



# UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

---

***FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y  
SERVICIOS***

---

**Proyecto:**  
**Videojuego Educativo de Carreras en Unity**

SEMANA 1: Planificación Técnica y Arquitectura Base

---

**DOCENTE:**  
Ing. Richart Smith Escobedo Quispe

**CURSO:**  
TECNOLOGÍA DE OBJETOS

**CARRERA:**  
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

**REPOSITORIO:**  
<https://github.com/rescobedoq/ryc>

- INTEGRANTES:**
- Choquehuanca Zapana, Hernan Andy
  - Larico Rodriguez, Bryan Fernando
  - Mamani Anahua Victor Narciso
  - Portugal Portugal, Eduardo Sebastián

**FECHA**  
10 - 11 - 2025

## **Revisión de la idea principal y objetivos**

El proyecto consiste en el desarrollo de un videojuego educativo de carreras en Unity, donde el jugador conduce un vehículo a través de un circuito que combina competencia, aprendizaje y toma de decisiones en tiempo real.

Durante la carrera, el jugador se enfrenta a preguntas educativas que aparecen en puntos estratégicos del circuito. En cada punto, se muestran tres portales translúcidos, cada uno representando una posible respuesta.

- Si el jugador atraviesa el portal correcto, el vehículo recibe un impulso de velocidad temporal.
- Si elige una respuesta incorrecta, la velocidad se reduce momentáneamente, obligando al jugador a recuperar el ritmo con habilidad.

Esta dinámica integra educación y entretenimiento, fomentando la atención, rapidez mental y aprendizaje activo, al mismo tiempo que ofrece una experiencia divertida e interactiva.

### **1. Objetivos específicos:**

- 1.1. Diseñar un entorno de carrera funcional y atractivo.
- 1.2. Integrar un sistema de preguntas y respuestas educativas.
- 1.3. Implementar mecánicas de retroalimentación inmediata (aumento o disminución de velocidad).
- 1.4. Aplicar patrones de diseño orientados a objetos para mantener un código modular y escalable.
- 1.5. Promover el aprendizaje lúdico mediante una experiencia inmersiva.

## Selección de patrones de diseño

Patrón	Uso dentro del proyecto
<b>Singleton</b>	Gestiona componentes globales como GameManager y AudioManager, asegurando que exista una sola instancia accesible desde cualquier parte del juego y evitando duplicación.
<b>Observer</b>	Permite la comunicación entre QuestionManager y otros módulos (por ejemplo UIManager o CarController) cuando se responde una pregunta, notificando eventos de manera eficiente y desacoplada.
<b>Factory</b>	Se encarga de la creación dinámica de objetos del entorno (portales, efectos visuales, partículas) a través de ObjectFactory, facilitando la expansión y reutilización de recursos.
<b>State</b>	Controla los distintos estados del vehículo (NormalState, BoostedState, SlowState) según la interacción del jugador con los portales y eventos del circuito.
<b>Strategy</b>	Ajusta dinámicamente la dificultad del juego mediante estrategias (EasyMode, MediumMode, HardMode), modificando penalizaciones, impulsos y tiempos de respuesta según el progreso del jugador.

## Definición de clases principales y roles

Cada clase cumple un rol específico dentro del sistema, contribuyendo al principio de modularidad, reutilización y bajo acoplamiento.

### 1. Gestores (Managers)

Ubicación: Assets/Scripts/Managers/

Clase	Descripción / Rol principal
<b>GameManager</b>	Coordina el flujo general del juego (inicio, pausa, finalización, reinicio). Mantiene el control global de estados y datos. Implementa el patrón Singleton para asegurar una única instancia accesible desde cualquier parte del código.
<b>QuestionManager</b>	Administra el sistema de preguntas y respuestas. Envía notificaciones a otros componentes (como UIManager o CarController) mediante el patrón Observer, y gestiona los efectos derivados de respuestas correctas o incorrectas.
<b>AudioManager</b>	Controla los efectos de sonido y música, integrándose con eventos del juego. También puede usar Singleton para acceso global.
<b>UIManager</b>	Gestiona la visualización de preguntas, resultados y retroalimentación visual en pantalla.
<b>DifficultyManager</b>	Ajusta dinámicamente la dificultad del juego según el rendimiento del jugador, implementando el patrón Strategy.

### 2. Controladores (Controllers)

Ubicación: Assets/Scripts/Controllers/

Clase	Descripción / Rol principal
<b>CarController</b>	Controla el movimiento, física y comportamiento del vehículo (aceleración, frenado, colisiones). Implementa el patrón State para representar los estados del vehículo: Normal, Impulsado (por respuesta correcta) y Penalizado (por respuesta incorrecta).
<b>CameraController</b>	Administra la cámara principal, siguiendo el vehículo y adaptando su posición o ángulo según la velocidad o el entorno.
<b>PortalController</b>	Detecta las colisiones del vehículo con los portales de respuesta y comunica la opción seleccionada al QuestionManager.
<b>CircuitController</b>	Gestiona los elementos del circuito (inicio, checkpoints, meta) y coordina eventos de progreso.

### 3. Patrones y comportamiento (Patterns)

Ubicación: Assets/Scripts/Patterns/

Clase	Descripción / Rol principal
<b>IVehicleState (interfaz)</b>	Define las operaciones base para los distintos estados del vehículo dentro del patrón State.
<b>NormalState/ BoostedState/SlowState</b>	Implementaciones concretas del patrón State, controlando la respuesta del vehículo según las condiciones del juego.
<b>IDifficultyStrategy (interfaz)</b>	Define el contrato para las distintas estrategias de dificultad.
<b>EasyMode/MediumMode /HardMode</b>	Clases que implementan el patrón Strategy, ajustando penalizaciones, velocidad y tiempo de respuesta.
<b>ObjectFactory</b>	Implementa el patrón Factory para la creación dinámica de objetos (portales, efectos visuales, partículas). Facilita la expansión del juego con nuevos elementos.

### 4. Utilidades y soporte (Utilities)

Ubicación: Assets/Scripts/Utilities/

Clase	Descripción / Rol principal
<b>DataLoader</b>	Encargada de cargar los datos de preguntas y configuraciones desde archivos JSON o recursos locales.
<b>TimerUtility</b>	Controla temporizadores o cuenta regresiva para mostrar preguntas o efectos de impulso.
<b>EventBus</b>	Implementa un sistema de comunicación entre objetos (Observer), simplificando el envío de eventos entre distintos módulos del juego.

### 5. Interfaz de Usuario y Recursos (UI y Prefabs)

Ubicación: Assets/UI/ y Assets/Prefabs/

Clase	Descripción / Rol principal
<b>CanvasPreguntas</b>	Muestra las preguntas y opciones de respuesta durante la carrera. Se comunica con QuestionManager.
<b>PortalPrefab</b>	Representa los portales interactivos con respuestas, creados dinámicamente por ObjectFactory.
<b>CarPrefab</b>	Contiene el modelo del vehículo y los scripts de control asociados (CarController).

## Estructura de carpetas propuesta

Para mantener un proyecto claro, modular y fácilmente mantenible, se propone la siguiente estructura de carpetas basada en buenas prácticas de desarrollo en Unity:

None

RyC/

- |— Assets/
  - | |— Scenes/ → Escenas del juego (niveles, menús, interfaz).
  - | |— Scripts/ → Código fuente del proyecto (organizado por módulos).
    - | | |— Managers/ → Clases principales (GameManager, AudioManager, QuestionManager).
    - | | |— Controllers/ → Controladores del vehículo, UI y cámara.
    - | | |— Patterns/ → Implementaciones de patrones (State, Strategy, Factory, Observer).
    - | | |— Utilities/ → Scripts de apoyo o funciones comunes.
  - | |— Prefabs/ → Objetos preconfigurados (vehículo, portales, obstáculos).
  - | |— Models/ → Modelos 3D (.fbx, .obj).
  - | |— Materials/ → Materiales usados por los modelos.
  - | |— Textures/ → Texturas aplicadas a materiales.
  - | |— Audio/ → Efectos de sonido y música.
  - | |— UI/ → Elementos de interfaz (preguntas, botones, paneles).
  - | |— Resources/ → Archivos cargados dinámicamente en tiempo de ejecución.
  - | |— Documentation/ → Documentos técnicos, cronogramas y reportes.
  - | |— Tests/ → Scripts de prueba o escenas para testeo de mecánicas.
- |— Packages/ → Paquetes de dependencias Unity.
- |— ProjectSettings/ → Configuraciones del proyecto (input, gráficos, física, calidad, etc.).