**PRO-C33**

Tema: Trucos y Consejos de Depuración

**Descripción:** Los alumnos aprenden consejos y trucos para minimizar errores y bugs en el código. Los alumnos, también aprenden cómo depurar los errores a través de las diferentes técnicas de depuración.

**Objetivos:** ● Aprender a minimizar errores y bugs en el código. ● Aprender diferentes técnicas para depurar códigos. ● Depurar la trayectoria del Angry Bird cuando se dispara varias veces.

Hemos hecho mu**ch**a programación en las últimas 32 clases. Hemos escrito cientos de líneas de código.

¿Han funcionado siempre como queríamos?. ¿Qué hacemos cuando los programas no funcionan como

queremos? REA Comprobamos si hay errores/bugs en el código, y los arreglamos.

Sí, encontramos los bugs/errores y los arreglamos. Sabías que… ¿Cualquier programador de computadoras, en su vida dedicará más tiempo a depurar que a escribir un nuevo código? La depuración es una parte normal de la programación. Esto se debe a que le estamos dando instrucciones a una computadora, y estamos tratando de pensar como una computadora. Sin embargo, somos humanos y no pensamos naturalmente como las computadoras.

Los buenos programadores aceptan que siempre habrá errores en su código, y DISFRUTAN del proceso de

depuración y corrección de errores en su código. La depuración es tanto un arte como una ciencia de esa manera. Es como un trabajo de detective, en el que intentas encontrar qué línea de código o patrones de pensamiento en tu mente te llevan a escribir algo que está causando un error.

Bien, describiré una serie de técnicas que los programadores usan para resolver unerror/bug en su código. ¡Entonces, juntos resolveremos un error en nuestro juego de Angry Birds!

¿Qué es lo primero que debes hacer cuando se te cuenta de que el código no está haciendo lo que tú quieres que haga?

¡Lo primero que debes hacer es NO ENTRAR EN PÁNICO! Los buenos programadores no se asustan. Se dan cuenta de que la depuración es una parte normal de la programación.

Los errores son normales. Todos los programas tendrán errores. Estarás en una posición mucho mejor para encontrar y resolver estos errores, si está tranquilo y sereno.

¡Lo segundo que debe hacer es, verificar si hay errores en el registro de la consola!

Si hay algún mensaje de error en la consola, a menudo te dicen qué sucedió y por qué ocurrió este error, o te indican en esa dirección.

A menudo, puedes corregir los errores en tu código leyendo estos mensajes de error.

Los errores más comunes son:

1. **Errores tipográficos:** Accidentalmente escribiste mal un nombre de variable o función, que la computadora no comprende.

2. **Uso Incorrecto de la función:** Usaste una función de una manera que no estaba destinada a usarse.

3. **Usar variables fuera de tu alcance:** Si estás utilizando variables fuera de tu alcance, la computadora no sabría el valor de estas variables.

A menudo, no hay mensajes de error en la consola y, sin embargo, tu programa no se comporta de la forma esperada.

O el mensaje de error no te dice todo lo que debe solucionarse. Existen algunas técnicas que te ayudarán a identificar qué está causando el error, o qué sección de tu código está provocando este error.

1. **Comentar secciones de tu código:** Puedes comentar ciertas secciones de tu código para simplificar tu código, y verificar si tu código todavía arroja errores. De esta manera, puedes limitarte a la parte del código que está causando este error.

2**. Impresión de valores de variables en la consola:** Puedes imprimir los valores de las variables críticas en la consola, para ver visualmente cómo están cambiando en el código. Si no cambian como esperabas, está sucediendo algo inesperado. Puedes manipular los valores de las variables para identificar lo que

está sucediendo.

3. **Imprime mensajes en tu código:** Puedes imprimir mensajes en tu código para comprender visualmente cómo se ejecuta el código. Por ejemplo: puedes imprimir mensajes en la consola para comprender si mientras ejecutas el código, se está ejecutando un 'bloque if' o el 'bloque else'.

También, puedes combinar los 3 pasos. Antes de ejecutar el código, uno siempre debe tener en mente lo que espera, y ver la diferencia entre lo que realmente está sucediendo.

Una última cosa, antes de comenzar nuestra misión de depuración: Escribir un código modular limpio,

formatear tu código correctamente, dar un nombre adecuado a tus funciones, objetos y variables,

agregar comentarios adecuados; todo esto toma solo un poco de tiempo,pero ahorra mucho tiempo cuando

estás depurando tu código.

Hace que tu código sea más fácil de leer/comprender y, por lo tanto, más fácil de identificar los errores.

En la última clase, no permitimos que los jugadores dispararan múltiples veces a los cerdos. Digamos que

queremos eso. Cambiemos el código para permitirle al jugador disparar a los cerdos tantas veces como quiera.

El profesor comenta y descomenta partes del texto, que permitirían al jugador disparar múltiples disparos al

objetivo.

***function mouseDragged(){ /\*if (gameState!=="launched"){\*/***

***Matter.Body.setPosition(bird.body, {x: mouseX , y: mouseY}); /}\*/ }***

El profesor ejecuta el código y dispara varias veces hacia los cerdos. - Hay múltiples trayectorias del ave en la pantalla.

¿Cuál es el error/comportamiento no deseado que tenemos en el código?. Cuando reiniciamos el ave de nuevo a la resortera. Quisiéramos que se borrara la trayectoria anterior. REA: - El ave se balancea

rápidamente cuando se reinicia. Esto no es deseado. También, provoca marcas de trayectoria adicionales.

Bien, sabemos que hay errores en nuestro código. ¿Por qué no intentas resolverlos usando los consejos y técnicas de los que acabamos de hablar?

¿Qué es lo primero que debes hacer para comenzar a depurar?: ¡Mantén la calma y no entres en pánico!

Bien. ¿Esperas algún error en la consola para este error?. ¿Por qué?: REA: No REA: El código no hace nada

inesperado. Solo el resultado es indeseable para nosotros.

¿Puedes identificar la parte del código que se encarga de trazar la trayectoria?. Dentro de Bird.js

***this.trajectory.push(position);***

***for(var i=0; i<this.trajectory.length; i++){image(this.smokeImage,this.trajectory[i][0], this.trajectory[i][1]); }***

Entonces, ¿qué harías si quisieras eliminar la trayectoria cuando se presiona la tecla de la barra

espaciadora?. Permita que el alumno experimente con su código y depure este comportamiento.

Todas las posiciones de las aves se almacenan en la propiedad array (matriz) de Bird (Ave), llamada trajectory (trayectoria). Simplemente podemos vaciar esta matriz cada vez que se presione la tecla de la barra espaciadora.

Guíe al alumno a vaciar la trayectoria del ave cuando se presione la tecla de la barra espaciadora. Simplemente puede vaciar una matriz sin asignar ningún valor entre corchetes vacíos.

***bird.trajectory = [];***

Entonces, la vieja trayectoria parece desaparecer cuando el ave se reinicia.

¿Cuál es el otro error ahora?: REA: El ave se balancea ampliamente cuando se sujeta a la resortera. ¿Por qué pasa eso? REA: Porque el ave se engancha a la resortera desde su última posición.

¿Qué podemos hacer para que no se balancee tanto? REA: Podemos restablecer la posición del ave cuando se presiona la barra espaciadora.

¡Impresionante! Vamos a hacer eso. Guíe al alumno para que utilice Matter.Body.setPosition (primero por

sí solo) para colocar al ave.

Permita que el alumno cometa errores, y luego use los consejos y técnicas de depuración cuando encuentre los errores. Dentro de sketch.js

***Matter.Body.setPosition(bird.body, {x:200, y:50});***

Esto parece haber solucionado el error.

Conclusión: Recuerda, cada código tiene errores. Incluso ahora, todavía tenemos un error. Cuando presionas la barra espaciadora mientras el ave aún se está moviendo, observa lo que sucede. REA: El alumno ve que el ave se reinicia mientras el ave

todavía se está moviendo.

¿Cómo arreglarías este error? ¡Puedes intentar corregir este error como tarea, y usar las técnicas que

aprendimos mientras lo hacías!

¡Felicidades por tu excelente trabajo! En la próxima clase, comenzaremos con un nuevo proyecto: un juego de carreras de autos multijugador, que tú y tus amigos pueden jugar en diferentes sistemas.

**RETO:** Nombre del Proyecto*: Juego del Motor Físico*

**Objetivo del Proyecto:** En la clase 33, aprendiste el concepto de depuración. En este proyecto, tendrás que

practicar y aplicar los conceptos aprendidos en la clase, y tendrás que usar un motor físico para crear tu

propio juego.

**Historia:** Si puedes recordar, la mayoría de tus juegos favoritos y adictivos eran muy realistas al jugar, o simplemente, el mundo del juego era muy similar a tu propio mundo. ¡Hoy vas a desarrollar un juego similar para deleitar a tus jugadores!.

¡¡Todo lo mejor!! Estoy muy emocionado de ver la solución de tu proyecto, y sé que lo harás realmente bien.