

M8 Modelado orientado a objetos FIN A

Actividad 1

| **Tutor:** | **Melissa Gamez Whijares** |
| --- | --- |
| **Estudiante:** | **José Ramón Ibáñez Posadas** |
| **Matricula:** | **BNL098377** |

| Monterrey, Nuevo León | domingo, 17 de agosto de 2025 |
| --- | --- |

Introducción

El mundo de los videojuegos ha experimentado una transformación asombrosa desde sus humildes orígenes. Uno de los hitos más tempranos y significativos fue el "Tennis for Two" de William Higinbotham en 1958, una simple simulación de tenis en un osciloscopio que marcó el nacimiento de una industria.

A partir de ahí, la evolución ha sido exponencial. Desde los gráficos de píxeles y las mecánicas simples de juegos como Pong y Space Invaders en la década de 1970, hasta los intrincados mundos en 3D y las narrativas complejas de hoy en día, cada avance tecnológico ha permitido una experiencia de juego más inmersiva y realista. La llegada de las consolas domésticas como la Atari 2600 y la Nintendo Entertainment System popularizó el medio, mientras que la revolución de las computadoras personales abrió la puerta a una nueva generación de juegos.

La transición a gráficos 3D, el desarrollo de sistemas de juego en red y la incorporación de tecnologías como la realidad virtual han redefinido por completo lo que significa interactuar con un videojuego.

Desarrollo

**Investigación**

La Programación Orientada a Objetos (POO) es un paradigma de programación que se ha convertido en la base del desarrollo de software moderno, incluidos los videojuegos. Sus orígenes se remontan a la década de 1960 con el lenguaje de programación Simula, diseñado para simulaciones. Sin embargo, no fue hasta la llegada de Smalltalk en la década de 1970 que los conceptos de objetos, clases, herencia y polimorfismo se formalizaron y popularizaron. Estos pilares de la POO permiten a los desarrolladores estructurar el código de manera que imite el mundo real, organizando los datos y las funciones en entidades llamadas "objetos". (Fuente: Tema 3, "Definición de clases")

Un objeto es una instancia de una clase, que actúa como un plano o plantilla para crear objetos. Un objeto en Java puede tener propiedades (campos) y acciones (métodos). Por ejemplo, en un juego, una clase "Personaje" podría definir atributos como la vida y la velocidad, y métodos como "mover" o "atacar". Cada personaje del juego sería un objeto individual de esa clase, con sus propios valores para la vida y la velocidad. (Fuente: Tema 2, "Objetos y clases")

La evolución de la POO ha sido crucial para la complejidad de los videojuegos actuales. En lugar de utilizar una programación lineal o estructurada, que puede volverse inmanejable a medida que los proyectos crecen, la POO permite a los desarrolladores crear componentes modulares y reutilizables. Esto es fundamental para los videojuegos, donde se necesitan múltiples instancias de los mismos elementos, como enemigos, proyectiles o elementos del entorno. La herencia es un concepto clave que permite crear una clase "EnemigoAéreo" que herede las propiedades y métodos de una clase "Enemigo" más genérica, lo que reduce la redundancia de código y facilita el mantenimiento. El polimorfismo permite que diferentes objetos respondan de manera distinta al mismo mensaje, lo que significa que un método "atacar" puede tener un comportamiento diferente si lo llama un personaje o un monstruo. (Fuente: Tema 4, "Interacción de objetos")

Según James Gosling, uno de los creadores de Java, la POO fue concebida para manejar la complejidad inherente a los grandes sistemas de software. Esta filosofía de diseño encaja perfectamente con el desarrollo de videojuegos, donde la interacción entre miles de objetos es constante. Desde la gestión de la física y la inteligencia artificial de los personajes hasta la renderización de gráficos y la administración de los menús, la POO permite un desarrollo escalable y organizado.

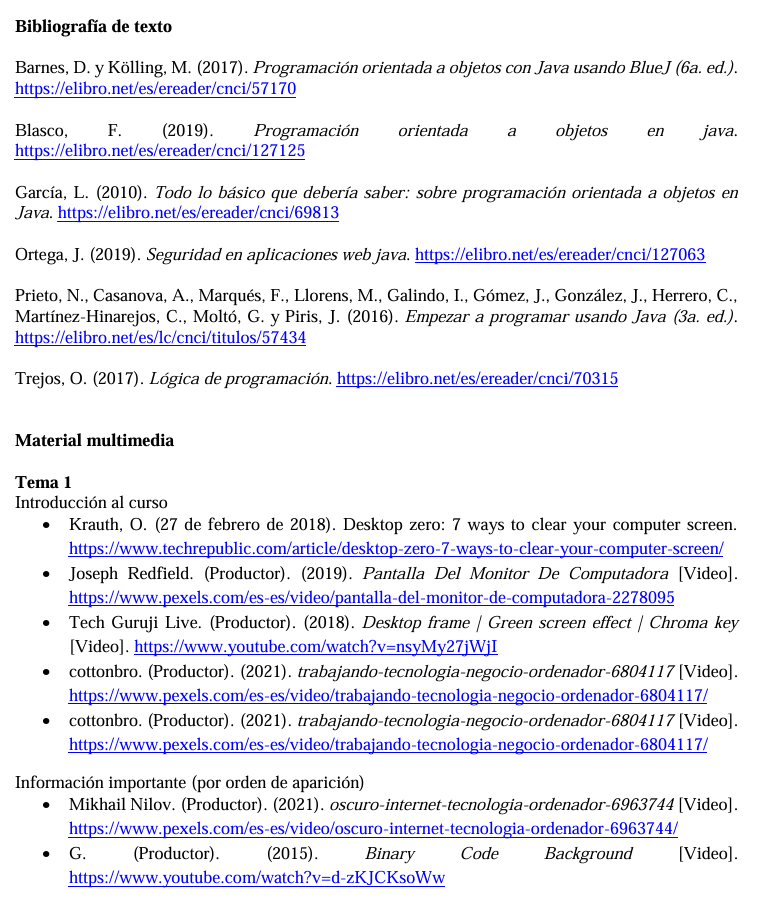
Además de los conceptos mencionados, la encapsulación ha sido fundamental. Esta propiedad de la POO permite ocultar los detalles internos de un objeto y exponer solo una interfaz controlada. Por ejemplo, la clase "Jugador" puede tener un método público reducirVida() que es seguro para ser llamado por otros objetos (como un enemigo), mientras que la variable interna salud permanece privada. Esto previene manipulaciones de datos no deseadas y hace que el código sea más robusto.

En resumen, la evolución de la POO ha pasado de un concepto de nicho en la década de 1960 a convertirse en el estándar de facto para el desarrollo de software complejo. Ha proporcionado a los desarrolladores de videojuegos las herramientas necesarias para gestionar la complejidad, mejorar la modularidad y crear mundos virtuales cada vez más ricos y dinámicos.

Conclusión

La Programación Orientada a Objetos ha influido de manera fundamental en la evolución de los videojuegos. La imagen que compara los gráficos antiguos con los modernos ilustra perfectamente este punto. Mientras que los juegos de la era inicial se basaban en la manipulación directa de gráficos de píxeles, la complejidad del diseño actual demanda una estructura de código más sofisticada. La POO ha permitido que los desarrolladores piensen en el mundo del juego como una colección de objetos que interactúan entre sí. Por ejemplo, cada personaje, arma o elemento del escenario puede ser una instancia de una clase, con sus propias propiedades y comportamientos definidos. Esta modularidad simplifica la creación de mundos virtuales expansivos y dinámicos. Sin la POO, sería prácticamente imposible gestionar la enorme cantidad de datos y operaciones que requiere un videojuego moderno, desde la física realista hasta la inteligencia artificial de los enemigos. La herencia permite que un diseñador de juegos cree una jerarquía de objetos, donde un "árbol" es una clase padre y "pino", "roble" y "palmera" son clases hijas que heredan sus propiedades. Esto ahorra tiempo y esfuerzo, y es lo que ha hecho posible la increíble riqueza visual y la complejidad de los juegos de hoy. En esencia, la POO es el andamiaje invisible que sostiene y da vida a los mundos que vemos en pantalla.

**Bibliografía**

****