Prevent Ozone Depletion – Part I

Hi Everyone!

Hope you are enjoying your role as a Climate Action Change Agent and becoming better at developing games. It is now time for us to get started with yet another mission. Let me give you a brief background.

You may have heard about Ozone, a gas that forms a thin protective layer around the Earth. It shields our planet from harmful solar radiation or ultraviolet rays by absorbing almost 98% of them before they hit the earth. But the ozone layer in the earth's atmosphere is getting thinner over time. Chemicals called chlorofluorocarbons or CFCs which are released on earth are leading to the depletion of the ozone layer. Chlorofluorocarbons are made of elements like carbon, fluorine, and chlorine. A single CFC molecule can destroy over 1,00,000 ozone molecules and unfortunately, ozone can be destroyed more quickly than it can be naturally created. The damaged areas in the ozone layer are generally referred to as ozone holes.

Our mission is to create a game in Scratch to save the ozone layer from getting depleted as a result of the harmful gases released by man-made things like vehicles and industries. The goal is to pop the toxic emissions before they hit the ozone layer. Remember, as these gases continue to reach the ozone layer, the colour of the ozone layer changes gradually and eventually when it becomes red, the game gets over.

In order to create the game we will work on the following four tasks –

a) Designing the screen and coding the car
b) Creating & popping toxic gas bubbles
c) Creating & coding the Ozone layer
d) Creating the Welcome & Game Over Screens

Task – I: Designing the screen and coding the car

Let us begin with our first task by designing the Screen for our game.

As usual, delete the default cat sprite and any other images on the screen so that it is empty. Add a backdrop of your choice. Go to the extreme right at the bottom and click on the image icon. Choose an image from the available backdrops, create an image in paint or upload an image. We are uploading the backdrop of a city from our own image library. Convert the image into a vector and Stretch it to fill the screen.

Our next step is to add the sprite of a car. We are sure that you are a pro at adding sprites by now. Do not forget to resize the car after you convert it into a vector. While you are you doing that, are you wondering why do we need a car in our game? The car will keep moving from left to right and gas bubbles that will affect our ozone layer will keep following it. The harmful gas bubbles are emissions from the car itself.

We now need to determine the starting position of the car. Since we want it to move from left to right, we feel an initial position of "-200" on the X axis and "-150" on the Y axis will work well. To initialize the Car position such that it always starts from this position that we have decided upon, we will create 2 variables, one each for both the x and y axes. Go to the variables section, click 'Make a variable' and select the option 'For this sprite only'. We are calling the variables "InitialPosX" & "InitialPosY" respectively.

Create a new event to set the default position of the car on both axes. Drag 'When Flag clicked' block from the events section. From the Variables section, attach 2 instances of 'Set Variable to 0' to this block. Select "InitialPosX" & "InitialPosY" from the drop downs and Set the values of the variables to -200 & -150 respectively. Drag 'go to X & Y' block from the motion section and replace the numbers with the variables 'InitialPosX' & 'InitialPosY'. This logic sets the default position of the car for our game.

The next step is to ensure that our car keeps moving continuously from left to right until the game is over. Create a variable called “GameOver” in the Variables section. Make sure that you make it accessible to all sprites in the game. To initialize the variable, drag ‘set GameOver to 0’ and place it between ‘when flag clicked’ and ‘Set InitialPosX to’ block.

Drag the ‘Repeat Until’ block from the Control section and attach it to the ‘go to x = InitialPosX & y = InitialPosY’ block. As you are aware, the repeat until block keeps doing something as long as the condition in the hexagonal slot is true. From the Operators section, drag ‘is equal to’ operator and place it in the <hexagonal slot>. Go to the variables section and drag “Gameover” variable and place it on the left-hand side of the equal to sign. Change the value on the right-hand side of the equal to sign to 1. This will make sure that the loop runs till the value of the “GameOver” variable becomes “1”.

To move the car from left to right on the axis, drag “Change X by 10” block from the motion section and place it inside the repeat until loop. The number 10 is the current speed of the car which is too fast. Replace it by 5 to make the car slower.

Let us test our car now. Great! The car is moving from left to right.

But there is a small problem. Currently, the car stops at the right edge of the screen once it reaches its maximum x-position. We need to ensure that the car goes back to its initial position i.e. the car should automatically go back to the left edge of the screen.

Create another variable “EndPosX” for this sprite only from the Variables section. Drag ‘Set EndPosX to 0’ and place it between ‘Set InitialPosY to -150’ and ‘go to x & y’. Set the value of “EndPosX” to 280. Drag the “If then else” block from the control section and place it inside repeat until block. Move the ‘change x by 5’ block inside the else part of the condition. Drag the ‘Greater than’ block from the operators section and place it inside the <hexagonal slot>.

Go to the Motion section, take ‘x position’ block and place it on the left-hand side of the greater than condition. Drag ‘EndPosX’ from the variables section place it on the right-hand side of the operator.

Drag ‘Set x to’ block from the motion section and place it inside the ‘if’ loop. Replace the number with the ‘InitialPosX’ block from the variables section. Let us recapitulate our logic. We have ensured that if the position of the car on the x-axis becomes greater than the “EndPosX”, then the car will go back to the “IntialPosX” else the car will keep on moving on the x axis at a speed of 5.

Time to check our coding and see if the car is working as expected.

Wow! We have completed our first task.

In the next video, we will create and pop toxic gas bubbles that will be emitted from the car.

Bye-bye. See you again!