Manual del usuario del Juego CocheBomba

*Cristian Darwin Flores Cueva (*cdfloresc@est.unap.edu.pe*)*
GoDot

Perú, 2025

Resumen

CocheBomba es un juego de combate vehicular desarrollado en **Godot 4**, Combina:

- Simulación de física realista de vehículos
- IA básica de conducción (con posibilidad de entrenar aprendizaje por refuerzo)
- Sistema de armas (hitscan y proyectiles)
- Editor de pistas mediante nodos `Path3D`
- Efectos visuales (partículas, *decals*, *shaders*)
- Recursos creados en Blender, Krita y Stable Diffusion

Este manual cubre desde la instalación hasta cómo extender y compilar el proyecto.

Índice

1. Requisitos del sistema

1. Requisitos de **hardware mínimo**:
   - CPU: `quad-core 2.0GHz`
   - GPU: compatible con `OpenGL 3.3` o `Vulkan`
   - RAM: `8GB`
2. Requisitos de **software**:
   - **Godot 4.0+**
   - Editor C#: `Visual Studio 2022` o `Rider` (opcional)
   - `Android SDK & NDK` (para exportar a APK)
   - `Java11+`
3. Espacio libre en disco:

> Se recomienda disponer de al menos `~500MB` para el proyecto base, sin contar assets adicionales.

2. Instalación y puesta en marcha

1. **Clona o descomprime** el proyecto bajo el nombre de carpeta `CocheBomba/`.
2. Abre Godot y pulsa **"Import"**, apunta a `CocheBomba/project.godot`.
3. Revisa en Project → Project Settings → Paths que las rutas a `addons/` y `res://` sean correctas.
4. Ejecuta la escena principal `res://world.tscn` con F5.

> **Nota:** Si quieres compilar los módulos C#, abre `Godot RL Agents.sln` en Visual Studio y genera la solución antes de volver a Godot.

3. Controles y navegación

Acción	Tecla / Botón
Acelerar	W / ↑
Frenar / Marcha atrás	S / ↓
Girar izquierda	A / ←
Girar derecha	D / →
Cambiar cámara	C
Disparar arma primaria	Click izquierdo
Disparar arma secundaria	Click derecho
Pausa / Menú	Esc

Figura 1: Carrera

4. Menús y HUD

4.1. Menú principal

- **Start Game**: inicia `world.tscn`.
- **Options**: ajustes de gráficos y audio.
- **Exit**: cierra la aplicación.

Listing 1: MainMenu.gd

```
extends Control

func _on_StartButton_pressed():
    get_tree().change_scene("res://world.tscn")

func _on_ExitButton_pressed():
    get_tree().quit()
```

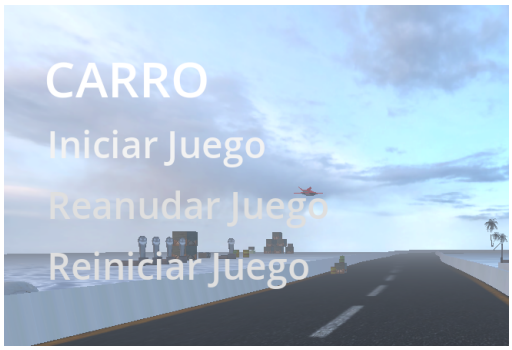

Figura 2: Menú Principal

4.2. HUD en partida

Contiene velocidad, salud del vehículo y munición. Se gestiona en `HUD.gd`:

Listing 2: HUD.gd

```
extends CanvasLayer

onready var speed_label = $SpeedLabel

func _process(delta):
    speed_label.text = str(
    round(get_parent()
    .car.velocity.length()
    * 3.6)) + " km/h"
```

5. Física de vehículos

La clase `GameCar.gd` extiende `VehicleBody3D` para simular suspensión, frenos y motor:

Listing 3: GameCar.gd

```
extends VehicleBody3D

@export var max_engine_force = 2000.0
@export var max_brake_force = 50.0

func _physics_process(delta):
    var throttle = Input.get_action_
    strength("ui_up") -
    Input.get_action_strength
    ("ui_down")
    engine_force =
    max_engine_force * throttle
    brake = max_brake_force *
    Input.is_action_pressed("ui_brake")
```

Figura 3: Suspensión activa

6. IA y conducción automática

En `script` se define un esqueleto para IA:

Listing 4: AIController.gd

```
extends Node

@export var target_path: Path3D

func _ready():
    var path = target_path.curve
    # recorrer puntos y aplicar torque / fuerza
```

Puedes conectar tu red neuronal usando el plugin de C# en `Godot RL Agents.sln`.

7. Armamento y proyectiles

- **Hitscan**: dispara rayos instantáneos.
- **Proyectiles**: instancian `Projectile.tscn` con `RigidBody3D`.

Listing 5: Weapon.gd

```
func shoot_hitscan():
    var from = muzzle.global_transform
       .origin
    var to   = from + -muzzle.
       global_transform.basis.z * 1000
    var space = get_world_3d().
       direct_space_state
    var result = space.intersect_ray(
       from, to)
    if result:
        spawn_impact(result.position)

func shoot_projectile():
    var proj = preload("res://
       Projectile.tscn").instantiate()
    proj.global_transform = muzzle.
       global_transform
    proj.apply_impulse(Vector3.ZERO, -
       muzzle.global_transform.basis.z
        * 200)
    get_parent().add_child(proj)
```

8. Creación y edición de pistas

1. Crea un nodo `Path3D` en tu escena.
2. Dibuja el recorrido con puntos de curva.
3. Instancia `road.gdshader` sobre un `MeshInstance3D` y asigna el Path para deformar la carretera.

Listing 6: road.gdshader (simplificado)

```
shader_type spatial;
uniform Path3D path;
void vertex() {
    // transforma v rtices siguiendo path.curve
}
```

9. Estructura de archivos y cómo extender el proyecto

```
CocheBomba/
 addons/                  # Plugins (debug, script-ide)
 assets/
    models/             # .glb / .obj
    textures/           # .png
    sounds/             # .ogg / .wav
    screens/            # capturas para manual
 script_templates/AIController/
 src/
    GameCar.gd
    world.gd
    HUD.gd
 world.tscn
 MainMenu.tscn
 project.godot
 export_presets.cfg
```

Para añadir un nuevo vehículo:

1. Copia `src/GameCar.gd` como `MyCar.gd`.
2. Crea `MyCar.tscn` con `VehicleBody3D` y asigna el script.
3. Ajusta parámetros de suspensión y frenos en el Inspector.

10. Generación de ejecutables y APK

10.1. Exportar a Windows (EXE)

1. En Godot, ve a Project → Install Export Templates (si no lo has hecho).
2. Abre Project → Export y añade el "Windows Desktop" preset.
3. Configura iconos y ajustes. Pulsa **Export Project** y guarda `CocheBomba-win64.zip`.
4. Sube `CocheBomba-win64.zip` a tu GitHub Releases y copia el enlace:

11. Descarga

Si encuentras bugs o quieres sugerir mejoras:

- **Descarga el Proeycto:** `https://github.com/ccggwp/CocheBomba.git`