INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE OCCIDENTE
Departamento de Electrónica, Sistemas e Informática.

ITESO
Universidad Jesuita de Guadalajara

Proyecto Final Programación con Memoria Dinámica

Slither.io by Ana & Val

Materia: Programación con Memoria Dinámica

Participantes: Ana María García Anaya
Ramírez López Valeria Guadalupe

Profesor: José Antonio Camarena Covarrubias

Fecha: 7/12/21

Repositorio de Github

ReadMe

Conclusiones

Ana María:

En lo personal Slither.io es uno de los juegos de internet que más disfruto. Cuando surgió este proyecto y una amiga sugirió que intentará programar este juego me emocione porque se me hizo una idea padre y un juego divertido, además de que disfruto mucho los retos y este proyecto prometía serlo.

Cuando Valeria y yo comenzamos a planear como programaríamos el juego y lo platicábamos con el maestro, para mí en lo personal me parecía sencillo, claramente cuando intentamos convertir esas sencillas ideas a código se transformaba en algo muchas veces más difícil de lo que inicialmente pensamos. Pero puedo decir que logramos superar casi todas las dificultades con éxito y siento que mi comprensión del lenguaje C aumento muchísimo durante este semestre.

A continuación, hablaré de algunas de las dificultades que se fueron presentando conforme íbamos haciendo el proyecto, y como logramos resolverlas. Uno de los primeros problemas que tuvimos fue también uno de los más bobos a mi parecer, sobre todo porque llevamos todo el semestre hablando de apuntadores. Cuando inicializábamos el gusano con distintas posiciones en vez de pasarle los valores de esas posiciones le estábamos pasando apuntadores a la misma posición, entonces cuando intentábamos modificar la "cabeza" del gusano en realidad modificábamos todas las posiciones del gusano y por eso parecía una pelota que seguía el mouse en vez de ser como un rastro, cuando corregimos ese error nos enfrentamos al nuevo problema de reservar espacios de memoria para cada bloque del cuerpo del gusano. Pero ahí ya habíamos logrado que el gusano se viera así:

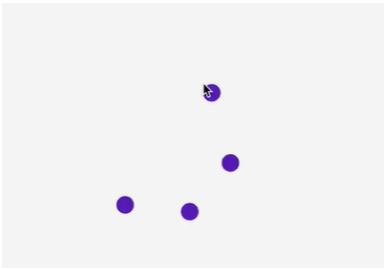

Imagen 1

Después de esa traba hicimos un gran avance en poco tiempo, y aunque nos volvimos a enfrentar a otros problemas no nos tomaron tanto tiempo como este y eso que eran problemas más complicados. Por ejemplo, ahora que nuestro gusano ya guardaba bien las posiciones y las iba recorriendo para poder asimilar el movimiento, como hacíamos para que siguiera al mouse, esto fue algo que nos tomo un poco de tiempo, pero nos dimos cuenta de que en realidad debía moverse en dirección al mouse no necesariamente seguirlo. Para lograr esto tuvimos que usar nuestros conocimientos de algebra lineal y recordar algunas de las definiciones de vectores, decidimos, después de muchísimos intentos y otras ideas fallidas, que debíamos normalizar el vector de posición del mouse y sumárselo al vector de posición de la cabeza para que poco a poco el gusano se moviera en dirección al mouse. Y esto funcionó, pero no se veía como queríamos porque no habíamos cambiado el juego a cámara2d.

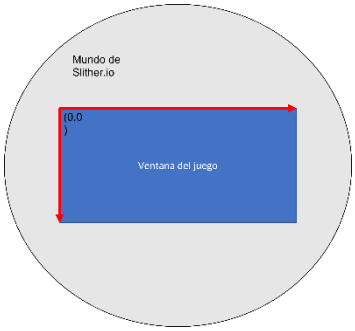

Imagen 3

La camara2d lo que hace, en pocas palabras, es apuntar a un lugar de la pantalla que tu le digas y si lo vas actualizando entonces la cámara lo sigue y siempre muestra esa coordenada en el centro. Es un concepto sencillo, pero trajo algunos problemas a nuestra implementación hasta ese momento. Para empezar nosotros nos habíamos estado manejando solamente con las coordenadas de nuestra ventana, que eran de 900x1800, el agregar la cámara crea otras coordenadas separadas de la ventana que pueden llegar tan lejos como tu quieras. De igual forma nos costó un poco de tiempo entender que algunos objetos, como la posición del mouse, se rigen por las coordenadas de la ventana, mientras que otros, como todo lo que dibujáramos, se guía por las coordenadas de la cámara2d. Entonces el algoritmo de movimiento que habíamos implementado en base a la posición del mouse, debió sufrir algunos cambios para hacer la conversión de coordenadas de la ventana a coordenadas de la cámara. Ya que conformes mueves la cámara para recorrer el mundo, el gusano tiene diferentes posiciones y coordenadas, pero las coordenadas de la ventana y por lo tanto del mouse siempre se quedarán en el mismo rango (0,1800) y (0,900) (Ver imagen 2). Es por esto que debíamos calcular el vector unitario de la posición del mouse después de hacer una conversión para que las coordenadas de la ventana se parecieran más a las de un plano cartesiano y luego sumárselo a las posiciones del gusano. Y que se moviera en la dirección que esperamos (Ver imagen 3).

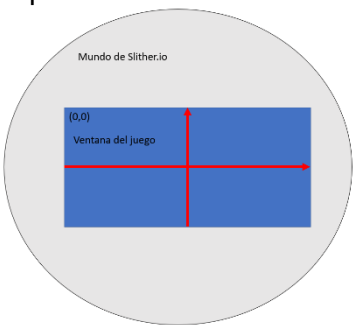

Imagen 2

Una vez que ya teníamos bien programado el movimiento del gusano llego el turno de dibujar la comida y hacer que cuando este chocará con la comida aumentará de tamaño, esto nos fue muy fácil porque ya teníamos algunas funciones para ayudarnos como addElement(), y las funciones para checar colisiones de raylib. Con esto terminado el siguiente reto fue programar a los gusanos de Inteligencia Artificial que jugarían en contra del usuario. Aquí otra vez fue de gran ayuda que hubiéramos implementado de manera correcta el movimiento del gusano porque pudimos usar la misma idea solo adaptada a la lista de gusanos artificiales.

Los comportamientos de los gusanos de I.A. o fake gusanos, como nos referimos a ellos en el código del programa, fueron más difíciles tanto de pensar como de programar. El primero que logramos hacer con cierta facilidad fue que el gusano se moviera hacia la comida, usando la función de la librería de raylib CheckCollisionCircles() e implementando algunos conceptos que aprendimos de libros de inteligencia artificial para videojuegos. Básicamente nuestro objeto debe tener una especie de escudo detector que al colisionar con comida, guarde la referencia de la posición de la comida para que se convierta en el target al que se debe mover el gusano. De igual forma el algoritmo para que los fake gusanos intentarán evadir a los otros fake gusanos funciona de manera parecida, excepto que cuando el escudo detector colisiona con otro gusano, este cambia la dirección del target en sentido opuesto para evitar la colisión. Este último aún tiene mucho por mejorar ya que a veces no evita bien a los otros gusanos o se queda un tiempo temblando en pantalla sin moverse y sin morir.

Finalmente, una de las últimas funciones que intentamos añadir a nuestro programa nos causó muchísimos problemas. Intentamos que al morir un gusano, deje un rastro de comida tras de él, tal como en el juego original. Tratamos de implementar esto de manera que cuando un gusano muere, antes de que el programa lo regrese a tamaño inicial, tomamos las posiciones del gusano y se las pasamos a nuestro arreglo estático de comida, para que ahora la comida se dibuje ahí. Esta función nos trajo problemas ya que a veces las colisiones no

terminaban en la muerte del gusano, pero si en el intento de dibujar el rastro de comida. Lo que hacía que el programa se trabara un momento. Igual al momento que chocaban dos gusanos fake, estos parecían quedarse pegados un momento y la comida se dibujaba varias veces por todo el mundo del videojuego. Nos dimos cuenta que esto era porque la condición que detectaba la colisión seguía siendo verdadera, entonces intentamos cambiarla para que una vez que se detectará la colisión ya no se volviera a realizar el mismo cambió de dirección, pero no pareció funcionar. Decidimos que lo mejor para que el juego funcionará de manera correcta era eliminar esa función, y mejor después revisarla con más calma para encontrar cual era el error. Si tiene curiosidad de como funciona solo debe descomentar las líneas 321 y 351 de la librería slitherio.c.

Como espero se haya notado después de la narración de los problemas y dificultades que tuvimos en este proyecto, aprendí muchísimo acerca de como programar videojuegos, inteligencia artificial y de la interfaz gráfica y como funciona esta. Aunque tal vez ese no era el objetivo primordial del trabajo, ayudó a que se dieran otros aprendizajes como el manejo de apuntadores, la encapsulación del programa, el uso de librerías, el uso de un VSC, etc. Este ha sido uno de los proyectos más gratificantes que he hecho y me hubiera gustado tener más tiempo para mejorar varios aspectos y aumentar su funcionalidad.

Valeria:

Sin duda alguna, elegí la carrera de Ing. en sistemas porque programar es algo que me ha gustado desde siempre, sin embargo, no fue hasta este semestre que me sentí completamente satisfecha con lo que estoy haciendo y todo eso se lo debo a esta materia. La verdad es que sentía mucho temor por la realización de este proyecto, pues es el primero que realizaría con una implementación gráfica, aunque creo que al final de todo esa era la motivación que se necesitaba para sacarlo adelante.

Fue increíble como antes de codificar Ana y yo íbamos hablando sobre como relacionaríamos cada uno de los conceptos aprendidos en el curso a nuestros gusanos. A pesar de que llevamos 3 semestres trabajando juntas, siempre sigue siendo un reto juntar nuestras ideas para obtener el mejor resultado posible. Lo principal era tener la estructura correcta de los gusanos, la cual se logró a través de listas, la cual solo se movía en línea recta, incluso cada bloque de gusano que se iba creando estaba desfasado, no estaban unidos en secuencia de puntos de ubicación, sin embargo, ya contaba con todas las características que el gusano debía de tener.

Lo siguiente a realizar era poder tener un movimiento más real del gusano, es decir que se notara como hacia curvas como normalmente lo haría un gusano, es aquí donde empezaron los problemas. Pues recuerdo que nos dimos cuenta de que las coordenadas que maneja raylib no son de la misma manera a la que estamos acostumbrados (tenía que ver con las escalas del mapa), por lo que tuvimos que hacer unas adecuaciones, creamos la función Vector2Transformacion justo como se muestra a continuación, de esta manera nuestro gusano ya se podía desplazar por la pantalla en perfectas condiciones. Los conceptos de la materia de algebra lineal resultaron indispensables para la elaboración de este proyecto, pues todo fue basado en vectores y en como a partir de sus múltiples transformaciones podríamos sacarle el mejor provecho posible sin tener que “sufrir” tanto en elaborar algo tan complicado.

```
//transformacion  de coordenadasde rahylib a un cartesiano normal
//aplicacion del speed
Vector2 Vector2Transformacion(Vector2 n){
    n.x=n.x-900;
    n.y=n.y-400;
    Vector2 resultado = {n.x/ Vector2Length(n),n.y/ Vector2Length(n)};//calcular unitario
    if(IsMouseButtonDown(MOUSE_BUTTON_LEFT)){
        return Vector2Scale(resultado,speed*2);
    }
    return Vector2Scale(resultado,speed);
}
```

La verdad es que esto me hizo tener un choque de realidad muy grande donde pude darme cuenta de que cada materia que es contemplada tiene una razón de ser y no debe de ser menos preciada. Al final de toda la computación es la matemática aplicada.

El siguiente paso fue hacer que el gusano creciera, eso fue a través de colisiones con una implementación que se encargaba de generar comida a través de todo el mundo (que no tuvo gran problema, basto con conocer las dimensiones de nuestro círculo del mundo), pues se basó en hacer crecer la lista. La función encargada de este crecimiento debía conocer el color del gusano con el cual se estaba trabajando, así como también su tamaño de radio y posiciones actuales, por lo que resulto indispensable contar con múltiples funciones “genéricas” que nos permitieran obtener esos datos cada vez que fueran necesarios. Me parece que eso fue un plus para nuestro código, nuestras funciones siempre fueron tan precisas que “reutilizar” código siempre fue una opción.

Después, realizamos lo que era de la cámara 2D, para poder desplazarnos por todo nuestro mundo. Creíamos que esta implementación sería mucho más difícil, sin embargo, solo basto

con conocer bien la librería de raylib para darnos cuenta de que era demasiado sencillo, pues esta cámara seguirá la posición que corresponde a la cabeza del gusano principal.

```
//camara 2D a partir de la posicion en la que el gusano se este moviendo
void initCamera(Camera2D *camera, List *gusano) {
    camera->target = (Vector2) {getPosicionGusano(gusano, 0).x + 20.0f, getPosicionGusano(gusano, 0).y +
20.0f};
    camera->offset = (Vector2) {screenWidth / 2.0f, screenHeight / 2.0f};
    camera->rotation = 0.0f;
    camera->zoom = 0.5f;
}
```

En ese momento ya contábamos con un juego “funcional” en cuestión de que tu como usuario ya podías realizar movimientos y crecer, lo siguiente a implementar era lo que más nos preocupaba, los otros gusanos, pues son los que deberían de ser implementados con IA. Teníamos varias ideas acerca de cómo realizaríamos los movimientos de los gusanos, la idea principal era tomar los valores que algunas gráficas matemáticas toman, sin embargo, optamos por cambiar de vector/posición cada determinada posición, con esto, la implementación de evadir otros gusanos y buscar comida se volvió más sencillo, pues depende de lo que detecte que esta cerca del será la forma de actuar el gusano.

Esta parte resulto interesante de plantear pues tuvimos que imaginar como si nuestros gusanos contaran con un cierto detector de todo lo que se encuentra cerca de el (a una distancia que esta predeterminada), y de ser así debería de buscar otra posición/vector hasta que se encontrara fuera de peligro.

Para entrar en cuestiones de gusanos que mueren, fue necesaria la eliminación de cada uno de sus bloques, aunque por cuestiones de que este es un juego “infinito” la cantidad de gusanos siempre debe de ser activa para que el juego no se volviera demasiado fácil, es por eso por lo que después de morir, otro gusano era generado de nuevo, desde un tamaño inicial predeterminado.

Otro de los retos a los cuales nos enfrentamos es que tu como jugador podías abandonar el mundo de juego sin tener consecuencia alguna, podías alejarte y el juego perdía sentido. Tratamos de muchas maneras evitar que el gusano saliera de ese mundo, ya fuera que colapsara en ese punto y no pudiera avanzar más hasta que cambiaras de posición tu mouse, o incluso que la posición del mouse cambiara automáticamente. Afortunadamente nos dimos cuenta a tiempo de que estábamos haciendo todo ese proceso muy complicado. Tuvimos serios problemas, el gusano realmente tronaba si se pasaba a un vector que correspondiera a un cuadrante negativo, se tenia que encontrar una solución rápida a ese problema.

Siempre que teníamos momentos de trabas o momentos de frustración procedíamos a jugar el slither.io original para ver como se comportaba y ver los detalles que le podíamos añadir, fue ahí cuando nos dimos cuenta de que el gusano no tendría que ni si quiera tocar el fin del mundo, pues moría al instante. Lo que teníamos con anterioridad solo se tuvo que adaptar y el problema quedo solucionado.

El mayor de nuestros problemas fue que un gusano al morir dejará rastro de comida, a pesar de que planteamos diversas implementaciones no obteníamos lo que verdaderamente estábamos buscando. Pues parecía que cuando el gusano colisionaba este lo hacia por mucho tiempo lo cual impedía una eliminación optima de ese cuerpo del gusano, en cuestiones que su cuerpo tardaba mucho tiempo en ser eliminado.

Algo que me hubiera parecido increíble lograr es que tuviéramos un main.c realmente limpio que solo llamará a una función para jugar, sin embargo, me parece que la encapsulación de código que logramos fue bastante buena.

Para terminar, quiero decir que fue un trabajo duro, siempre trabajamos con mucha disposición y el resultado fue mucho mejor de lo que me pude llegar a imaginar la primera vez que nos planteamos la idea de este proyecto. Claro está que aún queda mucho por mejorar, sobre todo a pesar de que los gusanos llegan a ser complicados de “matar” lo que más nos hubiera gustado es que sus movimientos se vieran aun con muchísima más naturalidad. Esta materia de verdad me hizo tomar una perspectiva diferente sobre lo poderoso que es C, yo la verdad nunca creí hacer la gran cosa (por lo menos cuando inicio el semestre) y eso es lo padre, que después, una sintaxis tan compleja como lo puede ser esta comparada con otros lenguajes lo tomas con tanta naturalidad ni si quiera es difícil ordenar y llevar a cabo tus ideas. Muchísimas gracias por todo Josean, sin duda alguna mi mejor materia cursada hasta el momento en el ITESO.