

GRADO EN DISEÑO Y PRODUCCIÓN DE VIDEOJUEGOS

DESARROLLO DE UN PROTOTIPO DE JUEGO EN RED

Memoria Intermedia

ERIK ESPUÑES JUBERÓ

TUTOR: JORDI ARNAL MONTOYA

CURSO 2020-2021



ABSTRACT

This project will focus on creating a prototype of an arcade racing videogame with online multiplayer.

The prototype will consist of two game modes. The first one will be a singleplayer game mode. And the second one will be multiplayer, that will be the focus of this project.

Additionally, a game server will be developed to make the multiplayer game mode work properly. This server will control and manage the cheaters.

RESUM

Aquest projecte se centrarà a crear un prototip de videojoc de conducció arcade amb multijugador en línia.

El prototip constarà de dos modes de joc. El primer serà un mode un jugador senzill. El segon serà el multijugador, que serà en el que se centrarà aquest treball.

A més, es farà un servidor perquè el multijugador pugui funcionar correctament i així poder gestionar i controlar els jugadors perquè no facin trampes.

RESUMEN

Este proyecto se centrará en crear un prototipo de videojuego de conducción arcade con multijugador en línea.

El prototipo constará de dos modos de juego. El primero será un modo un jugador sencillo. Y el segundo será el multijugador, que será el que se centrará este trabajo.

Además, se hará un servidor para que este segundo modo de juego pueda funcionar correctamente y así poder gestionar y controlar a los jugadores para que no hagan trampas.

ÍNDICE

[ÍNDICE DE FIGURAS VII](#_Toc65780508)

[ÍNDICE DE TABLAS IX](#_Toc65780509)

[GLOSARIO DE TÉRMINOS XI](#_Toc65780510)

[CAPÍTULO 1. INTRODUCCIÓN 1](#_Toc65780511)

[CAPÍTULO 2. ANÁLISIS DE REFERENTES 3](#_Toc65780512)

[2.1. REFERENTES DE JUEGOS MULTIJUGADOR EN LÍNEA 4](#_Toc65780513)

[2.1.1. ROCKET LEAGUE 4](#_Toc65780514)

[2.2. REFERENTES DE JUEGOS DE CARRERAS ARCADE 4](#_Toc65780515)

[2.2.1. MARIO KART 8 4](#_Toc65780516)

[2.2.2. TEAM SONIC RACING 4](#_Toc65780517)

[2.2.3. CRASH TEAM RACING NITRO FUELED 4](#_Toc65780518)

[CAPÍTULO 3. MARCO TEÓRICO 4](#_Toc65780519)

[3.1. DEL MULTIJUGADOR LOCAL AL MULTIJUGADOR EN LÍNEA 4](#_Toc65780520)

[3.1.1. EVOLUCIÓN DE LOS MULTIJUGADORES 4](#_Toc65780521)

[3.1.2. MULTIJUGADOR LOCAL 4](#_Toc65780522)

[3.1.3. MULTIJUGADOR EN LÍNEA 4](#_Toc65780523)

[3.2. PROTOCOLOS PARA LA COMUNICACIÓN EN TIEMPO REAL 4](#_Toc65780524)

[3.2.1. ¿CÓMO FUNCIONA INTERNET? 4](#_Toc65780525)

[3.2.2. TRANSPORT CONTROL PROTOCOL (TCP) 4](#_Toc65780526)

[3.2.3. USER DATAGRAM PROTOCOL (UDP) 4](#_Toc65780527)

[3.3. METODOLOGÍAS PARA EL DESARROLLO DE VIDEOJUEGOS EN LÍNEA 4](#_Toc65780528)

[3.3.1. CLIENTE-SERVIDOR 4](#_Toc65780529)

[3.3.2. PEER-TO-PEER (P2P) 4](#_Toc65780530)

[3.3.3. SERVIDOR-SERVIDOR 4](#_Toc65780531)

[3.4. FUNCIONAMIENTO DE UN COCHE 4](#_Toc65780532)

[3.4.1. MOTOR 4](#_Toc65780533)

[3.4.2. SUSPENSIÓN 4](#_Toc65780534)

[3.4.3. DIRECCIÓN 4](#_Toc65780535)

[CAPÍTULO 4. OBJETIVOS 4](#_Toc65780536)

[4.1. OBJETIVOS PRINCIPALES 4](#_Toc65780537)

[4.2. OBJETIVOS SECUNDARIOS 4](#_Toc65780538)

[CAPÍTULO 5. DISEÑO METODOLÓGICO Y CRONOGRAMA 4](#_Toc65780539)

[5.1. HERRAMIENTAS DE DESARROLLO 4](#_Toc65780540)

[5.1.1. HERRAMIENTAS DE DESARROLLO DEL CLIENTE 4](#_Toc65780541)

[5.1.2. HERRAMIENTAS DE DESARROLLO DEL SERVIDOR 4](#_Toc65780542)

[5.2. METODOLOGÍA 4](#_Toc65780543)

[5.2.1. METODOLOGÍA DE LA DOCUMENTACIÓN 4](#_Toc65780544)

[5.2.2. METODOLOGÍA DEL PRODUCTO 4](#_Toc65780545)

[5.3. PLANIFICACIÓN 4](#_Toc65780546)

[CAPÍTULO 6. DESARROLLO 4](#_Toc65780547)

[6.1. DISEÑO 4](#_Toc65780548)

[6.1.1. GDD 4](#_Toc65780549)

[6.1.2. UML 4](#_Toc65780550)

[6.1.3. BASE DE DATOS 4](#_Toc65780551)

[6.2. CLIENTE 4](#_Toc65780552)

[6.2.1. CONTROLADOR DEL VEHÍCULO 4](#_Toc65780553)

[6.2.2. REGLAS DEL JUEGO 4](#_Toc65780554)

[6.2.3. CONTROLADOR DEL VEHÍCULO 4](#_Toc65780555)

[6.2.4. INTERFAZ GRÁFICA DE USUARIO 4](#_Toc65780556)

[6.2.5. OBJETOS 4](#_Toc65780557)

[6.2.6. INTELIGENCIA ARTIFICIAL 4](#_Toc65780558)

[6.3. SERVIDOR 4](#_Toc65780559)

[6.3.1. BASE DE DATOS 4](#_Toc65780560)

[6.3.2. INSTANCIA DEL JUEGO 4](#_Toc65780561)

[6.3.3. API 4](#_Toc65780562)

[6.4. MODO MULTIJUGADOR 4](#_Toc65780563)

[6.4.1. COMUNICARSE CON EL SERVIDOR 4](#_Toc65780564)

[6.4.2. MATCHMAKING 4](#_Toc65780565)

[6.4.3. MOVIMIENTO DE LOS JUGADORES 4](#_Toc65780566)

[6.4.4. LANZAR OBJETOS 4](#_Toc65780567)

[6.4.5. PREDICCIÓN, RECONCILIACIÓN E INTERPOLACIÓN 4](#_Toc65780568)

[6.4.6. BOTS 4](#_Toc65780569)

[CAPÍTULO 7. BIBLIOGRAFÍA 4](#_Toc65780570)

# ÍNDICE DE FIGURAS

[Figura 3.1: Información que proporciona Rocket League de la connexión. 6](#_Toc64057088)

[Figura 3.2: Controles de Mario Kart. 7](#_Toc64057089)

[Figura 3.3: Tabla de probabilidad de los objetos en Mario Kart 8](#_Toc64057090)

[Figura 3.4: Modos de juego de Mario Kart. 9](#_Toc64057091)

[Figura 3.5: Controles de Team Sonic Racing 10](#_Toc64057092)

[Figura 3.6: Algunos wisp que hay en Team Sonic Racing. 11](#_Toc64057093)

[Figura 3.7: Modos de juego de Team Sonic Racing. 12](#_Toc64057094)

[Figura 3.8: Controles Crash Team Racing Nitro Fueled 13](#_Toc64057095)

[Figura 3.9: Todos los objetos de Crash Team Racing Nitro Fueled. 14](#_Toc64057096)

[Figura 3.10: Modos de juego de Crash Team Racing Nitro Fueled. 15](#_Toc64057097)

[Figura 4.1: Imagen del juego de Empire. 17](#_Toc64057098)

[Figura 4.2: Imagen del modo cooperativo de Final Fantasy Chrystal Chronicles. 19](#_Toc64057099)

[Figura 4.3: Imagen del modo competitivo de Call of Duty: Black Ops - Cold War. 20](#_Toc64057100)

[Figura 4.4: Secuencia de los paquetes en el protocolo TCP. 25](#_Toc64057101)

[Figura 4.5: Secuencia de los paquetes en el protocolo UDP. 27](#_Toc64057102)

[Figura 4.6: Topología cliente-servidor 28](#_Toc64057103)

[Figura 4.7: Secuencia de los paquetes en la topologia cliente-servidor. 29](#_Toc64057104)

[Figura 4.8: Topología p2p 30](#_Toc64057105)

[Figura 4.9: Ciclo del motor. 32](#_Toc64057106)

[Figura 4.10: Sistema que hace que gire el vehículo. 33](#_Toc64057107)

[Figura 5.1: Logotipos de Unreal Engine y JetBrains Rider. 36](#_Toc64057108)

[Figura 5.2: Logotipos de JetBrains WebStorm y Node.js. 36](#_Toc64057109)

# ÍNDICE DE TABLAS

[Tabla 4.1: Modelo de capas OSI (ISO/IEC JTC 1, 1994) 21](#_Toc64057110)

[Tabla 4.2: Tabla de una llamada DNS realizada con Packet Tracer. 23](#_Toc64057111)

[Tabla 4.3: Tabla de una respuesta DNS realizada con Packet Tracer. 24](#_Toc64057112)

[Tabla 4.4: Tabla de una llamada HTTP realizada con Packet Tracer. 24](#_Toc64057113)

[Tabla 4.5: Tabla de una respuesta HTTP realizada con Packet Tracer. 24](#_Toc64057114)

[Tabla 5.1: Cronograma de la documentación 40](#_Toc64057115)

[Tabla 5.3: Cronograma del producto 45](#_Toc64057116)

# GLOSARIO DE TÉRMINOS

|  |  |
| --- | --- |
| BOTS | Enemigos controlados por una IA en un juego en red. |
| DOTS | Data Oriented Technology Stack |
| ECS | Entity Component System |
| E-SPORTS | Deportes electrónicos. |
| HTML | HyperText Markup Language |
| HUD | Head-Up Display |
| IA | Inteligencia Artificial |
| IDE | Integrated Development Environment |
| LAG | Retardo producido por la comunicación en tiempo real. |
| LAN | Local Area Network |
| Packet Tracer | És un software de simulación de redes de la empresa CISCO. |
| RFC | Request for Comments, són los documentos oficiales de redes y deben estar aprobados por la Internet Engineering Task Force (IETF). |
| TFG | Trabajo de Fin de Grado |

# INTRODUCCIÓN

Desde hace años los juegos multijugador están en el orden del día. Cada vez se ha ido jugando más de esa forma hasta el punto de llegar a hacer torneos online, como son los e-sports.

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 autoritario, 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.

# 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

### ROCKET LEAGUE

Rocket League (Psyonix, 2015) es un juego de fútbol arcade en el que llevas coches. Los partidos están compuestos de dos equipos, entre uno y cuatro jugadores, y el objetivo es anotar más goles que el rival para ganar. Éstos tienen una duración de 5 minutos, si uno de los dos equipos gana. En caso de empate, se va a una prórroga y esta no se acaba hasta que uno de los dos marca un gol.

Analizándolo en su modo multijugador, Rocket League es un juego cliente-servidor. Y eso se puede deducir muy bien cuando se entra en la partida, sale un texto que indica que te estás conectando a una región y eso es un claro indicador que se trata de una metodología cliente-servidor.

Además, Rocket League hace una cosa muy interesante, que es informarte de los errores o anomalías que puedes tener mientras se está jugando un partido. Como se puede ver en la imagen siguiente.



Figura .: Información que proporciona Rocket League de la connexión.

Para este proyecto Rocket League es el gran referente de multijugador, ya que más o menos se trata de algo parecido al producto a desarrollar. Es decir aunque sean géneros diferentes, los dos usan vehículos para moverse y además de la información que da al usuario en torno a los problemas de conectividad es muy bueno.

## REFERENTES DE JUEGOS DE CARRERAS ARCADE

En este sub apartado 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 .: 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, este llegará a frenar el coche. Pero si se sigue frenando, una vez el vehículo llega a velocidad cero, este 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 aumentando en función de la duración del derrape.

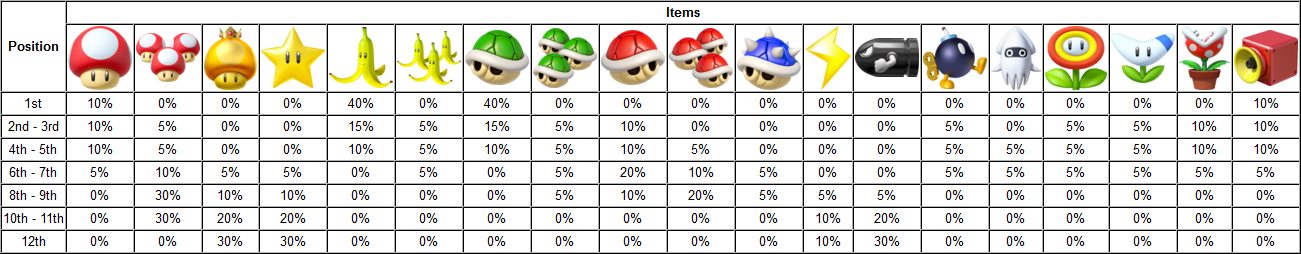


Figura .: Tabla de probabilidad de los objetos en Mario Kart

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. El objeto más curioso es el cohete, ya que te lleva hasta cierta posición que el juego cree.

#### MODOS DE JUEGO

Mario Kart tiene cuatro modos de juego.



Figura .: 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 .: 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, frenar, girar y derrapar no se ha notado ninguna diferencia.

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 equipo hace algún derrape, o te da algún objeto, ya que los objetos se pueden dar a un compañero de equipo, la barra del megaturbo sube. El megaturbo como bien dice el nombre, da un turbo a todos los integrantes del equipo bastante más largo que los objetos.

En el tema de los objetos, también es bastante parecido a Mario Kart. Aunque en este juego se llaman wisps, aparecen dependiendo de la posición que esté el personaje. El más curioso son el wisp de temblor, que hace aparecer pilares de piedra en el camino



Figura .: Algunos wisp que hay en Team Sonic Racing.

#### MODOS DE JUEGO

Team Sonic Racing tiene cuatro modos de juego de un jugador.

El primero es un modo aventura. Es el modo principal del juego En él se debe hacer una serie de misiones, que son carreras o contrarreloj.



Figura .: Modos de juego de Team Sonic Racing.

Después tiene los típicos modos de un jugador, el gran premio, que es jugar una serie de cuatro carreras en equipo, dos IAs, o individual. Además, se puede elegir la dificultad que son tres, fácil, normal y difícil. Al acabar este gran premio, en el caso de ganar, te dan la copa.

El siguiente modo de juego es una carrera normal. Eliges circuito, dificultad y si quieres ir por equipos y juegas una sola carrera.

El último es el modo contrarreloj, que es dar vueltas al circuito para mejorar tu tiempo o practicar.

### 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

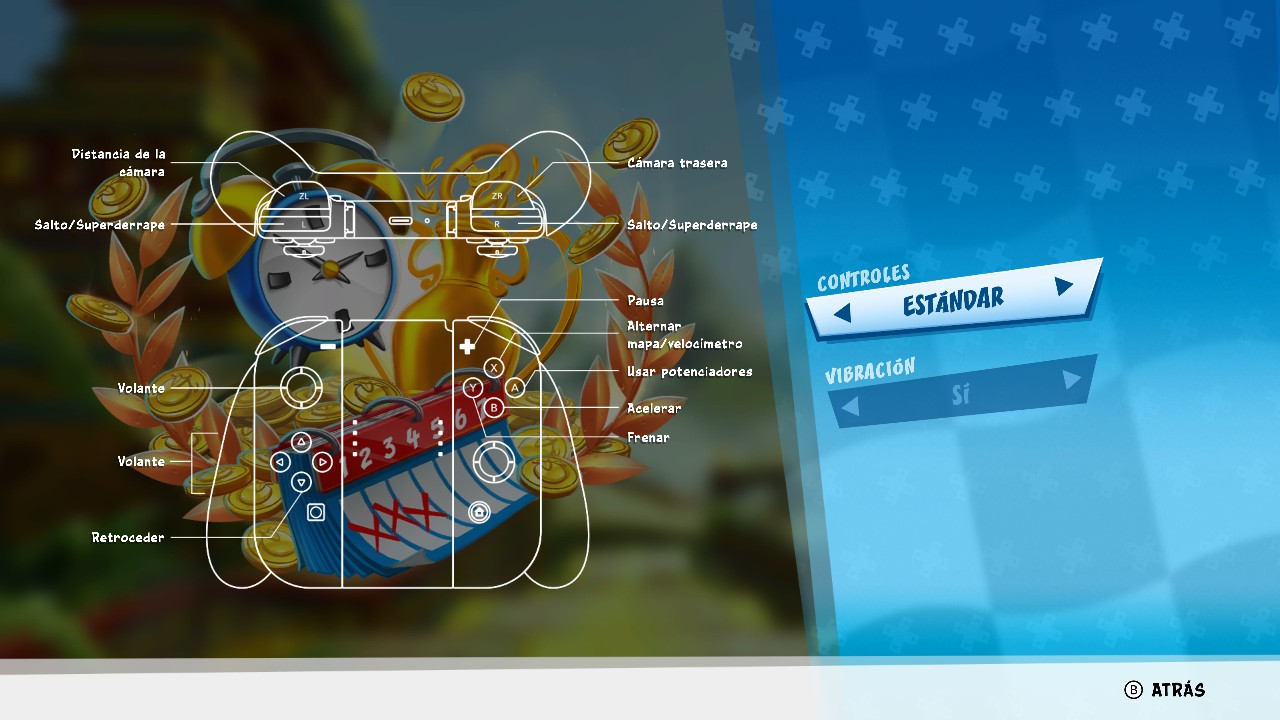


Figura .: Controles Crash Team Racing Nitro Fueled

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

Las mecánicas del movimiento del vehículo son bastantes parecidas a los dos juegos anteriores. Pero el derrape es lo que marca la diferencia. Ya que requiere de habilidad para el jugador. En los otros dos pulsas el botón y ya derrapa solo y te dan el turbo. En Crash una vez se está derrapando se debe pulsar el botón de derrapar, otra vez, en el momento preciso para tener el turbo.



Figura .: Todos los objetos de Crash Team Racing Nitro Fueled.

En cuanto a los objetos, hay algunos que requieren de habilidad del jugador. Tanto en Mario Kart como en Team Sonic Racing, los objetos se lanzan y ya no se debe hacer nada. En Crash hay objetos, como la caja de TNT, que para que no le explote al jugador debe pulsar el botón de lanzar objeto repetidamente para librarse.

En cuanto a la aparición, es bastante aleatorio. Es decir, hay algunos objetos que no saldrán estando en alguna posición específica, pero no parece que haya una probabilidad para que te salgan los objetos.

#### MODOS DE JUEGO

El juego tiene siete modos de juego de un jugador.

Su modo de juego principal es la historia. En este modo de juego hay como un mundo abierto para ir a las misiones o carreras.



Figura .: Modos de juego de Crash Team Racing Nitro Fueled.

Después hay los modos típicos que salen en los otros juegos, carrera, torneo y contrarreloj.

Además, Crash Team Racing Nitro Fueled tiene un modo batalla que en él se debe competir individualmente en diferentes batallas como, capturar la bandera, battle royale, etc. Este modo tiene mapas exclusivos.

La carrera de anillos se debe seguir el camino formado por los anillos que hay en el mapa para ganar puntuación y alargar el tiempo, que si este último llega a cero se acaba la carrera.

La carrera por la reliquia es una contrarreloj, pero mucho más avanzado, es decir, para ganar se debe hacer una vuelta en un tiempo límite, que es bastante difícil.

Para acabar Crash Team Racing Nitro Fueled tiene modos de desafío, que son carreras, pero que para ganar se deben conseguir ciertos objetivos, el CTR que se deben recoger las letras C, T y R que están repartidos en el mapa. El desafío de cristales, en cambio, es en los mapas de batallas y se deben conseguir veinte cristales en un tiempo límite.

# MARCO TEÓRICO

## DEL MULTIJUGADOR LOCAL AL MULTIJUGADOR EN LÍNEA

Desde el inicio de los videojuegos, el multijugador está presente. Así que en este apartado se verá la evolución que ha tenido los videojuegos multijugador y además se verán los dos métodos más usados y se explicará que ventajas e inconvenientes tienen.

### 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 y Warner, 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.

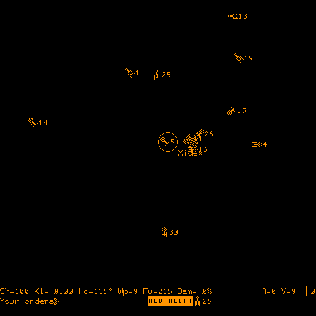


Figura .: Imagen del juego de Empire.

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 y 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). En él se podía jugar tanto cooperativo como competitivo hasta cuatro jugadores. Con este sistema muchos juegos se prepararon para la era del multijugador en línea.

Con el auge de internet en la década de los 90, los videojuegos empezaron a implementar los multijugadores en línea. Uno de los primeros en tener un multijugador en línea fue Epic Game’s Unreal (Epic MegaGames y Digital Extremes, 1998).

Sobre la década de los 2000, las consolas empezaron a implementar el multijugador en línea con la aparición de Xbox Live, PlayStation Network, entre otros.

### MULTIJUGADOR LOCAL

Según (Glazer y 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), los que salieron antes del 8, ya que el último 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.



Figura .: Imagen del modo cooperativo de 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 o muy cerca geográficamente.

### MULTIJUGADOR EN LÍNEA

Según (Glazer y 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.



Figura .: Imagen del modo competitivo de Call of Duty: Black Ops - Cold War.

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. Como bien dice su nombre es una red interconectada de dispositivos y encaminadores (routers). Por lo que todo el mundo esta conectado con cables, routers y dispositivos[[1]](#footnote-1).

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 está organizada por un protocolo llamado 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 .: Modelo de capas OSI (ISO/IEC JTC 1, 1994)

De arriba abajo, la capa siete serian las aplicaciones, del dispositivo, que hacen uso de la red.

La capa seis, la de presentación, es la encargada de hablar en el mismo idioma, por así decirlo. Por ejemplo, si envías una imagen en formato JPG, esta capa es la encargada de decir que esa imagen está en ese formato.

La capa de sesión es la encargada de mantener la sesión activa entre diferentes dispositivos. Se puede usar para realizar llamadas de voz o video a través de internet.

La capa cuatro, es la que se debe encargar de transmitir los datos. Sus protocolos estrella son UDP y TCP.

La capa de red trabaja un único protocolo, llamado Internet Protocol o IP. Si llega un paquete en que la dirección IP no es la misma que la del dispositivo, esta capa descartará el paquete, ya que sabrá que no es para él. La dirección IP está compuesta de números decimales. Esta puede variar dependiendo del tiempo, aunque siempre estará asociada a la red que esté conectado. Un ejemplo de dirección IP es la siguiente 192.168.1.123.

La capa dos, es la encargada de tener la dirección física del dispositivo, MAC, y al igual que la capa de red, ningún paquete pasará a la siguiente capa si la MAC no es la misma que la del dispositivo. La MAC es única en cada ordenador, por lo que no se pueden repetir y es un número hexadecimal. Un ejemplo de una dirección MAC sería esta 00:25:08:55:2B:AB.

Para acabar, la capa uno, es la encargada de convertir toda la trama o paquete generado en señales eléctricas para poderla transmitir por la corriente.

Las capas OSI, cuando se debe generar una trama, siempre empieza en la capa siete hasta abajo. En el caso de leer la trama, va al revés. Primero empieza por la capa uno, hasta llegar a la siete, si llega.

Ahora sí, se va a explicar cómo funcionaria una llamada a una página web.

Primero de todo se establecería la llamada HTTP (HyperText Transfer Protocol) desde la capa 7. Primero miraría si el dispositivo conoce la dirección IP donde se encuentra esa página web. En caso de no conocerlo enviaría un paquete del servicio DNS (Domain Name Service). El DNS es un servicio que contiene una lista de dominios web y sus respectivas IP. Por ejemplo, en esa lista estaría guardado algo parecido a así “google.com = 8.8.8.8”. Por lo que este servicio le envías un dominio web y te devuelve su IP. El paquete que enviaría nuestro dispositivo es parecido al siguiente[[2]](#footnote-2).

|  |  |
| --- | --- |
| Capa 7 | DNS – “google.com” |
| Capa 6 | - |
| Capa 5 | - |
| Capa 4 | UDP |
| Capa 3 | IP ORIG: 10.0.1.66  IP DEST: 172.16.0.3 |
| Capa 2 | MAC ORIG  MAC DEST |
| Capa 1 | - |

Tabla .: Tabla de una llamada DNS realizada con Packet Tracer.

Como se ve en la capa siete, marca que se trata de un servicio DNS y le envía el nombre del dominio. Como no le hace falta ningún protocolo de capa seis o cinco, los omite. En la capa cuatro establece un transporte tipo UDP (más adelante se verá que es). En la capa tres y dos establece una IP origen, IP destino y MAC origen, MAC destino, respectivamente. Este paquete pasaría todos los dispositivos de redes que haya hasta llegar a su destinatario este situado en cualquier parte del mundo, en el caso de que pueda llegar.

|  |  |
| --- | --- |
| Capa 7 | DNS – “8.8.8.8” |
| Capa 6 | - |
| Capa 5 | - |
| Capa 4 | UDP |
| Capa 3 | IP ORIG: 172.16.0.3  IP DEST: 10.0.1.66 |
| Capa 2 | MAC ORIG  MAC DEST |
| Capa 1 | - |

Tabla .: Tabla de una respuesta DNS realizada con Packet Tracer.

El paquete que le enviaría el servidor DNS al dispositivo sería igual, pero en la capa siete en vez de enviar el nombre del dominio llevaría la IP de la página web.

|  |  |
| --- | --- |
| Capa 7 | HTTP — GET (http://google.com) |
| Capa 6 | - |
| Capa 5 | - |
| Capa 4 | TCP |
| Capa 3 | IP ORIG: 10.0.1.66  IP DEST: 8.8.8.8 |
| Capa 2 | MAC ORIG  MAC DEST |
| Capa 1 | - |

Tabla .: Tabla de una llamada HTTP realizada con Packet Tracer.

Como pasa con la llamada DNS, las capas seis, cinco y uno no tienen ninguna función especial. La diferencia que hay entre la llamada HTTP y DNS es que, para empezar en la capa siete cambia el protocolo que usa. En este caso, para visualizar una página web. Después salta la atención que, en la capa tres cambie de protocolo a TCP. Para acabar se ve que la IP destino, en la capa tres, es la del servidor web que se ha pedido antes.

|  |  |
| --- | --- |
| Capa 7 | HTTP - HTML |
| Capa 6 | - |
| Capa 5 | - |
| Capa 4 | TCP |
| Capa 3 | IP ORIG: 8.8.8.8  IP DEST: 10.0.1.66 |
| Capa 2 | MAC ORIG  MAC DEST |
| Capa 1 | - |

Tabla .: Tabla de una respuesta HTTP realizada con Packet Tracer.

Lo que devolvería la llamada HTTP es un archivo HTML, que sería lo que abriría el navegador web.

Toda la información sacada a través de los servicios HTTP o DNS corresponden a sus respectivos documentos oficiales, RFC 2616 (Fielding et al., 1999) y RFC 1035 (Mockapetris, 1987) respectivamente.

### TRANSPORT CONTROL PROTOCOL (TCP)

Según su RFC (Postel, 1981), TCP es un protocolo de capa cuatro. Su función es gestionar el envio o recibo de datos entre dos dispositivos. Solo uno puede enviar archivos y el otro debe recibirlos.

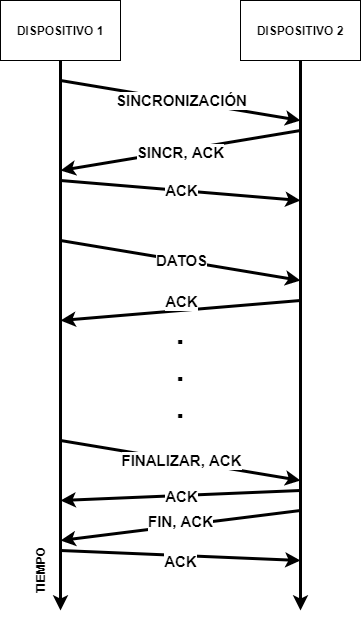


Figura .: Secuencia de los paquetes en el protocolo TCP.

TCP es un protocolo orientado a conexión, así que para poderse comunicar con otro dispositivo debe hacer un establecimiento. En TCP debe hacer un establecimiento a tres bandas. Eso es que el primer dispositivo que quiera hacer el establecimiento deberá enviarle un paquete informándole al otro que quiere enviar o recibir datos. A continuación el otro dispositivo, al recibir ese paquete, le comunicara que él también quiere hacer la conexión, además le enviara un comprobante de que ha llegado su paquete de establecimiento (ACK). Para acabar el primer dispositivo le enviará un nuevo ACK.

Además, en TCP siempre llegan los paquetes, es decir, ni un paquete de datos se pierde, se vuelve a enviar para que llegue al destinatario. Para saber si ha llegado o no, el destinatario debe enviar un paquete ACK. Si al cabo de un tiempo de enviar los datos, ese no recibe la confirmación de llegada, este reenviara los datos.

Para acabar TCP debe cerrar la conexión, para ello hace un cerramiento a cuatro bandas. Primero el dispositivo que quiera acabar la transmisión de datos deberá informal al otro enviándole un paquete de finalización. El otro dispositivo le enviara un ACK y seguidamente le enviara otro paquete informándole de que él también quiere acabar. Para acabar el primero enviará un nuevo ACK y se cerrará la transmisión de datos.

La ventaja de TCP es que siempre llegan los datos, así que es muy fiable.

La desventaja principal es que para enviar o recibir se necesita establecer una conexión, que a veces puede resultar un poco inútil, ya que puede suceder que los datos sean pequeños que todos los establecimientos hechos. Además, es bastante lento.

Este protocolo se usa cuando es necesario asegurar la fiabilidad. Como es el caso de hacer una petición HTTP.

### USER DATAGRAM PROTOCOL (UDP)

Según su RFC (Postel, 1980), UDP es un protocolo de capa cuatro. Por lo que debe poder gestionar el envío de los archivos.

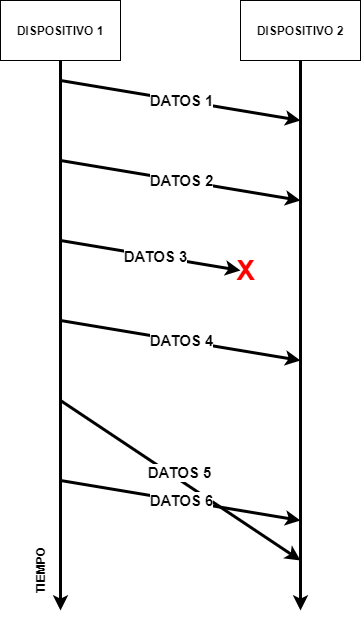


Figura .: Secuencia de los paquetes en el protocolo UDP.

La diferencia principal que se puede observar es que, UDP, no es orientado a conexión. Por lo que el dispositivo que quiera enviar archivos lo podrá hacer con total normalidad, sin que el destinatario le dé permiso. En UDP el destinatario no envía ningún ACK.

Por lo que se puede ver en la figura anterior, puede suceder que por el camino se pierda un paquete de datos, puede suceder que un paquete llegue más tarde que uno más reciente. Esto pasa porque no hay ningún concepto de confirmación de paquetes.

La ventaja de UDP es que es muy rápido, ya que no es necesario hacer ningún establecimiento previo.

La desventaja principal es no es nada fiable y se debe asumir que se pueden perder paquetes, que lleguen desordenados o duplicados.

Este protocolo se usa cuando es no necesario asegurar la fiabilidad. Como es el caso de hacer una petición DNS.

## 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. Toda la información que se encontrará a continuación ha sido extraída del libro Development and Deployment of Multiplayer Online Games (Ignatchenko, 2017)

### CLIENTE-SERVIDOR

Cliente-Servidor es la metodología más usada 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.

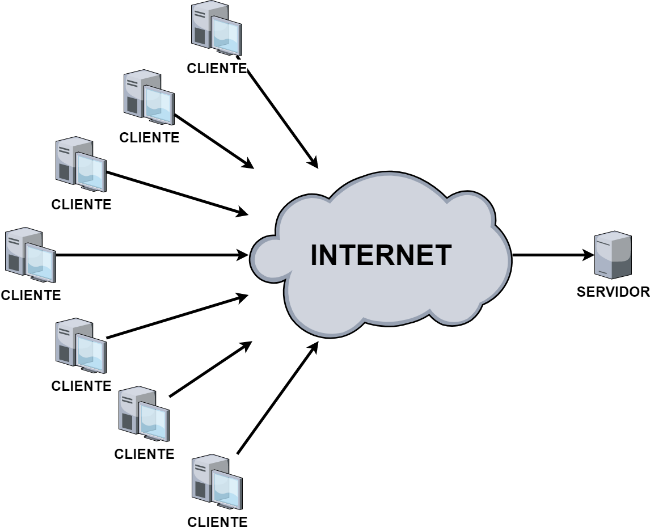


Figura .: Topología cliente-servidor

El funcionamiento del cliente-servidor es sencillo, tanto el cliente como el servidor ejecutan una instancia del juego, cada vez que el cliente hace alguna acción en el juego, esta se envía al servidor y notifica a los demás jugadores de ello.

#### PROBLEMAS DEL CLIENTE-SERVIDOR

El problema principal del cliente-servidor es mucho más costoso, que el peer-to-peer. Ya que, como se verá en el siguiente subapartado, el p2p tiene ningún servidor dedicado. Según (Ignatchenko, 2017), más o menos cada servidor, dependiendo del juego, puede gestionar las partidas entre mil o cinco mil jugadores. El precio de cada servidor puede suponer unos mil euros mensuales. También se puede optar por hacer que un cliente haga también de servidor, a cambio de hacer el servidor no autoritativo y por lo tanto más fácil de hacer trampas.

El segundo problema crucial es de rendimiento, ya que al hacerse que todas las acciones hechas por el jugador las gestione el servidor, hace que se deba enviar un paquete por internet a ese servidor. Por lo que tarda un tiempo en hacer esa acción. Por lo que se produce un input lag que es bastante grande, que ronda sobre los tres cientos milisegundos. Al ser un problema muy común hay diferentes soluciones.

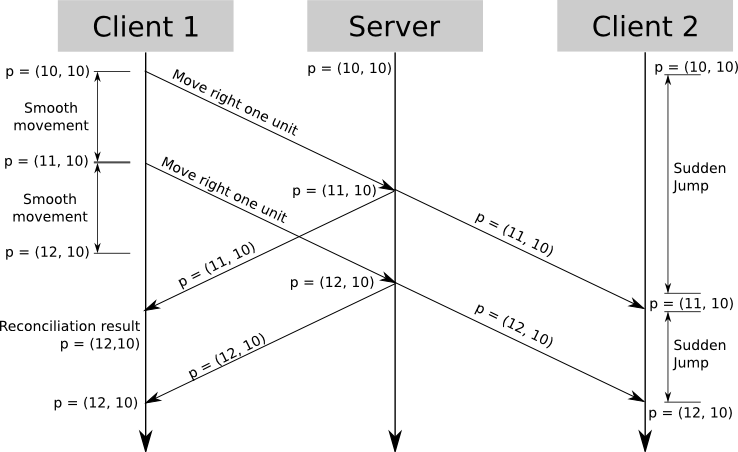


Figura .: Secuencia de los paquetes en la topologia cliente-servidor.

Como bien explica el señor Gambetta (Gambetta, s. f.), el cliente deber hacer una predicción de la acción, que viene a ser hacer una predicción del estado del juego en el servidor.

La segunda solución que debe hacer el cliente es reconciliarse con el servidor. Como se ve en la imagen anterior, cuando el jugador se ha movido dos unidades, el servidor le da el estado del servidor del primer movimiento. Así que el cliente debe analizar esos datos y ver si actualiza su estado del mundo o lo descarta.

La última solución es la interpolación de los demás jugadores. Esto se debe a que el movimiento hecho por el jugador tardara un tiempo en llegar a los demás jugadores. Así que el cliente debe interpolar las posiciones o acciones de los demás jugadores para que no haya saltos y se vea mal el juego.

### PEER-TO-PEER (P2P)

La otra gran metodología para los juegos multijugador en red es el peer-to-peer o p2p. Esta metodología se basa en que todos los clientes se comunican entre ellos sin la necesidad de un servidor de por medio.

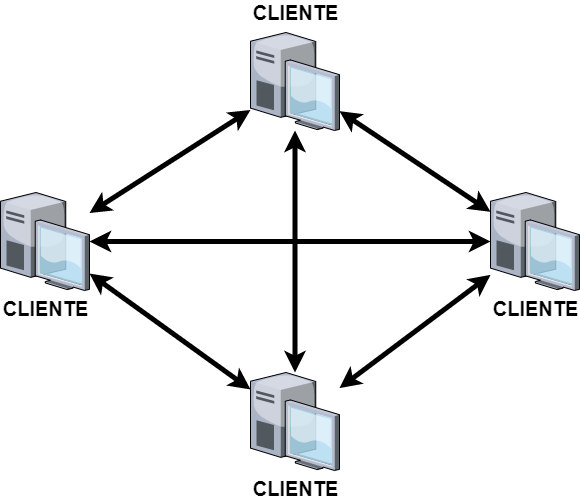


Figura .: Topología p2p

Como se ve en la figura anterior, cada cliente se comunica con el que tenga que hacerlo para enviarle los datos del juego. Así que es una metodología bastante barata, ya que no requiere de tener alojado un servidor.

Pero hace que nuestro juego no sea seguro y sea muy fácil de hacer trampas en él.

### SERVIDOR-SERVIDOR

La metodología de servidor-servidor se usa para los juegos orientados a reglas. Estos son los juegos de casino, juegos sociales, etc. Normalmente son los juegos que no requieren de arte 3D.

Como es una metodología que para el juego que se desarrollara no es útil, tampoco se desarrollará su explicación.

## FUNCIONAMIENTO DE UN COCHE

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

### MOTOR

El motor es uno de los ejes principales del coche, sin él el vehículo no se podría mover. El motor está compuesto de cilindros y pistones.

Lo que hace el motor es, mediante combustible, generar energía para que el vehículo pueda moverse. Para hacer esto el motor internamente debe hacer los siguientes movimientos.

Según (NSW Department of Education and Training, 2008), primero el pistón se comprime dentro del cilindro, mientras se comprime se inyecta el combustible dentro del cilindro. Cuando ya esté comprimido, se genera una explosión del combustible para que exista el movimiento, ya que al explotar el pistón baja, y entonces se vuelve a hacer el ciclo otra vez.

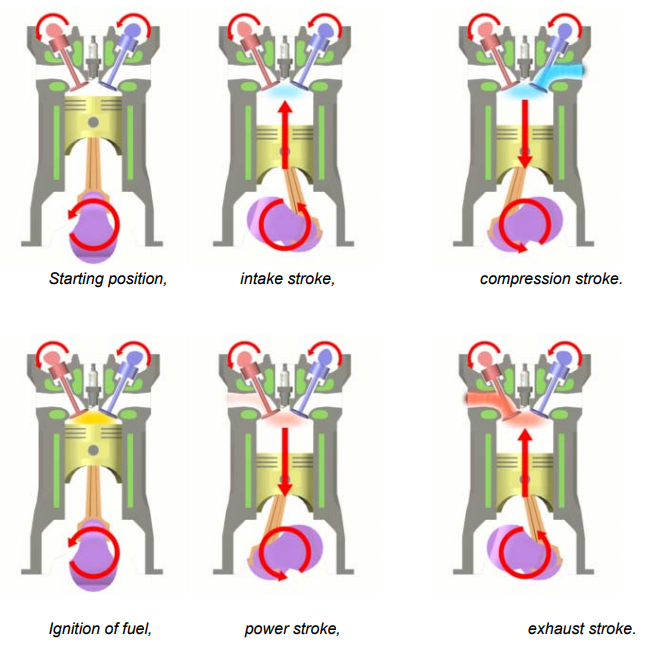


Figura .: Ciclo del motor.

### SUSPENSIÓN

Otro pilar fundamental es la suspensión.

Como bien dice Konieczny y Burdzik (Konieczny y Burdzik, 2017), sin ella el vehículo solo podría ir plano, es decir que cualquier bache en la carretera o desnivel se podrían hacer daño los ocupantes del vehiclo.

Para ello, la suspensión se basa en la ley física de Hook. Que su fórmula es la siguiente: , donde k es la constante elástica del muelle, x es la longitud del muelle aplicando la fuerza y x0 es la longitud del muelle sin aplicar la fuerza.

Por lo que cada rueda tiene un muelle y si hay algún desnivel, la suspensión balancea el vehículo para que no sea un problema para los ocupantes.

### DIRECCIÓN

La dirección del vehículo es lo que hace poder girarlo con el volante. Su sistema de giro es un poco complejo.

Según Schnubel (Schnubel, 2018) , Primero el conductor debe girar el volante para que el vehículo gire. Cuando el conductor ha hecho esa acción un sistema formado por un unas bolas de metal y un eje hace que gire un engranaje y este hace que las ruedas delanteras, traseras o ambas giren.

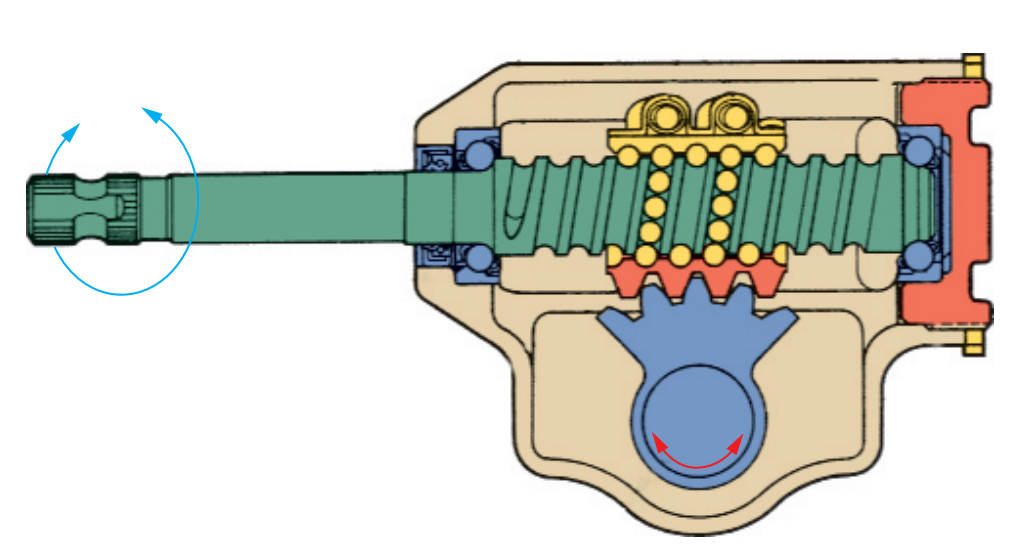


Figura .: Sistema que hace que gire el vehículo.

# 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

1. Hacer que el multijugador usando Unreal Engine y Node.js

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.

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

Como se intentará hacer un servidor autoritario, el objetivo principal es que nadie haga trampas. Así que se tendrá mano dura para los jugadores que hagan trampas de manera reiterada.

## OBJETIVOS SECUNDARIOS

1. 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.

1. Hacer que los objetos estén bien balanceados.

La gracia de los juegos de conducción arcade, aparte de la conducción, son los objetos que se pueden lanzar a los jugadores. Si estos aparecen en una posición incorrecta o el poder que tiene es desproporcionado, el jugador sentirá frustración y dejará de jugar a nuestro juego. Así que, dentro de lo posible, se intentará que estos estén bien balanceados.

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

Se intentará que, cuando un jugador quiera jugar el modo multijugador, el juego le permita. Pero si no hay más jugadores o no hay los suficientes para empezar la carrera se añadirán bots.

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

Cualquier decisión tomada que cambie drásticamente el producto o la planificación deberá ser documentada en el apartado de desarrollo.

# 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 por qué 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.

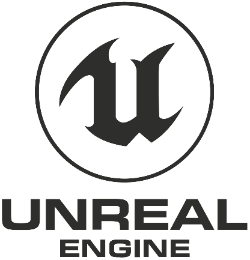
 

Figura .: Logotipos de Unreal Engine y JetBrains Rider.

### 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.

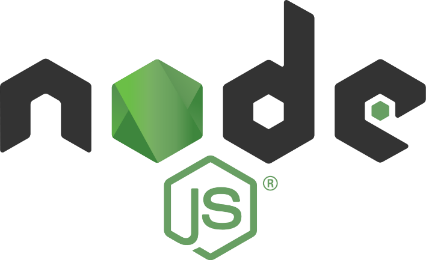


Figura .: Logotipos de JetBrains WebStorm y Node.js.

## METODOLOGÍA

Se intentará usar una metodología ágil. Para ello en el proyecto de Git, se crearán unas ramas 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 rama 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 cómo se hará esta documentación, se debe saber que, una vez entregado el anteproyecto, la memoria final empezará su producción. Para ello se dedicarán dos horas semanales para el desarrollo de esta, 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 cómo 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. Su fecha límite es el 4 de junio, y se debe tener hecho el multijugador del juego.

## PLANIFICACIÓN

Para mostrar la planificación se separará en dos partes principales. Primero se verá el cronograma de la documentación y después se verá el del producto.

Para que se pueda entender un poco el siguiente cronograma, las letras que se ven en la cabecera de la tabla corresponden a:

* Docs (Documentación)
  + A: Anteproyecto ≈ 90 h
  + M: Memoria ≈ 52 h
  + P: Presentación ≈ 35 h
* Diseño:
  + GDD: Game Design Document ≈ 10 h
  + UMLC: UML del Cliente ≈ 4 h
  + UMLS: UML del Servidor ≈ 6 h
  + ERD: Esquema relacional de la base de datos ≈ 4 h
* Cliente:
  + CV: Controlador del vehículo ≈ 15 h
  + RJ: Reglas del juego ≈ 15 h
  + GUI: Interfaz gráfica de usuario ≈ 15 h
  + IA: Inteligencia artificial ≈ 63 h
  + O: Objetos ≈ 30h
* Servidor:
  + BD: Base de datos ≈ 24 h
  + IJ: Instancia del juego ≈ 72 h
  + API: Crear la API ≈ 24 h
* Multijugador:
  + CAPI: Comunicarse con la API ≈ 24 h
  + MM: Matchmaking ≈ 10 h
  + MJ: Movimiento de los jugadores ≈ 38 h
  + LO: Lanzar objetos ≈ 24 h
  + SE: Solucionar errores ≈ 96 h
  + B: Bots ≈ 15 h

|  |  | **DOCS** | | | **DISEÑO** | | | | **CLIENTE** | | | | | **SERVIDOR** | | | **MULTIJUGADOR** | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | A | M | P | GDD | UMLC | UMLS | ERD | CV | RJ | GUI | IA | O | BD | IJ | API | CAPI | MM | MJ | LO | SE | B |
| **NOV** | **S1** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **DICIEMBRE** | **S2** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S3** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S5** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S6** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ENERO** | **S7** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S8** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S9** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S10** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **FEBRERO** | **S11** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S12** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S13** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S14** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MARZO** | **S15** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S16** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S17** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S18** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S19** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **ABRIL** | **S20** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S21** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S22** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S23** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **MAYO** | **S24** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S25** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S26** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S27** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **JUNIO** | **S28** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S29** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S30** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S31** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **S32** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **JUL** | **S33** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Tabla .: Cronograma del TFG

# DESARROLLO

## DISEÑO

### GDD

### UML

### BASE DE DATOS

## CLIENTE

A

### CONTROLADOR DEL VEHÍCULO

### REGLAS DEL JUEGO

### CONTROLADOR DEL VEHÍCULO

### INTERFAZ GRÁFICA DE USUARIO

### OBJETOS

### INTELIGENCIA ARTIFICIAL

## SERVIDOR

### BASE DE DATOS

### INSTANCIA DEL JUEGO

### API

## MODO MULTIJUGADOR

### COMUNICARSE CON EL SERVIDOR

### MATCHMAKING

### MOVIMIENTO DE LOS JUGADORES

### LANZAR OBJETOS

### PREDICCIÓN, RECONCILIACIÓN E INTERPOLACIÓN

### BOTS

# BIBLIOGRAFÍA

Activision. (2004). *Call of Duty*. https://www.youtube.com/watch?v=PrvX0sanW\_E

Beenox. (2019). *Crash Team Racing Nitro Fueled*. https://www.youtube.com/watch?v=51fqnxwQ1kg

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

Daleske, J., y Warner, S. (1973). *Empire*. https://www.youtube.com/watch?v=Yo8mrNPr\_3o

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

Epic MegaGames y Digital Extremes. (1998). *Epic Game’s Unreal*. https://www.youtube.com/watch?v=WTRmZ6aFEWM

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., y T. Berners-Lee. (1999). *Hypertext Transfer Protocol—HTTP/1.1*. Internet Official Protocol Standards. https://tools.ietf.org/html/rfc2616

Gambetta, G. (s. f.). *Fast-Paced Multiplayer*. https://www.gabrielgambetta.com/client-server-game-architecture.html

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

ID Software. (1993). *Doom*. https://www.youtube.com/watch?v=iFnOLFd\_ByQ

Ignatchenko, S. (2017). *Development and deployment of multiplayer online games*. ITHare.com.

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*. https://www.youtube.com/watch?v=9wX5L03r-qA

Konieczny, Ł., y Burdzik, R. (2017). Modern suspension systems for automotive vehicles and their test methods. *Vibroengineering PROCEDIA*, *14*, 233-237. https://doi.org/10.21595/vp.2017.19238

Mockapetris, P. (1987). *Domain Names—Implementation and Specification*. Internet Official Protocol Standards. https://tools.ietf.org/html/rfc1035

Mojang. (2011). *Minecraft*. https://www.youtube.com/watch?v=srd17aD7P3Y

Naughty Dog. (1999). *Crash Team Racing*. https://www.youtube.com/watch?v=OTLREpMM9tE

Nintendo. (1992). *Super Mario Kart*. https://www.youtube.com/watch?v=AlAmXXNz5ac

Nintendo. (2013). *Mario Kart 8*. https://www.youtube.com/watch?v=BR0l84gjEXg

NSW Department of Education and Training. (2008). *Inspect & Service Cooling Systems*.

Postel, J. (1980). *User Datagram Protocol*. Internet Official Protocol Standards. https://tools.ietf.org/html/rfc768

Postel, J. (1981). *Transmission Control Protocol*. Internet Official Protocol Standards. https://tools.ietf.org/html/rfc793

Psyonix. (2015). *Rocket League*. https://www.rocketleague.com/

Schnubel, M. (2018). *Classroom manual for automotive suspension & steering systems* (Seventh edition). Cengage Learning.

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*. https://www.youtube.com/watch?v=pOA0U6yWJqo

Sumo Digital. (2019). *Team Sonic Racing*. https://www.youtube.com/watch?v=s7I\_SI4pRzw

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

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

William Higginbotham. (1958). *Tennis for Two*. https://www.youtube.com/watch?v=s2E9iSQfGdg

1. Aunque hay mas que estos tres, también hay switches, hubs, etc. [↑](#footnote-ref-1)
2. El paquete está muy simplificado. En verdad estaría compuesto de valores binarios. [↑](#footnote-ref-2)