INFORME DE EVALUACIÓN DE DESEMPEÑO

CURSO DE ALGORITMOS – CC215
Carrera de Ingeniería de Software

Sección: SV21

Alumno:

U202411774 Fabrizio Alonso Santi Guerrero

2024-2

1. CONTENIDO

2. RESUMEN

En el curso de Algoritmos (CC215), como trabajo final, se ha desarrollado un videojuego con temática ambiental. Para su creación, se empleó el paradigma de Programación Orientada a Objetos (POO), Windows *Forms* y el lenguaje C++. Este proyecto tiene como objetivo demostrar los conocimientos adquiridos a lo largo del curso y generar conciencia sobre un tema crucial en la actualidad: la administración de recursos y el cuidado del medio ambiente. En el videojuego, los jugadores enfrentan el reto de decidir estratégicamente dónde colocar los recursos mientras evitan los contaminantes, reflejando así la importancia de tomar decisiones responsables en beneficio del entorno.

3. OBJETIVO DEL ESTUDIANTE

La problemática consistía en desarrollar un juego en el que el jugador pudiera ganar al reforestar el 70% de una zona. Además, el juego debía gestionar archivos, incluir aliados y enemigos, y ofrecer una jugabilidad entretenida para el usuario. Para abordar este desafío, se realizó un trabajo en equipo durante 4 semanas, en las que cada semana nos organizábamos. También, aplicaron buenas prácticas, como la legibilidad y la correcta estructuración del código, lo que facilitó la implementación de los requisitos. Asimismo, se experimentó con diversas funcionalidades de la herramienta *Windows Forms,* como el control del cursor del *mouse* y los eventos de entrada y salida para la gestión del guardado de archivos.

4. DISEÑA UNA SOLUCIÓN BASADA EN COMPUTACIÓN PARA CUMPLIR CON EL CONJUNTO DE REQUERIMIENTOS EN EL CONTEXTO DE SISTEMAS DE INFORMACIÓN

El programa fue diseñado con componentes de bajo nivel, lo que facilitó implementaciones rápidas a través de GitHub y minimizó los conflictos de fusión. Esto también permitió que, en la clase encargada del guardado de archivos (ver Anexo 1), se pudiera editar directamente la gestión de los datos del juego. De este modo, bastaba con modificar algunos valores para que el juego se adaptara automáticamente a los cambios. Por ejemplo, el puntaje y el porcentaje no necesitaban ser almacenados, ya que la clase controladora ajustaba esta información en tiempo real utilizando los demás datos disponibles.

Para asegurar la calidad del programa, se realizaron pruebas exhaustivas para garantizar una dificultad equilibrada y una experiencia agradable. Además, se detallaron cuidadosamente los datos necesarios para el jugador, permitiendo que este se enfoque únicamente en jugar.

El videojuego incorpora conceptos matemáticos, como la rotación del jugador. Para ello, se aplicaron teoría de vectores y matrices (transformaciones lineales), de modo que el ángulo de dirección se definiera mediante la posición del cursor del mouse (ver Anexo 2). Esto permitió implementar el efecto de rotación del *sprite* de manera precisa.

5. IMPLEMENTA UNA SOLUCIÓN BASADA EN COMPUTACIÓN PARA CUMPLIR CON

EL CONJUNTO DE REQUERIMIENTOS EN EL CONTEXTO DE SISTEMAS DE INFORMACIÓN

El trabajo del curso debía incluir elementos esenciales que no podían ser omitidos, por lo que fueron priorizados en el desarrollo, como el ranking y formato de los archivos de texto. Además, ciertas clases estaban estrechamente relacionadas y, al trabajar en conjunto, completaban funcionalidades clave. Estos elementos esenciales se reflejan en el informe entregado, especialmente en el formato de los archivos de texto y la inclusión de un sistema de ranking.

Las clases más interconectadas fueron *GestorArchivosBinario*, *GestorArchivosGuardado* y Nivel (ver Anexo 3). Estas tres colaboran en la función *Nivel_Jugar* de la clase Nivel, permitiendo que, según el nivel jugado, se registre en el ranking correspondiente, se detecte si es necesario guardar la partida o se confirme una victoria.

Asimismo, cabe destacar que la clase *GestionadorArchivo* (ver Anexo 4) actúa como clase base para las demás clases gestionadoras. Esto se debe a que tareas como el acceso a rutas y la creación de carpetas eran comunes. Esta estructura basada en herencia simplificó la implementación de funciones clave, como la pausa del juego, el guardado y carga de partidas, y el desbloqueo de niveles.

6. EVALÚA UNA SOLUCIÓN BASADA EN COMPUTACIÓN PARA CUMPLIR CON EL CONJUNTO DE REQUERIMIENTOS EN EL CONTEXTO DE SISTEMAS DE INFORMACIÓN

Este programa demuestra claramente el cumplimiento de la competencia ABET, ya que aborda la problemática planteada, mostrándolo en su diseño y mecánica.

Para garantizar la correcta implementación de este proyecto, se diseñó un diagrama de clases detallado que sirvió como guía estructural y bocetos para desarrollar una meta como equipo, de cómo debía ser el resultado. Este diagrama permitió definir las relaciones entre las clases, e integrar conceptos clave como la reutilización de código y la lógica orientada a objetos, asegurando que cada parte del programa contribuyera al desarrollo de la competencia ABET.

Este enfoque no solo destaca el dominio técnico, sino también la capacidad de aplicar conocimientos de ingeniería, trabajar en equipo y abarcar problemas reales.

7. CONCLUSIONES

- Un desarrollo detallado y bien estructurado a bajo nivel permite que las implementaciones entre los integrantes se realicen de manera eficiente y fluida. Además, el uso de herramientas especializadas brinda un control más preciso sobre el flujo del código, optimizando los resultados del proyecto.
- La comprensión profunda de los temas y métodos empleados es esencial, ya que no solo facilita la escritura de un código más eficiente, sino que también permite realizar modificaciones significativas con menor esfuerzo. Solo entendiendo a fondo las herramientas utilizadas es posible aprovechar todo su potencial y adaptarlas a las necesidades del proyecto.

- Finalmente, la organización y la capacidad para tomar decisiones estratégicas son fundamentales para el éxito del desarrollo. Estas habilidades permiten superar obstáculos, como los problemas que puedan enfrentar algunos integrantes, reduciendo su impacto y asegurando el cumplimiento de los objetivos del equipo.

8. RECOMENDACIONES

- **Diversidad de habilidades:** Es recomendable que cada integrante del grupo cuente con una característica o habilidad que lo diferencie, esto es para que, en cualquier requerimiento de algún programa, el grupo pueda contar con un integrante que pueda abarcar de mejor manera lo solicitado.
- **Compromiso equilibrado:** Se recomienda que cada integrante muestre un nivel de compromiso similar, ya que, esto es un factor que influye en la organización y desarrollo del proyecto. Para evitar problemas en el ritmo de trabajo.
- **Objetivo común definido:** Como equipo, es importante que el objetivo este definido para todos, y que todos puedan percibirlo, porque esto ayuda a una menor cantidad de conflictos.

9. ANEXOS

Anexo 1

En la clase *GestorArchivosGuardado* no es necesario obtener el valor de puntaje, ni porcentaje, ya que, con el tiempo se puede determinar estos valores.

```
public ref class GestorArchivosGuardado : public GestorArchivos {

public:
    // Cargar el ranking de un nivel especifico

    bool verificarNivel(int nivel) {
        ifstream nomArch;
        nomArch.open(obtenerRutaArchivo(nivel, "guardado.txt"), ios::in);
        if (nomArch.is_open()) {
            nomArch.close();
            return true;
        }
        return false;
    }

    void CargarNivel(int nivel, int& tiempo, int& puntaje, Controladora^ controladora) {
        ifstream nomArch;
        nomArch.open(obtenerRutaArchivo(nivel, "guardado.txt"), ios::in);

   nomArch >> comentario >> tiempo;


   nomArch.close();
   remove(obtenerRutaArchivo(nivel, "guardado.txt").c_str());
```

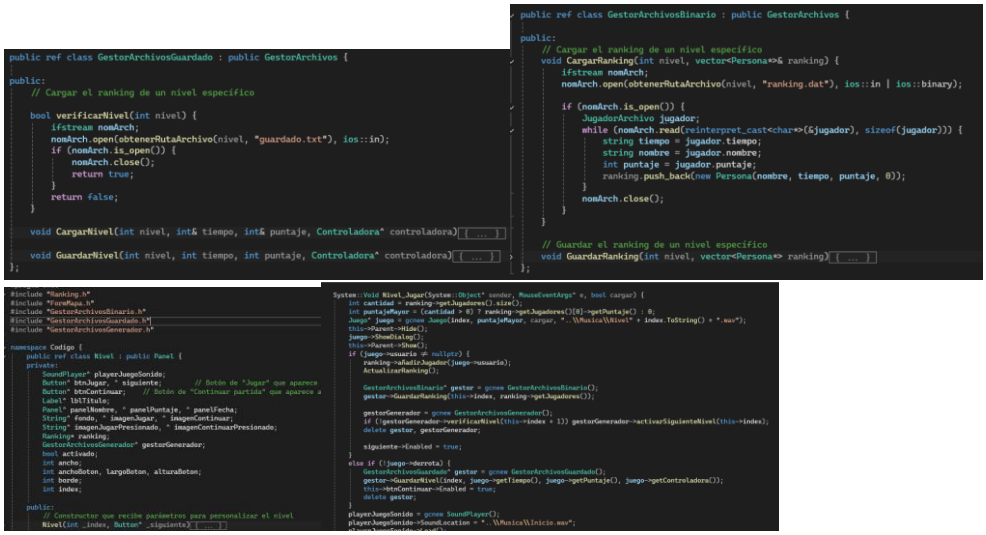
Anexo 2

Detección del evento de mover mouse, cálculo del ángulo y transformación del *sprite*.

```
private: System::Void Juego_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {
    if ((e->Y < 620 && e->Y > 144)) && !pausa  /* && (e->X < 814 && e->X > 14) */) {
        objControladora->GetObjJugador()->calcularAngulo((e->X) - objControladora->GetObjJugador()->GetArea()->GetCentroX
    }
void calcularAngulo(float deltaX, float deltaY) {
    angulo = Math::Atan2(deltaY, deltaX);
}
void mostrar(BufferedGraphics^ buffer, Bitmap^ bmpJugador, Bitmap^ bmpCorazon, Graphics^ g) {
    buffer->Graphics->TranslateTransform(area->GetX() + area->GetAncho() / 2, area->GetY() + area->GetAlto() / 2);
    // Rotar el sprite según el ángulo calculado (en grados)
    buffer->Graphics->RotateTransform(angulo * 180 / Math::PI);
    // Dibujar la imagen centrada en el nuevo origen
    buffer->Graphics->DrawImage(bmpJugador, -area->GetAncho() / 2, -area->GetAlto() / 2, area->GetAncho(), area->GetAlto());
    // Restablecer las transformaciones para que no afecten a otros dibujos
    buffer->Graphics->ResetTransform();
    vidas->Mostrar(buffer, bmpCorazon, g);
    //Entidad::mostrar(buffer, bmp, g);
}
```

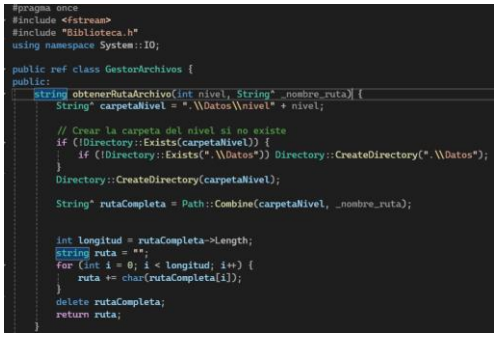
Anexo 3

Clases *GestorArchivosGuardado*, *GestorArchivosBinario*, y *Nivel*. Función *Nivel_Jugar().*



Anexo 4

Clase padre *GestorArchivos*



10. BIBLIOGRAFÍA

Brian (Able Opus). (21 de Agosto de 2010). *Play Sounds in Windows Forms App (C# .NET)* [Video]. YouTube. https://www.youtube.com/watch?v=qOh4ooHg1UU

Thảo Meo TV. (21 de Mayo de 2020). [C#] *Change cursor from image winform* [Video]. YouTube. https://www.youtube.com/watch?v=4qmOB6GJqgk

Profesores UPC – Libro digital – Independencia lineal bases y generadores