

Grado en diseño y producción de videojuegos

DESARROLLO DE UN PROTOTIPO DE JUEGO EN RED

Anteproyecto

ERIK ESPUÑES JUBERÓ

TUTOR: JORDI ARNAL MONTOYA

CURSO 2020-2021



Abstract

This project will focus on creating a prototype kart videogame with online multiplayer. The prototype will consist of developing a client, which will be the one that will have the player, and a server, which will be the one that manages the multiplayer.

Resum

Aquest projecte es centrarà en crear un prototip de videojoc de karts amb multijugador en línia. El prototip constarà en elaborar un client, que serà el que tindrà el jugador, i un servidor, que serà el que gestioni el multijugador.

Resumen

Este proyecto se centrará en crear un prototipo de videojuego de karts con multijugador en línea. El prototipo constará en elaborar un cliente, que será lo que tendrá el jugador, y un servidor, que será el que gestione el multijugador.

ÍNDICE

[ÍNDICE DE FIGURAS 3](#_Toc63365861)

[ÍNDICE DE TABLAS 5](#_Toc63365862)

[GLOSARIO DE TÉRMINOS 7](#_Toc63365863)

[CAPÍTULO 1. INTRODUCCIÓN 1](#_Toc63365864)

[CAPÍTULO 2. OBJETIVOS 3](#_Toc63365865)

[2.1. OBJETIVOS PRINCIPALES 3](#_Toc63365866)

[2.2. OBJETIVOS SECUNDARIOS 3](#_Toc63365867)

[CAPÍTULO 3. ANÁLISIS DE REFERENTES 5](#_Toc63365868)

[3.1. REFERENTES DE JUEGOS MULTIJUGADOR EN LÍNEA 5](#_Toc63365869)

[3.1.1. ROCKET LEAGUE 5](#_Toc63365870)

[3.1.2. AMONG US 5](#_Toc63365871)

[3.2. REFERENTES DE JUEGOS DE CARRERAS ARCADE 5](#_Toc63365872)

[3.2.1. MARIO KART 8 5](#_Toc63365873)

[3.2.2. TEAM SONIC RACING 9](#_Toc63365874)

[3.2.3. CRASH TEAM RACING NITRO FUELED 10](#_Toc63365875)

[CAPÍTULO 4. MARCO TEÓRICO 13](#_Toc63365876)

[4.1. DEL MULTIJUGADOR LOCAL AL MULTIJUGADOR EN LÍNEA 13](#_Toc63365877)

[4.1.1. EVOLUCIÓN DE LOS MULTIJUGADORES 13](#_Toc63365878)

[4.1.2. MULTIJUGADOR LOCAL 14](#_Toc63365879)

[4.1.3. MULTIJUGADOR EN LÍNEA 14](#_Toc63365880)

[4.2. PROTOCOLOS PARA LA COMUNICACIÓN EN TIEMPO REAL 15](#_Toc63365881)

[4.2.1. ¿CÓMO FUNCIONA INTERNET? 15](#_Toc63365882)

[4.2.2. TRANSPORT CONTROL PROTOCOL (TCP) 16](#_Toc63365883)

[4.2.3. USER DATAGRAM PROTOCOL (UDP) 16](#_Toc63365884)

[4.2.4. WEBSOCKETS 16](#_Toc63365885)

[4.3. METODOLOGÍAS PARA EL DESARROLLO DE VIDEOJUEGOS EN LÍNEA 16](#_Toc63365886)

[4.3.1. CLIENTE-SERVIDOR 16](#_Toc63365887)

[4.3.2. PEER-TO-PEER (P2P) 17](#_Toc63365888)

[4.3.3. SERVIDOR-SERVIDOR 17](#_Toc63365889)

[4.4. FUNCIONAMIENTO DE UN COCHE 17](#_Toc63365890)

[CAPÍTULO 5. DISEÑO METODOLÓGICO Y CRONOGRAMA 19](#_Toc63365891)

[5.1. HERRAMIENTAS DE DESARROLLO 19](#_Toc63365892)

[5.1.1. HERRAMIENTAS DE DESARROLLO DEL CLIENTE 19](#_Toc63365893)

[5.1.2. HERRAMIENTAS DE DESARROLLO DEL SERVIDOR 20](#_Toc63365894)

[5.2. METODOLOGÍA 21](#_Toc63365895)

[5.2.1. METODOLOGÍA DE LA DOCUMENTACIÓN 21](#_Toc63365896)

[5.2.2. METODOLOGÍA DEL PRODUCTO 22](#_Toc63365897)

[5.3. PLANIFICACIÓN 22](#_Toc63365898)

[CAPÍTULO 6. BIBLIOGRAFÍA 25](#_Toc63365899)

# ÍNDICE DE FIGURAS

[Figura 3.1: Controles de Mario Kart. 6](#_Toc63365900)

[Figura 3.2: Tabla de probabilidad de los objetos en Mario Kart 7](#_Toc63365901)

[Figura 3.3: Modos de juego de Mario Kart. 8](#_Toc63365902)

[Figura 3.4: Controles de Team Sonic Racing 9](#_Toc63365903)

[Figura 3.5: Controles Crash Team Racing Nitro Fueled 10](#_Toc63365904)

[Figura 4.1: Imagen del juego de Empire. 13](#_Toc63365905)

[Figura 4.2: Topología Cliente-Servidor 17](#_Toc63365906)

[Figura 5.1: Logotipos de *Unreal Engine* y *JetBrains Rider* respectivamente. 20](#_Toc63365907)

[Figura 5.2: Logotipos de *JetBrains WebStorm* y *Node.js*. 21](#_Toc63365908)

# ÍNDICE DE TABLAS

[Tabla 4.1: Modelo de capas OSI (ISO/IEC JTC 1, 1994) 16](#_Toc63365909)

[Tabla 5.1: Cronograma del TFG 23](#_Toc63365910)

# GLOSARIO DE TÉRMINOS

|  |  |
| --- | --- |
| HUD | Head-Up Display |
| LAG |  |
| LAN |  |
| SERVIDOR |  |
| TFG | Trabajo de Fin de Grado |
| IA | Inteligencia Artificial |
| ECS | Entity Component System |
| DOTS | Data Oriented Technology Stack |
| IDE |  |

# INTRODUCCIÓN

Por lo tanto, en este TFG se hará un juego de conducción arcade centrado en el multijugador. Así que para hacer este producto se separará en tres partes.

Primero se hará la parte del cliente, que será como tener el juego sin multijugador. Allí estará implementado todas las mecánicas que existan en el juego. Por ejemplo, el sistema de conducción, la inteligencia artificial, los objetos, las reglas del juego y el HUD

Una vez la parte del cliente funciona perfectamente se hará el servidor. Este será el encargado de gestionar la partida multijugador, contendrá una base de datos para poder guardar estadísticas de nuestros jugadores, entre otros. Será un servidor autoritativo, por lo que todos los inputs de los jugadores irán a la instancia del juego que estará ejecutando el servidor.

Para acabar se hará la integración del cliente con el servidor. Lo que se hará es hacer el modo de juego del multijugador, que hará que el cliente se comunique con el servidor.

# OBJETIVOS

Los objetivos que ahora se mencionarán se separarán en el objetivo principal y los secundarios. Los principales serán, los requisitos necesarios para que se pueda llegar a la conclusión de que el proyecto ha sido un éxito. Los secundarios serán los que ayudarán a saber si la conclusión del proyecto será sobresaliente o, por el contrario, suficiente.

## OBJETIVOS PRINCIPALES

*Hacer que el multijugador funcione sin errores importantes.*

Al ser un prototipo de un videojuego en red, el TFG estará centrado en eso. Por eso, ocupará dos tercios del desarrollo del prototipo, hacer el sistema multijugador. En cuanto a errores importantes se considera que son aquellos que hagan que la experiencia de juego de los jugadores de la partida sea mala.

*Evitar y gestionar a los jugadores que hagan trampas.*

*Tener un sistema para que el jugador siempre pueda jugar multijugador.*

## OBJETIVOS SECUNDARIOS

*Hacer que la conducción sea divertida.*

Este objetivo podría ser principal, pero se ha considerado que es secundario, ya que es muy subjetivo. Un sistema que sea divertido para alguien puede no serlo para otro. Aunque se intentará que la conducción sea divertida para la mayoría de gente que lo pruebe.

*Evitar que el juego no tenga bugs graves.*

Este debe ser también casi principal, ya que no se quiere que el juego sea injugable por errores. Así que se intentará que el juego esté lo más pulido posible.

*Hacer que los objetos estén bien balanceados.*

*Intentar que el juego funcione en la mayoría de las configuraciones que pueden tener los ordenadores.*

*Documentar todas las decisiones tomadas en el desarrollo del videojuego.*

# ANÁLISIS DE REFERENTES

Para analizar los referentes se debe separar en dos partes, los referentes de multijugador en línea y los referentes de juegos de karts.

## REFERENTES DE JUEGOS MULTIJUGADOR EN LÍNEA

En este subapartado solo se analizarán las comunicaciones del sistema multijugador, se analizará los envíos de paquetes del cliente al servidor, y se intentará ver lo que hacen bien o mal.

### ROCKET LEAGUE

TODO

#### FUNCIONAMIENTO ONLINE

TODO

### AMONG US

TODO

#### FUNCIONAMIENTO ONLINE

TODO

## REFERENTES DE JUEGOS DE CARRERAS ARCADE

En este subapartado solo se verán la configuración de los controles, las mecánicas y los modos de juegos que tiene cada juego que se analizará. Se analizarán tres juegos de este género, Mario Kart 8, Team Sonic Racing y Crash Team Racing Nitro Fueled.

### MARIO KART 8

Mario Kart 8 (Nintendo, 2013), que fue lanzado por Nintendo en el 2013, es el octavo juego de la saga Mario Kart. Dicha saga se caracteriza por unas carreras arcades asimétricas de, normalmente, doce jugadores. En dicha carrera solo hay una regla, seguir el trazado, aunque hay algún atajo. En las carreras, también se pueden coger objetos, que, aunque son aleatorios, son una manera muy fácil de ganar posiciones o defender la que tienes.

A continuación, se explicarán la configuración de los controles que tiene el juego, sus mecánicas y para acabar sus modos de juego.

#### CONTROLES



Figura 3.1: Controles de Mario Kart.

Teniendo como referencia el mando de la Switch, Mario Kart tiene los siguientes controles:

* A: Acelerar
* B: Frenar
* L: Lanzar objeto
* R: Derrapar
* Joystick Izquierdo: Girar (aunque también se puede habilitar la opción de girar rotando el mando como si fuese el volante.)

#### MECÁNICAS

En este apartado se analizará la mecánica del movimiento del personaje, como también los objetos que hay en el juego.

Primero de todo el personaje tiene cuatro mecánicas simples, acelerar, frenar, girar y derrapar.

La primera mecánica, la de acelerar, es muy simple, se pulsa el botón de acelerar y el vehículo acelera hasta llegar a su velocidad máxima. Si se deja de acelerar el vehículo llegará a frenarse por completo, pero tardando más que si frenas pulsando el botón de frenar.

El freno es muy parecido a la de acelerar, si se pulsa su botón, éste llegará a frenar el coche. Pero si se sigue frenando, una vez el vehículo llega a velocidad cero, éste irá marcha atrás. Yendo marcha atrás si dejas de pulsar el botón, el vehículo se frenará por completo, de la misma forma que pasa al acelerar.

El giro es bastante simple, se mueve el joystick, o el mando si se ha configurado de tal manera, y el coche gira. Esta mecánica se junta con la siguiente, ya que si se está mucho rato girando el kart empezará a derrapar.

La mecánica de derrape te ofrece hacer un giro con más ángulo que girando de manera normal y además te da un turbo extra que va augmentando en función de la duración del derrape.

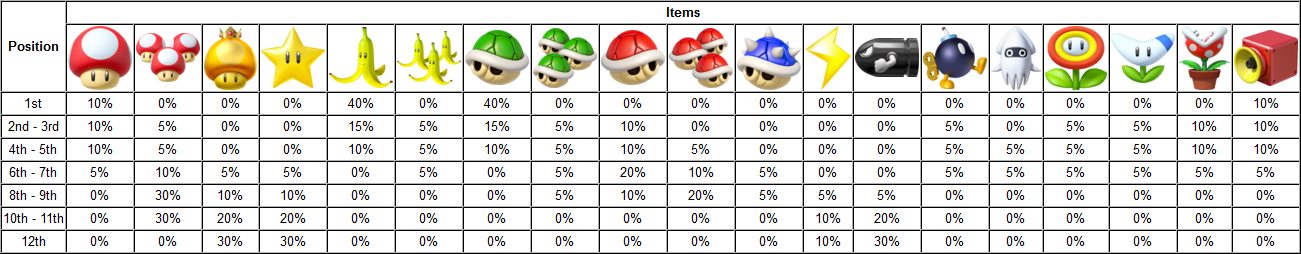


Figura 3.2: Tabla de probabilidad de los objetos en Mario Kart

Los objetos que pueden salir si se recoge una caja misteriosa, varían según la posición que esté el jugador. Como se ve en la figura, hay algunos objetos, los que más ventajas te dan, solo aparecen si estas en las últimas posiciones. En cambio, los objetos más defensivos aparecen cuando estas en las primeras posiciones.

#### MODOS DE JUEGO

Mario Kart tiene cuatro modos de juego.



Figura 3.3: Modos de juego de Mario Kart.

El primero, Gran Prix, se puede jugar tanto contra la máquina o contra otros jugadores y se caracteriza por disputar una copa que está compuesta de cuatro carreras normalmente de tres vueltas. Este modo de juego tiene cinco niveles de dificultad, 50cc, 100cc 150cc, espejo y 200cc. Cada dificultad es más velocidad para el vehículo del jugador, a excepción del modo espejo que tiene una velocidad de 150cc, pero con los mapas invertidos.

El segundo es más para practicar los circuitos, tienes normalmente tres vueltas en un circuito que el jugador elija, con tres turbos que puedes usar cuando quieras de la vuelta contrarreloj, sin la posibilidad de recoger objetos. La comunidad usa este modo de juego para establecer récords mundiales en los circuitos, además el jugador puede elegir si correr solo o contra un fantasma. La contrarreloj solo tiene modo un jugador.

El tercer modo de juego es bastante parecido al primero, pero el jugador puede elegir las reglas de la competición. El jugador elige la dificultad, si quiere que haya dos equipos, la dificultad de la inteligencia artificial, los tipos de vehículo de la máquina y los circuitos que aparecen, el jugador puede poner todos los circuitos del juego. En este modo de juego también hay la posibilidad de jugar contra otros jugadores.

El cuarto y último, también es un modo de juego multijugador. En este debes competir individualmente o por equipos a una serie de competición, por ejemplo, coger el máximo número de monedas, evitar que te golpeen, entre otros. Aquí los mapas que hay son originales para este modo en específico.

### TEAM SONIC RACING

En cuanto a Team Sonic Racing (Sumo Digital, 2019), es un juego de carreras arcade, muy parecido a Mario Kart 8, pero con los personajes del universo Sonic.

Seguidamente se detallará la configuración de los controles, sus mecánicas y para acabar sus modos de juego que hay en Team Sonic Racing.

#### CONTROLES



Figura 3.4: Controles de Team Sonic Racing

Teniendo como referencia el mando de la Switch, Team Sonic Racing tiene los siguientes controles:

* R: Acelerar
* L: Frenar/Derrapar
* B: Lanzar objeto
* A: Ceder Objetos
* X: Megaturbo
* Joystick Izquierdo: Girar

#### MECÁNICAS

En las mecánicas son bastantes parecidas a las de Mario Kart, aunque tiene alguna única.

La de acelerar, es básicamente lo mismo que tienen los otros dos juegos. Augmenta su velocidad hasta llegar a un máximo. Si dejas de pulsar ese botón el vehículo se para completamente por la fricción del suelo.

El cambio en este juego viene con el megaturbo, como se explicará en el apartado siguiente, la mayoría de las carreras en Team Sonic Racing son por equipos de tres, por lo que, si juegas en el modo de un jugador tienes dos IA que van en tu equipo. Cada vez que un miembro del

#### MODOS DE JUEGO

Team Sonic Racing tiene cuatro modos de juego.

El primero es un modo aventura. Ese modo es el único para un solo jugador.

### CRASH TEAM RACING NITRO FUELED

Para acabar Crash Team Racing Nitro Fueled (Beenox, 2019), remake del mítico juego Crash Team Racing (Naughty Dog, 1999) de PlayStation 1.

#### CONTROLES

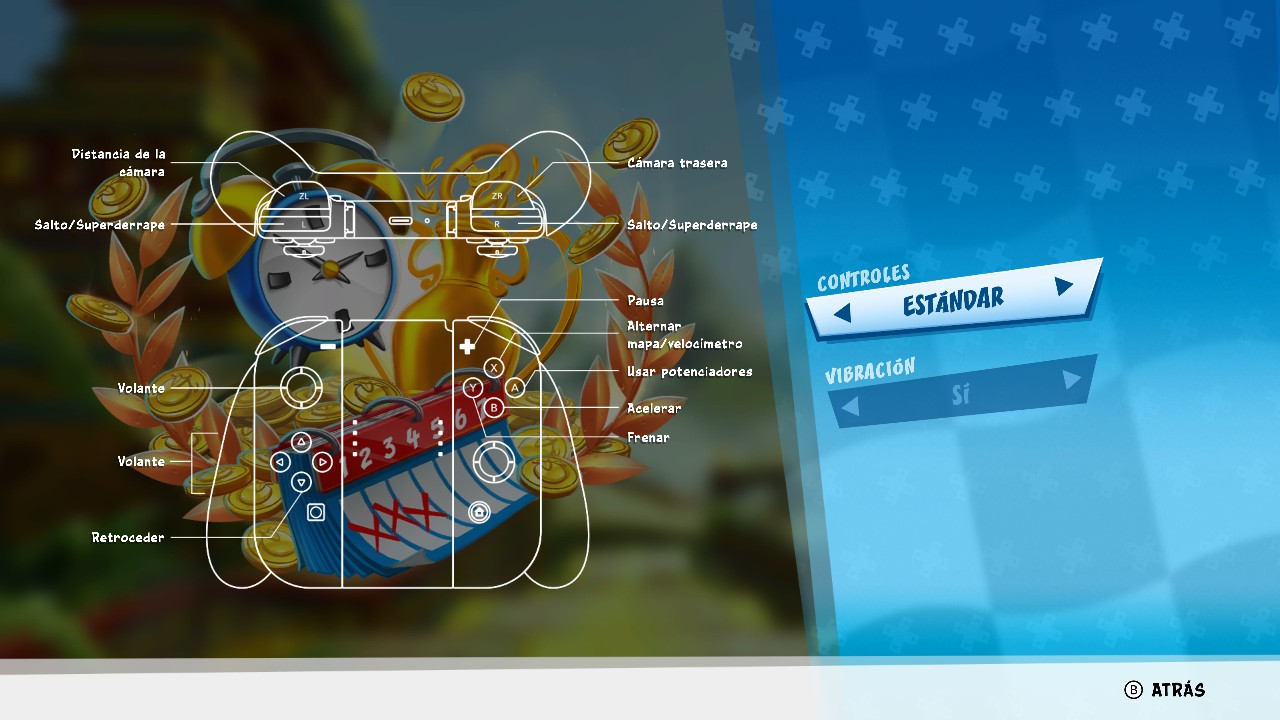


Figura 3.5: Controles Crash Team Racing Nitro Fueled

Teniendo como referencia el mando de la Switch, Crash Team Racing Nitro Fueled tiene los siguientes controles:

* B: Acelerar
* Y: Frenar
* L/R: Derrapar
* A: Lanzar objeto
* X: Habilidad de Equipo
* Joystick Izquierdo: Girar

#### MECÁNICAS

TODO

#### MODOS DE JUEGO

El juego



TODO

# MARCO TEÓRICO

## DEL MULTIJUGADOR LOCAL AL MULTIJUGADOR EN LÍNEA

### EVOLUCIÓN DE LOS MULTIJUGADORES

El multijugador en los videojuegos existe, prácticamente desde su creación. Tennis for Two (William Higginbotham, 1958), que fue hecho en un osciloscopio, era un juego para dos jugadores, por ello multijugador.

El primer juego multijugador en línea surgió el año 1973, para PLATO (Universidad de Illinois, 1960), que fue una de las primeras máquinas con un sistema de internet. Ese juego se llama Empire (Daleske, 1973), un juego de estrategia por turnos. Aunque prácticamente surgió a la par que Maze War (Jim Bowery, 1974) un juego de disparos en primera persona en línea.

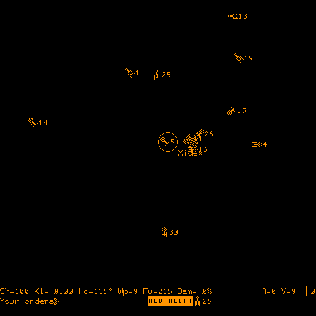


Figura 4.1: Imagen del juego de Empire.

Pero no fue hasta que salió el juego en LAN que hubo un incremento de demanda de este género de juegos. El juego en LAN, según (Glazer & Sanjay Madhav, 2016) se caracteriza ser un multijugador en red, pero requiere de estar en la misma red de internet.

El primero en usar de esa tecnología fue Doom (ID Software, 1993).

TODO

### MULTIJUGADOR LOCAL

Según (Glazer & Sanjay Madhav, 2016) un juego con multijugador local es aquel que está diseñado para jugar diferentes personas en el mismo ordenador. Un ejemplo de ellos son la saga Mario Kart (Nintendo, 1992) (antes del 8 que introduce el juego en línea), el Final Fantasy Chrystal Chronicles (Square Enix, 2004), entre muchos otros. Estos juegos implementaban un sistema que permitía a diversos jugadores jugar contra ellos, Mario Kart, o en cooperativo siguiendo una historia, Final Fantasy Chrystal Chronicles.

Este modo tiene alguna ventaja respecto al multijugador en línea, por ejemplo, no hay lag provocado por la conexión a internet.

Por el contrario, para poder disfrutar de las ventajas del multijugador local, necesitas que se esté jugando en la misma consola, por consiguiente, esos jugadores deben estar en el mismo espacio físico.

### MULTIJUGADOR EN LÍNEA

Según (Glazer & Sanjay Madhav, 2016) el multijugador en línea es aquel que requiere de, como mínimo, dos dispositivos conectados el uno del otro en la misma sesión de juego. Un ejemplo pueden ser la saga Call of Duty (Activision, 2004), el Minecraft (Mojang, 2011), entre muchos otros.

Mientras que, en uno, Call of Duty, su modalidad principal es jugar en competitivo contra otros jugadores. El otro se caracteriza por la exploración y la cooperación en un mundo que puedes hacer lo que se te pase por la cabeza.

El multijugador en línea, por lo que respecta al multijugador local, puedes jugar con gente de todo el mundo, conocidos o desconocidos.

El principal problema que hay es que requiere de conexión a internet, aunque actualmente casi toda la población mundial tiene de ese servicio. En la era moderna de las consolas, el multijugador online requiere de una subscripción de pago.

Además, el multijugador al depender de la conexión a internet puede provocar que haya lag en los juegos. Por lo que, como se explicará en más detalle en la siguiente sección, se pueden perder paquetes que se envían a internet y hacer que la experiencia del juego sea mala. Por lo que los desarrolladores tendrán que innovar para hacer que este problema no cause muchas malas experiencias para el jugador.

## PROTOCOLOS PARA LA COMUNICACIÓN EN TIEMPO REAL

Casi toda la población mundial tiene acceso a internet y para hacer un videojuego en línea es crucial disponer de esa tecnología. Pero para entender cómo funcionan los videojuegos multijugador en línea se debe hacer la siguiente pregunta. ¿Cómo funciona internet?

### ¿CÓMO FUNCIONA INTERNET?

Después de leer un artículo de Rus Shuler (Shuler, 2002) que te explica el funcionamiento de internet se puede deducir que internet, como bien dice su nombre es una red interconectada de dispositivos y encaminadores (*routers*).

Su funcionamiento, aunque tiene algunas complejidades, puede ser fácil de entender. Internet se debe ver como si se tratase un servicio de mensajería. Por ejemplo, un usuario de internet se quiere acceder a una página web. Para poder tener esta información, nuestro navegador enviará un paquete al servidor que esté alojado esa web. En ese paquete habrá toda la información requerida para llegar a ese servidor. Esa información esta organizada por un protocolo llamadas modelo de capas OSI.

|  |  |
| --- | --- |
| Capa 7 | Aplicación |
| Capa 6 | Presentación |
| Capa 5 | Sesión |
| Capa 4 | Transporte |
| Capa 3 | Red |
| Capa 2 | Enlace |
| Capa 1 | Física |

Tabla 4.1: Modelo de capas OSI (ISO/IEC JTC 1, 1994)

### TRANSPORT CONTROL PROTOCOL (TCP)

TODO

### USER DATAGRAM PROTOCOL (UDP)

TODO

### WEBSOCKETS

## METODOLOGÍAS PARA EL DESARROLLO DE VIDEOJUEGOS EN LÍNEA

En el desarrollo de videojuegos hay tres metodologías para encarar el multijugador. No hay una que sea mejor que la anterior, pero hay alguna metodología que es más recomendable que otra, dependiendo del género del juego que se haga.

### CLIENTE-SERVIDOR

Cliente-Servidor es la metodología más usado en la actualidad. Aunque se ha usado en la mayoría de los multijugadores masivo online. Se caracteriza por tener un servidor, ubicado normalmente por regiones, y en él se conectan todos los jugadores de la partida, como se puede ver en la topología siguiente.

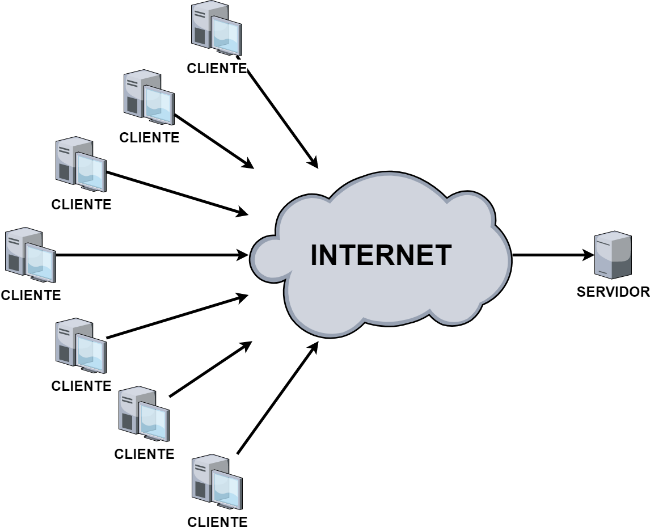


Figura 4.2: Topología Cliente-Servidor

El funcionamiento del cliente-servidor es sencillo, tanto el cliente como el servidor ejecutan una instancia del juego, aunque el servidor puede ser o no autoritativo, más a delante se vera la diferencia, la instancia del servidor suele ser la que manda.

#### PROBLEMAS DEL CLIENTE-SERVIDOR

El problema principal del cliente-servidor es mucho mas costoso, que el peer-to-peer. Ya que, como se verá en el siguiente subapartado, el p2p tiene ningún servidor.

El segundo problema crucial es de rendimiento, como se puede leer

### PEER-TO-PEER (P2P)

TODO

### SERVIDOR-SERVIDOR

TODO

## FUNCIONAMIENTO DE UN COCHE

En el juego el elemento principal es la conducción del vehículo y aunque no será un juego de simulación realista tendrá elementos muy parecidos al vehículo en la vida real. En está sección no se explicará nada sobre la evolución de los coches, solo se centrará en explicar su funcionamiento se manera sencilla.

# DISEÑO METODOLÓGICO Y CRONOGRAMA

En este capítulo se verán toda la metodología y planificación inicial que se tiene del proyecto. Para empezar, se verán cuáles son las herramientas del desarrollo del producto que se hará. A continuación, se explicarán los hitos que tiene el proyecto, así como la metodología y nomenclatura que se usará para elaborar el mismo. Para acabar se verá la planificación que se tiene del TFG y cuáles serán las fechas claves para que este proyecto salga adelante.

## HERRAMIENTAS DE DESARROLLO

Este proyecto, como ya se ha comentado varias veces, dispone de dos partes. Cada una de las partes tendrá unas herramientas distintas. Así que a continuación se verán cuáles son y porqué se han elegido esas herramientas.

### HERRAMIENTAS DE DESARROLLO DEL CLIENTE

El cliente lo primero que se debe saber es que motor de juego se usará. *Unreal Engine 4* (Epic Games, 2014). La versión será la última que ha salido que es la 4.26 i el motivo principal es por motivaciones personales del autor de este TFG. Ya que la mejor manera de aprender C++ es entrar en el desarrollo de algún software con ese lenguaje. Así que el autor después de barajar la idea de usar *Unity Engine* (Unity Technologies, 2005). Se descartó usar *Unity Engine, ya que*, aunque se podría haber hecho el juego usando la tecnología de ECS y DOTS. El autor prefirió hacer un juego en C++, ya que suele ser el lenguaje estándar en las empresas triple A.

Por lo que se puede intuir arriba, la programación será completamente en C++ y solo se usará la programación *Blueprints* en casos en que hacerlo en C++ resulte muy dificultoso. Por ejemplo, el HUD estará hecho en *Blueprints*, ya que es bastante más sencillo que hacerlo en C++.

El IDE que se usará es una versión *preview* del IDE de la empresa *JetBrains* que se llama *Rider*. Es decir, *Rider* es un IDE que se usa para desarrollar elementos del *framework* .NET y además se puede usar para programar los *scripts* de *Unity Engine*. Hace realmente poco *Rider* sacó su versión de IDE para poder programar exclusivamente en C++ y actualmente se encuentra en modo *preview*. El motivo de usar *Rider* es más por comodidad del autor, ya que desde segundo de carrera ha ido usando IDE de esta empresa, *JetBrains*.

Figura 5.1: Logotipos de *Unreal Engine* y *JetBrains Rider* respectivamente.

### HERRAMIENTAS DE DESARROLLO DEL SERVIDOR

Para programar el servidor, se usará básicamente un entorno de trabajo, que es *Node.js*. *Node.js*, según su página web (Dahl, 2009), es un entorno de trabajo basado en JavaScript y usado para programar servidores. Éste entorno está basado en eventos asíncronos, así que para hacer un servidor para un juego se necesitará un plugin llamado *Socket.io*, que lo que ofrece es la gestión de eventos en tiempo real usando *websockets*. El motivo de usar este entorno es por su comunidad que tiene detrás y el soporte para aprendizaje que hay por la red.

Para acabar el IDE usado será también de la empresa JetBrains y en este caso es su IDE para JavaScript llamado WebStorm.



Figura 5.2: Logotipos de *JetBrains WebStorm* y *Node.js*.

## METODOLOGÍA

Se intentará usar una metodología ágil. Para ello en el proyecto de *Git*, se crearán unas brancas de *master*, *develop¸*y *documentation* como brancas por defecto. Después para cada tarea se le pondrá las siglas del hito a que corresponde, y a continuación el nombre de lo que se va a desarrollar. Por ejemplo, si se está haciendo el controlador del personaje y justamente se está haciendo el controlador del vehículo, el nombre de esa branca se llamaría *c\_vehicleController,*

Este trabajo debe separarse en dos partes principales. La primera es la elaboración de la documentación. Y la segunda es la creación y desarrollo del producto. Así que primero se explicará la metodología de la documentación.

### METODOLOGÍA DE LA DOCUMENTACIÓN

La documentación tiene previstas dos hitos principales, que son las entregas puestas antes de empezar este trabajo. Así que el primer hito es elaborar el anteproyecto. En él la metodología es simple, se trabajará dos horas diarias desde noviembre, mientras se solapa con el diseño y comienzo del desarrollo de la parte del cliente.

A continuación, está la entrega de la memoria final. Para poder explicar como se hará esta documentación, se debe saber que, una vez entregado el anteproyecto, la memoria final empezara su producción. Para ello se dedicarán dos horas semanales para el desarrollo de la misma, hasta las dos últimas semanas antes de la fecha límite del hito. Esas dos semanas, se supondrá que el producto ya estará acabado, y por lo tanto se dedicaran las veinte horas para acabar de pulir y dejar perfecto este documento.

### METODOLOGÍA DEL PRODUCTO

El producto tendrá de tres hitos. Se entiende que la fase de preproducción del prototipo estará con el diseño del juego y está incluido en el primer hito.

El primero tiene fecha límite en el 28 de febrero, y se deberá tener hecho la parte del cliente. Así que este hito corresponde a ocho sprints, de una semana cada uno. Además antes de poder hacer ese hito se debe haber hecho un pequeño GDD en donde se explique cuáles son las mecánicas y como funciona el juego.

El segundo hito, es de cinco sprints y corresponde con el servidor. La fecha límite es el 2 de abril y para ello se deberá tener hecho el servidor entero.

El tercer y último hito es de nueve sprints y en él se hará la integración entre el cliente y el servidor.

## PLANIFICACIÓN

|  | ANTEPROYECTO | DISEÑO | CLIENTE | SERVIDOR | MULTIJUGADOR | MEMÓRIA | PREPARACIÓN PRESENTACIÓN |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NOVIEMBREE |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| DICIEMBRE |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| ENERO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| FEBRERO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| MARZO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| ABRIL |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| MAYO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| JUNIO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| JULIO |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Tabla 5.1: Cronograma del TFG

Como se puede ver en el cronograma hay siete grupos de tareas que hacer en este trabajo.

# BIBLIOGRAFÍA

Activision. (2004). *Call of Duty*.

Beenox. (2019). *Crash Team Racing Nitro Fueled*.

Dahl, R. L. (2009). *Node.js*. https://nodejs.org/en/

Daleske, J. (1973). *Empire*.

Epic Games. (2014). *Unreal Engine 4*. https://www.unrealengine.com/en-US/

Glazer, J. & Sanjay Madhav. (2016). *Multiplayer Game Programming*. Addison-Wesley.

ID Software. (1993). *Doom*.

ISO/IEC JTC 1. (1994). 7. Detailed description of the resulting OSI architecture. En *ISO/IEC 7498-1:1994* (1st Edition, p. 59). https://www.iso.org/standard/20269.html

Jim Bowery. (1974). *Maze War*.

Mojang. (2011). *Minecraft*.

Naughty Dog. (1999). *Crash Team Racing*.

Nintendo. (1992). *Super Mario Kart*.

Nintendo. (2013). *Mario Kart 8*.

Shuler, R. (2002). *How Does the Internet Work?* https://web.stanford.edu/class/msande91si/www-spr04/readings/week1/InternetWhitepaper.htm

Square Enix. (2004). *Final Fantasy: Crystal Chronicles*.

Sumo Digital. (2019). *Team Sonic Racing*.

Unity Technologies. (2005). *Unity Engine*. https://unity.com/es

Universidad de Illinois. (1960). *PLATO*. https://distributedmuseum.illinois.edu/exhibit/plato/

William Higginbotham. (1958). *Table for Two*.