ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA EN SISTEMAS

Informe De Desarrollo De Software Del Proyecto De Videojuego *"Bad Ice Cream"*

INTEGRANTES:

Samira Fernanda Arizaga Guallpa

Alison Lizeth Betancout Herrera

Alisson Viviana Espín Ortega

Fernando Eliceo Huilca Villagomez

Juan Mateo Quisilema Flores

DOCENTE:

Carlos Eduardo Anchundia Valencia

QUITO – 2024

ViKunn/ProyectoFinalProgramacion at BadIceCreamConInterfaz (github.com)

INDICE DE CONTENIDO

1. INTRODUCCIÓN

2. OBJETIVOS

3. MARCO TEÓRICO

3.1. Generalidades de Bad Ice Cream

"Bad Ice Cream", desarrollado por Nitrome, es un juego de arcade y acción que se ha ganado gran popularidad. Fue lanzado por primera vez en Flash el 10 de diciembre de 2010. Su dinámica, aunque sencilla, resulta adictiva. Esto lo convierte en un candidato ideal para ser recreado en un entorno de aprendizaje enfocado en la programación orientada a objetos, permitiendo así profundizar en los conceptos fundamentales de este paradigma de desarrollo de software.

En el juego, los jugadores asumen el papel de helados, es como un "Pacman" bajo cero con gráficos geniales y deliciosos postres. Viaja a través de laberintos y acumula puntos al recolectar frutas. El objetivo para lograr la victoria es eliminar toda la fruta sin que te atrapen.

Ilustración 2. Imagen inicial al empezar a jugar "Bad Ice Cream"

Ilustración 1. Personajes de "Bad Ice Cream"

Los helados, personajes a elegir, son de tres sabores: chocolate, fresa y vainilla. Cada personaje representa al usuario y puede lazar hielo hacia la dirección en la que esté mirando. Esta habilidad es doblemente útil, ya que no solo permite a los jugadores crear barreras de hielo, sino que también pueden destruir estas barreras lanzando hielo sobre ellas. En el código del proyecto, estos personajes son identificados como Player.

Además de los jugadores, el juego también presenta una variedad de enemigos. Los enemigos pretenden impedir que los jugadores completen sus objetivos y avanzar al siguiente nivel. El objetivo principal del juego es recolectar todas las frutas en cada nivel mientras se evita a los enemigos. Los jugadores pueden usar su habilidad para lanzar hielo para crear barreras y mantener a raya a los enemigos mientras recolectan frutas. Sin embargo, deben tener cuidado, ya que algunos enemigos pueden romper estas barreras de hielo.

Ilustración 3. Nivel 2 de "Bad Ice Cream"

3.2. Patrón de Diseño

El patrón de diseño implementado en el desarrollo del videojuego es el MVC (Modelo-Vista-Controlador). Este patrón se ha elegido debido a su capacidad para reducir la complejidad durante el desarrollo, al ofrecer una estructura clara y modular que separa las distintas responsabilidades del sistema:

Modelo: Contiene toda la lógica y los datos del juego. Es responsable de manejar la manipulación de datos, la lógica del juego y cualquier cálculo o procesamiento necesario. En esencia, representa el estado subyacente del juego.

Vista: Se encarga de la presentación de la información al jugador a través de la interfaz gráfica de usuario (GUI). Es responsable de mostrar los elementos visuales del juego, como personajes, escenarios, menús y demás elementos de la interfaz.

Controlador: Actúa como intermediario entre el Modelo y la Vista. Se encarga de recibir las entradas del usuario, procesarlas y actualizar el Modelo según corresponda. Además, se encarga de actualizar la Vista para reflejar los cambios en el Modelo.

La separación clara de responsabilidades que ofrece el patrón MVC facilita el mantenimiento, la escalabilidad y la reutilización del código. Al dividir el juego en estas tres partes distintas, se hace más fácil realizar cambios en una parte del sistema sin afectar a las otras, lo que permite un desarrollo más ágil y eficiente.

A continuación, se presenta una gráfica que ilustra cómo se relacionan entre sí el Modelo, la Vista y el Controlador en el contexto del desarrollo del videojuego:

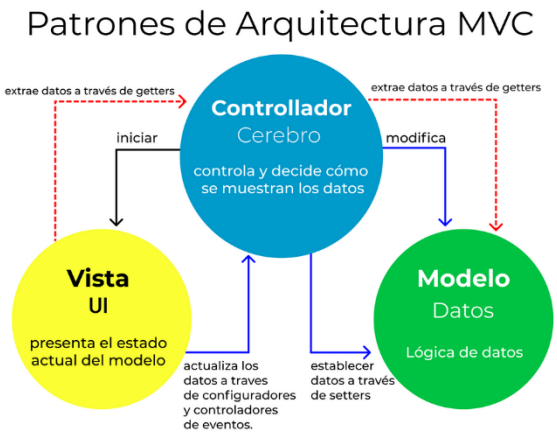


Ilustración 3. Modelo Vista - Controlador

Esta representación visual ayuda a comprender mejor la dinámica y las interacciones entre las diferentes partes del sistema, lo que facilita la implementación y el mantenimiento del videojuego a lo largo del ciclo de desarrollo.

3.3. Implementación de Interfaz gráfica

Para la implementación de la interfaz gráfica en el proyecto, primero se consolidó el código funcional en consola, lo cual representaba la lógica del juego y sus reglas. Tras establecer este paso, se integró la interfaz gráfica.

Se optó por utilizar Java Swing para la creación de la interfaz gráfica. En particular, se empleó JFrame como el contenedor principal para la ventana de la aplicación. JFrame ofrece una ventana con bordes, botones de cierre, minimizar y maximizar, y se usa en aplicaciones Java para crear interfaces de usuario.

Además, se utilizaron varios componentes Swing para la implementación de diferentes elementos de la interfaz, como menús y botones. JLayeredPane se utilizó para superponer imágenes y crear capas en la interfaz. JPanel se utilizó para organizar y manejar los diferentes menús y botones de la aplicación. JButton se usó para la creación de botones interactivos en la interfaz.

Para manejar las interacciones de usuario, se implementó ActionListener en los botones. ActionListener es una interfaz en Java que permite responder a eventos de acción, como hacer clic en un botón, y ejecutar el código correspondiente.

Respecto a la creación de imágenes, se recurrió a documentación del juego Bad Ice Cream para obtener inspiración y referencia en el diseño de los elementos gráficos. Además, se empleó inteligencia artificial para la generación de algunas imágenes, lo que proporcionó un enfoque innovador y personalizado en la creación de los elementos visuales del juego.

La implementación de la interfaz gráfica se realizó con Java Swing, con JFrame como contenedor principal y diversos componentes Swing para crear menús, botones e interacciones de usuario. Se aprovechó la documentación de juegos existentes y se exploraron enfoques creativos, como el uso de inteligencia artificial, para la creación de elementos gráficos personalizados.

3.4. Generación de animaciones

En la generación de animaciones, nos propusimos asegurar que cada dirección de movimiento del jugador tuviera al menos una imagen asociada. Esto nos permitió simular el

movimiento de manera más realista y fluida, proporcionando una experiencia de juego más inmersiva para el jugador. Para lograr esto, creamos una variedad de imágenes para cada dirección de movimiento, ajustando cada fotograma para reflejar los diferentes movimientos del personaje.

3.5. Modalidad de Colisión en Juegos 2D

La modalidad de colisión en juegos 2D, como en Bad Ice Cream de Nitrome, es un aspecto crucial para la jugabilidad y la experiencia del jugador.

En Bad Ice Cream, al igual que en muchos juegos 2D, la modalidad de colisión se basa típicamente en la detección de colisiones por píxeles o por formas geométricas simples, como cajas delimitadoras (bounding boxes) o polígonos. Estas técnicas permiten determinar cuándo y dónde los objetos del juego, como el jugador y los obstáculos, entran en contacto entre sí.

En particular, en Bad Ice Cream, los personajes interactúan con los bloques de hielo y otros elementos del entorno, así como con enemigos y objetos especiales. La detección de colisiones se utiliza para controlar aspectos como la capacidad de los personajes para moverse a través del entorno, recoger objetos y evitar peligros.

4. DESCRIPCION DEL PROCESO DE DESARROLLO

4.1. Fase I. Identificación

4.2. Fase II. Análisis

4.3. Fase III. Diseño

4.3.1. Diagrama UML

4.3.2. Personajes, Enemigos, Mapas y Menús

Dentro del videojuego, se han implementado dos enemigos principales con comportamientos distintos. El primero de ellos es el troll, que se desplaza en línea recta hasta encontrar un obstáculo. En ese caso, el troll decidirá cambiar su dirección de movimiento y continuará avanzando en una nueva dirección. Este cambio de dirección se realiza de manera

horaria, lo que significa que el troll girará en sentido de las agujas del reloj para evitar el obstáculo y continuar su trayectoria.

Ilustración 4. Troll: Primer enemigo del juego

El segundo enemigo es conocido como la Blue Cow, y su comportamiento es diferente al del troll. Este enemigo persigue al personaje principal o jugador a lo largo del nivel. La Blue Cow está constantemente buscando al jugador y ajusta su dirección de movimiento para seguirlo en todo momento. Su objetivo es acercarse al jugador y atacarlo, lo que agrega un desafío adicional para el jugador mientras intenta avanzar en el juego. Combinar estos dos enemigos con comportamientos distintos añade variedad y desafío al juego, requiriendo estrategias para enfrentar a cada uno.

Ilustración 5. BlueCow: Segundo Enemigo

Dentro del juego, se han incluido objetos que el jugador debe recolectar para obtener puntos y avanzar en la partida. Estos objetos proporcionan una manera adicional de desafiar al jugador y recompensarlo por su habilidad para recolectarlos. Entre los objetos disponibles se encuentran los plátanos, que otorgan un puntaje de 30 puntos cada uno al ser recolectados. Los plátanos representan una fuente de puntos inicial y son relativamente fáciles de encontrar en el nivel.

Además de los plátanos, se han incorporado las uvas como otro tipo de objeto recolectable. Las uvas proporcionan un puntaje mayor que los plátanos, otorgando 40 puntos cada una al ser recolectadas por el jugador. Esto añade un incentivo adicional para que el jugador busque activamente las uvas durante su recorrido por el nivel.

Ilustración 6. Frutas del Juego: Uvas y Bananas

Se han implementado diversos obstáculos que agregan complejidad y desafíos al recorrido del jugador. Uno de los principales obstáculos es el hielo, que se destaca por su capacidad única de romperse y recrearse cuando el jugador activa ciertos poderes. Esta característica del hielo añade una dinámica interesante al juego, ya que el jugador debe tener en cuenta cómo utilizar estratégicamente estos poderes para atravesar obstáculos y superar desafíos en el nivel.

Además del hielo, se ha incluido otro obstáculo en forma de bloque sólido e inmutable. Este tipo de bloque no puede ser roto ni recreado por el jugador y representa un desafío adicional en el recorrido. Los bloques sólidos obstaculizan el avance del jugador y requieren que este encuentre una manera alternativa de sortearlos, ya sea evitándolos o utilizando otros elementos.

Ilustración 7. Bloques del Mapa: Nieve, Metal y Hielo

Se ha agregado, además, un nivel personalizado al juego, titulado "Politécnico", con el objetivo de aportar un elemento distintivo y único al juego que estamos clonando. En este nivel, se han introducido personajes inspirados en figuras reconocidas dentro de la comunidad educativa, como el profesor de Programación Orientada a Objetos, Carlos Anchundia, y el profesor de Electrónica, Patricio Zambrano. Ambos personajes asumen el papel de "enemigos" dentro del juego, pero es importante destacar que este rol es puramente ficticio y no busca ofender a ninguna persona en particular. Los personajes del nivel "Politécnico" presentan comportamientos similares a los enemigos previamente mencionados, especialmente en cuanto a sus movimientos y acciones dentro del juego. Su inclusión como enemigos tiene como único propósito agregar un toque de diversión y familiaridad al juego, y honrar y reconocer la contribución de estas figuras en el ámbito educativo.

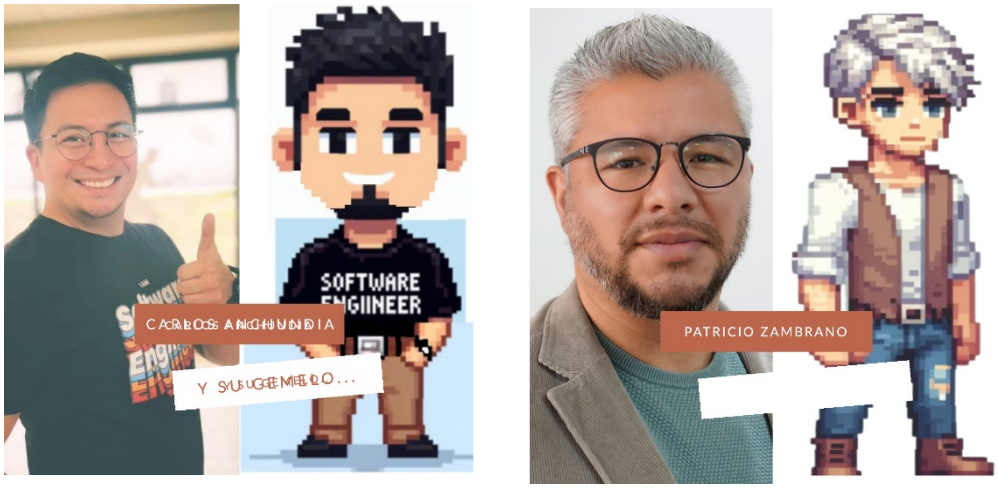


Ilustración 8. Enemigos de Nivel Politécnico: Carlos Anchundia y Patricio Zambrano

Además de los profesores, se ha incluido al estudiante Fernando Huilca, representado en el estilo de arte de píxeles, como el protagonista del nivel. Fernando Huilca tiene la tarea de recolectar exámenes con buenas calificaciones y logotipos de lenguajes de programación a lo largo del nivel. A su vez, este personaje tiene la capacidad de crear y destruir objetos dentro del juego, lo que añade un elemento estratégico y de resolución de problemas al nivel "Politécnico".

La inclusión de personajes inspirados en figuras reales y la creación de un nivel personalizado en el juego agregan un toque distintivo y personalizado a la experiencia de juego. Este enfoque permite a los jugadores sentirse más conectados con el entorno del juego y brinda una experiencia única que destaca entre los demás niveles.

Ilustración 9. Player de Nivel Politécnico: Fernando Huilca

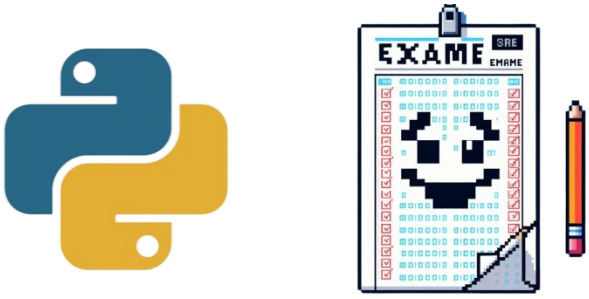


Ilustración 10. Objetos Recolectables en Nivel Politécnico: Phyton y un Examen

4.4. Fase IV. Implementación

4.4.1. Lógica del Juego

La lógica del juego se basa en varios elementos para avanzar en los niveles. En primer lugar, el jugador debe recolectar frutas que aparecen en diferentes etapas del juego. Cada fruta otorga puntos, lo que contribuye al puntaje total del jugador. Cuando se recolectaron las frutas del nivel, el nivel se desbloquea, permitiendo al jugador avanzar al siguiente nivel.

Sin embargo, el jugador debe enfrentarse a enemigos que se mueven de manera autónoma por el nivel. Estos enemigos representan una amenaza para el jugador, ya que, si colisionan con él, el jugador muere y el nivel se reinicia. Por lo tanto, el jugador debe evitar a los enemigos mientras recolecta las frutas.

Una habilidad especial del jugador es la capacidad de generar y romper hielo. El jugador puede crear bloques de hielo a partir de la posición siguiente a su dirección de movimiento. Estos bloques de hielo pueden ser utilizados para bloquear o desviar el camino de los enemigos, proporcionando una estrategia adicional para evitarlos. Además, el jugador puede romper los bloques de hielo existentes, lo que desencadena una reacción en cadena tipo dominó. Sin embargo, esta capacidad debe utilizarse con precaución, ya que puede crear obstáculos adicionales o bloquear el avance del jugador si no se utiliza de manera adecuada.

Control de colisiones: Dentro del juego, se implementan diferentes modalidades de colisión entre los objetos. A continuación, describiremos cada tipo de colisión y cómo se controla en el juego:

Colisión entre un enemigo y el jugador (player): Esta colisión se produce cuando un enemigo y el jugador ocupan la misma posición dentro de la matriz del juego. En este caso, se detecta que el jugador ha sido alcanzado por un enemigo, lo que resulta en la finalización del juego ya que el jugador muere.

Colisión entre una entidad y un bloque sólido: Esta colisión ocurre cuando una entidad (ya sea el jugador o un enemigo) colisiona con un bloque sólido, ya sea destruible o inmutable. Si el objeto que colisiona es un enemigo, este cambiará su dirección de movimiento e intentará

moverse en una dirección alternativa que no esté obstruida por un bloque sólido. Para lograr esto, se calcula la siguiente posición del enemigo teniendo en cuenta su dirección actual y se verifica si en esa dirección hay un bloque sólido. En caso afirmativo, el enemigo ajusta su dirección para evitar la colisión y continuar su movimiento.

Colisión entre monstruos (enemigos): Cuando dos enemigos colisionan entre sí, se produce un cambio en sus direcciones de movimiento. Si ambos enemigos no tienen un objeto sólido frente a ellos, ajustarán sus direcciones y continuarán moviéndose en una nueva trayectoria.

4.4.2. Package Data

Este paquete tiene las diferentes clases que gestionan el correcto uso y flujo de los datos y la implementación de los mismos para el funcionamiento del juego.

4.4.3. Package Presentation

Se encarga de tener todas las clases que permiten al usuario tener una experiencia visual al momento de interactuar con la lógica del juego la que se asemeja muchísimo con el juego del cual tiene inspiración este proyecto

4.4.5. Guardado de Partida y Scores

El juego da un valor a cada fruta la cual al momento de ser recolectada asigna un score al jugador, el cual puede ir subiendo su score de acuerdo a más frutas recolecta y avanza de nivel, la lógica de juego permite guardar los datos para su visualización posterior en un apartado “Score”

El juego permite guardar la partida al momento de pausar el juego lo que permite al usuario volver a jugar desde el punto en el que se encontraba, teniendo esta opción en la primera pantalla al momento de empezar el juego.

4.5. Tecnologías utilizadas

Para el desarrollo del proyecto, se utilizaron varias tecnologías fundamentales que permitieron gestionar de manera eficiente el trabajo en equipo y el proceso de desarrollo. En primer lugar, se empleó Git y GitHub como sistemas de control de versiones distribuidos. Git facilitó la gestión de las distintas ramas de desarrollo, permitiendo a cada miembro del equipo trabajar en su propia rama y fusionar los cambios de manera controlada. GitHub, por su parte, ofreció un

repositorio centralizado donde se almacenaba el código y se llevaba un registro de los cambios realizados, garantizando la integridad del código y facilitando la colaboración entre los desarrolladores.

Además, se utilizó IntelliJ como entorno de desarrollo integrado (IDE) para el lenguaje Java. Este IDE proporcionó un conjunto de herramientas robustas para la codificación, depuración y prueba del código, destacando el uso frecuente de Code With Me, una función que permitió la colaboración remota en tiempo real entre los miembros del equipo.

Para la comunicación y coordinación del equipo, se recurrió a Microsoft Teams, una plataforma que ofreció herramientas de chat, videollamadas y compartición de archivos, facilitando la organización de reuniones y la comunicación fluida entre los integrantes del proyecto. Asimismo, se utilizó WhatsApp como canal de comunicación informal y rápido para resolver consultas y coordinar tareas de manera ágil.

En cuanto a la generación de imágenes y diseño gráfico, se empleó Microsoft Designer, una herramienta que permitió la creación de elementos visuales y gráficos necesarios para la interfaz de usuario y otros aspectos del proyecto. Esta combinación de tecnologías proporcionó un entorno de desarrollo completo y eficiente, permitiendo abordar con éxito los desafíos del proyecto y alcanzar los objetivos establecidos.

5. PRUEBAS DE CALIDAD

5.1. Problemas encontrados y soluciones aplicadas

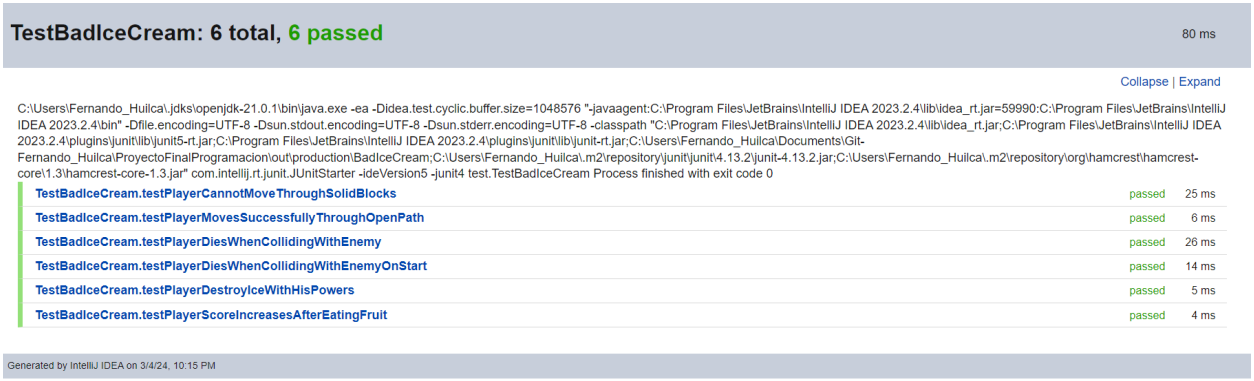


Ilustración 11. Tests Realizados

Durante el desarrollo del código, nos enfrentamos a varios problemas que requirieron soluciones creativas para ser resueltos. Uno de los desafíos iniciales surgió al unificar el código de la lógica del juego en la consola, ya que las imágenes no se estaban leyendo correctamente. Esto provocaba errores que afectaban el funcionamiento del juego. Para solucionar este problema, optamos por utilizar la dirección directa del proyecto en lugar de una dirección absoluta para la lectura de las imágenes, lo que permitió que estas se cargaran correctamente.

Sin embargo, el problema más significativo que enfrentamos fue la incapacidad de leer el teclado cuando implementamos la ventana y el KeyListener. Esto afectaba el control del movimiento del jugador y obstaculizaba la jugabilidad del juego. Para resolver este problema, desarrollamos una clase especializada en la lectura del teclado que se encargara exclusivamente de controlar el movimiento del jugador. Esta solución nos permitió aislar y gestionar de manera eficiente las entradas del teclado, garantizando un control preciso del jugador en el juego.

Además, nos encontramos con dificultades relacionadas con el movimiento de los enemigos, ya que estos eran independientes del movimiento del jugador. Para abordar este problema, estudiamos e implementamos el concepto de hilos (Threads) para independizar las acciones de los enemigos y el jugador. Esta solución nos permitió controlar el movimiento de los

enemigos de manera independiente, lo que mejoró la jugabilidad y la experiencia del juego en general.

ACTIVIDADES	MESES									
	Enero				Febrero				Marzo	
	SEMANAS									
	1	2	3	4	1	2	3	4	1	2
Fase I. Identificación										
Identificación del Juego										
Investigación del Informe de Software										
Fase II. Análisis										
Modelo-Vista-Controlador										
Jugabilidad de Bad Ice Cream										
Fase III. Diseño										
Diagrama UML										
Personajes, Enemigos, Mapas y Menús										
Fase IV. Implementación										
Lógica del Juego										
Package Data										
Package Presentation										
Guardado de Partida y Scores										
Fase V. Presentación										
Presentación del Juego										
Defensa del Juego										

Ilustración 12. Cronograma de Actividades

6. RESULTADOS OBTENIDOS

Desarrollo de un juego funcional: Hemos logrado crear un videojuego completamente funcional con una interfaz gráfica atractiva y una mecánica de juego sólida.

Implementación de características clave: Se han implementado características importantes, como la recolección de objetos, el control de personajes, la detección de colisiones y un sistema de puntuación.

Solución de problemas técnicos: Hemos enfrentado y resuelto varios problemas técnicos durante el desarrollo, como problemas de lectura de teclado, gestión de colisiones y generación de animaciones.

Mejora de habilidades: El proyecto nos ha brindado la oportunidad de mejorar nuestras habilidades en programación, diseño de juegos, gestión de proyectos y trabajo en equipo.

Creación de una experiencia de juego satisfactoria: El resultado final es un videojuego que ofrece una experiencia de juego entretenida y desafiante para los jugadores, lo que representa el éxito del proyecto.

Al momento de finalizar con todas las etapas previstas del proyecto se puede evidenciar el claro avance en la lógica y correcto entendimiento de la Programación Orientada a Objetos y el gran número de ventajas y ayuda que presenta está lógica al momento de llevarse a una generalización o implementación de nuevas funcionalidades.

7. CONCLUSIONES

En conclusión, el proyecto de creación del videojuego ha sido un proceso desafiante pero gratificante. Hemos logrado desarrollar un juego entretenido y dinámico que ofrece una experiencia inmersiva para los jugadores. A lo largo del desarrollo, enfrentamos diversos desafíos técnicos y creativos, como la implementación de la interfaz gráfica, la gestión de colisiones y la generación de animaciones. Sin embargo, gracias al trabajo en equipo y a la aplicación de soluciones innovadoras, pudimos superar estos obstáculos y alcanzar nuestros objetivos. El juego cuenta con un sistema de puntuación, una variedad de niveles y personajes, y un diseño de mapa estructurado que añade profundidad y complejidad a la experiencia del jugador. En resumen, estamos orgullosos del resultado final y esperamos que el juego brinde diversión y entretenimiento a todos los que lo jueguen.

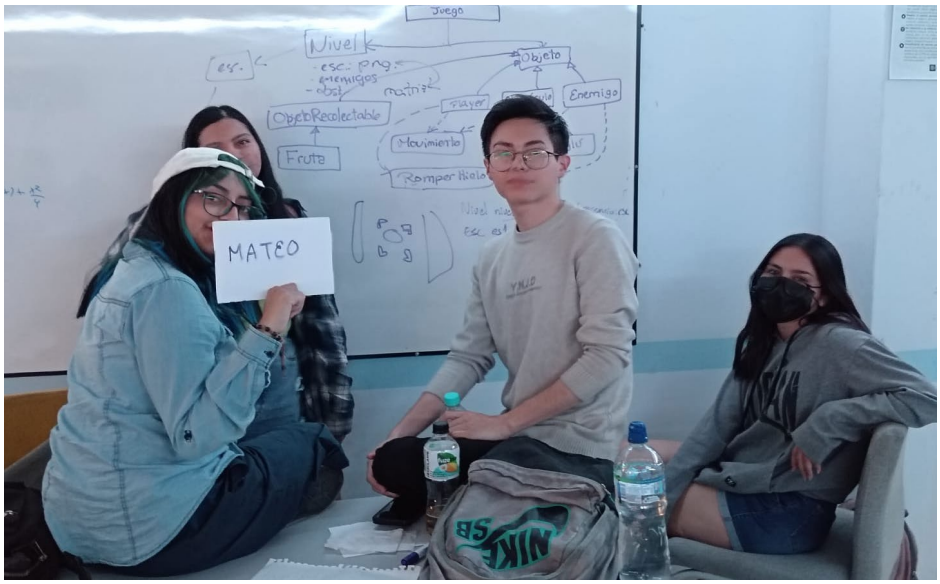
8. ANEXOS

Anexo 1

[Reunión]

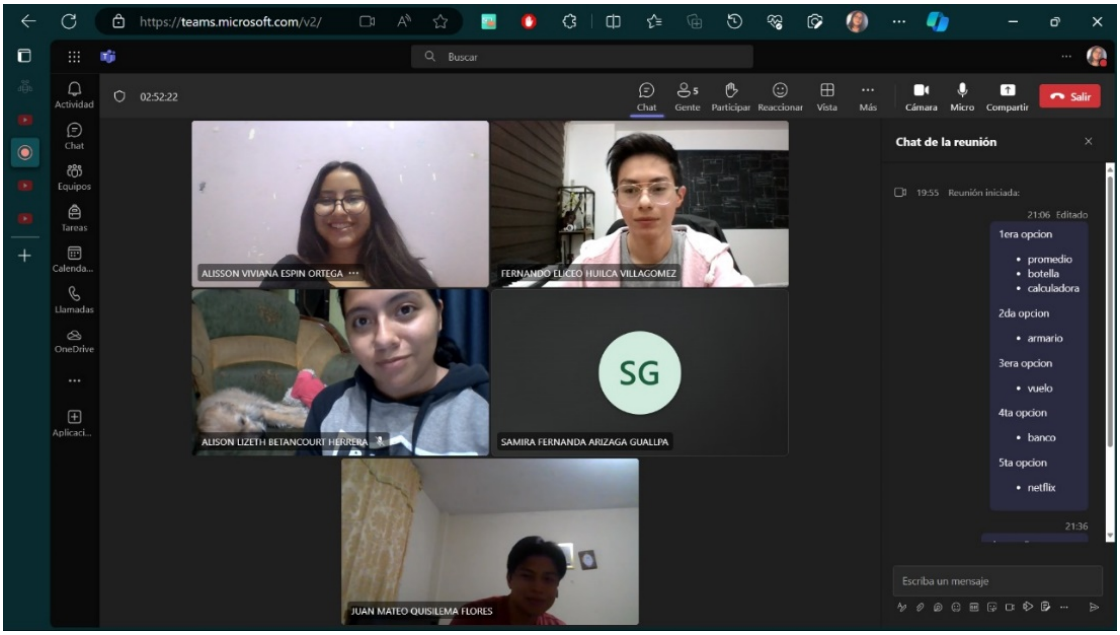
Anexo 2

[Ideas y Diseño del Diagrama de Clase]



Anexo 3

[Reunión Planificación de Horarios de Trabajo]

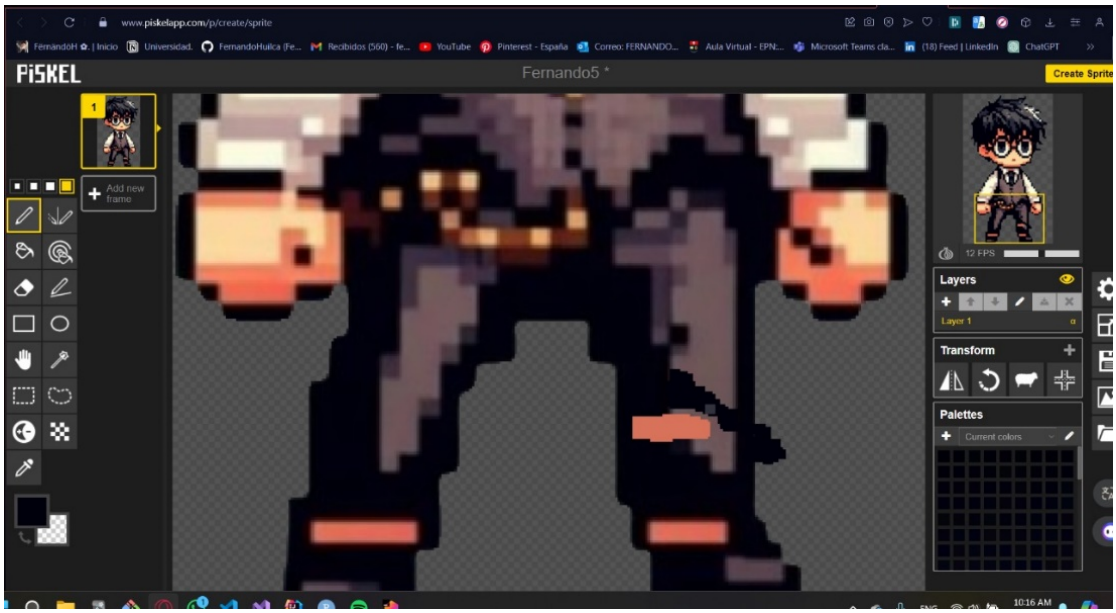


Anexo 4

[Reunión Revisión de Avances]

Anexo 5

[Realización y Personalización de Avatares]



Anexo 6

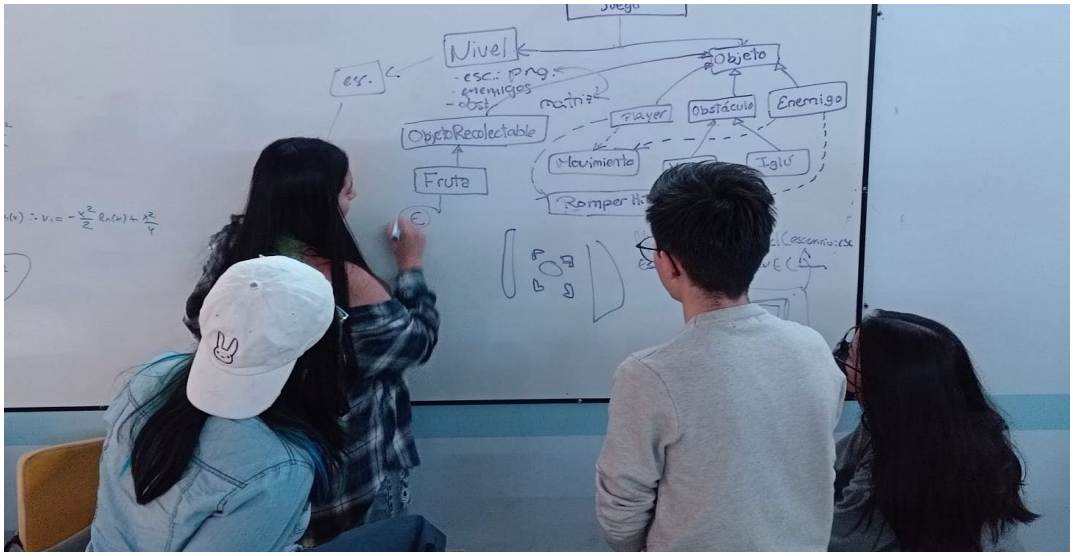
[Ideas y Discusiones de Diseño Inicial]

Anexo 7

[Referencia para avatar]

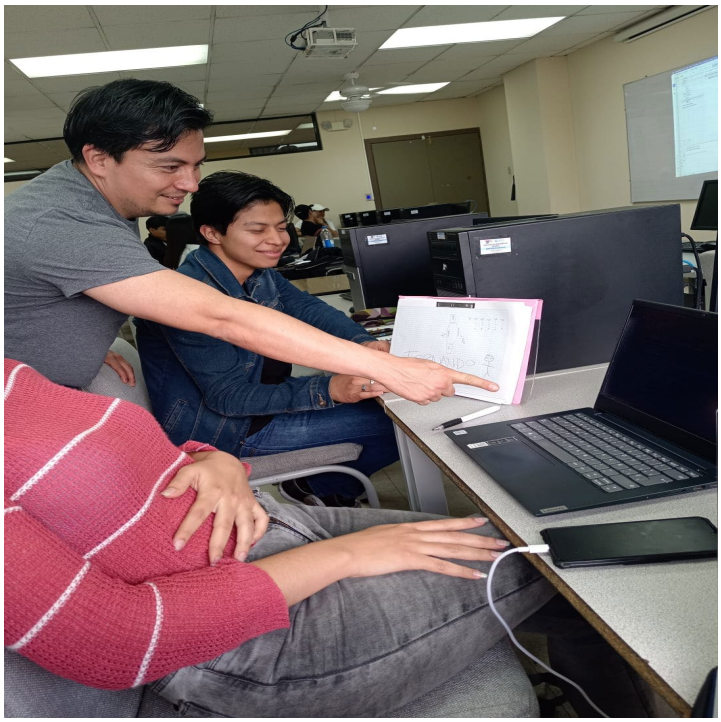
Anexo 8

[Discusión de Parámetros a Abstraer]



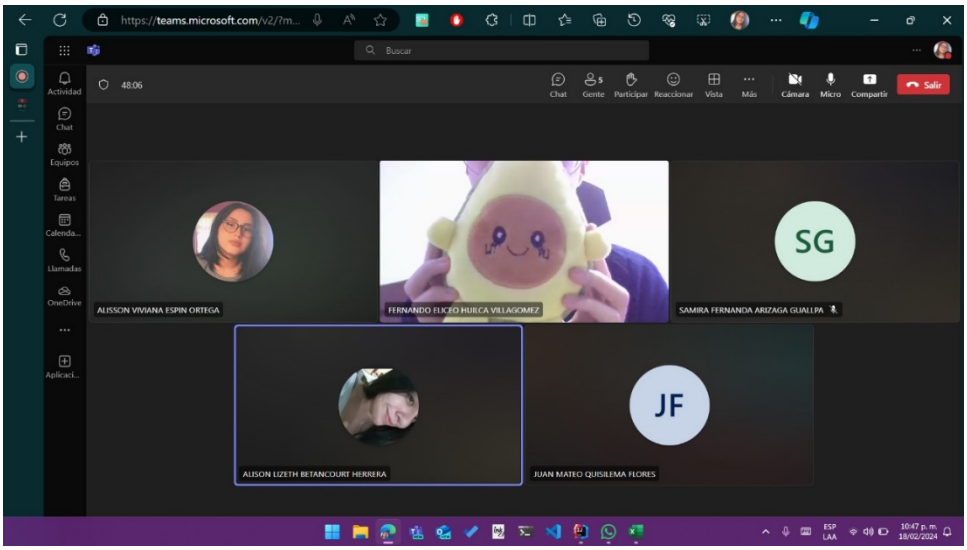
Anexo 9

[Revision UML]



Anexo 10

[Reunión Asignación de Funciones]



Anexo 11

[Horario de Trabajo]

	Lunes	Martes	Miércoles	Jueves	Viernes
Samira	no	si 2 a 4 (virtual)	si (hasta las 4)	si (hasta las 4)	si 1:30 a 4
Alison	si	si	4-6 no	si	si
Mateo	si 2 a 4 (virtual)	si 2 a 4 (virtual)	si (hasta las 4)	si (hasta las 4)	si
Fernando	si	si	si 2 a 4	si	si
Viviana	si	si	si	si	si
	1:30 a 5	1:30 a 5	1:30 a 5	1:30 a 4	1:30 a 5
	Presencial	Híbrido	Presencial	Presencial	Presencial

	Lunes	Martes	Miércoles	Jueves	Viernes
Samira	x	x			x
Alison					
Mateo					
Fernando			x		x
Viviana					
	En la noche de	8:00	10:00		
	Como máximo hasta	11:00			

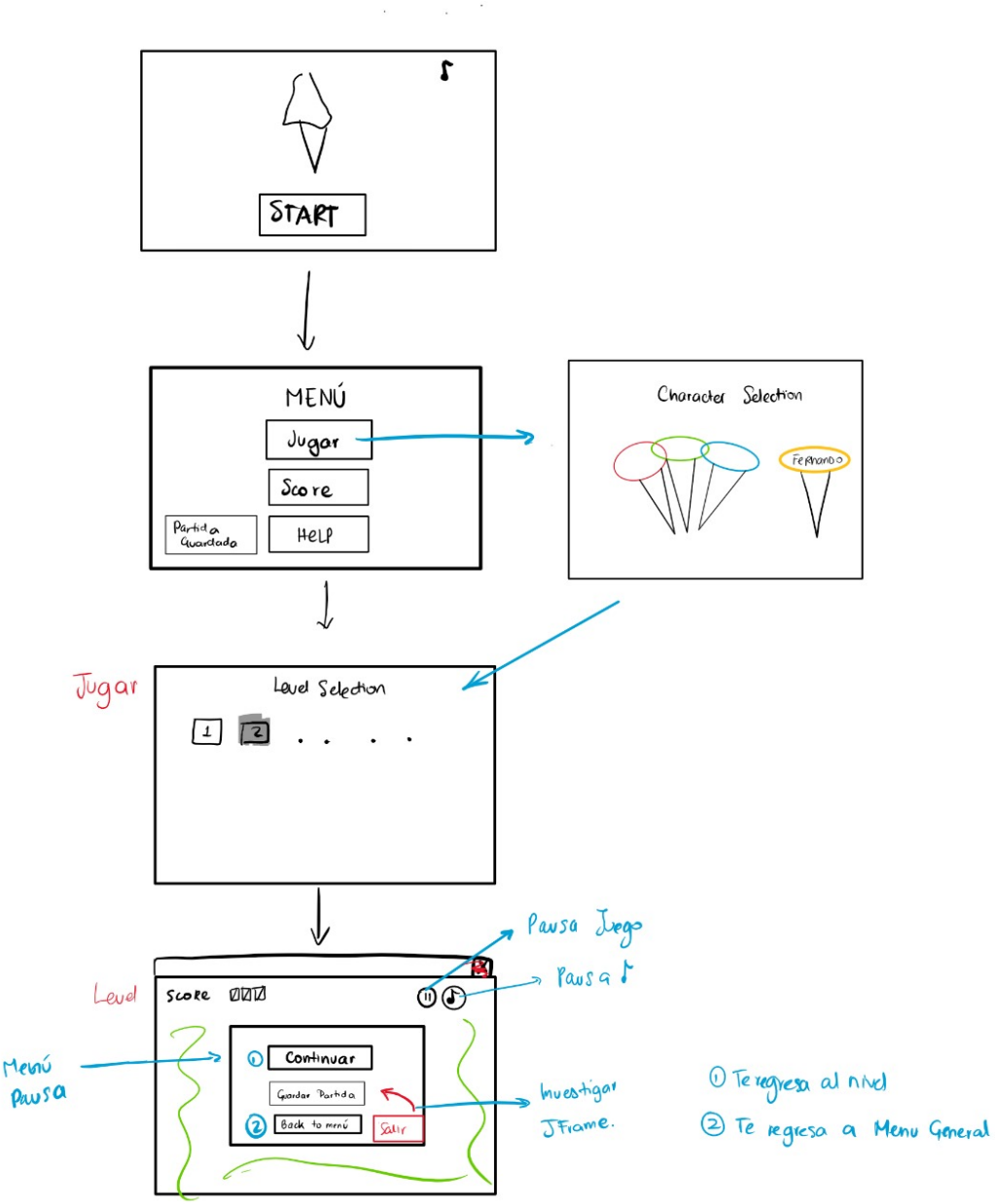
Anexo 12

[Cronograma a Seguir]

SÁBADO	DOMINGO	LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SÁBADO	DOMINGO
Situación Básica	Situación Básica	Situación Básica	Situación Básica	Situación Básica	Correcciones de Situación Básica	Situación Básica	Situación Básica	Correcciones Generales
Lógica Principal del Juego	PowerUps	Lógica Principal Niveles	Lógica Principal de Niveles	Interfaz Gráfica		Revisión del código	Interfaz Gráfica	Versión Final depurada con Interfaz
					Corrección ejecución nivel			
Movimiento primer monstruo	Romper hielos	Diferentes frutas por nivel	Menú Pausa	Inicialización Posiciones Enemigos		Correcciones lectura de dirección	Inicio de la ventana	Presentación Player
Muerte del jugador	Poner hielos		Selección del helado	Leer todo el tiempo posición player	Prueba de Guardado de Data	Correcciones lectura score frutas	Inicio de mapa en la ventana	Movimiento Player
Colisiones?		Pasar de Nivel		Colisiones entre enemigos		Guardado de Scores		
	Segunda Prueba Nivel 1	Movimientos enemigos (posicion player)	Enemigo que te sigue		Correcciones Generales	Guardado de Partida	Presentación Menú Principal	Presentación Frutas
Presentación en consola		Nivel 2		Prueba colisiones entre enemigos	Manejo de errores			
	Lógica Principal Niveles		Prueba enemigos		Depuración del código	Versión Final Código en Consola	Prueba de menus en la interfaz	Presentación Enemigos
Prueba Nivel 1		Prueba Nivel 2		Guardado de Data				Movimiento Enemigos
	Enemigo en otro hilo	Prueba cambio niveles			Prueba del Código Limpio	Documentación Funciones	Carga de Imágenes	
				Lógica de Scores		Actualización UML	Imágenes de bloques	Animación player
	Posición frutas	Menú de juego				Actualización Informe	Imágenes de animaciones	Animación de enemigos
	Recolección frutas			Imágenes				
	Jugador gane el nivel	Menú Selección de Nivel					Prueba de presentación Mapa	Prueba Nivel 1 con Interfaz
				Carga de imágenes de bloques				
	Nivel 1	Prueba con menus		Carga de imagen player			Documentación Funciones	Presentación Menús
				Carga de imagen de enemigos			Actualización UML	
	Tercera Prueba Nivel 1						Actualización Informe	Menú Help
								Menú Scores
								Menú Selección de Nivel
6:00 PM	8:00 PM							
hasta las 10 pm								Lógica Selección Character
								Menú Selección Character
								Prueba Menús

Anexo 13

[Bosquejos y Modelos]



9. REFERENCIAS

Ayala, M. (2022, 18 agosto). *Informe*. Lifeder. https://www.lifeder.com/informe/

Bomberman | Visual Paradigm User-Contributed Diagrams / Designs. (2021, 28 abril). Visual Paradigm Online. https://online.visual-paradigm.com/community/share/bomberman-j7hq2ji29

Contributors to Nitrome World. (s. f.). *Helado de chocolate*. Nitrome World. https://nitromeworld.fandom.com/es/wiki/Helado_de_Chocolate

CoolJuegos.com. (2011, 13 septiembre). *Bad Ice Cream - Online juego | CoolJuegos.com*. https://www.cooljuegos.com/juego-en-linea/bad-ice-cream/

Figura 17. Diagrama de clases para la versión La Olla. (s. f.). ResearchGate. https://www.researchgate.net/figure/Figura-17-Diagrama-de-clases-para-la-version-La-Olla_fig11_270892854

Hernandez, J. (2017, 02 abril). *Crear un juego en Java: Principiantes*[Video]. YouTube. Crear Un Juego En Java: Principiantes - YouTube

Hernandez, R. D. (2021, 28 junio). *El patrón modelo-vista-controlador: Arquitectura y frameworks explicados*. freeCodeCamp.org. https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/

KeepCoding, R. (2023, 12 septiembre). ¿Cómo aplicar la programación orientada a objetos en juegos? *KeepCoding Bootcamps*. https://keepcoding.io/blog/programacion-orientada-a-objetos-en-juegos/

Nitrome, C. T. (s. f.). *Bad Ice-Cream*. Nitrome. https://nitrome.fandom.com/es/wiki/Bad_Ice-Cream#:~:text=Bad%20Ice%2DCream%20es%20un,mientras%20se%20defienden%20de%20enemigos

RyiSnow. (2021, 03 octubre). *How to Make a 2D Game in Java*[Video]. YouTube. How to Make a 2D Game in Java - YouTube

Upn. (2024, 22 febrero). *Desarrollo de software: ¿Qué es y cuál es su importancia?* Blogs UPN. https://blogs.upn.edu.pe/ingenieria/2024/02/22/que-es-desarrollo-de-software/

Wiki, C. T. N. (s. f.). *Ice cream characters*. Nitrome Wiki. https://nitrome.fandom.com/wiki/Ice_cream_characters