GD 10
Capital Of The Universe, Part II

This is a continuation of GD 9. You will add code onto that website.

Gaining or losing points allows a game to have a reward system. This reward system motivates players to continue playing the game to acquire more points in order to reach the next level. A point is gained or lost based on the ball touching a ring. How do we know if the ball has touched one of the rings?

In this lesson, we will focus on the rising and falling animation of the ball and using the lower and upper boundary inside an "if" statement to check for collision detection.

Capital Of The Universe – Balance the ball with the laser while matching the city with the state capital.

1. Rising and falling animation
2. Hit box calculation

---

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.**

```
/* ======================================
* Ball Bounce Animation
*
* ====================================== */
 if( bounceFlag )
{
	cYDir		= -1;
	cYStep		= 3;
	currBounceCount--;

	if( currBounceCount <= 0 )
	{

		bounceFlag	= 0;
		cYDir		= 1;
		cYStep		= 1;
	}

}
else
{

}

cYPos += cYStep * cYDir;
```

Explanation

The code in **red** and **orange** is important because it is the code that creates the rising and falling animation of the ball.

> *It is assumed that the user has already pressed the space bar key, lightning has been released, and the lightning has touched the ball using hit box.*

The code in **red** will first cause the ball to **rise into the sky by** changing the direction of the ball from **positive to negative**, where negative will make it rise. Next, it sets the step amount to 3 steps.

So, after the screen is erased, the code will update the y position of the ball by adding 3 steps and then storing the new value into the variable `cYPos.` Finally, the code will redraw the ball in this new position, which is 3 steps more than the previous position. This is how the ball looks like it going up.

The code in **orange** will wait for the ball to rise a bit and then change the direction of the ball. Previously, the code in **red** made the direction of the ball negative to make the ball rise into the sky. The code in **orange** will now transition the direction from **negative** to **positive** and **this will make the ball drop**. We don't want the ball to drop too fast so we set the step amount to 1.

The ball keeps falling until we press the space bar to release the laser. When the laser touches the hit box of the ball, the "`if( bounceFlag )`" code is true and the code in **red** executes again.

CONTINUE TO THE NEXT PAGE

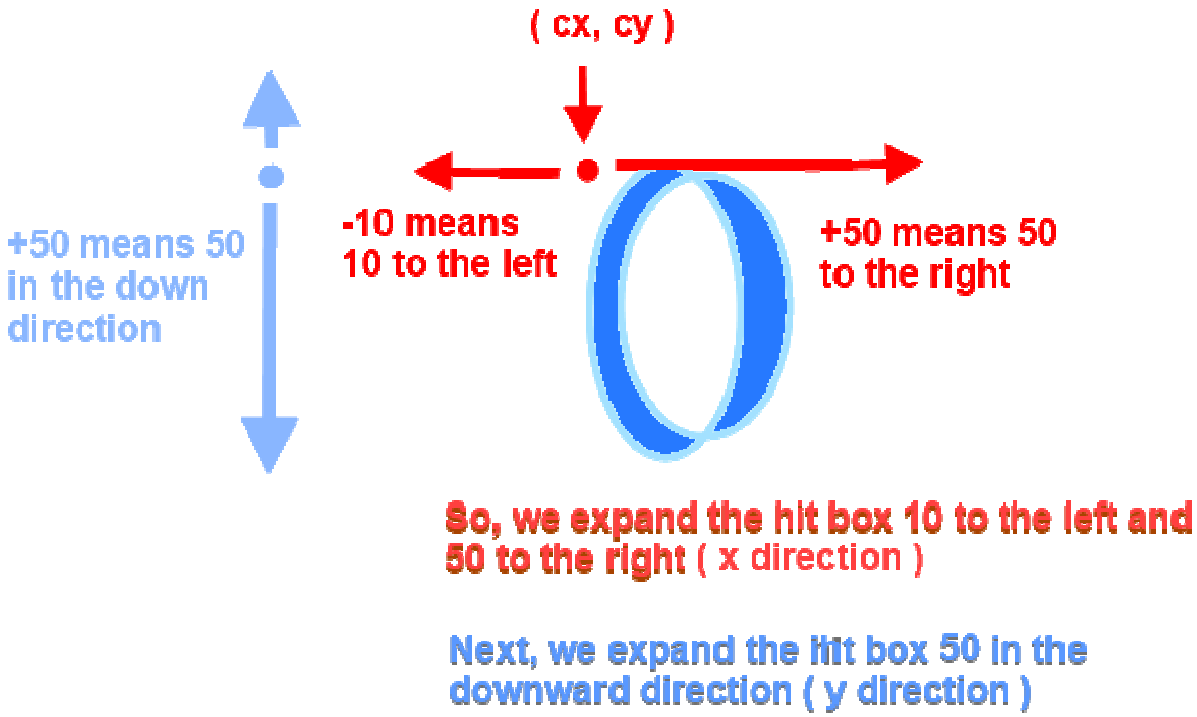
2. Write the code below in between <script> </script>. The **large, green banner** is your landmark. **Go to the coding website and look for it.** Next, write the code below underneath the **large, green banner**. **Write all of it, color code is for explanation.**

```
/* ============================================
* Hit box : ball touching ring
*
* ============================================ */
function check_hb_answer( cx, cy, cr )
{

      // -- check if ball is inside the hitbox of a ringe
      if( cx-10 <= cXPos && cXPos <= cx+50 && cy <= cYPos && cYPos <= cy+cr )
      {
        console.log( " *************** hit a ring *************** " );

        return 1;

      }
      else
      {
        return 0;

      }

}
```

Explanation

The important part is the long "if" statement, which is the code to detect collision between the ball and the ring. How do we check "if" there is a collision?

CONTINUE TO THE NEXT PAGE




```
// -- check if ball is inside the hitbox of a ring
if( cx-10 <= cXPos && cXPos <= cx+50 && cy <= cYPos && cYPos <= cy+cr )
```

The **first thing** we have to do is **create a boundary** for the ring. A boundary has a **lower boundary** and an **upper boundary**. The code in **orange** is the x and y boundary of the ring.

Second, we check **if the ball is INSIDE the lower boundary AND upper boundary of the ring**. If so, then it is a collision.

Notice that the code in **red** checks **if the ball is greater than the lower boundary AND less than the upper boundary** → this is how we check if a game object is within the boundaries.

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.**

```
/* ============================
* Hit box check
* ============================ */
// -- if statement here
  {
    //cYDir *= -1;
    this.cXDir *= -1;

  this.incr_speed();
  this.update_size( -10 );

  padHit = 1;
   }
```

Explanation

You don't need to write the code above, it is already there!!!

Instead, **write the code** to check "**if**" the ball is within the hit box of the paddle. Use the image below to help you write the "**if**" statement.

For the x position, what is the lower boundary and upper boundary?
For the y position, what is the lower boundary and upper boundary?