PROCESO DE DESARROLLO DE NUESTRO PROYECTO RECREACIÓN DEL JUEGO “BAD ICE CREAM”

1. Frase introductoria

"Imagina un mundo donde la creatividad se fusiona con el dulce placer de los helados. Hoy los llevaremos a un viaje detrás de escena del desarrollo de nuestro apetitoso videojuego sobre helados."

ANÁLISIS

2. Introducción

¿Qué es Bad Ice Cream?

Bad Ice Cream es un juego de arcade muy divertido y adictivo. El objetivo es recoger todas las frutas de cada nivel, evitando que los enemigos te atrapen, enemigos que te persiguen o que incluso pueden volar por encima de los bloques. Se puede jugar tanto a solas como en modo cooperativo y se puede escoger entre varios sabores de helado, como chocolate, fresa o vainilla. Bad Ice Cream es un juego ideal para pasar el rato y disfrutar de una aventura helada.

Mecánicas del juego

- Recoger todas las frutas
- Esquivar a los enemigos
- Poner y quitar hielos

3. Personajes

Jugadores

- Jugadores originales del juego: Strawberry, Chocolate and Vanilla Icecream
- Jugadores característicos de nuestro programa: Profe Anchundia, Profe Patricio, Fernando Huilca

Enemigos

Dinámica de movimiento de cada uno de los enemigos: Troll y BlueCow

Frutas

Dinámica de recolección de frutas: Banana, Uvas, C++, Phyton y Examen

DISEÑO

4. Metodologías utilizadas

Se utilizó el lenguaje Java para el desarrollo del código de nuestro proyecto, mediante el uso del paradigma de programación que nos enseñaron durante el semestre, la Programación Orientada a Objetos (POO). Así mismo para el proceso de desarrollo se necesitó establecer etapas claras de avance en nuestro proyecto, siendo estas el Análisis, el Diseño y la Implementación. De igual manera se utilizó una arquitectura multicapa, Three Tier Architecture que consiste: Presentation, Business y Data. La ventaja de este estilo de desarrollo es que se puede llevar a cabo en varios niveles y en caso de que se necesite realizar algún cambio solo afectará al nivel requerido, ahorrando a los desarrolladores el tener que revisar el código fuente de otros módulos.

Java, POO, Waterfall and Three Tier Architecture

5. Tecnologías utilizadas

- **IntelijIDEA**: fue el IDE que se utilizó durante todo el semestre
- **CodeWithMe:** una de las funcionalidades que implementa el IDE de JetBrains
- **Git y GitHub:** un versionador de código que cuenta con un repositorio local y remoto
- **ChatGPT:** Una inteligencia artificial en la que nos apoyamos para el desarrollo de cosas puntuales o para la implementación de librerías desconocidas

6. Problemas encontrados

La codificación en conjunto con CodeWithMe

En CodeWithMe se puede crear una sesión remota de tal manera que todos los participantes pueden trabajar sobre el mismo script, una herramienta que sin duda fue útil en el inicio de nuestro proyecto, sin embargo, en el momento en el que un participante necesitaba realizar la ejecución o debugging del código que desarrolló, se truncan los procesos de desarrollo de los demás participantes puesto que no permite la modificación del código hasta que la ejecución del programa haya finalizado. En este sentido hay veces en las que un desarrollador necesita realizar ejecuciones constantes de su código, siendo esta una limitación para el avance del proyecto optamos por manejar otras herramientas: Git y GitHub

Manejo de objetos como parámetros de una función en Java

Cuando nos encontrábamos desarrollando el movimiento de las entidades descubrimos que nuestro código por más lógico que parecía no estaba funcionando de la manera correcta. Después de un arduo proceso de búsqueda de soluciones e investigación en internet, incluso de preguntarle a chatGPT varias veces el porqué nuestro código no funcionaba, encontramos un artículo en StackOverflow acerca como Java maneja el flujo de datos.

El artículo de StackOverflow hablaba de como cuando una función en Java recibe un objeto como parámetro de entrada, a diferencia de C o C++, no se está enviando una copia del objeto, sino que se está enviando una dirección de memoria, por lo que las modificaciones realizadas a este objeto dentro de la función afectaran directamente al objeto enviado. Volviendo

innecesaria la devolución del objeto por parte de la función, pero complicando un poco más la lógica acerca del manejo de los datos que se envían a las funciones.

7. Soluciones aplicadas

Git y GitHub

Es así, que para avanzar en el desarrollo de nuestro proyecto comenzamos a utilizar GitHub, creando un repositorio remoto al que los integrantes estaban agregados como colaboradores, de esta manera cada uno encontró más comodidad trabajando en una funcionalidad específica por separado de tal manera que posteriormente utilizando la funcionalidad de merge que nos brinda Git pudiéramos unificar las versiones de código que estábamos trabajando.

Objetos como parámetros de una función en Java

Es entonces que, tras otro proceso de investigación, nos encontramos con la interfaz Cloneable, la cual permite crear una copia del objeto en cuestión ocupando una dirección de memoria distinta al original. Justo lo que buscábamos. Este nuevo conocimiento también nos llevó a la solución de la lógica del movimiento del enemigo BlueCow. (Mateo)

8. Presentación del UML

Capa Presentation

Es la que el usuario ve y la encargada de recibir/capturar la información ingresada por el usuario a través de un proceso mínimo.

Capa Business

Es donde se encuentran los programas que se ejecutan, recibiendo los mensajes enviados por la capa de Presentation. Es aquí donde se establecen todas las reglas del programa o la lógica principal del programa.

Capa Data

Es donde se encuentran los datos y es la encarga de acceder a los mismos.

IMPLEMENTACIÓN

Se presenta la ejecución del programa

9. Nos sentimos orgullosos

- Del movimiento implementado por el enemigo BlueCow
- De la facilidad que se tiene al momento de crear nuevos niveles