Instituto Tecnológico de Costa Rica  
Proyecto #2  
Juan Jose Rodríguez Chaves

CE-1101 Introducción a la Programación

Diego Andrés Mora Rojas

Jeff Schmidt Peralta  
I Semestre de 2025

# Introducción:

En este proyecto se desarrolla la programación de una versión moderna del clásico videojuego “Bomberman” utilizando Python. Este integra conceptos fundamentales como la iteración, la programación orientada a objetos (POO) y la creación de interfaces gráficas. Elementalmente el videojuego se compone de un laberinto compuesto por bloques que pueden ser destruidos por el jugador a través de bombas y otros que son indestructibles, enemigos varios con características que los distingan, una llave oculta en el laberinto que el jugador debe hallar para avanzar al siguiente nivel.

Objetivos principales:

* Aplicar principios de POO en el desarrollo del proyecto.
* Implementar una interfaz gráfica intuitiva para la correcta funcionalidad del programa.
* Crear un sistema funcional de niveles que progresivamente aumenten la dificultad.
* Desarrollar mecánicas funcionales y completas en el juego, (movimiento, bombas, enemigos, power-ups, etc).

# Descripción del Problema:

El problema abordado consiste en el desarrollo de un videojuego funcional que cumpla con toda especificación requerida, algunos ejemplos de esto:

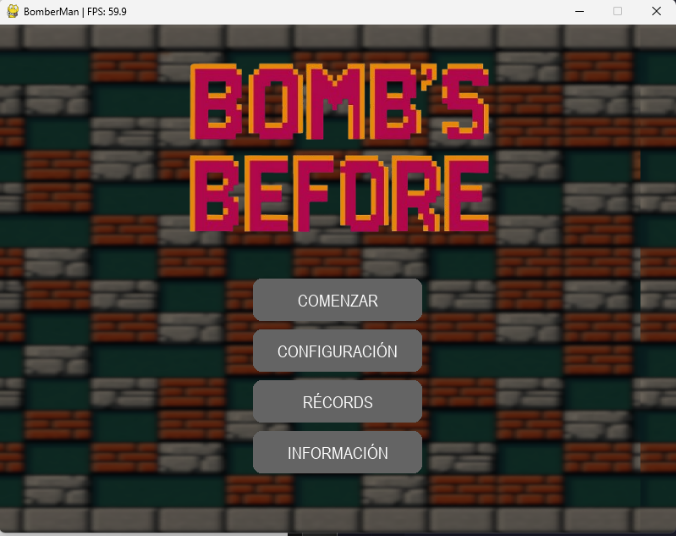
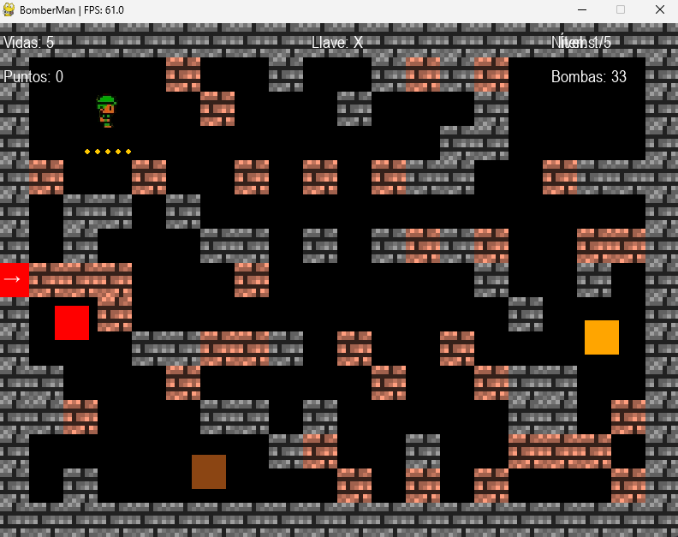
Problemas técnicos a resolver:

* Gestión de la colisión entre personajes, obstáculos, explosiones y enemigos.
* Implementación de un sistema de niveles con un sistema de dificultad progresiva.
* Creación de una interfaz gráfica que funciona como “menú” para el juego que contenga pantalla de inicio, configuración, selección de personajes, etc.
* Manejo de eventos en tiempo real en el videojuego, (explosiones, movimiento, interacciones).
* Generación aleatoria de objetos, (llave, puerta, bloques destructibles e indestructibles, aparición de powerups, generación de enemigos).

Elementos clave:

* Sistema de juego con 4 niveles ( 3 niveles + 1 jefe final).
* Personaje jugable con estadísticas y habilidades.
* Enemigos con características que los diferencia, (comportamientos diferentes, vida, daño, etc.).
* Mecánica de explosiones y bombas
* Sistema de puntos y mejora del personaje

# Análisis de Resultados

En el proyecto se implementaron varias mecánicas únicas como una pantalla inicial con un menú principal compuesto por configuraciones, récords e información, además, al comenzar el juego se incorpora la selección de personajes únicos, cada uno con una habilidad diferente y estadísticas únicas. En la jugabilidad se crea un laberinto de bloques con una puerta colocada de manera aleatoria en las paredes del mapa y una llave escondida en uno de los bloques destructibles colocados en el mapa, la HUD muestra la vida del jugador, la cantidad de bombas que posee, el nivel en el que se encuentra, el powerup que tiene almacenado, los ítems que ha obtenido, etc.

Por otro lado, la pantalla de game over muestra la puntuación obtenida junto con una instrucción para iniciar de nuevo. La batalla final contra el jefe es completamente diferente a los otros niveles y presenta un solo enemigo más fuerte que todos los otros.

El movimiento del jugador es fundamentalmente hecho con direcciones verticales, horizontales y diagonales con colisiones con los bloques y el recibimiento de daño al explotar una bomba cerca de él o tocar un enemigo

# Dificultades Encontradas:

Las dificultades encontradas en el juego han sido muchas durante el desarrollo del proyecto, aquí algunas junto con sus soluciones:

* Problemas al implementar un sistema de “damage invulnerability”, al principio, cuando la hitbox del personaje del jugador colisionaba con la explosión de una bomba o la hitbox de un enemigo sus puntos de vida bajaban a 0 en solo un segundo, ocasionando un game over inmediato.
  + Esto se logró resolver creando una función con nuevos atributos:
    - “self.invincible” es un atributo booleano que maneja cuándo el jugador es invencible, muy útil para ser utilizado con ítems nuevos y powerups que provoquen que el jugador sea invulnerable por cierto tiempo.
    - “self.invincible\_duration” es otro atributo constante que determina cuánto tiempo será invencible el jugador, junto a “self.invincible\_frames” se maneja la duración en frames.
    - De manera adicional, se implementa el atributo “self.visible” para manejar una animación de parpadeo para mostrar la invulnerabilidad temporal.
* Otro problema principal fue la generación de bloques. Para la generación de los bloques en el mapeado del nivel se prefirió un sistema que se generará tanto bloques destructibles como indestructibles para darle una aleatoriedad y distinción a los niveles. Sin embargo, esto ocasionó la creación de nuevas funciones que se asegurarán de varias cosas, entre ellas:
  + Una función que verifique al jugador para así no generar bloques indestructibles encima de él
  + Otra función que ahora asegure de no atraparlo entre estos bloques.
  + La puerta se genera a los costados centrales de los bordes del mapa, los bloques indestructibles también podrían obstaculizar el paso del jugador a esta, una nueva función se asegura de que esto no pase.
  + Con esto también se editaron otras funciones para no generar los bloques indestructibles en lugares como: encima de un enemigo, no generar la llave en un bloque indestructible.
  + Aunque no incluya los bloques indestructibles, los bloques destructibles también podían encerrar al jugador, provocando que necesitara de una bomba y perder 1 vida para poder escapar, esto también se solucionó con validaciones.

# Bitácora de Actividades:

Semana 1 (9-15 de julio):

Se empezó el proyecto formalmente creando el jugador, el laberinto, la funcionalidad de las bombas, el primer enemigo implementado, la funcionalidad básica de las llaves y la puerta:

* El jugador se definió como una clase desde el principio, con sus propios atributos “vida”, “velocidad”, “color”, “bombas disponibles”, “capacidad de bombas”; este último ya que a futuro se decidiría para los distintos personajes disminuir o aumentar la capacidad de las bombas.
* Para el laberinto se define un atributo “self.map” como una lista vacía [] en la que se concatenarán todos los bloques, sin embargo, se trabaja como un vector, usando valores de “x” y “y” para trazar el mapa
* Se dibujó de manera básica el hud, más que todo como un método debug incluso para determinar que todo estuviera funcionando de manera óptima.
* Por último, la llave y la puerta se definieron en sus clases respectivas y se dibujaban de manera básica como rectángulos con varios fallos al principio en las que se coloreaba la llave encima del bloque en el que estaba escondida, luego resuelto utilizando un método de aparición hasta que sea explotado el bloque.

Semana 2(16-22 de junio):

El enfoque se centró en enriquecer la experiencia de juego con elementos clave:

Power-Ups:

• Se implementaron tres tipos fundamentales: salud que recuperaba corazones, daño que aumentaba el alcance de las explosiones, y velocidad que temporalmente mejoraba la movilidad del personaje. Cada power-up tenía efectos visuales distintivos y sonidos característicos al ser recolectados.

Ítems Especiales:

• El sistema de ítems incluía habilidades únicas asignadas a teclas específicas: congelación que inmovilizaba enemigos, bolas de fuego que atacaban a distancia, y escudos que protegían temporalmente al jugador. Cada ítem mostraba iconos en el HUD cuando estaban disponibles.

Mejoras Visuales:

• Se añadieron sprites animados para dar vida al juego: el jugador mostraba diferentes posturas según su dirección de movimiento, las explosiones tenían una animación de tres fases, y los efectos especiales incluían partículas simples pero efectivas. El HUD se rediseñó para mostrar claramente la información esencial durante la partida.

Sistema de Puntuación:

• Se desarrolló un mecanismo de puntos con bonificaciones por acciones estratégicas como eliminar múltiples enemigos en una sola explosión. El sistema incluía un multiplicador que incentivaba jugadas arriesgadas y una pantalla de resumen al final de cada nivel.

Optimizaciones:

• Se mejoró el rendimiento agrupando sprites similares y optimizando las colisiones. La IA de los enemigos recibió un pre-cálculo de rutas para movimientos más inteligentes. Se implementó un sistema de capas para gestionar eficientemente los elementos del juego.

• La sincronización de animaciones utilizando spritesheets en obtenidos en línea trazándolo por columnas y filas en la imagen, mapeando adicionalmente la dirección que tendrá, ya sea arriba, abajo, izquierda o derecha, luego se extraen los frames de manera individual y utilizar un frame especifico cuando el jugador se queda quieto según la dirección a la cual está viendo

• Por otro lado, las explosiones y la animación de las bombas utilizan imágenes separadas extraídas de distintos archivos .gif, de esta manera se utiliza la carga de los frames ordenados por archivos

• Además, se añadieron las armas para los personajes, cada uno empieza con un arma por defecto, excepto el personaje tanque, este empieza con una escopeta, (los otros personajes también pueden obtenerla gracias al item que le corresponde), igualmente definido gracias a ángulos, vectores y librerías como “math”.

Semana 3(23-28 de junio):

1. Refinamiento del Código y Optimización:

• Se realizó una revisión exhaustiva de la estructura del código para eliminar redundancias y mejorar la legibilidad.

• Se implementó documentación interna (comentarios y docstrings).

• Se corrigieron bugs reportados durante pruebas internas, como problemas de sincronización en animaciones al salir al menú principal, errores en el spawn de la llave en bloques inaccesibles, fallos en la detección de daño al jugador en ciertas situaciones.

2. Pantalla de Configuración y Personalización:

• Se implementó una pantalla de ajustes funcional con las siguientes opciones:

Volumen: Control deslizante para ajustar música y efectos de sonido.

Pantalla: Opción de pantalla completa o ventana.

3. Sistema de Guardado y Puntuaciones:

• Se desarrolló la pantalla de mejores puntajes, que muestra:

Top 5 jugadores con sus nombres y puntuaciones.

Opción de reiniciar los records.

• Pantalla de Game Over/Victoria:

Muestra estadísticas finales (tiempo, puntos).

Opción de volver al menú.

5. Balanceo y Dificultad:

• Se ajustaron los parámetros de los enemigos según el nivel:

Nivel 1: Poca cantidad de enemigos y menos laberíntico.

Nivel 2-3: La cantidad de enemigos aumenta y el nivel es cada vez más laberíntico.

Nivel 4 (Jefe): Patrones de ataque únicos y mayor resistencia.

• Se modificó la generación de mapas para garantizar equilibrio:

Más enemigos y trampas en niveles avanzados.

6. Pruebas Finales y Ajustes:

• Se realizaron sesiones de testeo con diferentes jugadores para evaluar:

Fluidez: Sin caídas de FPS en hardware estándar.

Dificultad: Curva de aprendizaje justa.

Experiencia de usuario: Claridad en objetivos y controles.

• Se corrigieron detalles menores como:

Sonidos que no se reproducían correctamente.

Textos mal alineados en algunas resoluciones.

Sprites que parpadeaban en ciertas situaciones.

# Estadística de Tiempos:

|  |  |
| --- | --- |
| Diseño inicial | 28 horas aprox. |
| Desarrollo del núcleo | 60 horas aprox. |
| Implementación GUI | 20 horas aprox. |
| Pruebas y depuración | 10 horas aprox. |
| Documentación | 9 horas aprox. |
| Total | 127 horas |

# Conclusión:

El proyecto "Vintage Bomberman Game" cumplió satisfactoriamente con todos los requerimientos establecidos. Se logró implementar un juego funcional con:

* Logros principales:
  + Sistema completo de 4 niveles con dificultad progresiva
  + Todas las pantallas requeridas con funcionalidad completa
  + Mecánicas de juego fluidas y balanceadas
  + Documentación técnica exhaustiva
* Lecciones aprendidas:
  + Importancia de la planificación inicial en proyectos de juegos
  + Ventajas de la programación orientada a objetos para organización del código
  + Retos del manejo de tiempo real en videojuegos
* Mejoras futuras:
  + Adición de más tipos de enemigos y power-ups
  + Implementación de modo multijugador
  + Generación procedural de niveles

# Literatura o Fuentes Consultadas

Uso de Asistencia de IA:

Para el desarrollo de este proyecto, se utilizó la asistencia de ChatGPT (versión GPT-4) de OpenAI como herramienta de apoyo en las siguientes áreas:

* Orientación conceptual para la estructura del proyecto
* Solución de problemas técnicos relacionados con Pygame
* Revisión de código para optimización
* Generación de ideas para mecánicas de juego
* Asistencia en documentación y organización de contenidos
* Recursos Externos Utilizados

Gráficos y Assets Visuales:

* Arte conceptual y diseño de personajes:

PonaNers (2023). Bomberman-style character designs. ArtStation. <https://ponaners.artstation.com/projects/zWYWD>

* Animaciones y sprites:

Umplix (2024). Bomb Animation (24p DX). OpenGameArt. <https://opengameart.org/content/bomb-animation-24p-dx>

Erbarlow (2013). More NES-style RPG Characters. OpenGameArt. <https://opengameart.org/content/more-nes-style-rpg-characters>

TheTerrified (2020). 8bit Stone Brick. Pixilart. <https://www.pixilart.com/art/8bit-stone-brick-6ce8e7836c369d8>

Para la creación de assets de ítems y powerups se utilizó el uso de Sora, el modelo de generación de video e imágenes de OpenAI con el objetivo de crear diseños y sprites.

* Audio y Efectos de Sonido:

GFXSounds, Sound Effects Collection. <https://gfxsounds.com/>

Marshall Bitsmasher (2022). Marshall Bitsmasher - Cavern | 8-Bit-like Retro Music. YouTube. <https://www.youtube.com/watch?v=H85QN2YFC5M>

Sound Effect TV (2022). Wind Sound SOUND EFFECT - No Copyright[Download Free]. YouTube. <https://www.youtube.com/watch?v=5jlUVr6gkos>