



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



TFG del Grado en Ingeniería Informática

Fastastic Roads

Videojuego de carreras con modo a
contrarreloj



Presentado por Alejandro Goicoechea Román
en Universidad de Burgos — 7 de julio de 2021
Tutores: Jesús María Alonso Abad y
Mario Alaguero Rodríguez



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Jesús María Alonso Abad, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos, y D. Mario Alaguero Rodríguez, profesor del Grado en Comunicación Audiovisual.

Exponen:

Que el alumno D. Alejandro Goicoechea Román, con DNI 71313897M, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Fastastic Roads.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 7 de julio de 2021

Vº. Bº. del Tutor:

D. Jesús María Alonso Abad

Vº. Bº. del co-tutor:

D. Mario Alaguero Rodríguez

Resumen

Los videojuegos son una importante forma de entretenimiento. La evolución de los motores de videojuegos y su libre disponibilidad al público ha permitido que cualquier usuario que quiera aprender pueda desarrollar sus propios videojuegos. Gracias a ello, se ha podido hacer uso de Unity para este proyecto.

“Fastastic Roads” es un videojuego de carreras de karts. Pretende convertirse en un videojuego didáctico mediante el cual se pueda aprender la historia de cuatro importantes pioneras involucradas en el desarrollo y evolución de los vehículos que manejan en el juego.

En el presente trabajo se ha creado un modo a contrarreloj, con todos los componentes necesarios para que se comporte como tal, así como la implementación de modelos, menús y otros elementos. Además, se aborda la experiencia de un desarrollo coordinado en un equipo multidisciplinar.

Descriptores

Unity, C#, videojuego de carreras, karts, aplicación Windows, aplicación Linux.

Abstract

Video games are an important form of entertainment. The evolution of video game engines and their free availability to the public has allowed any user who wants to learn to develop their own video games. Thanks to this, it has been possible to use Unity for this project.

“Fastastic Roads” is a kart racing video game. It aims to become a didactic video game through which the history of four important female pioneers involved in the development and evolution of the vehicles they drive in the game can be learnt about.

In the present work, a time trial mode has been created, with all the necessary components for it to behave as such, as well as the implementation of models, menus and other elements. In addition, the experience of a coordinated development in a multidisciplinary team is addressed.

Keywords

Unity, C#, racing game, karts, Windows application, Linux application.

Índice general

Índice general	iii
Índice de figuras	v
Índice de tablas	vi
Introducción	1
1.1. Materiales adjuntos	2
Objetivos del proyecto	3
2.1. Objetivos generales	3
2.2. Objetivos técnicos	3
Conceptos teóricos	5
3.1. Motor de videojuegos	5
3.2. Videojuegos de carreras	15
Técnicas y herramientas	17
4.1. Unity	17
4.2. Lenguaje de programación C#	17
4.3. Visual Studio Community 2019	18
4.4. LaTeX	18
4.5. Texmaker	19
4.6. Blender	19
4.7. Kanban	19
4.8. GitHub	20
Aspectos relevantes del desarrollo del proyecto	21

5.1. Inicio del proyecto	21
5.2. Metodología de desarrollo Kanban	22
5.3. Formación	23
5.4. Desarrollo del proyecto	23
5.5. Problemas y resolución	27
5.6. Documentación	28
5.7. Exposición social	28
Trabajos relacionados	31
6.1. Mario Kart	31
6.2. Crash Team Racing	32
6.3. UBURACE	32
Conclusiones y Líneas de trabajo futuras	33
7.1. Conclusiones	33
7.2. Líneas de trabajo futuras	34
Bibliografía	37

Índice de figuras

3.1.	Escena vacía en el motor de videojuegos Unity.	6
3.2.	<i>GameObject</i> básico ubicado en la jerarquía de objetos.	7
3.3.	Se pueden añadir tanto componentes provistos por Unity como propios.	8
3.4.	Los <i>scripts</i> se programan en C#.	9
3.5.	Componente de cámara junto a su representación en pantalla. .	10
3.6.	Coordenadas en el <i>Transform</i> de un objeto.	11
3.7.	Se pueden incluir todo tipo de <i>assets</i> compatibles con Unity. .	13
3.8.	<i>Prefab</i> del personaje de Pilar Careaga.	14
5.9.	Fotograma de la <i>Release Candidate 1</i> de “Fastastic Roads”. .	26
5.10.	Modelos de personajes creados en <i>Blender</i>	27
5.11.	El progreso del proyecto ha podido ser visualizado en Internet. .	29

Índice de tablas

Introducción

Los videojuegos son una forma de entretenimiento extendida a nivel global. La riqueza en variedad de géneros, estilos a nivel gráfico y narrativo, bandas sonoras, animaciones y demás aspectos artísticos es muy amplia, por lo que nos permite experimentar creando juegos casuales, serios, simuladores y otros más experimentales.

Según la Asociación Española de Videojuegos (AEVI), el mercado internacional del videojuego creció un 9,6 % en el año 2019 con respecto al año anterior, alcanzando una facturación total de 133.670 millones de euros [1]. A nivel estatal, según el primer informe económico en el que se analizó el impacto de la industria de los videojuegos sobre la contabilidad nacional realizado en enero de 2018, la industria de los videojuegos equivale al 0,11 % del PIB español, donde por cada euro invertido en este sector se tiene un retorno de 3 euros, dando un total de 3577 millones de euros y aproximadamente 22.800 empleos a fecha 2016 [2].

Teniendo en cuenta estos datos, se es consciente del alcance que tienen los videojuegos y de la oportunidad que puede ser aprovechar los recursos disponibles para crear uno que pueda atraer a las personas y tenga un sentido didáctico. Esta idea hace unos años era concebible, pero muy difficilmente realizable, no solo por los escasos recursos que podía haber en Internet o en los libros, sino porque las herramientas no estaban disponibles para un usuario medio que no pudiese invertir en programas con licencias de pago, o en su defecto no estaban lo suficientemente desarrolladas.

Por suerte, gracias sobre todo al avance tecnológico, hoy en día se dispone de herramientas muy potentes que nos ofrecen un ecosistema preparado para poder crear y desarrollar videojuegos al alcance de cualquier persona, tenga o no conocimientos previos, y que tenga un ordenador que cumpla

los requisitos mínimos para un correcto funcionamiento, además de una conexión a Internet para poder descargarlos.

En este proyecto se ha partido desde un motor escogido en base al análisis general de los disponibles, sin conocimiento sobre el funcionamiento del mismo, hasta tratar de conseguir un videojuego jugable, disfrutable y accesible al usuario, así como trabajado en un entorno multidisciplinar relacionado con los videojuegos. Con todo ello, se pretende principalmente comprender la complejidad de un proyecto de esta magnitud, cómo funcionan los motores de videojuegos, cómo se pueden solventar los problemas al respecto y, sobre todo, disfrutar del trayecto y del resultado obtenido.

1.1. Materiales adjuntos

Los materiales del proyecto adjuntados son los siguientes:

- Aplicación para Windows y Linux “Fastastic Roads”.
- Códigos fuente y proyecto para su edición en el motor Unity.
- Vídeo explicativo del videojuego en funcionamiento.
- Memoria del Trabajo de Fin de Grado y la documentación técnica de programación, ubicada en el Anexo D del documento de anexos.

Objetivos del proyecto

En este apartado se detallan los objetivos que se pretenden lograr con este proyecto, dividiéndose en generales y técnicos.

2.1. Objetivos generales

Los objetivos generales que se pretenden alcanzar en este proyecto son:

- Desarrollar un videojuego de carreras para PC, compatible con Windows y Linux.
- Ofrecer una buena jugabilidad y accesibilidad al usuario, con una curva de aprendizaje sencilla.
- Aprender a usar Unity, sus funcionalidades y la programación en el lenguaje que utiliza.
- Asentar unas bases para proyectos similares.

2.2. Objetivos técnicos

Así mismo, los objetivos técnicos que se pretenden alcanzar son:

- Crear un videojuego óptimo y fácilmente extensible, para poder introducir mejoras posteriores.
- Desarrollar los distintos sistemas que componen a un videojuego de carreras contrarreloj.
- Implementar todos los componentes necesarios.
- Utilizar Git y GitHub como sistemas de control de versiones.
- Aplicar la metodología Kanban en el desarrollo de *software*.

Conceptos teóricos

Para comprender todo el ecosistema que envuelve a un videojuego, han de ser explicados los distintos términos con los que el proyecto ha tomado contacto y con aquellos en los que se profundizará más adelante.

En este apartado se detallan los conceptos principales y cómo influyen en el desarrollo del trabajo.

3.1. Motor de videojuegos

El corazón de un videojuego es el motor. Como definición, se trata de una serie de rutinas de programación que forman el núcleo de un juego, permitiendo el diseño, la creación y el funcionamiento de éste. Así mismo, se llama motor de videojuegos al software que permite crearlos [3].

Los videojuegos, en su desarrollo, requieren de un entorno que permita englobar todas las herramientas necesarias para poder crearlos. Los motores son el ecosistema perfecto, pues proporcionan herramientas de desarrollo visual y componentes *software* que se pueden reutilizar. Esto es, no es necesario crear desde cero un sistema de físicas, sonido, interacción entre objetos, efectos y otros, sino que todos esos elementos ya los contiene el propio motor, pudiendo hacer uso de ellos a necesidad del usuario.

Todo ello conforma el esqueleto del juego, ayudando a los diseñadores, programadores y demás participantes en el proyecto a centrarse en las tareas que han de realizar.

Hay numerosos motores de videojuegos disponibles al público, disponibles tanto de pago como gratuitos [4]. Actualmente, los tres más potentes en el mercado son gratuitos, donde puede hacerse un uso académico, personal o

comercial de ellos (en este último caso hay un límite de ingresos, donde a partir de determinada cifra se empieza a pagar un porcentaje a la compañía para rentabilizar el producto).

En este proyecto se hace uso del motor gráfico Unity.

Escena

Las escenas son el principal elemento fundamental de Unity, motor del que se hablará en el capítulo 4, “[Técnicas y Herramientas](#)”. Contienen todos los objetos que componen un nivel, como puede ser todo lo que conforma a los menús, a niveles individuales y cualquier otra cosa. Cada fichero de escena se considera como un marco único [5].

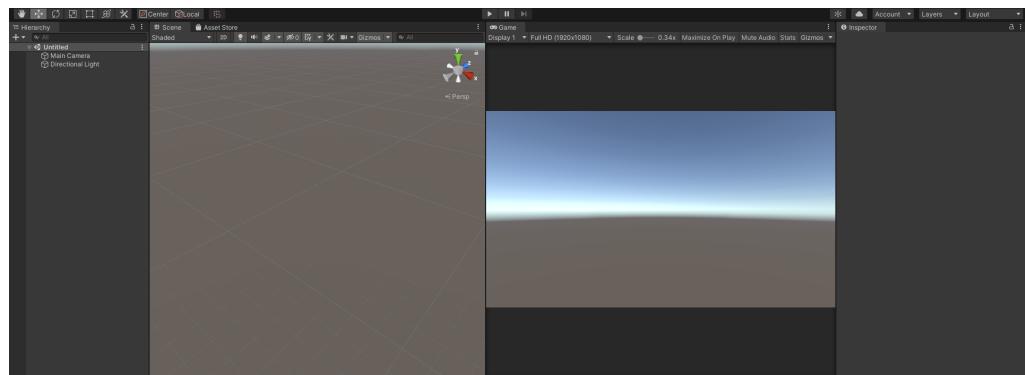


Figura 3.1: Escena vacía en el motor de videojuegos Unity.

GameObject

Se trata de la unidad más básica dentro de una escena. Representa personajes, *props* o elementos de atrezo, y el escenario. Son objetos inertes, sin ninguna funcionalidad de base, por lo que no tienen un comportamiento por sí mismos, pero sí funcionan como contenedores para componentes, los cuales son los que proporcionan las funcionalidades a los mismos, como puede ser el movimiento, conteo de vueltas o el sistema de puntos de control entre otros dentro de este proyecto [6].

Todos los *GameObject* tienen un componente *Transform* o transformada del objeto, que sirve para representar la posición, orientación y escala del objeto respecto al objeto padre del mismo, y en caso de no tener objeto padre será respecto a la escena [7]. Es uno de los elementos necesarios para poder mover libremente el objeto por el escenario, así como cambiar su

tamaño y orientarlo en la posición en la que desee el usuario. La posición del *GameObject* es de tipo *Vector3* y la orientación es de tipo *Quaternion* (en este caso se hace uso de una función llamada “*Euler*”, siendo los ángulos de *Euler* una representación de una rotación alternativa a los *quaternions* expresada como una secuencia de rotaciones entre los ejes cartesianos, primero rotando en Z, luego en X y luego en Y, para devolverla en un vector de tres ángulos).



Figura 3.2: *GameObject* básico ubicado en la jerarquía de objetos.

Componentes

Los componentes son las distintas piezas requeridas para dotar a los objetos de la capacidad de ejercer determinadas acciones o características, como pueden ser las cámaras, el movimiento, el sonido, la luz, las colisiones, los sistemas de conteo y todo lo que aporte Unity por defecto o quiera crear el usuario [8].

Se crean como *scripts* programados en C#. Estos *scripts* se adjuntan a los *GameObjects*, dando la funcionalidad correspondiente a los objetos en los que se hallen. Gracias a *UnityEngine*, a través de una clase base “*MonoBehaviour*” se tienen todas las herramientas para poder programar *scripts* interactuando con los distintos *GameObjects* y el entorno de Unity. No obstante, algunas veces puede ser limitante a la hora de querer modificar alguna característica a bajo nivel, ya que Unity tiene sus propias restricciones a las que el usuario tiene que adaptarse [9].

Los componentes les otorgarán comportamientos específicos, donde los más destacables para este proyecto suministrados por defecto por Unity han sido:

- ***Mesh Renderers* o renderizadores de mallas:** definen la geometría de los objetos obtenida del *Mesh Filter* (el cual es el encargado de

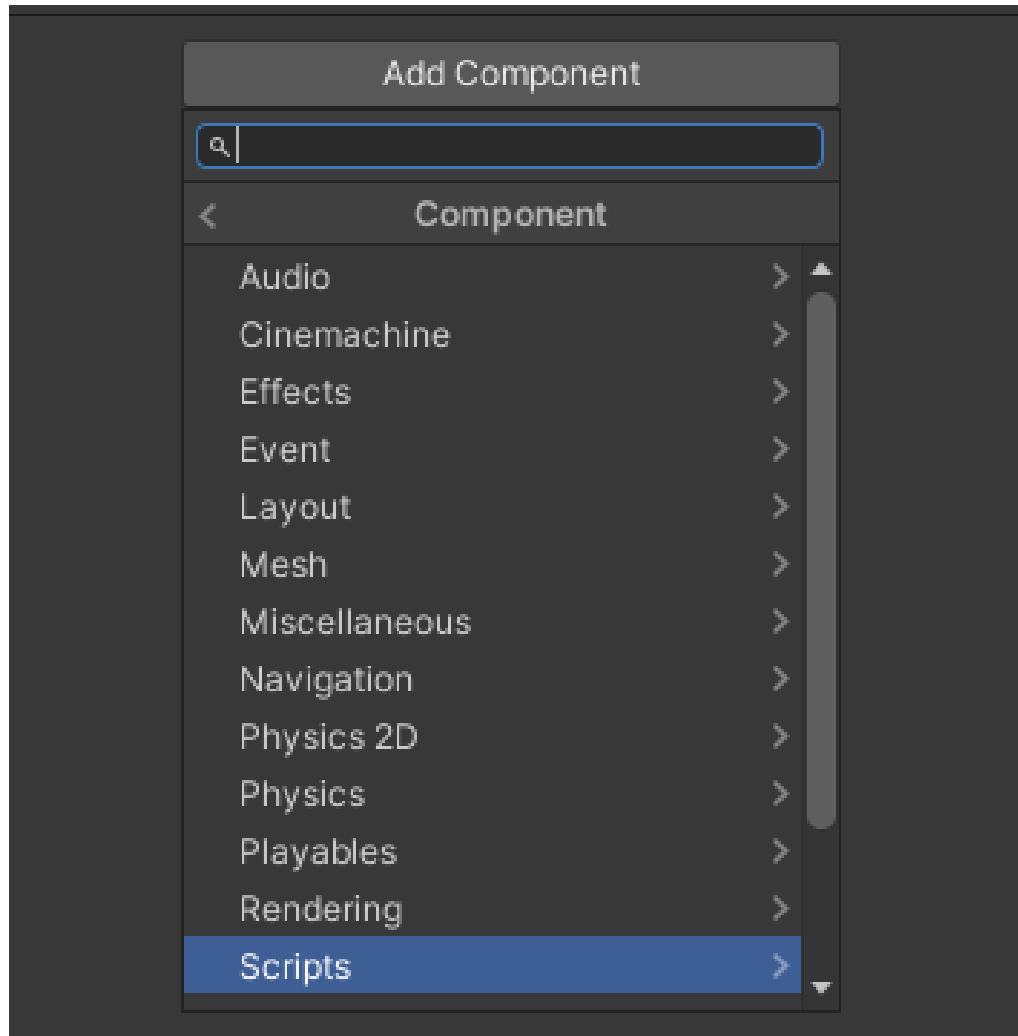


Figura 3.3: Se pueden añadir tanto componentes provistos por Unity como propios.

cargar una malla de los *assets*) y la renderiza en la posición definida por el componente de transformada del objeto para mostrarla en pantalla [10].

- **Colliders o colisionadores:** definen la forma de un objeto con el propósito de ejercer colisiones físicas. Este componente es invisible, y no necesariamente tiene la forma exacta de la malla del objeto en cuestión (de hecho, por lo general, una aproximación a menudo es más eficiente para el rendimiento y prácticamente indistinguible en el juego) [11].

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ScriptPorDesarrollar : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
    }

    // Update is called once per frame
    void Update()
    {
    }
}

```

Figura 3.4: Los *scripts* se programan en C#.

- ***Mesh Colliders* o colisionadores de malla:** toma la forma de una malla y construye el colisionador basado en ella. Pueden usarse como disparadores o triggers si son convexos y se indican como tal [12].
- ***Wheel Colliders* o colisionadores de ruedas:** son colisionadores especiales para vehículos. Tiene detección de colisiones integrada, física de ruedas y un modelo de deslizamiento basado en la fricción de la malla. Puede utilizarse para objetos que no sean ruedas pero está específicamente diseñado para ello [13].
- ***Camera* o cámara:** dispositivos que capturan y muestran el mundo al jugador. Se explica más detalladamente en el apartado de “Cámaras”.
- ***Lights* o luces:** partes esenciales de cada escena, definen el esquema de iluminación de una escena.
- ***Rigidbody*:** permite que un objeto tenga un comportamiento físicamente correcto. Se explica más detalladamente en el apartado de “Físicas” [14].

En el caso del vehículo a manejar en el juego, consta de un conjunto de renderizadores de malla que conforman la apariencia del vehículo, y un conjunto de colisionadores, dividido en un colisionador de malla, cuatro específicos de ruedas y cuatro de esferas.

Para el presente trabajo se han desarrollado, además, nuevos componentes que se describirán en detalle en el anexo C.

Cámaras

Las cámaras en Unity son objetos que definen una vista en una escena. La posición y orientación del objeto definen el campo visual [15].

Las escenas en Unity se crean mediante el posicionamiento y movimiento de objetos en un espacio tridimensional. Puesto que la pantalla del espectador es bidimensional, las cámaras permiten capturar estas vistas, donde se aplica una proyección para convertir el espacio en tres dimensiones a un espacio bidimensional (aunque no se pierde la información de la profundidad).

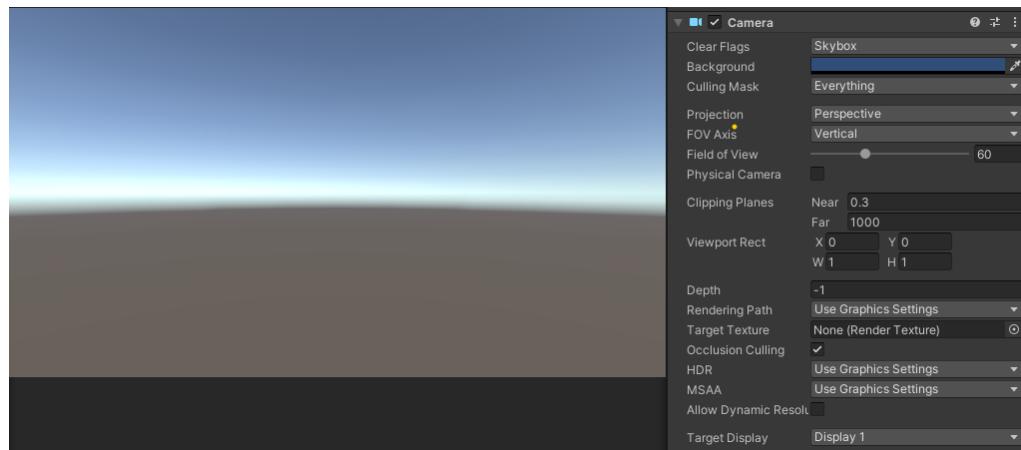


Figura 3.5: Componente de cámara junto a su representación en pantalla.

Una de las proyecciones que permite visualizar la escena lo más fiel a la realidad es la proyección en perspectiva cónica. Esta perspectiva es un sistema de representación gráfico basado en la proyección de cuerpos tridimensionales sobre un plano, auxiliándose en rectas proyectantes que pasan por un punto. Este sistema es el que más se asemeja a la visión humana, ya que se logra una aparente profundidad [16], y es el empleado en Unity para la representación de objetos en la escena.

La forma en la que se aprovecha de cara a un videojuego de carreras es empleando la cámara en tercera persona, donde la cámara sigue al vehículo ligeramente alejada. Para un mayor realismo, se ha hecho uso de un sistema de cámaras dinámico ya programado más fluido llamado “Cinemachine” [17]. Este componente ha sido instalado desde propio Unity.

Sistema de coordenadas

El sistema de coordenadas engloba a toda la escena en la que nos hallamos. Esto provoca que las coordenadas de los *GameObject* que se hallan en la jerarquía de objetos y que no tengan ningún otro objeto en niveles superiores de la jerarquía sean globales, es decir, que sean relativas a la escena en la que se halle el usuario.



Figura 3.6: Coordenadas en el *Transform* de un objeto.

En caso de que un *GameObject* tenga un objeto padre en la jerarquía de objetos, su sistema de coordenadas será local en vez de global, lo que implica que su transformación será relativa a la transformación del objeto padre, esto es, dependerá de la jerarquía de objetos y no de la escena en sí.

Es importante tener en cuenta este concepto y su funcionamiento, ya que, aunque en apariencia es simple, puede provocar confusión a la hora de la colocación de un objeto, debido a que unas mismas coordenadas no generan el mismo posicionamiento en la escena con un objeto que depende de otro objeto (o de una jerarquía de objetos) que con un objeto que depende de la propia escena.

Físicas

Unity integra un sistema de físicas que permite a los *GameObjects* tener un comportamiento físico convincente, donde deben poder tener una aceleración adecuada y verse afectados por las colisiones, la gravedad y otras fuerzas. Este sistema proporciona distintos componentes que manejan la

simulación física, de manera que modificando los distintos ajustes se puede obtener objetos que se comporten pasivamente de una manera realista. Da la posibilidad de controlar estas físicas desde *script*, por lo que se puede dotar a los objetos de un dinamismo muy diverso. En Unity, los conceptos principales de las físicas para los juegos en 2D y en 3D son iguales, pero la implementación es diferente, por lo que hay un motor de físicas específico para 2D y otro para 3D [18].

Los principales componentes en relación a las físicas son los *Rigidbody* y los *colliders* o colisionadores (estos últimos han sido explicados en el apartado de *GameObject*).

Un *Rigidbody* es el componente principal que permite el comportamiento físico de un objeto. Al añadir este componente a un objeto, éste responderá de manera inmediata a la gravedad. Si, además, se añaden uno o más colisionadores al objeto, entonces éste será movido por las colisiones entrantes. La manera en la que ahora responderá el objeto ante el movimiento será mediante fuerzas aplicadas al mismo, y no mediante cambios directos en su transformada [19].

También hay que tener en cuenta a los *triggers*, los cuales no son componentes físicos como tal sino eventos que nos permiten modificar el comportamiento de un objeto y actúan dentro de los colisionadores. Estos eventos se activan indicando si un colisionador es un *trigger* o no. A partir de esa activación, el colisionador deja de tener comportamiento físico y ahora puede ejercer las acciones que indique un componente mediante *script*.

Asset

Los *assets* o activos son representaciones de cualquier ítem que pueda ser utilizado en el juego. Puede partir de un archivo creado de manera externa a Unity, como puede ser un modelo 3D, un archivo de audio, una imagen o cualquiera de los tipos de archivo que Unity soporta, así como aquellos que pueden ser creados dentro del mismo motor, como los controladores de animación o los mezcladores de audio [20].

En “Fastastic Roads” los *assets* principales son los modelos de los vehículos y del escenario, los cuales son objetos 3D creados en Blender, acompañados de sus respectivas texturas.

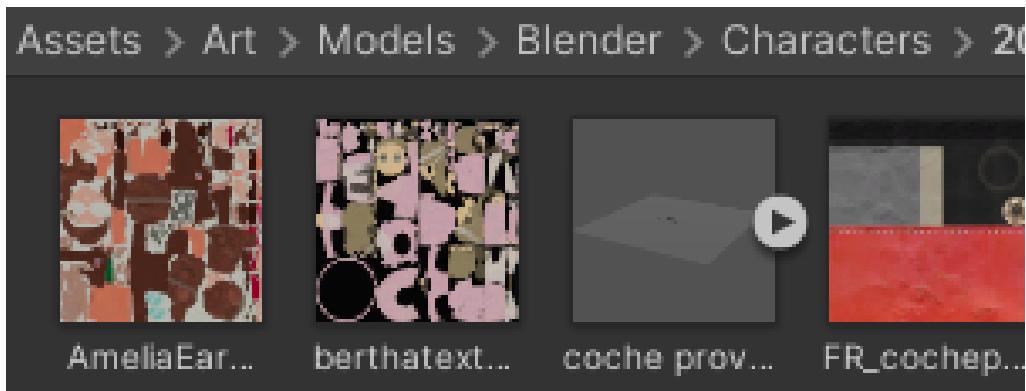


Figura 3.7: Se pueden incluir todo tipo de *assets* compatibles con Unity.

Prefab

Los *prefabs* u objetos prefabricados son *GameObjects* a los cuales se les ha añadido distintos componentes y ajustado sus propiedades al valor adecuado al diseñador, de tal manera que se almacena como un conjunto para poder ser instanciado en adelante sin tener que integrar cada modelo individual con sus mismas propiedades, optimizando el tiempo de creación de objetos y pudiendo ser reutilizados tantas veces como se necesite, esto es, actuando como una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena [21].

En el momento en el que un *prefab* sea editado, se reflejará en todas las instancias producidas a partir de éste. No obstante, se pueden realizar modificaciones individuales para cada instancia.

Renderizado

El renderizado 3D es el proceso de producción de una imagen bidimensional basado en datos tridimensionales. Con el renderizado 3D, se convierten las mallas de modelos 3D (estas mallas se componen de vértices y caras triangulares que conectan estos vértices) en imágenes 2D [22].

Existen dos tipos principales:

- **Renderizado sin conexión (o prerenderizado):** el coste de generar las imágenes es demasiado elevado para realizarlo a una velocidad suficiente y, por tanto, se procesan con lentitud y se graban para reproducirlas posteriormente. Es el sistema utilizado, por ejemplo, en el cine de animación.

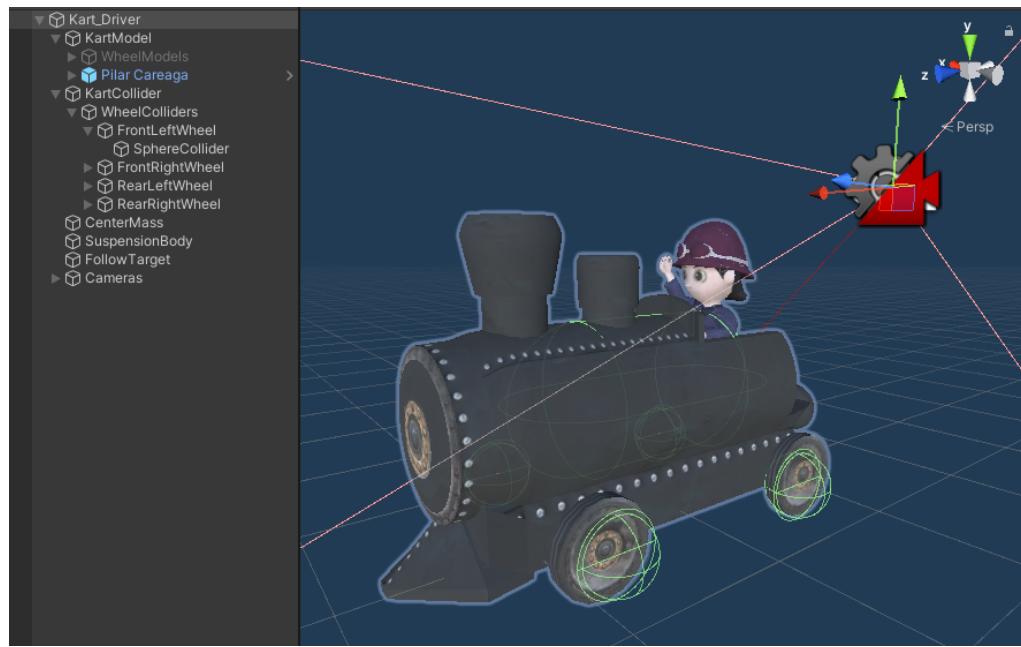


Figura 3.8: *Prefab* del personaje de Pilar Careaga.

- **Renderizado en tiempo real:** es el más común en videojuegos, donde el renderizado de fotogramas es suficientemente rápido como para que al renderizarse secuencialmente los diferentes fotogramas den la sensación de animación y movimiento de manera fluida. El objetivo es tratar de alcanzar una velocidad de renderizado mínima aceptable para el jugador, siendo 24 fotogramas/segundo el mínimo necesario para que el ojo humano pueda crear “ilusión de movimiento”.

Sistemas de entrada

Los sistemas de entrada o *inputs* permiten tener una interacción directa con los objetos del escenario a la hora de jugar. Unity soporta distintos dispositivos de entrada, entre ellos los convencionales como son el teclado, los *joysticks*, *joypads* y otros, así como pantallas táctiles, detección de movimiento de dispositivos móviles, micrófonos y cámaras web [23].

Para poder moverse por el juego se requiere de un ratón para la selección de las opciones entre los menús y de un teclado o mando para manejar el vehículo en una partida.

Sonido

El sonido es un aspecto muy importante en un videojuego, pues es el encargado de transmitir las sensaciones auditivas respecto a las acciones que realiza el jugador o los elementos con los que interactúa, como puede ser accionar un botón, activar algún objeto, correr o el sonido de un motor entre otros, así como el sonido ambiente que engloba a una escena [24].

Unity provee al usuario de un sistema de sonido espacial 3D completo, así como de un sistema de mezcla y masterizado en tiempo real, jerarquías de mezcladores, instantáneas y efectos.

En el caso del juego, carece de sonido debido a las limitaciones temporales del proyecto. No obstante, es un objetivo a cumplir, proveyendo al juego de efectos de sonido y banda sonora.

Multijugador

El modo multijugador, bien sea en red local, red Internet o a pantalla partida (diversos jugadores en una misma pantalla), es un tipo de juego en el cual varios participantes compiten o colaboran en una partida de juego. Este es un modo muy atractivo para el usuario, ya que permite jugar e interactuar con más personas en el mismo juego.

Este modo actualmente no se encuentra disponible en el juego, pero, de nuevo, es una idea planteada con intención de ser integrada más adelante, pues hará más atractivas las partidas al tener más jugadores involucrados en ellas.

3.2. Videojuegos de carreras

Mencionado con anterioridad, hay multitud de géneros en los videojuegos. Entre ellos se encuentran los videojuegos de conducción, donde, como su nombre indica, se tratan de juegos en los cuales se emula la conducción de un vehículo en un entorno virtual.

Esta conducción puede ser muy similar a la real, factor principal de los videojuegos simuladores de conducción, o puede ser “arcade”, cuyo término era originario de las primeras máquinas recreativas (o *máquinas arcade*) y se utiliza también actualmente para designar un estilo de videojuegos que siguen los principios básicos que tenían esas máquinas, entre ellos el tener un diseño sencillo, controles fáciles de asimilar y dominar, niveles no muy

extensos, con dificultad ascendente y con una escasa interrupción de juego entre niveles [25].

El entorno virtual puede representarse de diversas formas, como son los circuitos cerrados, donde se pueden realizar distintas pruebas como carreras competitivas o a contrarreloj y otra serie de pruebas como minijuegos (saltos de altura con el vehículo, superar obstáculos, etc.) u otras, o los escenarios de “mundo abierto”, en los cuales se pueden realizar también estas pruebas pero con un elemento diferenciador, el cual es la libertad de conducción por el escenario sin interrupciones de ningún tipo en la partida.

Las carreras pueden ser muy diversas, pero los tipos principales generalmente son dos, que son las carreras competitivas, donde se cumple un determinado número de vueltas mientras se rivaliza contra varios contrincantes por llegar al primer puesto, y las carreras contrarreloj, donde la prioridad es realizar la vuelta más corta en tiempo al circuito, habiendo también un límite de vueltas a realizar.

En el caso de “Fastastic Roads”, se trata de un videojuego de carreras arcade. Estas carreras son únicamente contrarreloj, pero el objetivo en adelante es añadir un modo de carreras competitivas en las que se puedan utilizar habilidades acordes a los vehículos que maneje el usuario. Por ello, las partidas se realizan en circuitos que son cerrados (no son abiertas, no hay posibilidad de moverse libremente fuera del escenario) y ha de cumplirse el número de vueltas requerido mientras un cronómetro de tiempo cuenta el tiempo que se tarda en dar una vuelta y un marcador indica el mejor tiempo realizado. Al acabar las vueltas indicadas, la partida finaliza devolviendo al usuario al menú hasta que se decida empezar una nueva partida.

Puntos de control

Los *checkpoints* o puntos de control son posiciones del circuito, generalmente estratégicas y con un sentido dentro del juego (previo a bifurcaciones, caídas peligrosas y otros) indicadas previamente a elección del diseñador de juego. Estos puntos registran el cruce del jugador a través de ellos, de manera que permite realizar un conteo de vueltas legal, en el cual el jugador habrá pasado en un orden concreto sin posibilidad a realizar trampas de ningún tipo, y así obtener una vuelta válida.

Así mismo, en caso de que el vehículo se salga de la pista o vuelque, estos puntos de control permiten recolocar el vehículo del jugador en el último punto de control atravesado, para poder continuar la carrera de nuevo.

Técnicas y herramientas

La decisión de escoger Unity como motor de videojuegos en el que trabajar fue motivada por el conocimiento previo del lenguaje que maneja para sus *scripts*, el cual es C#, frente al que usan los otros dos motores principales, Unreal Engine y CryEngine, el cual es C++.

Considerando esto, las herramientas así como las metodologías fueron escogidas teniendo en cuenta el posible flujo de trabajo que podía llevar el proyecto en adelante. En este capítulo se procederá a explicarlas.

4.1. Unity

Unity es un motor de videojuegos multiplataforma (esto es, permite exportar videojuegos a distintos sistemas y dispositivos, como Windows, Mac, Linux, iOS, Android, PlayStation y Xbox, entre otros) que ofrece la posibilidad de crear videojuegos en 2D, 3D y otras aplicaciones y experiencias audiovisuales.

Entre todos los disponibles, se ha hecho uso de este motor para desarrollar “Fastastic Roads”.

4.2. Lenguaje de programación C#

C# es un lenguaje de programación multiparadigma (soporta más de un paradigma de programación, i.e., una propuesta adoptada por multitud de programadores en cuanto a la resolución de determinados problemas claramente delimitados), desarrollado por Microsoft para su plataforma .NET.

Su sintaxis deriva de C/C++ y utiliza el modelo de objetos de la plataforma de .NET, similar al de Java.

Destaca respecto a otros lenguajes en varios aspectos:

- Es un lenguaje seguro. Si no se inicia una variable no puede ser utilizada.
- Es sencillo. Frente a otros lenguajes como C++, C# intenta simplificar la sintaxis para ser más consistente y lógico.
- Es accesible. La creación de componentes es directa, así como la referencia a esos componentes en código y el uso de espacios de nombres o *namespaces*.
- Es gratuito.

Unity tiene como lenguaje base para sus scripts C#, por lo que para programar para este motor es la única opción disponible.

4.3. Visual Studio Community 2019

Como entorno de programación se decidió escoger Visual Studio. Se trata de un entorno de desarrollo integrado (IDE) para Windows y macOS. Es compatible con numerosos lenguajes de programación, tales como C#, C++ y otros.

Además, Unity tiene integración directa con Visual Studio, por lo que carga correctamente sus *namespaces*, pudiendo hacer uso el usuario de todas las herramientas que ofrece (e.g., “IntelliSense”) de manera cómoda y sencilla.

4.4. LaTeX

Para la documentación se ha hecho uso de LaTeX. Es un sistema de composición tipográfica (textos) de alta calidad, incluyendo funcionalidades diseñadas para la producción de documentación técnica y científica. Permite al usuario centrarse en el contenido de la documentación antes que en el diseño del mismo, pues ejerce de plantilla fija sin tener que preocuparse de formatear continuamente [26].

4.5. Texmaker

Texmaker es un editor gratuito multiplataforma para escribir documentos de texto. Este editor incluye por defecto numerosas herramientas necesarias para desarrollar documentos con LaTeX, por lo que es una ventaja frente a otras disponibles debido a que no hay que descargar e instalar nuevas bibliotecas para su uso [27].

Se escogió frente a otras plataformas, además, por su uso como programa *offline*. Los inconvenientes encontrados han sido respecto a la hora de la compilación donde, en el caso de haber código erróneo, no marca adecuadamente dónde se encuentra el fallo, por lo que puede ser más costoso encontrarlo aun siendo muy leve.

4.6. Blender

Para el apartado artístico, el equipo ha hecho uso de Blender. Se trata de un programa multiplataforma dedicado especialmente al modelado, animación, creación de gráficos 3D y renderizado, además de composición digital. Además de ser una herramienta muy potente, es gratuito y de código abierto, lo cual permite a los usuarios notificar fallos y corregirlos con más rapidez y facilidad que otros *software* dependientes de una única empresa [28].

Aunque el proyecto de “Fastastic Roads”, desde la parte relativa al presente trabajo, no requería el uso de este programa, algunas veces ha sido necesario modificar la configuración de determinados modelos adjuntados para poder ser implementados correctamente en Unity. Es por ello que se cuenta el uso de Blender como relevante en el proyecto.

4.7. Kanban

Kanban es una metodología ágil para gestionar el trabajo. Se formuló como una aproximación al proceso evolutivo e incremental y al cambio de sistemas para las organizaciones de trabajo, estando más enfocado en llevar a cabo las tareas pendientes.

Se basa en los tablones *kanban* de la metodología japonesa, donde lo que tenemos es un ciclo de vida con cuatro fases: pendiente, en proceso, evaluación y finalizado. Las tareas comienzan como pendientes, a posteriori pasan a en proceso. Después, cuando se dan por terminadas, se trasladan a

evaluación para que las valide el tutor, y en caso de pasar la validación, son marcadas como cerradas, y en caso contrario se devuelven a en progreso.

Es más adecuado para los procesos de integración continua en los que no se tiene que hacer periódicamente entregas o micro entregas y se definen una serie de tareas para esas entregas, sino que se van tomando esas tareas y, en función de su prioridad, se van haciendo y se van validando tan pronto como estén antes de pasar a la siguiente.

La idea principal de Kanban es medir los tiempos, y su sencillez y flexibilidad es mayor frente a SCRUM, ya que no impone roles en el equipo.

4.8. GitHub

En el control de versiones se ha optado por el uso de GitHub. Se trata de una de las plataformas de desarrollo colaborativo más extendidas actualmente, permitiendo alojar proyectos utilizando el sistema de control de versiones Git. Esta plataforma permite gestionar las tareas a realizar mediante el uso de tablas estilo Kanban, la creación de una Wiki por cada proyecto, gráficos para ver cómo los desarrolladores trabajan en sus repositorios y más funciones.

Aspectos relevantes del desarrollo del proyecto

Realizar un videojuego es un proceso largo y costoso, con numerosos factores que influyen en la rapidez y dificultad de poder desarrollar e implementar correctamente todas las funciones y requisitos deseados.

Gracias a los recursos *online* disponibles, entre ellos la propia documentación de Unity, se ha podido llevar a cabo determinadas resoluciones, así como las propias planteadas para el proyecto. Por ello, en este apartado se explica en profundidad los aspectos clave del desarrollo del proyecto, entre ellos los problemas surgidos y la manera de solventarlos.

5.1. Inicio del proyecto

El 10 de marzo de 2020 se inauguró oficialmente la actividad de ÍTACA (Centro en Innovación y Tecnología de Videojuegos y Comunicación Audiovisual) en la Universidad de Burgos, cuya finalidad es desarrollar iniciativas y actividades de investigación, docencia y emprendimiento universitario en el ámbito de la industria de los videojuegos y otras nuevas tecnologías de la comunicación audiovisual [29].

Este centro ofertó la posibilidad de realizar un Trabajo de Fin de Grado con ellos en forma de videojuego. Una estudiante del curso pasado, Claudia Calvo Cuetos, realizó un Trabajo de Fin de Grado relacionado con una idea a futuro de realizar un videojuego de carreras divulgativo con mujeres ingenieras como protagonistas [30].

Fue este llamativo conjunto de ideas y forma la motivación para escogerlo como trabajo, pues el ámbito de los videojuegos siempre ha sido algo muy atractivo y poder indagar en profundidad en este campo podía ser muy apasionante.

Escogido el proyecto y el motor de videojuego en el cual se realizaría, y una vez se acordó quiénes lo tutorizarían, se procedió a comenzar la organización del proyecto.

Al tratarse de un proyecto multidisciplinar en el cual se ha participado por parte de Ingeniería Informática y por parte de ÍTACA, donde se han visto involucrados otros Trabajos de Fin de Grado/Máster en el proyecto, primeramente, se mantuvieron varias reuniones de toma de contacto y para organizar las tareas a lo largo de las semanas.

Inicialmente, debido al desconocimiento de este motor, se dedicaron los primeros meses a la investigación para aprender a entenderlo y usarlo. Una vez se comprendieron las bases, se dio comienzo al desarrollo.

5.2. Metodología de desarrollo Kanban

Al tratarse de un proyecto en el cual el flujo de trabajo es continuo, la metodología que se ha seguido ha sido Kanban. Permite realizar las tareas a medida que van surgiendo y acabándolas según la importancia que tengan. Además, es una metodología más flexible que otras existentes actualmente, lo que ha permitido poder dedicar tiempo a investigar sobre el funcionamiento de las herramientas, ya que en el caso de usar otra metodología con ciclos predefinidos habría imposibilitado el correcto cumplimiento de las fechas.

El funcionamiento ha sido explicado en el capítulo 4 “[Técnicas y Herramientas](#)”. Aplicándolo a este proyecto, ha permitido tener un margen de aprendizaje sobre el funcionamiento de Unity, incluyendo las pruebas realizadas con modelos, componentes y otros elementos.

Este flujo de trabajo permite, además, ajustar en el futuro las estimaciones que se hagan respecto a nuevas tareas similares y tomar este histórico de valores como referencia para el coste de cuánto supondría, ayudando también a que usuarios de otros proyectos similares no tengan que obtener datos, sino que ya pueda partir de estas mediciones para saber cuánto tiempo puede conllevar realizarlas.

Gracias a esta metodología de desarrollo, se ha podido mantener un ritmo de trabajo constante, así como ha permitido realizar las paradas necesarias

para asimilar nuevos conceptos, corregir fallos y reformular la forma de completar las tareas, de manera que la presión en base al tiempo, teniendo en cuenta el contexto, ha sido menor y se ha podido llevar a cabo de una manera más fluida.

De igual forma, sí que se llevaron a cabo *sprints*, de una duración de aproximadamente dos semanas por *sprint* para ir llevando el control mencionado atrás del número tareas completadas y el cambio de fases.

5.3. Formación

Dado que al principio del proyecto había desconocimiento sobre el funcionamiento de Unity, se requirió de documentación externa para obtener los conocimientos necesarios para la realización correcta del proyecto.

Inicialmente se recurrió a recursos tutoriales en forma de vídeo adjuntados por el tutor, donde previamente explicaba los principios básicos de funcionamiento de Unity, así como sus elementos principales y el funcionamiento. Términos como “transformada”, “*GameObject*”, “*Quaternion*”, “componente”, “asset” o “prefab” entre otros, cuyo conocimiento previo sobre ellos era nulo, sentaron las primeras bases para poder empezar a hacer pruebas en el motor.

Una vez comenzó el desarrollo se recurrió a la misma documentación de Unity para comprender la utilidad y el *manejo* de determinadas funciones del motor por código, así como a más recursos tutoriales en forma de vídeo respecto a la creación ciertos métodos, de los cuales se obtenía la forma de realizarlo y luego se integraba en base a la necesidad en el proyecto, dando muchos casos en los que la creación de los métodos partía totalmente de cero y otros en los que se dio la necesidad de refactorizar y adaptar.

5.4. Desarrollo del proyecto

Como se ha comentado previamente, los primeros meses fueron de formación, aunque durante el desarrollo requiriése también de cierto aprendizaje respecto a nuevos conceptos y herramientas, por lo que comenzó algo más tardío el *arranque* del mismo.

Para explicar las fases de desarrollo, se dividirá en tres fases correspondientes a las versiones del proyecto: *alpha*, *beta* y *Release candidates*.

Alpha

Inicialmente, se comenzó creando un escenario de pruebas simple conformado por dos planos, siendo estos dos objetos de tipo cubo con el grosor mínimo para obtener la forma aplanada, y ensanchados y alargados, siendo uno de ellos algo menor de tamaño que el otro para hacer superficie sobre superficie. De esta manera, se obtenía un pequeño desnivel por el que, a continuación, bajo un modelo de vehículo simple hecho a partir de cilindros como cuerpo y ruedas, se realizaban las pruebas.

Este modelo de vehículo simple, por sí mismo, no tiene utilidad, pues es solo un cuerpo “inerte”. Ahí es donde comenzaron los primeros pasos grandes con la programación. Debido a que un sistema de conducción básico es relativamente complejo, se decidió partir de uno ya existente a partir del cual se iba a refactorizar las nuevas funciones a implementar en caso de ser necesario con el mismo, además de la creación de un sistema de cámara en tercera persona básico. Las pruebas que se hicieron al respecto llevaban a ver cómo el vehículo tenía comportamientos no esperados con el componente añadido para la conducción, con vibraciones no deseadas y otros aspectos. Hubo que realizar cambios en la configuración del componente hasta tenerlo en un estado aceptable, pero tampoco se indagó mucho en ello, pues seguía siendo un escenario de pruebas y no era rentable esforzarse en hacer cambios grandes si no iban a ser el escenario y vehículo finales.

Más adelante, se obtuvo un escenario de pruebas similar al escenario final en estructura adjuntado por el equipo de ÍTACA, de manera que ahí se podía ir realizando cambios en la configuración mencionada del vehículo previo respecto al escenario.

En este escenario se empezaron a integrar los sistemas que componen un juego de carreras contrarreloj. Primero se creó un controlador de carrera encargado de instanciar los vehículos en la escena. Este controlador iría en un objeto al cual se llamaría de la misma manera para tener un control correcto de todos los componentes de la escena, sistema de nomenclatura que se seguiría de ahí en adelante.

Antes de continuar con los componentes de carrera contrarreloj, se sustituyó la cámara básica por *Cinemachine*, el cual es un *package* gratuito descargable desde Unity, aportando un sistema de cámaras dinámico, configurable y con más realismo, fluyendo con el movimiento del vehículo y siendo más agradable que una cámara estática fija.

El siguiente elemento básico que se desarrolló fue el sistema de puntos de control, con el cual se podría crear más adelante un conteo de vueltas y un

sistema de recolocación. Aprovechando los objetos que provee Unity como son los cubos, se hicieron planos en los cuales se introducían los componentes correspondientes a los puntos de control, los cuales fueron creados por código. Inicialmente eran componentes individuales, pero más adelante se cambió el sistema para que fuese el controlador de carrera el encargado.

Con el sistema creado se podía proceder a la creación del sistema de *respawn* o recolocación del vehículo (manual y automático), donde gracias a ello se puede recolocar el vehículo en caso de volcar, caer fuera del escenario u otras ocasiones en las cuales no se puede realizar ningún tipo de movimiento con el vehículo, y el conteo de vueltas para que tenga un final la partida. Entre medias también se creó un cronómetro de tiempo, el cual serviría para obtener el mejor tiempo realizado en la carrera mediante la creación de otro componente. Con estos elementos creados se tenía el principal funcionamiento de una contrarreloj.

Para el último *sprint* antes de la beta se “pulieron” todas las funciones, refactorizaron métodos, actualizado el mapa a una versión mejorada e incluidas nuevas funcionalidades, entre ellos la creación de una cuenta atrás (con el correspondiente bloqueo de vehículo previo a comenzar que conlleva), la posibilidad de acabar la carrera una vez acaben las vueltas y la creación de los menús (incluido el de pausa de partida).

Beta

Es la fase previa a la versión candidata a ser publicada. En esta fase se realizó una “limpieza” de componentes innecesarios en la escena, recolocación de *assets*, se corrigieron *bugs* varios y se cambió la lógica de algunas funciones como el conteo de puntos de control, el sistema de recolocación y el cálculo del mejor tiempo por vuelta.

Lo más destacable en esta fase es el comprobar cómo el desarrollo de las funciones tenía una cierta complejidad, pero que es a la hora de perfeccionar, corregir y cambiar la lógica de las funciones cuando aumenta esta complejidad y se hace denotar.

Release candidates

En este último *sprint* se ha actualizado el modelo de vehículo, el mapa a una versión mejorada visual y estructuralmente, así como la redacción final de la documentación.

Es la versión candidata a ser publicada como juego final en el TFG, con la *build* correspondiente lista para probar y jugar.



Figura 5.9: Fotograma de la *Release Candidate 1* de “Fastastic Roads”.

Desarrollo coordinado en un equipo multidisciplinar

Uno de los puntos destacables de este proyecto es el haber trabajado formando parte de un equipo multidisciplinar, pues se ha tenido que hacer en conjunto a un equipo de personas dedicadas a otros aspectos del juego, como han sido el escenario y los modelos 3D, así como la idea principal del juego.

Ha habido constante comunicación respecto a dudas, problemas que pudiesen surgir o necesidades momentáneas, así como reuniones cada al menos dos semanas para actualizar el progreso de cada miembro del equipo y mandar nuevas tareas en base a ese avance.

Narrativa

Aunque la narrativa está prácticamente ausente, no carece de importancia, pues la idea principal aportada por el equipo es el mostrar históricamente la implicación de los personajes en los vehículos que manejan, aportando unas características especiales que el resto de vehículos no tienen. Debido al tamaño del proyecto, se tuvo que simplificar esta idea en un juego más simple con modo a contrarreloj, teniendo la intención de continuarlo más adelante.

Arte

Para la parte artística, el resto del equipo ha sido el encargado de crear los modelos 3D para los vehículos, escenarios y animaciones bajo el software de código abierto *Blender*. Estos modelos se insertaban en Unity directamente, haciendo el mismo motor la debida importación, y en ocasiones se ha tenido que exportar a formato “.fbx” (*Filmbox*) para determinadas configuraciones, pero en general han sido en el mismo formato *Blender*.

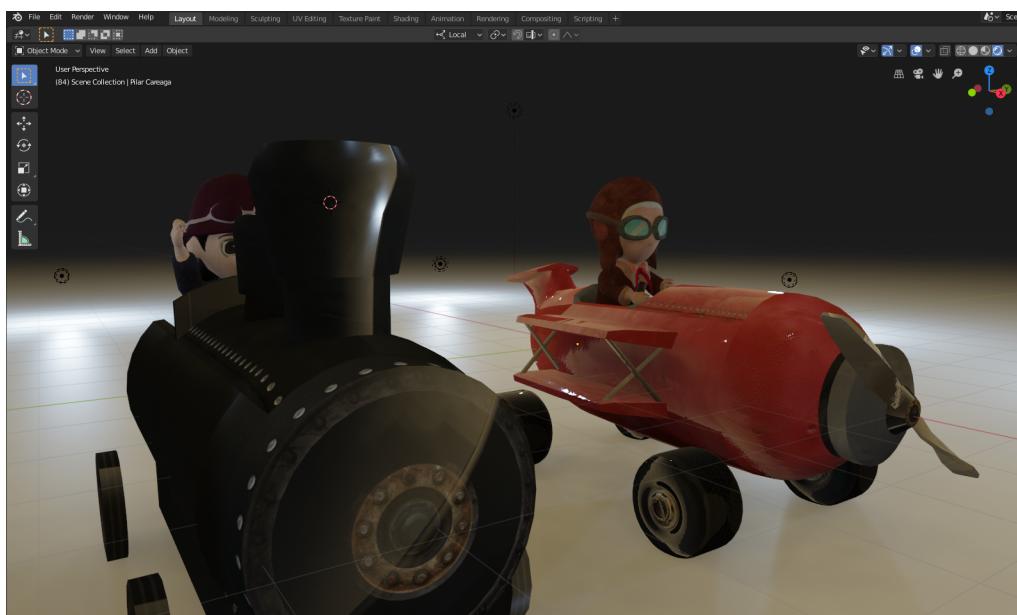


Figura 5.10: Modelos de personajes creados en *Blender*.

5.5. Problemas y resolución

Realizar un videojuego es más complicado de lo que puede aparentar, donde hasta las funciones más simples pueden tener ciertos problemas que ralentizan el correcto transcurso del desarrollo.

Al tratarse de un motor de videojuegos con su propia estructura, las pruebas automáticas no son un aspecto “amigable” con esta manera de trabajar, por lo que todas las pruebas realizadas eran manuales y realizadas según fuesen ocurriendo problemas, o bien si surgía una idea a probar.

Estos han sido varios de los problemas surgidos:

- Los *meshcolliders* del terreno quitaron un elevado número de horas de trabajo, pues ocasionaban comportamientos imprevistos en el vehículo que, aun configurándolo, impedía el correcto funcionamiento de éste. Al actualizar el mapa a una versión más reciente y con más polígonos, así como añadiendo *colliders* esféricos a las ruedas para que no atravesaran el suelo, se solucionaron gran parte de esos problemas ocasionados.
- Las funciones creadas, una vez fueron desarrolladas, funcionaban sin algún tipo de problema. En cambio, cuando se tuvieron que implementar nuevos aspectos, componentes u otros, algunos provocaron fallos inesperados en esas funciones previamente creadas, aparentemente, de manera correcta. Un ejemplo fue el sistema de conteo de vueltas, el cual, al introducir los *colliders* de las ruedas, empezó a realizar un mal conteo provocando la finalización inesperada de la partida al aumentar el número de vueltas completo, lo que hizo necesario un replanteamiento de la lógica de los *scripts* general para que nada empezara a fallar por distintos lados.
- Distintos fallos menores solventados en el momento en el que se conocía su existencia.

5.6. Documentación

En los inicios del proyecto, se planteó realizar la documentación en documento OpenOffice, pero se vio menos óptimo respecto al tiempo que conlleva formatear el documento que en otras herramientas como LaTeX, de la cual también se tenía plantilla, con lo cual se procedió a comenzar la documentación ahí haciendo uso de Texmaker.

Gracias a esta herramienta se ha podido escribir la documentación centrándose en el contenido con la correcta estructuración de los apartados, índices y elementos gráficos. Los inconvenientes encontrados han sido debidos al desconocimiento del mismo (escritura de algunos caracteres o estructuras erróneas que provocaba una compilación fallida) así como algún fallo menor fácilmente solucionable (reestructurado y reescritura de textos) ralentizando un poco el progreso en las etapas iniciales de la documentación.

5.7. Exposición social

El progreso de este proyecto se ha ido plasmando en las redes sociales de ÍTACA, obteniendo un *feedback* positivo de las personas que han visualizado las publicaciones. Además, se explicó parte de su creación y los problemas a

resolver en unas jornadas online organizadas por el mismo Centro el día 16 de junio de 2021, obteniendo también buenas valoraciones y un gran interés por el público [31].

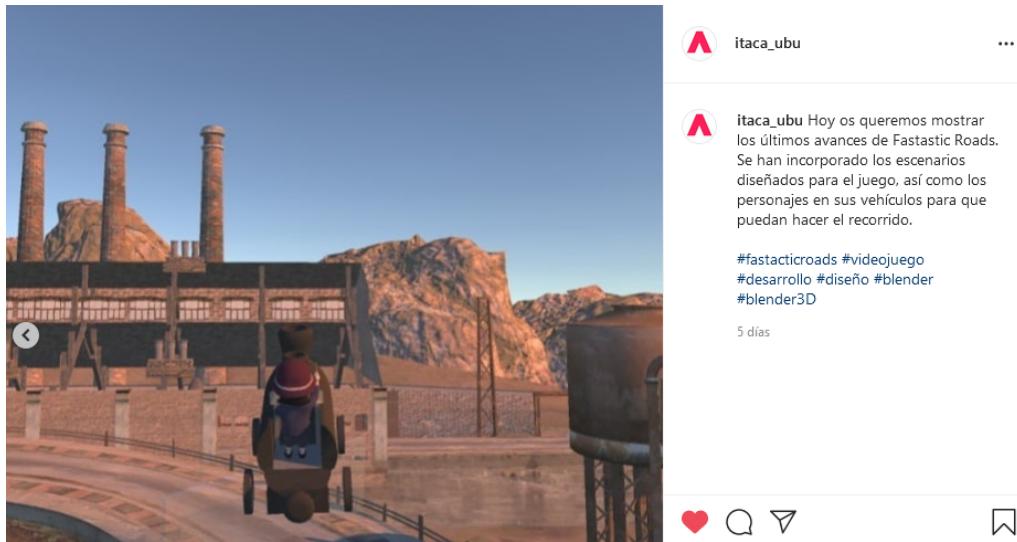


Figura 5.11: El progreso del proyecto ha podido ser visualizado en Internet.

Trabajos relacionados

El número de videojuegos disponibles al público es ingente, incluso considerando únicamente aquellos que se han creado en Unity, debido a la disponibilidad global del motor así como a la cantidad de juegos que se crean mensual y diariamente.

Es por ello que los trabajos relacionados que se mencionan a continuación tienen en común el género de carreras al cual pertenecen, entre ellos un Trabajo de Fin de Grado desarrollado en Unity y realizado en la Universidad de Burgos.

6.1. Mario Kart

Mario Kart es un videojuego de carreras de karts y a la vez una serie de videojuegos con el mismo nombre desarrollados por la compañía japonesa Nintendo. Es mundialmente conocido, además de por los personajes pertenecientes a la saga “Super Mario” como por su fácil accesibilidad para jugadores no experimentados, así como para aquellos que tengan más habilidad con este género de videojuegos.

Las mecánicas de este videojuego son una inspiración y modelo a seguir para “Fastastic Roads”, por el subgénero de carreras al que pertenecen ambos y por sus modos de juego, ya que contiene tanto el modo a contrarreloj, el cual ha sido implementado en este proyecto, como el modo Grand Prix, en el cual se compite con los distintos contrincantes para llegar antes a la meta haciendo uso de distintos objetos distribuidos por el escenario que ofrecen ventajas al jugador, tanto para “atacar” al resto de jugadores como para beneficiar temporalmente al mismo, siendo este modo un objetivo a implementar en “Fastastic Roads”. Aparte incluye otros dos modos, un

“versus” (todos contra todos en carreras) y uno de batalla múltiple (equipo contra equipo con objetivos).

Además, cada vehículo tiene mejores o peores habilidades respecto a velocidad, aceleración, peso, manejo y agarre.

6.2. Crash Team Racing

Se trata de un videojuego de karts muy inspirado en “Mario Kart”. Contiene sus mismos modos de juego, así como alguno nuevo añadido, pero siguiendo en general las mismas líneas de jugabilidad, con personajes relacionados con los videojuegos de *Crash Bandicoot*, vehículos con mejoras en velocidad, aceleración y giro, etc.

6.3. UBURACE

Este proyecto, creado por Uxía Doval Blanco en el año 2016 para el “Grado en Comunicación Audiovisual”, es un videojuego de carreras creado con la intencionalidad de aprender a realizar un juego cumpliendo las indicaciones de un cliente con todos los pasos que conlleva. Está planteado como prototipo desde el ámbito de la Comunicación Audiovisual, siendo la parte de desarrollo de *software* secundaria y, por lo tanto, poco relevante para este estudio.

Este proyecto era ambicioso respecto a que una misma persona era la encargada de todas las fases: ilustración, modelado y texturizado 3D, animación y programación.

Los requisitos para la realización de este proyecto de juego de carreras son un contador de vueltas y un sumador de puntos en base a ciertos objetos disponibles. Los modelos tienen *colliders* sencillos y el funcionamiento es más simple, aprovechando en el apartado de programación *assets* previamente programados sin seguir ningún tipo diseño de *software*. Contiene dos escenarios, estando uno de ellos basado en la Facultad de Económicas de la Universidad de Burgos. Al tratarse de un proyecto de Comunicación Audiovisual, está más enfocado a ese campo junto a los principios de diseño de videojuegos que a ser un proyecto enfocado al *software*.

Conclusiones y Líneas de trabajo futuras

Para finalizar, en este capítulo se exponen las conclusiones extraídas del proyecto, así como las líneas de trabajo a seguir en el futuro.

7.1. Conclusiones

Tras haber finalizado el desarrollo del proyecto, se ha podido comprobar cómo se han cumplido los objetivos del proyecto de manera satisfactoria.

Como se ha mencionado anteriormente, la realización de un videojuego requiere de varias personas distribuidas por los diferentes campos (programación, modelaje, etc.), donde se ha tenido la oportunidad de poder contar con un equipo dedicado a la parte artística, para así dedicarse de manera individual a la parte relacionada con la programación y el *software*.

Por ello, es un proyecto con bastante ambición, donde se calcularon una serie de requisitos que sí podían llegarse a completar con este número reducido de personas encargadas, y el resultado ha sido altamente satisfactorio.

Se ha partido del desconocimiento absoluto del motor, requiriendo de su aprendizaje en funcionalidades, manejo, gestión de archivos, configuración por proyecto, programación y otros aspectos, consiguiendo en pocos meses los conocimientos básicos para poder completar este juego y con un gran camino por delante para seguir aprendiendo.

Así mismo, se ha conseguido realizar los propósitos planteados como videojuego de carreras contrarreloj, con menús amigables y comprensibles

tanto para aquellas personas que no estén familiarizadas como para las que sí y con una ausencia mayoritaria de errores en los componentes esenciales.

Y no solo se ha obtenido un juego de carreras contrarreloj jugable, también disfrutable, pudiendo completar vueltas en un entorno virtual agradable, invitando al usuario a tratar de batir su propio récord empezando tantas partidas como desee, hasta que voluntariamente decida dejar de jugar.

Por supuesto, se ha comprendido el costoso proceso que conlleva realizar un proyecto de esta magnitud, concretamente un proyecto relacionado con videojuegos. Muchos elementos básicos que poseen los videojuegos generalmente y se presuponen que son intrínsecos de base no son triviales.

En definitiva, ha sido una oportunidad para aprender y comprender el ámbito de los videojuegos y todo lo que le rodea, sentando unas bases para poder continuar otros proyectos similares relacionados con videojuegos o diseño y programación en Unity.

7.2. Líneas de trabajo futuras

Este proyecto no ha llegado a su fin, pues “Fastastic Roads” es un videojuego con intención de continuar su desarrollo en ÍTACA. Por ello, los objetivos a futuro del proyecto son los siguientes:

- Integrar nuevos circuitos.
- Integrar los nuevos modelos de vehículos para escoger.
- Crear un comportamiento determinado para cada vehículo, implementando habilidades que hagan a cada uno de ellos único y diferente del resto.
- Investigar la creación de jugadores no humanos (CPU) para competir en las partidas con ellos e integrar ese sistema en el videojuego.
- Crear un modo de carrera competitiva. Este modo permitirá competir con otros jugadores no humanos, donde el ganador es el primero en llegar a meta. Para hacer el modo más atractivo, los vehículos podrán recoger determinados ítems que estarán en la escena colocados para su uso contra los rivales.
- Crear un modo multijugador a pantalla partida, permitiendo al usuario competir con otra persona en un mismo ordenador en los distintos modos que ofrece el juego.
- Mejorar la interfaz gráfica para que sea más agradable y llamativo para el usuario.

- Integrar un sistema de audio, así como los respectivos sonidos de efectos y música al juego.
- Hacerlo más accesible, dando la posibilidad de configurar distintas opciones como son las opciones de volumen, pantalla y controles, entre otros.
- Nuevas funcionalidades, mejoras y correcciones que vayan surgiendo durante el transcurso del tiempo.

Bibliografía

- [1] Asociación Española de Videojuegos. El videojuego en el mundo. URL: <http://www.aevi.org.es/la-industria-del-videojuego/en-el-mundo/>, 2020. Último acceso: 03/07/2021.
- [2] Asociación Española de Videojuegos. El videojuego en españa. URL: <http://www.aevi.org.es/la-industria-del-videojuego/en-espana/>, 2020. Último acceso: 03/07/2021.
- [3] Manuel Santos. Motores gráficos: qué son y por qué son tan útiles. URL: <https://hardzone.es/2018/05/06/motor-grafico-juegos/>, May 2018. Último acceso: 04/07/2021.
- [4] Manuel Delgado. Los 11 mejores motores gráficos para introducirse en el desarrollo de videojuegos. URL: <https://vandal.elespanol.com/reportaje/los-11-mejores-motores-graficos-para-introducirse-en-el-desarrollo-de-videojuegos>, February 2021. Último acceso: 04/07/2021.
- [5] Unity Documentation. Escena. URL: <https://docs.unity3d.com/es/2020.1/Manual/CreatingScenes.html>, 2020. Último acceso: 04/07/2021.
- [6] Unity Documentation. Gameobject. URL: <https://docs.unity3d.com/es/2019.4/Manual/class-GameObject.html>, 2019. Último acceso: 04/07/2021.
- [7] Unity Documentation. Transform. URL: <https://docs.unity3d.com/es/2021.1/Manual/class-Transform.html>, 2021. Último acceso: 04/07/2021.

- [8] Unity Documentation. Introducción a los componentes. URL: <https://docs.unity3d.com/es/2019.3/Manual/Components.html>, 2019. Último acceso: 04/07/2021.
- [9] Unity Documentation. Creando y usando scripts. URL: <https://docs.unity3d.com/es/2021.1/Manual/CreatingAndUsingScripts.html>, 2021. Último acceso: 04/07/2021.
- [10] Unity Documentation. Mesh renderer. URL: <https://docs.unity3d.com/es/2020.1/Manual/class-MeshRenderer.html>, 2020. Último acceso: 04/07/2021.
- [11] Unity Documentation. Colliders. URL: <https://docs.unity3d.com/es/2019.1/Manual/CollidersOverview.html>, 2019. Último acceso: 04/07/2021.
- [12] Unity Documentation. Mesh colliders. URL: <https://docs.unity3d.com/es/2019.2/Manual/class-MeshCollider.html>, 2019. Último acceso: 04/07/2021.
- [13] Unity Documentation. Wheel colliders. URL: <https://docs.unity3d.com/es/2021.1/Manual/class-WheelCollider.html>, 2021. Último acceso: 04/07/2021.
- [14] Unity Documentation. Lights (luces). URL: <https://docs.unity3d.com/es/2021.1/Manual/Lights.html>, 2021. Último acceso: 04/07/2021.
- [15] Unity Documentation. Cámaras. URL: <https://docs.unity3d.com/es/2019.3/Manual/CamerasOverview.html>, 2019. Último acceso: 04/07/2021.
- [16] EDU Xunta Xunta de Galicia. Perspectiva cónica. URL: https://www.edu.xunta.gal/espazoAbalar/sites/espazoAbalar/files/datos/1464946300/contido/9_perspectiva_cnica.html. Último acceso: 04/07/2021.
- [17] Unity Docs. About cinemachine. URL: <https://docs.unity3d.com/Packages/com.unity.cinemachine@2.1/manual/index.html>, 2017. Último acceso: 04/07/2021.
- [18] Unity Documentation. Física. URL: <https://docs.unity3d.com/es/2019.3/Manual/PhysicsSection.html>, 2019. Último acceso: 04/07/2021.

- [19] Unity Documentation. Rigidbody. URL: <https://docs.unity3d.com/es/2021.1/Manual/RigidbodiesOverview.html>, 2021. Último acceso: 04/07/2021.
- [20] Unity Learn. Assets, objects and serialization. URL: <https://learn.unity.com/tutorial/assets-resources-and-assetbundles>, 2018. Último acceso: 04/07/2021.
- [21] Unity Documentation. Prefabs. URL: <https://docs.unity3d.com/es/2018.2/Manual/Prefabs.html>, 2018. Último acceso: 04/07/2021.
- [22] Unity. Renderizado en tiempo real en 3d. URL: <https://unity3d.com/es/real-time-rendering-3d>, 2021. Último acceso: 04/07/2021.
- [23] Unity Documentation. Input. URL: <https://docs.unity3d.com/es/2017.1/Manual/Input.html>, 2017. Último acceso: 04/07/2021.
- [24] Unity Documentation. Audio. URL: <https://docs.unity3d.com/es/2018.1/Manual/class-AudioManager.html>, 2018. Último acceso: 04/07/2021.
- [25] Wikijuegos. Arcade (género). URL: [https://videojuegos.fandom.com/es/wiki/Arcade_\(g%C3%A9nero\)](https://videojuegos.fandom.com/es/wiki/Arcade_(g%C3%A9nero)), 2021. Último acceso: 04/07/2021.
- [26] The LaTeX Project. Latex - a document preparation system. URL: <https://www.latex-project.org/>, 2021. Último acceso: 04/07/2021.
- [27] Pascal Brachet. Texmaker - free cross-platform latex editor. URL: <https://www.xm1math.net/texmaker/>. Último acceso: 04/07/2021.
- [28] Blender. Blender. URL: <https://www.blender.org/about/>, 2021. Último acceso: 04/07/2021.
- [29] UBU. Itaca arranca de forma online. URL: <https://www.ubu.es/noticias/itaca-arranca-de-forma-online>, March 2020. Último acceso: 04/07/2021.
- [30] E. Lera. Un videojuego de ingeniería en clave femenina. URL: <https://diariodevalladolid.elmundo.es/articulo/innovadores/videojuego-ingenieria-clave-femenina/20200422180931378664.html>, April 2020. Último acceso: 04/07/2021.
- [31] ÍTACA. Jornadas de creación de videojuegos en el entorno universitario. URL: https://youtu.be/13em_VtbpGE?t=4112, June 2021. Último acceso: 04/07/2021.