



## Complejidad y automatización del juego

### ¿Cuál es nuestro objetivo para este módulo?

Hacer nuestro juego más complejo y asignar inteligencia artificial a la paleta para que pueda jugar por sí sola.

### ¿Qué logramos en clase el día de hoy?

Entendimos los operadores lógicos OR (o), AND (y), NOT (no).

Entendimos estos operadores a través de la programación.

Aumentamos la complejidad en el Juego de Escape a medida que aumenta la puntuación.

Asignamos IA a la paleta, agregando lógica a la paleta de la computadora para que se mueva por sí sola.

Asignamos velocidad a los ladrillos para que comiencen a moverse en el estado PLAY (Jugar).

### ¿Qué conceptos/bloques de código cubrimos el día de hoy?

Operadores lógicos.

Combinar condiciones usando operadores lógicos.

Aumentar la velocidad de la pelota.

### ¿Cómo hicimos las actividades?

1. Usar programación condicional para escribir un programa que nos ayude a verificar si el número es divisible por 2 y si el número debe ser mayor que 60.



```
1
2 function check(num1) {
3
4     if(num1%2==0 && num1>60)
5         console.log("verdadero");
6     else
7         console.log("falso");
8
9 }
10
11 check(32);
12 check(62);
13 check(65);
14
```

Comandos para depurar

Interrumpir Step over

Consola

```
"falso"
"verdadero"
"falso"
```

2. Escribir otro programa si el número es divisible por 2 o es menor que 30.



The screenshot shows a code editor with a purple header bar containing a gear icon, the text "Mostrar el cuadro de herramientas", and "Espacio de trabajo:". The code is as follows:

```
1
2 function check(num1) {
3
4     if(num1%2==0 || num1<30)
5     | console.log("verdadero");
6     else
7     | console.log("falso");
8
9 }
10
11 check(15);
12 check(32);
13 check(65);
14
```

Below the code editor is a panel with a dropdown arrow, the text "Comandos para depurar", and a "Consola" tab. The "Consola" tab is active, showing the output of the code execution:

```
"verdadero"
"verdadero"
"falso"
```

3. Agregar el código para verificar si velocityY (velocidad Y) es menor que 12; solo entonces debemos aumentar la velocidad de la pelota. De lo contrario, no aumentaremos más la velocidad.

```
function brickHit(ball, brick) {  
  playSound("sound://category_hits/puzzle_game_button_04.mp3")  
  brick.remove();  
  score = score+5;  
  
  if(ball.velocityY<12)  
  { ball.velocityX *= 1.05;  
    ball.velocityY *= 1.05;  
  }  
}
```

4. Además, escribir una condición más para comprobar que velocityY no debe bajar también de -12.

```
function brickHit(ball, brick) {  
  playSound("sound://category_hits/puzzle_game_button_04.mp3")  
  brick.remove();  
  score = score+5;  
  
  if(ball.velocityY >-12 && ball.velocityY<12)  
  { ball.velocityX *= 1.05;  
    ball.velocityY *= 1.05;  
  }  
}
```

5. Escribir el código para asignar YEach velocity al grupo bricks (*ladrillos*), usando la función **setVelocityYEach()** (establecer velocidad y a cada una) para que los ladrillos se muevan hacia abajo.

```
function mousePressed()  
{  
  if(gamestate == "start")  
  {  
    gamestate = "play";  
    ball.velocityY = -7;  
    ball.velocityX = 7;  
  
    bricks.setVelocityYEach(0.2);  
  }  
}
```

6. Agregar automatización al juego, al dar la posición x de la pelota (**ball.x**) a la posición x de la paleta (**paddle.x**) para que la paleta siga a la pelota.

```
function gameplay()  
{  
  //paddle.x = World.mouseX;  
  paddle.x = ball.x; //automatizado  
  if(paddle.x < 60)  
  {  
    paddle.x = 60;  
  }  
  
  if(paddle.x > 340)  
  {  
    paddle.x = 340;  
  }  
  drawSprites();  
}
```

### ¿Qué sigue?

En la próxima clase construiremos el juego más difícil del mundo.

### Amplía tu conocimiento:

1. Guarda el siguiente enlace: será una referencia para `group.setVelocityXEach()`:  
<https://studio.code.org/docs/gamelab/setVelocityXEach/>