UNIVERSIDAD DE LAS FUERZAS ARMADAS

Computación Gráfica

Departamento de Ciencias de la Computación

Videojuego 2D

Estudiantes:
Erick Andrade, Daniel Armas, Isaac Escobar, Josué Merino

Docente:
Ing. Darío Morales

27 de agosto de 2024

Introducción

El presente informe detalla el desarrollo de un videojuego 2D creado en el entorno de Windows Forms utilizando el lenguaje de programación C. Este proyecto ha sido diseñado para proporcionar una experiencia de juego inmersiva, incorporando múltiples aspectos del diseño gráfico computacional, tales como sprites, animaciones, detección de colisiones y manejo de eventos. A lo largo del informe se analizarán las estructuras principales del código, los formularios utilizados, y las clases que conforman la base del juego, ofreciendo una comprensión profunda de su funcionamiento.

Objetivos

- Diseñar un videojuego 2D que ofrezca una experiencia de juego envolvente, integrando varios aspectos de diseño gráfico computacional, y que sea ejecutable en un entorno de Windows Forms utilizando C.
- Desarrollar un concepto claro del juego, incluyendo una historia o contexto, objetivos definidos, y mecánicas de juego que proporcionen una progresión lógica y un desafío creciente a medida que el jugador avanza en los niveles.
- Crear gráficos y animaciones atractivas, utilizando sprites para la representación de personajes, objetos y fondos, junto con la implementación de animaciones dinámicas para personajes y objetos, mejorando la inmersión del jugador.
- Aplicar técnicas avanzadas de computación gráfica, tales como transformaciones 2D (traslaciones, rotaciones, escalados), detección de colisiones utilizando bounding boxes o a nivel de píxel, y manejo de capas y Z-order para asegurar la correcta representación visual de los elementos en la pantalla.
- Implementar controles de usuario responsivos que permitan una jugabilidad fluida y natural, acompañados de sonidos y efectos visuales que realcen la experiencia del jugador.
- Desarrollar un sistema de puntuación y manejo de vidas o niveles que motive al jugador a continuar jugando, proporcionando un incentivo a través del progreso y la superación de desafíos.
- Asegurar una alta calidad del código, estructurándolo de manera modular y reutilizable, con comentarios claros que expliquen la lógica detrás de las principales funciones y procesos, y garantizando un manejo eficiente de los recursos del sistema, como memoria y procesamiento.

Desarrollo

Estructura

El proyecto está organizado en múltiples archivos, cada uno de los cuales desempeña un rol específico en la ejecución del juego. Entre los archivos principales, se encuentran:

- Base.cs y Base.Designer.cs: Define una clase base que establece la estructura común que comparten diferentes partes del juego.
- Game.cs y Game.Designer.cs: Contiene la lógica principal del juego, manejando la inicialización, actualización de estados y renderización de gráficos.
- DrawHandler.cs: Se encarga de la lógica de dibujo en pantalla, gestionando la representación visual de sprites y otros elementos gráficos.
- GameTime.cs: Administra la sincronización del juego, asegurando que las actualizaciones y animaciones ocurran de manera fluida y consistente.
- Keyboard.cs: Maneja la entrada del usuario, detectando y respondiendo a las acciones realizadas a través del teclado.
- Sprite.cs: Define la estructura de los sprites, que son las imágenes gráficas utilizadas para representar los personajes y objetos en el juego.

Código

Base.cs

El archivo Base.cs define una clase base que puede ser utilizada por otros formularios o componentes del juego. Este archivo establece la estructura fundamental sobre la que se construye el resto del juego, proporcionando métodos y propiedades comunes.

- Inicialización: Este archivo probablemente incluye un constructor que establece las propiedades iniciales de la clase base, como el tamaño de la ventana, la configuración del contexto gráfico, y la carga de recursos iniciales como imágenes y sonidos.
- Métodos Comunes: Métodos como Initialize() y LoadContent() podrían ser responsables de la configuración de elementos compartidos entre diferentes partes del juego, como la inicialización de texturas y la asignación de recursos.

Game.cs

El archivo Game.cs es el núcleo del juego. Aquí se encuentra el bucle principal que gestiona la ejecución continua del juego, incluyendo la actualización de los estados de

los objetos y su representación visual en pantalla.

- Bucle de Juego: Este archivo contiene el bucle principal del juego, que incluye métodos como Update() y Draw(). El método Update() se encarga de manejar la lógica del juego, como el movimiento de los personajes, la detección de colisiones y la interacción con el entorno. Por otro lado, Draw() es responsable de la renderización de gráficos, dibujando los sprites en la pantalla en sus posiciones actualizadas.
- Manejo de Eventos: Game.cs también maneja eventos clave, como las entradas del teclado y el mouse, que permiten al jugador interactuar con el juego. Por ejemplo, las teclas de movimiento o de acción se capturan y procesan aquí para actualizar la posición del jugador o realizar alguna acción específica.

DrawHandler.cs

El archivo DrawHandler.cs gestiona la lógica de dibujo de los objetos del juego. Este archivo contiene métodos especializados para dibujar sprites y otros elementos gráficos en la pantalla.

- Métodos de Dibujo: Este archivo probablemente incluye métodos como Draw(Sprite sprite), que toma un objeto Sprite y lo dibuja en la posición correcta en la pantalla. Estos métodos aseguran que los elementos gráficos se rendericen en el orden correcto y con las transformaciones adecuadas, como rotaciones o escalados.
- Optimización del Renderizado: Además, el archivo podría incluir técnicas de optimización para reducir la carga en el procesador gráfico, como el uso eficiente de buffers o la minimización de llamadas a funciones de dibujo.

GameTime.cs

El archivo GameTime.cs es crucial para la sincronización del juego. Este archivo maneja el tiempo transcurrido entre cada fotograma, lo cual es esencial para que las animaciones y las actualizaciones de estado sean fluidas y consistentes.

- Sincronización del Juego: GameTime probablemente calcula el tiempo transcurrido desde el último fotograma y ajusta la lógica del juego en consecuencia, permitiendo que los movimientos y las animaciones sean independientes del rendimiento del hardware.
- Manejo de Retrasos: También es probable que este archivo incluya funciones para manejar retrasos o pausas en el juego, asegurando que la experiencia del usuario sea lo más fluida posible.

Keyboard.cs

El archivo Keyboard.cs maneja las entradas del usuario a través del teclado, detectando qué teclas están siendo presionadas y ejecutando la lógica correspondiente.

- Detección de Entrada: Este archivo incluye métodos como IsKeyDown(Keys key) que verifica si una tecla específica está siendo presionada en un momento dado. Esto permite al juego responder a la entrada del usuario, ya sea moviendo un personaje, saltando, disparando, o realizando cualquier otra acción.
- Mapeo de Teclas: Además, se maneja el mapeo de teclas a funciones específicas del juego, lo que permite configurar los controles del juego de manera intuitiva para el usuario.

Sprite.cs

Sprite.cs El archivo Sprite.cs define una clase Sprite, que es la unidad básica de gráficos en el juego. Los sprites representan personajes, objetos y otros elementos visuales dentro del juego.

- Propiedades del Sprite: La clase Sprite incluye propiedades como Position, Texture, Velocity, y Rotation. Estas propiedades definen la ubicación, apariencia y comportamiento de los sprites en el juego.
- Métodos de Movimiento y Dibujo: La clase Sprite probablemente incluye métodos para actualizar su posición y dibujar el sprite en la pantalla. Estos métodos permiten a los sprites moverse por la pantalla, interactuar con otros objetos y responder a eventos del juego.

Resultados

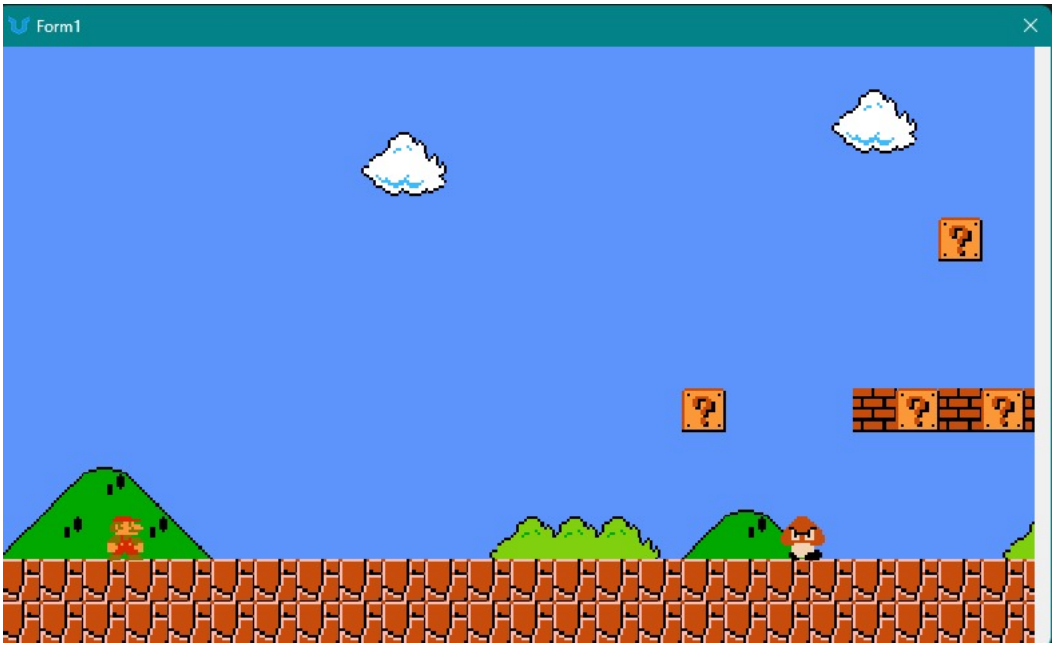

Figura 1: Resultado

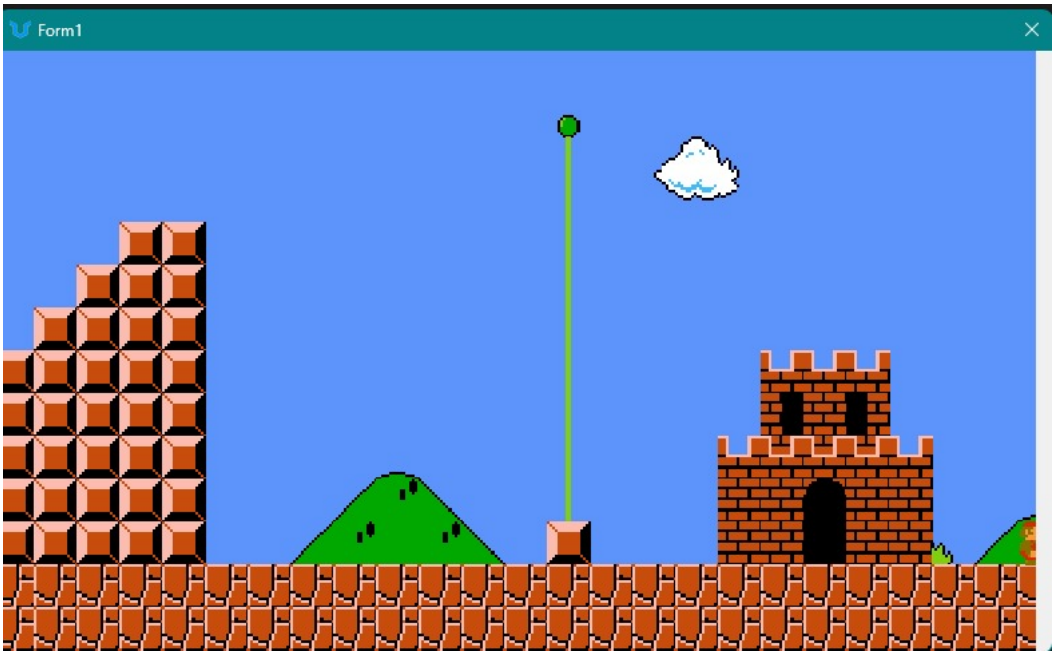

Figura 2: Resultado

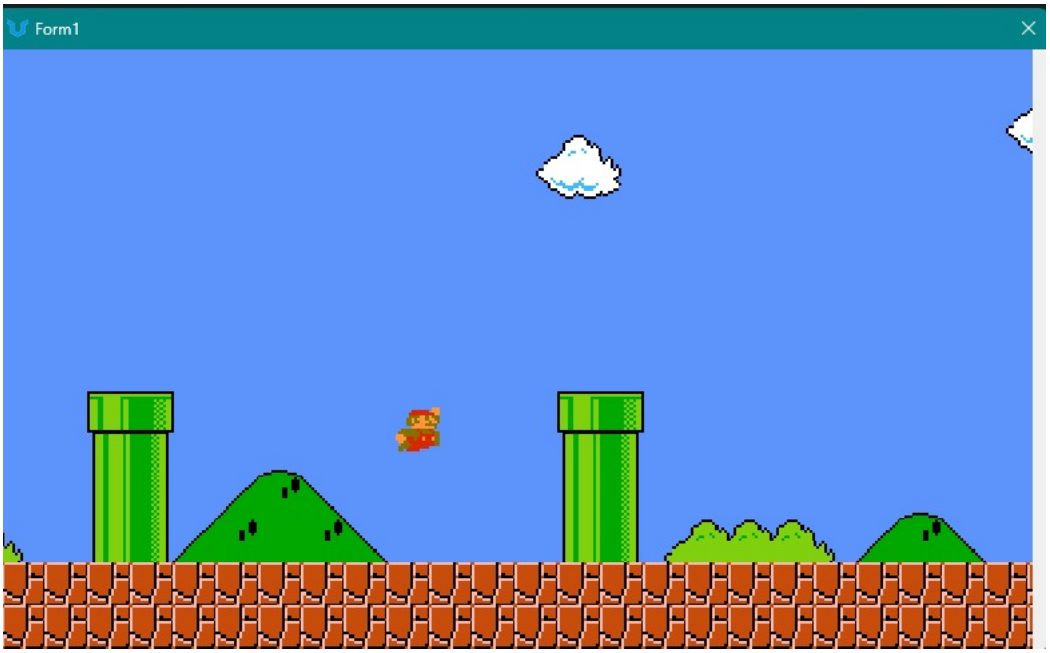

Figura 3: Resultado

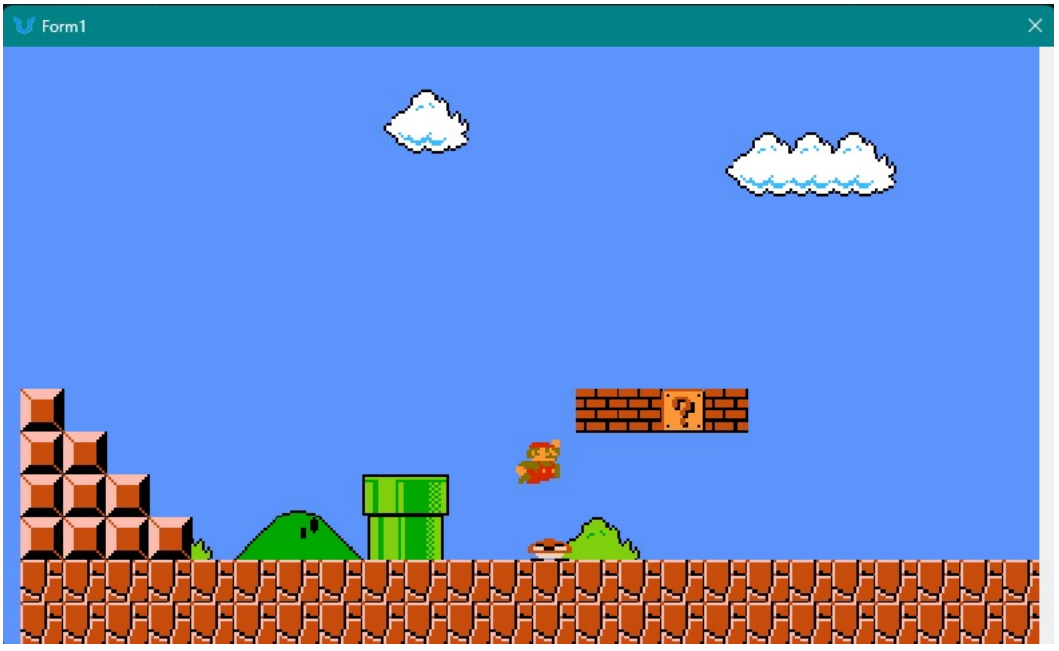

Figura 4: Resultado

Conclusiones

El desarrollo de este videojuego 2D en C# utilizando Windows Forms ha requerido la integración de diversas técnicas de programación y diseño gráfico computacional. Cada archivo y clase en el proyecto desempeña un papel vital en la creación de una experiencia de juego envolvente y fluida. Desde la gestión del bucle de juego hasta el manejo de gráficos y eventos, el código está estructurado para ser modular, reutilizable y eficiente, cumpliendo con los requisitos establecidos.

Este proyecto no solo cumple con los objetivos de diseño gráfico y programación, sino que también proporciona una base sólida para futuras expansiones o mejoras en el juego, demostrando una comprensión profunda de los principios de desarrollo de videojuegos.