

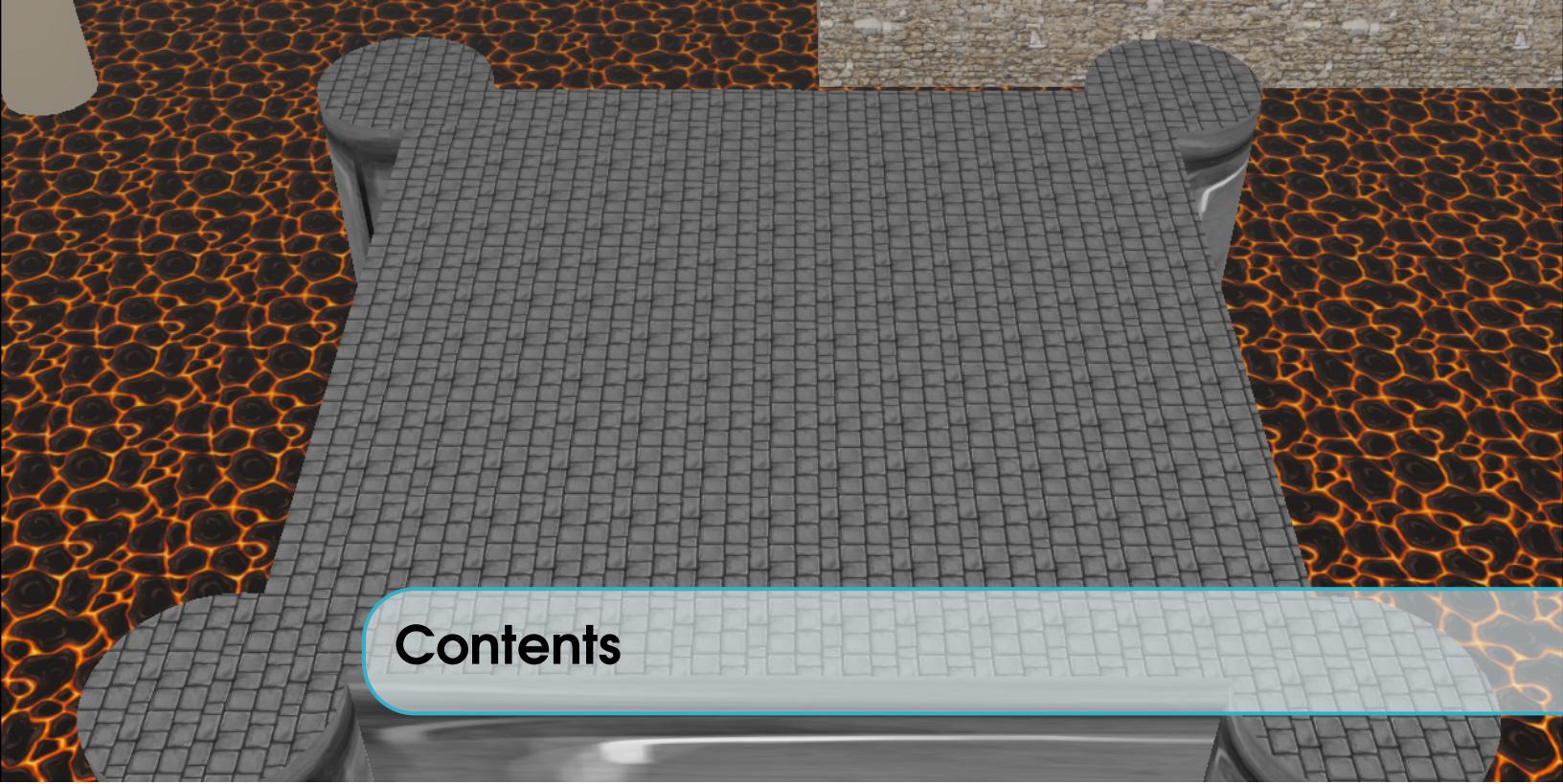
LEGO Duels

Alcoceba Álvarez, Diego
Palestino Infante, Melissa
Rodríguez Pérez, Pablo
Santamaría Martín, Adrián



DUELS

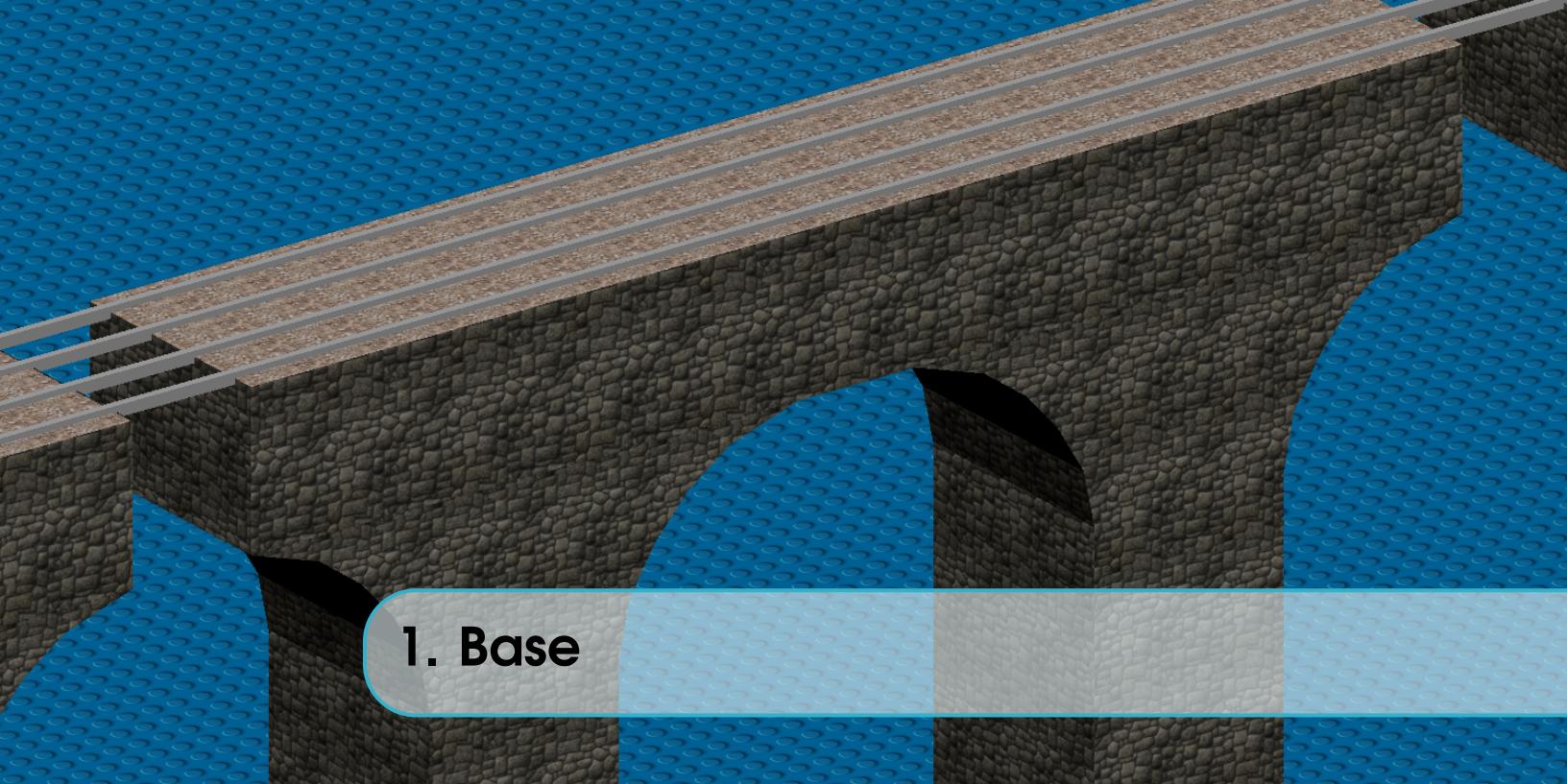




Contents

1	Base	1
1.1	Resumen	1
1.2	Mecánica del juego	1
1.3	Cámara	1
1.4	Controles	2
1.5	Mundo del juego	2
1.6	Niveles	2
1.7	Personaje principal	2
1.8	NPCs y/o Enemigos	2
2	Detalles	3
2.1	Interfaz dentro del juego	3
2.2	Menús y/u otras pantallas auxiliares	3
2.3	Armas y/o equipables	3
2.4	Objetos e ítems	4
2.5	Guión	4
2.6	Logros y progreso	4
2.7	Técnicas	4

3	Anexos	7
3.1	Lógica y scripts	7
3.2	Arte	16
3.3	Componentes	22



1.1 Resumen

Consiste en un juego de peleas con espadas. El objetivo es quitar toda la vida al otro jugador para ganar el combate.

Se intenta otorgarle un carácter arcade. Para ello se emplea una mecánica de juego sencilla que, en función de los jugadores, pueda derivar en batallas con cierto nivel estratégico o facilitar partidas rápidas y alocadas.

1.2 Mecánica del juego

Los jugadores son puestos en el campo de batalla armados para enfrentarse entre ellos. Se han incorporado elementos adicionales a la batalla en sí misma, como regeneradores de vida y trenes que pasan por uno de los escenarios, añadiendo así dinamismo al duelo.

Asimismo cada jugador dispone de controles de movimiento espacial en el plano XY, y las acciones de atacar o cubrirse. Los personajes tienen armas que les permiten hacerlo (una espada a una mano y un escudo) y una barra de vida sin autoregeneración.

Cada acción proporciona un feedback en forma de efectos de sonido.

1.3 Cámara

Dispone de una cámara global aérea a fin de concederle cierto grado arcade y retro. Es isométrica y se busca que sea dinámica y facilite la visualización de la batalla cuando ambos personajes se aproximan y una visualización más completa del escenario cuando éstos se alejan el uno del otro.

1.4 Controles

Control en menú: Hay una pantalla en la que se puede elegir entre ir a la selección de escenario, ver los controles o ver los créditos. Al elegir la primera, se pasa a otra pantalla en la que se elige el nivel en el que se desea jugar. Después de ésto se pasa a la pantalla de juego. En la pantalla inicial se ha establecido la posibilidad de probar los controles, consiguiendo ver las animaciones, modelado y texturizado de cerca.

Control durante la partida: Cada jugador utiliza 6 botones: 2 para avanzar y retroceder, 2 para rotar sobre sí mismo, 1 para atacar y 1 para defender.

1.5 Mundo del juego

El juego está basado en el mundo de LEGO. Los personajes, los medios de transporte, las calles, los animales y todos los escenarios están diseñados en base a estas características. Si bien se incorporan elementos más realistas en algunos aspectos. El juego enfrenta a dos personajes completamente diferentes que entran al mundo de LEGO para demostrar sus habilidades en batalla. Los escenarios de batalla no se corresponden con la ambientación de los personajes; a fin de concederle variedad visual.

1.6 Niveles

El concepto del videojuego es que cada nivel tenga una temática en particular y en diferentes épocas de la historia por lo que hemos creado un nivel llamado “Medieval” donde el concepto es un estilo medieval con lava al fondo del escenario, es un estilo antiguo y sencillo. El segundo nivel consiste en la época de la primera revolución industrial, mostrando un puente antiguo reconvertido para el paso de trenes, los cuales se basan en los mismos de esta época pero en estilo juguete. El tercer nivel se desarrolla en el espacio, que simula un tiempo más futurista. La música de todos los escenarios es acorde a la temática de cada uno de ellos.

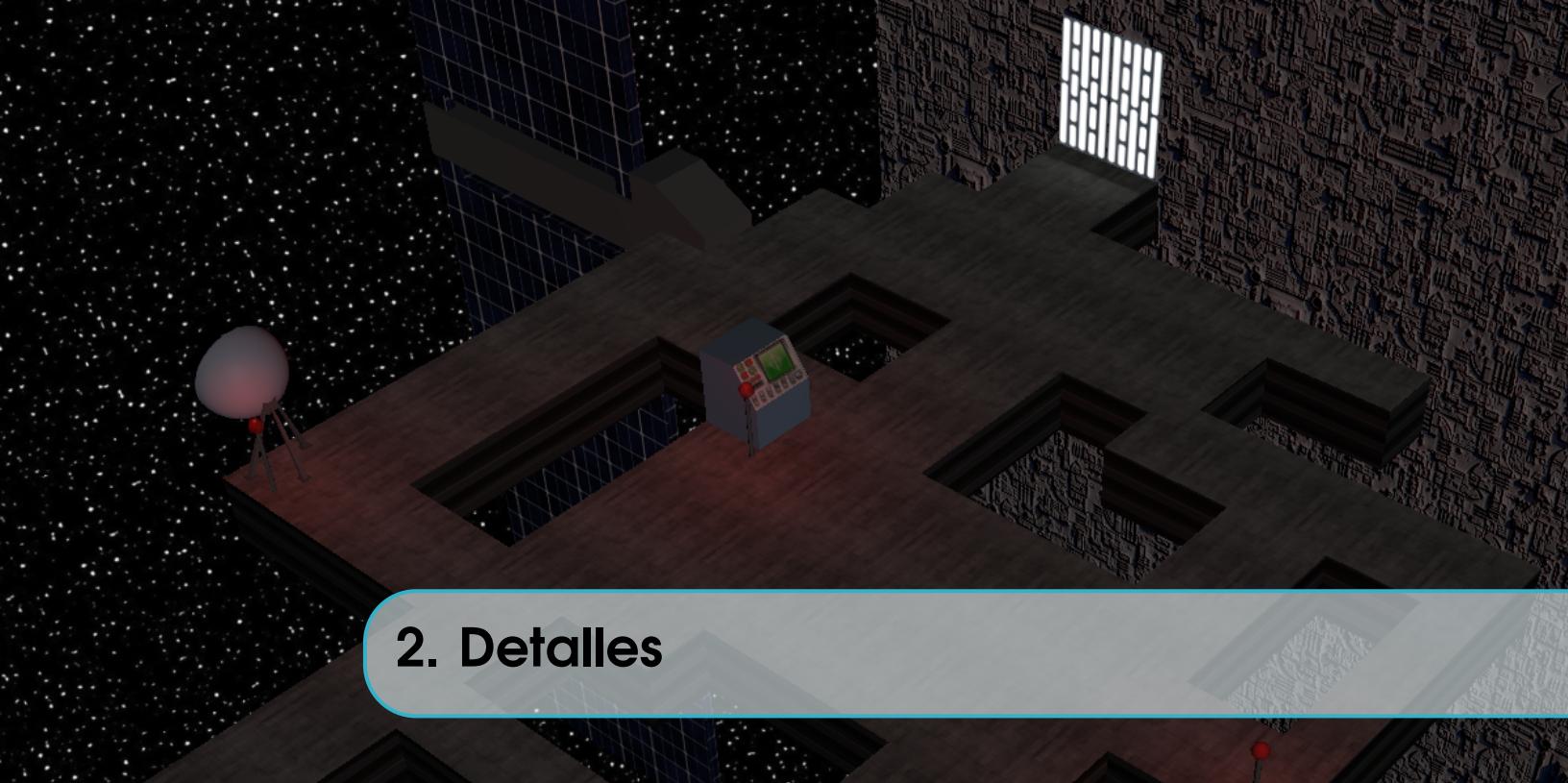
1.7 Personaje principal

Gracias a la estética tipo LEGO se evita que los personajes tengan pretensiones realistas; lo cual evade el posible efecto de disgusto por parte del usuario si considera el modelado/texturizado/animación insuficientemente real, además refuerza la estética arcade -o no real- buscada para el juego. Tenemos animaciones de movimiento, ataque, defensa, victoria y derrota. Así como una adaptación de la animación de victoria para cuando se caen de la plataforma de batalla.

Nuestros personajes son Link del videojuego “The Legend of Zelda” que luchará contra Aragorn, del libro y película “The Lord of the Rings”.

1.8 NPCs y/o Enemigos

El juego está diseñado de forma eminentemente multijugador, por lo que el enemigo es siempre la persona que juegue contigo. No se contemplan enemigos adicionales NPCs, aunque sí trenes ocasionales en “Industrial” y corazones regeneradores de vida en todos los escenarios.



2. Detalles

2.1 Interfaz dentro del juego

Los jugadores tienen acceso en pantalla al valor de la vida de ambos personajes. No se han incluido más datos con el objetivo de mantener un funcionamiento sencillo.

2.2 Menús y/u otras pantallas auxiliares

Existe un menú que dispone de múltiples opciones:

- Ver controles y créditos
- Probar personajes en el propio menú
- Elegir escenario de batalla

Se ha incorporado un menú de pausa dentro de las batallas (P), con sus propias instrucciones de uso. Asimismo la pantalla de Victoria es capaz de iniciar la revancha (Enter) y de volver al menú principal (Backspace).

Cabe destacar que desde el menú se disponen de copias de los personajes exactas, excepto porque no se pueden desplazar. Además en la elección del escenario se dispone de una copia de cada uno, donde se puede ver, por ejemplo, la evolución lumínica de Espacial y algún tren en Industrial.

2.3 Armas y/o equipables

Los personajes siempre portarán un elemento de defensa y otro de ataque, escudo y espada respectivamente; los cuales podrán usar en ciertos instantes. A fin de limitar el modo de juego y aumentar su estabilidad, se ha limitado, por ejemplo, atacar mientras están en movimiento. Además un personaje no puede atacar mientras se defiende.

2.4 Objetos e ítems

Sólo se ha incorporado el powerup de regeneración de vida, el cual aparecerá ocasionalmente. Su efecto es inmediato y desaparece del campo de batalla en el instante en que se recoge.

2.5 Guión

El juego trata de una competición de batallas entre personajes del mundo LEGO. La misión de los personajes es luchar en el campo de batalla para ver quien gana cada contienda. A nivel global podemos decir que los personajes van a luchar en un determinado escenario con el único objetivo de derrotar al contrario.

2.6 Logros y progreso

Lamentablemente no se ha dispuesto de tiempo para incorporar un desbloqueo secuencial de los escenarios, lo cual conseguía remarcar la evolución temporal que poseen. Tampoco se ha podido implementar un marcador de victorias.

2.7 Técnicas

Relativas a la creación del personaje y sus objetos:

Siempre se ha trabajado sobre el modificador “Mirror”, pues facilita la simetría de los objetos. Cuando se ha invertido mucho tiempo en una parte de la malla, el modificador asegura su replicación. Ha resultado de utilidad el modificador “Boolean” para la creación de la mano del personaje; aunque un detalle muy importante es colocar la mano en la capa de trabajo y el cilindro restador en otra capa.

En todos los casos se han buscado en Google imágenes guías para poder dimensionar correctamente los modelados, aunque generalmente no se encuentra un front y un right. En el caso del personaje sí se encontró una imagen con ambas posturas, la cual dividí en dos imágenes. Resulta importante indicar que ambas imágenes deben tener las mismas dimensiones, ya que el escalado de Blender puede distorsionar las dimensiones relativas entre ambas. Por lo que se agregaron los píxeles necesarios al right para tener las mismas dimensiones que el front. Asimismo se ha prestado especial atención a la eliminación y fusionado de caras, cuidando la calidad del resultado final.

En ninguno de estos casos se ha incurrido en utilización de material ajeno. Si bien sí se han utilizado algunas imágenes como guía para su modelado.

Resulta interesante destacar que el Constraint “Follow Path” tiene un efecto mejorable mediante una animación de rotación; cuando nos referimos a la iluminación de un Sol, como en Espacial. Además del uso de corrección de orientación de caras, especialmente importante en el pelo de Aragorn, mediante Ctrl+N; ya que al emplear Backface culling interpretaba algunas caras orientadas de forma errónea.

Como técnica de modelado de la plataforma espacial se procedió a acotar toda la plataforma

diseñada en unidades de cubos. Estos cubos son el cubo por defecto que se genera en Blender. Supuso un diseño laborioso pero que guardaba las dimensiones correctamente y facilitó el paso diseño-modelado. Posteriormente se eliminaron todas las caras internas y fusionaron las externas.

Respecto a la iluminación, hay que destacar el escenario Espacial, el cual ha sido minuciosamente trabajado para conseguir diversos efectos lumínicos: oscuridad completa, parcial, etc. Además de la animación por rotación de la lámpara tipo Sol. En principio se ideó este movimiento de luz mediante un “Constraint” de tipo “Follow Path” en el que el Sol rodeaba el escenario; finalmente con el consejo de Enrique Rendón Angulo se cambió por la animación de rotación actual. Los detalles se han cuidado en situación de poca luz, evitando poner demasiados focos puntuales e incorporando iluminación a la puerta, la cual consigue mayor inmersión por parte del jugador en la escena.

En Medieval, también denominado Mazmorra internamente, se han eliminado múltiples focos puntuales dedicados a la iluminación ambiente con el fin de evitar cómputo excesivo por parte del PC. Se ha modificado el carácter shadeless del material correspondiente a la lava, puesto que ésta emite luz en el mundo real y en el juego no queríamos poner focos dedicados a su iluminación, por el motivo del ahorro mencionado.

En Industrial, se ha iluminado de forma sencilla simulando el mediodía. Además se han hecho copias enlazadas de las vías para reducir al máximo la carga generada. De la misma forma con los puentes, ya que el central es el original y los laterales son copias enlazadas. Conviene detallar que el tren dispone de pocas caras, dado su tamaño relativo en el escenario. También se realizaron copias enlazadas de todas las ruedas.

En general no se han empleado modificadores para el modelado de los objetos de escenario, ya que en su mayoría son asimétricos y, cuando no lo eran no representaba un ahorro importante el uso del modificador “Mirror”. En Medieval se ha empleado el modificador “Boolean” para la creación de la plataforma, ya que une un cubo con cuatro cilindros.

Relativas al texturizado:

Todas las texturas se han realizado creando el mapa UV del objeto y exportándolo como una imagen png. Fuera de Blender he utilizado Adobe Illustrator para realizar los dibujos y Adobe Photoshop para pintar con la ayuda de una tableta gráfica. No he usado la herramienta para colorear sobre el modelo 3D dentro de Blender en ningún momento.

Ha sido especialmente complicado conseguir que coincidan los detalles que rodean completamente a los personajes, como el cinturón, las botas o las mangas; y que todas las caras coincidan.

Los logos, tanto de la empresa como del videojuego también han sido creados por nosotros.

Relativas a la animación:

Las animaciones se han hecho siguiendo los tutoriales de youtube de la clase de SAI, para un personaje de malla continua. Se realizaron diferentes esqueletos hasta que se decidió el que mejor fuera con el personaje. Para las restricciones de movimiento se decidió por restricciones IK y se hicieron animaciones de andar, defensa, ataque y inactividad (el cual al final no se tiene en uso) Y para la segunda entrega se implementaron acciones de victoria y derrota. Se buscaron en youtube otros videos sobre animación en combates y también se buscaron videos de LEGO para poder tener una mejor referencia del movimiento del cuerpo del LEGO.

Relativas al sonido:

Para el sonido se decidió que fuera música basada en un estilo de 8bits. Se buscó en Google y en la plataforma de Youtube música libre de derechos de autor con estas características. Ya que nuestro juego es sencillo y cuenta con una temática de batalla, se buscó la música de fondo y los efectos siguieran estas temáticas. Hay una canción para cada uno de los escenarios y se utilizó el programa Logic Pro X para poder editar los archivos de audio, cortarlos y cambiar su tamaño.

Conviene señalar que se ha incluido el efecto “3D Sound” en los trenes. Es un efecto que, si bien puede consumir un poco más de recursos computacionales, es muy deseable a fin de agregar cierto sentido del espacio al feedback que proporciona el sonido. A fin de lograr este hecho, se han modificado los parámetros del actuador de sonido dentro de la lógica del propio tren. Es importante calcular la distancia a la que se tendrá mayor ganancia (nivel o volumen), para conseguir un efecto acertado. Asimismo, se ha elevado notablemente la ganancia mínima buscando el “efecto aviso” que los jugadores pueden interpretar para apartarse del camino del tren. Respecto al sonido de aparición de un tren, se ha incluido dentro de los empty correspondientes con sonidos previamente paneados por Melissa en un editor de audio.

Relativos a la documentación:

Durante el curso hemos escrito los documentos usando Google Docs para poder trabajar todos al mismo tiempo y evitarnos tener que andar enviándonos los textos por correo. El documento de la entrega final ha sido escrito con L^AT_EX.

Lógica y Scripts

En el juego los objetos que incluyen lógica son: Los avatares de los personajes, sus cuerpos, los esqueletos y el propio escenario. También las cámaras, las barras de vida, los menús, los powerups y los trenes. Se detalla la de cada uno a continuación:

Avatar:

(Como ambos personajes tienen un comportamiento similar, sus lógicas son análogas, tanto en los avatares como en los esqueletos y cuerpos)

Utilizados para el movimiento de los personajes por el escenario. Solo funcionan cuando ambos personajes se encuentran con vida (sensor Property):

Acción	Cuándo	Detalles
Avanzar (Actuador Motion)	AND{ <ul style="list-style-type: none"> • Al pulsar la tecla de avance (Sensor Keyboard) • No tiene nada delante (Sensor Radar) • No está en medio de una acción de ataque o defensa (Sensores Property) }	Se utiliza un sensor Radar para evitar que el personaje avance si tiene al otro delante (en un ángulo de 70°, para evitar que se queden bloqueados), evitando así que se puedan empujar el uno al otro. Además, se evita que se mueva al mismo tiempo que se defiende/ataca
Girar a la izquierda (Actuador Motion)	Al pulsar la tecla de giro a la izquierda (Sensor Keyboard)	El personaje rota sobre sí mismo
Retroceder (Actuador Motion)	AND{ <ul style="list-style-type: none"> • Al pulsar la tecla de retroceder (Sensor Keyboard) • No tiene nada detrás (Sensor Radar) • No está en medio de una acción de ataque o defensa (Sensores Property) }	Se utiliza un sensor Radar para evitar que el personaje retroceda si tiene al otro detrás(en un ángulo de 70°, para evitar que se queden bloqueados), evitando así que se puedan empujar el uno al otro. Además, se evita que se mueva al mismo tiempo que se defiende/ataca
Girar a la derecha (Actuador Motion)	Al pulsar la tecla de giro a la derecha (Sensor Keyboard)	El personaje rota sobre sí mismo

Esqueletos:

(Como ambos personajes tienen un comportamiento similar, sus lógicas son análogas, tanto en los avatares como en esqueletos y cuerpos)

Utilizados para las animaciones de los personajes. Restringidas a cuando ambos personajes se encuentran con vida (sensor Property):

Acción	Cuándo	Detalles
Lanzar animación de andar (Actuador Action en modo Loop Stop)	AND{ <ul style="list-style-type: none">• Al pulsar la tecla de avanzar/retroceder (Sensores Keyboard)• No está en medio de una acción de ataque o defensa (Sensores Property)} }	El personaje lanza la animación de andar cuando recibe la orden de desplazarse hacia adelante o hacia atrás, y no está atacando/defendiéndose
Lanzar animación de defensa(Actuador Action en modo Flipper)	AND{ <ul style="list-style-type: none">• Al mantener pulsada la tecla de defenderse (Sensor Keyboard)• No está en medio de una acción de ataque (Sensor Property)• No están pulsadas las teclas de avanzar/retroceder (Sensores Keyboard)} }	El personaje sube su escudo mientras se mantiene pulsada la tecla de defensa. Además, no es posible defenderse mientras está atacando, ni tampoco si se intenta desplazar.
Lanzar animación de ataque (Actuador Action en modo Play)	AND{ <ul style="list-style-type: none">• Al pulsar la tecla de atacar (Sensor Keyboard)• No está en medio de una acción de defensa (Sensor Property)• No están pulsadas las teclas de avanzar/retroceder (Sensores Keyboard)} }	El personaje ataca al pulsar la tecla de ataque. No es posible atacar mientras está defendiéndose, ni tampoco si se intenta desplazar.
Corregir animación de defensa (Actuador Action en modo Play)	Si la Property del fotograma de la animación de defensa se sale del rango (Frames 0-30)	El modo flipper de la animación de defensa puede llegar a fallar en ocasiones cuando se reciben muchas órdenes simultáneas, por lo que para evitar bloqueos si se sale del rango automáticamente la reinicia.

Lanzar animación de victoria/derrota (Actuadores Action en modo Loop End/Play)	Muerte del contrario/propia (sensores Property)	Cuando un personaje pierde se inclina, y el contrario celebra.
Lanzar animación caída	No hay suelo debajo del personaje(sensor Radar hacia abajo)	

Propiedades en el esqueleto: Para cada personaje, para comprobar si se encuentra en medio de una animación de ataque o defensa, se utilizan dos propiedades para guardar el frame de las mismas.

Cuerpos de los personajes:

(Como ambos personajes tienen un comportamiento similar, sus lógicas son análogas, tanto en los avatares como en esqueletos y cuerpos)

Utilizados para las lógica de daño:

Acción	Cuándo	Detalles
Descender vida del personaje (Actuador Property)	AND{ <ul style="list-style-type: none"> Al hacer contacto con espada del oponente (sensor Collision) No está en el periodo de inmunidad ni en guardia (sensores Property) Oponente está vivo (sensor Property) }	Se interpreta que el personaje debe recibir daño cuando la espada del oponente impacta contra su cuerpo, siempre que no esté en guardia o haya recibido daño hace muy poco. También se evita que el personaje reciba daño una vez derrotado su oponente.
Resetear periodo de inmunidad (Actuador Property)	Mismas condiciones que el anterior	Para evitar que de un solo golpe se pudieran hacer varios tics de daño, se establece un pequeño periodo de inmunidad tras ser golpeado durante el cual el personaje es inmune al daño.
Establecer si personaje está en guardia (Actuador Property)	AND{ <ul style="list-style-type: none"> La animación de defensa está en posición con el escudo en alza (sensor Property) Mira hacia el oponente (sensor Radar) }	Si el personaje se encuentra con el escudo en alto, y está mirando hacia el oponente, es inmune al daño.
Mantener vida entre 0 y 100	Si la vida se sale del rango 0-100	Si la vida toma un valor no válido,

(Actuador Property)	(Sensor Property)	se corrige al momento.
Enviar mensaje a cámara (Actuador Message)	Personaje muere (Sensor Property)	Cada personaje envía un mensaje diferente a la cámara para que ésta sepa a quién enfocar
Incrementar contador de inmunidad (Actuador Property)	Always	El contador de inmunidad crece siempre, para saber cuánto tiempo lleva el personaje sin recibir daño, y se resetea cada vez que lo sufre.
Matar al personaje (Actuador Property)	Personaje toca un plano de muerte bajo el escenario (Sensor Collision)	Bajo los escenarios existe un plano invisible para detectar cuando un personaje ha caído por el borde, matándolo.
Curación (Actuador Property)	Personaje toca un corazón (Sensor Message)	Cuando el personaje toca un corazón, éste le envía un mensaje que le notifica que ha de incrementar su vida.

Propiedades que contienen: Vida del personaje, boolean de guardia/no guardia y contador de inmunidad.

Powerups (curación):

Por el escenario hay una serie de powerups que curan a los personajes cuando éstos los tocan. Están siempre presentes, pero sólo permanecen visibles cuando están activos. Su activación/desactivación es aleatoria, y se incorpora un timer global para que no ocurra con demasiada frecuencia. La lógica que incluyen es la siguiente:

Acción	Cuándo	Detalles
Girar (Actuador Motion)	Always	Los corazones se encuentran girando permanentemente
Cambiar visibilidad (Actuadores Visibility)	Cuando el corazón se activa/desactiva (Sensor Property)	Cuando el estado del corazón cambia, se altera su visibilidad.
Enviar mensaje reseteo del timer (Actuador Message)	Cuando el corazón aparece, o cuando desaparece porque lo ha cogido un jugador	Cuando el corazón aparece, se resetea el timer para asegurar que está activo durante un tiempo. Cuando un jugador lo coge, también se resetea para que no aparezca ningún otro durante un pequeño periodo.
Activar/Desactivar corazón	Cuando coinciden un conjunto de	Los corazones se activan y

(actuador Property)	Sensores Random, y el timer de control supera un umbral. También cuando es utilizado por un jugador	desactivan de forma aleatoria, con un timer de control para que no se produzca con demasiada frecuencia. Obviamente también se desactivan cuando son utilizados.
Mandar mensaje de curación (Actuador Message)	Cuando un personaje toca el corazón (Sensores Collision)	Cuando un personaje toca un corazón, se le envía un mensaje para que incremente su vida.

Trenes:

Los trenes tienen un Actuador Motion activado permanentemente mediante un Sensor Always que les hace ir siempre hacia adelante

Lanzadores de los trenes:

Cada 10 segundos, aparece un tren en el escenario de Industrial, siendo controlado por unos Actuadores Random el que sea desde un lado o el otro.

Menús:

Menú Principal:

Acción	Cuándo	Detalles
Ir a pantalla de selección de escenario	Tecla INTRO (sensor Keyboard)	
Ver controles	Tecla C (sensor Keyboard)	Uso un actuador Camera para darle un poco de movimiento a la cámara
Ver créditos	Tecla Y (sensor Keyboard)	Uso un actuador Camera para darle un poco de movimiento a la cámara
Cerrar juego	Tecla ESCAPE (sensor Keyboard)	

Seleccionar Escenario:

Acción	Cuándo	Detalles
Comenzar batalla	Tecla INTRO (sensor Keyboard)	
Cambiar de escenario	Teclas de dirección derecha o izquierda (sensor Keyboard)	Cambia entre escenas de cada escenario
Volver a menú principal	Tecla BACKSPACE (Keyboard)	

Batalla:

Acción	Cuándo	Detalles
Cargar barras de vida	Al cargar escenario de batalla	Se cargan como escenas <i>Overlay</i>
Pausar juego	Tecla P (sensor Keyboard)	Pausa la batalla y permite ir al menú de selección de escenario o volver a la batalla.
Iniciar música (Actuador Sound en modo Loop End)	Siempre, al iniciar el juego (Sensor Always)	Se utiliza el escenario como objeto para lanzar la música, pero se podría utilizar cualquier otro que permaneciera siempre activo en el juego.

Pausa:

Acción	Cuándo	Detalles
Volver a la batalla	Al cargar escenario de batalla	
Salir de la batalla	Tecla L (sensor Keyboard)	
Salir del juego	Tecla ESCAPE (sensor Keyboard)	

Victoria:

Acción	Cuándo	Detalles
Revancha	Tecla ENTER (sensor Keyboard)	Reinicia el escenario actual
Cambiar escenario	Tecla ESCAPE (sensor Keyboard)	Vuelve al menú de elección de escenario

Además, en algunos de los objetos existen actuadores sound utilizados para lanzar los efectos de sonido que corresponden a algunas acciones, como golpear al oponente, moverse por los menús, etc.

Cámara:

Hemos utilizado un script de python proporcionado por Enrique Rendón Angulo para que la cámara mantenga en encuadre a sendos personajes, mediante movimiento en el plano de la cámara y acercar o alejarse; de este modo se consigue visualizar correctamente a los personajes cuando éstos están cerca el uno del otro. El ajuste de la misma ha sido realizado de forma particular a cada escenario, buscando evitar el movimiento continuo de la cámara y que los jugadores dispongan de tiempo de reacción ante el movimiento hacia un precipicio fuera de encuadre. A continuación se observa el script configurado para el escenario

Medieval:

```
# Script de control para una cámara que mantiene encuadrados
# dos jugadores u objetos
# enriqueblender@gmail.com      28 Abril 2017
# Para Lego Duels de Insert Coin
import bge # importar los símbolos de blender game engine
from bge import logic # e importar los símbolos de la lógica
import mathutils

# Funciones auxiliares-----
# Punto en la mitad de la línea que une los orígenes de los dos jugadores
def punto_medio():
    global pers1
    global pers2
    return (pers1.worldPosition + pers2.worldPosition)/2

# Vector que va del punto medio entre los dos jugadores a la cámara
def vector_a_camara(pos_camara):
    return pos_camara - punto_medio()

# Devuelve True si el punto (px,py) está dentro de un borde de bx ancho y by alto
# de un cuadrado unidad
def in_border( px, py, bx, by):
    if ((px < bx) or (px > 1-bx)):
        return True
    if ((py < by) or (py > 1-by)):
        return True
    return False

def init():
    # Este código se ejecuta una vez-----
    global camara
    global pers1
    global pers2
    global vector_uni_a_camara
    global dxalejar
    global dxacercar
    global dyalejar
    global dyacercar

    cont = logic.getCurrentController() # acceder a este controlador python
    escena = logic.getCurrentScene() # escena actual
    camara = cont.owner
    pers1 = escena.objects["AvatarAragorn.004"] # Nombre del objeto avatar jugador 1
    pers2 = escena.objects["AvatarLink.001"] # Nombre del objeto avatar jugador 2

    # Para mover la cámara nos basamos en el punto medio entre los dos jugadores
    # y el vector desde ese punto a la cámara

    # Calculamos los valores iniciales del punto medio y ese vector
    posicion_inicial_camara = camara.worldPosition
    posicion_inicial_punto_medio = punto_medio()
    vector_inicial_a_camara = vector_a_camara(posicion_inicial_camara)

    # El vector hacia la cámara lo obtendremos multiplicando un vector unitario en esa dirección
    # por la longitud de ese vector que llamamos distancia

    # Calculamos la distancia inicial y el vector unitario
    distancia = vector_inicial_a_camara.length
    camara["distancia"] = distancia
    vector_uni_a_camara = vector_inicial_a_camara/distancia # vector unitario de punto medio a cámara

    # Las coordenadas de pantalla están definidas entre 0 y 1 en x e y
```

```
# Se acercará la cámara cuando ambos jugadores estén más hacia el centro
# que de un borde de este ancho:
dxacercar = 0.3
dyacercar = 0.3

# Se alejará la cámara cuando alguno de los jugadores este en el borde exterior definido
# por estos valores:
dxalejar = 0.15 # dxalejar < dxacercar
dyalejar = 0.2 # dyalejar < dyacercar

# Comprobación inicial de si la cámara está fuera de rango y meter en rango
if ( distancia < camara["mind"]):
    distancia = camara["mind"]
    print("Cuidado camara demasiado cerca")

if (distancia > camara["maxd"]):
    distancia = camara["maxd"]
    print("Cuidado camara demasiado lejos")

def main():
    global camara
    distancia = camara["distancia"]

    # Si no está a verdadero la propiedad "activo" salimos, no funciona el script
    if ( not camara["activo"]):
        return

    # Ahora la posición de la cámara la determinamos sumando a el punto medio entre los jugadores
    # la distancia por el vector unitario en dirección a la cámara
    pto_medio = punto_medio()
    camara.worldPosition = pto_medio + vector_uni_a_camara*distancia

    # Ahora decidimos si hay que acercar o alejar la cámara, que tendrá efecto
    # en el siguiente frame
    (x1,y1) = camara.getScreenPosition(pers1)
    (x2,y2) = camara.getScreenPosition(pers2)

    acercar = False
    alejar = False

    # Hay que alejar si uno de los jugadores esta en el borde exterior
    if ( (in_border(x1, y1, dxalejar, dyalejar)) or (in_border(x2, y2, dxalejar, dyalejar)) ):
        alejar = True
    # Hay que acercar si los dos jugadores están más adentro del borde interior
    elif ((not in_border(x1, y1, dxacercar, dyacercar)) and (not in_border(x2, y2, dxacercar, dyacercar))):
        acercar = True

    # El alejar o acercar se hace multiplicando la distancia por un factor
    # y solo si estamos en el rango de distancias definido en las propiedades
    if ((alejar) and (distancia < camara["maxd"])):
        distancia = distancia * 1.02
        camara["distancia"] = distancia
    elif ((acercar) and (distancia > camara["mind"])):
        distancia = distancia * 0.98
        camara["distancia"] = distancia

    return
```

Los parámetros que se deben modificar son donde pone “AvatarAragorn.004” y “AvatarLink.001”, cambiarlos por los nombres de los personajes u objetos que la cámara debe encuadrar. También es recomendable ajustar la cámara en un encuadre inicial deseable, ya que todo el trabajo del script se basa en el encuadre inicial. A fin de perfeccionar el movimiento de la cámara, se pueden modificar los valores de: “dxacercar”, “dyacercar”, “dxalejar” y “dyalejar”.

Este script tiene asociadas unas *properties* que le ayudan a funcionar: “activo”, “maxd”, “mind” y “distancia”. Se recomienda ajustar correctamente “maxd” y “mind”, pues se corresponden con la distancia máxima y mínima que habrá entre la cámara y el punto medio entre los personajes. A continuación se incluye un ejemplo de implementación de la lógica utilizada en la cámara, partiendo de un archivo generado en el editor de Blender con el contenido del script anterior y denominado “camaraMedieval.py”. Se han eliminado los bloques no relevantes para este cometido, quedando visible aquello que se recomienda incluir para el correcto funcionamiento del script.



Además, se ha configurado la cámara para que enfoque al jugador ganador una vez derrote a su oponente.

Barra de vida:

La barra de vida ha sido implementada con los siguientes tutoriales de Enrique ([LINK](#)). Se crean como una “Add Overlay Scene” al cargar el escenario y el valor de la vida de cada personaje se pasa con el script de Python “EscribeVida.py”, ajustado para cada personaje.

Arte

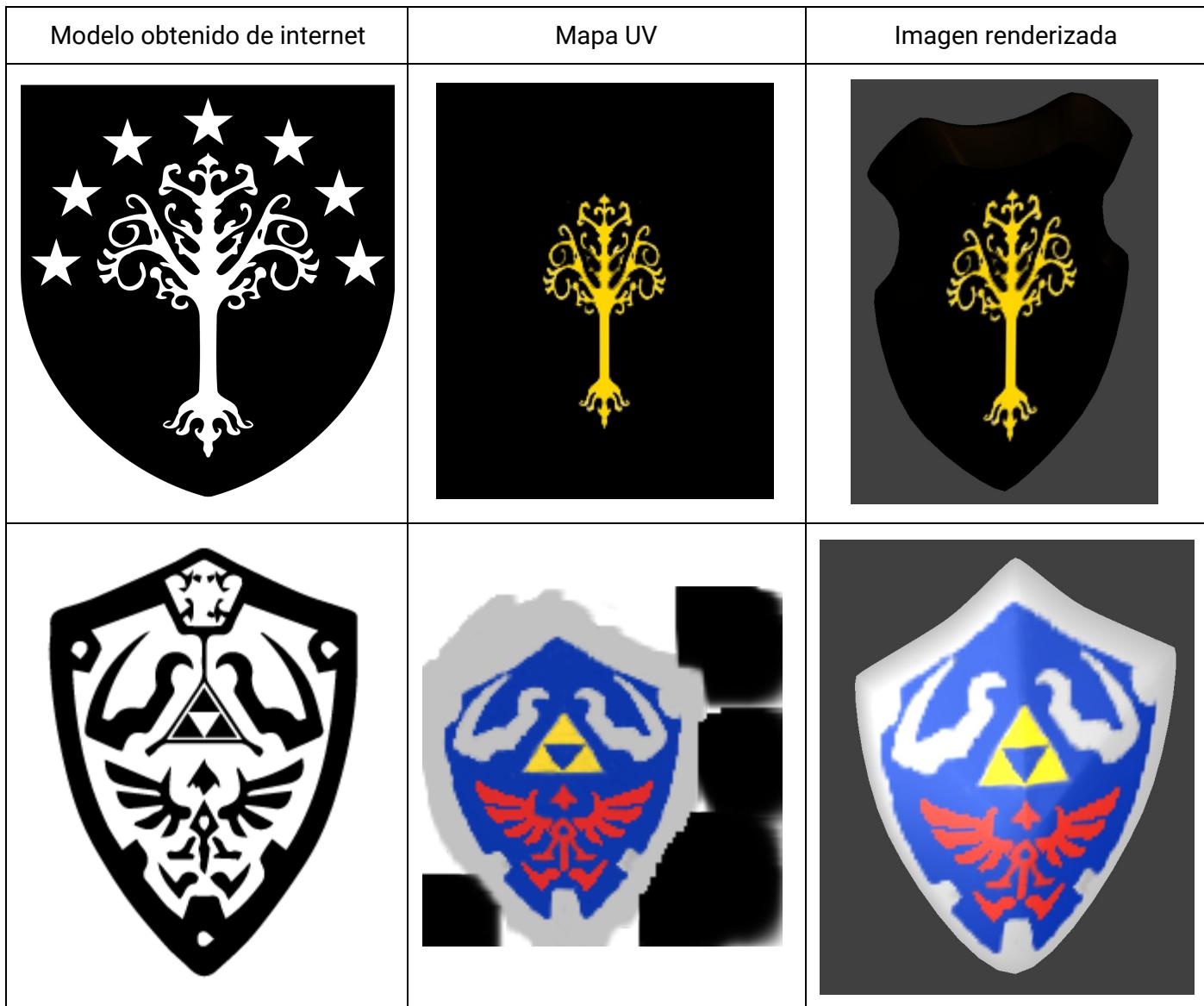
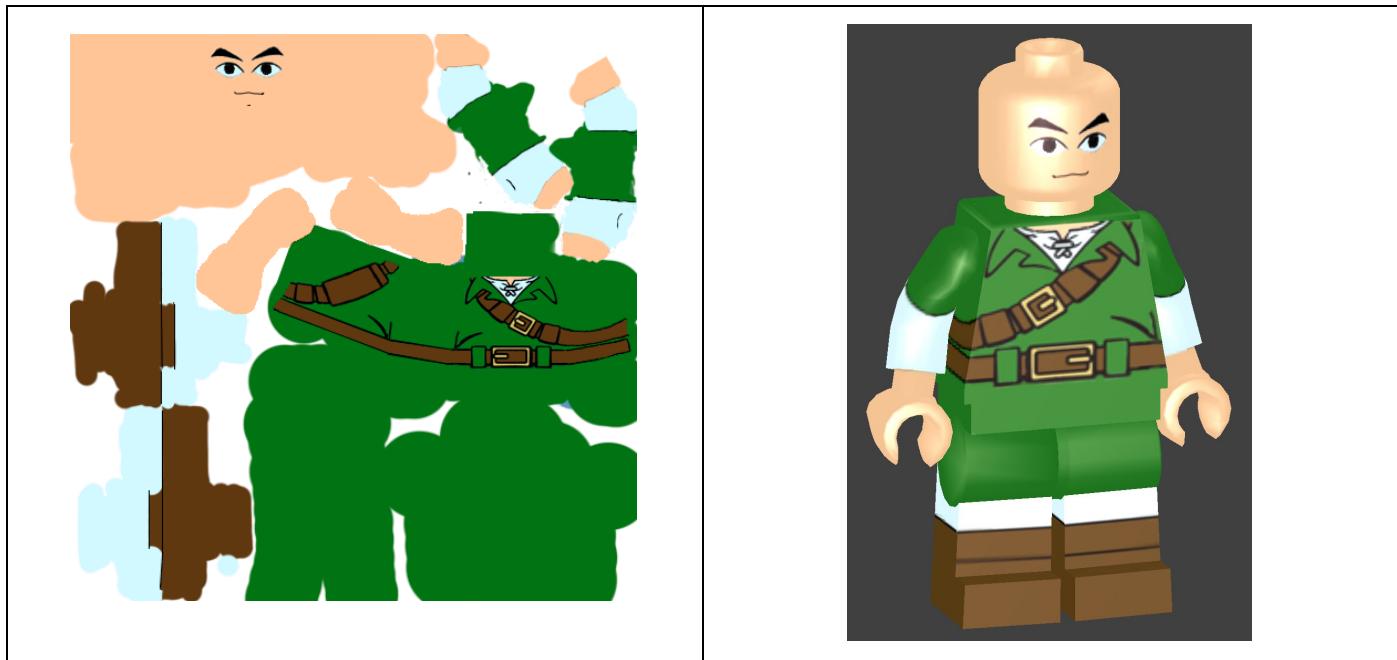
Inicialmente se estableció que deseábamos dos personajes lego con espada y escudo. A las pocas semanas se decidió hacer imitaciones de Aragorn -personaje de *El Señor de los Anillos*- y Link -personaje del videojuego *The Legend of Zelda*. En primer lugar se procedió a modelar un personaje común basado en una figura de *Lego*, y posteriormente se modelaron sus armas -basadas en las originales de cada personaje- y cabellera o gorro. Sin obtener mayor recurso externo que imágenes guía para el modelado.

Respecto a su texturizado, en el caso de Aragorn se han obtenido de internet las imágenes correspondientes al torso y la cara del modelo real de juguete y se ha texturizado el resto del personaje para que quedase acorde. Para su escudo hemos optado por uno de los símbolos que lleva en el escudo de la película.

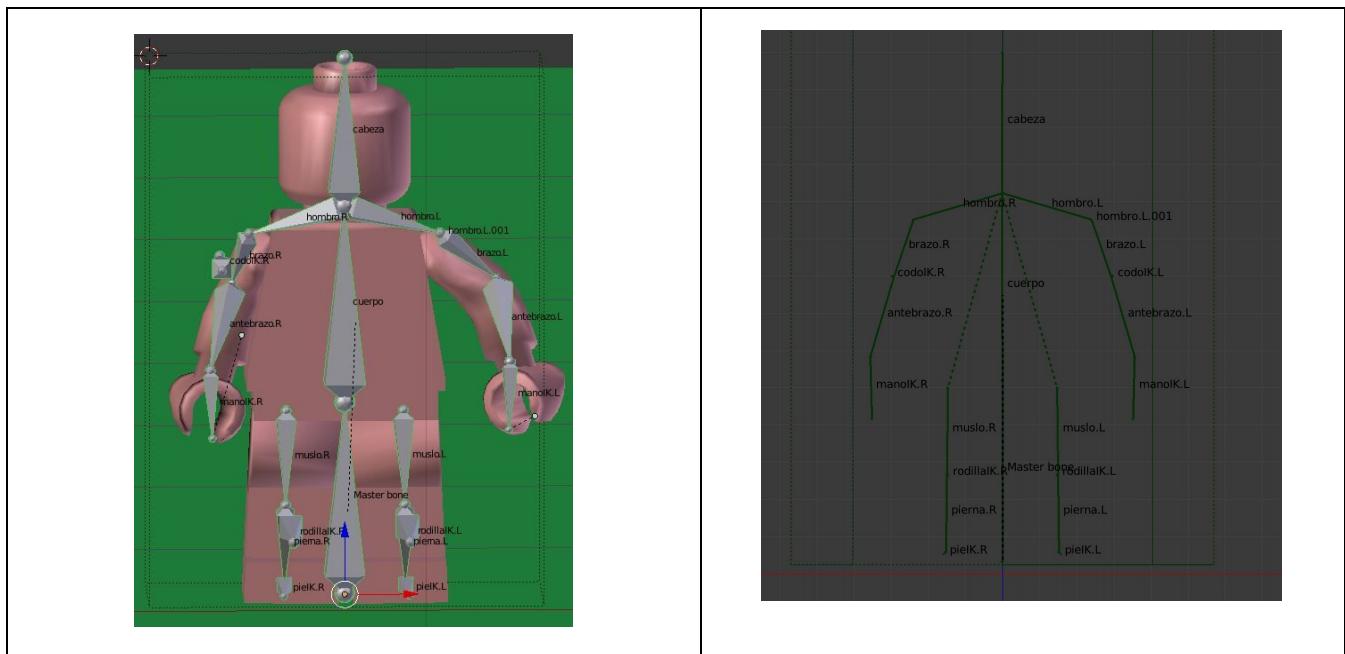
En el caso de Link, ya que no existen juguetes oficiales hemos obtenido un torso dibujado por un fan y el resto de las caras han sido generadas por nosotros, incluyendo la cara, la cual ha sido un gran reto. Para el escudo he tenido que insertar la imagen en Illustrator para ajustar el aspecto de altura/anchura con nuestro modelo 3D y eliminar detalles que se veían mal dada la baja resolución de la textura.

A continuación incluyo las imágenes originales, el mapa UV creado y el resultado final tanto de las figuras como de los escudos.





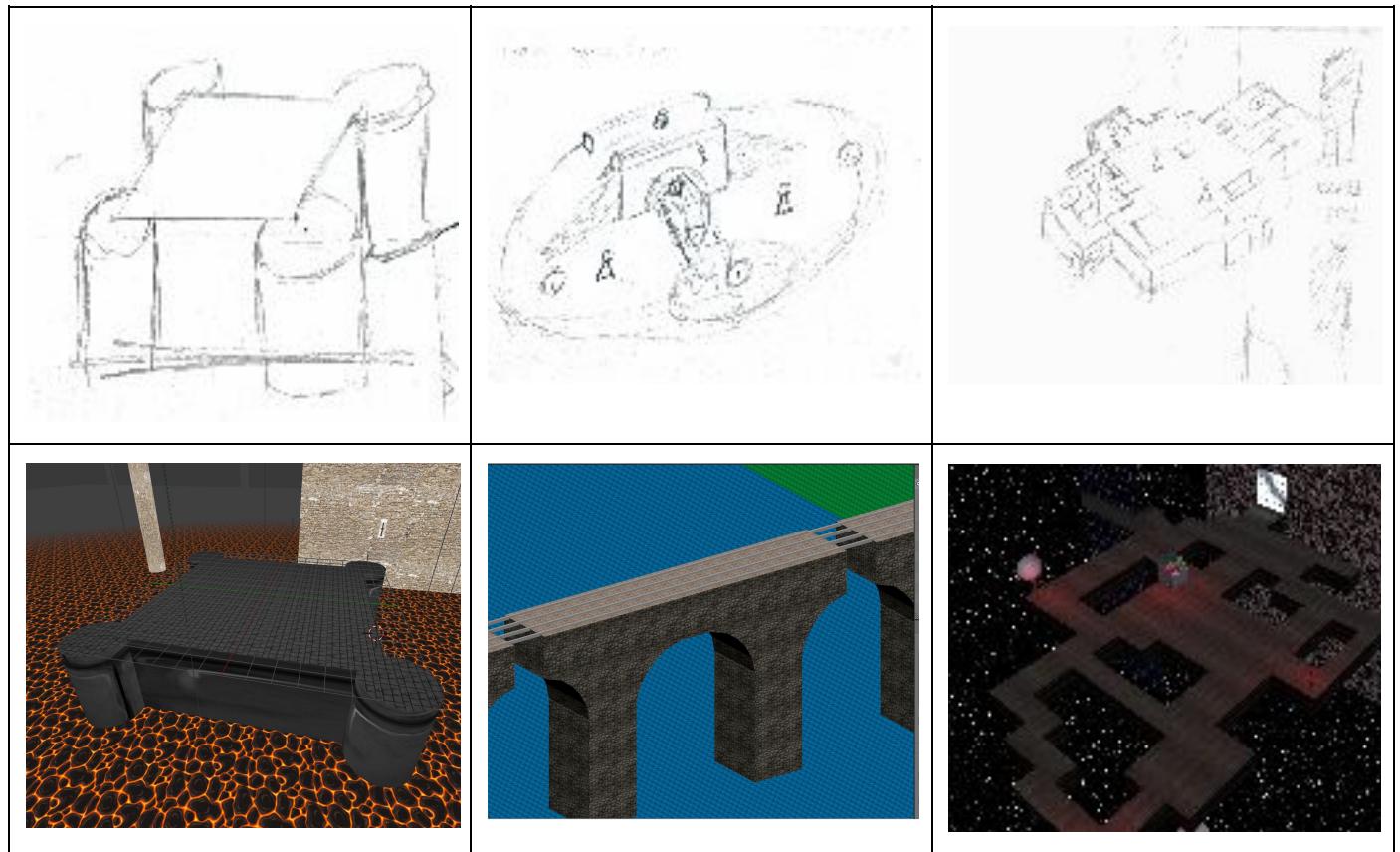
Para realizar la animación lo primero que se hizo fue crear el esqueleto del personaje. Cuenta con huesos en la cabeza, hueso de hombro, antebrazo, mano, muslo, pierna y pie. En la siguiente imagen se muestra como están distribuidos estos huesos. Para las restricciones se han hecho en modo IK para lo que se han creado huesos de rodilla IK, codo IK y pie IK. Se eligió esta forma de movimiento ya que es más fácil mover al personaje, pues solo se debe mover la mano o pie IK y los otros huesos se mueven con ellos.



Después de la creación de esqueleto y de las restricciones se comenzó la animación del personaje. Ambos personajes utilizan el mismo esqueleto y las mismas animaciones. Se unió la espada y el escudo al hueso de la mano para que fuera posible animarlo con los objetos. Se crearon animaciones de defensa, ataque, caminar, de inactivo de victoria y de derrota, estas dos últimas para la segunda entrega. Además se corrigieron las animaciones que ya se tenían.

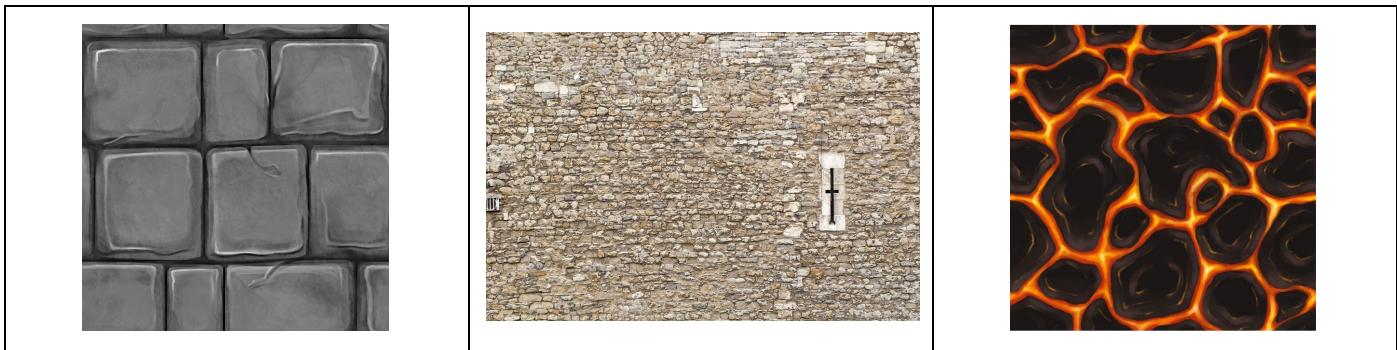


Respecto al escenario, inicialmente se concibieron una serie de bocetos:



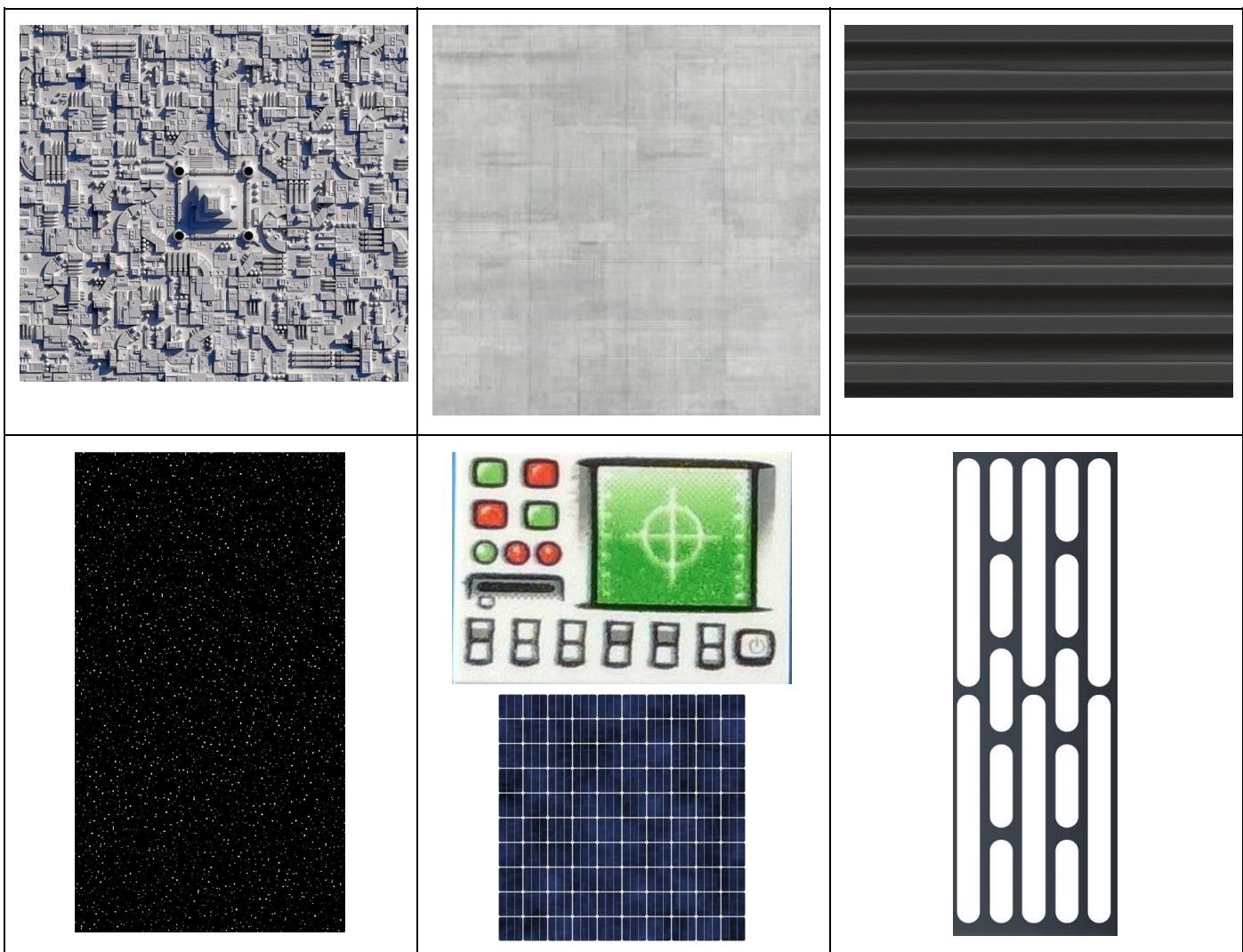
El primer escenario elegido fue Medieval. El modelador principal llevó a cabo el escenario y su iluminación, de acuerdo al concepto original. No se ha empleado ningún material externo para el modelado. En su diseño se buscaba recrear una época estilo medieval ficticia, como puede ser una caverna en el Señor de los Anillos.

Para su texturizado se han usado 3 fuentes distintas con repetición: piedra plataforma, piedra edificio y lava.

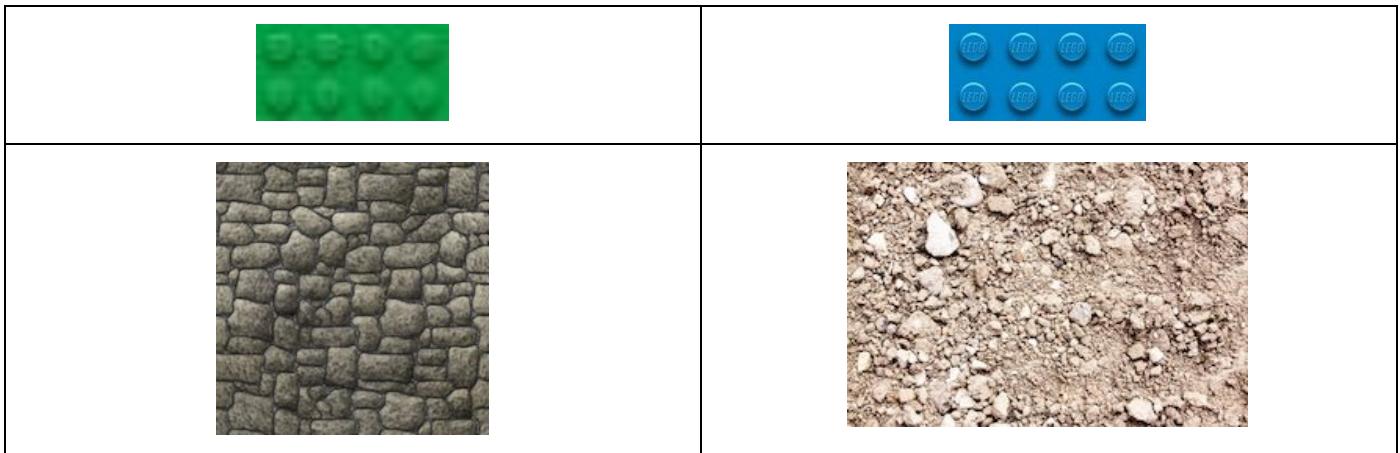


El segundo escenario implementado fue Espacial. Al igual que en Medieval, el modelador principal construyó el escenario y su iluminación, con la luz en movimiento. No se ha empleado ningún material externo para el modelado. En su diseño se le ha otorgado especial importancia a la iluminación, a fin de aprovechar esta cualidad del movimiento a través del espacio. También se aprovechó este objetivo con la inclusión de cierto grado de dificultad en el desplazamiento sobre la plataforma, mediante múltiples pequeñas plataformas unidas por estrechas pasarelas.

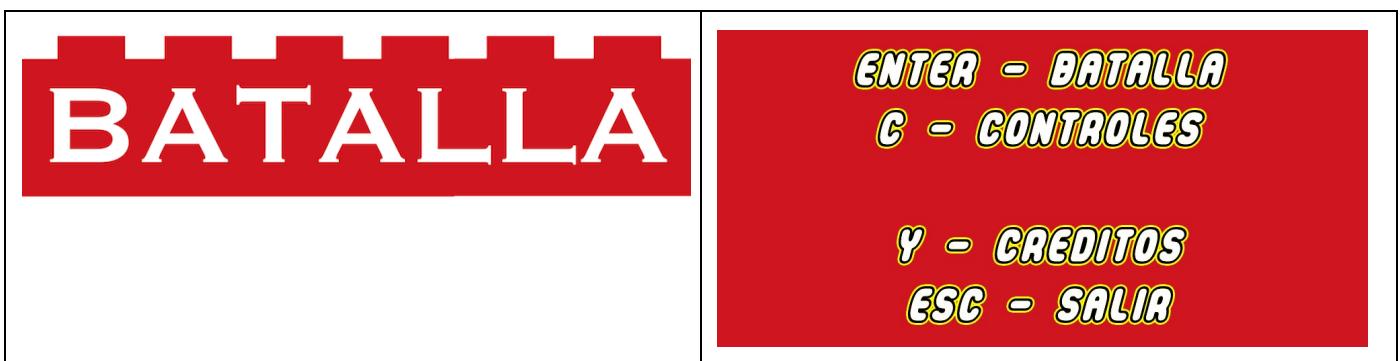
Para su texturizado se han usado las siguientes fuentes:



El tercer escenario implementado fue el Industrial. Al igual que con los anteriores, el modelador principal se basó en el concepto previo, aunque en este caso sufrió un giro al ambientarse exclusivamente encima del puente. Los trenes también se modelaron para la ocasión y se mueven, de acuerdo con el diseño original. Se diseñó de acuerdo a la existencia de dos vías con dos sentidos, a fin de incrementar la dificultad de la batalla agregando el elemento de esquivar el tren. Los sonidos emitidos por los trenes son ligeramente distintos en función de su procedencia, se diseñó así para que el jugador experimentado pueda perfeccionar su habilidad en el juego; de la misma forma que el movimiento de los fantasmas de PacMan es predecible con la experiencia suficiente.



Para los menús he optado por dos estilos: La fuente utilizada en el logo del videojuego y una fuente de letra parecida a las oficiales de LEGO:



Componentes

El fichero "LEGO Duels.blend" contiene diversos grupos creados para facilitar la acción "Append"; dentro de los mismos se incluyen todos los objetos con su jerarquía. Los grupos con sus objetos adheridos son los que siguen:

- Grupo_Mazmorra
 - CamaraMazmorra - Consiste en la única cámara del escenario, con el script CamaraMedieval.py asociado para su encuadre automático.
 - CampoBatalla - Iluminación de la plataforma de juego.
 - Columnas - Objetos columna que dan ambiente.
 - ParedPiedra - Material que identifica aquellos componentes decorativos de piedra.
 - Edificio - Elemento decorativo.
 - ParedPiedra - Material que identifica aquellos componentes decorativos de piedra.
 - LuzColumnas - Única luz que da ambiente a las columnas.
 - LuzEdificio - Conjunto de luces que simulan iluminación de lava.
 - PlanoLava - Plano con textura de lava que da suelo al escenario.
 - Lava - Material que se identifica con la lava.
 - Plataforma - Objeto sobre el que los personajes están jugando.
 - SueloMazmorra - Material relativo a la plataforma de batalla.
- Grupo_Industrial
 - CamaraTrenes - Consiste en la única cámara del escenario, con el script CamaraEspacial.py asociado para su encuadre automático.
 - Colina - Laterales del escenario
 - MatColina - Material que imita bloques verdes de lego
 - Locomotora - Tren que aparece durante la batalla
 - Puente - Laterales del puente
 - MatPiedra
 - Suelo - Suelo del puente
 - MatTierra
 - Rio
 - MatRio - Material que imita bloques azules de lego
 - Via Tren
 - SolPuente - Iluminación de la escena
- Grupo_Espacial
 - CamaraEspacial - Consiste en la única cámara del escenario, con el script CamaraEspacial.py asociado para su encuadre automático.
 - Sol - Lámpara tipo Sol con animación de rotación, responsable de cambios de iluminación.
 - Antenas - Elementos decorativos conformados por soporte de antena y cabeza de antena.
 - MatAntena - Material de la cabeza de antena.
 - Soporte - Material que se identifica con soporte de antena, panel solar y de luces rojas.
 - Nave - Elemento decorativo.
 - MatNave - Material que se identifica con la nave decorativa.
 - Obj Luz Roja - Elementos que simulan palos con luz al final. Tienen emparentados la luz puntual roja.
 - LuzRoja - Material de la bola roja superior del objeto, simula trasparencia para un mayor realismo.
 - Soporte - Material que se identifica con soporte de antena, panel solar y de luces rojas.

- Panel Solar - Elemento decorativo, con dos paneles solares y el soporte.
 - MatPanSolar - Material que se identifica con las placas fotovoltaicas.
 - Soporte - Material que se identifica con soporte de antena, panel solar y de luces rojas.
- PanelControl - Elemento decorativo central, simula un panel de control de la plataforma.
 - MatPanelControl - Material que se identifica con la consola de control de la plataforma.
- Plano Muerte - Plano invisible encargado de gestionar la muerte por caída.
 - PlanoMuerte - Material asociado al plano de muerte.
- Plataforma_Esp - Plataforma sobre la que los personajes luchan, dispone de los laterales como un objeto a parte para facilitar su texturizado.
 - MatSueloEspacio - Material del suelo de la plataforma.
 - MatSueloLaterales - Material de los laterales de la plataforma.
 - Invisible - Material de un plano incorporado para evitar que el personaje entre en la nave, la cual es *No collision*.
- Puerta - Plano creado para facilitar la inclusión de texturizado de la puerta.
 - MatPuertaEstacion - Material de la puerta de la Nave.
- PersonajeAragorn
 - AvatarAragorn - Figura con material invisible que gestiona las colisiones entre personajes.
 - EsqueletoAragorn - Esqueleto del personaje de Aragorn con los siguientes huesos
 - Master bone - Hueso maestro del esqueleto
 - Cuerpo - Hueso del pecho del personaje
 - Cabeza - Hueso de la cabeza del personaje
 - Hombro.L - Hueso del hombro izquierdo
 - Hombro.R - Hueso del hombro derecho
 - Brazo.L - Hueso del brazo izquierdo
 - Brazo.R - Hueso del brazo derecho
 - Antebrazo.L - Hueso del antebrazo izquierdo
 - Antebrazo.R - Hueso del antebrazo derecho
 - Mano.L - Hueso de la mano izquierda
 - Mano.R - Hueso de la mano derecha
 - Muslo.L - Hueso del muslo izquierdo
 - Muslo.R - Hueso del muslo derecho
 - Pierna.L - Hueso de la pierna izquierda
 - Pierna.R - Hueso de la pierna derecha
 - Pie.L - Hueso del pie izquierdo
 - Pie.R - Hueso del pie derecho
 - PielK.L - Hueso par restricciones IK de pie izquierdo
 - PielK.R - Hueso par restricciones IK de pie derecho
 - RodillaIK.L - Hueso para restricción de movimiento de la rodilla izquierda
 - RodillaIK.R - Hueso para restricción de movimiento de la rodilla derecha
 - ManoIK.L - Hueso de restricción IK para movimiento de brazo izquierdo
 - ManoIK.R - Hueso de restricción IK para movimiento de brazo derecho
 - CodoIK.L - Hueso para restricción IK para codo izquierdo
 - CodoIK.R - Hueso para restricción IK para codo derecho
 - Aragorn - Modelado del muñeco genérico de Lego con la textura de Aragorn.
 - MatAragorn - Textura Aragorn
 - EscudoAragorn - Escudo del personaje con su textura.

- MatEscudoAragorn - Textura escudo Aragorn
- EspadaAr - Espada del personaje, la cual no requiere textura.
- PeloAragorn - Objeto que simula la cabellera de Aragorn, sin textura.
- PersonajeLink
 - AvatarLink - Figura con material invisible que gestiona las colisiones entre personajes.
 - EsqueletoLink -Esqueleto del personaje de Link con los siguientes huesos
 - Master bone - Hueso maestro del esqueleto
 - Cuerpo - Hueso del pecho del personaje
 - Cabeza - Hueso de la cabeza del personaje
 - Hombro.L - Hueso del hombro izquierdo
 - Hombro.R - Hueso del hombro derecho
 - Brazo.L - Hueso del brazo izquierdo
 - Brazo.R - Hueso del brazo derecho
 - Antebrazo.L - Hueso del antebrazo izquierdo
 - Antebrazo.R - Hueso del antebrazo derecho
 - Mano.L - Hueso de la mano izquierda
 - Mano.R - Hueso de la mano derecha
 - Muslo.L - Hueso del muslo izquierdo
 - Muslo.R - Hueso del muslo derecho
 - Pierna.L - Hueso de la pierna izquierda
 - Pierna.R - Hueso de la pierna derecha
 - Pie.L - Hueso el pie izquierdo
 - Pie.R - Hueso del pie derecho
 - PielK.L - Hueso par restricciones IK de pie izquierdo
 - PielK.R - Hueso par restricciones IK de pie derecho
 - RodillaIK.L - Hueso para restricción de movimiento de la rodilla izquierda
 - RodillaIK.R- Hueso para restricción de movimiento de la rodilla derecha
 - ManoIK.L - Hueso de restricción IK para movimiento de brazo izquierdo
 - ManoIK.R - Hueso de restricción IK para movimiento de brazo derecho
 - CodoIK.L - Hueso para restricción IK para codo izquierdo
 - CodoIK.R - Hueso para restricción IK para codo derecho
 - EscudoLink - Escudo del personaje con su textura.
 - MatEscudoLink - Material principal del escudo de Link.
 - EspadaLink - Espada del personaje, sin textura.
 - GorroLink - Objeto que simula el pelo, orejas y gorro del personaje. Carece de textura.
 - Link - Modelado del muñeco genérico de Lego con la textura de Link.
 - MatLink - Material principal del personaje Link.
- Animaciones
 - AtaqueAragornFINAL - Animación de ataque con espada del personaje Aragorn.
 - DefensaAragornFINAL - Defensa con escudo del personaje Aragorn.
 - AndarAragornFINAL - Animación para el caminar del personaje Aragorn.
 - VictoriaAragornFINAL - Animación de victoria para el personaje de Aragorn.
 - DerrotaAragornFINAL - Animación de derrota para el personaje de Aragorn.
 - AtaqueLinkFINAL - Animación de ataque con espada del personaje Link.
 - DefensaLinkFINAL - Defensa con escudo del personaje Link.
 - AndarLinkFINAL - Animación para el caminar del personaje Link.
 - VictoriaLinkFINAL - Animación de victoria para el personaje de Link.

- DerrotaLinkFINAL - Animación de derrota para el personaje de Link.
- Menu Start
 - Camera Start - Cámara que apunta a los personajes al inicio.
 - Camera Creditos - Cámara que apunta a los créditos.
 - Plane Controles - Plano que contiene los controles.
 - MatControles - Material del plano asociado.
 - Plane Creditos - Plano que contiene los créditos.
 - MatCreditos - Material del plano asociado.
 - Plane Enter - Plano que contiene las instrucciones iniciales.
 - MatStart - Material del plano asociado.
 - Plane Logo Coin - Plano que contiene el logo de la empresa.
 - MatLogoCoin - Material del plano asociado.
 - Plane Logo Duels - Plano que contiene el logo y nombre del juego.
 - MatLogoDuel - Material del plano asociado.
 - Plane Pared - Plano con la pared decorativa.
 - SueloMazmorra - Material de la pared trasera.
 - Plane Suelo - Plano de suelo.
 - SueloMazmorra - Material del suelo del menú.
 - Sun - Luz que ilumina la escena.
- Menu escenario
 - Camera - Cámara que enfoca el escenario desde la perspectiva vista en el menú.
 - Sun - Luz que ilumina los carteles.
 - Plane Espacial - Plano que contiene el cartel del escenario Espacial.
 - Mat LetEspacial - Material del plano asociado.
 - Plane Flecha D - Plano que contiene la flecha que indica la posibilidad de cambiar de escenario hacia la derecha.
 - MatFlecha - Material del plano asociado.
 - Plane Flecha I - Plano que contiene la flecha que indica la posibilidad de cambiar de escenario hacia la izquierda.
 - MatFlecha - Material del plano asociado.
 - Plane Industrial - Plano que contiene el cartel del escenario Industrial.
 - Mat LetIndustrial - Material del plano asociado.
 - Plane Luchar - Plano que contiene el letrero de luchar.
 - Mat LetLuchar - Material del plano asociado.
 - Plane Medieval - Plano que contiene el cartel del escenario Medieval.
 - Mat LetMedieval - Material del plano asociado.
 - Plane Volver - Plano que contiene el letrero de volver.
 - Mat LetVolver - Material del plano asociado.
 - Plane Tecla Enter - Plano que contiene la tecla Enter, indicando la forma de comenzar a luchar.
 - MatEnter - Material de la tecla Enter.
 - Plane Tecla Backspace - Plano que contiene la tecla Backspace o Retroceso, indicando la forma de volver al menú principal.
 - MatBack - Material de la tecla Backspace.
- Pausa
 - Camera - Cámara que encuadra *Plane Pausa* y *Plane Transparente*, superpuesta luego con la cámara de la escena que se esté pausando.

- Plane Pausa - Plano que contiene la información de la pantalla de pausa.
 - MatPausa - Material del plano asociado.
- Plane Transparente - Plano oscuro pero con transparencia que tapa toda la pantalla.
 - Material - Material semitransparente que deja en segundo plano la batalla.
- Victoria
 - Camera - Cámara que encuadra *Plane Victoria* y *Plane Transparente*, superpuesta luego con la cámara de la escena de batalla.
 - Plane Victoria - Plano que contiene la información de la pantalla de victoria.
 - MatVictoria - Material del plano asociado.
 - Plane Transparente - Plano oscuro pero con transparencia que tapa toda la pantalla.
 - Material - Material semitransparente que deja en segundo plano la batalla.
- Barra de Vida
 - Camera - Cámara que encuadra la barra de vida donde se espera encontrar durante la batalla.
 - Barra - Contiene las dos barras y su lógica
 - BarraDetras - Barra roja
 - Vida - Barra verde
- Sonidos
 - GolpeEfecto - Efecto de golpe entre personajes.
 - MovimientoMenú - Click que se escucha al navegar en el menú.
 - MenúSelecciónSonido - Música al seleccionar algo en el menú.
 - MúsicaVictoria - Música que se escucha al ganar la batalla.
 - MúsicaNiveldeLava - Música para la pelea en nivel medieval.
 - MúsicaParaEspacio - Música para nivel del espacio con nave.
 - MúsicaParaNivelDeTrenes - Música para el tercer nivel que será de trenes en movimiento (no implementado aún).