Conclusiones

Lore Ruelas

En conclusión, siento aprendí mucho en esta materia, y eso lo pude confirmar en que logré entender el código ejemplo que estaba en Raylib y que logramos hacer los requerimientos que nos pidió. El código en el que nos basamos de Raylib tenía varias estructuras, condiciones, llamadas a funciones, entre otros, que ahora entiendo en su perfección, y siento eso me ayudó a poder adaptar el código y agregar lo que nos pedía. Por ejemplo, no se accedía a memoria heap en el código original, nosotros en cierto tiempo lo hacemos, usamos apuntadores a arreglos, entre otras cosas que aprendimos en la materia. Siento lo que aprendimos durante el semestre nos facilitó la implementación de los requerimientos, me gustó cómo quedó el código, siento las soluciones o la forma de implementar las cosas fueron claras y sencillas. Lo que más se nos complicó siento fue hacer el MultiBall, pero decimos "reutilizar" la función de UpdateGame, y considerando que esta está dentro de un ciclo, la solución fue llamarla varias veces, a la hora de activar el poder, pero con las diferentes pelotitas. Sinceramente, fue un reto este proyecto, pero gracias a que entendí bien los temas siento no se me dificulto tanto lograr lo que se nos pedía. En cuanto al trabajo en equipo siento es importante estar todos en la misma página y tener en claro que queremos lograr; nos sirvió primero entender bien el código ejemplo de Raylib, para así aprender de Raylib, el funcionamiento del juego, y poder después ver cómo podríamos lograr nosotros los objetivos. Par finalizar, reafirme mis conocimientos, me gustaron las maneras creativas de cumplir con los requerimientos y aparte aprendí sobre la importancia del trabajo en equipo y la disponibilidad de todos para estarnos ayudando entre nosotros.

Yamile Alvares

La verdad es que este proyecto me causo bastante entusiasmo realizarlo ya que nunca había trabajado con raylib, así que fue algo nuevo para mí, tuve el conocimiento de nuevos comandos que existen dentro de esta librería que están padres, pudimos personalizar tamaños, colores, velocidades etc. La verdad es que al equipo y a mi nos fue de bastante ayuda que la base del código del juego ya existiera, porque a pesar de que teníamos la idea de cómo lo debíamos de implementar, el código que estaba en raylib nos ayudó a simplificarlo mucho más, lo primero que hicimos como equipo que fue útil fue plasmar las ideas que teníamos, como tenían que trabajar y que es lo que queríamos lograr, siento que fuimos de los primeros equipos en empezar a realizar el código y eso nos ayudó bastante ya que pues teníamos más cosas que hacer y nos ayudó a no estresarnos, no necesitamos ayuda externa de nadie más que una valoración del profe para que nos dijera que mejorar y que no, por las tardes nos queda vamos a avanzarle y cabe resaltar que cada vez que nos quedábamos lográbamos algo nuevo, los tiempos que le dedicamos fueron justos y como mencioné, a tiempo, siento que fue un reto para nosotros que nos llamaba la atención completarlo, porque era un juego padre y con mucha funcionalidad de back bien padre, otra de las cosas que decidimos como equipo fue que hasta el último lo separáramos por contenedores, y así desde un inicio tenerlo en main, tuvimos uno que otro bloqueo de ideas al final, ya que no sabíamos cómo corregir esos pequeños detalles que nos faltaban pero si se logró,

Lore diseñó variables globales que arreglaron esos pequeños tropiezos, el trabajo fue muy en equipo ya que lo que hacíamos o implementábamos estábamos nosotros 4 presentes. Aprendí bastante en este proyecto, fue padre poder lograr un resultado esperado y poder entender cada funcionalidad del código.

Agregamos elementos que aprendimos en este semestre porque el código original no usaba cosas tan avanzadas, lo más avanzado que pudimos notar fueron structs si mal no recuerdo, desde mi opinión lo más complicado que nos tocó hacer fueron los niveles, fue en lo que más nos tardamos ya que no sabíamos cómo crearlos, los poderes estuvieron rápidos excepto por las funciones de IsKey... ya que inicialmente debías de sostener la tecla para que los poderes funcionaran y utilizábamos dos if, pero pudimos simplificarlo a uno solo. También otra cosa padre que pudimos hacer es la función random en los bloques a partir del nivel 2, ya que tienen un acomodo distinto en cada juego. Para cerrar la verdad si me interesó trabajar con raylib, me gustaría seguir utilizándolo a futuro o conocer algunas otras librerías similares donde permitan utilizar gráficos más padres.

Diego Orozco

Honestamente, al usar una base concreta que ya existía para nuestro proyecto fue de mucha ayuda, pero lo que nosotros le agregamos ayudó a reforzar algunos de los temas que vimos a lo largo del semestre. Por ejemplo, en el código base que usamos, no se usaban nada de apuntadores, por lo que bien pudo haber sido un proyecto de Programación Estructurada donde lo más avanzado que vimos eran estructuras con puntito en vez de flechita (sin apuntadores).

Con los requerimientos del proyecto, usamos apuntadores para almacenar los poderes, usando un arreglo dónde el primer elemento era el poder de SuperBall, y el segundo el de MultiBall. Además de reservar memoria para estos poderes.

```
int *arrayPowers = malloc( Size: 3*sizeof(int));
arrayPowers[0] = 1;
arrayPowers[1] = 1;
```

Para arreglar uno de los bugs sobre el poder de SuperBall, personalmente tuve que volver a poner el número de golpes a uno para cada bloque, y al estar desactivado el poder se regresaba a uno y dos golpes.

```
if (IsKeyPressed( key: 'S') == 1 && arrayPowers[0] == 1) {
    Num = 1;
    //printf("HOLAA S");
    for (int i = 0; i < LINES_OF_BRICKS; i++)
    {
        for (int j = 0; j < LINES_OF_BRICKS; j++)
        {
            brick[i][j].numHits = 1;
        }
    }
}
```

Dentro de la condición del poder, se usaron dos ciclos for anidados para recorrer todos los bloques, y se le puso el. numHits a uno para que la pelota rompa sin problema todos los bloques.

```
if (Num == 1) // Checa si PowerBall está activo
{
    // Se desactiva el poder /  Num vuelve a su valor original
    Num = -1;
    arrayPowers[0] = 0;
    for (int i = 0; i < LINES_OF_BRICKS; i++)
    {
        for (int j = 0; j < LINES_OF_BRICKS; j++)
        {
            if ((i + j) % 2 == 0) // bricks oscuros mas resistentes
                brick[i][j].numHits = 1;
            else //bricks de color claro mas vulnerables
                brick[i][j].numHits = 2;
        }
    }
}
```

Ya que el poder se desactiva, entonces se regresa a la condición de uno o dos golpes para el primer nivel.

Otro aspecto que resolvimos de buena manera fue el de los niveles, usando un switchcase para ir cambiando las condiciones de cada nivel.

```
switch(NIVEL)
{
    case 1:
        break;
    case 2:
        // Se modifica el "mundo de bricks"
        brick[i][j].numHits = rand() % 2;
        break;

    case 3:
        if ((i + j) % 2 == 0) // bricks oscuros mas resistentes
            brick[i][j].numHits = 2;
        else //bricks de color claro mas vulnerables
            brick[i][j].numHits = 3;
    default:
        player.size = (Vector2){  x: screenWidth/14,  y: 20 };

}
```

El primer case es simplemente la condición de unas líneas arriba, dónde se asignaba el número de golpes. Para el nivel dos, lo cambiamos a que el orden de bloques cambie de manera aleatoria, para el nivel dos, pusimos el número de golpes a dos y tres, y a partir de ahí hasta que se pierda, se cambió el tamaño de la pelota que el jugador controla.

Como tal, creo que es un proyecto muy completo, usando una base existente añadiendo lo que nos pidió el profe y demás aspectos nuevos para aplicar lo aprendido a lo largo del semestre. Disfruté mucho de la clase, y espero que me sirva para mis próximas clases de programación.