GD 5
Scrolling In Space

In GD 4, we learned about side scrolling games and then implement a basic game that scrolls left or right based on the direction key the user pressed down on. In this game, we will add onto the side scrolling game and combine all previous topics learned.

The game is called "**Scrolling For Money**". There is a lot of debris in space. We are going to use Pac Man to avoid the debris and only collect the money to increase our score by 1. The extra feature is when and how the background will scroll. We no longer control the main character using the keyboard. Instead, we use the computer mouse for character control.

Scrolling For Money – If we have a positive score, the background scrolls to the right while a negative score means the background scrolls to the left.
.

1. Control Pac Man using mouse
2. Side scrolling game

---

0. P1 SCORE. LEVEL 1 - INPUT COMMANDS
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.**

```
/* ========================================
* Variables to display player score
*
* ======================================== */
var
p1Score  = 0,
levelUp  = 1,
p1LF     = 0,
p1LArr   = [],
laserLim = 2;

var
p1ScoreText = "P1 Score: ";
```

Explanation

With HTML 5 Canvas, there is an invisible pen that is controlled by using variables to store input commands. The HTML 5 Canvas tool draw uses our input commands to draw game objects onto the screen. So far, we have drawn game objects but we can also draw text.

The code above uses variables to set how the text will look when it is drawn onto our screen. **This is level 1 where we load data into variables and those variables will be used as input commands.**

1. DRAW TEXT ON SCREEN. LEVEL 2 – DRAWING
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 text
*
* ==================================== */
ctx.font = '28px serif';
ctx.fillText( p1ScoreText, 10, 50);
ctx.fillText( p1Score, 140, 50);
ctx.fillText ( "Laser Count: ", 320, 50 );
ctx.fillText ( laserLim, 480, 50 );
```

Explanation

This is our first time seeing `ctx.fillText()`, which is level 2 because it will draw for us. The drawing function takes 3 input commands.

> The **first argument** is the data that is going to be drawn onto the screen.
>
> The **second argument** is the x position that the invisible pen will begin to draw.
>
> The **third argument** is the y position that the invisible pen will begin to draw.

The `other 3 red ctx.fillText()` are given **different input commands** and so will draw something different and at different (x,y) positions.

CONTINUE TO THE NEXT PAGE

2. WHICH ONE IS MONEY

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.**

```
/* ====================================
* Check Money Ball
*
* ==================================== */
check_money_ball()
{
       this.cID = Math.floor(Math.random() * 2);


        if( this.cID )
        {
              this.imgIndex = asteroidImgSrcArr.length-1;
             console.log( "----------imgIndex: " + this.imgIndex );
       }
        else
        {
             this.imgIndex = Math.floor(Math.random() * (asteroidImgSrcArr.length-1) );
        console.log( "asteroidImgSrcArr.length-1: " + (asteroidImgSrcArr.length-1) );
             console.log( "////////////imgIndex: " + this.imgIndex );
       }

        console.log( "++++++++++++imgIndex: " + this.imgIndex );


}
```

Explanation

The code in purple will randomly pick if the game object should be a money sign. First, it will pick a random number between 0 or 1 and load the number into a private variable called `this.cID.`

Next, the **"if"** statement will check if `this.cID` is equal to 1. If so, the costume number for money is assigned to `this.imgIndex.`

Else, a regular costume is picked.

3. RESET TO RIGHT SIDE
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.**

```
/* ========================================
* Reset when touching left side or Pac Man
*
* ======================================== */
reset_to_right_side()
{
      this.update_pos( 580, Math.floor(Math.random() * 180 ) );
      console.log( " ---------- reset_to_right_side() ----------" );
      this.check_money_ball();
}
```

Explanation

If a game object touches either the Pac Man or the left side of the boundary, then the game object will reset to the right side and start again. This gives another impression of a side scrolling game.

Once the game object is reset to the right side, it will be given a new costume number. Since resetting to the right side is a brand new round, the game object will be given a new costume. This is why `this.check_money_ball();` is called again.

CONTINUE TO THE NEXT PAGE

4. DRAW GAME OBJECT & CHECK FOR COLLISION
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 Circle and check for collision
*
*
* ======================================== */
for( var c = 0; c < cArr.length; c++ )
{

       ctx.fillStyle = "#" + cArr[c].cColor;
        cArr[c].move_one_step();
        cArr[c].check_sb();

         // -- check if asteroid touched pac man
         if( cArr[c].check_hb_pad( pImgClipX, pImgClipY, pImgClipW, pImgClipH ) )
        {
                if( cArr[c].cID )
              {
                   p1Score++;
                   panROffset++;
                   panRLim += 150;
                   panDirNS = 1;
              }
              else
              {
                    p1Score--;
                    if( p1Score < 0 )
                    {
                        panLOffset++;
                        panLLim += 150;
                        panDirNS = -1;
                    }
                    else
                    {

                    }

               }

        }


         // -- Level 2, Drawing
         ctx.beginPath();
         ctx.drawImage( asteroidImgSrcArr[cArr[c].imgIndex], cArr[c].cXPos,
cArr[c].cYPos, cArr[c].cRad, cArr[c].cRad );
         ctx.fill();

}
```

Explanation

Our game has many circle game objects and each circle game object has a different costume. **Remember that every 16 milliseconds,** we have to erase everything from the screen and then redraw every game object in their updated (x,y) position.

We can't leave out one object and forget to draw it. This is why the "**for**" loop is needed to cycle through and redraw every circle game object. Notice that we are using the digital key of **c** and doing this allows us to start at circle 0, then circle 1, then circle 2, and etc. We go from one circle to the next by using **c++;**

The **code in gold** updates the x and y position of the circle (`cArr[c].move_one_step();`) and then check if the circle has touched the side boundary (`cArr[c].check_sb();`)

The **code in green** checks if any circle game objects touched our Pac Man. If so, then the code in **blue** checks if it is a money costume and adds 1 to the player's score.

Finally, the **code in grey is** redraws each circle game object in their updated (x,y) position.

5. MOUSE CONTROL EVENT LISTENER
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.**

```
/* ======================================
*  Control using mouse
*
*
* ====================================== */
window.addEventListener( "mousemove", update_pman_pos );
```

Explanation

Our game now uses the computer mouse to move the Pac Man in the y axis. The code above sets the event listener for the window, which is the entire website.

The window will "**listen**" any time the computer mouse changes position and the event "**mousemove**" is the event that is triggered when the computer mouse changes position.

The event must be taken care of and the event handler, which is our action, is to call the JS function definition "**update_pman_pos**"

6. MOUSE CONTROL EVENT HANDLER

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.**

```
/* ====================================
* Use mouse y position to control
* main character
*
* ==================================== */
function update_pman_pos( eData )
{
      //mouseX = eData.offsetX;
      mouseY = eData.offsetY;

}
```

Explanation

The code above is the event handler that is triggered or called whenever the computer mouse changes positions. We want to move Pac Man in the vertical direction that this is why we load the y position of the mouse into the variable **mouseY**.

The Pac Man character will then be redrawn in the position that the mouse cursor is at on the y axis.

HTML CHALLENGE

Pick an image from the internet and use it as your background.

Look for an image and change the code in **red**

```
<!-- ============================
* Background Image
*
* ============================== -->
<img id="gameBG"
src="http://vuongducnguyen.com/images/gamebackground.jpg"
alt="gameBG" width="600" height="500">
```

JS CHALLENGE 1

You can change the costumes of the asteroids. Pick one or many images from the internet and change the code in red.

```
/* ================================================
*  Image for circle --> asteroid array
*
*
* ================================================ */
imgTemp        = new Image();
imgTemp.src    = "https://vuongducnguyen.com/images/asteroid.png";
// imgTemp.src    = "../images/packM.png";
asteroidImgSrcArr.push( imgTemp );

imgTemp        = new Image();
imgTemp.src    = "https://vuongducnguyen.com/images/space_debris.png";
asteroidImgSrcArr.push( imgTemp );

imgTemp        = new Image();
imgTemp.src    = " https://vuongducnguyen.com/images/spike.png";
asteroidImgSrcArr.push( imgTemp );

imgTemp        = new Image();
imgTemp.src    = " https://vuongducnguyen.com/images/money.png";
asteroidImgSrcArr.push( imgTemp );
```

JS CHALLENGE 2 – can do in Chat of Zoom

1. Create an array named "`costumeArr`"
2. Make it an empty array
3. Use the `.push()` method to put data into the array. You can push any data you want
   3.1. does the `.push()` method put data at the end or at the front?

JS CHALLENGE 3 – can do in Chat of Zoom → This is a continuation from JS Challenge 2

Write a "`for`" loop and loop and loop through "`costumeArr`"

1. Create a digital key named `c` and load the first index into it
2. Loop through until you are at the end of "`costumeArr`"
3. Jump to the next costume by 1 position
4. Inside the body of the "`for`" loop, do the following
   3.1. create a variable named "`myCostume`"
   3.2. use square brackets and the digital key from step 1 to select a single costume
   3.3. load the data from the selected array position into the variable "`myCostume`"