all, we wanted to let you know that we tried our best to be complex and use the board to the best of our ability, but weren’t able to do so. We also did go to office hours and asked for help on Piazza but this was just way too difficult for us to understand. However, we included all the board code we were able to implement incase you wanted to take a look, or we later figure out a way to implement it.

Video Link: https://youtu.be/ml0ZtDCybps