Tella – Videojuego Educativo para Enseñar Matemáticas a Niños

Autor: Jorge Orta López
Tutor: Jordi Linares Pellicer

ANDROID APP ON Google play

Download on the App Store

AGRADECIMIENTOS

Son muchas a las personas a las que tengo que agradecer pues sin ellas nada de esto tendría sentido.

A mi familia, por estar siempre ahí y por ser el motor de mi vida ahora y por siempre. Sin vuestro apoyo incondicional, tanto en los buenos como en los malos momento, me habría sido imposible llegar hasta aquí, gracias.

A mis amigos, pues aunque sobre todo estos últimos años no he pasado todo el tiempo que me hubiese gustado con vosotros, los que estáis sois los que tenéis que estar, y me encantáis, gracias.

A mis profesores, de los que he podido aprender mucho en las clases y de los que he podido aprender aún mucho más fuera de ellas, gracias.

A mis compañeros de trabajo y estudios, con los cuales nos hemos aventurado en esta aventura del saber y si volviera a repetir os querría a todos en el mismo barco de nuevo, gracias.

Y a mi novia, que estos últimos años es la que más me aguanta de todos y se hace a la idea que esto acaba de empezar, por que sabe que me gusta mucho lo que hago y que por el momento no tengo fin, gracias amor.

Todas entráis en mi vida en alguno de estos grupos y muchos en varios a la vez, de nuevo muchas gracias a todos. Va por vosotros.

-¿Que los videojuegos son malos? Eso mismo decían del Rock&Roll. (Shigeru Miyamoto, creador de Mario Bros, Donkey Kong y The Legend of Zeld).

-Los videojuegos son el futuro del aprendizaje. (Steve Jobs, fundador de Apple).

INDICE

Los dispositivos móviles, y los recursos interactivos, cada vez más , se estan haciendo un hueco, dentro de las aulas, utilizándose como un complemento más para el aprendizaje y la evolución de los niños en sus etapas educativas iniciales. Cada vez son más los países en todo el mundo, que acogen las ventajas que proporciona la tecnología móvil y la utilizan con la intención de impulsar los procesos de enseñanza y aprendizaje.

Inmersos en una era digital que avanza a pasos agigantados, cada vez son más los niños que manejan con soltura y naturalidad los dispositivos tecnológicos que tienen a su alcance, por lo que esto nos da una gran oportunidad para integrarlos en su vida educativa, adaptando tecnologías del entretenimiento, como medios educativos no convencionales, que, mediante la diversión, el juego y la experimentación proporcionan a los más pequeños una oportunidad única de descubrir, interactuar, compartir y asimilar conocimientos de manera incosciente mientras se divierten.

Es por estas razones que , desde hace unos años los educadores han empezado a fijarse en las posibilidades pedagógicas de los videojuegos como un recurso para los profesores. Desde el punto de vista educativo, es interesante analizar el éxito de los videojuegos a la luz de las teorías de la motivación y del aprendizaje. Las últimas teorías mantienen que el sujeto no se limita a reaccionar ante el entorno, y que juegan un papel importante la observación, la capacidad humana de emplear símbolos y procesos cognitivos y la capacidad de autorregulación de los sujetos.
De modo sintético, la contribución de la Psicología del Aprendizaje Social al proceso de enseñanza-aprendizaje es la siguiente:

La capacidad humana de emplear símbolos permite representar los fenómenos, analizar su experiencia consciente, planear, imaginar, y actuar de manera previsora.

Los procesos de autorregulación juegan un papel central, seleccionando, organizando y filtrando las influencias externas. El sujeto no se limita a reaccionar
Si bien las investigaciones no son definitivas, la mayoría de ellas indican que muchos videojuegos favorecen el desarrollo de determinadas habilidades, de atención, concentración espacial, resolución de problemas, creatividad, etc. por lo que se concluye que en su conjunto, desde el punto de vista cognitivo, los videojueos suponen algún tipo de ayuda en el desarrollo intelectual, ya que adquieren mejores estrategias de conocimiento, modos de resolver problemas, se benefician en sus habilidades espaciales y aumenta su precisión y capacidad de reacción. No hay evidencia de los efectos contrarios. Sin embargo hay una preocupación creciente entre los padres y educadores, temerosos de que el apego de los niños hacia los videojuegos provoque un mayor aislamiento y reducción de contactos con sujetos de la misma edad.
Aunque no se pueda concluir una causalidad, se puede afirmar que aquellos que son más jugadores tienen una mayor vida social, ven más a sus amigos, demuestran mayor extraversión y mayor iniciación social.

Es importante también destacar dos aspectos, tal y como ha quedado reflejado más arriba. Muchos de los juegos admiten más de un jugador y fomentan en cierto modo el juego en grupo. Por otra parte, aproximadamente un 70 % de los jugadores de videojuegos afirman jugar acompañados, por lo que los supuestos efectos nocivos del juego solitario no parecen tener mucho fundamento.

Del conjunto de investigaciones analizadas podemos sacar la conclusión de que el uso de los videojuegos en la ayuda para determinados aprendizajes y entrenamientos es muy positivo, tal y como se demuestra en el terreno del tratamiento de los problemas de aprendizaje, la ayuda para resolver problemas, para responder a cuestiones relacionadas con la escuela, la familia, aspectos morales, etc. Los videojuegos permiten aumentar la motivación para el aprendizaje de diversas materias como las matemáticas y las ciencias, y el conjunto de las enseñanzas

ANTECEDENTES DEL PROYECTO

De un contexto tan propicio, surge la posibilidad de crerar un proyecto interdisciplinar entre equipos pertenecientes a la universidad de Berguen (Noruega) y la universidad politécnica de Alcoy (España). Este debía incorporar la tecnología de los viedojuegos, para el aprendizaje de matemáticas en niños con necesidades especiales. Siendo este un país activo en cuanto a destinar subvenciones que mejoren la calidad de la enseñanza, el equipo pedagógico del que se hablará más adelante, presentó una propuesta a la Dirección General de Educación del Gobierno Noruego.

En el año 2013 se da luz verde al proyecto, y el Gobierno Noruego aportó la parte económica necesaria para desarrollar dicha aplicación.

EL EQUIPO

En este proyecto, ha participado dos equipos encargados de las siguientes áreas:

EQUIPO PEDAGÓGICO:

Colegio universitario de Berguen, se trata de una institución universitaria relativamente joven situada en la segunda ciudad más grande del País. Cada año acoge a numerosos estudiantes de más de 100 países de procedencia distintos, teniendo una orientación internacional en la educación y en sus instituciones de investigación. La Universidad de Bergen está constantemente en comunicación con el entorno internacional, a fin de ampliar sus fronteras. Está muy involucrada en la cooperación internacional, en la investigación y la educación, firmando así, acuerdos bilaterales con universidades, instituciones de investigación y centros académicos de todas las partes del mundo.

En un principio asumió la parte gráfica de la aplicación, aunque finalmente se acabó dedicando más a la elaboración de mecánicas de juego y programación pedagógica.
De este colegio universitario participan en el equipo pedagógico:
Henning Klafstad, quien se ha encargado de poner la idea original del proyecto (el concepto) y asumir el rol junto a su compañera de Game Designer.
Truda Løvskar, encargada de hacer el desarrollo conceptual.

Statped Vest, que es una organización pública Noruega especializada en docencia a personas de necesidades especiales. Este servicio nacional esta formado por cuatro regiones, bajo la dirección de una oficina principal ubicada en Oslo.
Su función es velar por la integración y la accesibilidad de las personas con necesidades especiales a una educación de calidad. Trabaja con seis areas definidas:

1. Discapacidad auditiva
2. Discapacidades visuales
3. Personas con dificultades en el lenguaje
4. Personas con daño cerebral adquirido
5. Personas con dificultades en el aprendizaje

Garantizan en la medida de lo posible la incorporación de personas con esta serie de necesidades en la escuela local. Para ello prestan asesoramiento y orientación, así como servicios especiales a nivel individual y colectivo.
La representante de dicha organización en Tella es Ragna B. Langlo, y su papel en el proyecto es decisivo a la hora de hacer un desarrollo conceptual y una organización educativa.

EQUIPO TÉCNICO:

UNIVERSIDAD POLITÉCNICA DE VALENCIA, CAMPUS DE ALCOY.

Dentro del Campues de Alcoy de la UPV, existe un equipo de investigación tecnológica, llamado Vertexlit, que se dedica a realizar aplicaciones móviles y videojuegos con el uso de la realidad virtual y augmentada entre otras disciplinas.

Uno de los mayores intereses del equipo es el desarrollo de los denomido como Serius Games. Esta disciplina incorpora la tecnología del entretenimiento con un trasfondo que va más allá para lograr un fin terapéutico o educativo. El director del equipo, es el profesor e investigador Jordi Linares, un profesional que lleva largos años dentro del mundo de los videojuegos, e impartiendo docencia en este campo.
La programación del juego, así como la preparación del mismo para las dos principales plataformas donde se distribuye el juego, Android e IOS, las han llevado a cabo:

Jorge Orta , como programador principal
Juan Jesús Izquierdo
Marc Guillem
Ambos como programadores adicionales.

Y Carolina Oliver, el diseño gráfico y la ilustración.

CARACTERÍSTICAS

TELLA es un juego pedagógico para tabletas en el que los niños pueden aprender matemáticas a través del juego y la superación. Esta pensado principalmente para niños con necesidades especiales (asi como sindrome de down, autísmo, parálisis cerebral, hiperactividad…). Aunque también pueden utilizarlo niños con capacidades intelectuales "normales", de entre 4 y 7 años, pero también puede adaptarse a niños más pequeños dependiendo de su evolución cognitiva.

Se compone de varios módulos (niveles) con las tareas que hay que resolver en un orden predefinido. La progresión del juego y el aprendizaje se planifica cuidadosamente con una progresión lenta. Dentro de la mísma se pueden encontrar más de 250 minijuegos a través de los que superar nuevos retos durante más de dos horas de juego.

ANÁLISIS PEDAGÓGICO

ESTÍMULO-RESPUESTA

En Tella, se pretende esplotar los beneficios de la interactividad, que permite una respues instantanea y su recompensa. Se basa en modelos conductivistas de estímulo-respuesta clasica. La teoría de estímulo-respuesta y el conecepto de la memoria implícita también arroja luz sobre lo que sucede cuando los niños automatizan algunos patrones de comportamiento, lo cual es uno de los propósitos del juego. La automatización de ciertas tareas es vital al realizar tareas matemáticas , no solo en niveles elevados sino también en niveles elementales.
En un proceso matemático, no se puede abstraer toda la lógica implicita para entender el proceso en su totalidad, sino que, en muchas ocasiones, se siguen una serie de pasos basados en una automatización de tareas.

ZONA DE DESARROLLO PRÓXIMO

Así mísmo, el juego debe ser considerado como una herramienta más que tiene el niño de aprendizaje.

La teoria de Vygotsky describe el nivel de desarrollo efectivo del alumno (aquello que es capaz de hacer por sí solo) y el nivel de desarrollo potencial (aquello que sería capaz de hacer con la ayuda de un adulto o un compañero más capaz).

Vigotsky desarrolló a partir de la obsrvación de cómo los niños pequeños desarrollan sus capacidades en la interacción con los demás.

En el contexto del desarrollo del juego, se vuelve importante examinar cómo las características de diferentes mecánicas de juego, influyen en el razonamiento del individuo en diversas maneras.

Define como la diferencia entre el nivel que el niño puede desarrollar por los independientes resoluciones del problema y su posible nivel de desarrollo. Los niños pueden llegar a un nivel más alto en la colaboración con los demás. La zona de desarrollo próximo puede ser visto coo un puente entre el alumno individuo y su entorno cultural. Los juegos de ordenador son claramente artefactos importantes en la cultura que rodea a muchos niños.

Tella está diseñado para explotar estos principios (memoria implícita, de desarroll proximal, instantánios retroalimentación y recompensa, y la automaticidad).

PROCESO DE APRENDIZAJE SUBCONSCIENTE

La asimilación de habilidades y construcción de conocimiento se llevara a cabo en distintas etapas.

En primer lugar la información visual que recoge el niño se proceso de manera incosciente. A través del avance de los niveles, el niño ira tomando conciencia de los conceptos aprendidos hasta llegar a un proceso consciente.

Esto es posible gracias a la memoria implicita.

Siendo así, el resultado deseado consiste en traer al jugador en cortos peridods de frustración, donde lucha para resolver los problemas que se le presentan a lo largo del juego, y por lo tanto mantienen el compromiso. Cuando existe un profesor que constatntemente estimula y desafia al alumno para mejorar, el desarrollo cognitivo puede ser muy interesante.

Se ha diseñado cuidadosamente un argumento de juego divertido y entretenido que enganche al jugador. Porque si se sienten frustrados, o no consiguen superar los retos propuestos, perderán el interés, bajará el nivel de atención, y se dedicarán a otro tipo de tareas.

Es también por este hecho que no se recomienda un tiempo de juego superior a treinta minutos.

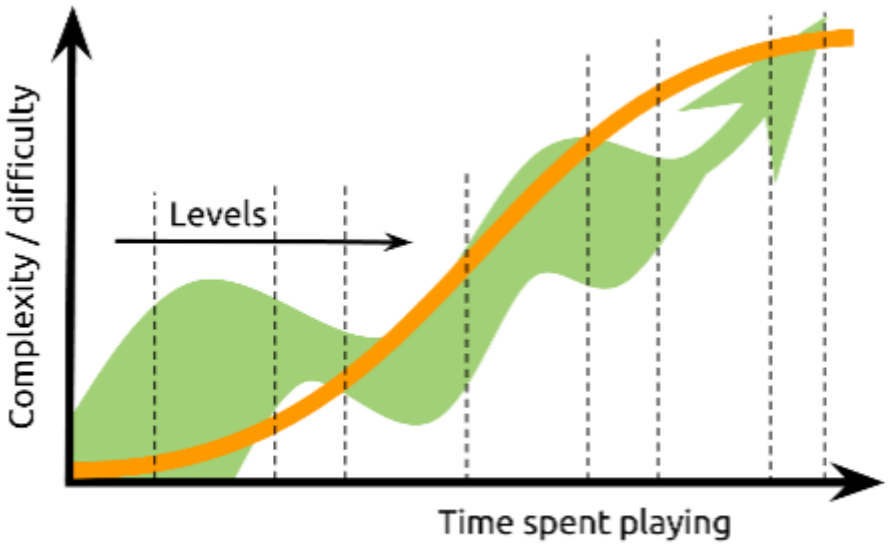


Figure 2: A game for users with special educational needs to keep the technical challenges at an almost even level. The orange line indicated the level of mathematical complexity, which increases throughout

Tella está pensado para poder jugar de forma individual, en compañía de otro niño con el que compartir y relaccionarse socialmente, o bajo la supervisión de un profesor.

El juego funciona mejor en combinación con otra actividad. El niño aprende más cuando trabajan con ejemplos concretos –externos e internos- paralelos a TELLA.

Un ejemplo: El juego "1D" consiste en colocar tamaños en relación con los otros. El aprendizaje se refuerza si el niño juega colocando objetos físicos en orden. Esto puede hacerse con todo tipo de objetos, desde muñecas o coches a bolas de nieve o piedras.

Otro ejemplo: En el juego número 5 el niño trabaja con combinaciones de tamaños. Aquí puede continuar la práctica con dados, botones u otras piezas. Tira un dado, di el número que ha salido y escríbelo en un trozo de papel. Cuenta botones o piedras pequeñas y planas -azules por un lado y rojas por otro- de forma que puedan tirarse y contarse. Aquí tanto las matemáticas como pintar puede ser igual de divertido, y el niño aprende que las piezas pueden formar distintas combinaciones de azul y rojo.

Para un buen aprendizaje es necesario que el niño utilice varios sentidos, también la motricidad y el tacto. Por tanto, es importante "jugar a las matemáticas" con distintos tipos de objetos además de los que aparecen en la pantalla. Además, siempre es divertido compartir con alguien las actividades, pues esto también cuenta desde el punto de vista comunicativo y social.

El juego es adecuado para uso doméstico y también para la enseñanza de matemáticas en la escuela. Ha sido adaptado pedagógicamente para adaptarse tanto en la enseñanza convencional como en educación especial.

CONTENIDO

La mejor manera de conocer el contenido de la aplicación es hacer un paseo por la mísma desde el menu inicial.
Menu inicial
Cuando se inicia el juego, despues de un splash inicial, aparece un menu sencillo con solo tres botones iconográficos muy sencillos en el centro de la pantalla.
Desde ellos se puede acceder:

- BOTÓN PLAY: Acceder directamente al game play para empezar la diversión.

- BOTÓN AJUSTES: Nos lleva al apartado de ajustes para adaptar la configuración a las necesidades del usuario.

- BOTÓN CREDITS: En este caso se puede acceder a información perteneciente al equipo que ha realizado el proyecto y las instituciones implicadas.

El Game play esta compuesto de un scroll que navega por seis mundos (módulos de aprendizaje), los cuales están ordenados por nivel de dificultad.
Cada uno de estos mundos, tiene una temática distinta, y contiene un set de actividades que se suele agrupar en de cuatro a cinco niveles por mundo.

CONTENIDO DE LOS MÓDULOS

1A : Encontrar el objeto más grande sin importar cantidad, color o movimiento.
1B : Encontrar el objeto más pequeño sin importar cantidad, color o movimiento.
1C : Colocar objetos según el tamaño.
1D: Colocar objetos según el tamaño sobre una línea numérica.

tivo

Juego Extra: Laberinto: mover la tableta de manera que las bolas caigan en el agujero correcto.

2A : Números del 1 al 5: aprender los números escuchando y contando.
2B : Números del 6 al nueve: aprender los números escuchando y contando.
2C : Números del 1 al 5: Trazar los números
2D: Números del 6 al 9: Trazar los números
2E: Aprender la relacción entre los numeros hablados y escritos
Juego Extra: Laberinto: mover la tableta de manera que las bolas caigan en el agujero correcto.

3A : Contar. Trabajar con cantidades y números.
3B : Construir cantidades hasta el 9
3C : Combinar cantidades y números – Hasta el 5.
3D: Combinar cantidades y números – Hasta el 9
Juego Extra: Contar el número que aparece en los dados que caen al agitar la tableta.

4A : Contar, por orden, y colocar números en la línea numérica.
4B : Colocar números en la línea numérica.
4C : Encontrar el número más bajo y el más y el más alto. Colocar números en la línea numérica.
4D: Dominio de la linea numérica. Caractéres.
Juego Extra: Atrapar los números adecuados para completar la línea numérica.

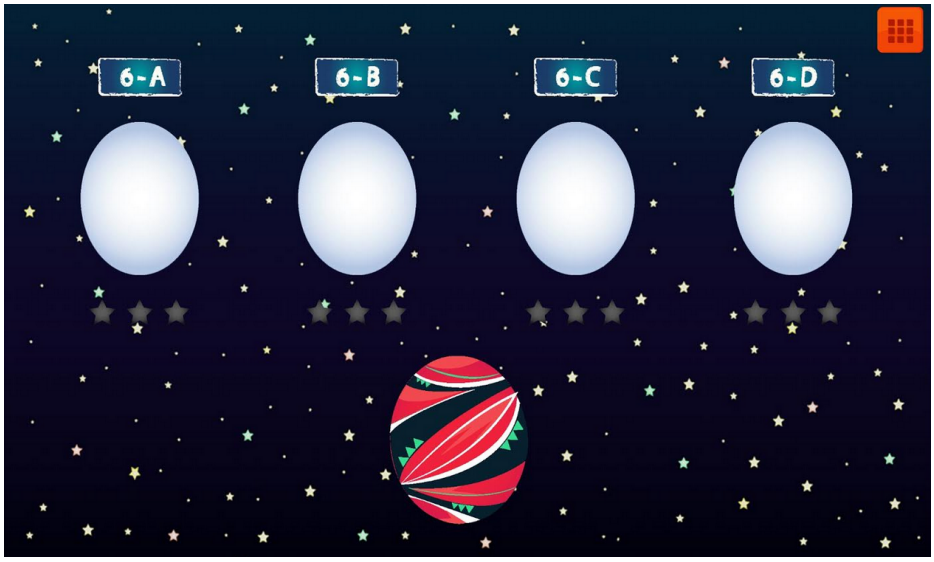
5A : Sumar hasta el 6.
5B : Sumar hasta el 9.
5C: Construir cantidades, colocar "partes del cálculo".
5D: Combinación de cantidades. Sumar y números hasta el 6
5E: Combinación de cantidades. Sumar y números hasta el 9.

Juego Extra: Encontrar el coche con el número correcto como respuesta a la cuenta.

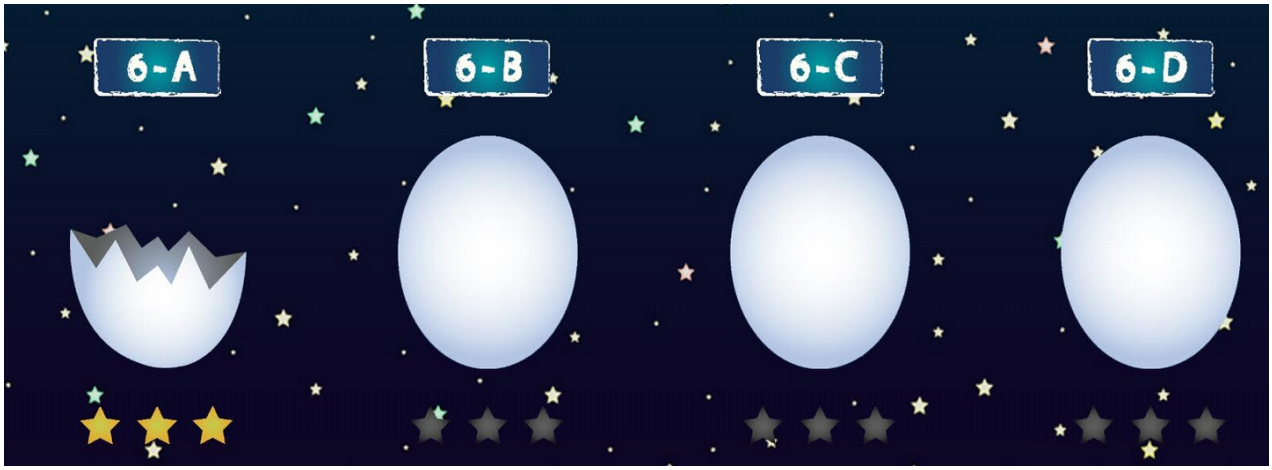
6A : Contar y restar

6B : Restar – Número más Alto: 6.
6C : Restar – Número más alto: 9
6D: Practicar resta y suma sin objetos.
Juego Extra: Atrapar la cantidad adecuada moviendo la tableta.

Cada nivel esta representado con un huevo que hay que romper y resolver a lo largo de todas sus pantallas , y de esta manera, cuando se hayan cubierto todos o gran parte de los retos del mundo que se está jugando, se desactivará un huevo extra, con un divertido juego para premiar la evolución y para poner a prueba los nuevos conocimientos adquiridos en esta nueva etapa.

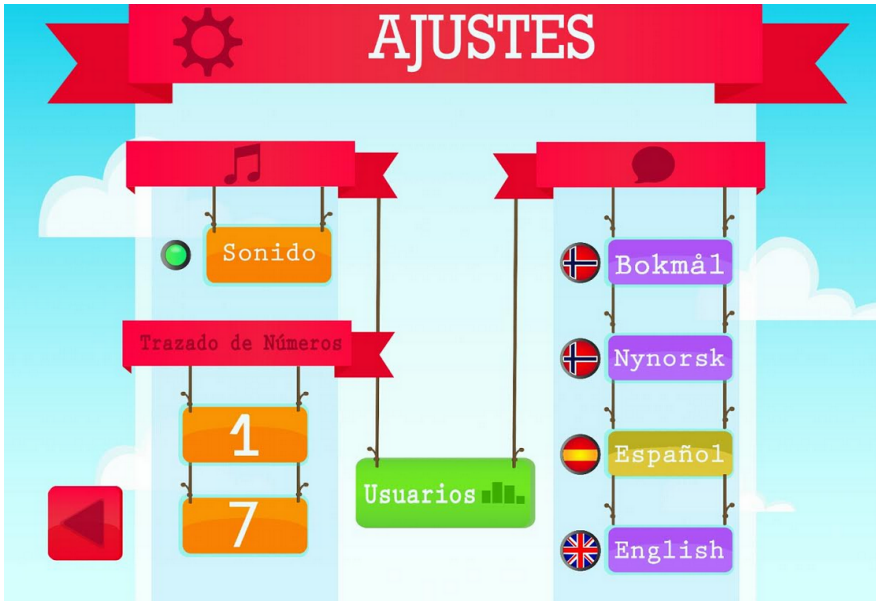


Cada vez que se supera un nuevo nivel, el huevo se rompe, y las estrellas que se encuentran debajo de el se iluminan en función de los intentos realizados para pasar el nivel.



AJUSTES

En el apartado de ajustes, se pueden configurar los aspectos importantes, que condicionaran el juego en función de las necesidades del usuario que esté utilizandolo en un momento determinado.



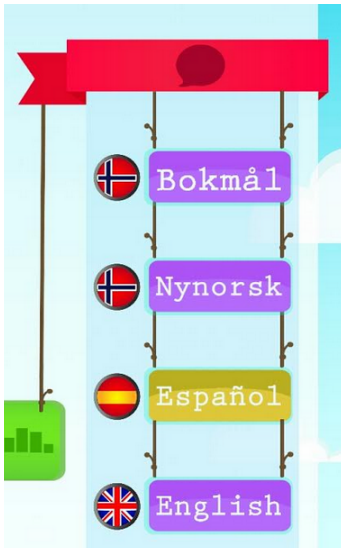
SONIDO

En el juego hay varios tipos de sonidos:

- Hilo musical: Esta presente durante el desarrollo de todo el juego
- Sonidos ambientales: complementan movimientos, anuncian fallos y aciertos y hacen más reales a los personajes que aparecen en el juego
- Locuciones: Se trata de las indicaciones que da Ella, la "ayudante" del niño dentro de la app, para resolver dudas o explicar la mecánica de juego de las distintas pantallas.

Todos estos sonidos, se pueden activar o desactivar, con el botón que hay en la parte superior izquierda de la pantalla.

Cuando pulsas el botón sonido se activa , lo que se evidencia con una luz verde encendida.
Cuando vuelves a pulsar ese mismo botón, el sonido se desactiva y se evidencia con una ausencia de luz, o color gris.



IDIOMA

Una de las cosas más importantes para la buena comunicación y difusión de la aplicación es el idioma en el que esté traducida.
En este caso, inicialmente, Tella, se tradujo a dos idiomas:

- Norsk y Bokmal: Teniendo en cuenta que esta subvencionada por el gobierno de Noruega, se encuentra traducida a los 2 dialectos Locales principales, Norsk y Bokmal.

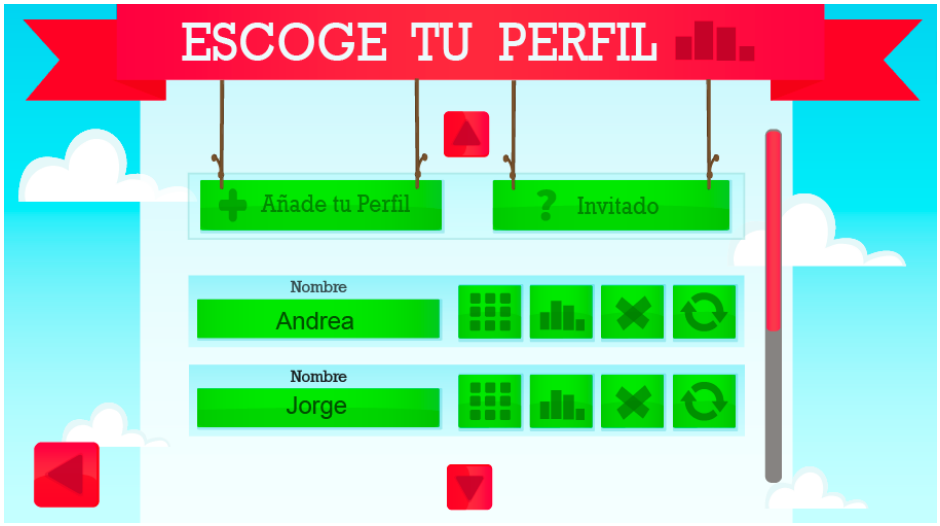
-Español: Por la colaboración con un grupo de investigación español, y considerando que hay más de 500 millones de personas que practican el habla hispana en todo el mundo, se vio una buena opción traducir y adaptar la app al castellano para ampliar sus horizontes.

-Además, en la última actualización también se incluyó una versión en Ingles, que, como todos sabemos, es el idioma más universal y extendido por todo el mundo, lo que consigue hacer más extensible el uso de este recurso en practicamente cualquier rincón del planeta.

En la esquina superior derecha de la pantalla se puede ver una lista con los 4 opciones linguísticas, de las que se puede seleccionar la opción deseada haciendo click sobre el nombre de la mísma.

PERFILES

Para jugar a Tella existe la posibilidad de registrar un perfil de usuario en el que se guarda la evolución y puntuaciones obtenidas a lo largo del recorrido de la aplicación. El código de colores significa lo siguiente:



No es necesario registrarse obliatoriamente, de hecho hay un perfil invitado de usuario para poder jugar sin necesidad de hacer un registro, pero eso implica que no habrá una histórico de nuestra actividad dentro del juego.

PERFIL PROFESOR

Tella debía tener un sistema similar al funcionamiento de una aula común. Esto es posible gracias al perfil Profesor.
El perfil profesor, tiene la capacidad de poder crear distintos perfiles, de todas las personas que van a utilizar la aplicación.()
Una vez realizado un registro muy sencillo, es importante que cada vez, que un usuario vaya a jugar, entre a la aplicación con su nombre de usuario.

A partir de este momento, el profesor que haya configurado el perfil de usuario de un alumno, podrá tener acceso a las "éstadísticas de este usuario después de cada partida, accediendo a la información que le inidique: el tiempo que se ha tardad en resolver cada nivel, cuales son los apartados que le suponen más dificultad o cuales supera con más facilidad… de esta forma se pueden reforzar las areas en las

que se muestren más carencias o saber cuando se ha asimilado correctamente un nuevo concepto y es el momento de avanzar a un nivel más avanzado.
El maestro decide qué desafíos encontrará el alumno. Las actividades/huevos que se desactiven serán invisibles para el alumno. Así, podrá conseguir un"todo correcto" sin tener que enfrentarse a todos los desafíos del juego. Así experimenta el alumno los desafíos.

Puntos negros = inacabado

Puntos verdes = completado

Puntos rojos = invisible para el jugador

El alumno puede jugar cada juego varias veces. Los resultados que se muestran en las estadísticas son los de la última vez que se jugó.

El juego comienza poniendo en práctica la colocación por orden de tamaños. Después continúan los números, conceptos numéricos y cantidades. La línea numérica es importante para comprender la relación entre los números y su colocación por orden. Para finalizar, encontrará actividades de sumas y restas sencillas.

Algunos de los juegos requieren habilidades motoras más allá de "hacer clic" o "arrastrar y soltar": Habrá que mover la tableta para equilibrar los elementos. Habrá que escribir números con los dedos sobre la pantalla. Aquí puede ayudar un adulto. Hay a la venta fundas de plástico que pueden mejorar el agarre de la tableta.

MERCADO ACTUAL DE LOS VIDEOJUEGOS.

Según un estudio realizado por los analistas Juniper Research, en el año 2010 dos tercios de la población mundial tenían teléfonos móviles, y de ellos, más de 450 millones se descargaron juegos en sus terminales, actualmente en 2016 casi 3000 millones de personas tienen un Smartphone. Los ingresos mundiales conseguidos con los videojuegos en 2014 en el mundo fueron de 75.000 millones de dólares. El crecimiento en este negocio y las cifras son abrumadoras por lo que ninguna compañía importante en el sector quiere dejar pasar este tren. Pero a pesar del éxito esperado, el mercado de los juegos en móviles está en plena maduración, y para la industria existe incertidumbre en cuanto a cómo se van a realizar las cosas, qué compañías externas a la telefonía móvil van a participar en esto, qué sistemas dominarán el mercado y si es necesario algún tipo de estándar al respecto.

La falta de una plataforma hardware común o la creación de unos estándares sigue siendo el factor más importante que impide el crecimiento en este mercado. La gran diversidad de plataformas de desarrollo como Android, iOS, BREW, Flash Lite, Java, Linux, Symbian, WAP o Windows Phone, a lo que se une las casi infinitas configuraciones de hardware (resoluciones de pantalla, memoria RAM, la potencia gráfica, formatos de audio, teclado, etc.) obliga a los desarrolladores a crear demasiadas versiones por cada juego que crean.

Para un gran editor de videojuegos mundial el despliegue de un juego en telefonía móvil, tiene un coste adicional considerable en contraste con los sistemas tradicionales de videojuegos, donde un producto puede ser realizado a lo sumo para tres sistemas. Está claro que los costes de producción de un juego tradicional son mucho más altos que cualquiera existente en telefonía móvil, pero de todas formas la existencia de tantas configuraciones diferentes es un coste que quieren eliminar.

Existe un factor adicional al hardware de los teléfonos, y los sistemas de desarrollo empleados, en esta película también tienen mucho que decir las operadoras, muy implicadas en el proceso y que no facilitan para nada la portabilidad de los juegos entre sistemas, ni la creación de estándares. Si nos vamos a países como Japón o Corea, relativamente tienen menos móviles diferentes y también menos operadoras, principal motivo por el cual estos países están superando a los Estados Unidos y Europa en el desarrollo de juegos móviles.

Según datos de Gameloft, una de las principales empresas desarrolladoras de videojuegos en dispositivos móviles, del cinco al siete por ciento de americanos y europeos compra juegos regularmente, mientras que en Japón y Corea la tasa es cercana al 15 por ciento. Pero las operadoras no tienen intención de adoptar un estándar que facilite la creación de videojuegos, debido a las inversiones realizadas en las plataformas existentes y que de alguna forma el software que consigan para ellos, es un factor de diferenciación con la competencia. La fragmentación es un factor cada vez más evidente, si ya supone un obstáculo la presentación de cientos de teléfonos móviles cada año, cada una con sus propias especificaciones y teclado.

Con la llegada de los móviles con tecnología de pantalla táctil, un nuevo componente ha sido añadido a la lista de obstáculos, que termina fragmentando más la cantidad de sistemas donde desarrollar.

El caso del iPhone es realmente especial, es un elemento innovador en el ámbito de la telefonía móvil, pero completamente cerrado e independiente a otras plataformas existentes. Apple está confiada en su éxito, aportando gran cantidad de novedades técnicas y un enorme presupuesto de marketing. Pero en el mercado no siempre funciona lo que es considerado mejor, es necesario que los grandes grupos se pongan de acuerdo y propongan una plataforma común.

Ciertamente los móviles con pantallas táctiles no las ha inventado Apple, pero está consiguiendo cambiar la visión del mercado, ya que seis meses después del lanzamiento del primer iPhone varios fabricantes importantes de dispositivos móviles ya vendían sus propios dispositivos con pantalla táctil.

Existen otros factores que fragmentan el mercado, como la especialización 3D del hardware o el desarrollo de videojuegos basados en la web. La fragmentación es un elemento difícil de separar de la industria móvil en general, es algo a lo que los desarrolladores deben acostumbrarse y deberán moverse hacia las plataformas con más éxito para conseguir más ingresos.

COSAS A TENER ENCUENTA A LA HORA DE DESARROLLAR UN VIDEOJUEGO.

Para abordar el desarrollo de un videojuego, es necesario contar con un equipo multidisciplinar. Aportan variedad en las respuestas y soluciones a un mismo problema, al tener diferentes puntos de vista. Aunque esto complica las cosas, dado que suelen tardar más en organizarse y en sacar conclusiones que un grupo homogéneo.

Se requiere de un alto nivel de abstracción para poder simplificar de manera adecuada los problemas asociados a la implementación de los requisitos.
El uso de una programación a alto nivel nos ayudará a traducir los conceptos recogidos en el proceso de interpretación de peticiones del cliente a un lenguaje comprensible por la plataforma que ejecutará el videojuego.

Se hace gran uso de fundamentos físicos para simular el comportamiento de los diferentes elementos de una escena entre ellos y con el entorno. Esto a su vez implica el uso de complejas fórmulas matemáticas que se encarguen de dar respuesta a estas leyes físicas. Además, y sobre todo en los videojuegos 3D, es imprescindible una capacidad extraordinaria de percepción espacial. En todo momento se debe ser consciente de la ubicación de los diferentes componentes así como las operaciones a realizar para trasladarlos, rotarlos y escalarlos.

Estas exigencias se ven a su vez transferidas a la máquina encargada de interpretar las instrucciones que componen el código de la aplicación. Ésta deberá realizar las múltiples y complejas operaciones matemáticas que previamente hemos diseñado varias veces por segundo. Cuanto más complejas sean dichas fórmulas, y más limitada la máquina en términos de computación, más tiempo tardará en

calcularlas, dando lugar a una restricción en la cantidad de imágenes o cuadros (frames) procesada por segundo.

Todos los efectos y filtros gráficos, las texturas más avanzadas, y la mayor cantidad de polígonos no servirían de nada si no se plasmasen correctamente en pantalla, moviéndose con fluidez .El dato que indica dicha fluidez son los frames por segundo, o frame rate.

Los juegos deben ser ejecutados en lo que se denomina tiempo real, es decir, el ordenador o dispositivo debe ser capaz de visualizar elementos gráficos y animaciones a un mínimo de refrescos por segundo. Si no se consigue un mínimo de refrescos por segundo se pone en peligro dos elementos fundamentales en los juegos: la correcta percepción de las animaciones y la interactividad con el usuario. El número de refrescos de la pantalla por segundo es lo que se conoce en inglés como framerate o frames per second(fps). Este valor indica cuál es la capacidad del juego en cada momento de ofrecer nuevas imágenes y, de manera informal, se estima en un mínimo de 30 fps si se quiere obtener una visualización fluida de animaciones y una respuesta rápida a las acciones del usuario. A mayor fps, mejores animaciones pueden ofrecerse y respuestas más inmediatas al usuario, incrementando la sensación de interactividad, elemento éste imprescindible en videojuegos.

El fps puede depender de muchos factores. Podemos clasificarlos en tres grupos:

· Factores dependientes del dispositivo

· Factores dependientes de la aplicación (videojuego)

· Factores dependientes de los elementos gráficos

En el primer grupo tenemos todo lo que tiene que ver con las características del hardware del dispositivo, principalmente dos elementos: la CPU y la GPU. La GPU (Graphic Processing Unit) es la encargada de llevar a cabo las tareas de visualización de elementos en pantalla.

La tecnología y potencia de los componentes hardware del dispositivo pueden permitir unos fps determinados. A mayor potencia, más velocidad de actualización de los elementos en pantalla y, por tanto, mayor fps. Además, una mayor potencia puede permitirnos mayor complejidad en los elementos gráficos a visualizar.
Los factores dependientes de la aplicación tienen que ver con la complejidad del juego y la cantidad de 'código' que la CPU tiene que ejecutar por cada uno de los refrescos. Cuanto mayor sea la complejidad y cantidad de instrucciones que la CPU tiene que ejecutar menor será el fps.

Los factores dependientes de los elementos gráficos hacen referencia a la complejidad de los mismos con respecto al tiempo requerido por la GPU para su visualización en pantalla.
En los gráficos 3D el elemento fundamental de complejidad es el número de polígonos que componen el objeto gráfico. A mayor número de polígonos, mayor

tiempo necesario para la GPU. Otros elementos a considerar son las texturas utilizadas en los modelos 3D, en este caso su número y tamaño, así como el número de materiales (que determinan sus características ante fuentes de luz).
En los gráficos 2D, como es el caso de Tella, el elemento fundamental de complejidad es su tamaño en píxeles. Si el objeto a visualizar es mayor en píxeles que el área donde tiene que visualizarse el GPU debe realizar un proceso de escalado basado en filtrado de imagen (varios píxeles deben sintetizarse en un píxel final). Si el objeto a visualizar el menor que el área donde tiene que visualizarse la GPU debe llevar un proceso de sobre muestreo que permita duplicar determinados píxeles con objeto de aumentar su tamaño. En ambos casos, las diferencias de tamaño exigen un mayor esfuerzo en la GPU con lo que siempre es conveniente el diseño de los elementos gráficos en tamaños lo más próximos posibles a las áreas donde van a ser visualizados.

La resolución o píxeles

Los dispositivos de juegos tienen todos finalmente que representar sus resultados en un dispositivo de visualización. La pantalla es, por tanto, elemento común en ordenadores, consolas y dispositivos móviles.

Su consideración es imprescindible para la consecución de resultados visuales óptimos. En concreto, el elemento más fundamental que define la pantalla es su resolución o tamaño en píxeles.

Los píxeles son cada uno de los puntos de la matriz de puntos que permite visualizar la pantalla. La resolución se define indicando cuántos píxeles en horizontal y vertical se permiten. Por ejemplo, un iPad permite una resolución de 2048x1536. Esto implica que es necesario adecuar los diseños y gráficos a visualizar al tamaño de pantalla del dispositivo final sobre el que el videojuego va a ejecutarse.

El principal problema radica en la gran variedad de resoluciones que presentan los diferentes dispositivos. Una estrategia conservadora implica dimensionar los gráficos de forma que cubran su tamaño deseado en el dispositivo de mayor resolución. De esta forma, en uno de estos dispositivos lo gráficos tendrán una relación 1:1, también conocido como 'pixel perfect'. El problema viene cuando el dispositivo que finalmente ejecuta el juego dispone de una resolución menor que los dispositivos de mayor resolución. Como se ha comentado anteriormente, en ese caso la GPU tendrá que reescalar los gráficos y, consecuentemente, con un mayor esfuerzo y menor fps final. No utilizar la estrategia anterior puede provocar que en dispositivos de altas resoluciones los gráficos no aparezcan nítidos (han sido ampliados).

EL ASPECT RATIO O RELACIÓN DE ASPECTO

Otro aspecto fundamental de las pantallas junto a su resolución es la relación de aspecto o aspect ratio. Este ratio viene dado por la división entre el número de píxeles en horizontal entre el número de píxeles en vertical.

Las pantallas actuales tienen muy diversos aspect ratio, siendo los más comunes: 1:1, 2:1, 3:2, 4:3, 16:10, 16:9

El aspect ratio debe tenerse muy en cuenta cuando el juego va a desarrollarse sobre una plataforma donde los dispositivos finales pueden tener diferentes relaciones de aspecto. Este es el caso de los dispositivos móviles, que generalmente pueden tener pantallas de relaciones de aspecto dispares, como 3:2, 4:3, 16:10 y 16:9 entre los más comunes.

Diseñar una pantalla de interfaz o un nivel de un juego supone un importante reto en este caso, dado que el diseñador y el programador del juego deben tomar importantes decisiones de qué debe ocurrir ante las diferentes relaciones de aspecto.

Si no se tiene en cuenta, los resultados pueden ser muy insatisfactorios. Por ejemplo, si la pantalla del juego y sus elementos se han diseñado en una proporción 16:9 (muy apaisada), ¿qué debe visualizarse si el dispositivo tiene una proporción final 4:3? Hay tres posibilidades:

· Se encaja la imagen 16:9 sobre 4:3, ajustando la imagen en horizontal y vertical. El resultado es un escalado aniso trópico que tiene como consecuencia romper la proporcionalidad original de la imagen. El resultado final es poco satisfactorio.

· Se reduce la imagen 16:9 hasta que su horizontal encaje la horizontal 4:3. El resultado es que se precisaran de dos márgenes verticales. Es un escalado isotrópico, se conserva la proporción original de la imagen, pero no se aprovecha la totalidad de la pantalla (aparecen márgenes) y el resultado final también es generalmente muy poco satisfactorio.

· Se escala la imagen para que encaje en vertical y se centra en horizontal. Implica, por tanto, un recortado de imagen. Dos fragmentos laterales de la imagen original no van a ser visualizados. El resultado puede ser también insatisfactorio a no ser que esta situación se anticipe adecuadamente.

Solución propuesta en Tella

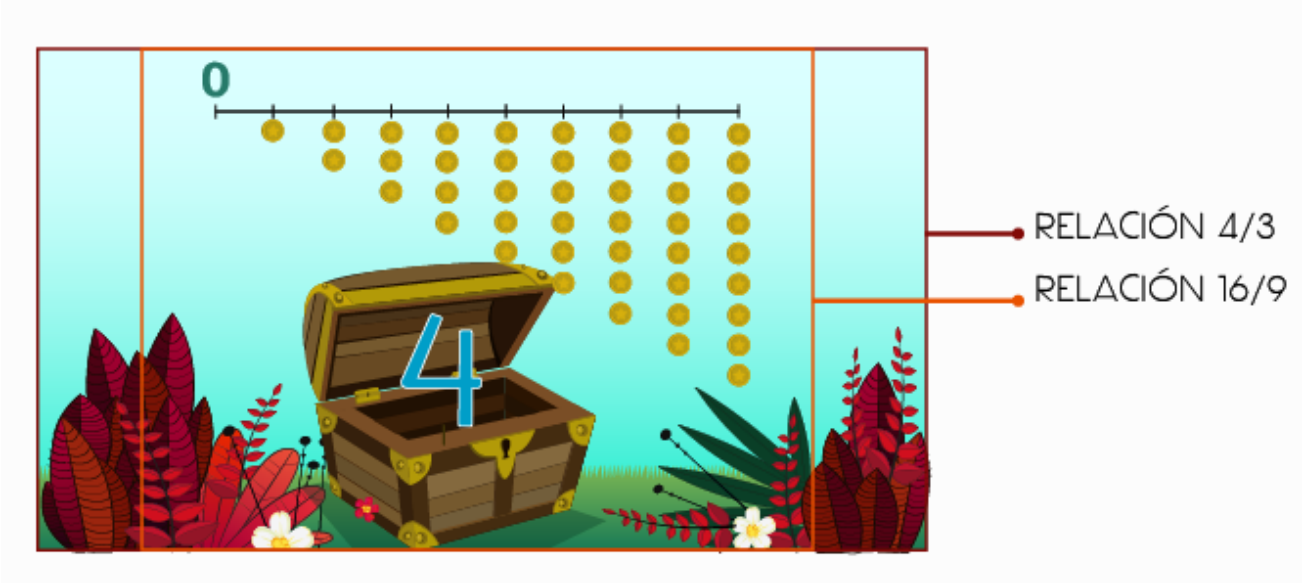
Tella es un videojuego que va destinado a dispositivos móviles, donde podemos encontrar gran diversidad de resoluciones y aspect ratios.

La solución adoptada se puede considerar como una aproximación conservadora. El proceso que se ha elegido es el siguiente:

· Elegir el iPad retina como dispositivo de mayor resolución (2048x1536).
Todos los gráficos se dibujarán teniendo en cuenta esta resolución final. De esta forma los diseños aparecerán nítidos en tabletas con altas resoluciones y el juego realizará un escalado a resoluciones menores en dispositivos inferiores. El objetivo

es ser capaces de sacar la mejor calidad posible de imagen en dispositivos de todas las gamas.

· El lienzo de trabajo será un 16:9, es decir, el más apaisado del mercado.



· Dado que se ha seleccionado como resolución final 2048x1536 (relación de aspecto 4:3) pero queremos cubrir 16:9, se ha definido como resolución de trabajo una resolución de 2730x1536. Esta resolución es 16:9.

· Para que el resultado del diseño de pantallas a resolución 2730x1536 (16:9) aparezca de forma adecuada en otras relaciones de aspecto, el diseño se ha efectuado de forma que los elementos esenciales se centran en el espacio 2048x1536 centrales, dejando dos márgenes laterales donde se dibujarán elementos 'no esenciales' que sólo aparecerían en dispositivos 16:9.

En dispositivos poco apaisados, 4:3, estos márgenes no aparecerán, serán recortados, pero sin que ello conlleve pérdida significativa en el diseño visualizado.

PRINCIPALES PLATAFORMAS MÓVILES.

De entre todas las plataformas móviles mencionadas debemos destacar las 2 con más mercado potencial en la actualidad: iOS y Android.

Estos sistemas acaparan la gran mayoría de usuarios de telefonía móvil y cuentan cada uno con su propio mercado de aplicaciones: App Store y Android Market respectivamente.

Dada la cantidad de terminales que acumulan en sus ventas dichas plataformas, se hace obligatorio tenerlas en cuenta a la hora de abordar un nuevo proyecto de aplicación para dispositivos móviles.

Una de las principales diferencias entre ambos resulta evidente: iOS cuenta con un número determinado de modelos que crece lentamente a razón de 3 o 4 modelos al año, lo que hace que la fragmentación sea infinitamente menor que en Android, donde cientos de dispositivos son presentados por las diferentes marcas cada año. Además, según estudios, resulta evidente que en promedio, el usuario de Ios tiene una mayor predisposición al pago por aplicaciones descargadas de la tienda de la compañía.

Sin embargo, esto no hace menos interesante a Android, donde se han conseguido fórmulas de ingresos por publicidad bastante eficientes. Android utiliza una arquitectura monolítica con un núcleo escrito en C. Utiliza una máquina virtual Dalvik para traducir el código a su lenguaje nativo. Se suele programar en Java,

aunque se puede utilizar NDK para C. Su biblioteca de gráficos 2D y 3D está basada en OpenGL.

IOS tiene un núcleo híbrido dividido en un micronúcleo y servicios. Es un sistema derivado de Mac OS X que integra una interfaz de usuario basada en gestos multi-touch, los cuales se están incorporando a Mac OS X gradualmente, y que han adquirido una gran importancia en Lion. Para desarrollar para esta plataforma, es necesario utilizar Objective-C y Swift.

LIMITACIONES

Recientemente y debido al crecimiento de aplicaciones cada vez más potentes para plataformas móviles, éstas están centrándose principalmente en dotar a sus dispositivos de la tecnología más avanzada en términos de computación capaz de mover los juegos más potentes.
Sin embargo, el afán en obtener modelos más pequeños y ligeros, manteniendo pantallas de grandes resoluciones, con un consumo energético aceptable, además de intentar no disparar su precio de venta, limita las características de hardware que pueden albergar estos dispositivos.
Esto complica la implementación de, por ejemplo, juegos 3D con grandes cargas de cálculos físicos, matemáticos y espaciales.

RETOS

Uno de los grandes retos en el desarrollo de videojuegos para dispositivos móviles, es la potencia de los mismos. Es indudable que ésta aumenta con el tiempo y que cada vez existen modelos más capaces, pero tampoco se puede negar que los jugadores se hacen más exigentes y la tecnología de los motores gráficos evoluciona buscando una respuesta adecuada.

Una característica de los dispositivos móviles, por su propia definición, es el tamaño de sus pantallas. Esto, muchas veces, supone una dificultad añadida, al tener que ubicar los diferentes elementos de la interfaz gráfica estratégicamente, a fin de incluir toda la información necesaria durante el juego, sin perjudicar a visibilidad del escenario, y en definitiva a la jugabilidad.

Como ya se ha hablado anteriormente, existe un gran número de modelos en el mercado, con diferentes sistemas operativos y características hardware. Esta fragmentación obliga a enfocar el desarrollo y pruebas de un videojuego en un subconjunto de dispositivos a fin de garantizar el buen funcionamiento del mismo.
A pesar de todos los problemas mencionados, este mercado supone un gran volumen de negocio que recién empieza a despegar, en el que incluso grandes compañías todavía no están participando y en el que hay mucho pastel a repartir.

¿QUÉ ES UNITY?

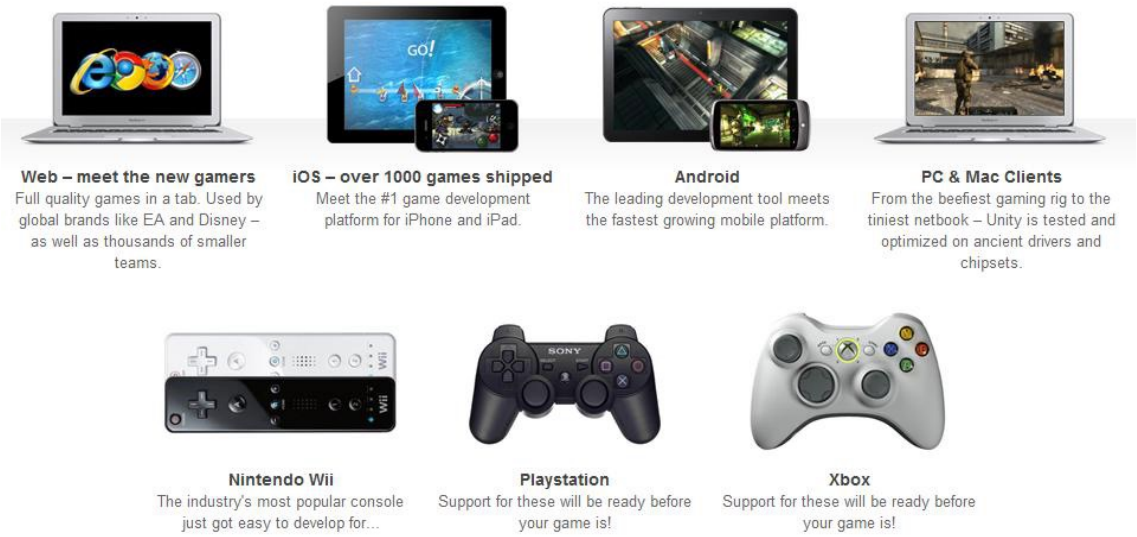
Unity es un entorno de desarrollo para la creación de videojuegos 3D u otros contenidos interactivos, tales como visualizaciones arquitectónicas o animaciones 3D en tiempo real. El entorno se ejecuta en Microsoft Windows y Mac OS X, y los juegos que produce se pueden ejecutar en Windows, Mac, Xbox 360, PlayStation 3, Wii, iPad, iPhone, así como Android. También puede producir juegos para navegador Web utilizando un plugin propio compatible con Mac y Windows, pero no con Linux, aunque actualmente esta implementando la tecnología de WebGL para este fin.

Unity consta de un editor para el desarrollo / diseño de contenidos y un motor de juego para la ejecución del producto final. Unity es similar a Director, Blender game engine, Virtools, Torque Game Builder, y Gamestudio, donde utilizar un entorno gráfico integrado es el método principal de desarrollo. Unity está disponible en dos versiones, una gratuita y otra bajo licencia. Se diferencian en la disponibilidad de algunas funcionalidades y en las posibilidades de compilado. En los últimos años Unity ha evolucionado notablemente y se ha convertido en una de las herramientas para desarrollo de videojuegos más importantes del momento, la cantidad de premios internacionales recibidos así lo atestigua.

PLATAFORMAS ADMITIDAS.

Unity 3D permite desarrollar para todas las plataformas desde una sola herramienta. En un solo proyecto es posible controlar por completo la distribución multiplataforma. Unity hace que sea fácil mantener el código de trabajo a través de muchos dispositivos, abstrae la mayoría de las diferencias entre plataformas.

Cuando requiere un control preciso se puede utilizar #ifdef para especificar código para cada plataforma.
.
Las plataformas admitidas son:

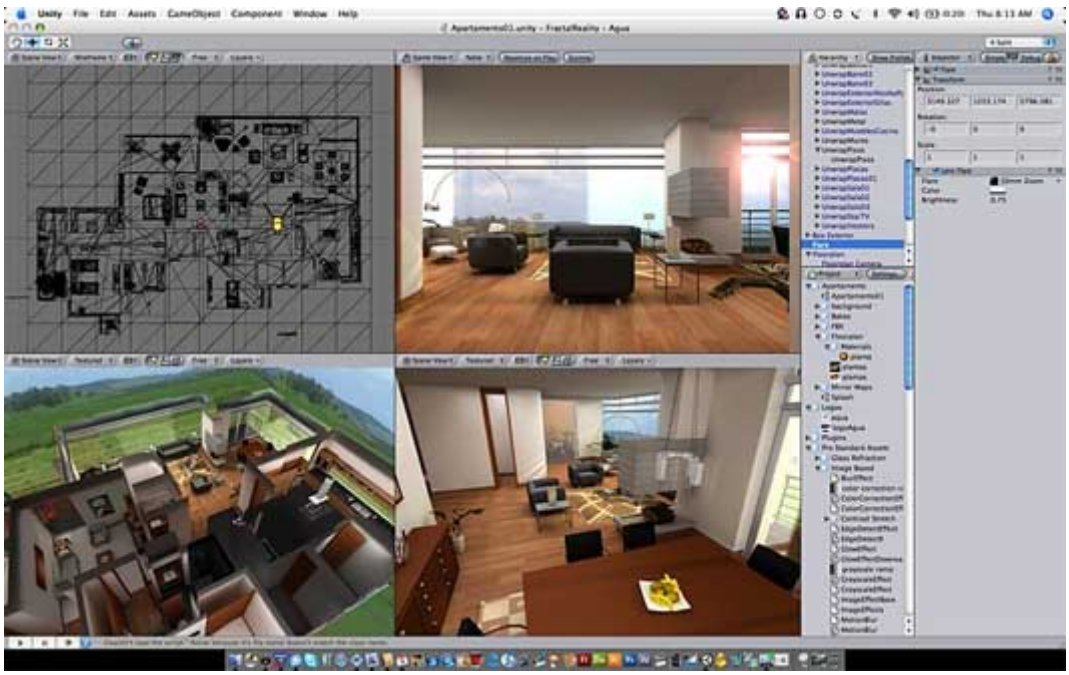


ENTORNO DE TRABAJO:

Empecemos por lo primero, abrir Unity.

Al abrir Unity nos encontraremos con la pantalla inicial, con el proyecto por defecto cargado, si hemos abierto un proyecto antes, lo encontraremos abierto.

Cada sección de la ventana principal es una Vista (View). Existen muchos tipos de vistas en Unity, pero no es necesario verlos todos a la vez. Los distintos modos de interfaz (Layout modes) contienen diferentes configuraciones para las vistas. Puedes observar las combinaciones de vistas haciendo click en el menú contextual de Layout, en la esquina superior derecha, y eligiendo una distribución distinta.



Menú contextual para la distribución de ventanas

Puedes identificar rápidamente una vista con solo ver el nombre que aparece en la esquina superior izquierda de la vista. Las vistas que existen son:
Scene View (Vista de escena) – usada para colocar objetos del juego.
Game View – representación de cómo quedará el juego al ejecutarse.
Hierarchy – una lista de todos los objetos del juego presentes actualmente en la escena.
Project – todos los assets y recursos disponibles para el proyecto.
Inspector – muestra detalles y propiedades del objeto seleccionado.

Vista de escena

Scene View es tu caja de arena interactiva. Podrás usar la Scene View para seleccionar y mover el jugador, la cámara, enemigos y cualquier otro objeto del juego (Gameobject). Mover y manipular objetos con la Scene View es una de las más importantes características de Unity. Esta vista es la mejor forma de ver el juego a través de los ojos de un diseñador en lugar de los de un jugador. Eres libre de utilizar la vista para mover y modificar los objetos como quieras, pero existen unos comandos básicos que debes conocer para usar la Scene View efectivamente.

El primer comando que debes conocer es Frame Selected. Este comando centrará la Scene View en el objeto que tengas seleccionado actualmente. Para usarlo, haz click en el objeto en la vista Hierarchy, mueve tu ratón hasta la Scene View, y pulsa F en el teclado. La Scene view se moverá hasta centrarse en el objeto seleccionado. Este comando es extremadamente útil.

Manipulando en la Scene View

Sobre la scene view encontraremos las herramientas utilizadas para navegar por la scene view, manipular los objetos y definir cómo y dónde se encuentra el pivot point para los objetos seleccionados.

La primera de las herramientas es la view tool, de la cual hablare más tarde. Las otras tres herramientas de la izquierda son las herramientas de manipulación, y las de la derecha son las herramientas de control de posición.

Cuando tengas un objeto seleccionado, verás un Gizmo dibujado alrededor delos 3 ejes dimensionales X, Y, y Z.

Cada una de las herramientas de manipulación tiene un Gizmo diferente que se dibuja alrededor del Objeto (GameObject).

Haz Click y arrastra cualquier eje del Gizmo actual para trasladar, rotar o escalar el componente Trasform del GameObject seleccionado. También puedes hacer Click y arrastrar en el centro del Gizmo para manipular todos los ejes a la vez. Si tienes un ratón con tres botones, puedes hacer click en el botón medio para ajustar el ultimo eje modificado sin hacer Click directamente en él.

Herramienta de Control de Posición

La herramienta de control de posición (Handle Position Tool) es usada para controlar cómo y dónde se muestra el pivot point de un objeto o grupo de objetos. Para alternar entre Center o Pivot haga click sobre la herramienta y alternara entre uno y otro.

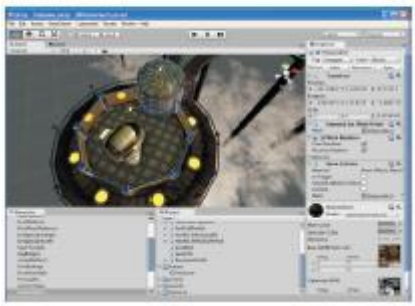
Seleccionando Center hará que el pivot point sea colocado en el centro de la selección actual, mientras que seleccionando Pivot hará que se use el pivot point actual. Herramienta de control de posición en Center, el centro de la selección es usada como Pivot Point. Herramienta de control de posición en Pivot, el Pivot Point actual es el utilizado.

Navegando por la Scene View

Existen distintas formas de moverse por la escena según el ratón que utilices. Recomendamos usar un ratón con tres botones para usar Unity.
Para usar el Orbit mode, mantén pulsada la tecla Alt y arrastra con el botón izquierdo del ratón pulsado.

Para usar el Drag mode, mantén la tecla Alt y arrastra con el botón central del ratón pulsado.

Para usar el Zoom mode, mantén pulsada la tecla Alt y arrastra con el botón derecho del ratón pulsado. Si tienes un ratón con rueda de scroll puedes usar la rueda para hacer zoom.



Modos del View Tool

Ahora hablaremos del primer botón, a la izquierda de las herramientas de manipulación, el view tool. Como su nombre indica es la herramienta para mover la vista y tiene distintas opciones:

View Tool en Drag Mode
(Modo arrastrar) - Tecla "Q"

Con el drag mode seleccionado, mueva el ratón y arrastre haciendo click.
Esto moverá tu vista arriba, abajo, a la izquierda y a la derecha. Necesitarás también acceder a los modos Orbit y Zoom para hacer un uso completo del View Tool.

Para entrar en el modo Orbiy, mantenga la view tool seleccionada y mantenga la tecla option. Ahora haga click y arrastre el ratón y verá como la vista gira alrededor.

Fíjese también en que el icono de la view tool ha cambiado de una mano a un ojo. Puede acceder al modo Zoom presionando la tecla comando. En este modo podrá hacer zoom haciendo click y arrastrando el ratón. Fíjese en que el icono del modo Zoom es una lupa.

Usar los modos de la view tool y arrastrar el ratón es la forma fundamental de moverse por la vista de escena.

View Tool en Orbit Mode
(Modo girar) - Tecla Alt

View Tool en Zoom Mode
(Modo zoom) - Tecla comando

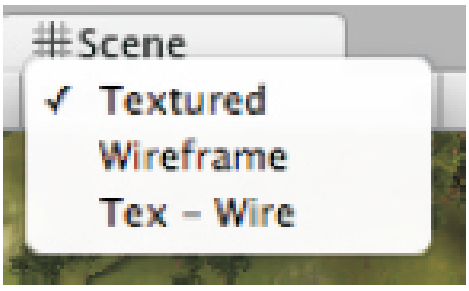
Barra de Control Scene View

Todas las vistas tienen una barra de control diferente en la parte de arriba de la ventana. La barra de control de la scene view tiene varias opciones y se ve así:
El primer menú contextual es el modo de dibujado para la escena. Puede elegir entre ver los objetos de la escena con texturas, en wireframe o con texturas y wireframe. Esto no tiene efecto sobre tu juego cuando es publicado.
El segundo menú contextual es el modo de renderizado de la vista de escena. Puedes elegir ver los valores RGB o Alpha de todos los objetos de la escena. De nuevo, esto no afectará al juego una vez se publique.

El botón de la izquierda activa y desactiva la luz genérica. Cuando el botón esta desactivado, veras una iluminación plana en toda tu escena. Cuando este activado, verás los efectos de luces que has colocado. Tener este botón activado te permitirá ver las luces que tendrá tu juego cuando lo publiques.

El botón de la derecha activa y desactiva distintos efectos como la rejilla de la vista de escena, el cielo y elementos de la interfaz.

Mantener el botón encendido te permitirá ver el juego como quedará una vez publicado.



Menú contextual de dibujado

Menú contextual de Renderizado

Dos botones en la barra de control de la vista de escena.

Opciones de Pestañas

En la esquina superior izquierda nos encontramos con el menú despegable de las opciones de pestaña, en la que podremos maximizar esa pestaña, cerrarla o añadir una nueva. Este menú es común a todas las pestañas. Por último nos encontraremos, a la izquierda de la pestaña con las distintas distribuciones, con el selector de capas, que nos permitirá seleccionar que capas queremos mostrar y cuáles no.

Vista de Juego (Game View)

La vista de juego (Game View) es renderizada utilizando la información de las cámaras en tu juego. Esta vista mostrará lo que se verá cuando ejecutes tu juego. Puedes ver cambios en esta vista si mueves la cámara principal.
Necesitarás usar una o más cámaras para controlar lo que el jugador vea cuando esté jugando.

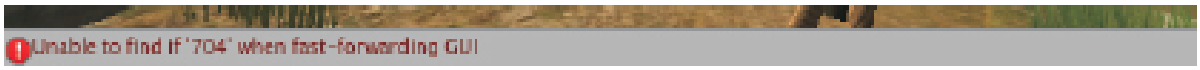
Botones de Reproducción

Los botones de reproducir, pausar y cámara lenta están exactamente en la parte superior central. En cualquier momento mientras estés construyendo una escena, puedes entrar en el modo de reproducción y ver cómo está funcionando tu juego. Pulse el botón de reproducir (Play) para entrar en el modo de reproducción. Mientras que tu escena esté en modo de reproducción también puedes mover, rotar, añadir y borrar objetos. También puedes cambiar configuraciones variables. Cualquier cambio que hagas en el modo de reproducción es temporal y se deshará en el momento en que salgas del modo de reproducción.

Puedes salir pulsando el botón reproducir de nuevo. Mientras estés en el modo de reproducción, puedes pausar y poner el juego a cámara lenta. Pausar e inspeccionar tu escena es una buena forma de ver qué está fallando y si hay algún problema, corregirlo.

Barra de Estado

El área de la barra de estado bajo la vista de juego sirve para múltiples funciones. Informará con mensajes y consejos, mensajes de error, salidas de los scripts... Si hay un problema en tu juego, echar un ojo a la barra de estado es la mejor forma de encontrar el error. Puedes hacer doble click en la barra de estado para traer al frente la ventana de consola, que almacena todos los errores mostrados en la barra de estado.



Barra de Control de la vista de Juego

A la izquierda está el menú de aspecto. Aquí puedes forzar la resolución de la vista de juego a distintos valores. Esto afectará a la posición de cualquier elemento de la interfaz que tengas dentro de Unity. Se puede usar para ver cómo quedará tu juego en diferentes monitores.
En la derecha, nos encontramos primero con el botón de Maximize on Play, con el que podremos activar/desactivar que la vista de juego se maximice al reproducir.

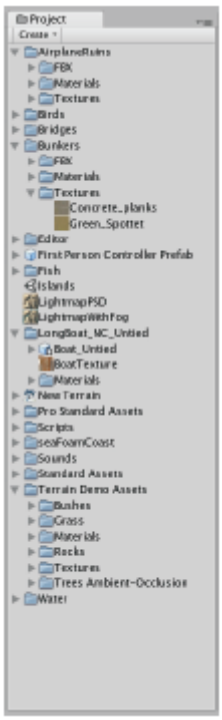
A la derecha está el botón de los Gizmos, que controla que Gizmos dibujados por el usuario son mostrados en la vista de juego.
El último a la derecha en la barra de control es el botón de Stats. Muestra la ventana con las estadísticas de renderizado, que es muy útil para la optimización gráfica.

Vista Proyect View

Cuando creas un proyecto, se genera un grupo de carpetas. Una de esas carpetas se llama Assets folder. La vista del proyecto muestra el contenido de la carpeta Assets del proyecto.

Si abres la carpeta Assets, encontrarás lo que se muestra en la vista del proyecto. La diferencia es que dentro de la vista de proyecto, crearás y enlazarás objetos. Estas relaciones son almacenadas en el grupo de carpetas. Mover los assets en la vista de proyecto mantendrá esas relaciones.

Moverlos desde el Finder romperá esas relaciones. Por este motivo, solo deberás usar el Finder para añadir assets a la carpeta assets. Cualquier otra acción debe realizarse desde la vista de proyecto. Podremos crear carpetas desde el menú contextual Create.



Vista Hierarchy

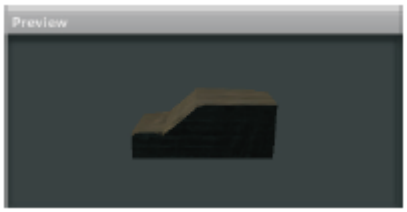
La Hierarchy muestra todos los objetos que se encuentran en la escena actualmente abierta.
Se utiliza para seleccionar y agrupar objetos. Cuando los objetos son eliminados o creados en la escena, aparecerán y desaparecerán en la hierarchy.

Si no encuentras un objeto en la vista de escena, puedes buscarlo y editar sus características desde aquí. Unity utiliza un sistema de herencias, es decir, un objeto se puede agrupar con otro y "heredar" características del primero.

Vista Inspector

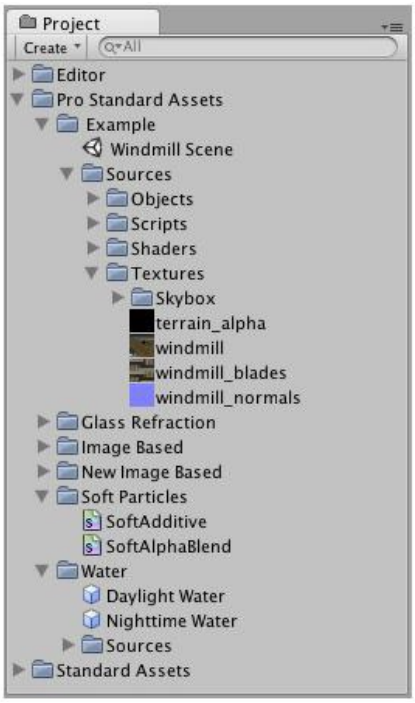
El Inspector muestra información básica sobre el objeto seleccionado, así como la información de sus componentes y sus propiedades. Aquí es donde modificaras las características de tus objetos. Todos los objetos contienen diferentes componentes. Cuando estás viendo un objeto en el inspector, cada componente tiene su propia barra de título. Por ejemplo, todos los objetos tienen un componente Transform. Todas las características variables de un objeto se pueden cambiar en el inspector.

El inspector también incluye, desde la versión 2.5 un espacio para previsualizaciones. En este espacio podremos previsualizar texturas, modelos 3d, lightmaps, sonidos... Una opción muy práctica para navegar entre nuestros objetos.



CONCEPTO DE ASSET.

Un Asset es cualquier tipo de recurso que necesitemos utilizar en el juego. Ejemplos de Assets son las texturas, las mallas 3D de los modelos de los objetos, audio, video. Al margen de los anteriores también pueden ser assets plugins que se integran mediante código, shaders y por supuesto, cualquier script que creemos nosotros mismos.
Existen dos tipos de assets especiales, los Prefabs y la Escenas. Sobre los primeros se hablará más detenidamente en el siguiente apartado. Las Escenas son el método empleado por Unity para crear espacios o secciones del juego, se guardan como un único fichero en forma de asset.
Cabe destacar que Unity es capaz de soportar ficheros de multitud de aplicaciones distintas, al margen de los estándares. Además permite configurar la importación de recursos externos, como por ejemplo ajustando la resolución de texturas. Facilita enormemente la edición de este tipo de recursos lanzando las aplicaciones apropiadas.
Otro punto interesante sobre el concepto de Asset es la Asset Store, una tienda online propiedad de Unity donde se pueden comprar Assets.



CONCEPTO DE GAMEOBJECT.

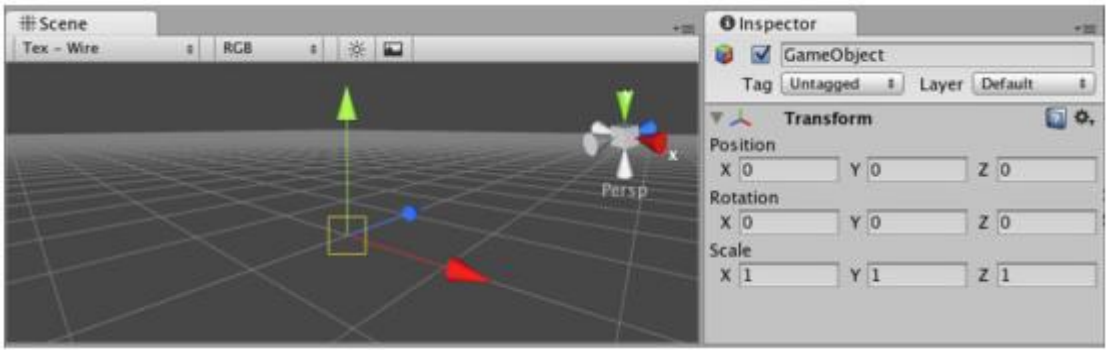
Todo lo existente en una escena del juego son GameObjects. Un GameObject es una especie de contenedor que alberga diferentes piezas que le dotan de aspecto y

comportamiento. Cada una de estas piezas se denomina Componente, más adelante se hablará de ellos más detenidamente.

Los GameObjects pueden agruparse formando jerarquías de padres e hijos. Es posible ejecutar acciones sobre jerarquías completas de GameObjects, por ejemplo el movimiento.

Todo GameObject se compone de al menos un Componente denominado Transform. Éste es el encargado de implementar las agrupaciones jerárquicas además de la posición, rotación y escalado del GameObject.

Inicialmente un GameObject es un contenedor vacío (salvo por el Transform), pueden existir GameObjects que sólo implementen comportamiento sobre otros GameObjects sin visualizarse en la escena.



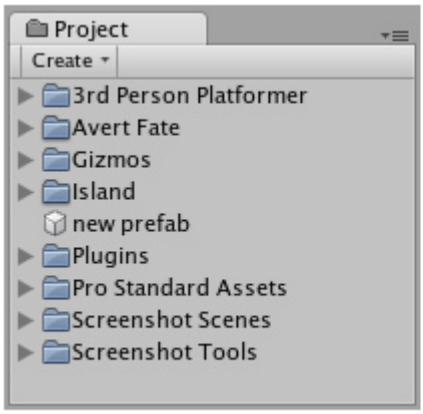
CONCEPTO DE PREFAB.

Agrupando y combinando GameObjects podemos llegar a crear estructuras complejas que actúan como una sola. Seguramente necesitemos tener en una escena o en escenas distintas copias de dicha estructura.

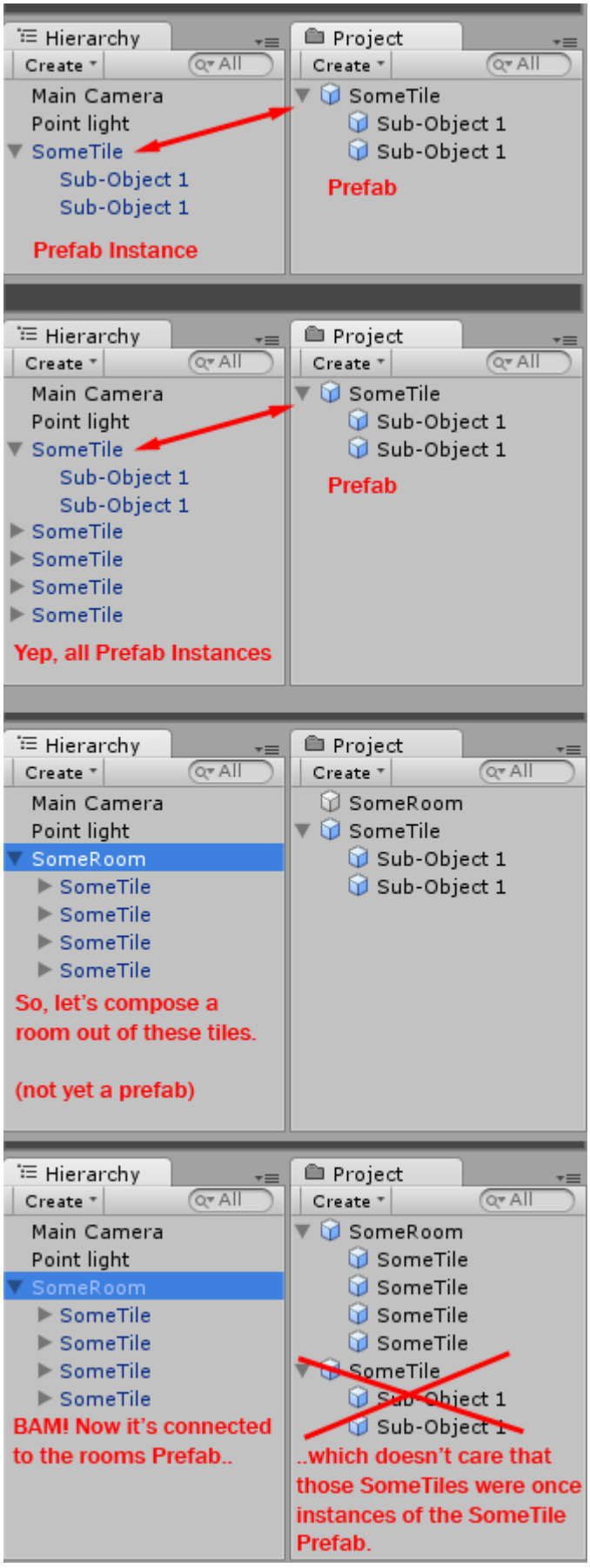
Un Prefab es un Asset que agrupa la estructura de GameObects en un único "objeto" (es posible una visualización desplegada), de esta forma podemos crear tantas instancias de nuestro Prefab como necesitemos sin necesidad de duplicar trabajo ni memoria.
Trabajar con Prefabs es realmente cómodo. Cuando creamos una instancia de un Prefab, podemos modificarla cambiando el valor de propiedades y variables públicas de modo que se puede adaptar cada instancia según convenga.

Un prefab puede ser editado. Cuando cambiamos alguna propiedad de un prefab, este cambio se aplica a todas las instancias (excepto los cambios relativos a

Transform). También es posible realizar cambios sobre una instancia y aplicarlos al Prefab original.
Trabajar con prefabs es el único modo de poder crear GameObjects en tiempo de ejecución mediante scripting.



A new, empty Prefab. It cannot be instanced until you fill it with a GameObject.

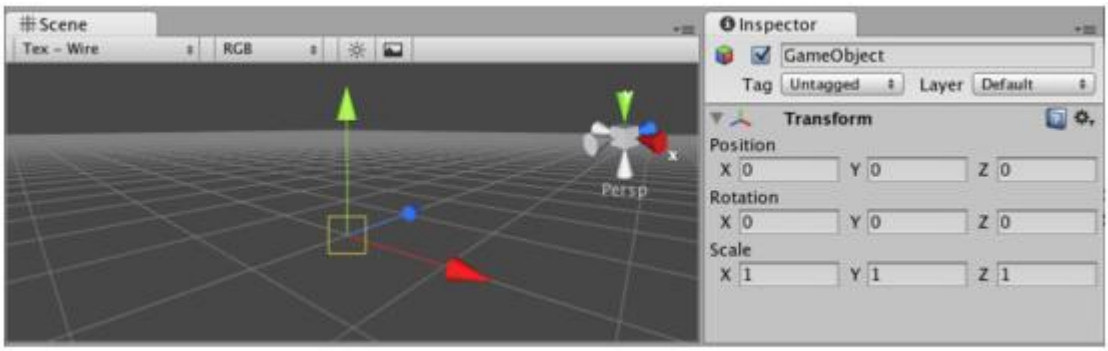

Hierarchy
Create
All
Main Camera
Point light
SomeTile
Sub-Object 1
Sub-Object 1
Prefab Instance
Project
Create
All
SomeTile
Sub-Object 1
Sub-Object 1
Prefab
Hierarchy
Create
All
Main Camera
Point light
SomeTile
Sub-Object 1
Sub-Object 1
SomeTile
SomeTile
SomeTile
SomeTile
Yep, all Prefab Instances
Project
Create
All
SomeTile
Sub-Object 1
Sub-Object 1
Prefab
Hierarchy
Create
All
Main Camera
Point light
SomeRoom
SomeTile
SomeTile
SomeTile
SomeTile
So, let's compose a room out of these tiles.
(not yet a prefab)
Project
Create
All
SomeRoom
SomeTile
Sub-Object 1
Sub-Object 1
Hierarchy
Create
All
Main Camera
Point light
SomeRoom
SomeTile
SomeTile
SomeTile
SomeTile
BAM! Now it's connected to the rooms Prefab..
Project
Create
All
SomeRoom
SomeTile
SomeTile
SomeTile
SomeTile
SomeTile
Sub-Object 1
Sub-Object 1
..which doesn't care that those SomeTiles were once instances of the SomeTile Prefab.

CONCEPTO DE COMPONENTE Y PRINCIPALES COMPONENTES.

Tal y como ya se ha mencionado anteriormente los objetos presentes en una escena del juego están formados por combinaciones de componentes. Un GameObject puede tener muchos componentes distintos que determinan su aspecto y su comportamiento.

Algunos de los tipos de componente más importante son:

Transform: Ya se ha hablado anteriormente de este componente. Su función es mantener la jerarquía de relación entre GameObjects y determinar su posición, rotación y escalado.

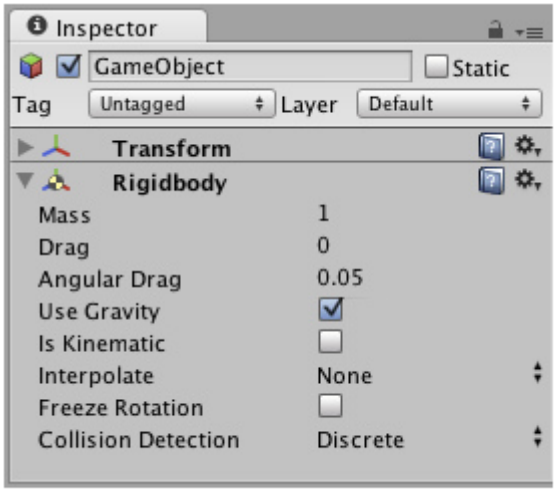


*The Transform Component is viewable and editable in the **Inspector***

Variables	
position	The position of the transform in world space.
localPosition	Position of the transform relative to the parent transform.
eulerAngles	The rotation as Euler angles in degrees.
localEulerAngles	The rotation as Euler angles in degrees relative to the parent transform's rotation.
right	The red axis of the transform in world space.
up	The green axis of the transform in world space.
forward	The blue axis of the transform in world space.
rotation	The rotation of the transform in world space stored as a Quaternion.
localRotation	The rotation of the transform relative to the parent transform's rotation.
localScale	The scale of the transform relative to the parent.
parent	The parent of the transform.
worldToLocalMatrix	Matrix that transforms a point from world space into local space (Read Only).
localToWorldMatrix	Matrix that transforms a point from local space into world space (Read Only).
root	Returns the topmost transform in the hierarchy.
childCount	The number of children the Transform has.
lossyScale	The global scale of the object (Read Only).

Functions	
Translate	Moves the transform in the direction and distance of translation.
Rotate	Applies a rotation of eulerAngles.z degrees around the z axis, eulerAngles.x degrees around the x axis, and eulerAngles.y degrees around the y axis (in that order).
RotateAround	Rotates the transform about axis passing through point in world coordinates by angle degrees.
LookAt	Rotates the transform so the forward vector points at /target/'s current position.
TransformDirection	Transforms direction from local space to world space.
InverseTransformDirection	Transforms a direction from world space to local space. The opposite of Transform.TransformDirection.
TransformPoint	Transforms position from local space to world space.
InverseTransformPoint	Transforms position from world space to local space. The opposite of Transform.TransformPoint.
DetachChildren	Unparents all children.
Find	Finds a child by name and returns it.
IsChildOf	Is this transform a child of parent?

RigidBody: Permite aplicar (o recibir) fuerzas a un GameObject para moverlo por el espacio. Permite establecerlo en modo Kinemátic lo que le impide recibir fuerzas autocalculadas por el motor físico de Unity.



An empty GameObject with a Rigidbody Component attached

Variables

velocity	The velocity vector of the rigidbody.
angularVelocity	The angular velocity vector of the rigidbody.
drag	The drag of the object.
angularDrag	The angular drag of the object.
mass	The mass of the rigidbody.
useGravity	Controls whether gravity affects this rigidbody.
isKinematic	Controls whether physics affects the rigidbody.
freezeRotation	Controls whether physics will change the rotation of the object.
constraints	Controls which degrees of freedom are alowed for the simulation of this Rigidbody.
collisionDetectionMode	The Rigidbody's collision detection mode.
centerOfMass	The center of mass relative to the transform's origin.
worldCenterOfMass	The center of mass of the rigidbody in world space (Read Only).
inertiaTensorRotation	The rotation of the inertia tensor.
inertiaTensor	The diagonal inertia tensor of mass relative to the center of mass.
detectCollisions	Should collision detection be enabled? (By default always enabled)
useConeFriction	Force cone friction to be used for this rigidbody.
position	The position of the rigidbody.
rotation	The rotation of the rigdibody.
interpolation	Interpolation allows you to smooth out the effect of running physics at a fixed frame rate.
solverIterationCount	Allows you to override the solver iteration count per rigidbody.
sleepVelocity	The linear velocity, below which objects start going to sleep. (Default 0.14) range { 0, infinity }
sleepAngularVelocity	The angular velocity, below which objects start going to sleep. (Default 0.14) range { 0, infinity }
maxAngularVelocity	The maximimum angular velocity of the rigidbody. (Default 7) range { 0, infinity }

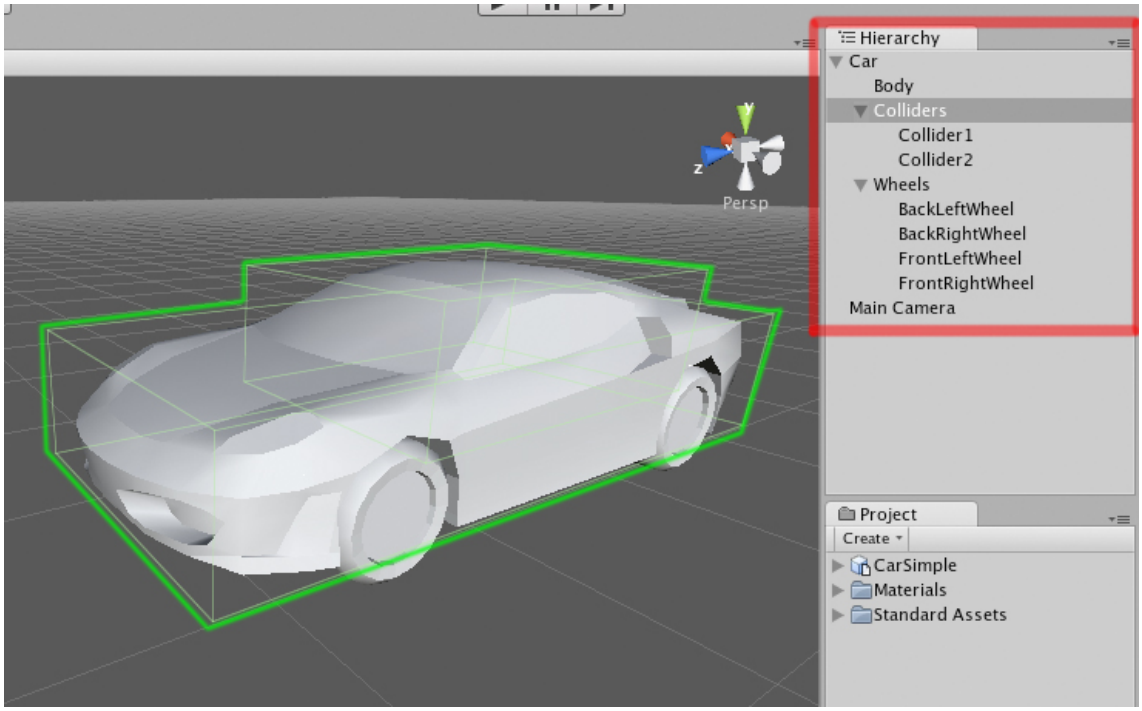
Functions

SetDensity	Sets the mass based on the attached colliders assuming a constant density.
AddForce	Adds a force to the rigidbody. As a result the rigidbody will start moving.
AddRelativeForce	Adds a force to the rigidbody relative to its coordinate system.
AddTorque	Adds a torque to the rigidbody.
AddRelativeTorque	Adds a torque to the rigidbody relative to the rigidbodie's own coordinate system.
AddForceAtPosition	Applies force at position. As a result this will apply a torque and force on the object.
AddExplosionForce	Applies a force to the rigidbody that simulates explosion effects. The explosion force will fall off linearly with distance to the rigidbody.
ClosestPointOnBounds	The closest point to the bounding box of the attached colliders.
GetRelativePointVelocity	The velocity relative to the rigidbody at the point relativePoint.
GetPointVelocity	The velocity of the rigidbody at the point worldPoint in global space.
MovePosition	Moves the rigidbody to position.
MoveRotation	Rotates the rigidbody to rotation.
Sleep	Forces a rigidbody to sleep at least one frame.
IsSleeping	Is the rigidbody sleeping?
WakeUp	Forces a rigidbody to wake up.
SweepTest	Tests if a rigidbody would collide with anything, if it was moved through the scene.
SweepTestAll	Like Rigidbody.SweepTest, but returns all hits.

Messages Sent

OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionExit	OnCollisionEnter is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionStay	OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

Collider: Es el componente que detecta las colisiones con otros objetos. Por defecto es controlado por el motor físico de Unity quien calcula las reacciones a las colisiones. También es posible establecerlo en modo Trigger, en este modo sólo se detectan colisiones y se lanzan eventos para que el programador pueda determinar el comportamiento exacto ante una colisión.

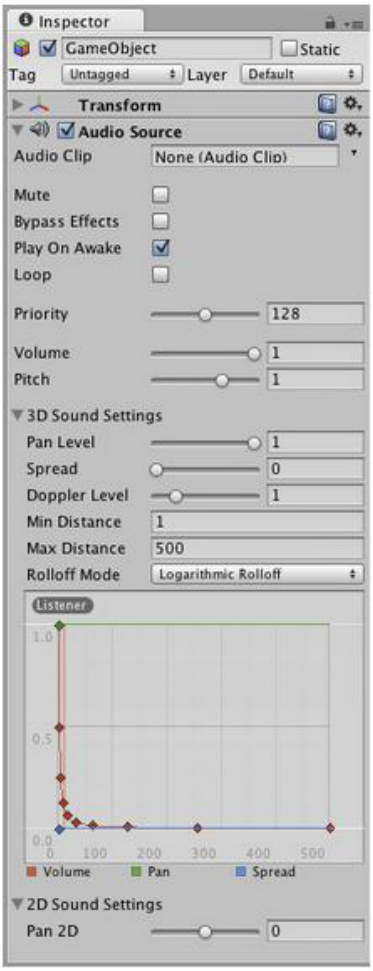


Variables	
enabled	Enabled Colliders will collide with other colliders, disabled Colliders won't.
attachedRigidbody	The rigidbody the collider is attached to.
isTrigger	Is the collider a trigger?
material	The material used by the collider.
sharedMaterial	The shared physic material of this collider.
bounds	The world space bounding volume of the collider.

Functions	
ClosestPointOnBounds	The closest point to the bounding box of the attached collider.
Raycast	Casts a Ray that ignores all Colliders except this one.

Messages Sent	
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerStay	OnTriggerStay is called *almost* all the frames for every Collider **other** that is touching the trigger.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionStay	OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.

AudioSource: Permite reproducir clips de audio en 3D.



Variables	
volume	The volume of the audio source.
pitch	The pitch of the audio source.
time	Playback position in seconds.
timeSamples	Playback position in PCM samples.
clip	The default AudioClip to play
isPlaying	Is the clip playing right now (Read Only)?
loop	Is the audio clip looping?
ignoreListenerVolume	This makes the audio source not take into account the volume of the audio listener.
playOnAwake	If set to true, the audio source will automatically start playing on awake
velocityUpdateMode	Whether the Audio Source should be updated in the fixed or dynamic update.
panLevel	Sets how much the 3d engine has an effect on the channel.
bypassEffects	Bypass effects (Applied from filter components or global listener filters)
dopplerLevel	Sets the Doppler scale for this AudioSource
spread	Sets the spread angle a 3d stereo or multichannel sound in speaker space.
priority	Sets the priority of the AudioSource
mute	Un- / Mutes the AudioSource. Mute sets the volume=0, Un-Mute restore the original volume.
minDistance	Within the Min distance the AudioSource will cease to grow louder in volume.
maxDistance	(Logarithmic rolloff) MaxDistance is the distance a sound stops attenuating at.
pan	Sets a channels pan position linearly. Only works for 2D clips.
rolloffMode	Sets/Gets how the AudioSource attenuates over distance

MeshFilter: Es el componente que determina la forma del objeto (la malla).

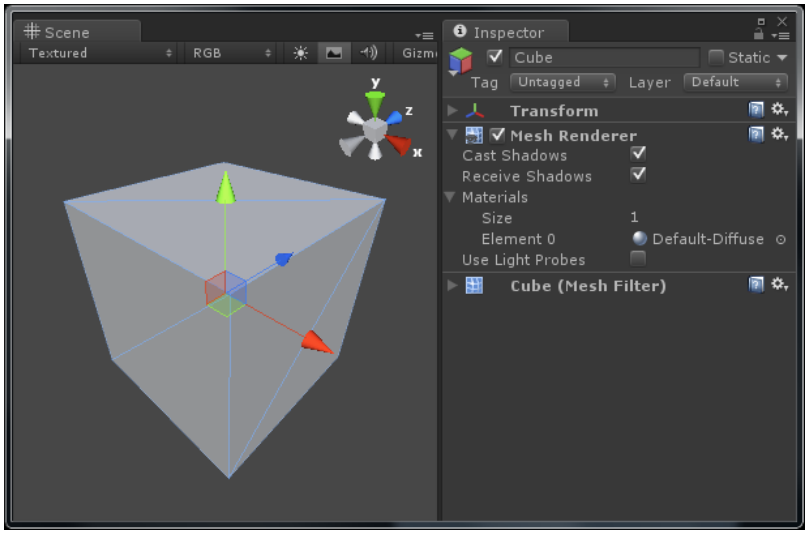
Variables	
mesh	Returns the instantiated Mesh assigned to the mesh filter.
sharedMesh	Returns the shared mesh of the mesh filter.

Variables	
vertices	Returns a copy of the vertex positions or assigns a new vertex positions array.
normals	The normals of the mesh.
tangents	The tangents of the mesh.
uv	The base texture coordinates of the mesh.
uv2	The second texture coordinate set of the mesh, if present.
bounds	The bounding volume of the mesh.
colors	Returns the vertex colors of the mesh.
triangles	An array containing all triangles in the mesh.
vertexCount	Returns the number of vertices in the mesh (Read Only).
subMeshCount	The number of submeshes. Every material has a separate triangle list.
boneWeights	The bone weights of each vertex
bindposes	The bind poses. The bind pose at each index refers to the bone with the same index.

Constructors	
Mesh	Creates an empty mesh

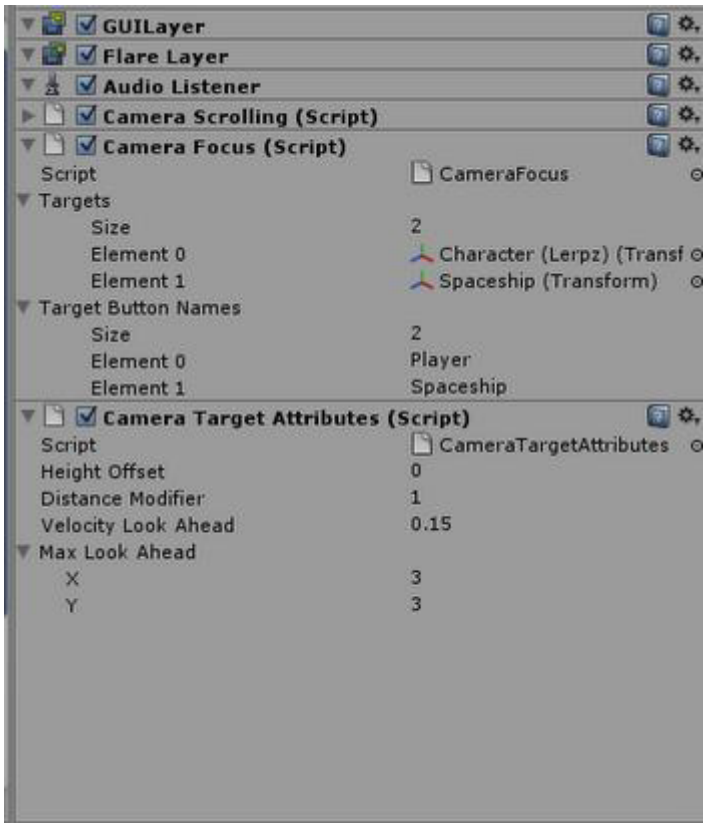
Functions	
Clear	Clears all vertex data and all triangle indices.
RecalculateBounds	Recalculate the bounding volume of the mesh from the vertices.
RecalculateNormals	Recalculates the normals of the mesh from the triangles and vertices.
Optimize	Optimizes the mesh for display.
GetTriangles	Returns the triangle list for the submesh.
SetTriangles	Sets the triangle list for the submesh
CombineMeshes	Combines several meshes into this mesh.

MeshRenderer: Determina como se visualiza la forma de un objeto (el material).



Variables	
isPartOfStaticBatch	Has this renderer been statically batched with any other renderers?
worldToLocalMatrix	Matrix that transforms a point from world space into local space (Read Only).
localToWorldMatrix	Matrix that transforms a point from local space into world space (Read Only).
enabled	Makes the rendered 3D object visible if enabled.
castShadows	Does this object cast shadows?
receiveShadows	Does this object receive shadows?
material	The material of this object.
sharedMaterial	The shared material of this object.
sharedMaterials	All the shared materials of this object.
materials	All the materials of this object.
bounds	The bounding volume of the renderer (Read Only).
lightmapIndex	The index of the lightmap applied to this renderer.
lightmapTilingOffset	The tiling & offset used for lightmap.
isVisible	Is this renderer visible in any camera? (Read Only)

Scripting: Cualquier script que el desarrollador programe con el objetivo de controlar el comportamiento de un GameObject, también debe ser añadido como componente. El panel de Inspección de Unity nos permite visualizar y modificar (sin necesidad de abrir el código) los valores de las variable públicas de nuestros scripts.



Variables

useGUILayout	Disabling this lets you skip the GUI layout phase.

Functions

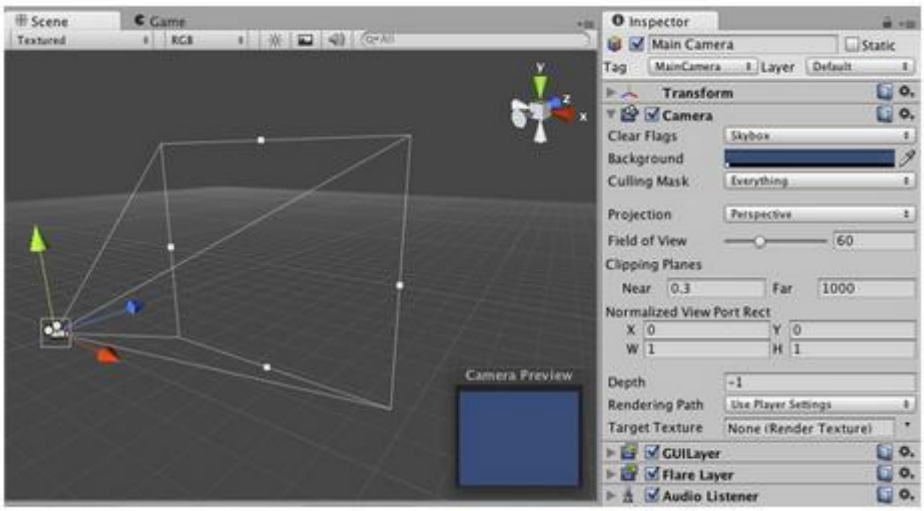
Invoke	Invokes the method methodName in time seconds.
InvokeRepeating	Invokes the method methodName in time seconds.
CancelInvoke	Cancels all Invoke calls on this MonoBehaviour.
IsInvoking	Is any invoke on methodName pending?
StartCoroutine	Starts a coroutine.
StopCoroutine	Stops all coroutines named methodName running on this behaviour.
StopAllCoroutines	Stops all coroutines running on this behaviour.

Overridable Functions

Update	Update is called every frame, if the MonoBehaviour is enabled.
LateUpdate	LateUpdate is called every frame, if the Behaviour is enabled.
FixedUpdate	This function is called every fixed framerate frame, if the MonoBehaviour is enabled.
Awake	Awake is called when the script instance is being loaded.
Start	Start is called just before any of the Update methods is called the first time.
Reset	Reset to default values.
OnMouseEnter	OnMouseEnter is called when the mouse entered the GUIElement or Collider.
OnMouseOver	OnMouseOver is called every frame while the mouse is over the GUIElement or Collider.
OnMouseExit	OnMouseExit is called when the mouse is not any longer over the GUIElement or Collider.
OnMouseDown	OnMouseDown is called when the user has pressed the mouse button while over the GUIElement or Collider.
OnMouseUp	OnMouseUp is called when the user has released the mouse button.
OnMouseUpAsButton	OnMouseUpAsButton is only called when the mouse is released over the same GUIElement or Collider as it was pressed.
OnMouseDrag	OnMouseDrag is called when the user has clicked on a GUIElement or Collider and is still holding down the mouse.
OnTriggerEnter	OnTriggerEnter is called when the Collider other enters the trigger.
OnTriggerExit	OnTriggerExit is called when the Collider other has stopped touching the trigger.
OnTriggerStay	OnTriggerStay is called once per frame for every Collider other that is touching the trigger.
OnCollisionEnter	OnCollisionEnter is called when this collider/rigidbody has begun touching another rigidbody/collider.
OnCollisionExit	OnCollisionExit is called when this collider/rigidbody has stopped touching another rigidbody/collider.
OnCollisionStay	OnCollisionStay is called once per frame for every collider/rigidbody that is touching rigidbody/collider.
OnControllerColliderHit	OnControllerColliderHit is called when the controller hits a collider while performing a

OnJointBreak	Called when a joint attached to the same game object broke.
OnParticleCollision	OnParticleCollision is called when a particle hits a collider.
OnBecameVisible	OnBecameVisible is called when the renderer became visible by any camera.
OnBecameInvisible	OnBecameInvisible is called when the renderer is no longer visible by any camera.
OnLevelWasLoaded	This function is called after a new level was loaded.
OnEnable	This function is called when the object becomes enabled and active.
OnDisable	This function is called when the behaviour becomes disabled () or inactive.
OnDestroy	This function is called when the MonoBehaviour will be destroyed.
OnPreCull	OnPreCull is called before a camera culls the scene.
OnPreRender	OnPreRender is called before a camera starts rendering the scene.
OnPostRender	OnPostRender is called after a camera finished rendering the scene.
OnRenderObject	OnRenderObject is called after camera has rendered the scene.
OnWillRenderObject	OnWillRenderObject is called once for each camera if the object is visible.
OnGUI	OnGUI is called for rendering and handling GUI events.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image
OnDrawGizmosSelected	Implement this OnDrawGizmosSelected if you want to draw gizmos only if the object is selected.
OnDrawGizmos	Implement this OnDrawGizmos if you want to draw gizmos that are also pickable and always drawn.
OnApplicationPause	Sent to all game objects when the player pauses.
OnApplicationFocus	Sent to all game objects when the player gets or looses focus.
OnApplicationQuit	Sent to all game objects before the application is quit.
OnPlayerConnected	Called on the server whenever a new player has successfully connected.
OnServerInitialized	Called on the server whenever a Network.InitializeServer was invoked and has completed.
OnConnectedToServer	Called on the client when you have successfully connected to a server
OnPlayerDisconnected	Called on the server whenever a player disconnected from the server.
OnDisconnectedFromServer	Called on the client when the connection was lost or you disconnected from the server.
OnFailedToConnect	Called on the client when a connection attempt fails for some reason.
OnFailedToConnectToMasterServer	Called on clients or servers when there is a problem connecting to the MasterServer.
OnMasterServerEvent	Called on clients or servers when reporting events from the MasterServer.
OnNetworkInstantiate	Called on objects which have been network instantiated with Network.Instantiate
OnSerializeNetworkView	Used to customize synchronization of variables in a script watched by a network view.

Camera: Es el componente que permite controlar lo que se visualiza en el modo juego. Es posible tener varias cámaras en una misma escena, se pueden mezclar imágenes o incluso dividir la pantalla para mostrar "ventanas" de distintas cámaras.



Unity's flexible Camera object

Variables

fieldOfView	The field of view of the camera in degrees.
nearClipPlane	The near clipping plane distance.
farClipPlane	The far clipping plane distance.
renderingPath	Rendering path.
actualRenderingPath	Actually used rendering path (Read Only).
orthographicSize	Camera's half-size when in orthographic mode.
orthographic	Is the camera orthographic (*true*) or perspective (*false*)?
depth	Camera's depth in the camera rendering order.
aspect	The aspect ratio (width divided by height).
cullingMask	This is used to render parts of the scene selectively.
backgroundColor	The color with which the screen will be cleared.
rect	Where on the screen is the camera rendered in normalized coordinates.
pixelRect	Where on the screen is the camera rendered in pixel coordinates.
targetTexture	Destination render texture **(Unity Pro only)**.
pixelWidth	How wide is the camera in pixels (Read Only).
pixelHeight	How tall is the camera in pixels (Read Only).
cameraToWorldMatrix	Matrix that transforms from camera space to world space (Read Only).
worldToCameraMatrix	Matrix that transforms from world to camera space.
projectionMatrix	Set a custom projection matrix.
velocity	Get the world-space speed of the camera (Read Only).
clearFlags	How the camera clears the background.
layerCullDistances	Per-layer culling distances.
depthTextureMode	How and if camera generates a depth texture.

Functions

ResetWorldToCameraMatrix	Make the rendering position reflect the camera's position in the scene.
ResetProjectionMatrix	Make the projection reflect normal camera's parameters.
ResetAspect	Revert the aspect ratio to the screen's aspect ratio.
WorldToScreenPoint	Transforms position from world space into screen space.
WorldToViewportPoint	Transforms position from world space into viewport space.
ViewportToWorldPoint	Transforms position from viewport space into world space.
ScreenToWorldPoint	Transforms position from screen space into world space.
ScreenToViewportPoint	Transforms position from screen space into viewport space.
ViewportToScreenPoint	Transforms position from viewport space into screen space.
ViewportPointToRay	Returns a ray going from camera through a viewport point.
ScreenPointToRay	Returns a ray going from camera through a screen point.
Render	Render the camera manually.
RenderWithShader	Render the camera with shader replacement.
SetReplacementShader	Make the camera render with shader replacement.
ResetReplacementShader	Remove shader replacement from camera.
RenderToCubemap	Render into a cubemap from this camera.
CopyFrom	Makes this camera's settings match other camera.

Messages Sent

OnPreCull	OnPreCull is called before a camera culls the scene.
OnPreRender	OnPreRender is called before a camera starts rendering the scene.
OnPostRender	OnPostRender is called after a camera has finished rendering the scene.
OnRenderImage	OnRenderImage is called after all rendering is complete to render image
OnRenderObject	OnRenderObject is called after camera has rendered the scene.
OnWillRenderObject	OnWillRenderObject is called once for each camera if the object is visible.

Class Variables

main	The first enabled camera tagged "MainCamera" (Read Only).
current	The camera we are currently rendering with, for low-level render control only (Read Only).
allCameras	Returns all enabled cameras in the scene.

MOTOR FÍSICO Y CARACTERÍSTICAS.

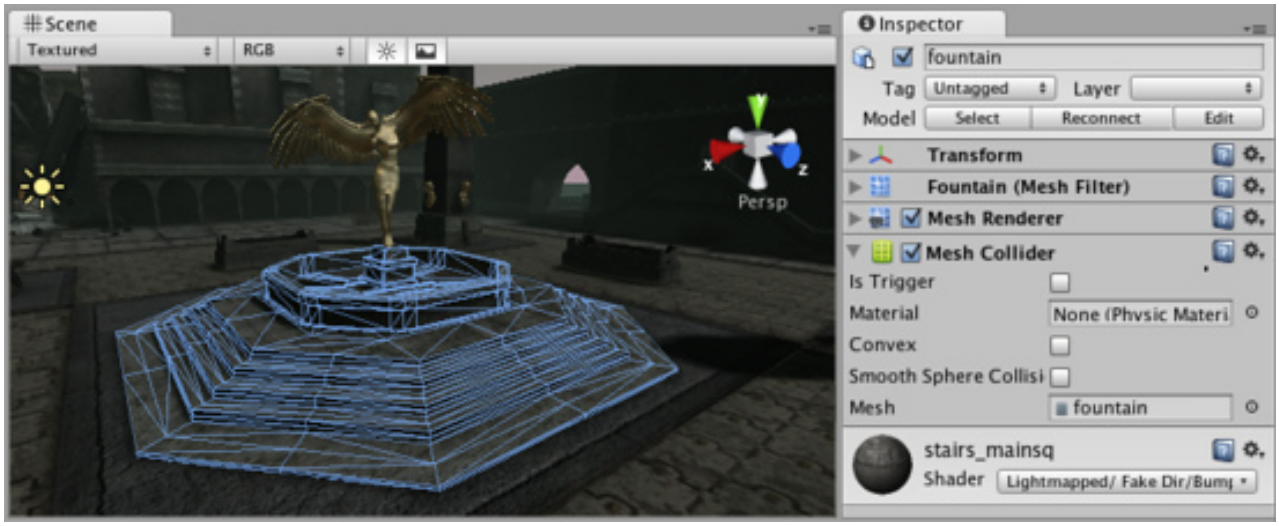
Unity incorpora un simulador de fuerzas físicas del mundo real, el Motor Físico de NVIDIA PhysX. Se basa principalmente en dos de los componentes definidos anteriormente, el RigidBody y el Collider, todo ello combinado con la utilización de materiales físicos.

Gracias al RigidBody es posible aplicar fuerzas y rotaciones a GameObects y dejar que el Motor Físico se encargue de moverlos apropiadamente.

El Collider permite detectar colisiones y aplicar las oportunas reacciones a los objetos en movimiento.

Los materiales físicos determinan el movimiento de los GameObjects. Es posible emular materiales reales como la madera, el hierro o el hielo, configurando propiedades del material físico tales como la fricción, elasticidad y la forma de comportarse frente a otros materiales físicos.

Unity realiza todos los cálculos físicos de forma independiente del frameRate y con una frecuencia constante, lo que simplifica la programación de acciones y garantiza comportamientos similares sobre dispositivos distintos.



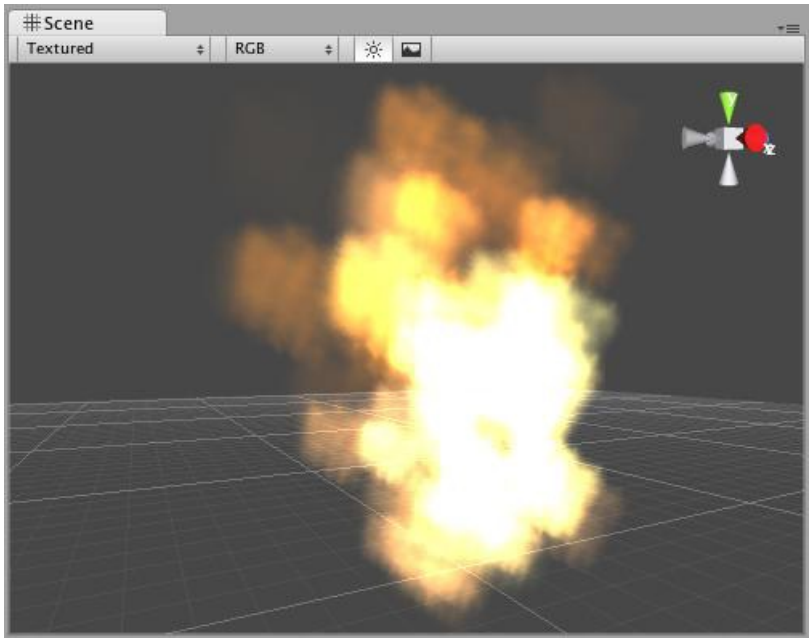
SISTEMAS DE PARTÍCULAS

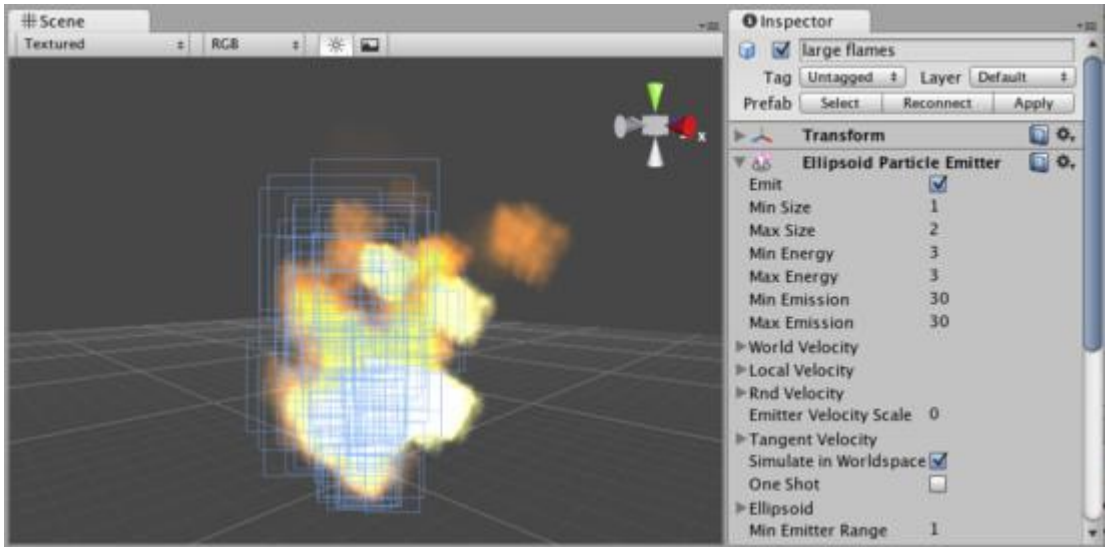
Las partículas son esencialmente imágenes 2D renderizadas en un espacio 3D. Se usan principalmente para obtener efectos como humo, fuego, gotas de agua, hojas o vapor, entre otros. Un sistema de partículas está formado por tres componentes separados: emisor de partículas, animador de partículas y renderizador de partículas.

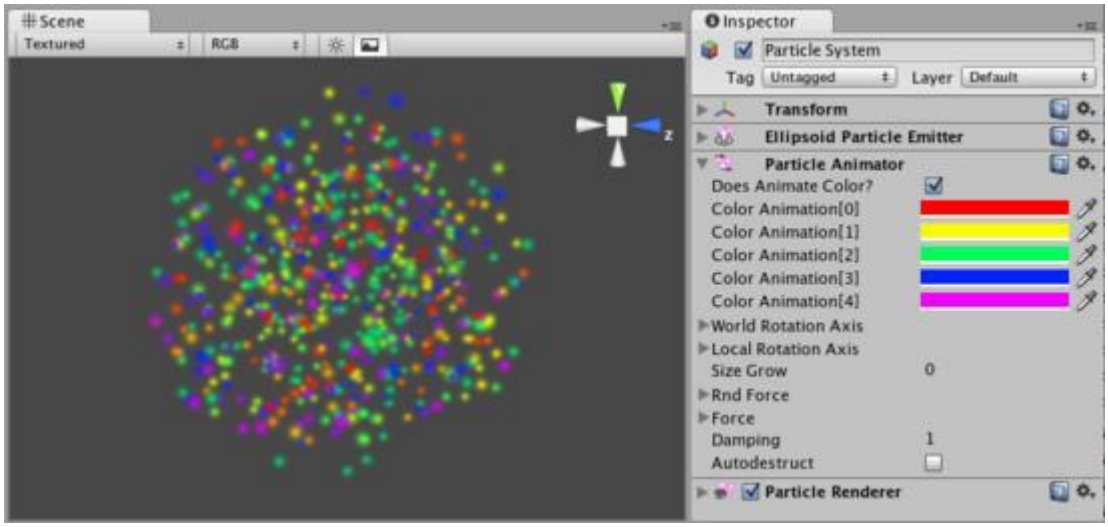
El emisor de partículas genera las partículas, el animador de partículas las mueve a lo largo del tiempo y el renderizador de partículas las dibuja en pantalla.

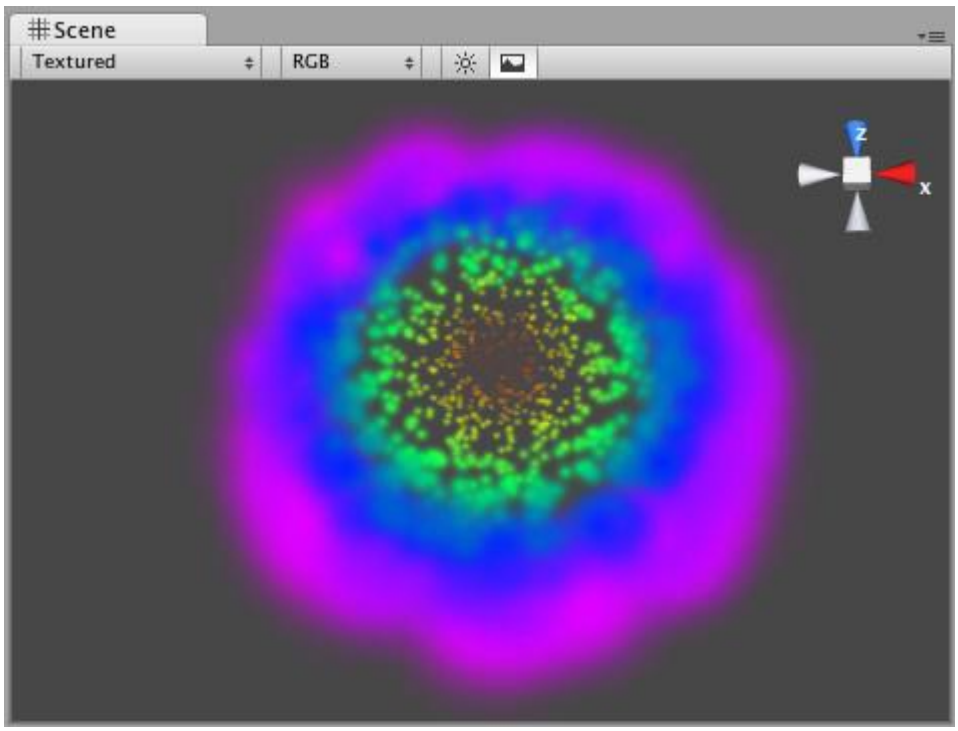
Se puede utilizar un emisor de partículas y un renderizador juntos para obtener partículas estáticas. El animador de partículas moverá las partículas en diferentes direcciones, pudiendo además cambiarlas de color. También se tiene acceso a cada partícula de un sistema de partículas vía scripting, pudiendo crear comportamientos únicos de esta manera.

Los sistemas de partículas funcionan usando una o dos texturas y dibujándolas muchas veces, creando un efecto caótico.
Se puede lograr que las partículas interaccionen con el entorno añadiendo un componente colisionador de partículas al GameObject.









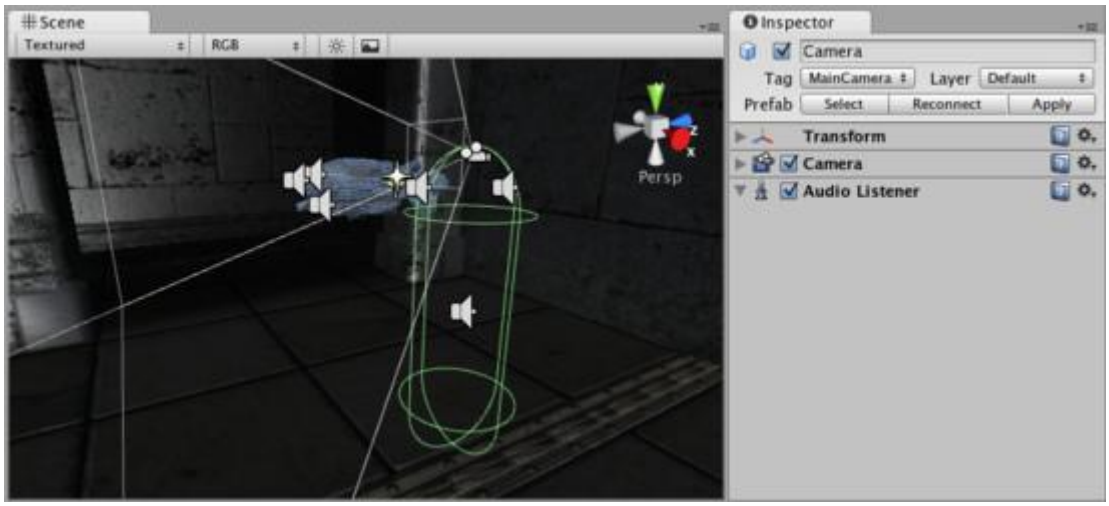
SONIDO - AUDIO LISTENER

El Audio Listener actúa como un micrófono. Recibe entrada desde cualquier

Audio Source en la escena y reproduce sonidos a través de los altavoces del dispositivo. Para la mayoría de aplicaciones lo más lógico es adjuntarlo a la cámara principal. Si el Audio Listener está en los límites de una Reverb Zone, se aplica reverberación a todos los sonidos audibles de la escena. Además, se pueden aplicar efectos de sonido al Listener y serán aplicados en todos los sonidos.

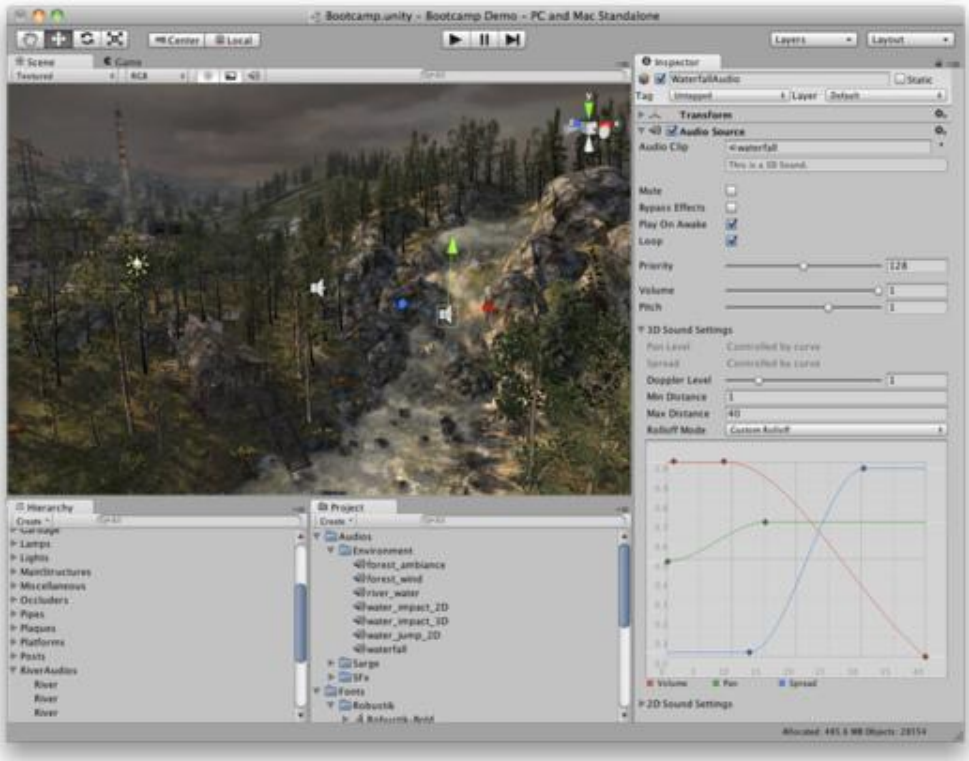
El Audio Listener trabaja en conjunto con los Audio Sources, permitiendo crear la experiencia auditiva de los juegos. Cuando el Audio Listener es adjuntado a un GameObject en la escena, cualquier Source que esté lo suficientemente cerca del

Listener será reproducido mediante los altavoces del sistema. Cada escena debe tener un único Audio Listener para que funcione correctamente.



SONIDO - AUDIO SOURCE

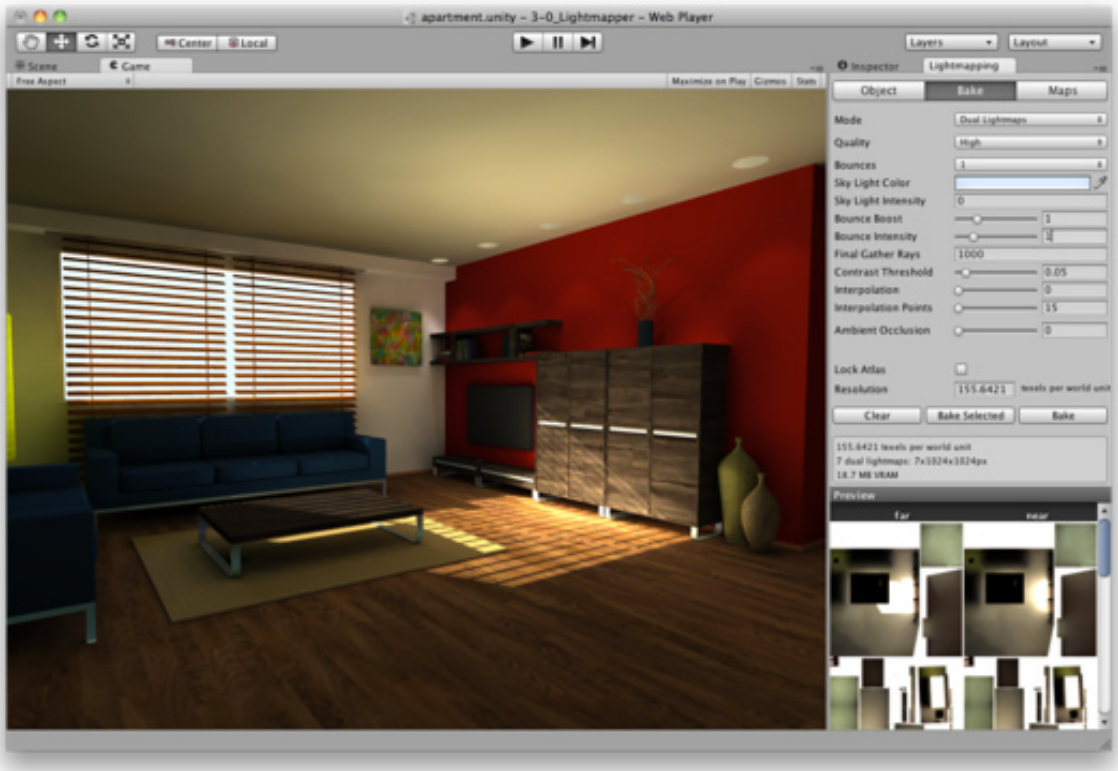
El Audio Source reproduce un Audio Clip en la escena. Si el Audio Clip es un clip 3D, la fuente es reproducida en la posición indicada y será atenuada conforme a la distancia.



OTRAS CARACTERÍSTICAS AVANZADAS.

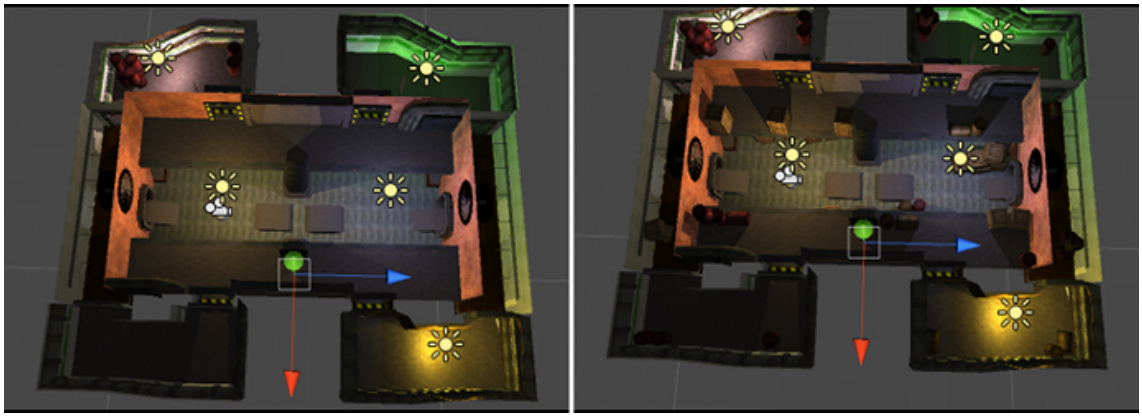
LIGHTMAPPING

Unity tiene pleno soporte para lightmapping, así que permite diseñar un nivel entero desde el editor, añadir luces y los materiales cogerán la luz necesaria automáticamente. Lightmapping en Unity significa que todas la propiedades de las luces serán mapeadas directamente al Beast lightmapper e integradas en las texturas para un rendimiento óptimo.



OCCLUSION CULLING

Occlusion culling es una característica que permite deshabilitar el renderizado de objetos cuando no son visibles por la cámara, porque están siendo tapados por otros objetos. Esto no ocurre automáticamente en los gráficos 3D por computador debido a que los objetos más lejanos a la cámara son dibujados antes que los más cercanos a ella. Occlusion Culling es diferente a Frustum Culling. Esta última técnica sólo deshabilita los renderers para los objetos que están fuera del área visible por la cámara, pero no deshabilita nada que esté siendo tapado por otros objetos dibujados encima. Cuando se utiliza Occlusion culling, se combina con las ventajas del Frustum Culling.



SCRIPTING

MONOBEHAVIOR

MonoBehavior es la clase base de la que todo script deriva. De hecho, cuando utilizamos JavaScript, cada script deriva automáticamente de MonoBehaviour. En C# O Boo se deberá derivar explícitamente.

MÉTODOS PRINCIPALES

Dentro de un script cuya clase extiende a MonoBehavior, existen ciertos métodos principales que son utilizados frecuentemente y a los que debemos prestar cierta atención.

Awake: Este método es llamado al crear el objeto que contiene el script asignado. Es llamado en todos los objetos en escena antes de llamar a cualquier otro método.

Start: Se ejecuta antes de la primera llamada a cualquier Update o FixedUpdate.

Update: Este método es llamado antes de renderizar un frame. Aquí es donde se sitúa la mayoría del código de comportamiento, excepto los cálculos físicos.

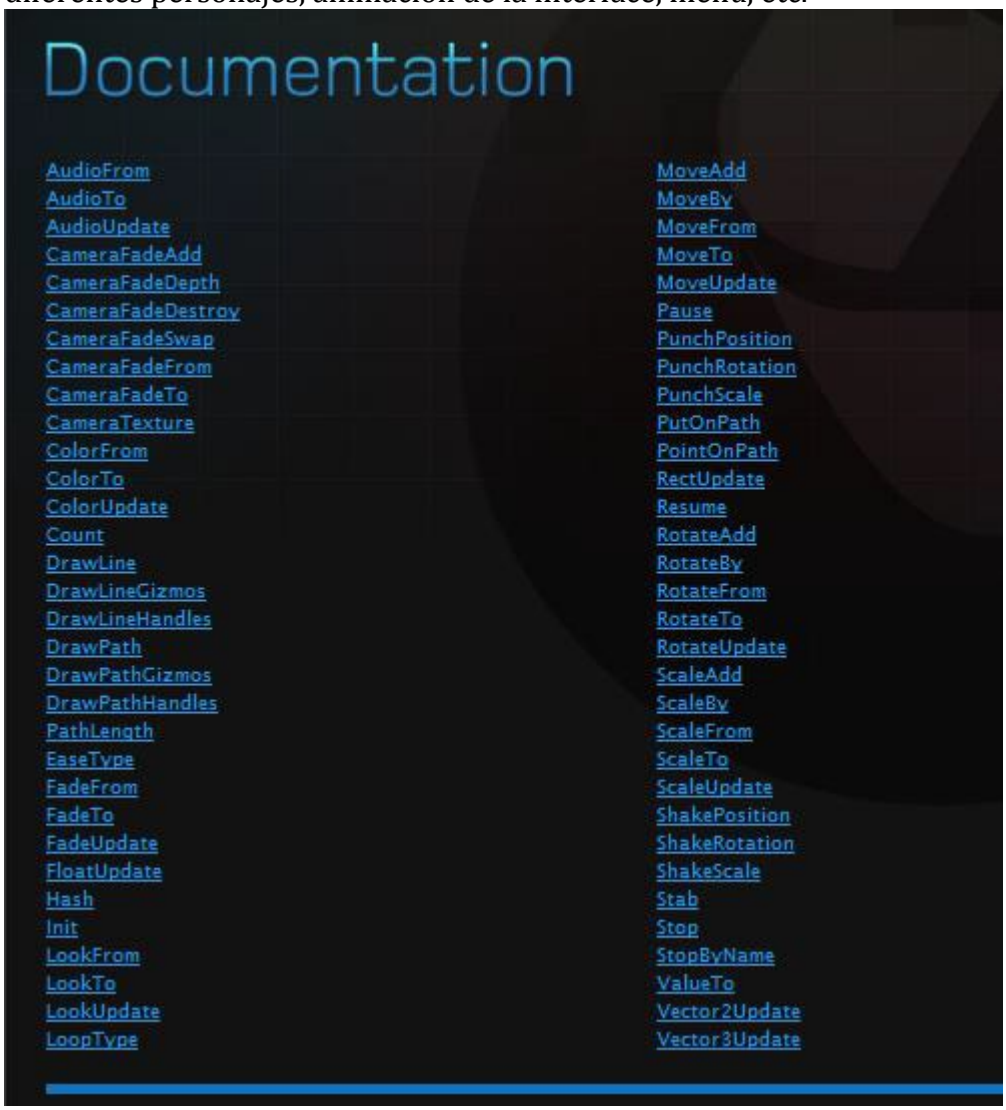
FixedUpdate: Esta función se llama en cada paso del motor físico. Aquí es donde haremos todos los cálculos relacionados con la simulación física.

PLUGINS UTILIZADOS PARA LA REALIZACIÓN DEL JUEGO.

ITWEEN

Este plugin se compone únicamente de un script que alberga una clase con una gran cantidad de métodos estáticos.
La funcionalidad de estos métodos es muy variada y sirve para implementar

de forma fácil y legible multitud de efectos de movimiento de objetos, fade de cámara, audio, etc.
En nuestro caso hemos hecho uso de muchos estos métodos para animar los diferentes personajes, animación de la interface, menú, etc.



NGUI

Ngui es un sistema para la realización de interfaces de usuario de gran alcance y que cuenta con un código limpio y un enfoque simple. Permite la creación de Atlas con diferentes texturas para posteriormente utilizar en el diseño. Bien también con un amplio conjunto de efectos de animación ya programados y fáciles de utilizar ser pueden realizar interfaces profesionales en relativo poco tiempo. Funciona muy bien importando las texturas de photoshop.

Algunas características más:
Integración completa en el editor no es necesario reproducir la scene para ver cómo queda el diseño. Lo que se ve en la vista de escena es lo que hay en la vista del juego.
Soporte completo para iOS / Android y Flash.

Flexible sistema de evento.
Hacer interfaces de usuario complejas que tienen sólo 1 draw call.
Soporte para paneles cortados con bordes.
Limpio, corto, simple y optimizado en código C #.

TESTEO DE USUARIO

A lo largo de todo este documento se ha remarcado en numerosas ocasiones, la importancia de asegurarnos, que el material que se ha generado para la aplicación es totalmente adecuado. En el caso de no ser así, nos arriesgamos a que nuestro publico final sea incapaz de hacer uso de la app porque no se adapta a sus necesidades.

Es por esa razón que la tarea de testing en un proyecto de estas características es totalmente necesaria. Por ello desde el momento en que empezó a generarse contenido de desarrollo, empezaron las pruebas en grupos reducidos, para detectar posibles fallos con el tiempo suficiente para poder reaccionar.
Las pruebas se han llevado a cabo, durante y después de la finalización del proyecto en tres escuelas Noruegas, y una Española.
Uno del testeo más extenso se realizó en la escuela española, en un grupo de cuatro niños de cinco años con distintos tipos de discapacidad, durante un periodo de cinco meses.

Los niños padecían diversos trastornos como, trastorno del espectro autista ASD, déficit de atención, Hiperactividad (TDAH) con retraso mental y trastorno generalizado del desarrollo (PDD).
Por su parte, en las pruebas realizadas en las escuelas noruegas, se trataron sobre todo dificultades de aprendizaje y problemas de concentración, siendo trastornos menos graves.

Métodos similares fueron utilizados en ambos países: Cada niño era seguido por un investigador, que estaba senado al lado e iba observando las actividades del niño, y su comportamiento durante el tiempo de juego.

Se detectaron diversos problemas:

- Dificultades al apilar varios objetos en grupos.
- Las dificultades cuando se trata de seleccionar los objetos más pequeños, por lo que se tuvo que cambiar el tamaño.
- Problemas al intentar realizar actividades que dependen de movimiento de toda la tableta.

  Todos estos temas se abordaron, y se intentó encontrar la mejor solución a estas problemáticas, que se resolvieron principalmente mediante la optimización de la mecánica del juego y los ajustes de tamaños y velocidades.

Evidentemente el grado de autonomía del alumno dentro del juego, dependerá también en gran parte del grado de minusvalía que posea.
En trastornos más graves, tienen que ser acompañados por el profesor. Sin embargo, incluso estos alumnos pueden tener períodos relativamente largos

donde juegan solos, y el profesor puede encontrar tiempo para ayudar a otros alumnos.
A pesar de ello, los resultados de las pruebas resultaban muy prometedores, y los niños quedaron fascinados por el juego.

REPERCUSIÓN MEDIÁTICA Y APLICACIÓN

Tella se publicó a finales de Septiembre de 2013, en los dos markets principales del mercado de las aplicaciones móviles, como lo son, la Google Play para aplicaciones Android, y la AppStore para aplicaciones IOS.

Desde entonces, se han conseguido el número de descargas que se muestra en las imágenes siguiente separadas en las dos plataformas donde se distribuye el juego:

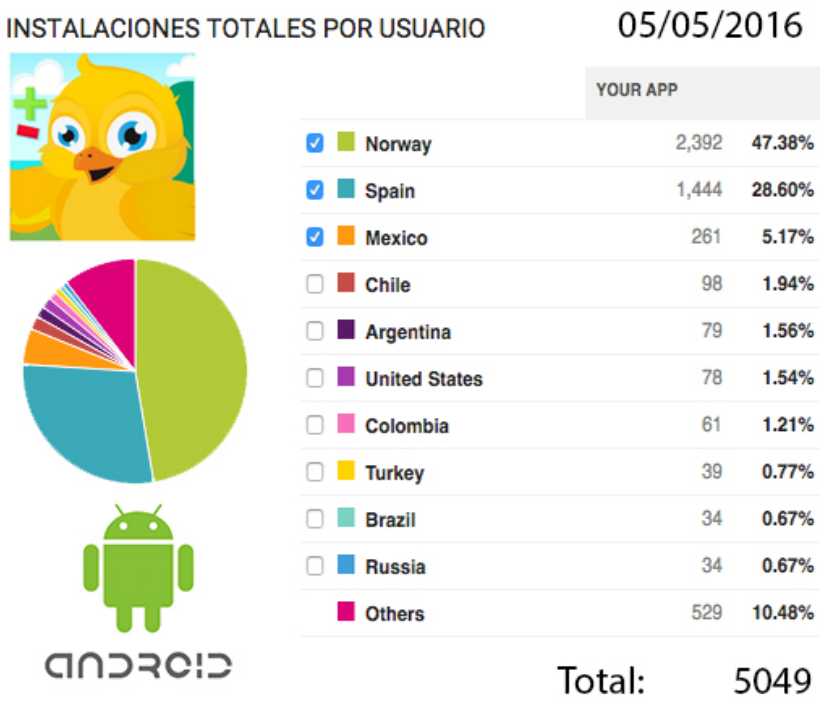
	Territorio	Unidades
1	Europa	31.9K
2	América Latina y el Caribe	166
3	EE. UU. y Canadá	98
4	Asia-Pacífico	27
5	África, Oriente Medio e India	7

Total: 32198

INSTALACIONES TOTALES POR USUARIO

iOS

05/05/2016



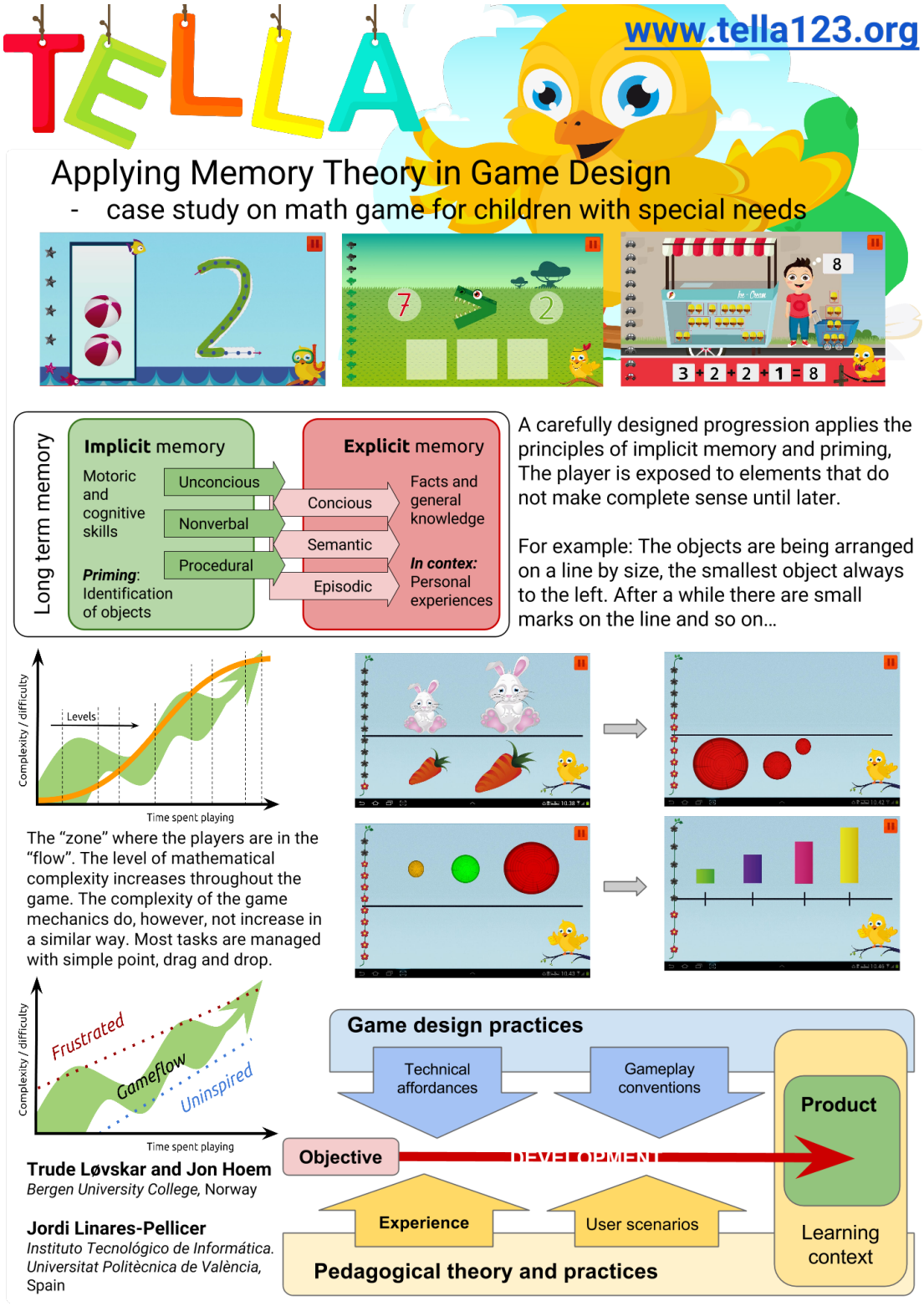
Aunque no se ha llevado a cabo una estrategia de comunicación, se han publicado artículos que hablan sobre nuestro proyecto en blogs educativos y prensa tanto escrita como digital.

En los anexos se adjuntan algunos de los recortes de periódicos donde ha salido la noticia.

APLICACIÓN

- Actualmente, TELLA tiene una aplicación real.
  Tenemos constancia de que se está utilizando como método de aprendizaje complementario a los tradicionales, en distintos centros educativos y de educación especial en Noruega, y en distintos centros y asociaciones de discapacitados en España, como APSA (la asociación de discapacitados más grande de la provincia de Alicante).
  También está teniendo una aplicación bastante numerosa en el uso doméstico. Aunque los padres a veces se muestran reticentes a que sus hijos usen dispositivos móviles, que TELLA sea una aplicación con contenido educativo testado y abalado por estudios pedagógicos, les transmite confianza y les parece una buena manera de que sus hijos aprendan jugando.

Tella participo el European Congress on Game Based Learning (ECGBL 2015), que se celebró en Octubre de 2015 en Noruega. A el asistieron tanto miembros del equipo pedagógico, como miembros del equipo de desarrollo de la aplicación. Una vez allí realizaron una ponencia, cuyo contenido se resume en el cartel que se adjunta a continuación, y se que presento a TELLA como parte del concurso en la categoría de mejor juego educativo 2015.



PRÓXIMOS OBJETIVOS A CORTO PLAZO

Existen en el mercado pocas aplicaciones que se vayan dirigido a colectivos de niños con necesidades especiales, que tengan un estudio pedagógico y psicológico detrás que las abale.

Es por esa razón que estamos seguros de que podría ser una digna candidata en algunos concursos de nuevos métodos de aprendizaje o aplicaciones educativas. Es por ello que ya hay programadas varias ponencias en congresos de este tipo.

A MEDIO-LARGO PLAZO

En Julio del año pasado, se presentó un proyecto a la Dirección General de Educación del Gobierno Noruego, que contemplaba la posibilidad de hacer una adaptación de TELLA para pizarras digitales y ampliación de nuevos niveles.

Las pizarras digitales, son ya una realidad en todas las aulas de muchos países europeos, y la progresión es que su presencia en los centros educativos vaya aumentando progresivamente.

Es por ello, que una versión de esta aplicación para este tipo de dispositivos, sería un recurso muy positivo para el profesor, y fomentaría el uso colectivo de la misma, lo que resulta muy interesante para el desarrollo de las habilidades sociales de los niños.

CONCLUSIONES

Tras alrededor de 2 años de trabajo en este juego, la experiencia para mí ha sido inmejorable, pues considero que en todo este tiempo se me ha blindado la oportunidad de trabajar a la vez de seguir aprendiendo en un campo como el de desarrollo de juegos y aplicaciones para dispositivos móviles que me apasiona y del cual he adquirido una amplia experiencia como programador.

A lo largo de este tiempo se han presentado problemas inesperados por el desconocimiento y la falta de experiencia, los cuales tanto yo como mis compañeros hemos sido capaces de afrontar de manera muy positiva y satisfactoria logrando en todo momento hacer frente a dichos hándicaps. De los errores se aprende y eso hemos hecho.

Pienso que se han cumplido las expectativas con respecto a el funcionamiento del juego y a su jugabilidad y tengo la impresión que tanto los organismos implicados del gobierno de Noruego como la EPSA estarán contentos del resultado obtenido.

Por mi parte no podría estar más contento pues pienso que hemos hecho un buen trabajo y que este estará ahí disponible para recordárnoslo. Además, el saber que algo que hemos realizado esta siendo utilizado como herramienta educativa para niños y sobre todo, niños con déficit de atención, es algo que nos enorgullece y nos auto realiza aún más, tanto en lo profesional como en lo personal.

Considero a este proyecto como mi inicio en el mundo de desarrollo de juegos y gracias a él he descubierto un sinfín de posibilidades tanto en materia de desarrollo como en creatividad, que este mundo de los juego ofrece.

Veo el futuro de manera muy positiva y contento por las posibilidades futuras que sé que me esperan.

-Página oficial del proyecto. “Dirección de Educación de Noruega”,[en línea]. Noviembre 2014. Disponible en la Web:
http://tella123.org/

-Página oficial de Unity. “Unity Technologies”,[en línea]. Disponible en la Web:
http://www.unity3d.com/

-Programa Redes – “Cómo nos influyen los videojuegos”,[en línea]. Octubre 2012. Disponible en la web:
http://www.rtve.es/alacarta/videos/redes/redes-como-influyen-videojuegos/1557690/

-Libro Serius Games – “De Joan Morales y Moras” ,[ebook]. 2015. Disponible en la Web:
http://www.casadellibro.com/libro-serious-games/9788490646977/2572075

-Articulo visto en blogs.lainformacion.com. “Autor: Zoomboomcrash”,[en línea]. Enero 2015.
http://blogs.lainformacion.com/zoomboomcrash/2015/01/12/las-cifras-que-mueven-los-videojuegos-hacen-temblar-a-la-industria-del-cine/