UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Juego de realidad virtual sobre recogida de setas: UBUSetasVR 1.0 Documentación Técnica

Presentado por Rodrigo Jurado Pajares
en Universidad de Burgos — 27 de junio de 2018
Tutor: D. José Francisco Díez Pastor y D. Raúl Marticorena Sánchez

Índice general

Índice general I

Índice de figuras III

Índice de tablas V

Apéndice A Plan de Proyecto Software 1
- A.1. Introducción . . . 1
- A.2. Planificación temporal . . . 2
- A.3. Estudio de viabilidad . . . 11

Apéndice B Especificación de Requisitos 15
- B.1. Introducción . . . 15
- B.2. Objetivos generales . . . 15
- B.3. Catálogo de requisitos . . . 15
- B.4. Especificación de requisitos . . . 16

Apéndice C Especificación de diseño 24
- C.1. Introducción . . . 24
- C.2. Diseño de datos . . . 24
- C.3. Diseño arquitectónico . . . 28
- C.4. Diseño de interfaces . . . 30

Apéndice D Documentación técnica de programación 32
- D.1. Introducción . . . 32
- D.2. Estructura de directorios . . . 32
- D.3. Manual del programador . . . 35
- D.4. Compilación, instalación y ejecución del proyecto . . . 36
- D.5. Pruebas del sistema . . . 42

Apéndice E Documentación de usuario **46**
E.1. Introducción 46
E.2. Requisitos de usuarios 46
E.3. Instalación 46
E.4. Manual del usuario 47

Bibliografía **55**

Índice de figuras

A.1. Gráfico de la iteración 1. 3
A.2. Gráfico de la iteración 2. 4
A.3. Gráfico de la iteración 3. 5
A.4. Gráfico de la iteración 6. 7
A.5. Gráfico de la iteración 7. 8
A.6. Gráfico de la iteración 9-10. 9
A.7. Gráfico de la iteración 17. 11
A.8. Especificaciones técnicas del ordenador utilizado. 13
A.9. Especificaciones de la tarjeta gráfica del ordenador utilizado. . . . 13

C.1. Tablas de la base de datos SQLite. 25
C.2. Diagrama de clase de la estructura de datos de las setas. 26
C.3. Diagrama de clases propias del proyecto. 26
C.4. Diagrama de clases del espacio *UBUSetasVR.Database*. 27
C.5. Diagramas de clases del espacio *UBUSetasVR.EditorScripts*. 27
C.6. Diagrama de clases del espacio *UBUSetasVR.Managers*. 28
C.7. Diagrama de clases del espacio *UBUSetasVR.Placement*. 29
C.8. Diagrama de clases del espacio *UBUSetasVR.ScriptableObjects*. . . 29
C.9. Diagrama de clases del espacio *UBUSetasVR.UI*. 29
C.10. Diagrama de secuencia para el proceso de actualizar puntuación. . 30
C.11. Diseño de las interfaces del juego. 31
C.12. Flujo de las interfaces del juego durante su ejecución. 31

D.1. Estructura de directorios del proyecto 33
D.2. Ubicación del menú *Build Settings* 37
D.3. Vista del menú *Preferences* 37
D.4. Ubicación del menú *Preferences* 38
D.5. Vista del menú *Preferences* 39
D.6. Ubicación del menú *Player Settings* 39
D.7. Vista del menú *Player Settings* 40

D.8. Ubicación de la opción para instalar apps desconocidas. 41
D.9. Aplicaciones necesarias para que Daydream funcione 41
D.10. Pantalla que muestra cómo colocar el móvil en el visor. 42
D.11. Botones del editor de Unity para controlar la ejecución de la aplicación. 43
D.12. Componentes para simular en el editor el mando de Daydream . . 44
D.13. Ejemplo de la consola de Unity en funcionamiento. 44
D.14. Ejemplo del Profiler de Unity en funcionamiento. 45

E.1. Esquema del mando del *Daydream View* 47
E.2. Representación gráfica de los movimientos de rotación 48
E.3. Ejemplo del menú de selección de niveles 48
E.4. Elemento visual de ayuda para los controles de movimiento. 49
E.5. Elemento visual de ayuda para los controles de interacción. 50
E.6. Panel de información de una seta. 51
E.7. Menú principal de la aplicación. 51
E.8. Menú de selección de niveles y panel de objetivos 51
E.9. Colocación del móvil y activación del mando 52
E.10. Pantalla de carga entre niveles. 53
E.11. Cestas donde colocar las setas 53
E.12. Texto de final de juego con la puntuación alcanzada. 54

Índice de tablas

A.1. Costes de personal. 12
A.2. Costes del Personal Docente Investigador. 12
A.3. Costes totales del hardware por el tiempo usado. 13
A.4. Costes software. 13
A.5. Costes totales del proyecto. 14

B.1. RF-1.1: Visualizar el simulador a través del Daydream View. . . . 17
B.2. RF-1.2: Controlar la aplicación con el mando del Daydream View. 17
B.3. RF-1.2.1: Controlar los menús con el mando. 18
B.4. RF-1.2.1.1: Abrir/cerrar menú de pausa. 18
B.5. RF-1.2.1.2: Seleccionar nivel. 18
B.6. RF-1.2.2: Controlar el desplazamiento del jugador por el escenario con el mando. 19
B.7. RF-1.2.3: Controlar la interacción con las setas con el mando. . . . 19
B.8. RF-1.2.3.1: Coger/soltar las setas y desplazarlas. 20
B.9. RF-1.2.3.2: Rotar las setas. 20
B.10. RF-1.2.3.3: Mostrar/ocultar información de las setas. 21
B.11. RF-1.2.4: Interactuar con las cestas. 21
B.12. RF-1.2.4.1: Sumar/restar puntuación. 21
B.13. RF-2.1: Usar datos de setas de una base de datos. 22
B.14. RF-2.2.1: Crear escenario. 22
B.15. RF-2.2.1.1: Crear y colocar árboles. 22
B.16. RF-2.2.1.2: Crear y colocar setas. 23
B.17. RF-2.2.2: Definir reglas del nivel. 23

D.1. Requisitos del sistema para desarrollar con Unity 36

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Este proyecto ha sido exclusivamente de ámbito software, y aunque por ello se debería haber seguido una planificación acorde a cualquier otro proyecto software, el hecho de tratarse del desarrollo de un videojuego hace que tenga diferencias notables en su planificación y diseño respecto a un proyecto software convencional.

Para este desarrollo se ha seguido una metodología ágil, Scrum, en la que se ha trabajado en iteraciones enmarcadas en una unidad de tiempo definida con antelación[1]. En este caso, la unidad de tiempo empezó siendo de dos semanas porque había muchas tareas de investigación y planificación donde no era necesario reunirse con mucha frecuencia y fue modificada a una semana cuando se empezó a trabajar sobre aspectos de programación y desarrollo. El tener esta variabilidad en las unidades de tiempo, previo aviso y consenso entre las partes implicadas, es algo que se aplica bastante en el desarrollo de videojuegos por la gran diversidad de tareas y de aspectos a tratar en un desarrollo de estas características y permite cierta flexibilidad a la hora de organizarse cuando se esperan tareas de larga duración. Puede no ser muy buena práctica como tal según el manifiesto ágil, pero permite compaginar mejor las necesidades y el tiempo de los desarrolladores.

Otro de los aspectos destacables respecto a un desarrollo software convencional, es el tipo de programación con el que se trabaja en Unity. En un proyecto de software se trabaja normalmente con un único tipo de programación, orientada a objetos, estructurada o funcional, que son por ejemplo las más utilizadas. En Unity en cambio se trabaja con programación orientada a componentes, lo que en resumen implica que habrá multitud de elementos

[1] Ver apartado "Técnicas y Herramientas de la memoria"

pequeños que aportan funcionalidades muy concretas a los objetos sobre los que trabajamos en el editor. Esto es muy similar al concepto de reusabilidad o responsabilidad única tratado en los principios SOLID[2] de la programación orientada a objetos, a tener clases pequeñas de funcionalidades muy definidas y propósitos muy concretos, pero en este caso en el código se van a apreciar multitud de clases a primera vista inconexas entre sí que no ganarán sentido hasta que sean utilizadas por el editor de Unity en algún objeto[3]. Este tipo de estructura en el código hace que haya muchas tareas planificadas que no sean de programación en sí, sino que se basen en añadir componentes en los objetos del editor o en modificar las propiedades de estos.

A.2. Planificación temporal

El proyecto se ha realizado en iteraciones de una semana de duración al principio, cambiando a iteraciones cada semana, teniendo una reunión con los tutores al final de cada iteración para ver los objetivos alcanzados y marcar los objetivos de la siguiente iteración. Se ha realizado el proyecto en un periodo de veintiséis semanas, realizando quince iteraciones durante todo el proyecto. Se comentarán a continuación aspectos destacables de cada iteración y se incluirán en algunas de las iteraciones los gráficos *burndown* generados con ZenHub [25].

Resumen de las iteraciones

Iteración 1 (18/10/2017 - 01/11/2017)

En esta primera iteración se empezó con el planteamiento del proyecto, la elección de herramientas de gestión a utilizar y la estructura general del repositorio una vez elegido el control de versiones y su alojamiento. En esta iteración se empieza a mirar el funcionamiento de LaTeX para la documentación y el uso de ZenHub como herramienta de gestión y planificación, sobretodo el apartado de utilización de las gráficas, nunca usado hasta este momento. En relación al código, se crea una escena básica y limpia de Unity para ir sentando las bases de la futura estructura de directorios dentro del editor (ver sección D.2 de los anexos) y empezar con la estructura de ramas del repositorio según la metodología Gitflow[4]. Se puede ver el gráfico de esta iteración en la figura A.1.

[2] *Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion* [24].

[3] Ver sección "GameObjects" del apartado "Conceptos teóricos" de la memoria.

[4] Ver sección "Gitflow" del apartado "Técnicas y herramientas" de la memoria.

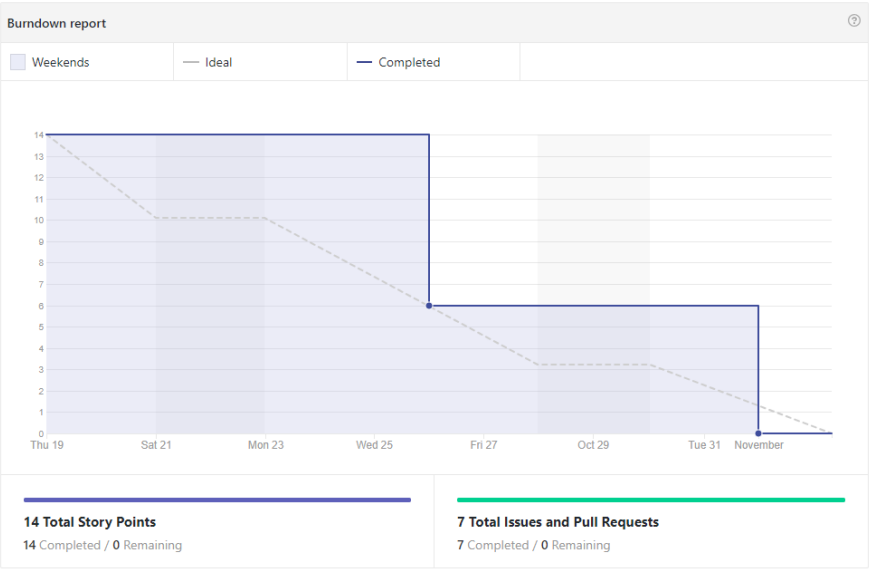

Figura A.1: Gráfico de la iteración 1.

Iteración 2 (01/11/2017 - 15/11/2017)

Con la estructura básica en Unity y el modelo de ramas preparado, se empezó en esta iteración con pruebas de generación procedimental[5] de vegetación y pruebas de rendimiento generales en diferentes ramas, ya que ambos eran aspectos claves en este proyecto. La idea era ir viendo cómo se podía generar vegetación realista abundante con bajo impacto en el rendimiento en un dispositivo móvil. Se iniciaron las pruebas de forma general, un poco "a ciegas", para conocer mejor estos ámbitos, obtener algún resultado y organizar poco después el primer contacto con la artista externa del proyecto junto a los tutores. En esta reunión se marcó la línea estética que seguiría el proyecto y al tener ya en este momento algún resultado del posible impacto que tendría el tipo de vegetación usado, se pudieron tratar mejor los aspectos más importantes a tener en cuenta en la realización de los modelos 3D requeridos. La línea estética elegida para los modelos fue un estilo *low poly*, modelos 3D muy simples de pocos polígonos que tienen poca carga para la tarjeta gráfica. Esta elección fue marcada sobretodo por no tener elegida aún ni la plataforma ni el dispositivo móvil a utilizar, por lo que se marcaron objetivos de rendimiento muy bajos para ir sobre seguro. Se puede ver el gráfico de esta iteración en la figura A.2.

[5]Ver apartado "Generación aleatoria y procedimental" de la memoria.

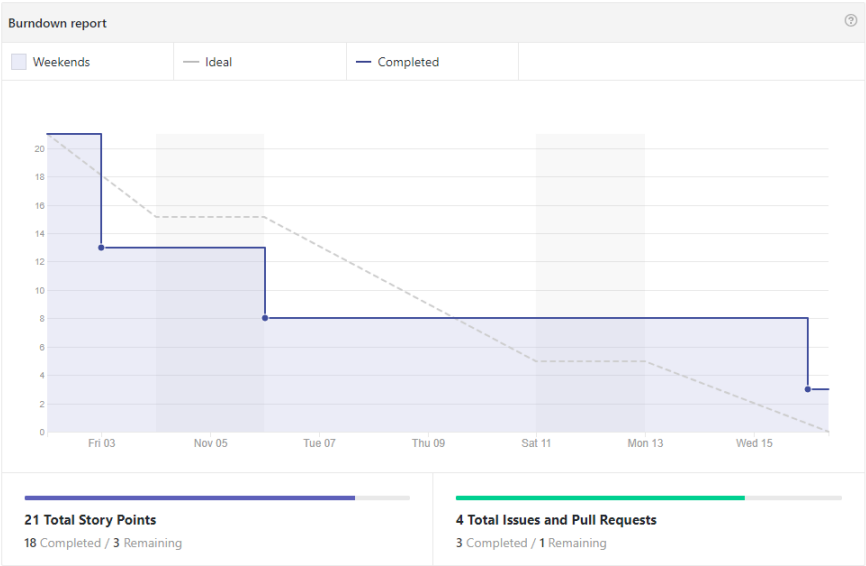

Figura A.2: Gráfico de la iteración 2.

Iteración 3 (15/11/2017 - 30/11/2017)

En esta iteración se eligió la plataforma de desarrollo, Daydream[6], y se realizó un estudio sobre el hardware compatible y disponible para esta plataforma [9]. Se realizó un estudio de coste/rendimiento de las mejores opciones teniendo en cuenta el presupuesto disponible y se empezaron los trámites para su adquisición, ya que cuanto antes se contase con el dispositivo para poder realizar pruebas, mejor se podría analizar y cuidar el rendimiento de la aplicación. Durante este tiempo se empezó con un periodo de documentación y experimentación propia sobre la plataforma Daydream, desconocida su existencia hasta este momento, y su uso e integración con Unity. Se puede ver el gráfico de esta iteración en la figura A.3.

Parón exámenes y vacaciones (30/11/2017 - 01/02/2018)

Durante este periodo se hizo un parón de las reuniones con los tutores debido a los exámenes del primer semestre y las vacaciones de navidad. No obstante, el desarrollo siguió, a menor ritmo, con las tareas pendientes de la iteración anterior, sobretodo con la de documentación y conocimiento de la plataforma Daydream. En este periodo la artista terminó una primera versión de algunos modelos, pudiendo probar desde este momento la integración de los modelos en Unity, comprobando los problemas en la importación al editor y corrigiendo los mismos. Destacar que inicialmente hubo pequeños problemas en relación a las escalas y formatos manejados por Unity y el software

[6] Ver sección “Daydream” del apartado “Conceptos teóricos” de la memoria.

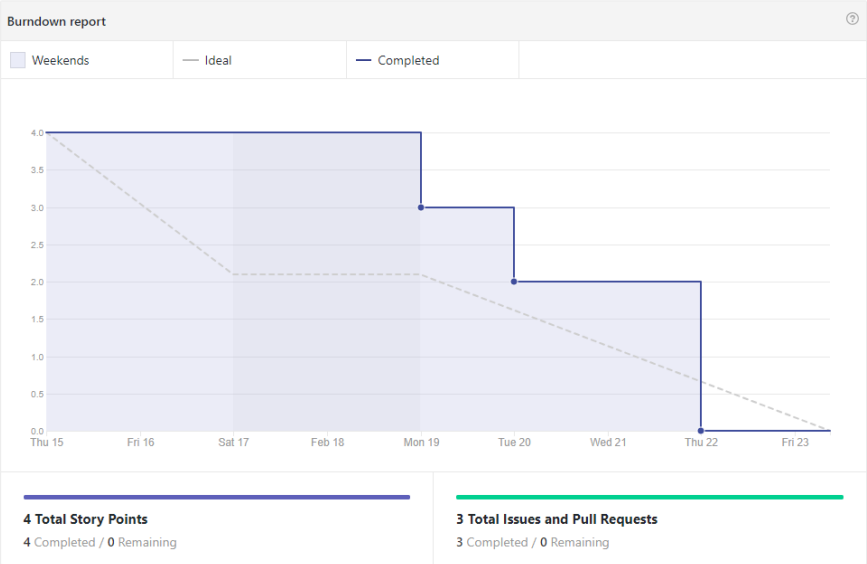

Figura A.3: Gráfico de la iteración 3.

de modelado utilizado, Blender [23], pero gracias a la buena comunicación existente se solventaron rápidamente. Durante este periodo también se empezó a experimentar con la generación y manejo de terrenos en Unity, lo que facilitó las pruebas y correcciones sobre las escalas de los modelos. Fue durante este trabajo de integración de modelos y terrenos en una rama del repositorio donde se empezaron a ver problemas a trabajar con la metodología Gitflow en Unity[7].

Iteración 4 (01/02/2018 - 15/02/2018)

Esta iteración estuvo dedicada casi en su totalidad a solventar el problema de Gitflow en Unity y a generar el menú principal de la aplicación y una escena de prueba, integrando en este momento los conocimientos adquiridos acerca de Daydream. De aquí en adelante se omitió el uso de las ramas que no fuesen *master* o *develop*, dejando de esta forma una rama para las *releases* o versiones finales de la aplicación y otra rama para las versiones de pruebas. Es en esta iteración donde se integran los primeros controles del videojuego, destacando sobretodo la aparición de los dos modos de control principales: el modo manipulación y el modo teletransporte. Se decide además que será prioritario centrarse en la creación de un nivel y de las herramientas necesarias para crear un nivel, para poder extrapolarlo a nuevos niveles en un futuro sin mucha dificultad: creado un nivel, creados todos. Se introduce también una librería de sonido espacial [10] y los primeros sonidos ambientales y de efectos

[7] Ver apartado “Gitflow y Unity” del apartado de la memoria.

en el menú principal. Esta librería de sonido generará un error que evitará más progresos con el sonido y será arrastrado a la siguiente iteración.

Iteración 5 (15/02/2018 - 23/02/2018)

En esta iteración se empiezan a hacer las reuniones más a menudo, bajando el tiempo de las iteraciones a una semana para agilizar el desarrollo y el seguimiento del mismo. Es en este periodo cuando se arregla el error de la librería de sonido tras actualizar la librería y la versión de Unity utilizada a una versión con correcciones [18], siendo necesario revisar que el resto de librerías siguiesen funcionando correctamente tras el cambio. Durante esta iteración se realiza también todo el trabajo de integración de la base de datos heredada del proyecto UBUSetas[8], introduciendo de esta forma las librerías necesarias para trabajar con Sqlite en Unity[9] e introduciendo el concepto de los contenedores de datos[10] para las setas. En relación a los contenedores de datos se crean también herramientas que funcionarán en el editor para facilitar la creación automática de estos contenedores de datos, las cuales se comentarán con más detalle en el apartado D.3 de estos anexos. También se van realizando pruebas con la interacción del controlador y las setas, pudiendo empezar a coger las setas desde este momento, y se integra el sistema de transición entre niveles, haciendo una separación inicial para futuros niveles en base al tipo de bosque.

Iteración 6 (23/02/2018 - 08/03/2018)

En esta semana se realiza la primera *release*, la primera versión de prueba de la aplicación, aunque todavía no se puede probar en su totalidad en un dispositivo móvil compatible por no contar aún con él. No obstante, se prueba que en un móvil con Android 7.1 la aplicación se instala y ejecuta por lo menos. A raíz de empezar a realizar compilaciones se hace necesario introducir una nomenclatura de versiones, usando para ello un versionado semántico [14]. Se seguirá a partir de ahora un nombrado de versiones "Major.Minor", donde se incrementará Major cuando se considere una versión final, y se incrementará Minor cuando se considere una versión de pruebas estable. Se usará para marcar las versiones, y con ello las *releases*, el etiquetado incorporado en Git y GitHub. Se empieza a trabajar también sobre el sistema de juego y qué flujo seguirá para considerar una elección como buena o mala y se modifica también en este periodo la estructura de datos usada en los contenedores de datos, por ser esta demasiado sencilla para todos los posibles casos que se quieren poder manejar en un futuro. Se puede ver el gráfico de esta iteración en la figura A.4.

[8] Ver apartado "Trabajos relacionados" de la memoria principal.
[9] Ver sección "Sqlite" del apartado "Técnicas y herramientas" de la memoria.
[10] Ver sección "ScriptableObjects" del apartado "Conceptos teóricos" de la memoria.

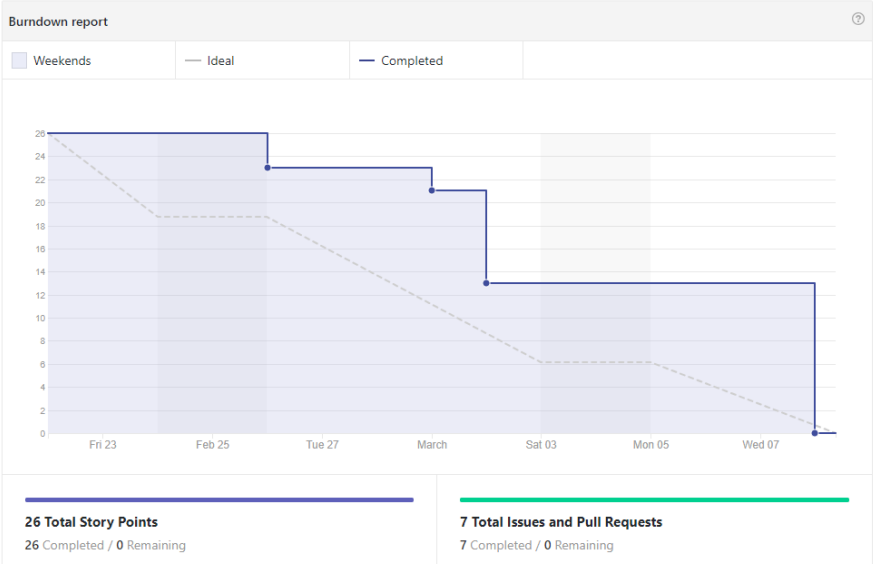

Figura A.4: Gráfico de la iteración 6.

Iteración 7 (08/03/2018 - 22/03/2018)

En esta iteración, algo más larga que las anteriores, se van haciendo progresos en el sistema de juego, la presentación de elementos de interfaz y se reestructura ligeramente la jerarquía de directorios para que esta sea más clara. Se centran esfuerzos en la parte de interfaz de usuario: se estudian opciones de mejora visual de las interfaces y se añade el plugin TextmeshPro [16] para tal fin, aunque no se aplican todas las mejoras planificadas, como se puede ver en el gráfico de la figura A.5. Tras esta iteración ya se pueden ver elementos de interfaz tales como un marcador de puntuación o un panel de información en las setas.

Iteración 8 (22/03/2018 - 28/03/2018)

Esta iteración estuvo destinada a mejorar la calidad del proyecto y refactorizar elementos de la interfaz para que fuesen más claros y funcionasen mejor en conjunto a otros sistemas de la aplicación, así como centrar esfuerzos en terminar un primer nivel completo. Por ejemplo, en el marcador de la interfaz se separa la lógica de la visualización para que pueda funcionar mejor con los eventos originados por el controlador de nivel. Se incorporan nuevos modelos de setas y se localizan y corrigen errores en estos y otros modelos ya importados. Se incorpora también un sistema de colocación totalmente aleatorio de árboles y setas en el escenario para ir trabajando con la generación de niveles, aunque los resultados obtenidos con este sistema no son nada realistas. Se introduce también una primera forma de acabar el nivel en el que se está, haciendo que

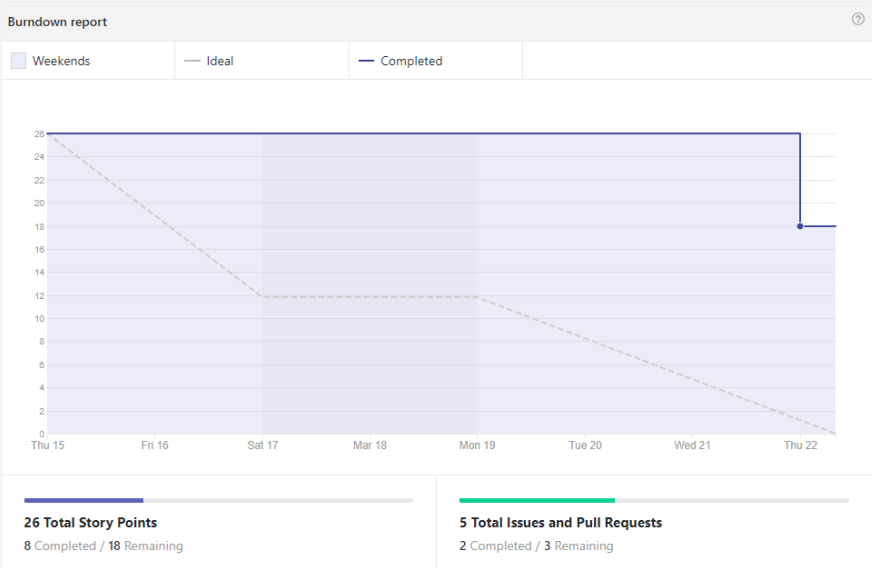

Figura A.5: Gráfico de la iteración 7.

la versión generada en esta iteración pueda ser jugable en términos de tener los sistemas necesarios para poder empezar, jugar y acabar un nivel, aunque tenga muchas carencias, sobretodo en usabilidad, variedad y dificultad.

Iteración 9 (28/03/2018 - 12/04/2018)

En esta iteración se corrigen errores introducidos en la iteración anterior y se empieza a trabajar en un sistema de colocación de setas alrededor de los árboles utilizando probabilidades ponderadas, de forma que se le dé cierta aleatoriedad controlada a los niveles y se consiga una mejora en el realismo de la colocación de las setas.

Iteración 10 (12/04/2018 - 19/04/2018)

En esta iteración se centran esfuerzos en realizar mejoras en el rendimiento teórico de la aplicación al cambiar parámetros de importación de los modelos de las setas, los cuales hacen que la instanciación de modelos en el nivel esté a cargo de la tarjeta gráfica del dispositivo, que tiene mayor potencia y rapidez de cálculo, así como que ocupen menos memoria en el dispositivo en ejecución. Aprovechando estas mejoras se introducen nuevos modelos de setas y se actualizan algunos ya existentes, pudiendo subir así el nivel de calidad de los mismos al tener menos exigencia de procesamiento.

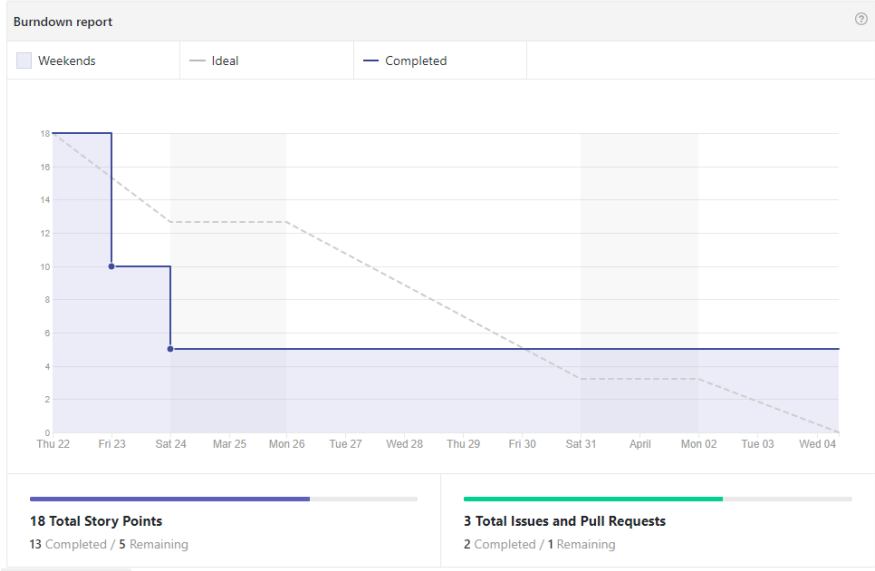

Figura A.6: Gráfico de la iteración 9-10.

Iteración 11 (19/04/2018 - 03/05/2018)

En esta iteración se vio un problema de comunicación entre artista y desarrollador que ocasionó que hubiese especies de setas que no estaban presentes en la base de datos pero de las que ya se habían realizado modelos 3D. Tras consultarlo con los tutores, se decidió modificar la base de datos heredada para introducir las nuevas especies y así no desperdiciar los modelos ya generados y perder tiempo en la generación de otros nuevos. La parte mala de este problema fue que se perdió una iteración casi completa en solucionarlo y en actualizar elementos ya presentes en el proyecto para funcionar bien con los nuevos cambios. En esta iteración se implementó el sistema selectivo de pintado en cámara por distancia[11].

Iteración 12 (03/05/2018 - 10/05/2018)

En esta semana se termina con el sistema de colocación ponderada por porcentajes, incluyendo la herramienta para su uso en el editor, teniendo así una herramienta muy potente para la creación de este y futuros niveles, aunque por ahora solo funciona en un terreno plano. Se crea también un agregado de funciones auxiliares que serán utilizadas en varias partes del código del proyecto y aparece por primera vez el espacio de nombres del proyecto para tener agrupadas las funciones propias de este proyecto de las de otras librerías utilizadas.

[11] Ver sección “Rendimiento y pruebas en dispositivos móviles” del apartado “Aspectos relevantes” de la memoria.

Iteración 13 (10/05/2018 - 17/05/2018)

En esta iteración se recibe, tras una larga espera, el dispositivo móvil que se va a utilizar para las pruebas y para la demostración de la aplicación, por lo que se utiliza esta iteración para poner a punto dicho dispositivo para comprobar que la aplicación funciona como es debido y que el rendimiento es bueno, aunque se dispone ya de poco tiempo para realizar pruebas o cambios más extensos en la arquitectura de los sistemas tras ver los resultados. No obstante, no se cuenta aún con el sistema Daydream View [7] de casco y mando que permite la inmersión en la Realidad Virtual, por lo que no se puede probar todavía la experiencia real de usuario en la aplicación.

Iteración 14 (17/05/2018 - 24/05/2018)

En esta semana se recibe el Daydream View y con ello se tiene la totalidad de los elementos hardware necesarios y se puede probar en su totalidad la aplicación, comprobando con satisfacción que funciona acorde a lo planeado, aunque se aprecian errores en los sistemas de cámara que sólo aparecen en tiempo de ejecución y que previamente no eran apreciables por no contar con el visor. Es en esta semana también cuando se trabaja en integrar el tipo de terrenos final, usando para ello el plugin [22] y en mejorar la colocación de elementos en terrenos no planos.

Iteración 15 (24/05/2018 - 31/05/2018)

En la que en principio era la última semana del proyecto se remata el estilo de las interfaces de usuario y se realizan correcciones menores de los errores apreciados anteriormente y se prepara la versión final de la aplicación para probar en el dispositivo móvil. Lamentablemente en esta versión que iba a ser final se aprecian errores internos en la aplicación, no visibles anteriormente por el tipo de compilación, lo que obliga a posponer la entrega del proyecto y alargar su desarrollo tres semanas más.

Iteración 16 (31/05/2018 - 06/06/2018)

En esta iteración se centran todos los esfuerzos en corregir los errores encontrados en la iteración anterior. Se establece que la causa de los mismos era la alta complejidad de las mallas de colisiones de los modelos 3D utilizados, siendo necesario reemplazarlas por mallas más simples (de menos polígonos) a costa de tener menos fidelidad en las colisiones con el terreno. Esto no solo aligera la compilación de la aplicación, sino que mejora drásticamente el rendimiento de la misma, mejorando los tiempos de carga entre niveles de forma brutal. Se detectan también nuevos errores menores, por lo que no se genera todavía una versión final.

Iteración 17 (06/06/2018 - 21/06/2018)

Subsanados los errores más importantes, se realiza una nueva versión de la aplicación, contando además ahora con un logotipo de carga en la pantalla entre la carga de niveles. Se generan además niveles extra y se corrigen más problemas encontrados, así como se mejora la usabilidad de la aplicación, introduciendo ayudas visuales para el usuario entre otras cosas.

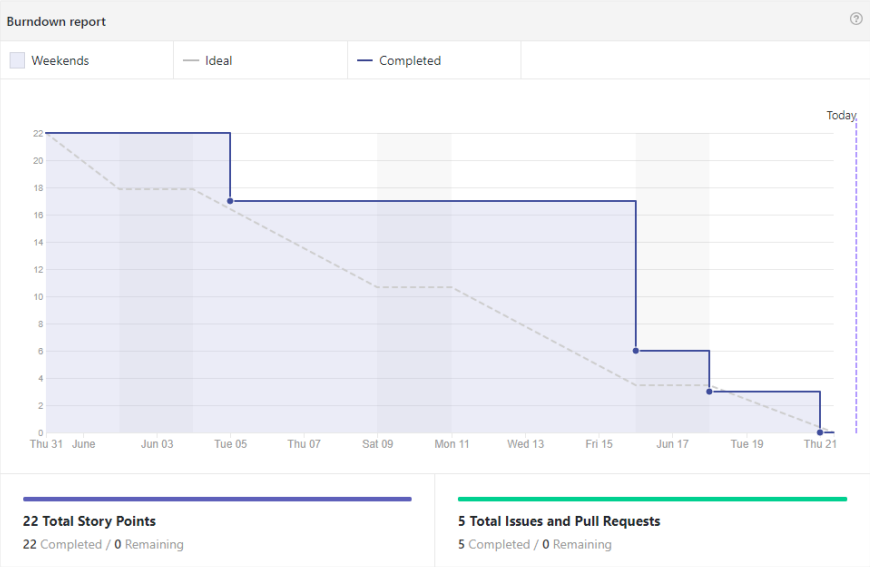

Figura A.7: Gráfico de la iteración 17.

A.3. Estudio de viabilidad

Viabilidad económica

Se ha realizado un cálculo de los costes que ha supuesto realizar este proyecto para futuras referencias de proyectos similares.

El proyecto ha contado con un único desarrollador durante 24 semanas en régimen de 15 horas semanales para la realización del diseño, la programación y el sistema de jugabilidad de la aplicación, y otras 2 semanas extra para mejoras y corrección de errores. Se ha requerido también el trabajo de un artista externo al proyecto durante 4 semanas a tiempo parcial para realizar toda la parte de modelado de los objetos 3D utilizados. Se considera para ambos un sueldo de 35€ por hora trabajada.

Además el proyecto ha contado con 2 profesores contratados durante 4 meses para este proyecto. Por cada profesor se asignan 0,5 créditos de trabajo

y se ha tenido en cuenta la tabla salarial de la Universidad de Burgos[12]. Con todo ello, el coste humano de este proyecto será aproximadamente el mostrado en las tablas A.1 y A.2.

Personal	Sueldo medio por hora	Horas de trabajo	Coste total
Desarrollador	35€	390h	13650€
Artista	35€	80h	2800€
Total	70€	470h	16.450€

Tabla A.1: Costes de personal.

Personal	Sueldo por crédito trabajado	Créditos trabajados	Coste total
Ayudante doctor	907,5€	0,5 créditos	453,75€
Doctor permanente	871,97€	0,5 créditos	435,98€
Total	1779,47€	1 crédito	889,73€

Tabla A.2: Costes del Personal Docente Investigador.

Para los costes materiales en este proyecto hay que tener en cuenta sobretodo la variedad de dispositivos móviles disponibles para utilizar con Daydream [9], ya que el coste puede variar significativamente dependiendo de la opción elegida, aunque el impacto en el rendimiento también será notable en función de la opción elegida. En los cálculos de los costes se ha supuesto una adquisición de todos los elementos hardware necesarios, incluyendo el ordenador sobre el que desarrollar, y se ha supuesto también que dicho ordenador será de características más o menos similares a las que aparecen en las figuras A.8 y A.9. Suponiendo una amortización de 5 años para el ordenador y una amortización de 3 años para el dispositivo móvil y el Daydream View, con un uso de 4 meses para ambos, el coste será aproximadamente el que aparece en la tabla A.3 y como se puede apreciar hay una gran diferencia de costes dependiendo de qué tipo de dispositivos se elijan.

Respecto a los costes de software del proyecto, se puede ver en la tabla A.4 que el único gasto producido es el de la compra de los assets de Unity necesarios para el proyecto, el resto de elementos software utilizados han sido de uso totalmente gratuitos o bajo licencias de uso gratuito.

Con estas cuentas habría que amortizar unos 17.500€ de costes del desarrollo mediante las ventas de la aplicación, pudiendo aumentar los ingresos obtenidos

[12]Según el documento de la Universidad de Burgos con los datos salariales del Personal Docente Investigador [2].

Sistema operativo:	Windows 10 Pro 64 bits (10.0, compilación 17134)
Idioma:	español (configuración regional: español)
Fabricante del sistema:	Micro-Star International Co., Ltd.
Modelo del sistema:	CX61 2PC
BIOS:	E16GDIMS.50N
Procesador:	Intel(R) Core(TM) i5-4210M CPU @ 2.60GHz (4 CPUs), ~2.6GHz
Memoria:	8192MB RAM
Archivo de paginación:	10127MB usados, 3873MB disponibles
Versión de DirectX:	DirectX 12

Figura A.8: Especificaciones técnicas del ordenador utilizado.

Nombre:	NVIDIA GeForce 820M
Fabricante:	NVIDIA
Tipo de chip:	GeForce 820M
Tipo de DAC:	Integrated RAMDAC
Tipo de dispositivo:	Dispositivo de pantalla para representación
Memoria total aprox.:	6046 MB
Memoria de pantalla (VRAM):	1990 MB

Figura A.9: Especificaciones de la tarjeta gráfica del ordenador utilizado.

Hardware	Coste óptimo aproximado	Rango de costes
PC	40€	33€ - 100€
Móvil Daydream Ready	55€	33€ - 133€
Daydream View	12€	8€ - 12€
Total	107€	74€ - 245€

Tabla A.3: Costes totales del hardware por el tiempo usado.

Software	Coste
Assets de Unity3D	55€
Total	55€

Tabla A.4: Costes software.

Concepto	Coste
Humano	17.339,73€
Hardware	107€
Software	55€
Total	17.501,73€

Tabla A.5: Costes totales del proyecto.

integrando algún tipo de publicidad entre niveles o integrando nuevo contenido a la aplicación. Por ello, eligiendo como modelo de negocio la venta exclusiva de la aplicación y suponiendo un PVP de 9,99€, se recuperaría la inversión al llegar a las 1.752 copias vendidas, cifra bastante asequible teniendo en cuenta la tendencia actual de ventas de los juegos de RV, donde las cifras de venta normales se sitúan entre las 1.000 y 2.000 copias vendidas [11].

Viabilidad legal

Todas las librerías utilizadas de Daydream y GoogleVR están bajo licencia Apache 2.0[13], la cual permite el uso comercial, distribución, modificación y uso privado, siempre y cuando haya una copia de la licencia y se comenten las modificaciones pertinentes respecto al código original.

Todos los plugins utilizados en el proyecto están bajo los términos de uso de la Asset Store de Unity Technologies, que utiliza la licencia MIT, la cual permite su uso comercial.

Respecto a la publicación en la App Store de Android, se tienen que cumplir los términos legales generales marcados por Google para la publicación de aplicaciones en dicha plataforma.

Se deja constancia que todo lo mencionado en este apartado se cumple a día de presentación de este Trabajo de Final de Grado.

[13] Se pueden comprobar los detalles de muchas licencias en este enlace: https://choosealicense.com/licenses/

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se detallarán los requisitos marcados para cumplir los objetivos buscados por este proyecto.

B.2. Objetivos generales

Tal y como se comenta en la memoria, el objetivo de este proyecto es desarrollar un videojuego de Realidad Virtual que funcione en un dispositivo móvil de gama alta que permita la accesibilidad económica a cualquier usuario que pretenda probar la Realidad Virtual. Además, el videojuego debería ser fácilmente extensible pudiendo crear nuevos niveles facilitando herramientas para ello.

B.3. Catálogo de requisitos

De los objetivos generales se pueden extraer estos requisitos.

Requisitos funcionales

- **RF-1:** Utilizar la aplicación.
  - **RF-1.1:** Visualizar toda la aplicación a través del Daydream View.
  - **RF-1.2:** Controlar la aplicación con el mando del Daydream View.
    - **RF-1.2.1:** Controlar los menús con el mando.
      - **RF-1.2.1.1:** Abrir/cerrar menú de pausa.
      - **RF-1.2.1.2:** Seleccionar nivel.
    - **RF-1.2.2:** Controlar el desplazamiento del jugador por el escenario con el mando.

        - **RF-1.2.3:** Controlar la interacción con las setas con el mando.
            - **RF-1.2.3.1:** Coger/soltar las setas.
            - **RF-1.2.3.2:** Rotar las setas.
            - **RF-1.2.3.3:** Mostrar/ocultar información de las setas.
        - **RF-1.2.4:** Interactuar con las cestas.
            - **RF-1.2.4.1:** Sumar/restar puntuación.
- **RF-2:** Diseñar la aplicación.
    - **RF-2.1:** Usar datos de setas de una base de datos.
    - **RF-2.2:** Crear nivel.
        - **RF-2.2.1:** Crear escenario.
            - **RF-2.2.1.1:** Crear y colocar árboles.
            - **RF-2.2.1.2:** Crear y colocar setas.
        - **RF-2.2.2:** Definir reglas del nivel.

Requisitos no funcionales

- **RNF-1:** Ejecutar la aplicación en un dispositivo móvil Android compatible con Daydream.
- **RNF-2:** Visualizar toda la aplicación a través de las lentes del Daydream View.
- **RNF-3:** Rendimiento. La aplicación va a ser ejecutada en un dispositivo móvil con recursos limitados y la Realidad Virtual requiere de alto rendimiento estable para evitar mareos al usuario.
- **RNF-4:** Extensibilidad. La aplicación ha de ser extensible teniendo la opción de añadir nuevos niveles.
- **RNF-5:** Mantenibilidad. La aplicación ha de ser fácilmente mantenible por otro desarrollador.
- **RNF-6:** Usabilidad. La aplicación ha de tener controles claros y sencillos, en la medida de lo posible, para garantizar la accesibilidad de cualquier persona a la Realidad Virtual.

B.4. Especificación de requisitos

Los requisitos funcionales de la aplicación se han detallado en las siguientes tablas.

RF-1.1: Visualizar toda la aplicación a través del Daydream View.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de visualizar toda la aplicación a través de las lentes del Daydream View.
Precondiciones	Ninguna.
Acciones	El usuario puede mover su cabeza en cualquier dirección.
Postcondiciones	La posición y la orientación de la cámara será la misma que la de la cabeza del jugador.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.1: RF-1.1: Visualizar el simulador a través del Daydream View.

RF-1.2: Controlar la aplicación con el mando del Daydream View.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de controlar toda la aplicación con el mando del Daydream View.
Precondiciones	Ninguna.
Acciones	El usuario puede utilizar el mando para interactuar con la interfaz, moverse por el escenario e interactuar con las setas.
Postcondiciones	La interfaz habrá respondido adecuadamente, el usuario se habrá desplazado o la interacción con la seta será adecuada.
Excepciones	Ninguna.
Importancia	Muy alta

Tabla B.2: RF-1.2: Controlar la aplicación con el mando del Daydream View.

RF-1.2.1: Controlar los menús con el mando.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de controlar los menús de la interfaz con el mando.
Precondiciones	La aplicación debe estar en funcionamiento. La interfaz debe estar visible en pantalla.
Acciones	El usuario apunta con el mando a los elementos de la interfaz y utiliza los botones del mando para interactuar.
Postcondiciones	El elemento responde, primero sobresaliendo de la interfaz cuando es apuntado, y después realizando la acción que lleve asociada.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.3: RF-1.2.1: Controlar los menús con el mando.

RF-1.2.1.1: Abrir/cerrar menú de pausa.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de abrir y cerrar el menú de pausa.
Precondiciones	La aplicación debe estar en funcionamiento.
Acciones	El usuario presiona el botón central inferior del mando.
Postcondiciones	El menú de pausa aparece o desaparece dependiendo de cómo se encontrase previamente.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.4: RF-1.2.1.1: Abrir/cerrar menú de pausa.

RF-1.2.1.2: Seleccionar nivel.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de seleccionar un nivel para cargarlo.
Precondiciones	La aplicación debe estar en funcionamiento. Debe estar visible el menú de pausa/selección de niveles.
Acciones	El usuario apunta al botón correspondiente al nivel que quiera cargar y presiona el botón del panel táctil del mando.
Postcondiciones	Se cambia momentáneamente a una pantalla de carga y se carga el nivel seleccionado.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.5: RF-1.2.1.2: Seleccionar nivel.

RF-1.2.2: Controlar el desplazamiento del jugador por el escenario con el mando.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de desplazarse por el escenario mediante teletransporte.
Precondiciones	Debe estar cargado un nivel. Debe estar activo el modo de desplazamiento.
Acciones	El usuario apunta con el mando para seleccionar la zona donde teletransportarse y presiona el botón del panel táctil para finalizar el desplazamiento.
Postcondiciones	El usuario se habrá movido instantáneamente a la posición apuntada.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.6: RF-1.2.2: Controlar el desplazamiento del jugador por el escenario con el mando.

RF-1.2.3: Controlar la interacción con las setas con el mando.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe ser capaz de interactuar con las setas, cogiéndolas y mostrando información de las mismas.
Precondiciones	Debe estar cargado un nivel. Debe estar activo el modo de interacción.
Acciones	El usuario apunta con el mando a la seta con la que quiere interactuar y utiliza diferente controles del mando para coger, soltar y rotar la seta, y mostrar u ocultar información.
Postcondiciones	El usuario habrá podido interactuar con la seta seleccionada.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.7: RF-1.2.3: Controlar la interacción con las setas con el mando.

RF-1.2.3.1: Coger/soltar las setas y desplazarlas.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe poder coger las setas y desplazarlas mientras las tiene cogidas. También debe poder soltar las setas cuando las tenga cogidas.
Precondiciones	Debe estar cargado un nivel. Debe estar activo el modo de interacción.
Acciones	El usuario apunta con el mando a la seta que quiera coger o soltar y presiona el botón del panel táctil del mando.
Postcondiciones	La seta pasará a estar atada a distancia al mando, siguiendo los movimientos de este.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.8: RF-1.2.3.1: Coger/soltar las setas y desplazarlas.

RF-1.2.3.2: Rotar las setas.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe poder rotar la setas sobre su eje cuando la tiene cogida.
Precondiciones	Debe estar cargado un nivel. El usuario debe tener una seta cogida
Acciones	El usuario coloca el dedo sobre uno de los laterales del panel táctil del mando.
Postcondiciones	La seta irá rotando sobre su eje mientras se mantenga el dedo en uno de los laterales del panel táctil.
Excepciones	Ninguna.
Importancia	Media

Tabla B.9: RF-1.2.3.2: Rotar las setas.

RF-1.2.3.3: Mostrar/ocultar información de las setas.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe poder mostrar u ocultar la información relativa a una seta que tenga cogida.
Precondiciones	Debe estar cargado un nivel. El usuario debe tener una seta cogida
Acciones	El usuario presiona el botón central del mando.
Postcondiciones	Un panel con la información relevante de la seta aparecerá o desaparecerá justo encima de esta.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.10: RF-1.2.3.3: Mostrar/ocultar información de las setas.

RF-1.2.4: Interactuar con las cestas.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe poder colocar e interactuar con las cestas.
Precondiciones	Debe estar cargado un nivel.
Acciones	El usuario activa el modo de interacción.
Postcondiciones	Las dos cestas aparecerán en el suelo, a los laterales del usuario, permitiendo la interacción con ellas.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.11: RF-1.2.4: Interactuar con las cestas.

RF-1.2.4.1: Sumar/restar puntuación.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El usuario debe poder sumar/restar puntuación al echar setas en las cestas.
Precondiciones	Debe estar cargado un nivel. Deben estar las cestas visibles y colocadas en el suelo.
Acciones	El usuario echa una seta a la apertura de una de las cestas.
Postcondiciones	En función de si la seta era correcta para esa cesta, se habrá sumado o restado puntación al usuario.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.12: RF-1.2.4.1: Sumar/restar puntuación.

RF-2.1: Usar datos de setas de una base de datos.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El desarrollador debe poder usar datos sacados de una base de datos de setas desde el editor de Unity.
Precondiciones	Ninguna
Acciones	El desarrollador utiliza la herramienta de consulta de la base de datos introduciendo el nombre de una seta.
Postcondiciones	La base de datos es leída y se genera el contenedor de datos de la seta.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.13: RF-2.1: Usar datos de setas de una base de datos.

RF-2.2.1: Crear escenario.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El desarrollador debe poder crear nuevos escenarios desde el editor de Unity de forma sencilla.
Precondiciones	Debe existir un terreno ya en el nivel.
Acciones	El desarrollador utiliza la herramienta de creación de escenarios con los parámetros deseados.
Postcondiciones	Se genera un escenario con los árboles y setas deseados.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.14: RF-2.2.1: Crear escenario.

RF-2.2.1.1: Crear y colocar árboles.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El desarrollador debe poder crear árboles en el nivel.
Precondiciones	Debe existir un terreno ya en el nivel.
Acciones	El desarrollador utilizará la herramienta de creación de terrenos definiendo aspectos de los árboles que se generarán, como el número de estos o el radio de generación.
Postcondiciones	Se generarán árboles en el nivel según los parámetros definidos.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.15: RF-2.2.1.1: Crear y colocar árboles.

RF-2.2.1.2: Crear y colocar setas.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El desarrollador debe poder crear setas en el nivel.
Precondiciones	Debe existir un terreno ya en el nivel.
Acciones	El desarrollador utilizará la herramienta de creación de terrenos definiendo aspectos de las setas que se generarán, como el número o la probabilidad de colocación por especies.
Postcondiciones	Se generarán setas en el nivel según los parámetros definidos.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.16: RF-2.2.1.2: Crear y colocar setas.

RF-2.2.2: Definir reglas del nivel.	
Versión	1.0
Autor	Rodrigo Jurado Pajares
Descripción	El desarrollador debe poder definir las reglas de los niveles desde el editor de Unity.
Precondiciones	Debe existir un gestor de niveles en el nivel editado.
Acciones	El desarrollador definirá las reglas adecuadas en la herramienta integrada en el gestor de niveles.
Postcondiciones	El nivel editado contará con las reglas definidas cuando se ejecute.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.17: RF-2.2.2: Definir reglas del nivel.

Apéndice C

Especificación de diseño

C.1. Introducción

En esta sección se describen aspectos relacionados con el diseño de la aplicación. Se comentarán aspectos como la estructura de datos o el flujo de las interfaces.

C.2. Diseño de datos

En esta sección se mostrará como están implementados los datos en la base de datos SQLite y a continuación, se mostrarán los diagramas de clases que detallan la estructura de datos de los contenedores de datos usados.

Tablas de la base de datos

Para guardar la información de las diferentes especies de setas de la aplicación se han creado las tablas de la figura C.1

- TablaComestible: Almacena la comestibilidad de una determinada especie en español e inglés.
- TablaDescripciones: Almacena la descripción de una determinada especie en español e inglés.
- TablaGeneros: Almacena el género al que pertenece una especie.
- TablaEnlaces: Almacena la url de la *Wikipedia* de la especie.

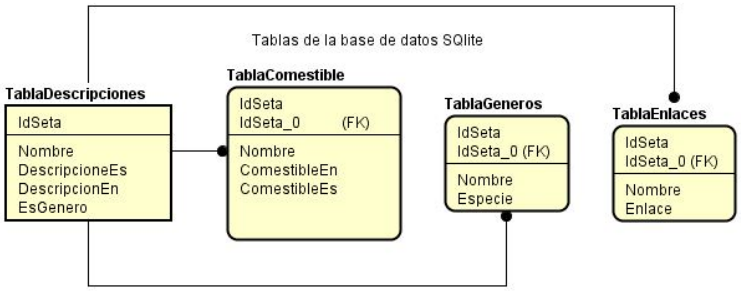

Figura C.1: Tablas de la base de datos SQLite.

Estructuras de datos

La información de la base de datos se tiene que poder usar en la aplicación, y para ello se ha creado una clase en C# que contendrá una equivalencia de los campos de las tablas de la base de datos. El campo de comestibilidad de la base de datos es una cadena de texto, una estructura de datos poco útil en el código para lo que se quería hacer, así que ese campo se dividió en varias de las opciones posibles de comestibilidad que se habían visto en la base de datos, pasando a tener de esta forma booleanos en la estructura de datos. Este procedimiento, aunque poco escalable, permite que la estructura de datos utilizada en el código sea más útil, simplificando las comparaciones que se realizan. la clase creada no utiliza todos los campos disponibles de la base de datos, pero si los mínimos necesarios para tener información relevante de las setas en español, pudiendo realizar en un futuro una ampliación de la clase para permitir la internacionalización.

Diagramas de clases

Todo el código propio realizado se ha agrupado bajo un espacio de nombres concreto, *UBUSetasVR*, que cuenta con espacios de nombres anidados. De esta forma se ha intentado realizar una agrupación lógica del código según la funcionalidad general del código. Esta agrupación permite obtener datos del espacio de nombres, como en el caso de las métricas de código[1], y además permite generar un diagrama de clases en Visual Studio. En la figura C.3 se puede ver el diagrama de clases del espacio de nombres *UBUSetasVR*. Como se puede apreciar, las clases aparecen inconexas y solo se ve una única herencia de clase dentro del espacio de nombres, lo que refleja lo ya comentado en la introducción del apartadoA.1 sobre la modularidad y las clases inconexas. Destacar también que todas las clases utilizadas dentro del editor de Unity deben heredar de

[1] Ver sección “Métricas de código” del apartado “Técnicas y herramientas” de la memoria.

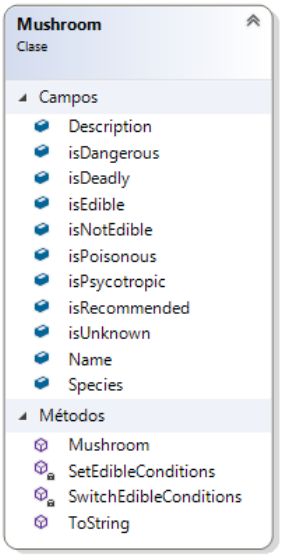

Figura C.2: Diagrama de clase de la estructura de datos de las setas.

la clase *MonoBehaviour*[2] para que funcionen como *GameObjects*. No se han incluido diagramas de clases de las librerías ni de los plugins utilizados.

Figura C.3: Diagrama de clases propias del proyecto.

Los espacios de nombres anidados al espacio raíz, el cual contiene el código de uso general, son 6: *Database*, *EditorScripts*, *Managers*, *Placement*, *ScriptableObjects* y *UI*.

En la figura C.4 se muestra el espacio de nombres *UBUSetasVR.Database*, el cual tiene otro espacio de nombres anidado, *DataStructures*. Estos espacios

[2]Esta clase es la principal para trabajar en Unity, ya que cualquier script que no herede de ella no puede ser utilizado en el editor como un componente. Ver secciones "GameObjects" y "Componentes" del apartado "Conceptos teóricos" de la memoria.

de nombres contendrán lo necesario para acceder a la base de datos y guardar la información de esta en estructuras de datos.

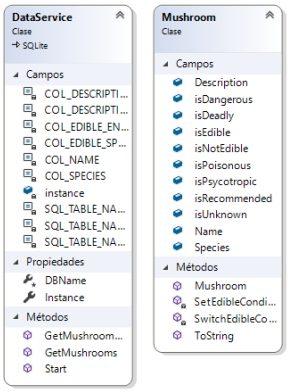

Figura C.4: Diagrama de clases del espacio *UBUSetasVR.Database*.

En la figura C.5 se muestra el espacio de nombres *UBUSetasVR.EditorScripts*[3], el cual contiene todo el código de ayuda que será ejecutado en el editor de Unity durante el desarrollo de la aplicación, como por ejemplo los inspectores del editor [17] modificados o las herramientas de acceso a la base de datos.

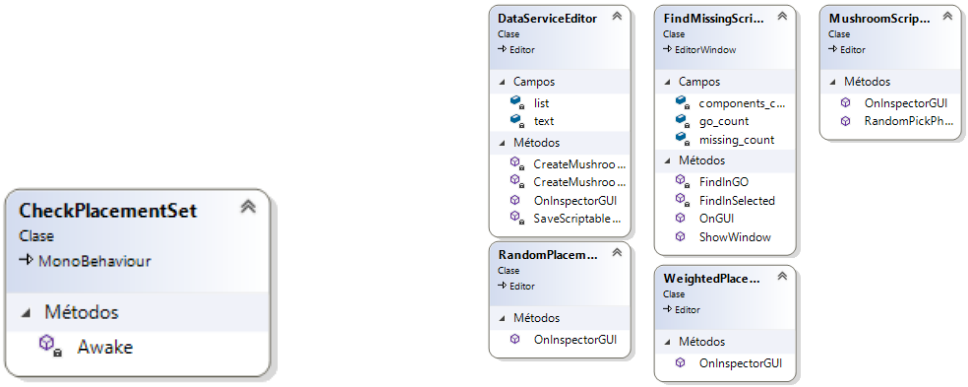

Figura C.5: Diagramas de clases del espacio *UBUSetasVR.EditorScripts*.

En la figura C.6 se muestra el espacio de nombres *UBUSetasVR.Managers*, el cual agrupa todos los gestores propios utilizados en la aplicación[4].

En la figura C.7 se muestra el espacio de nombres *UBUSetasVR.Placement*, el cual agrupa las clases utilizadas para la colocación automática en el escenario

[3]En Visual Studio Unity crea una solución con dos proyectos, uno para los scripts de modificación del editor y otro para el código de Unity en general, de ahí que este espacio de nombres aparezca dividido en dos diagramas de clases

[4]Ver sección "Managers" del apartado "Técnicas y herramientas" de la memoria.

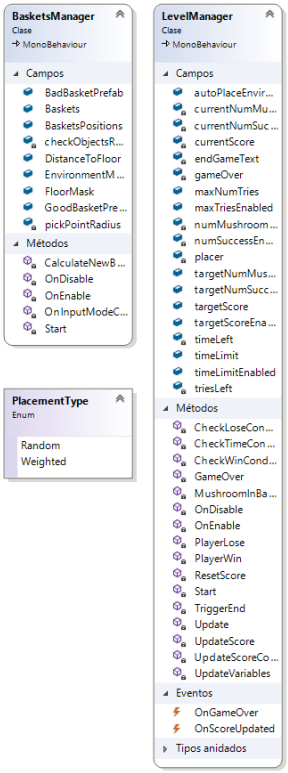

Figura C.6: Diagrama de clases del espacio *UBUSetasVR.Managers*.

de los elementos del entorno en base a ciertos parámetros editables desde el editor de Unity.

En la figura C.8 se muestra el espacio de nombres *UBUSetasVR.ScriptableObjects*, el cual contiene las clases necesarias para generar los contenedores de datos[5] que serán utilizados desde el editor.

En la figura C.9 se muestra el espacio de nombres *UBUSetasVR.UI*, el cual agrupa todo el código propio relacionado con la interfaz de usuario o de elementos de representación planos.

C.3. Diseño arquitectónico

La aplicación está diseñada para funcionar en base a eventos relacionados con la interacción del usuario en el escenario y con los sistemas de la aplicación, por ello sólo se ha plasmado en este anexo un único diagrama de secuencia por considerar que el resto de interacciones son muy similares a esta, salvando particularidades.

[5] Ver sección "ScriptableObjects" del apartado "Conceptos teóricos" de la memoria.

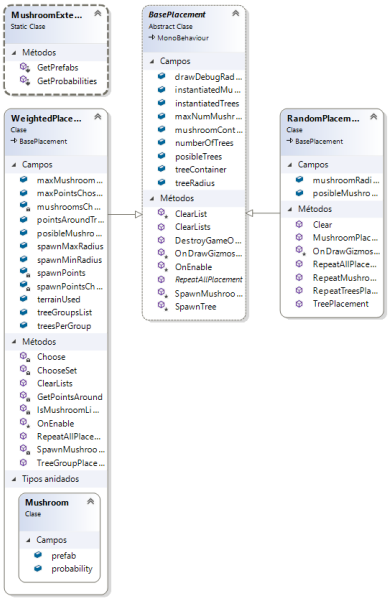

Figura C.7: Diagrama de clases del espacio *UBUSetasVR.Placement*.

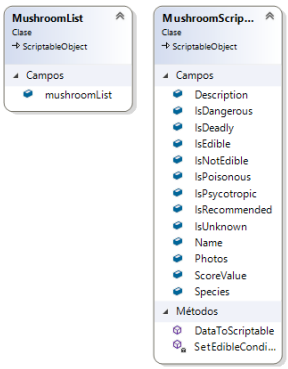

Figura C.8: Diagrama de clases del espacio *UBUSetasVR.ScriptableObjects*.

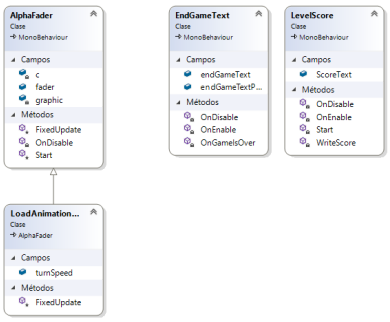

Figura C.9: Diagrama de clases del espacio *UBUSetasVR.UI*.

La figura C.10 representa el diagrama de secuencia para el proceso de actualizar la puntuación cuando se ha cogido una seta. En este proceso, el jugador coge y suelta una seta sobre una cesta, lanzando un evento *OnMushroomInBasket()*. Este evento será recibido por la propia cesta, que guardará la información de la seta y lanzará otro evento *OnMushroomInBasket()* con la información necesaria. Tras esto, el *LevelManager* escuchará y procesará el evento, actualizando internamente la puntuación y lanzando un evento para actualizar la puntuación en la interfaz correspondiente.

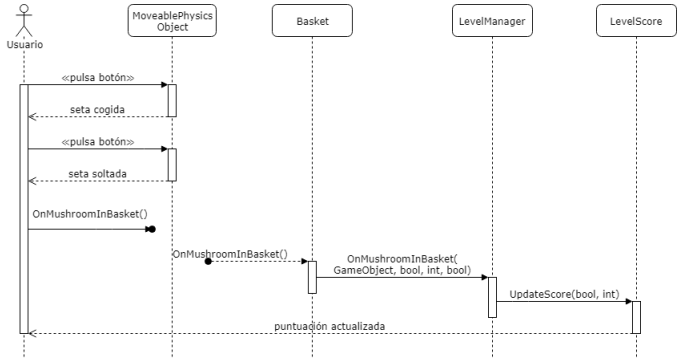

Figura C.10: Diagrama de secuencia para el proceso de actualizar puntuación.

C.4. Diseño de interfaces

Para el diseño de interfaces de la aplicación se ha seguido en la medida de lo posible el *Manifiesto de diseño de interfaces en RV* [1], el cual aporta información muy relevante a la hora de diseñar interfaces cómodas y útiles en RV para el usuario final. No obstante, el diseño gráfico de interfaces no ha sido prioritario en el desarrollo de la aplicación, pudiendo ser un área importante de mejora en iteraciones futuras.

En la figura C.11 se puede ver un esquema de las interfaces diseñadas para la aplicación. Solo tienen relación entre sí las interfaces que cuentan con un primer y un segundo nivel de profundidad. Por otro lado, en la figura C.12 se puede ver el flujo de aparición de las interfaces desde el inicio de la aplicación y durante la ejecución del juego.

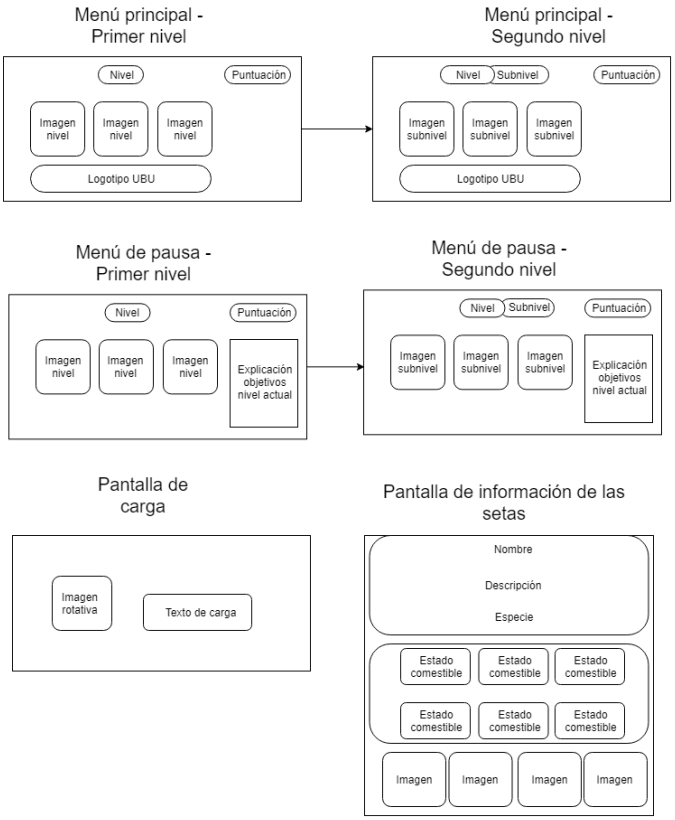

Figura C.11: Diseño de las interfaces del juego.

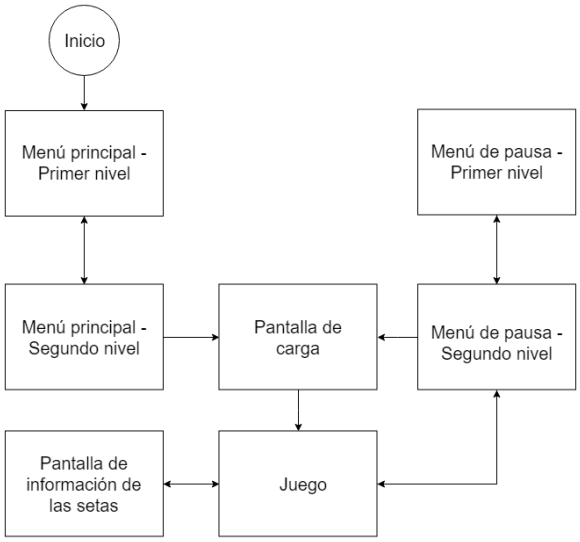

Figura C.12: Flujo de las interfaces del juego durante su ejecución.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se encuentra toda la documentación relacionada con la estructura del proyecto, así como de su funcionamiento y su ejecución.

D.2. Estructura de directorios

En este apartado se va a detallar la estructura de directorios presentes en la carpeta "Assets" que genera el editor de Unity, que es donde se encuentran todos los elementos relevantes de este proyecto, ya que el resto son carpetas de librerías propias de Unity. En la figura D.1 se puede ver una representación gráfica de la estructura de carpetas.

- **/Art** Todos los archivos propios relacionados con el arte y elementos gráficos.
  - **/Art/2D** Archivos gráficos.
    - **/Art/2D/Mushrooms** Fotografías reales (.jpg) de las especies de setas.
    - **/Art/2D/UI** Elementos gráficos para las interfaces de la aplicación.
  - **/Art/Materials** Materiales (ya en formato de Unity) de uso general para aplicar sobre los modelos.
  - **/Art/Models** Modelos de todos los objetos.
    - **/Art/Models/Mushrooms** Modelos de las setas.

```
Assets
├── Art
│   ├── 2D
│   ├── Materials
│   ├── Models
│   │   ├── Mushrooms
│   │   │   └── Materials
│   │   └── Trees
│   ├── Terrains
│   │   └── Meshes
│   └── Textures
│       └── Mushrooms
├── Data
│   ├── DB
│   ├── Mushrooms
│   └── UI
│       └── LevelSelect
├── DaydreamElements
├── DaydreamRenderer
├── Extended Colliders 3D
├── GoogleVR
├── Plugins
│   ├── Android
│   └── SQLiter
├── Prefabs
│   ├── Mushrooms
│   │   └── Low Poly
│   └── Trees
├── Quantum Theory
├── Scenes
├── Scripts
│   ├── DataStructure
│   ├── Editor
│   ├── Managers
│   ├── Placement
│   ├── ScriptableObjects
│   └── UI
└── TextMesh Pro
```

Figura D.1: Estructura de directorios del proyecto

        - **/Art/Models/Mushrooms/Materials** Materiales de los modelos de las setas.
    - **/Art/Models/Trees** Modelos de los árboles.
  - **/Art/Terrains** Datos de los terrenos de Unity.
    - **/Art/Terrains/Meshes** Modelos de los terrenos "Low Poly" generados con Polyworld.
  - **/Art/Textures** Texturas de uso general para ser utilizadas en los materiales.
    - **/Art/Textures/Mushrooms** Texturas de las setas.
- **/Data** Directorio que contiene los aspectos relacionados con los datos de las setas y los niveles.
  - **/Data/DB** Base de datos de las setas.
  - **/Data/Mushrooms** Directorio con todos los *ScriptableObjects* con la información relevante de cada seta utilizada.
  - **/Data/UI** Datos relativos a la interfaz de usuario.
    - **/Data/UI/LevelSelect** *ScriptableObjects* con datos relativos a los botones de carga de niveles.
- **/DaydreamElements** Carpeta de la librería de Daydream Elements que proporciona elementos específicos de RV avanzados como la gestión del movimiento del controlador o animaciones de las interfaces.
- **/DaydreamRenderer** Carpeta de la librería de Daydream Renderer que proporciona utilidades avanzadas de iluminación (no usadas en este proyecto, pero la librería es necesaria).
- **/Extended Colliders 3D** Carpeta del plugin Extended Colliders 3D.
- **/GoogleVR** Carpeta de la librería de GoogleVR que proporciona los elementos básicos necesarios para implementar RV en Unity.
- **/Plugins** Carpeta con datos generados por los plugins.
  - **/Plugins/Android** Carpeta con un manifiesto de Android modelo.
  - **/Plugins/SQLiter** Carpeta del plugin SQLiter.
- **/Prefabs** Contiene todos los prefabs del proyecto[1].
  - **/Prefabs/Mushrooms** Prefabs de las setas que pueden ser utilizadas en los niveles.
  - **/Prefabs/Trees** Prefabs de los árboles utilizados en los niveles.
- **/Quantum Theory** Carpeta del plugin PolyWorld: Woodland Low Poly Toolkit.
- **/ResonanceAudio** Carpeta del plugin Resonance Audio.

[1] Los prefabs son plantillas de GameObjects que simplifican el trabajo en el editor de Unity [19]

- **/Scenes** Carpeta con todas las escenas del proyecto. Puede contener carpetas con los nombres de las escenas si estas tienen elementos de iluminación precocinados.
- **/Scripts** Código propio de la aplicación.
  - **/Scripts/DataStructure** Scripts con las estructuras de datos equivalentes a las tablas de la base de datos.
  - **/Scripts/Editor** Scripts que modifican el editor de Unity y proporcionan herramientas personalizadas sobre otros scripts propios.
  - **/Scripts/Managers** Scripts que controlan el funcionamiento de varios sistemas de la aplicación.
  - **/Scripts/Placement** Scripts que manejan la colocación de los objetos en el escenario.
  - **/Scripts/ScriptableObjects** Scripts que manejan las estructuras de datos de los "ScriptableObjects" creados.
  - **/Scripts/UI** Scripts que controlan el comportamiento de la interfaz de usuario.
- **/TextMesh Pro** Carpeta del plugin TextMesh Pro.

D.3. Manual del programador

Para poder editar este proyecto son necesarias principalmente dos cosas: tener una versión de Unity3D instalada y compatible, y tener un editor de C# soportado por Unity3D. Para el primer requisito, se necesitará utilizar una versión de Unity3D igual o superior a la 2017.2.1 cumpliendo con los requisitos mínimos especificados en la tabla D.1, aunque preferiblemente se utilizará la versión 2017.2.1p4 que incluye bastantes correcciones de errores, siendo obligatorio usar esta versión como mínimo si se va a utilizar el plugin de sonido espacial Resonance Audio [10].

Si se va a compilar para Android, como es el caso de este proyecto, se necesitará además instalar el JDK de Java [12] y el SDK de Android [3] correspondiente a la versión para la que queramos compilar, aunque esto se explicará con más detalle en la siguiente sección.

Para el segundo requisito, el propio instalador de Unity3D nos dará opciones para descargar e instalar el editor MonoDevelop y el editor Visual Studio Community Edition, siendo recomendable instalar este último[2]. Teniendo ambos elementos instalados se podrá proceder a clonar o descargar el proyecto del repositorio de GitHub [13], donde estará el código fuente de forma pública.

[2]Ver sección "Entorno de programación" del apartado "Técnicas y herramientas" de la memoria.

Requisitos del sistema para desarrollar con Unity.	
Procesador	Cualquiera con soporte para el conjunto de instrucciones SSE2. Recomendados de la gama Intel i3 en adelante
Gráficos	Tarjeta de vídeo compatible con DX10 (shader modelo 4.0). Recomendadas tarjetas con mínimo 2GB de VRAM.
Memoria	4GB RAM. Recomendados 8GB de RAM.
Disco duro	Dependerá del tamaño del proyecto en disco. Recomendados 500GB.
Android en Unity	Android SDK (API 24 o superior) y Java Development Kit (JDK 8).

Tabla D.1: Requisitos del sistema para desarrollar con Unity

Con todo lo necesario en el ordenador para abrir el proyecto, se abre Unity, se elige abrir proyecto y se selecciona la carpeta del proyecto descargado o clonado. En el repositorio se encuentran solo los elementos mínimos necesarios, por lo que Unity se tomará un tiempo para regenerar los archivos de las librerías y recursos necesarios para el proyecto a partir de los archivos ya existentes. Este proceso será necesario una única vez.

D.4. Compilación, instalación y ejecución del proyecto

Compilación en Unity

Un proyecto en Unity puede ser desarrollado para multitud de plataformas, por lo que para realizar una compilación hay que seleccionar primero qué plataforma tiene por objetivo el proyecto. Por defecto, Unity compilará para la plataforma "Windows Satandlone" por ser la única para la que tiene las herramientas de compilación tras una instalación mínima. Para cambiar la plataforma objetivo hay que ir al menú "Build Settings", accesible desde el menú "File" situado en la parte superior izquierda de la ventana principal de Unity una vez se ha abierto el proyecto, tal y como se puede ver en la figura D.2. En este menú, que aparece en la figura D.3, se podrá seleccionar entre otras cosas la plataforma para la que compilar con el botón "Switch Platforms" y se podrá realizar una compilación completa para generar un ejecutable del proyecto, lo que se conoce como generar una *build*. Destacar que cuando se produce un cambio de plataforma Unity fuerza una reimportación de todos los assets del proyecto y tardará unos minutos, dependiendo del tamaño del proyecto, en compilar de nuevo.

Si el proyecto tiene por objetivo la plataforma Android, Unity necesitará tener la ruta a la carpeta de instalación del SDK de Android para poder

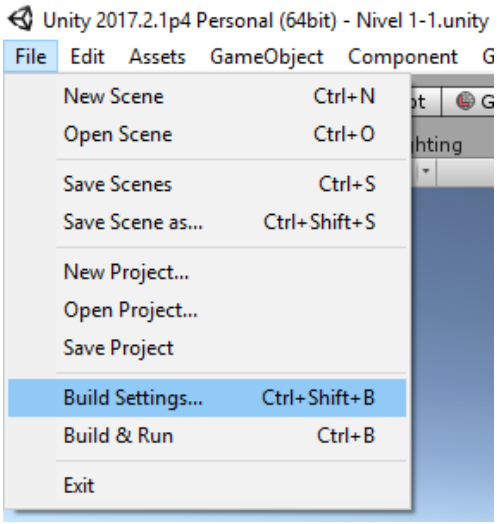

Figura D.2: Ubicación del botón para abrir las opciones de la *build*.

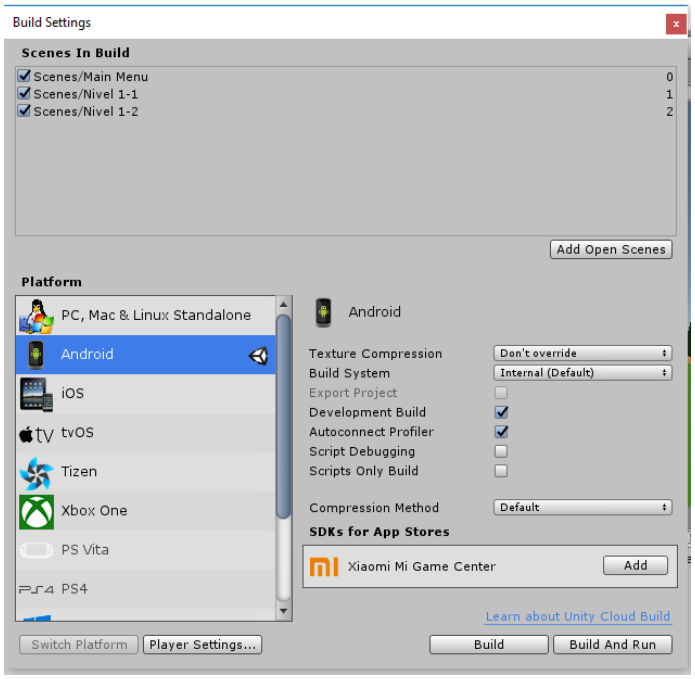

Figura D.3: Vista del menú de opciones de la *build*.

realizar una compilación. La primera vez que se intente hacer una *build* Unity sacará una ventana emergente pidiendo dicha ruta, pero es posible indicársela con antelación en la sección "External Tools" del menú "Preferences" como se ve en la figura D.5, accesible desde el menú "Edit" ubicado en la parte superior izquierda, como se puede ver en la figura D.4. Desde este menú se podrá indicar también la ruta al JDK que se haya instalado previamente. Es importante que en el SDK descargado se encuentren las *build tools* correspondientes a la versión de Android para la que se quiera compilar, o en su defecto, de alguna versión superior, aunque en este último caso se puede generar algún problema en la compilación por las herramientas de compatibilidad del propio SDK.

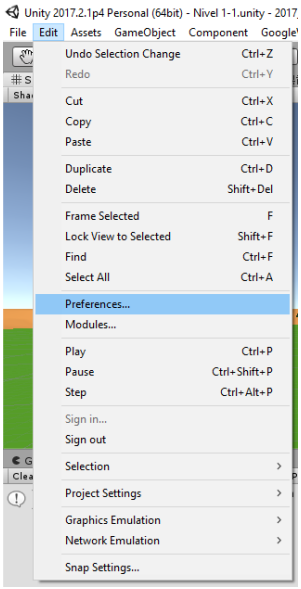

Figura D.4: Ubicación del botón para abrir las opciones de personalización del editor.

Cuando se hace una *build* para Android, Unity genera un ejecutable, un fichero *apk*, y le asigna un nombre de paquete[3] definido previamente. Unity fuerza a que este nombre de paquete sea válido y diferente al nombre que pone por defecto el editor antes de generar el ejecutable. El nombre de paquete, así como otras muchas opciones de configuración del proyecto como la versión de Android objetivo o la rotación de pantalla utilizada, se pueden modificar en el menú "Player Settings", accesible desde el menú "Project Settings", ubicado dentro del menú "Edit", tal y como se ve en la figura D.6. En la sección "Other Settings", como se ve en la figura D.7, se puede introducir el nombre válido del paquete. También se puede ver en esta sección el desplegable para seleccionar la versión mínima de Android que será necesaria para ejecutar la aplicación.

[3]Un nombre de paquete en Android identifica inequívocamente a una aplicación en un sistema Android y sigue una nomenclatura específica

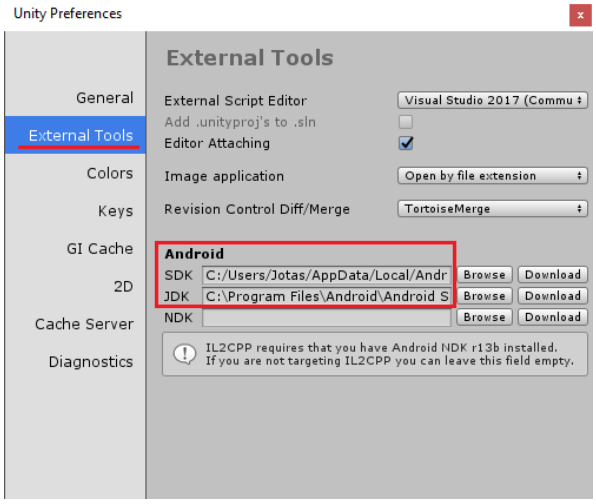

Figura D.5: Vista del menú de opciones de personalización y de los campos donde introducir las rutas necesarias para la plataforma Android.

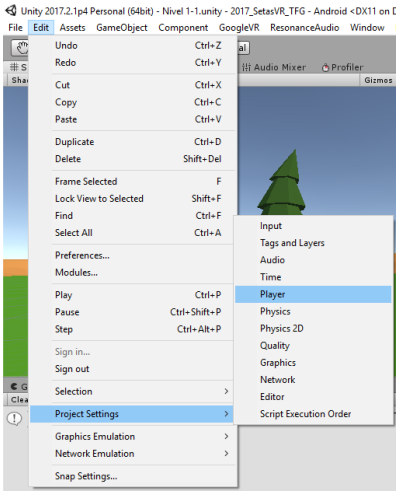

Figura D.6: Ubicación del botón para abrir el menú *Player Settings*.

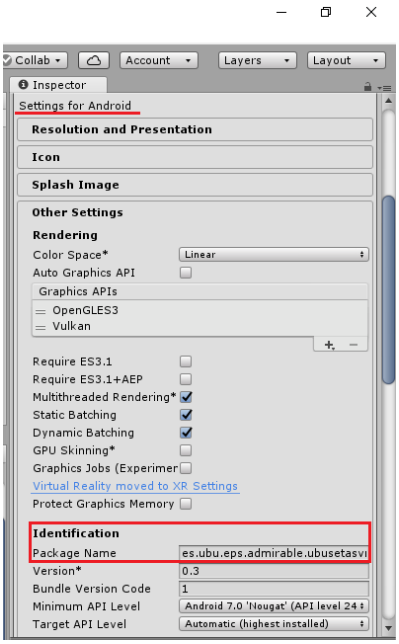

Figura D.7: Vista del menú *Player Settings* con opciones generales de la aplicación.

Instalación en Android

Una vez que el archivo *apk* ha sido generado, hay que copiarlo en el dispositivo móvil correspondiente e instalarlo. Es posible que en el móvil salte una alerta por intentar instalar una *apk* que no procede de la App Store[4], la cual se puede evitar con la opción de "Instalar apps desconocidas" presente en los ajustes de seguridad del dispositivo. En la figura D.8 se ve la ubicación de esta opción en un Samsung Galaxy S8, aunque el nombre y la ubicación exacta puede cambiar en función del dispositivo usado[5].

Con esto se habrá instalado el ejecutable del proyecto en el dispositivo móvil, pero es posible que todavía no funcione correctamente porque pueden faltar dos aplicaciones complementarias y necesarias, ambas accesibles desde la App Store[6]: los servicios de RV de Google [8] y Daydream [5], visibles en la figura D.9. La primera es una aplicación que instala todo el ecosistema de librerías generales necesarias para que la RV de Google funcione en el dispositivo, tanto Daydream como Cardboard [4]. La segunda aplicación es de

[4]La App Store es el mercado digital de aplicaciones de la plataforma Android y pertenece a Google.

[5]En versiones anteriores a Android Oreo (8.0) la opción para instalar apps desconocidas se llama "Permitir fuentes desconocidas"

[6]Si alguna de estas aplicaciones no aparece en la App Store, es posible que el dispositivo móvil no sea compatible con la Realidad Virtual o con Daydream.

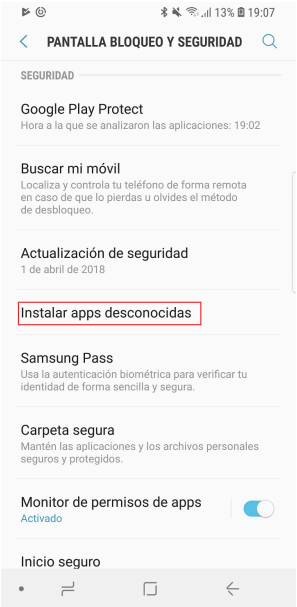

Figura D.8: Ubicación de la opción para instalar apps desconocidas.

uso íntegro con el Daydream View [7] y, además de instalar librerías necesarias, permite explorar e instalar aplicaciones móviles o de RV compatibles con Daydream.

Figura D.9: Aplicaciones de la *Play Store* necesarias para que Daydream funcione en el dispositivo móvil.

Ejecución en Android

Con todo lo necesario instalado en el móvil, será necesario emparejar el mando del Daydream View con el dispositivo móvil para poder iniciar y manejar la aplicación cuando se ejecute. Para ello, hay que activar el Bluetooth del dispositivo móvil desde los ajustes de conectividad y mantener pulsado el botón de inicio del controlador de Daydream, que desde una vista superior, se corresponde con el botón situado en la parte central e inferior del mando[7]. El paso de emparejamiento solo será necesario la primera vez, produciéndose la conexión entre ambos dispositivos de forma automática el resto de ocasiones en cuanto se habilite el Bluetooth del dispositivo móvil.

Una vez se tenga el mando emparejado, solo quedará ejecutar la aplicación y colocar el dispositivo móvil apropiadamente en el casco Daydream View como se indica en la pantalla y se puede ver en la figura D.10. Si el controlador se encuentra apagado, en la pantalla aparecerán a continuación unas instrucciones de cómo activarlo de nuevo.

Figura D.10: Pantalla que muestra cómo colocar el móvil en el visor.

D.5. Pruebas del sistema

En un proyecto de estas características es muy importante ir probando y depurando el código que se va generando, y gana mucha más importancia con la ausencia de pruebas automatizadas[8], aunque sea probando de forma manual.

En esta sección se van a comentar diferentes técnicas y herramientas para probar y depurar manualmente la aplicación en el entorno de Unity.

[7]Se entiende por parte inferior del mando a la contraria donde se encuentra la parte táctil del mando.

[8]Ver sección “Testing automatizado” del apartado “Técnicas y herramientas” de la memoria.

Ejecución en Unity

Unity permite ejecutar la aplicación en el editor en cualquier momento, siempre que no existan errores de compilación, pudiendo ver un resultado bastante aproximado del funcionamiento del proyecto cuando este se encuentre desplegado en el dispositivo móvil. Para ello, en la parte superior central de la ventana principal del editor se encuentran los controles de ejecución, visibles en la figura D.11. El primer botón empezando por la izquierda iniciará la ejecución de la escena activa[9] en ese momento, el segundo la pausará si se encuentra en ejecución, y el último botón permitirá ir avanzando la secuencia de ejecución de la escena *frame a frame*[10], o paso a paso, si la escena se encuentra pausada.

Figura D.11: Botones del editor de Unity para controlar la ejecución de la aplicación.

A primera vista, uno de los inconvenientes de ejecutar el proyecto en Unity puede parecer que es que no se disponen de elementos físicos para simular el movimiento del casco o del mando, pero lo cierto es que en las librerías de Daydream para Unity existen una serie de componentes[11] que permiten simular los movimientos de ambos dispositivos usando la combinación de teclado y ratón, pudiendo ver en la figura D.12 dichos componentes con la descripción de los controles. Existe la posibilidad incluso de utilizar un dispositivo móvil que cuente con acelerómetro y giroscopio para simular el control del mando de Daydream [6].

Consola de Unity

Unity cuenta con una consola [21] para mostrar registros por pantalla, tanto como cuando se está ejecutando la aplicación como cuando se está editando, de cualquier elemento que esté funcionando en el editor. Gracias a esto será una de las herramientas principales del desarrollador, ya que le permite conocer información relevante de los elementos en funcionamiento. Esta consola filtra los registros en tres categorías o niveles para facilitar su visualización en pantalla: mensajes, alertas y errores, tal y como se ve en la figura D.13. Existe una clase implementada en Unity que permite manipular esta consola en el código, permitiendo así sacar registros personalizados e incluso categorizados por el propio desarrollador, facilitando enormemente el desarrollo.

[9] Ver sección "Escenas" del apartado "Conceptos teóricos" de la memoria.
[10] Ver sección "Frames por segundo" en el apartado "Conceptos teóricos" de la memoria.
[11] Ver sección "Componentes" del apartado "Conceptos teóricos" de la memoria.

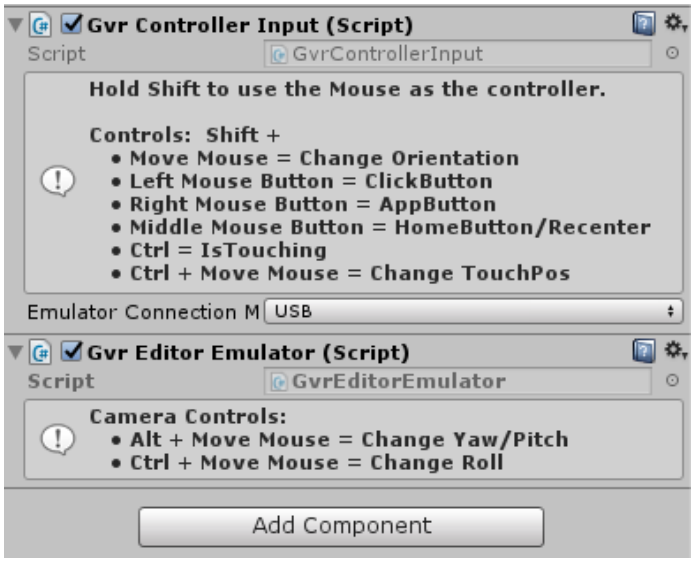

Figura D.12: Componentes necesarios para simular en el editor el control del mando de Daydream con teclado y ratón.

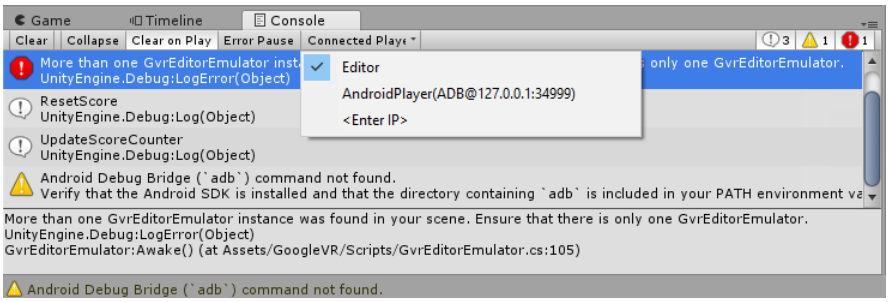

Figura D.13: Ejemplo de la consola de Unity en funcionamiento.

Esta consola es bastante versátil, permitiendo incluso mostrar registros de una aplicación de Unity que esté en funcionamiento en otra instancia del programa o, incluso, que esté en funcionamiento en otro dispositivo externo, ya que permite mostrar registros a través de la red. Esta funcionalidad permite tener versiones de prueba de la aplicación en funcionamiento en el dispositivo objetivo mientras se obtienen registros de actividad en el ordenador de desarrollo.

Depurador

Si se ha seguido la recomendación de utilizar Visual Studio como entorno de desarrollo, se podrá comprobar que en su instalación cuenta con una extensión de Unity que permite utilizar su depurador, el cual es bastante

potente. Funcionará siempre y cuando se ejecute el depurador del proyecto de VS asociado al proyecto de Unity antes de iniciar la ejecución en Unity, como se ha comentado ya previamente en esta sección. El depurador va a mantener todas sus funciones, pudiendo crear puntos de interrupción para parar la ejecución del código y avanzar instrucción a instrucción por ejemplo, pero no podrá entrar a depurar el código propio del motor gráfico.

Profiler

El Profiler [20] es una herramienta propia del editor Unity3D que permite analizar en tiempo real el rendimiento de la aplicación cuando se encuentra en ejecución, tanto en el editor como fuera de el. Esta herramienta permite ver y analizar métricas de rendimiento tales como la ocupación de la CPU, de la GPU o de la memoria, que recursos están más ocupados y por qué componentes, el tiempo que tarda en ejecutarse un determinado script o el tiempo que tarda en pintarse un *frame* en pantalla, como se puede apreciar en la figura D.14. Es una de las herramientas más potentes para el análisis de rendimiento de una aplicación hecha con Unity y es fundamental para conseguir una buena optimización.

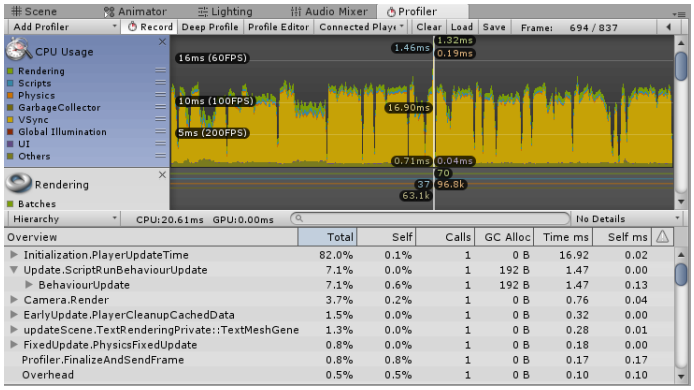

Figura D.14: Ejemplo del profiler de Unity en funcionamiento con alguna de las métricas en pantalla.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se comentarán todos los aspectos relevantes para el usuario final de la aplicación

E.2. Requisitos de usuarios

El usuario no necesita conocimientos previos sobre Realidad Virtual, aunque quizás si sean deseables ciertos conocimientos básicos sobre micología por el tipo de descripciones que pueden aparecer en las setas, ya que no se trata de contenido filtrado o adaptado de ninguna forma para personas que desconocen la materia y la fuente de datos puede ser bastante técnica en ocasiones. En cuanto a requisitos técnicos, el usuario necesitará disponer de un dispositivo móvil que cumpla con los requisitos necesarios para ejecutar la aplicación, es decir, uno de los teléfonos *Daydream Ready*[9], con una versión de Android igual o superior a Nougat 7.0 y con las aplicaciones que se ven en la figura D.9 instaladas y actualizadas a la última versión disponible. También necesitará disponer de un visor de Realidad Virtual *Daydream View*[7]. Existen dos versiones de este visor y dará igual cual de ellas se utilice, aunque se recomienda utilizar la versión del visor de 2017 que tiene mejoras notables de calidad y de disipación de calor.

E.3. Instalación

La instalación de la aplicación en el dispositivo será fácil, pues solo será necesario realizar los pasos normales a la hora de instalar una aplicación de Android que no pertenezca a la tienda digital de Google Play: copiar el fichero apk en el dispositivo, ejecutarlo y esperar a que finalice la instalación de la aplicación.

E.4. Manual del usuario

Controles

En la aplicación se pueden diferenciar diferentes tipos de controles: los controles de cámara, los controles de menús, los controles de movimiento y los controles de manipulación. Todo los controles que requieran de mando utilizan el mismo tipo de mando que viene junto con el visor Daydream View, el cual se puede ver desglosado en la imagen E.1, simplemente se consideran diferentes modos de control y se ha decidido desglosarlos para aclarar su funcionamiento. Como control general, independiente de cualquier modo o menú en el que se encuentre el usuario, se puede destacar el control de volumen, realizado con los botones laterales del mando marcados como el elemento 4 de la figura E.1.

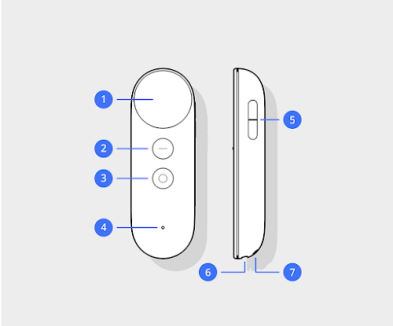

Figura E.1: Esquema del mando del *Daydream View*

Controles de cámara

El control de la cámara se realizará moviendo la cabeza con el visor puesto y el usuario contará con 3 grados de libertad[1], es decir, podrá realizar movimientos de rotación solamente, tal y como se ve en la figura E.2. Se podrá realizar una reorientación de cámara en cualquier momento de la aplicación manteniendo pulsado el botón central *hundido* o que no sobresale del mando, marcado como 3 en la figura E.1.

Controles de menús

El control de menús se realizará mediante el desplazamiento de una retícula en el espacio tridimensional y con la pulsación de un único botón para interactuar con los elementos de interfaz. La retícula se moverá mediante movimientos del mando en el espacio, el cual permite también movimientos con 3 grados de libertad[2]. El botón del mando con el que se interactúa con los elementos

[1] Ver sección "Grados de libertad" del apartado "Conceptos teóricos" de la memoria.

[2] Puede parecer que se goza de más de 3 grados de libertad, pero lo cierto es que estos grados extra están simulados y no se corresponden con movimientos reales del mando

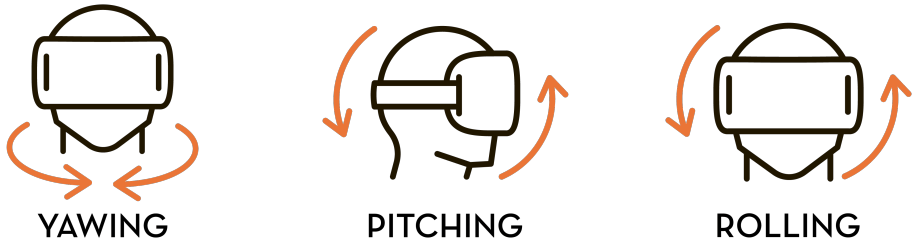

Figura E.2: Representación gráfica de los movimientos de rotación. De izq. a der.: guiñada, cabeceo y alabeo [15].

de interfaz es el panel táctil central del mando, el cual se corresponde con el elemento 1 marcado en la figura E.1, que funciona como botón si se presiona. Se puede volver para atrás un nivel en los menús, e incluso salir de los mismos, presionando el botón central que sobresale del mando, que se corresponde con el elemento 2 marcado en la figura E.1. Si el usuario se encuentra dentro de cualquier nivel, manteniendo pulsado este botón durante 1 segundo se podrá acceder al menú de selección de niveles. En la figura E.3[3] se puede ver un ejemplo de menú de selección de niveles cuando el mando, en la parte inferior de la figura, apunta a un elemento de la interfaz.

Figura E.3: Ejemplo del menú de selección de niveles

Controles de movimiento

El movimiento en el espacio tridimensional se realizará mediante un proceso de traslación o teletransporte. Dicho proceso puede dividirse en varias fases:

- Antes de empezar será necesario comprobar que el mando está libre de cualquier objeto y que este se encuentra a distancia suficiente de cualquier otro elemento del escenario para evitar posibles colisiones.

[3] Las imágenes relativas a las categorías y a los niveles son meramente orientativas, pudiendo ser diferentes en la aplicación final.

- Primero será necesario entrar en el *modo de movimiento*, al cual se accede deslizando un dedo desde la mitad inferior del panel táctil del mando, que se corresponde con el elemento 1 marcado en la figura E.1, hasta la parte superior del mismo. Se habrá accedido con éxito a este modo cuando se visualicen alrededor del mando los elementos de ayuda o se aprecie un láser blanco con forma de arco saliendo desde el extremo superior del mando, tal y como se ve en la figura E.4.

- Una vez en el modo de desplazamiento, se podrá mover el mando en el espacio para apuntar con el arco láser proyectado desde el mando a la posición de destino deseada. Es posible que el arco láser no sea visible en algún punto porque no se permiten desplazamientos sobre otros elementos presentes en el escenario, por lo que será necesario apuntar a un espacio vacío del escenario o que tenga distancia suficiente respecto a otros elementos.

- Finalmente, cuando el punto de destino sea adecuado, habrá que presionar el panel táctil del mando en el extremo superior para completar el desplazamiento.

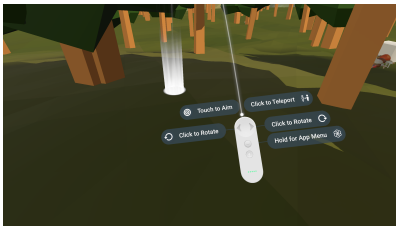

Figura E.4: Elemento visual de ayuda para los controles de movimiento.

Controles de manipulación

En la aplicación se pueden manipular las setas del escenario, pudiendo desplazar estas en el espacio tridimensional cercano al usuario, y se puede interactuar con ellas para obtener información relevante. El proceso de interacción se realizará en varias fases:

- Primero será necesario entrar en el *modo de manipulación*, al cual se accede deslizando un dedo desde la mitad superior del panel táctil del mando, que se corresponde con el elemento 1 marcado en la figura E.1, hasta la parte inferior del mismo. Se habrá accedido con éxito a este modo cuando se visualicen alrededor del mando los elementos de ayuda de la figura E.5 y/o se aprecie un láser blanco recto saliendo desde el extremo superior del mando o la aparición de unas cestas a los pies del usuario.

- Una vez en el modo de manipulación, se podrá mover el mando en el espacio para apuntar con el láser proyectado desde el mando a la seta con la que se quiera interactuar. Es posible que aunque la retícula se encuentre sobre una seta no se pueda interactuar con ella si la distancia es inferior o superior a la permitida para la manipulación, por lo que habrá que posicionarse a una distancia adecuada antes de realizar una interacción.

- Finalmente, estando a una distancia adecuada y teniendo la seta apuntada con el láser, se podrá presionar la parte inferior del panel táctil para realizar la interacción. Se habrá interactuado correctamente con la seta si esta pasa a estar cogida o *atada* al extremo del láser proyectado desde el mando. Llegado este punto, la seta se desplazará por el espacio siguiendo los movimientos del mando.

- Con la seta cogida, ésta se podrá acercar y alejar del usuario cierta distancia realizando sucesivos desplazamientos verticales con el dedo sobre el panel táctil. También será posible rotar la seta sobre su eje vertical manteniendo el dedo sobre uno de los extremos laterales del panel táctil.

- Para mostrar la información relativa a la seta, habrá que presionar el botón central que sobresale del mando, que se corresponde con el elemento 2 marcado en la figura E.1. Se mostrará un panel informativo como el de la figura E.6 en el que se podrá ver el nombre, la descripción, la especie, la comestibilidad y, si hubiese, fotos reales de la seta[4]. Se podrá cerrar el panel informativo volviendo a pulsar el botón presionado inicialmente para abrirlo.

- Para soltar la seta, habrá que pulsar en la parte inferior del panel táctil del mando. La seta pasará a estar libre del mando y caerá al suelo afectada por la gravedad, ignorando cualquier otro tipo de aceleración o impulso.

Figura E.5: Elemento visual de ayuda para los controles de interacción.

[4]No todas las setas de la aplicación cuentan con fotos reales del ejemplar.

Figura E.6: Panel de información de una seta.

Objetivos del juego

La aplicación cuenta con varios niveles repartidos en categorías en función del tipo de escenario planteado, como se puede ver en la figura E.7. Dentro de cada categoría se podrán encontrar varios niveles planteados sobre el mismo escenario, en los cuales habrá variaciones en las reglas de juego, en las especies presentes o incluso en las condiciones ambientales. En función del tipo de nivel elegido, las reglas de juego y los objetivos del nivel pueden variar. En cualquier momento el usuario puede abrir el menú de selección de niveles y consultar tanto su puntuación como los objetivos del nivel actual en la parte derecha del menú, como se puede apreciar en la figura E.8.

Figura E.7: Menú principal de la aplicación.

Figura E.8: Menú de selección de niveles con el panel de información de objetivos en el lateral derecho del menú.

Secuencia de ejecución y de juego

Nada más ejecutar la aplicación, el usuario deberá seguir las instrucciones en pantalla sobre cómo colocar el móvil en el visor y cómo encender el controlador, tal y como se ve en la figura E.9. Tras estos pasos, comunes a todas las aplicaciones de Daydream, se iniciará el juego como tal y el usuario se encontrará en un entorno blanco, recibido con música de fondo y el menú principal enfrente de él, como se puede ver en la figura E.7. Tras desplazarse por el menú y seleccionar el nivel al que desea jugar, el usuario verá un fundido a negro de la pantalla, seguido por una pantalla de carga con una duración variable dependiendo del nivel cargado, tras la cual aparecerá en mitad del escenario del nivel elegido. La pantalla de carga será como la que se puede ver en la figura E.10.

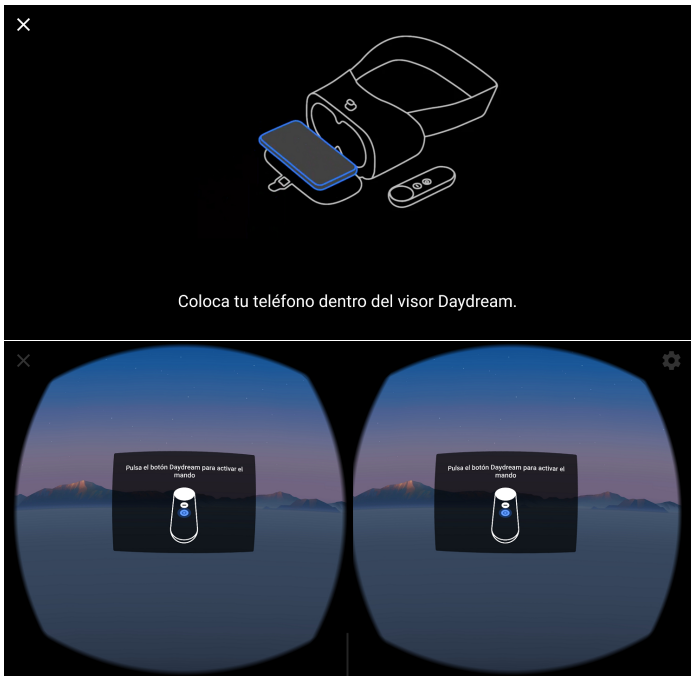

Figura E.9: Colocación del móvil en el visor y activación del mando con el visor puesto.

Llegados a este punto el usuario deberá consultar los objetivos del nivel en el menú de pausa o de selección de niveles y proceder a su finalización, pudiendo cambiar de nivel cuando quiera. Puede encontrarse, no obstante, con niveles que no cuenten con objetivos y sean solo para disfrute del entorno, no teniendo ninguna restricción de tiempo por ejemplo. Se puede ver el ejemplo de un nivel con objetivos en la figura E.8

Con el objetivo claro, el cual será normalmente llegar a una puntuación

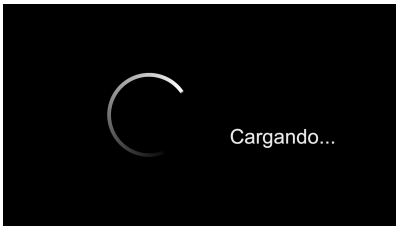

Figura E.10: Pantalla de carga entre niveles.

determinada, el usuario deberá buscar e interaccionar con las setas por el escenario. Deberá alternar entre los modos de desplazamiento y de interacción para ello. Cuando se encuentre con una seta e interaccione con ella, aparecerán dos cestas alrededor del usuario como las de la figura E.11, una para las setas comestibles y otra para las que no lo son. El usuario deberá decidir entonces a qué categoría pertenece la seta cogida y deberá colocarla en una u otra cesta. Mientras tiene una seta cogida, el usuario puede consultar un panel informativo acerca de la seta, como el de la figura E.6, para que le ayude a tomar la decisión, aunque esto tiene un inconveniente y es que una vez consultada la seta ésta no contará para la puntuación del nivel, aun cuando la elección posterior a la consulta sea correcta o incorrecta. Este sistema obliga al usuario a pensar antes de actuar y a memorizar las diferentes especies de setas del escenario, pues estará obligado a buscar otras setas de la especie cada vez que abra el panel de información.

Figura E.11: Cestas diferenciadas donde colocar las setas comestibles y no comestibles.

Cuando el usuario cumpla o falle el objetivo del nivel, aparecerá una pantalla frente a él similar a la de la figura E.12 indicándole que el nivel ha finalizado y el motivo por el que ha finalizado con el texto remarcado con colores llamativos, siendo necesario que después abra el menú de selección de niveles para cambiar a un nuevo nivel.

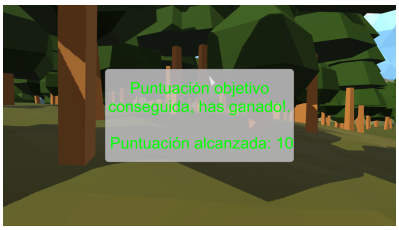

Figura E.12: Texto de final de juego con la puntuación alcanzada.

Bibliografía

[1] M. W. R. Alger. Visual design methods for virtual reality. 2015.

[2] Universidad de Burgos. Retribuciones salariales del pdi laboral. http://www.ubu.es/sites/default/files/portal_page/files/pdi_laboral_2017_2.pdf, 2017.

[3] Google. Android studio - google developers site. https://developer.android.com/studio/, 2018.

[4] Google. Cardboard. https://vr.google.com/cardboard/, 2018.

[5] Google. Daydream app - play store. https://play.google.com/store/apps/details?id=com.google.android.vr.home, 2018.

[6] Google. Daydream controller emulator app - google developers site. https://developers.google.com/vr/daydream/controller-emulator, 2018.

[7] Google. Daydream view. https://vr.google.com/intl/es_es/daydream/smartphonevr/, 2018.

[8] Google. Google vr services app - play store. https://play.google.com/store/apps/details?id=com.google.vr.vrcore&hl=es, 2018.

[9] Google. Phones compatible with daydream. https://vr.google.com/daydream/smartphonevr/phones/, 2018.

[10] Google. Resonance audio - google developers site. https://developers.google.com/resonance-audio/, 2018.

[11] Tristan Parrish Moore. What steam's data reveals about the health of vr's ecosystem. https://www.vrfocus.com/2016/07/what-steams-data-reveals-about-the-health-of-vrs-ecosystem/, 2017.

[12] Oracle. Java se downloads site. http://www.oracle.com/technetwork/java/javase/downloads/index.html, 2018.

[13] Rodrigo Jurado Pajares. Github - ubusetasvr. https://github.com/Kiszaner/2017_SetasVR_TFG, 2017.

[14] Tom Preston-Werner. Semantic versioning 2.0.0. https://semver.org/#semantic-versioning-200, 2018.

[15] Andrew Lucas Studios. A quick guide to degrees of freedom in virtual reality. https://lucas-studios.com/quick-guide-degrees-of-freedom-virtual-reality-vr/, 2018.

[16] Unity Technologies. Textmesh pro. https://www.assetstore.unity3d.com/en/#!/content/84126, 2017.

[17] Unity Technologies. The inspector window. https://docs.unity3d.com/Manual/UsingTheInspector.html, 2018.

[18] Unity Technologies. Patch releases. https://unity3d.com/unity/qa/patch-releases, 2018.

[19] Unity Technologies. Prefabs overview. https://docs.unity3d.com/Manual/Prefabs.html, 2018.

[20] Unity Technologies. Profiler overview. https://docs.unity3d.com/Manual/Profiler.html, 2018.

[21] Unity Technologies. Unity3d built-in console - unity documentation. https://docs.unity3d.com/Manual/Console.html, 2018.

[22] Quantum Theory. Polyworld: Woodland low poly toolkit. https://assetstore.unity.com/packages/3d/environments/landscapes/polyworld-woodland-low-poly-toolkit-18572, 2018.

[23] Wikipedia. Blender. https://en.wikipedia.org/w/index.php?title=Blender_(software)&oldid=842678183, 2018.

[24] Wikipedia. Solid. https://en.wikipedia.org/w/index.php?title=SOLID&oldid=841159967, 2018.

[25] ZenHub. Zenhub. https://www.zenhub.com/, 2017.