****

Módulo Profesional 06: Acceso a datos **UF3 – ACT1 - Base de datos documental**

**Actualizado: 3 / 4 / 2025**

**CICLO FORMATIVO DE GRADO SUPERIOR EN**

**Videojuegos y ocio digital**

**MODALIDAD PRESENCIAL**

**Nombre y apellidos del alumno**



**Introducción**

****

En esta actividad crearás un sistema de registro y visualización de estadísticas usando una base de datos documental

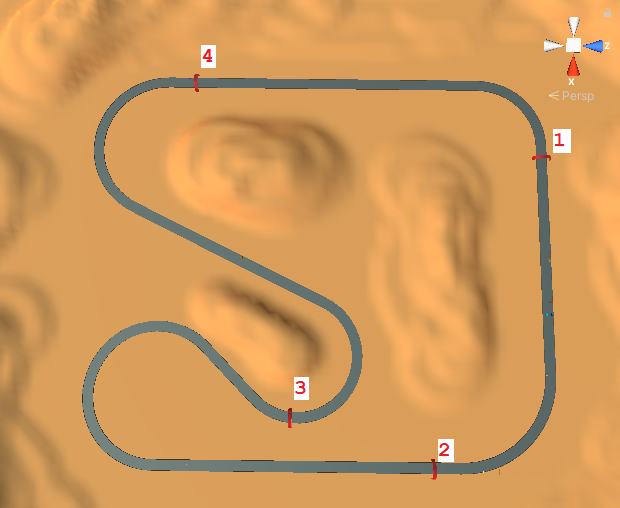
|  |
| --- |
| **Objetivos** |
| Aprender a guardar datos en una base de datos documental  Extraer información útil de los datos guardados |
| **Metodología de evaluación** |
| 50% Registro de datos  50% Visualizador |
| **Entrega** |
| A través del campus con nombre UF3-ACT1- NombreApellidos.zip incluyendo enunciado, proyecto unity, build del juego y vídeo demostrativo de uso si no lo alojas en un servicio externo. |
| **Documentos de referencia** |
| Manual y referencia de Unity3D  Documentación del driver para C# de MongoDB |

**Situación (Versión 1)**

Queremos añadir un sistema de rastreo de estadísticas a un juego de carreras usando una base documental como soporte.

**Diseño de juego (Versión 1)**

Repartidas por el nivel tenemos una serie de puertas que registran el tiempo que ha tardado el coche del jugador en alcanzarlas y la velocidad que tenía al hacerlo.



Estas puertas sólo registran estas estadísticas una vez y para que lo hagan deben visitarse en orden.

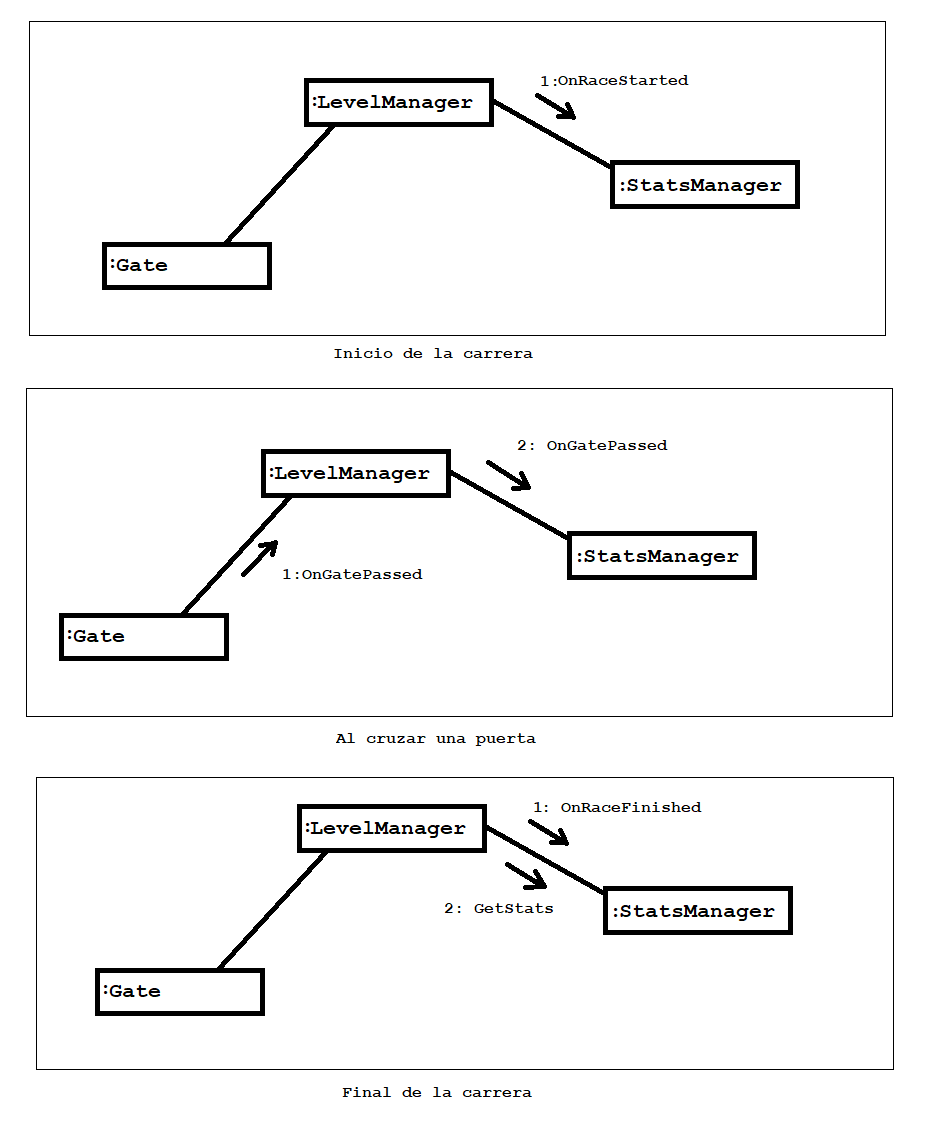
La carrera finaliza cuando el jugador visita todas las puertas en orden o bien cuando se agota el límite de tiempo de que dispone.

Al finalizar la carrera se muestran las estadísticas recogidas en cada puerta visitada en orden.



**Diseño de programa (Versión 1)**

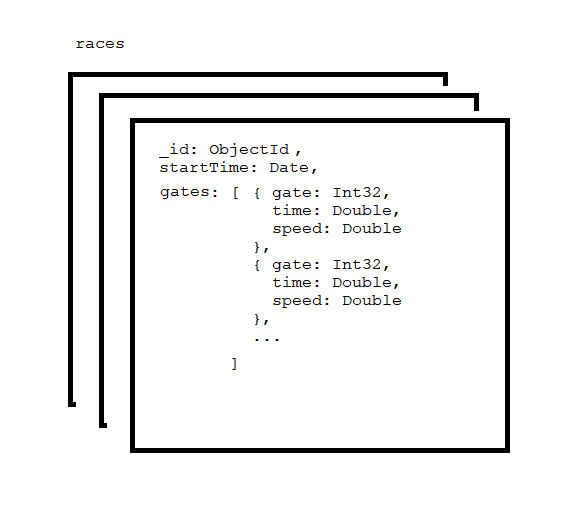
**Secuencias de llamadas**



**Clase StatsManager**

|  |  |  |
| --- | --- | --- |
| **StatsManager** | | |
| **Propiedad** | **Tipo** | **Descripción** |
| serverAddress | string | Dirección del servidor |
| database | string | Nombre de la base de datos |
| collection | string | Nombre de la colección |
| **Método** | **Descripción** | |
| OnRaceStarted() | Inicia un nuevo registro en memoria para la carrera | |
| OnGatePassed(int index, float time, float speed) | Registra en memoria el tiempo de paso de la puerta | |
| OnRaceFinished() | Inserta un nuevo documento en la base de datos | |
| int GetStats(Stat[] stats) | Recupera de la base de datos las estadísticas de la carrera | |

**Estructura de la base de datos**



**Ejercicio1:** Añade el código necesario a la clase StatsManager para que realice el guardado y la carga de las estadísticas en/desde la base de datos.

**Situación (Versión 2)**

En esta versión añadiremos un sistema de análisis de datos que registrará las posiciones y orientaciones del coche durante la carrera (telemetría) con una cierta frecuencia, por ejemplo 5 veces por segundo.

También situaremos barreras en la pista.

**Diseño de juego (Versión 2)**

Repartidas por el nivel tenemos una serie de puertas que registran el tiempo que ha tardado el coche del jugador en alcanzarlas y la velocidad que tenía al hacerlo.

A race track with a road and a plane

AI-generated content may be incorrect.

Estas puertas sólo registran estas estadísticas una vez y para que lo hagan deben visitarse en orden.

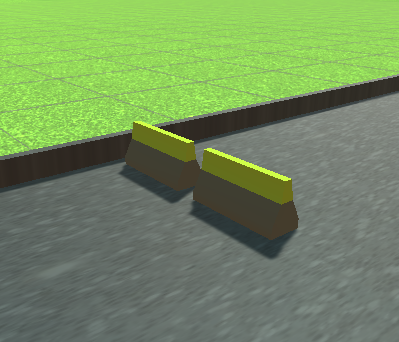
La carrera finaliza cuando el jugador visita todas las puertas en orden o bien cuando se agota el límite de tiempo de que dispone.

Al finalizar la carrera se muestran las estadísticas recogidas en cada puerta visitada en orden.

A screenshot of a video game

AI-generated content may be incorrect.

Repartidas por el nivel podemos encontrar una cierta cantidad de barreras con las que el coche puede chocar.

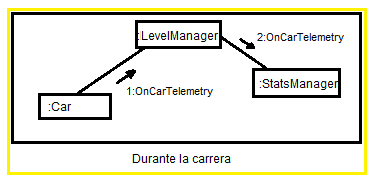
****

**Diseño de programa (Versión 2)**

**Secuencias de llamadas**

A diagram of a project

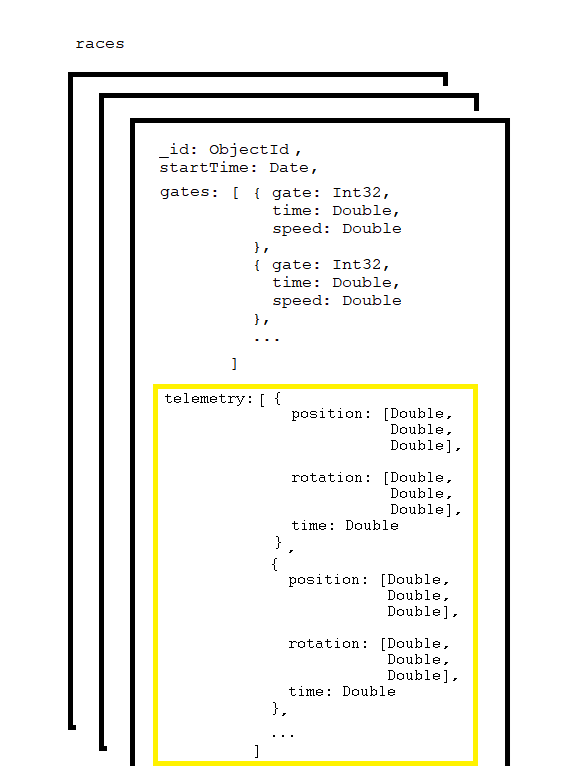
AI-generated content may be incorrect.



**Clase StatsManager**

|  |  |  |
| --- | --- | --- |
| **StatsManager** | | |
| **Propiedad** | **Tipo** | **Descripción** |
| serverAddress | string | Dirección del servidor |
| database | string | Nombre de la base de datos |
| collection | string | Nombre de la colección |
| **Método** | **Descripción** | |
| OnRaceStarted() | Inicia un nuevo registro en memoria para la carrera | |
| OnGatePassed(int index, float time, float speed) | Registra en memoria el tiempo de paso de la puerta | |
| OnRaceFinished() | Inserta un nuevo documento en la base de datos | |
| int GetStats(Stat[] stats) | Recupera de la base de datos las estadísticas de la carrera | |
| OnCarTelemetry(Vector3 pos, Vector3 rot, float time) | Registra la telemetría del coche | |

**Estructura de la base de datos**



**Ejercicio2:** Añade los obstáculos al circuito y el código necesario a la clases Car, LevelManager y StatsManager para que realice registro de la telemetría.

**Situación (Versión 3)**

En esta versión realizaremos una explotación de los datos de telemetría para nuestro juego que consistirá en una guía muostrará una de las mejores carreras.

**Diseño de juego (Versión 3)**

Repartidas por el nivel tenemos una serie de puertas que registran el tiempo que ha tardado el coche del jugador en alcanzarlas y la velocidad que tenía al hacerlo.

A race track with a road and a plane

AI-generated content may be incorrect.

Estas puertas sólo registran estas estadísticas una vez y para que lo hagan deben visitarse en orden.

La carrera finaliza cuando el jugador visita todas las puertas en orden o bien cuando se agota el límite de tiempo de que dispone.

Al finalizar la carrera se muestran las estadísticas recogidas en cada puerta visitada en orden.

A screenshot of a video game

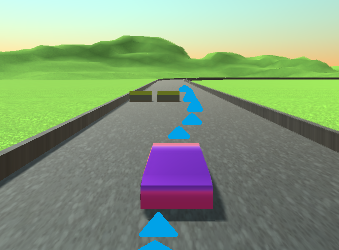
AI-generated content may be incorrect.

Repartidas por el nivel podemos encontrar una cierta cantidad de barreras con las que el coche puede chocar.

**A yellow and brown road blocks

AI-generated content may be incorrect.**

Durante la carrera se mostrará a modo de guía la trayectoria del jugador que haya completado la carrera en menos tiempo de un conjunto de al menos 5 carreras seleccionadas al azar.

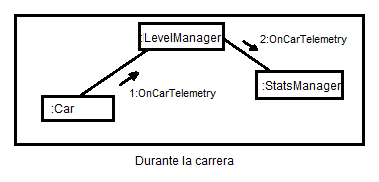
****

**Diseño de programa (Versión 3)**

**Secuencias de llamadas**

A diagram of a project

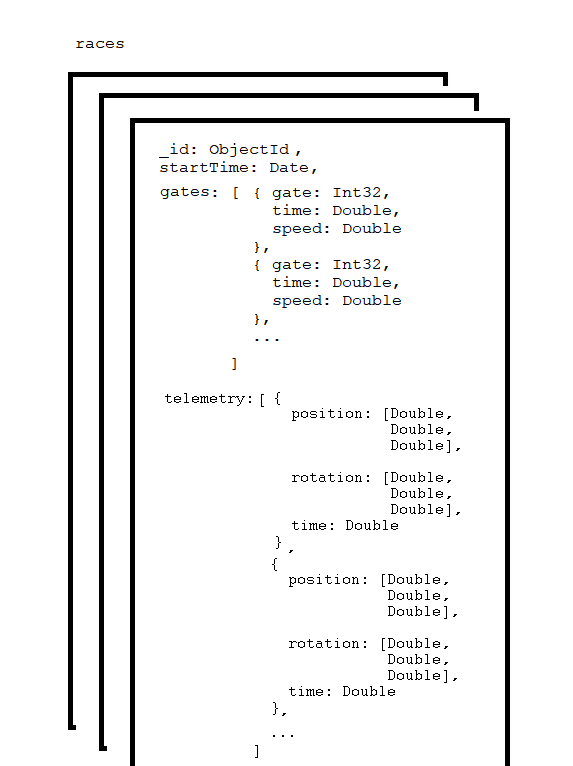
AI-generated content may be incorrect.



**Clase StatsManager**

|  |  |  |
| --- | --- | --- |
| **StatsManager** | | |
| **Propiedad** | **Tipo** | **Descripción** |
| serverAddress | string | Dirección del servidor |
| database | string | Nombre de la base de datos |
| collection | string | Nombre de la colección |
| **Método** | **Descripción** | |
| OnRaceStarted() | Inicia un nuevo registro en memoria para la carrera | |
| OnGatePassed(int index, float time, float speed) | Registra en memoria el tiempo de paso de la puerta | |
| OnRaceFinished() | Inserta un nuevo documento en la base de datos | |
| int GetStats(Stat[] stats) | Recupera de la base de datos las estadísticas de la carrera | |
| OnCarTelemetry(Vector3 pos, Vector3 rot, float time) | Registra la telemetría del coche | |
| Start() | Busca la mejor trayectoria de entre un conjunto de carreras seleccionado al azar y almacena su telemetría. | |
| Update() | Dibuja la telemetría de la mejor carrera. | |

**Estructura de la base de datos**



**Ejercicio3:** Añade el código necesario a la clase StatsManager para que calcule el mejor camino y lo dibuje durante la carrera. Una vez lo consigas, graba un vídeo explicativo (3 – 5 minutos) en el que muestres el juego, el código relevante y los cambios que se producen en la base de datos según se desarrolla una partida. Si lo alojas en un servicio externo, incluye un enlace a continuación.

|  |
| --- |
|  |