GD 4
Scrolling Background

The Falling Dot in GD 3 had costumes that would change after every successful click by the player. After 3 points were achieved, the background would "**scroll**" to the right and switch scenes. This was an introduction to side scrolling game.

In this lesson, **we learn why side scrolling games are important and then use VIEW FRAME to implement side scrolling.** When done, our Pac Man can **run left and right** while deceiving the player.

Running Pac Man – let's use deception to make our Pac Man run sideways.

1. Video game industry crash
2. Side scrolling game.
3. Run left and right

---

0. DECEPTION

Did you know that you are being deceived? Video games are all about deceiving the player into thinking one thing while the reality is opposite. One of the most important video games is Super Mario Bros. This game helped save the video game industry and brought it back to the fore front.

Source: https://supermariobros.io/data/image/super-mario-bros.png

0.1. POWERFUL ARCADE BOXES

Traditionally, video games are found in shopping malls in the form of arcade boxes. They used to be popular in the US and are still popular in Japan. In the early eras of video game development, entertainment meant that expensive and big hardware components were needed to be able to process the game play and graphics. This is why arcade boxes had all the big graphic intensive games.

Source: https://www.recroommasters.com/v/vspfiles/photos/RM-XT-32-PRO-EE-2T.jpg

0.2. PROBLEM : BIG AND BULKY

Arcade boxes housed powerful hardware components that were needed to handle the game play and graphics that many people desired. However, the problem with arcade boxes is that they are big, bulky, and expensive.

Who can afford it? NOT the average consumer

Where do you put it? Arcade boxes are huge and take up a lot of real estate.

Where does the energy come from? Intensive game play and graphics requires a lot of energy. The average consumer does not want to pay that type of money

How often is it used? A big, bulky, and expensive arcade box is a big investment that only pays off if it is used often. The average consumer might be too occupied with school and work and this prevents daily use.

From the above, the game industry did crash due to low demand.

0.3. SIMPLE IS ALWAYS BEST

0.3.1. GAMEPLAY

So, how did the video game industry recover? People began to realize that video games can still be fun if the game play has the **following attributes**

1. goals are simple to understand
2. simple to achieve
3. has repetitive actions (this promotes mastery)
4. breaking a video game into levels promotes achievement in incremental steps.

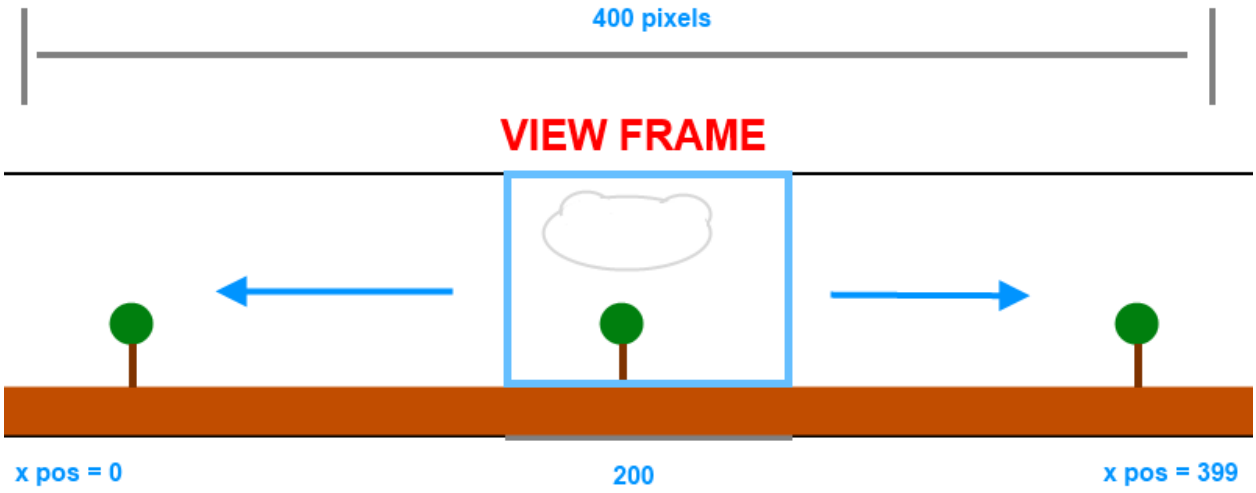
0.3.2. RENDER

With game play taken care of, **how can video game developers simplify the rendering (or drawing) of video game objects**? **The simple answer is to deceive** the player and Super Mario Bros. did this very well.

Look at the game play of Super Mario Bros and ask yourself, is Mario actually moving? If so, then answer the questions below

1. What direction is Mario moving in?
2. What direction is the background moving in?
3. What direction are the enemies moving in?

Based on the answers from above, we do realize that Mario is not actually moving. The game play made it look like Mario is moving and this deception is **called side scrolling game.**



0.3.3. VIEW FRAME

In order to create a side scrolling game, we need a **VIEW FRAME**. This **VIEW FRAME** is a box that only partially shows the entire background and **this is part of the deception**.

The game character moves the **VIEW FRAME** left or right based on the arrow keys on your keyboard. The character is centered in the **VIEW FRAME**.

However, the character does not move and is static (meaning the character doesn't move). Instead, the **VIEW FRAME** moves left and right and the changing background gives the perception of a character moving.

0.4. SIDE SCROLLING GAME

This type of game simplifies game play and rendering by making the main character the focal point. This means that the main character is like the sun and everything and everyone revolves around that main character.

Since the main character is the focal point, he/she is static and stays in the same (x,y) position the entire game. To make it look like the main character is moving, we simply scroll the background sideways and this is why it is called a side scrolling game.

0.4.1. RULE OF OPPOSITE

To make Mario run to the right, we have the background scroll in the opposite direction, which is to the left.

0.4.2. COSTUME CHANGE → MORE DECEPTION

In the last project, GD 3, we focused on using arrays to store costumes and then using a digital key to select a costume within the array. To make Mario look like he is running, we load different costumes in an array and each costume shows Mario moving his legs.

If we quickly change costumes, the background scrolls left and Mario's legs change positions. This gives the effect of Mario running to the right.

0.5. Recovery

Using a **VIEW FRAME** and static game character was important because it became a standard way of making video games. So, instead of having to figure out what works and what doesn't work, developers have a side scrolling style game that **DOES work** and is simple to use.

Now that the foundation building block has been set, developers can focus on adding more block on top.

1. ARROW KEY ACTIVATES SCROLL - 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.**

```
/* ======================================
*Arrow keys event listener
*
*
* ======================================*/
window.addEventListener( "keydown", check_key );
```

Explanation

The code in **green** is an event listener and this is how JS listens for events and then acts. In our case, we are saying that the **window**, which is your entire website, should "**listen**" for a depression of the keyboard.

When this event does happen, the code in **blue is the event handler** (which is our reaction to the event).

2. EVENT HANDLER (OUR REACTION)

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 key
*
*
* ====================================== */
function check_key( e )
{
      console.log( "e.key: " + e.key );

      /* ==========================================
      Why does going up mean subtracting by 10?

      */
      switch( e.key )
      {
        // --
        case 'w': move_left_paddle( -50 );
              break;

        // --
        case 's': move_left_paddle( 50 );
              break;

        // --
        case 'o': move_right_paddle( -1 * p2YOffset );
              break;

        // --
        case 'l': move_right_paddle( p2YOffset );
              break;

        // -- space bar
        case " ": console.log( "space bar was pressed" );
                  p1LF = 1;
                  //audioDOM.play();
                  console.log( "((((( done playing audio effect ))))))" );
              break;

        // -- up arrow
        case "ArrowUp": console.log( "^^^^^^^^ arrow up ^^^^^^" );
                        p1JF        = 1;
                        costumeNum  = 0;
                        break;
```

```
        // -- right arrow
        case "ArrowRight": console.log( "&&&& arrow right &&&&&" );
                           panDirNS    = 1;
                           panROffset = .5;
                           panRLim    = 150;
                           costumeNum  = 2;
                           break;

        // -- left arrow
        case "ArrowLeft": console.log( "$$$ arrow left $$$" );

                          break;

        // -- down arrow
        case "ArrowDown": console.log( "///arrow down////" );
                          panDirNS    = 0;
                          panLOffset  = 0;
                          panLLim     = 0;

                          break;

        // -- safety net
        default: console.log( "nothing happened" );

      }

}
```

Explanation

The event handler code from above will first determine which key was pressed down by the player. We use the switch statement to map a key to an action.

In the code in **blue**, "**ArrowRight**" is mapped to the code that changes the x position of the background image and this is how we implement the scrolling.

When the game starts, the x position of the background image is 0. When we press on the right arrow key on the keyboard, the code will add 5 to the x position of the background image. The result is that we are swiping left a little bit and the background goes to the right.

3. SCROLL TO X POSITION – LEVEL 1 → INPUT COMMAND

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
*
*
* ================================ */
panDirCS = panDirNS;

// -- scroll right
if( panDirCS == 1 )
{
	panOffsetCS = panROffset;
	panLimCS    = panRLim;

	// -- since changing direction, reset the scroll left vars
	panLOffset = 0;
	panLLim    = 0;

}
// -- scroll left
else if( panDirCS == -1 )
{
	panOffsetCS = panLOffset;
	panLimCS    = panLLim;

	// -- since changing direction, reset the scroll right vars
	panROffset = 0;
	panRLim    = 0;

}
// -- user pressed on down arrow key, stop
else
{
	panDirCS = 0;
	panDirNS = 0;

	panROffset = 0;
	panRLim    = 0;

	panLOffset = 0;
	panLLim    = 0;

}


// --
if( panLimCS > 0 )
{
	clipX += panOffsetCS * panDirCS;
	panLimCS -= .5;
}
```

Explanation

In the code above, we are working on level 1, which is to set the input commands.

Remember the switch statement that was written in step 2? Look back and see that it was used to choose which direction to scroll, either left or right.

The **orange "if" statement** checks if the user wants to scroll right, which would be a **code number of 1**. If so, then set the input commands to make the background image scroll right.

However, if the **code number is -1**, then we want to scroll left and the **blue "if" statement** sets the input command to make the background image scroll right.

Finally, the **code in purple** will continue to scroll the background until the desired x position is reached.

4. BACKGROUND SCROLL – 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 background
*
*
*
* ================================ */
ctx.beginPath();
ctx.drawImage( img, clipX, clipY, clipW, clipH, imgX, imgY, imgW, imgH );
ctx.fill();
```

Explanation

In the previous game development, GD 3, we drew a background image and this background image scrolls to the right when a score of 3 or more is achieved. This was our first look at a side scrolling game.

The previous section added 5 to the x position of the background and this was level 1 - setting variables to be used as input commands.

The code in green above is level 2, which is the actual drawing. The code in green above uses the variables as input commands to the drawing function named **`ctx.drawImage()`, which will draw on our behalf.**

HTML CHALLENGE

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

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

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

JS CHALLENGE 2

You don't have to use Pac Man. Pick an image from the internet to be used as your main character and change the code in **red**.

```
/* ========================
* Main character
*
*
* ========================= */
pImg          = new Image();
pImg.src      = "http://vuongducnguyen.com/images/packM.png";
costumeArr.push( pImg );
costumeNum    = 1;
costumeOffset = 109;
hangTimeCount = 0;
intervalCount = 0;
```

JS CHALLENGE 3

Go back to the event handler code and add a case for "`ArrowLeft`". This time, we are going in the opposite direction, so is that plus or minus 5?

JS CHALLENGE 4

We got the background to scroll at the same speed. **This is ok but it means that our character will always be walking.** What happens if an enemy is running towards us and our character can only walk? The enemy has a feature that we don't have and that feature is running.

Implement running so that your character can also run away from the enemy. Look at the "`ArrowRight`" switch case and make changes to the right side of the colon. **The right side of**

the colon has variables that are used as input commands and changing the input commands changes the behavior of the scrolling.

> **JS 4.1.** Modify the code so that every time the player presses down on the right arrow key, the background will scroll faster
>
> **HINT:** how does a variable add onto itself?

JS CHALLENGE 5

Repeat JS Challenge 4 and do the same for the "`ArrowLeft`" case