

## Proyecto Pong en OpenGL Moderno - Variantes y Desafíos

**Fecha de Entrega y Presentación: Jueves 09 de Mayo**

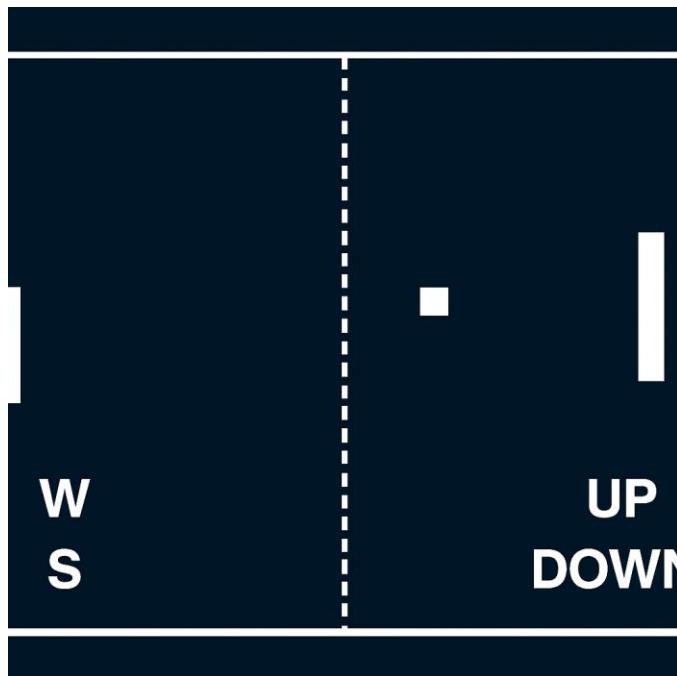
Este proyecto tiene como objetivo aplicar conceptos de OpenGL moderno mediante la creación de variantes del juego Pong. Cada estudiante o equipo de dos personas debe desarrollar una de las siguientes variantes. La base para este trabajo incluye los ejemplos de barra móvil y pelota con rebote, los cuales ya han sido estudiados previamente.

### **Variante 1: Pong con control de ambas barras (jugador vs jugador) Valentín - Nori**

Se implementa un modo multijugador local en el que ambos jugadores pueden controlar sus respectivas barras: el jugador 1 con 'W' y 'S', y el jugador 2 con las flechas '↑' y '↓'. La pelota rebota entre las barras, y se incrementa la puntuación cada vez que uno de los jugadores falla.

Desafíos técnicos:

- Implementar control simultáneo para dos entradas independientes.
- Añadir sistema de puntuación simple.



## **Variante 2: Aceleración progresiva de la pelota cada 5 rebotes (Ignacio Alfaro - Félix Fuentes)**

La velocidad de la pelota aumenta después de cada 5 colisiones con las barras. Esta aceleración puede hacerse multiplicando la velocidad original por un factor (por ejemplo, 1.1).

Desafíos técnicos:

- Implementar un contador de rebotes.
- Ajustar la velocidad sin afectar la dirección.

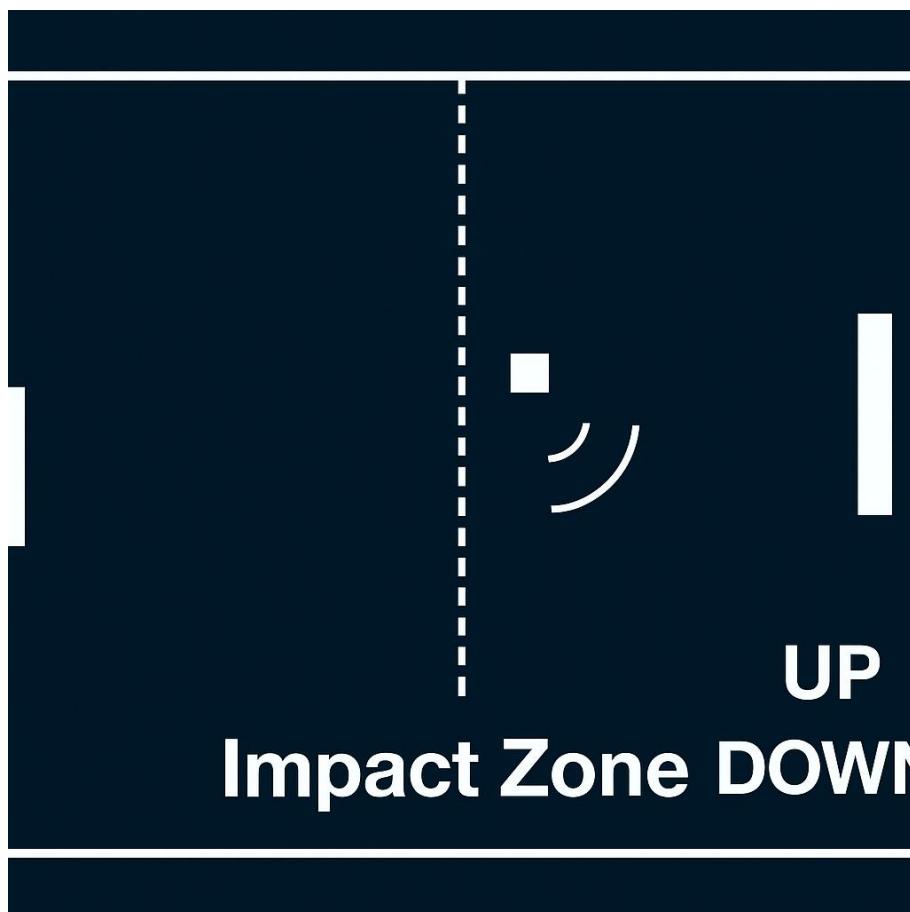


### Variante 3: Rebote según zona de impacto en la barra (Pedro Morales)

La dirección del rebote de la pelota depende del punto de impacto sobre la barra. Si impacta en el centro, rebota recto; si impacta en los extremos, el ángulo de rebote será más inclinado.

Desafíos técnicos:

- Calcular el desplazamiento relativo entre el centro de la barra y el punto de impacto.
- Ajustar el ángulo de rebote en función de esa distancia.





#### Variante 4: Obstáculos estáticos en pantalla (Iván Obando - Raúl Sobarzo)

Se colocan uno o más rectángulos fijos en el centro o zonas aleatorias del campo de juego. La pelota debe rebotar contra estos obstáculos, además de las barras y los bordes.

Desafíos técnicos:

- Detectar colisiones con múltiples objetos.
- Usar estructuras de datos para almacenar las posiciones y dimensiones de los obstáculos.



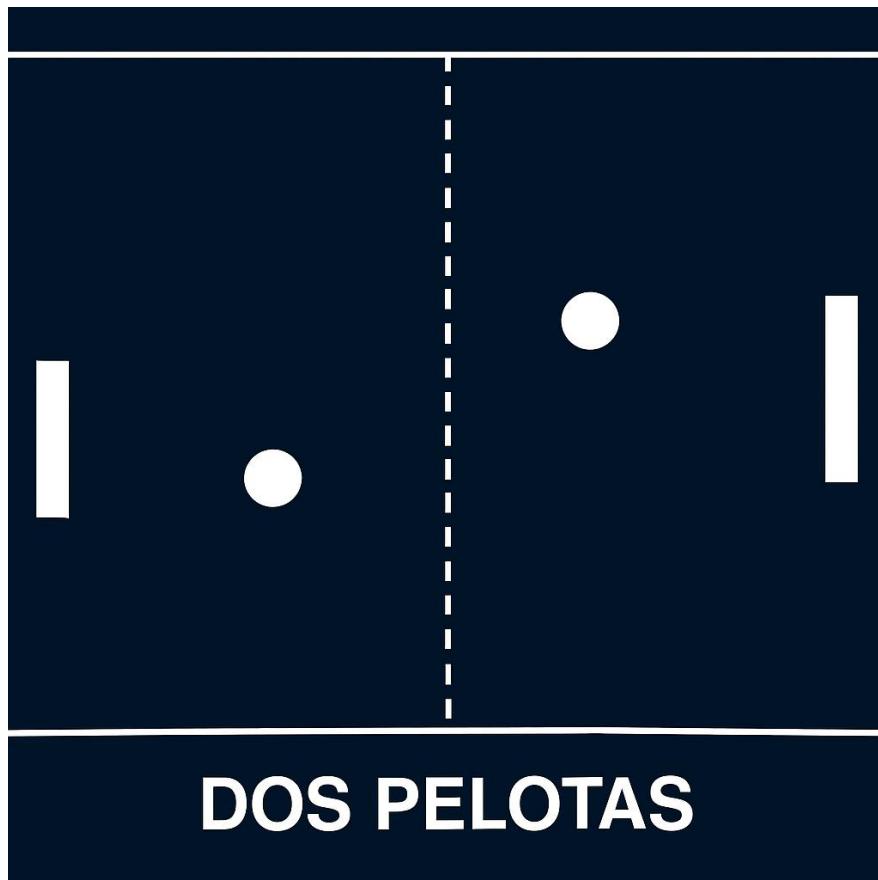


### Variante 5: Dos pelotas simultáneas (Rodrigo Diaz - Simón Gallardo)

El juego inicia con dos pelotas que se mueven simultáneamente en direcciones distintas. Ambas deben rebotar en las barras y bordes. El jugador pierde si falla cualquiera de ellas.

Desafíos técnicos:

- Manejo de múltiples objetos móviles.
- Control independiente de velocidad, dirección y rebote para cada pelota.



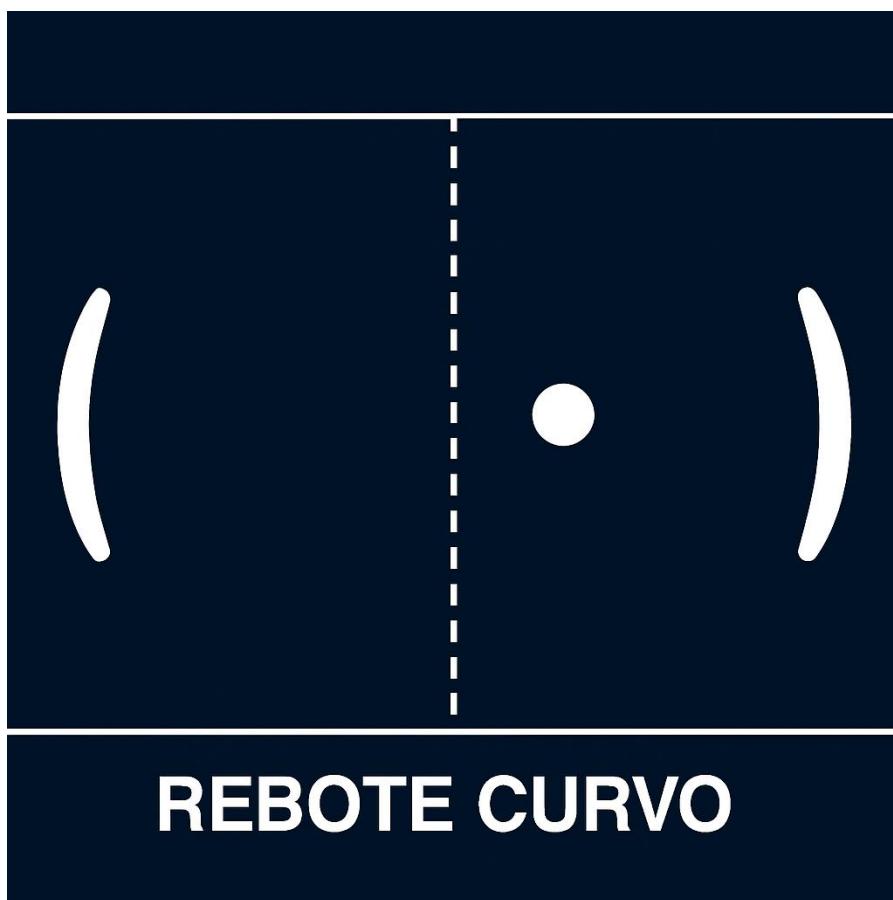


### Variante 6: Rebote curvo (barras curvas)

Las barras tienen forma curva (parabólica visualmente). El rebote de la pelota se calcula considerando la tangente al punto de impacto de la curva.

Desafíos técnicos:

- Representación visual curva mediante vértices o shaders.
- Cálculo del ángulo de rebote basado en la curva (trigonometría, normal/tangente).





### Variante 7: Fondo dinámico (color que cambia al rebotar) (Bastián Correa - Diego Lara)

El color del fondo cambia aleatoriamente o de forma programada (ciclo de colores) cada vez que la pelota rebota en una barra.

Desafíos técnicos:

- Control y actualización de `glClearColor` en cada frame.
- Uso de variables globales o uniformes para animar color.





### Variante 8: Pong multibarra (dos barras por jugador) (Fernando González - Emanuel Diaz)

Cada jugador controla dos barras simultáneamente: una arriba y una abajo. Pueden moverse sincronizadas o con controles separados.

Desafíos técnicos:

- Dos instancias por jugador, cada una con su posición, colisión y renderizado.
- Definir controles combinados o diferenciados.





### Variante 9: Movimiento en diagonal de las barras (modo avanzado) (Matías Ortega - Seba Oñate)

Las barras pueden moverse en dirección diagonal, es decir, no solo 'arriba/abajo' sino también 'izquierda/derecha', con cuatro teclas por jugador (por ejemplo: 'WASD').

Desafíos técnicos:

- Movimiento bidimensional para las barras.
- Lógica adicional para mantenerlas dentro de límites sin solaparse.



## Rúbrica de Evaluación - Proyecto Pong en OpenGL

### Funcionamiento general (20 pts)

- 0 pts - El programa no compila o no corre.
- 10 pts - Corre con errores graves en las funcionalidades.
- 15 pts - Corre con funcionalidades principales implementadas.
- 20 pts - Cumple completamente los requisitos y funciona correctamente.

### Comentarios y estilo del código (10 pts)

- 0 pts - Código sin comentarios o ilegible.
- 4 pts - Comentarios escasos o poco útiles.
- 7 pts - Comentado pero con inconsistencias de estilo.
- 10 pts - Comentarios claros, estilo uniforme y legibilidad alta.

### Casos de prueba o validaciones (10 pts)

- 0 pts - Sin pruebas ni validaciones.
- 4 pts - Validaciones mínimas.
- 8 pts - Pruebas y validaciones parciales.
- 10 pts - Validaciones robustas y pruebas variadas incluidas.

### Presentación: Claridad y organización (10 pts)

- 0 pts - Presentación desorganizada o incompleta.
- 5 pts - Presentación básica o poco clara.
- 8 pts - Bien estructurada pero falta profundidad.
- 10 pts - Presentación clara, completa y bien organizada.

### Duración de la presentación (5 pts)

- 0 pts - Presentación menor a 5 min o mayor a 12 min.
- 2 pts - Presentación entre 6 y 7 o entre 11 y 12 minutos.
- 4 pts - Presentación entre 7 y 8 o entre 10 y 11 minutos.
- 5 pts - Duración entre 8 y 10 minutos, como solicitado.

### Respuestas a preguntas del docente (10 pts)

- 0 pts - Sin respuestas o incorrectas.
- 5 pts - Respuestas superficiales o vagas.
- 8 pts - Correctas pero sin mucha profundidad.
- 10 pts - Respuestas claras, justificadas y argumentadas.

### Precisión técnica (10 pts)

- 0 pts - Errores lógicos graves.
- 5 pts - Implementación con errores menores.
- 8 pts - Implementación correcta en su mayoría.
- 10 pts - Implementación precisa y técnicamente sólida.

### Capacidad crítica (10 pts)

- 0 pts - No se justifica la elección de estructuras.
- 4 pts - Justificación vaga o poco técnica.
- 8 pts - Justificación razonable pero con fallas.
- 10 pts - Justificación clara, lógica y bien fundamentada.

## TIPS ADICIONALES

### Tips Avanzados por Variante - Proyecto Pong en OpenGL

#### Variante 1: Pong con control de ambas barras (jugador vs jugador)

- 🎮 Usa `glfwGetKey(window, GLFW\_KEY\_W)` y `GLFW\_KEY\_S` para la barra izquierda, y `GLFW\_KEY\_UP`/`DOWN` para la derecha.
- 💡 Duplica la lógica y VAO del archivo `barra.cpp` para que cada jugador tenga una barra independiente.

#### Variante 2: Aceleración progresiva de la pelota cada 5 rebotes

- ⚡ Crea una variable `int contadorRebotes`.
- 📈 Aumenta velocidad con `velocidad \*= 1.05f` cada vez que `contadorRebotes % 5 == 0`.

#### Variante 3: Rebote según zona de impacto en la barra

- 📐 Usa `(yPelota - yCentroBarra) / alturaBarra` para calcular el rebote relativo.
- ↵ Ajusta el ángulo de salida con un factor proporcional de desviación.

#### Variante 4: Obstáculos estáticos en pantalla

- 🧱 Representa cada obstáculo con `struct Obstacle { float x, y, w, h; };`.
- 🔍 Detecta colisiones AABB con cada obstáculo antes de mover la pelota.

#### Variante 5: Dos pelotas simultáneas

- 📅 Usa `struct Pelota { float x, y, vx, vy; };` y un array para múltiples pelotas.

- ▣ Itera sobre ellas en cada frame y aplica la lógica de movimiento y colisión por separado.

#### Variante 6: Rebote curvo (barras curvas)

- ▎ Representa la barra curva con más vértices siguiendo una parábola `y = -a(x - h)^2 + k`.
- 📐 Para calcular el rebote: estima tangente entre dos puntos consecutivos, luego aplica `n = (-t.y, t.x)` y usa reflexión `v - 2 \* dot(v, n) \* n`.
- ⌚ Puedes usar shaders para animar visualmente la curvatura, aunque requiere más experiencia con GPU.

#### Variante 7: Fondo dinámico (color que cambia al rebotar)

- 🌈 Usa `glClearColor(r, g, b, 1.0f)` con colores que cambian en cada rebote.
- 🎲 Genera color con: `rand() % 256 / 255.0f` o usa una lista de colores cíclica.

#### Variante 8: Pong multibarra (dos barras por jugador)

- ⭐ Declara dos VAOs por jugador y renderiza ambas barras.
- ↔ Control conjunto (sincronizado) o separado por teclas. Verifica colisión con cualquiera de las barras.

#### Variante 9: Movimiento en diagonal de las barras

- ✳️ Añade `offsetX` además de `offsetY` para mover la barra en 2D.
- ▲ Usa `WASD` o flechas y limita ambos ejes con `clamp()`.
- 💡 Puedes pasar la posición como `vec2` a shaders con `glUniform2f`.