Prevent Ozone Depletion – Part II

Welcome back!

In the last video, we finished designing the screen for our game and coding the car. In this video, we are going to create random gas bubbles and pop those that are harmful.

Our first step is to create a new sprite for a gas bubble. You can create it in paint or choose to upload it. We are uploading a gas bubble of 02 or oxygen that we had created earlier. Go to the costumes section, convert it into a vector and set an appropriate size for the gas bubble.

Now we will have to add more good gases and harmful gases to this section to make the game interesting.

Since we already have O2, we are adding O3, CO, CO2 and NO2. Reorder them such all the good gases, that is, Oxygen and Ozone appear one after the other and all the harmful gases, that is, Carbon Monoxide, Carbon Dioxide, Nitrogen Dioxide appear one after the other.

The good gases are going to heal the Ozone layer and the harmful gases are going to damage the ozone layer. We are also adding 4 bubble burst effects in descending order so that our gas bubbles get a proper burst effect. Make sure that they are named appropriately. We are done with our uploads for now. Time to begin coding.

The gas bubbles should appear like they are coming out from the car and should always follow it until the game is over. Drag 'When flag clicked' from the events section and attach 'repeat until' block from the control section to it. Go to the operators section and drag the 'equal to' block to the <hexagonal slot>. We had created a variable named 'GameOver' while doing our first task. 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".

Go to the motion section and drag ‘go to random position’ inside the repeat until loop. From the drop down, select the car sprite. Since the car is moving to the right, we would like the gas bubbles to move to the left so that it appears that they are coming out of the car. Drag the ‘change X by 10’ block from the same section and place it after the ‘go to car’ block inside the repeat until block. Change the value of in this block to -80.

Now let us test the game. The bubble is following the car at the x-position. All other bubbles will also come out from the same points.

It is now time to create clones of the gas bubble when the car is moving. For that, drag ‘create clone of myself’ block from the control section and place it after ‘change x by -80’ block. From the same section, drag ‘wait 1 seconds’ block and place it after the create clone of block. Change its value to 0.25 seconds.

The gas bubbles are cloning themselves after 0.25 secs but they are getting cut on the edges of the screen. This happens when the car appears from the edges. Hence, we need to ensure that the clones are created only when the car appears fully on the screen. Drag the ‘if then’ condition from the control section and place it after the ‘change x by -80’ block. Move the ‘create clone…’ & ‘wait 0.25 secs’ blocks inside it. Go to the operators section and drag the ‘greater than’ condition to the <hexagonal slot>. Place ‘x position’ from the motion section in the left-hand slot of the ‘greater than’ block. Test for the optimum x-value. For us it is -180. Place this value in the right-hand slot of the ‘greater than’ block.

Let us run the game and check.

Till now, we have accomplished the following. If the x-position of the car is greater than -180, then, a clone of gas bubbles appears after every 0.25 secs on the screen in tandem with the moving car.

To hide the default bubble from which the other bubbles are getting cloned, drag the ‘Hide’ block from the looks section and place it after the ‘when flag clicked’ block.

Now let us code the clones. Drag ‘when I start as a clone’ from the control section and place it in the code editor. Attach ‘show’ to it from the looks section.

Currently, only the first gas is being cloned. However, we have 5 types of gases and we need to create random clones of the four other gases also. Let us create a variable called “GasType” in the Variables section for this sprite only.

Drag ‘set variable to 0’ and place it after ‘when I start as a clone’. From the drop-down menu, select the variable ‘GasType’. Drag ‘pick random 1 to 10’ block from the operators section and replace 0 with it. Change the value from 10 to 5 because we only have five gases.

Now we need to correlate each gas bubble with one of the random numbers. Drag an ‘if then’ block from the control section and place it between ‘set gas type to ’ and show block. Drag ‘equal to’ block from the operators section and place it in the <hexagonal slot> of the ‘if then’ condition. Drag the ‘GasType’ variable from the variables section and place it on the left-hand side slot. On the right-hand side, change the value to 1 to represent the first costume of the gas bubbles sprite. Drag ‘switch costume to’ from the Looks section to inside the ‘if then’ block. Select the O2 costume. This will ensure that when the random value for the variable GasType is 1, an O2 gas bubble will be generated.

We will have to duplicate this process for the other 4 gases. Drag 4 more ‘if then’ blocks between the first ‘if then’ block and show. In each hexagonal slot, drag an ‘equal to’ operator from the operators section. Drag the ‘GasType’ variable and place it on the left hand side slot of each equal to operator .Go one by one and change the numerical values on the right hand side of each equal to operator to the numbers 2, 3, 4 and 5. Drag ‘switch costume to’ from the Looks section inside each ‘if then’ block. Select appropriate costumes from the drop downs. In our case, we are selecting O3, CO, CO2 and NO2 respectively.

Now that we have finished switching costumes, we need to test our code. Wow, random gas bubbles of different gases are being created now.

Our gas bubbles should move up on the Y axis towards the sky. Drag the 'forever' loop from the control section and place it after the show block. Drag 'change y by 10' block from the motion section and place it inside the forever loop. Change the value from 10 to 2 to slow down the movement of the bubbles. You can choose a higher value of your choice to make the game tougher.

Test your code. The bubbles are now moving upwards at a speed of 2 due to the forever loop.

We now need to code the bubbles such that they burst when clicked. Drag the 'when this sprite clicked' block from the events section and place it in the code editor window. From the looks section, drag 'switch costume to…' and attach it to 'when this sprite clicked block. From the drop down, select the first costume of the bursting effect. You can check the name of the effect in the costumes section and select accordingly.

Play the sequence of other effects to give a proper bursting effect. Drag a 'repeat' block from the control section and attach it to the 'switch costume to…'. Change the value from 10 to 3 because we only have 3 more bursting effects in our costume section. Drag 'next costume' block from the looks section inside the repeat loop.

Test the game and try bursting bubbles by clicking them. The game seems to be working fine.

We need to take care of two more things before we sign off. One, let us rename this sprite to Gas Bubbles for ease of usage. Second, while we cannot see the bubbles after they burst because of the empty costume, they still remain on the screen. We need to make sure that they disappear.

Drag 'delete this clone' block from the control section and attach it to 'when this sprite clicked' code after the 'repeat 3…' block. This will delete the bubble completely from the screen when one clicks on it.

Congratulations, you have finished the first two tasks of our game to prevent the ozone layer from depletion.

In the next video, we will create the ozone layer and code it so that it gets affected by the gas bubbles.

Bye-Bye. See you soon!