GD 12
Build The Roller Coaster, Part 2

This is a continuation of GD 11. We will be adding onto that code.

In GD11, we created the ramp by using "**M**" to move the line to the previous mouse position. Next, we use "**L**" to draw a line from the previous mouse position to a point extending all the way to the current mouse position.

In this lesson, we will activate the game and focus on code that will build the ramp on a correct guess.

Build The Roller Coaster – Press down on the left button of the mouse. Next, move the mouse around to draw your ramp. The goal is to match the national flag with the correct nation. Answer quickly and more ramp is build to extend the or else the roller coaster will fall into the pit of misery and the game is over.

1. Ramp Control 2. Pit of Misery

---

1. Write the code below in between <script> </script>. The **large, green banner** is your landmark. **Go to the coding website and look for it.** Next, write the code below underneath the **large, green banner**. **Write all of it, color code is for explanation.**

```
/* ==================================
* Start Game
* ================================== */
case 'd':
case 'D': init_sys();
     break;
```

Explanation

The code above uses a switch to map a letter to an action. In our case, we assume that the player is down drawing the ramp. When the player clicks on the letter "d" on the keyboard, the letter "d" is linked to the action "**`init_sys()`**". The will then cause the game to actually begin.

2. Write the code below in between <script> </script>. The **large, green banner** is your landmark. **Go to the coding website and look for it.** Next, write the code below underneath the **large, green banner**. **Write all of it, color code is for explanation.**

```
/* ======================================
* Pit of Misery
*
* if we are beyond the array, then drop down.
* Else, there is still track left so keep going
* ====================================== */
if( rcPointCount >= rcRamp+(p1Score*rcRampOffset) )
{
      pitOMFlag = 1;

}
```

Explanation

After the player has drawn the ramp and presses down on the ‘d’ button on the keyboard, the game draws the first part of the ramp. The goal is to answer questions quick enough to build more ramp. However, if we don’t build ramp fast enough, then the roller coaster fill fall down.

The code above checks if the x position of the roller coaster is greater than the last ramp built. **If true, then the roller coaster is beyond the ramp**. This means that the roller coaster has moved to an (x,y) position where there is no more ramp left. So, we signal to execute the falling down animation.

3. Write the code below in between <script> </script>. The **large, green banner** is your landmark. **Go to the coding website and look for it.** Next, write the code below underneath the **large, green banner**. **Write all of it, color code is for explanation.**

```
/* ======================================
* Falling Down
* ====================================== */
if( pitOMFlag )
{
  ctx.strokeStyle = "#5B74FF";
}
else
{
  ctx.strokeStyle = "#ffa500";
}

ctx.stroke(pathObj);

console.log( "timeDelay: " + timeDelay );

// --
if( pitOMFlag )
{
    rcXPoint = currArr[rcRamp+(p1Score*rcRampOffset)].x;
     rcYPoint = currArr[rcRamp+(p1Score*rcRampOffset)].y++;
```

```
    }
    else
    {
        rcXPoint = currArr[rcPointCount].x;
        rcYPoint = currArr[rcPointCount].y-10;
    }
```

Explanation

The code above executes the falling down animation. First, we change the color of the ramp from orange to blue.

Remember that falling down is the y position. So, the code in **orange** adds 1 to the y position while the x position stays the same. If we keep adding 1 to the y position, then the roller coaster falls down slowly.

4. Write the code below in between <script> </script>. The **large, green banner** is your landmark. **Go to the coding website and look for it.** Next, write the code below underneath the **large, green banner**. **Write all of it, color code is for explanation.**

```
/* ====================================
* Draw Roller Coaster
* ==================================== */
for( var p = 1; p < rcRamp+(p1Score*rcRampOffset); p++ )
{
    pointString += " L " + currArr[p].x + " " + currArr[p].y + " ";
    pointStringShadow += " L " + currArr[p].x + " " + (currArr[p].y+2) + " ";
}

// -- create path using points
var pathObj = new Path2D( pointString );
var shadowObj = new Path2D( pointStringShadow );

// console.log( "hi 2" );
// -- draw shadow
ctx.beginPath();
ctx.strokeStyle = "#303030";
ctx.stroke(shadowObj);

ctx.beginPath();
```

Explanation

The code above draws the ramp onto the screen. Notice that the code in **purple** uses the player score to control how much ramp is build. The higher the score, the more ramp is build.

For example, assume that the player score is **5**. This means that **5** (x,y) points are connected together to make the ramp.

If the player correctly matches a flag with a nation, the player's score goes up by 1, which is now **6**. This means that **6** (x,y) points are connected together to make the ramp.

If the player has a score of **15**, this means that **15** (x,y) points are connected together to make the ramp. Thus, the player's score controls the amount of (x,y) points that are connected together and this creates the ramp that we see.

JS Challenge 1

Right now, the background scrolls right when we match the national flag with the nation. We can improve upon the game by adding another distraction in the form of a ghost.

The ghost starts at the bottom of the screen and flies in the upwards direction.

1. **Write Level 1 Code**

1.1. Create variables for the x position, y position, width of the ghost, and height of the ghost
1.2. The width and height of the ghost should both be 75
1.3. What values should be loaded into the variables for x and y position? Use the text in **red** as a hint
1.4. Put your code under the banner

```
/* ==================================
* JS Challenge 1
* ================================== */
```

JS Challenge 2

Now, let's focus on moving the ghost in the upwards direction one step at a time.

1. **Write Level 2 Code**

1.1. How do I move the ghost in the upwards direction? Remember that (0,0) starts at the top left. So, going down can be done by adding by 1 **while going up can be done by subtracting by 1. HINT: only change the variable that controls vertical movement.**
1.2. Use and modify the code below

```
ctx.drawImage( ghostImage, );    // -- variables here? Finish it
ctx.fill();
```

1.3. Put your code under the banner

```
/* ==================================
* JS Challenge 2
* ================================== */
```

JS Challenge 3

We got the ghost to start at the bottom and fly in the upwards direction. **The problem is that the ghost will continue to fly in the upwards direction forever**. We need to tell the ghost to come back after it is beyond the ceiling of the screen.

Write Code To Reset The Ghost

1.1. Write the code to check **if the ghost's y position is less than 0. If true, load the value of 550 into the ghost's y position variable**

1.2. Put your code under the banner

```
/* ==================================
* JS Challenge 3
* ================================== */
```