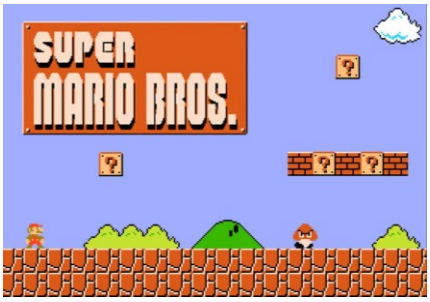
Proyecto cuatrimestral

Modelado e implementación de una versión simplificada de un juego.
Caso de estudio: Super Mario Bros.



Super Mario Bros. es un icónico videojuego de plataformas desarrollado por Nintendo. Lanzado en 1985 para la consola NES, el juego sigue las aventuras de *Mario*, un fontanero que debe rescatar a la *Princesa Peach* del malvado *Bowser*. El juego se desarrolla en el Reino Champiñón, donde *Mario* debe superar diversos niveles llenos de obstáculos, enemigos y trampas. Con su diseño de niveles ingenioso y su jugabilidad adictiva, *Super Mario Bros.* se ha convertido en uno de los videojuegos más influyentes y reconocidos de todos los tiempos.

El juego está dividido en ocho mundos, cada uno con cuatro niveles. Cada nivel presenta una combinación de plataformas, enemigos y obstáculos. *Mario* puede correr, saltar y usar *power-ups* para superar desafíos. La jugabilidad se centra en saltar sobre *plataformas* y *enemigos* para avanzar. El juego incluye varios *power-ups* como el *Super Champiñón* (que hace crecer a Mario), la *Flor de Fuego* (que permite lanzar bolas de fuego) y la *Estrella* (que otorga invulnerabilidad temporal). También existen enemigos, que varían desde *Goombas* y *Koopa Troopas* hasta el temido *Bowser*, cada uno con patrones de movimiento y ataques específicos.

A lo largo de los niveles, hay monedas y bloques que *Mario* puede romper para obtener objetos o puntos adicionales. Los jugadores obtienen puntos al recolectar monedas, derrotar enemigos y completar niveles. También hay un sistema de vidas basado en la acumulación de 100 monedas. El objetivo principal del juego es llegar al final de cada nivel y, finalmente, rescatar a la *Princesa Peach* del malvado *Bowser*.

Aunque existen infinidad de videos que muestran el juego original, en este enlace puede observar un resumen.

A partir de la descripción general realizada previamente, el proyecto consiste en el diseño e implementación en Java de una versión simplificada e inspirada en *Super Mario Bros,* en la que **se deben respetar todas y cada una de las siguientes condiciones**:

- ***Al menos 2 modos de juego***: la aplicación debe permitir, al iniciar la partida, seleccionar el modo de juego utilizado. El modo describe el dominio de aplicación a utilizar, respecto de las cuestiones gráficas. Por ejemplo, un dominio podría ser el juego original, y otro dominio, basados en otro tipo de personajes. Esto queda a libre elección, pero gráficamente deben ser bien diferentes. El comportamiento de las entidades, sin embargo, seguirá siendo el mismo independientemente del dominio de aplicación seleccionado.
- ***Al menos 3 niveles***: la aplicación debe ser desarrollada con al menos 3 niveles por los que el jugador avanza, teniendo en cuenta que en cada nivel se va incrementando la dificultad del juego. La dificultad quedará dada por la cantidad de enemigos, power-ups, la cantidad de plataformas, el tiempo disponible, etc., todo a libre elección de la comisión. Cada uno de los niveles se deben cargar desde un archivo de texto plano que describe el nivel. El formato para el archivo de texto plano de cada uno de los niveles es a libre elección de la comisión, pero debe quedar documentado en un archivo *Formateo-de-niveles.pdf*, donde se describa claramente.
- ***Al menos 6 enemigos***: el juego debe permitir interactuar con al menos los 6 enemigos descriptos en la *Tabla 1*.

- ***Al menos 4 power-ups***: el juego debe permitir interactuar con al menos los 4 power-ups descriptos en la *Tabla 2*.
- ***Al menos 5 plataformas***: el juego debe permitir moverse sobre al menos las 5 plataformas de la *Tabla 3*.
- ***Patrones de diseño:*** se espera que, en el modelado e implementación, se observe el uso claro y no forzoso de al menos un patrón de diseño creacional y un patrón de diseño de comportamiento.
- ***Transiciones y movimientos:*** todos los efectos visuales respecto de movimientos deben ser implementados con *Threads* (ninguna otra opción será considerada válida).
- ***Efectos e implementación de cuestiones gráficas:*** el desarrollo gráfico del proyecto debe realizarse usando únicamente *Java Swing*. Se espera que el proyecto no requiera de librerías adicionales a las por defecto en Java.
- ***Jugador:*** el juego permitirá que el usuario pueda mover únicamente al personaje *Mario*, el resto se moverán e interactuarán automáticamente. Para mover a *Mario*, se deberán utilizar las teclas AWD (para los movimientos a izquierda, saltar, derecha, respectivamente), y la barra espaciadora para el lanzamiento de bolas de fuego.
- ***Información disponible:*** el juego debe mostrar, en todo momento, el nivel actual, puntaje acumulado y tiempo disponible antes de finalizar la partida. Los puntajes deben observarse en la *Tabla 4*.
- ***Ranking:*** el juego debe permitir llevar control, de forma persistente, un ranking con el Top 5 de jugadores de la aplicación. Los jugadores deben tener nombre y puntaje asociado. El ranking debe poder accederse en la pantalla inicial de la aplicación (cuando se selecciona el modo de juego) y al finalizar una partida (si es que el jugador debe ser registrado en el Top 5).
- ***Sonidos:*** se espera que el juego presente diferentes sonidos cuando el personaje cambia de estado, interactúa con las restantes entidades, etc.

Tabla 1: Tipos de enemigos requeridos.

Enemigo	***Comportamiento***
Goomba	*Hongo que se mueven lentamente de un lado a otro. Se pueden derrotar saltando una vez sobre ellos.*
Koopa Troopa	*Son Tortugas que caminan de un lado a otro y tienen un caparazón en la espalda. Se pueden derrotar saltando dos veces sobre ellas.*
Piranha Plant	*Plantas que emergen de tuberías y muerden a Mario si entra en contacto con ellas. Son invulnerables cuando están retraídas en la tubería. Solo pueden derrotarse tirando bolas de fuego.*
Lakitu	*Enemigo volador que lanza Spinys desde una nube. Se pueden derrotar saltando una vez sobre ellos.*
Spiny	*Erizos que caminan por el suelo y tienen pinchos en la parte superior. No se pueden derrotar con un salto, pero pueden ser eliminados con bolas de fuego.*
Buzzy Beetle	*Escarabajos con caparazones duros que caminan por el suelo. Se pueden eliminar con bolas de fuego o golpeando una vez su caparazón.*

Tabla 2: Tipos de power-ups requeridos

Power-up	***Efecto***
Super Champiñón	*Hace que Mario crezca y se convierta en Super Mario. Le permite romper bloques de ladrillo y recibir un golpe adicional antes de perder una vida.*
Flor de Fuego	*Permite a Mario lanzar bolas de fuego que pueden eliminar enemigos y destruir ciertos bloques. Mario debe estar en estado Super Mario para usarla.*
Estrella	*Otorga a Mario invulnerabilidad temporal, durante la cual puede eliminar enemigos al tocarlos y es inmune a daños.*
Champiñón Verde	*Da una vida extra a Mario. Suele aparecer en bloques ocultos y en ubicaciones especiales.*

Tabla 3: Tipos de plataformas.

Plataforma	***Descripción de la regla***
Bloque solido	*Permiten que los personajes puedan moverse libremente sobre ellos. No pueden atravesarse ni romperse, en ningún caso.*
Ladrillo sólido	*Se comporta igual que el bloque sólido. Incorpora la siguiente funcionalidad: Mario puede romperlos solamente si se encuentra en estado Super Mario y lo golpea por abajo.*
Bloque de Pregunta	*Se comporta igual que el bloque sólido. Incorpora la siguiente funcionalidad: si Mario golpea este bloque por abajo, puede obtener monedas o power-ups, si es que las contiene.*
Tuberías	*Estructuras verticales. Se encuentran sobre el suelo. Algunas de ellas pueden contener al enemigo Piranha Plant. Mario puede caminar y saltar sobre las mismas. No puede romperse. Las tuberías no trasladas a Mario a ningún nivel oculto.*
Vacío	*Representa un hueco dentro del mapa. Mario puede caer en dichos orificios y perder una vida.*

Tabla 4: Puntajes.

Power-up / Enemigo / Plataforma	***Puntaje asociado en caso de verse afectado***
Super Champiñón	*+10 en estado normal; +50 si Mario está en estado Super Mario.*
Flor de Fuego	*+5 en estado normal; +30 si Mario está en estado Super Mario; +50 si Mario está bajo los efectos de una flor de fuego previa.*
Estrella	*+20 en estado normal; +35 si Mario está bajo los efectos de una estrella previa.*
Champiñón Verde	*+100 en cualquier estado.*
Moneda	*+5 por cada moneda consumida.*
Goomba	*+60 por cada uno destruido; -30 por cada muerte de Mario a manos de este enemigo.*
Koopa Troopa	*+90 por cada uno destruido; -45 por cada muerte de Mario a mano de este enemigo.*
Piranha Plant	*+30 por cada una destruida; -30 por cada muerte de Mario a mano de este enemigo.*
Lakitu	*+60 por cada uno destruido.*
Spiny	*+60 por cada uno destruido; -30 por cada muerte de Mario a mano de este enemigo.*
Buzzy Beetle	*+30 por cada uno destruido; -15 por cada muerte de Mario a mano de este enemigo.*
Vacío	*-15 por cada muerte de Mario a mano de este enemigo.*

Tips adicionales:

- ***Dominio de aplicación***: los dos temas del juego son de libre elección, y no debe verse necesariamente como el juego original. Cada comisión tiene libertad de modificar el *look and feel* respetando las características y comportamiento de todas las entidades.
- ***Desarrollo de cuestiones gráficas:*** el proyecto puede profundizar todo lo que deseen en las cuestiones gráficas, pero partiendo de la base que todo proyecto bien modelado, será calificado positivamente cuanto mejor desarrollo gráfico tenga. Sin embargo, no vale la inversa: todo proyecto que visualmente se muestre excelente, pero esté mal modelado, estará desaprobado. Moraleja: primero se modela bien, se valida con el cuerpo docente, luego (en caso de existir tiempo disponible) se pueden dedicar a mejorar la estética.
- ***Casos no contemplados***: toda cuestión que no se encuentre descripta en este enunciado, por lo general, seguirá la política original del juego. Sin embargo, antes de avanzar en cualquier cuestión no indicada, consultar y solicitar autorización con el ayudante asignado o el asistente de la materia.

El cuerpo docente de TdP se reserva el derecho de modificar los requerimientos del presente proyecto, durante el cuatrimestre, en función de las necesidades del curso.

Condiciones de aprobación y entrega:

- El proyecto cuenta con **Fecha límite de entrega** y **Fecha límite de reentrega**.
- **Desaprobada** la entrega, **deberá reentregarse** el proyecto con las correcciones de **todas las observaciones** realizadas por el ayudante a cargo.
- **Desaprobada** la reentrega, el cursado de la materia estará desaprobado.
- El proyecto cuenta con **7 defensas obligatorias y de evaluación individual**. Cualquier alumno que desapruebe o se ausente (con o sin justificación) en una o más defensas, **deberá recuperar el o las defensas en el recuperatorio escrito**. Los temas evaluados en el recuperatorio escrito serán indicados oportunamente, y dependerán de la cantidad de defensas desaprobadas/ausentes.
- Las **condiciones de evaluación y entrega** para cada una de las defensas obligatorias serán publicadas con un mínimo de 7 días de anticipación, en un enunciado particular, siempre dentro del Aula Virtual. Esta regla no aplica para la Defensa 1, donde los alumnos solo debatirán con el ayudante a cargo las características y objetivos del proyecto, trazando algunos tips para su inicio.
- El proyecto será calificado con una escala de nota A-B-C-D, donde A, B y C indican proyectos aprobados, mientras que D indica un proyecto desaprobado. La **nota del proyecto impactará en la nota de promoción**, en caso de corresponder.
- Toda entrega de código fuente, diagramas o cualquier requerimiento adicional respecto del proyecto, debe **seguir estrictamente (sin ningún tipo de excepción)** lo indicado en cada enunciado de cada una de las defensas, y será realizado siempre (**sin ningún tipo de excepción**) a través del repositorio privado creado para cada comisión de cursado, dentro de la organización de la materia en GitHub. No cumplir con estas condiciones en tiempo y forma implicarán **desaprobar la defensa correspondiente**.
- A la hora de realizar la corrección del proyecto, **se analizará específicamente el nivel de participación de cada uno de los integrantes de la comisión de cursado**. En caso de detectar que un alumno no realizó *commits* para la concreción del proyecto, o que el contenido de los mismos no impliquen un estimado del 20% del desarrollo del proyecto, el alumno desaprobará automáticamente el mismo.
- El asistente de la materia se guarda derecho de **solicitar cualquier instancia especial de justificación** por parte de uno o más alumnos, cuando el nivel de participación en el proyecto no quede claro o sea deficiente. No asistir o no contestar satisfactoriamente las consultas en dichas instancias de justificación, implican la **desaprobación del cursado de la materia.**