GD 21
Rock Climbing, Part 2

In this second lesson on Rock Climbing, we will focus on deceiving the game player by writing code to have the background scroll up or down. Next, we will focus on using many "if" statement chains or a single, long "**`if-else if-else`**" chain to select a message to be displayed onto the GUI.

Rock Climbing. How well do you know California? The goal of Rock Climbing is to start at the bottom and climb up to the top. The Rock Climber makes it up to the top by match the question with 1 of 4 possible answers. The Rock Climber can get into position by zip lining side ways or vertically.

1. arrays 2. class definition 3. "**this.**" versus object name

---

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.**

```
/* ==========================================
* Background image scroll animation
* ========================================== */
if( stateCS == 0 ) // -- reset
{
        newSetFlag = 0;
        stateCS    = 4;
        currCount  = 100;
}
else if( stateCS == 1 ) // -- climb up
{
      // if( clipY >= clipYLimit )
        if( currCount > 0 )
        {
            currCount--;

             clipY -= .5;
            cArr[0].cYPos += 1;
            cArr[1].cYPos += 1;
            cArr[2].cYPos += 1;
            cArr[3].cYPos += 1;

            playerArr[0].pYPos += 2.5;
        }
        else
        {
            stateCS = 3;
        }

      }
```

```
else if( stateCS == 2 ) // -- climb down
{
     playerWB         = 0;
     // selectedPlayerId = 0;

     // if( clipY <= clipYLimit )
     if( currCount > 0 )
     {
        currCount--;
         clipY += .5;
        cArr[0].cYPos = 150;
        cArr[1].cYPos = 150;
        cArr[2].cYPos = 150;
        cArr[3].cYPos = 150;

        playerArr[0].pYPos = 350;
     }
     else
     {
        stateCS = 3;
     }
}
else if( stateCS == 3 ) // -- new set
{
     chosenPos   = Math.floor( Math.random() * indexObjArr.length );
     newSetFlag  = 1;
     stateCS     = 0;
}
else if( stateCS == 4 ) // -- wait for input
{
     stateCS = 4;
}
else
{

}
```

Explanation

Remember from GD 4 that you are being deceived. The main game character is not moving but the background is. The background moving means that there is a change of scenery and this gives the impression that the game character is advancing forward or backwards.

The Rock Climbing game makes heavy use of background side scrolling to give the impression that the main game character is climbing up or down the rock. Remember that in video game development, the top left is the starting x and y position. This means that (0,0) is on the top left. So, in order to climb, the background must scroll down and climbing down means the background scrolls up.

While the background scroll is going on, we want to reset the 4 possible answers.

If we are correct, the code in grey will cause the 4 possible answers to move towards the bottom of the screen and this also gives the impression that the climber is going up.

If our guess is off, then the code in **orange** will cause the 4 possible answers to reset to the top of the screen. Since the answers are now towards the top, it look like the rock climber has gone down the mountain.

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.**

```
/* ====================================
* Game message calculation
* ==================================== */
if( p1Score >= scoreLimitArr[3] )
{
      gameMsg = "You Beat The Game!!!";
}
else if( p1Score >= scoreLimitArr[2] )
{
      gameMsg = "Level 4 reached";
      // levelUp = 4;
}
else if( p1Score >= scoreLimitArr[1] )
{
      gameMsg = "Level 3 reached";
      // levelUp = 3;
}
else if( p1Score >= scoreLimitArr[0] )
{
      gameMsg = "Level 2 reached";
      // levelUp = 2;
}
else
{
      gameMsg = "";
}
```

Explanation

The code above is the score keeping and messaging to motivate the game player when a new level is reached.

The use of "**`if-else if-else`**" is used because the game player can only be on one level at any one time. The "**`if-else if-else`**" statement only picks one of the conditions to be true and then exits the entire chain.

If we were to write many "**`if`**" statements then each "**`if`**" statement would be the start of a new and different chain. Since there are multiple chains and each chain is started with an "**`if`**" statement, then the code would have to check each chain. This is ok but if we know for sure that a game player can only be on one level at any time, then using the "**`if-else if-else`**" chain is better.

JS Challenge 1 – Array Combined with Objects. Type the answer in Zoom chat

In previous projects, we loop through an array of numbers. What if we want to loop through an array of object? We still use a "**for**" loop but we also need to use the square brackets and the member access operator.

Assume that **class GamePlayer** and array named **gpArr** already exists. Write a "**for**" loop and do the following

1. Declare a digital key named g and load the data 0 into it
2. The end of the loop is the length of the array
3. Jump by 2

Inside the body of the "**for**" loop, do the following

1. Declare a variable named **currObj**.
2. Create a **new** object of **class GamePlayer**
3. Load the newly created object into the variable **currObj**
4. Select position **g of gpArr**. Next, use the **member access operator** to call the **private function load_new_obj().**
5. The input into the **private function load_new_obj()** is the variable **currObj**;