

Desarrollo de Aplicaciones Móviles para Android

Dr. Marco Aurelio Nuño-Maganda

Universidad Politecnica de Victoria

https://github.com/mnunom-upv/Curso_Desarrollo_Aplicaciones_Moviles_2023

Noviembre 2025



1 Tic-Tac-Toe

2 Pong

1 Tic-Tac-Toe

2 Pong

Tic-Tac-Toe: La Estrategia de la Sencillez

Mecánica y Tablero

- Es un juego para **dos jugadores** que se desarrolla en un tablero de 3×3 casillas.
- Un jugador marca con "equis" y el otro con "círculo".

Objetivo del Juego

- El objetivo es ser el primero en colocar tres de sus símbolos de forma consecutiva.
- Las combinaciones ganadoras pueden ser **horizontales, verticales o diagonales**.

Desarrollo de la Partida

- Los jugadores se turnan para colocar un símbolo en una casilla vacía.
- Si el tablero se llena y nadie logra la línea de tres, la partida se declara **empate** (o *gato*).

Análisis Estratégico

Se usa a menudo como una herramienta pedagógica para introducir conceptos de **teoría de juegos** y búsqueda de árboles de decisión en programación.

1 Tic-Tac-Toe

2 Pong

Creador y Compañía

- Desarrollado para la empresa **Atari** en 1972, convirtiéndose en el **primer videojuego de éxito comercial masivo**.
- El concepto se inspira en el deporte del tenis de mesa (ping pong).

Revolución Doméstica (1975)

- Atari lanzó una versión casera, *Home Pong*, vendida exclusivamente a través de las tiendas Sears durante la temporada navideña de 1975.
- Esto introdujo los videojuegos en el **hogar** y provocó el nacimiento del mercado de consolas domésticas.

Legado Técnico y Cultural

- A pesar de su simplicidad gráfica (dos paletas y una pelota), sentó las bases para mecánicas de juego fundamentales y estableció a los videojuegos como una **forma de entretenimiento viable y lucrativa**.

1 Configuración de Pantalla Completa

- Se asegura una experiencia inmersiva eliminando la barra de título (FEATURE_NO_TITLE) y la barra de acción (supportActionBar.hide()), y estableciendo el modo de **pantalla completa** (FLAG_FULLSCREEN).

2 Motor de Juego Principal (GameView)

- La lógica del juego (dibujo, física, bucle de actualización) reside en una vista personalizada (GameView), la cual se añade dinámicamente al contenedor (FrameLayout) de la actividad.

3 Comunicación y Actualización de Puntaje

- Se establece un **Listener** (ScoreUpdateListener) en la GameView para recibir los nuevos puntajes.
- La actualización de los TextView de los jugadores se realiza de forma segura en el **Hilo Principal** de Android (runOnUiThread) para evitar fallos en la interfaz de usuario.

4 Manejo del Ciclo de Vida del Juego

- La actividad maneja el estado del juego mediante los métodos onPause() y onResume(), llamando a gameView.pauseGame() y gameView.resumeGame() respectivamente. Esto es esencial para **conservar recursos** al cambiar de aplicación.

La clase GameView es la superficie de dibujo principal y el motor de lógica de la aplicación.

1 Gestión de Hilos (Threading)

- Contiene un **GameThread** que ejecuta el bucle de juego para **actualizar la lógica** (`update()`) y (`draw()`) en un ciclo constante.
- Esto asegura que las animaciones se ejecuten fluidamente sin bloquear el Hilo Principal (UI).

2 Implementación de la Lógica del Juego

- Maneja la posición y el movimiento de los objetos (pelotas, paletas, etc.) y la detección de colisiones.
- Es responsable de incrementar los contadores de puntaje tras eventos clave, como anotar un punto.

3 Control de Estado y Comunicación

- Utiliza la interfaz `ScoreUpdateListener` para notificar a la actividad sobre los cambios de puntaje, separando la lógica del juego de la actualización de la UI.