ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

The house of crimes: Videojuego e-Learning de alemán con temática de suspense

Francisco Madueño Chulián

16 de octubre de 2015

ESCUELA SUPERIOR DE INGENIERÍA

GRADO EN INGENIERÍA INFORMÁTICA

The house of crimes: Videojuego e-Learning de alemán con temática de suspense

- Autor del proyecto: FRANCISCO MADUEÑO CHULIÁN
- Director del proyecto: PABLO DE LA TORRE MORENO
- Departamento: Ingeniería Informática
- Codirectora del proyecto: ANKE BERNS
- Departamento: Filología Francesa e Inglesa

Cádiz, 16 de octubre de 2015

Fdo: Francisco Madueño Chulián

Agradecimientos

A mi director y co-directora por sus consejos.
Mención especial al Profesor Manuel Palomo-Duarte por su implicación en el desarrollo del proyecto.
A mis compañeros de universidad por sus ánimos durante todo este año.
A Mercedes Páez Piña, Andrea Calderón Márquez y Alicia Garrido Guerrero por su gran contribución para mejorar la experiencia de juego.
A mis padres.

Licencia

Resumen

En las últimas décadas los videojuegos con carácter educativo, utilizados para el aprendizaje de lenguas extranjeras, han ido ganando popularidad. Tanto es así que, la industria del videojuego está cambiando para adaptarse a estos nuevos tiempos, en los que los videojuegos no sólo sirven para entretener, sino también para educar y aprender.

Sin embargo, dependiendo del ambiente en el que se encuentre el jugador, este no aprende ni de la misma manera ni al mismo ritmo. Este proyecto trata de comprobar si un ambiente de suspense puede incrementar la atención del jugador y con ello su grado de aprendizaje. La particularidad del juego, que hemos diseñado específicamente para este propósito, consiste en que cada partida se genera dinámicamente. Esto significa que los retos a los que se enfrenta el jugador no serán los mismos, y que un jugador no jugará dos partidas iguales. De este modo logramos por un lado que el jugador pueda jugar varias partidas y por otro que el jugador, a priori, nunca llegue a conocer las soluciones a los retos, a corto tiempo. Esto último resultaría perjudicial para el experimento que pretende realizarse.

The house of crimes contiene la mayoría de los elementos de una aventura gráfica tradicional. Permite al jugador desplazarse libremente por un mapa, interactuar con los personajes y con los escenarios, también dispone de un inventario para la gestión de objetos. Pero también ofrece otros elementos menos usuales, como la opción multidioma o la generación de partidas aleatorias.

El presente proyecto cuenta con cuatro versiones, las cuales se diferencian entre el suspense o no suspense de los objetos y el ambiente de las habitaciones. Así hay, objetos y habitaciones sin suspense, objetos con suspense y habitaciones sin suspense, objetos sin suspense y habitaciones con suspense y objetos y habitaciones con suspense. Detrás de la idea de crear diferentes versiones está el objetivo de averiguar si diferentes entornos pueden provocar distintas emociones en el jugador e influir en el grado de su aprendizaje.

Todas las acciones, que realice el jugador, se almacenarán en un historial, el cual se almacenará, a su vez, en un servidor para su posterior análisis. Este proceso servirá para tener un control sobre las acciones que realiza el jugador durante una partida. Además el fichero, que se genera automáticamente durante este proceso, permitirá obtener valiosos datos para comprobar qué tipo de entorno incide de qué manera en el aprendizaje del alumno.
[Delatorre et al., 2015]

Índice general

Índice de figuras

Índice de cuadros

Parte I

Prolegómeno

Capítulo 1

Introducción

1.1. Motivación

Los idiomas se han convertido en una parte fundamental de nuestra formación académica. Tanto es así que ser políglota puede marcar la diferencia entre conseguir un puesto de trabajo o no. Es por ello que aprender nuevas lenguas es fundamental en nuestra formación.

La forma de aprender un nuevo idioma es mediante la exposición y práctica constante a dicho idioma, que debe ser complementada además por la repetición continuada de ejercicios. Estos ejercicios se suelen realizar fuera del horario lectivo. Por ello un entorno e-learning es ideal para el aprendizaje de idiomas, ya que provee al alumno de retos online, sin tener que salir de casa. Además se trata de entornos y herramientas accesibles en cualquier momento, lo cual permite que el alumno organice su aprendizaje según su tiempo y necesidades.

La idea inicial de diseñar un videojuego e-learning para aprender inglés enfocado a alumnos de primaria fue cambiando, a medida que se fueron manteniendo las primeras reuniones. Aún así se mantuvo la característica de e-learning. A partir de entonces el proyecto adquirió otras nuevas características. Ya no estaba enfocado a niños, sino a jugadores universitarios con un nivel A1 en algún idioma extranjero (inglés, alemán, francés, etc...). Además se decidió usar el proyecto como un experimento para saber si estímulos externos, en este caso el suspense, influyen en el aprendizaje y, si es así, en qué medida. A raíz de esto también se descartó la idea de que el juego estuviera formado por minijuegos. De hecho se decidió que la mejor forma de realizar el experimento pasaba por desarrollar una historia, de modo que el juego pasó a ser una aventura gráfica con narrativa. Por último, se decidió usar el alemán, en vez del inglés, como idioma a aprender porque se nos brindó la posibilidad de probar el juego con alumnos de la Universidad de Cádiz y que estuvieran matriculados en la asignatura de Alemán II (nivel A1).

1.2. Alcance

El objetivo de este proyecto es proporcionar una herramienta para la medición y comparación del grado de aprendizaje conseguido en una competencia muy concreta: aprender idioma extranjero, como el alemán, tanto en un contexto neutral como de suspense.

1.3. Organización del documento

El documento es la memoria del Trabajo Fin de Grado de Ingeniería Informática (GII) de la Universidad de Cádiz realizada por Francisco Madueño Chulián.

Este documento está dividido en los siguientes capítulos:

- Introducción: Se resume la idea y la motivación que nos ha llevado a desarrollar este proyecto.
- Conceptos básicos: Aporta una pequeña descripción del juego, además de justificar las decisiones tomadas en relación al suspense.
- Planificación: Se explica cómo ha sido la organización temporal del proyecto. Se aporta un diagrama de Gantt.
- Análisis del sistema: Se detallan los objetivos, requisitos funcionales y no funcionales, se describen los casos de uso y se muestra el diagrama de clases.
- Diseño del sistema: Se detallan las herramientas utilizadas para el desarrollo, así como la descripción de la interfaz gráfica del juego.
- Implementación: Se resumen los aspectos más importantes de la implementación del proyecto.
- Pruebas del sistema: Se explican qué pruebas y cómo se han llevado a cabo.
- Manual de ampliación: Se adjunta un manual que permite añadir nuevos elementos al juego.
- Manual de usuario: Se presenta un manual de instalación y una guía del juego.
- Conclusiones: Se aporta una reflexión sobre el proyecto, así como mejoras futuras.

Capítulo 2

Conceptos básicos

2.1. Descripción del juego

Tal y como se indica en [Álvarez, 2009] la inclusión del e-learning en el modelo pedagógico hace que aumente la responsabilidad y flexibilidad del alumno en su proceso de aprendizaje y, en consecuencia una mejora del proceso educativo.

Actualmente, el valor del e-learning cotiza en alza en los mercados, por lo que es un sector muy estable. Se calcula que en el año 2011, la industria e-learning movilizó cerca de 35.600 millones de dólares. Las estadísticas más recientes, referentes a 2013, elevan esta cifra a los 56.200 millones de dólares esto supone un crecimiento de un 55,2 % en apenas tres años. Pocos son los valores que han crecido tanto en la situación de recesión económica que se vive hoy en día. Las perspectivas para 2015 auguran que la actividad de la enseñanza on-line duplique su volumen hasta superar los más de 100.000 millones de dólares. Europa Occidental se alza, actualmente, como el segundo consumidor de herramientas e-learning con un volumen de inversión cercano a los 6.800 millones de dólares.[Santamas, 2014]

Llegados a este punto es obligatorio hablar sobre los serious games o juegos serios. Este nombre se le asigna a los videojuegos cuyo principal objetivo es la formación de su usuario, dejando el entretenimiento en un segundo plano. Aunque esta idea es relativamente nueva, cada año va ganando más popularidad. Tal es así que grandes compañías de videojuegos como Electronics Arts o EA llevan ya varios años colaborando con universidades para fomentar la creación y el uso de videojuegos educativos. [Marcano, 2008]

Sin embargo, la integración en las aulas no es algo trivial. Para captar la atención del alumno el sistema debe permitir una experiencia interactiva plena, mientras que, al mismo tiempo debe facilitar del desarrollo de un entorno pedagógico. [Peinado et al., 2005]

2.2. Affective Norms for English Words (ANEW)

Para desarrollar satisfactoriamente el experimento, el videojuego debe ser capaz de generar ambientes que expresen suspense. Con ambiente nos referimos solamente a objetos y habitaciones. Aquí se nos plantea la siguiente pregunta, ¿cómo podemos lograr que el jugador sienta lo que nos proponemos? Y lo más importante: ¿qué justifica que los ambientes elegidos impliquen suspense para toda persona que lo juegue?

Para dar respuesta a ambas preguntas contamos con un estudio sobre las Affective Norms For English Words, o como se conocen comúnmente ANEW. Este estudio ha sido llevado a cabo varias veces, por diferentes universidades y en varios idiomas. El estudio muestra el efecto en la afectividad que tiene cada palabra, de una lista de unas mil, puntuada con valencia, cuán negativa o positiva resulta la palabra, excitación y sensación de control. Esta última variable es menos empleada en psicología cuando se usa el modelo. [Bradley and Lang, 1999]

Entendiendo lo anterior, se han formado dos grupos de objetos: uno con valencias medias-altas y excitación media-baja. Se trata del grupo de control, es decir, objetos que generan una afectividad neutra y otro con valencias medio-bajas y excitación media-alta, que resulta más propenso a generar suspense en el contexto adecuado. A continuación ofrecemos una lista con las palabras, que se han usado para nuestro estudio, incluyendo sus respectivas puntuaciones:

Palabra	Valencia	Excitación
Ataúd	1.63	6.06
Basura	2.06	4.31
Bruja	2.63	4.83
Calavera	2.09	3.58
Daga	2.81	4.53
Jaula	2.83	4.09
Máscara	2.50	4.07
Navaja	3.37	6.08
Pistola	2.28	5.41
Rifle	2.47	5.20
Serpiente	2.40	4.85
Tabaco	2.57	5.24
Veneno	2.06	4.44

Cuadro 2.1: Palabras adecuadas para generar suspense.

Palabra	Valencia	Excitación
Anillo	4.91	3.38
Azúcar	4.66	3.83
Bolígrafo	4.31	3.31
Bombilla	4.36	3.33
Café	4.50	3.79
Caramelo	5.27	3.65
Diamante	5.56	3.65
Espejo	4.52	3.28
Libro	4.89	3.11
Llave	4.44	3.44
Moneda	4.88	3.45
Reloj	4.50	3.79
Zapato	4.48	3.53

Cuadro 2.2: Palabras neutras.

2.3. Colores y emociones en los videojuegos

Si anteriormente hemos visto cómo conseguir un ambiente de suspense mediante la elección de determinados objetos, ahora toca abordar el problema de cómo conseguir este mismo efecto con el diseño de las habitaciones. Para que el experimento sea válido todas las versiones del juego deben tener los mismos escenarios garantizando de esta manera que haya las menos variables posibles. Por ello hay que conseguir cambiar el ambiente de la habitación sin cambiar la imagen, que representa a la misma.

La solución a este problema está relacionada con las emociones que transmiten los colores. Existen estudios [Joosten et al., 2010] que afirman que el color rojo está asociado con la ira, el azul claro con la sorpresa y el verde con la tristeza. El primero tiene una excitación elevada, mientras que los dos últimos tienen una valencia muy baja. Por ello, aplicando un filtro de color a la habitación, combinando el rojo con alguno de los otros dos colores, conseguimos un ambiente de suspense similar al anteriormente conseguido con los objetos.

2.4. Otros elementos

Anteriormente hemos hablado de cómo conseguir que el jugador experimente el suspense usando simplemente los objetos y los colores de las habitaciones. A continuación surge otra pregunta ¿podemos conseguir suspense con otros elementos que formen parte del videojuego? ¿Se puede variar la música o el aspecto de los personajes?

Actualmente no hay ningún estudio que haya demostrado por qué una serie de notas musicales genera o no suspense. Puesto que no podemos basarnos ningún estudio previo para elegir una u otra melodía, consideramos que la música debía ser la misma en todas las versiones.

Con respecto a los personajes, hay que introducir un nuevo concepto, la empatía. Hay muchos aspectos que influyen en el nivel de empatía hacia una persona, como las expresiones faciales, la

voz o la postura. Se han identificado cuatro componentes de la empatía; el afecto compartido, el control de las emociones, la capacidad de aprender sobre las situaciones que afectan a otras personas y la conciencia en uno mismo. [Gerdes et al., 2010] [Paiva et al., 2005]

En este experimento se está probando si la valencia y la excitación de los conceptos del juego, objetos y habitaciones, influye en el aprendizaje. Por ese motivo introducir la empatía como factor adicional no nos permitiría realizar esta observación, ya que los datos finales estarían mezclados. De allí nuestra decisión de mantener los mismos personajes en todas las versiones.

Capítulo 3

Planificación

3.1. Etapas

A lo largo de su desarrollo el proyecto ha pasado por diferentes etapas. En cada una de ellas se han realizado diferentes tareas y, en consecuencia, han aparecido problemas que ha habido que resolver. A continuación se explica con detenimiento cada etapa.
Para este proyecto se decidió a usar un modelo tradicional en cascada. Nos decidimos por este modelo porque, desde un principio, sabíamos qué tipo de juego queríamos desarrollar. Por lo tanto, la elicitación de los requisitos se completó en una etapa muy temprana del desarrollo. Otra razón importante para elegir este modelo de desarrollo fueron las pruebas. Es muy difícil dividir el juego en grandes módulos porque el mundo que presenta está interconectado. Por esta razón, las pruebas, excepto las unitarias y las de integración, tenían más valor si se realizaban una vez terminado el desarrollo.

3.1.1. Etapa inicial

El proyecto se inició con una serie de reuniones entre los integrantes. En ellas se decidieron las mecánicas básicas del juego y el lenguaje de programación en el que se desarrollaría. En reuniones posteriores se empezó a idear cómo introducir el suspense en el videojuego y a decidir como se realizarían los experimentos una vez que el proyecto estuviera terminado.

3.1.2. Etapa de aprendizaje

Una vez decidido el tipo de juego que se quería desarrollar, los objetivos del mismo y la tecnología que iba a emplearse, comenzó la fase de aprendizaje.

En esta etapa se investigó el funcionamiento de la librería LibGDX, usando como material foros oficiales de la propia herramienta, libros disponibles en la biblioteca de la Escuela Superior de Ingeniería, [Márquez and Sánchez, 2014] y vídeos tutoriales donde se enseñaba su funcionamiento.

Esta etapa se ha ido repitiendo a lo largo del desarrollo del juego, ya que iban surgiendo nuevas ideas y necesidades, que requerían el uso de distintas tecnologías. Por ejemplo, cuando se decidió hacer que los retos fueran dinámicos y que XML era la mejor solución para desarrollar esta funcionalidad, tuvimos que aprender como funcionaba este lenguaje. [Hunter, 2005] Lo mismo

ocurrió cuando se puso cómo requisito, que la aplicación permitiera subir archivos a Dropbox. En ese momento fue necesario aprender a manejar la API correspondiente.

La etapa de aprendizaje también incluyó la lectura de los estudios realizados con las ANEW. Esta parte sin duda ha sido la más difícil de la etapa de aprendizaje. La idea de estos estudios es proporcionarnos palabras que puedan ser usadas como objetos dentro del juego. El principal problema que se nos planteó a la hora de diseñar el juego y de elegir el vocabulario ha sido que, en todos los estudios realizados, la mayoría de las palabras eran nombres abstractos, adjetivos o verbos. Por tanto se disponían de muy pocos objetos tangibles. Y, de entre las pocas palabras que podrían servir, hemos tenido que desechar muchas ya que no eran comunes en el vocabulario de un alumno del nivel A1 de alemán o sus puntuaciones de valencia o excitación no eran adecuadas para generar los ambientes que queríamos conseguir. Todo este trabajo ha afectado a la planificación, provocando retrasos de semanas.

3.1.3. Etapa de desarrollo

La etapa de desarrollo comprende todo el desarrollo del juego, se trata de la etapa más larga.

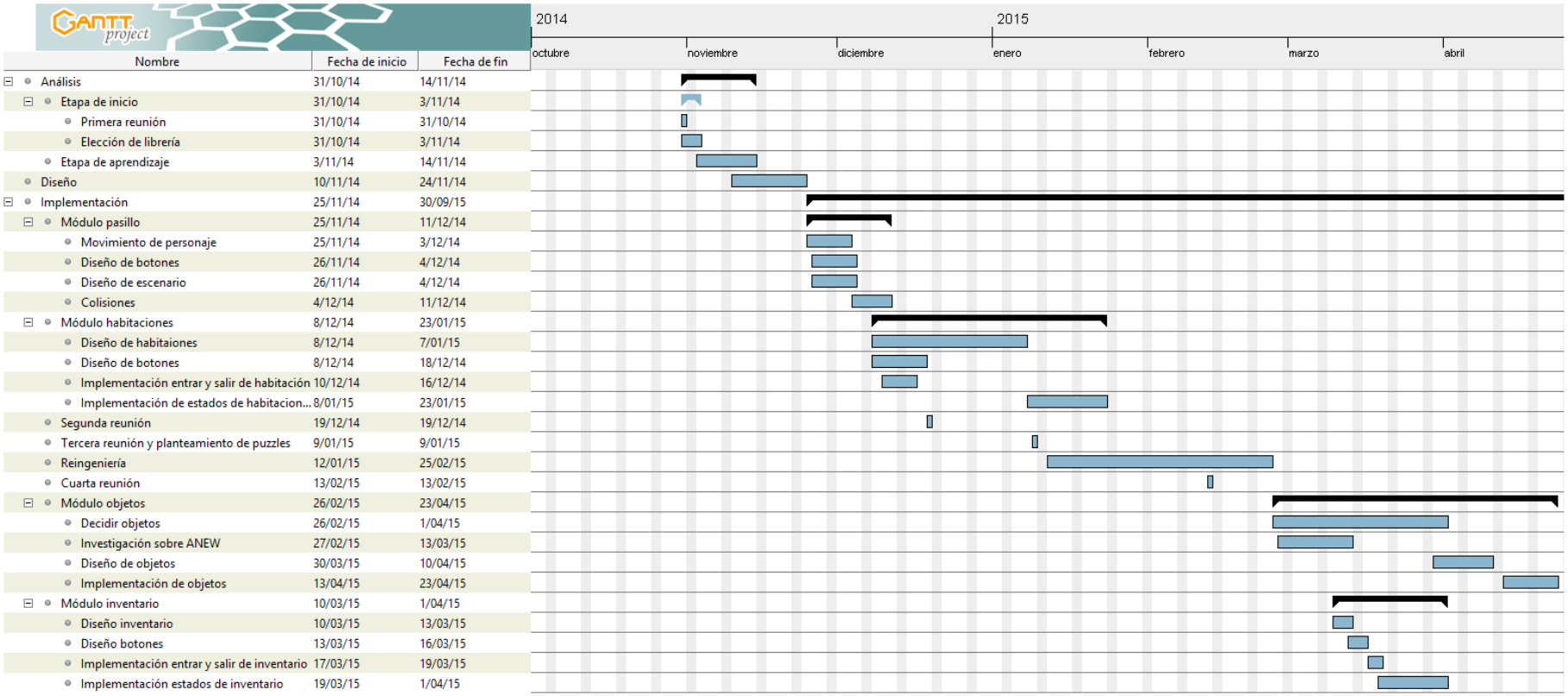
El desarrollo comenzó en diciembre del año 2014 y duró nueve meses. Este tiempo tan amplio se debe a que en ese momento aún me quedaron por aprobar dos asignaturas del grado de Ingeniería Informática, de modo que durante los meses de diciembre, enero y febrero tenía que compaginar mis estudios con el trabajo en el proyecto. Además a finales del mes de febrero empecé unas prácticas de empresa que consumían todo mi tiempo durante la mañana. A esto hay que añadir que me estaba preparando para el First Certificate in English (University of Cambridge), lo cual me obligaba a dividir las tardes para poder dedicarle tiempo a ambas obligaciones.

El desarrollo fue lento hasta que no terminé las prácticas y superé mi examen de First, en el mes de julio. A partir de entonces le dediqué todo mi tiempo al proyecto, lo cual hizo que el desarrollo fuera mucho más rápido, llegando a terminar el proyecto a finales de septiembre.

3.1.4. Etapa de pruebas

Las pruebas unitarias y de integración se realizaron cada vez que se terminara la implementación de un módulo. Con esto nos aseguramos que si surgiera algún fallo durante la implementación del siguiente módulo el problema no estaría causado por algo previamente desarrollado. Las pruebas de sistema, aceptación y funcionales se realizaron unas semanas antes de la finalización del desarrollo del videojuego. El resultado de estas pruebas hicieron que la etapa de desarrollo tuviera que ampliarse, con el fin de resolver problemas referentes al idioma y a la integración del suspense. Estas pruebas, además, revelaron la necesidad de añadir nuevas funcionalidades para mejorar la experiencia de juego.
Antes de concluir la etapa de pruebas se eligieron varios alumnos colaboradores de alemán para que probasen el juego y nos ayudarán detectar y corregir eventuales errores lingüísticos o relacionados con la interfaz de usuario.

3.2. Diagrama de Gantt



Nombre	Fecha de inicio	Fecha de fin
Análisis	31/10/14	14/11/14
Etapa de inicio	31/10/14	3/11/14
Primera reunión	31/10/14	31/10/14
Elección de librería	31/10/14	3/11/14
Etapa de aprendizaje	3/11/14	14/11/14
Diseño	10/11/14	24/11/14
Implementación	25/11/14	30/09/15
Módulo pasillo	25/11/14	11/12/14
Movimiento de personaje	25/11/14	3/12/14
Diseño de botones	26/11/14	4/12/14
Diseño de escenario	26/11/14	4/12/14
Colisiones	4/12/14	11/12/14
Módulo habitaciones	8/12/14	23/01/15
Diseño de habitaiones	8/12/14	7/01/15
Diseño de botones	8/12/14	18/12/14
Implementación entrar y salir de habitación	10/12/14	16/12/14
Implementación de estados de habitacion...	8/01/15	23/01/15
Segunda reunión	19/12/14	19/12/14
Tercera reunión y planteamiento de puzzles	9/01/15	9/01/15
Reingeniería	12/01/15	25/02/15
Cuarta reunión	13/02/15	13/02/15
Módulo objetos	26/02/15	23/04/15
Decidir objetos	26/02/15	1/04/15
Investigación sobre ANEW	27/02/15	13/03/15
Diseño de objetos	30/03/15	10/04/15
Implementación de objetos	13/04/15	23/04/15
Módulo inventario	10/03/15	1/04/15
Diseño inventario	10/03/15	13/03/15
Diseño botones	13/03/15	16/03/15
Implementación entrar y salir de inventario	17/03/15	19/03/15
Implementación estados de inventario	19/03/15	1/04/15

Figura 3.1: Diagrama de Gantt. Parte 1.

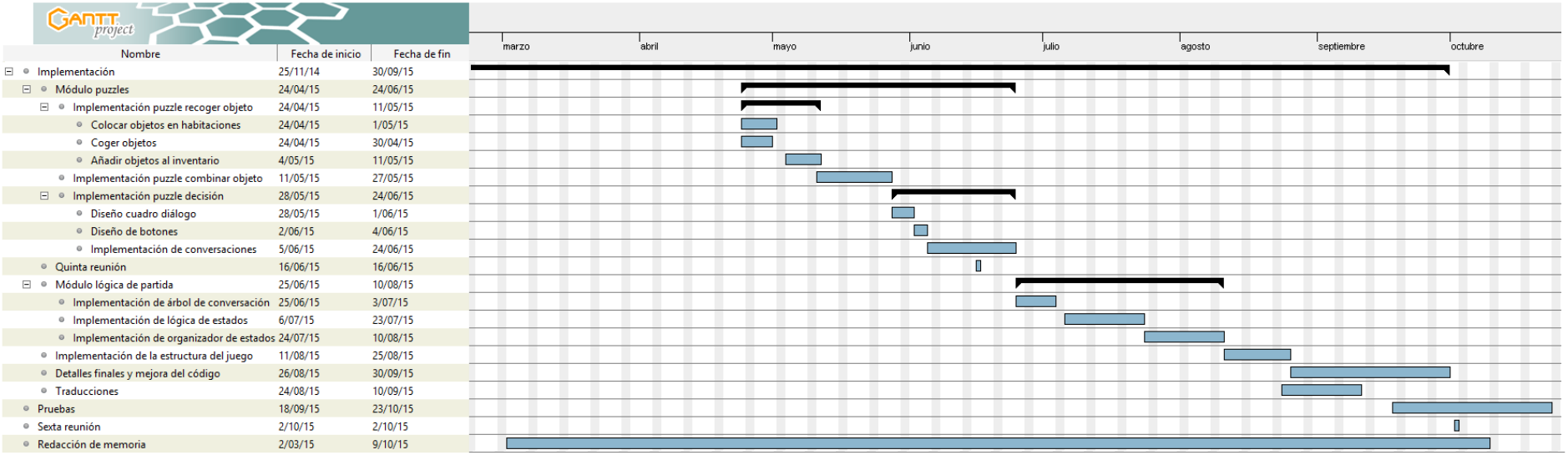


Figura 3.2: Diagrama de Gantt. Parte 2.

Parte II

Desarrollo

Capítulo 4

Análisis del Sistema

4.1. Objetivos del Sistema

A continuación se detallan los objetivos del sistema:

OBJ-01	Permitir jugar partidas en un mundo de ficción
Descripción	El sistema debe de ser capaz de permitir que el jugador pueda completar partidas satisfactoriamente.

Cuadro 4.1: Objetivo jugar partidas.

4.2. Requisitos funcionales

A continuación se detallan los requisitos funcionales del sistema:

FRQ-01	Iniciar partida
Descripción	El sistema debe ser capaz de permitir que el jugador inicie una partida nueva.

Cuadro 4.2: Requisito iniciar partida

FRQ-02	Salir de la partida
Descripción	El sistema debe ser capaz de permitir que el jugador salga de la partida.

Cuadro 4.3: Requisito salir de la partida

FRQ-03	Moverse a la derecha
Descripción	El sistema debe ser capaz de permitir que el jugador mueva al protagonista a la derecha.

Cuadro 4.4: Requisito moverse a la derecha

FRQ-04	Moverse a la izquierda
Descripción	El sistema debe ser capaz de permitir que el jugador mueva al protagonista a la izquierda.

Cuadro 4.5: Requisito moverse a la izquierda

FRQ-05	Moverse abajo
Descripción	El sistema debe ser capaz de permitir que el jugador mueva al protagonista hacia abajo.

Cuadro 4.6: Requisito moverse abajo

FRQ-06	Moverse arriba
Descripción	El sistema debe ser capaz de permitir que el jugador mueva al protagonista hacia arriba.

Cuadro 4.7: Requisito moverse arriba

FRQ-07	Abrir inventario
Descripción	El sistema debe ser capaz de permitir que el jugador abra el inventario.

Cuadro 4.8: Requisito abrir inventario

FRQ-08	Entrar en habitación
Descripción	El sistema debe ser capaz de permitir que el jugador entre en las habitaciones.

Cuadro 4.9: Requisito entrar en habitación

FRQ-09	Salir de habitación
Descripción	El sistema debe ser capaz de permitir que el jugador salga de las habitaciones.

Cuadro 4.10: Requisito salir en habitación

FRQ-10	Investigar habitación
Descripción	El sistema debe ser capaz de permitir que el jugador investigue una habitación.

Cuadro 4.11: Requisito investigar habitación

FRQ-11	Terminar investigación
Descripción	El sistema debe ser capaz de permitir que el jugador termine de investigar una habitación.

Cuadro 4.12: Requisito terminar investigación

FRQ-12	Conversar
Descripción	El sistema debe ser capaz de permitir que el jugador converse con los personajes.

Cuadro 4.13: Requisito conversar

FRQ-13	Terminar conversación
Descripción	El sistema debe ser capaz de permitir que el jugador termine una conversación con un personaje.

Cuadro 4.14: Requisito terminar conversación

FRQ-14	Decidir
Descripción	El sistema debe ser capaz de permitir que el jugador escoja entre varias posibles respuestas que le proporciona un personaje.

Cuadro 4.15: Requisito decidir

FRQ-15	Coger objeto
Descripción	El sistema debe ser capaz de permitir que el jugador coja objetos que haya repartidos por las habitaciones.

Cuadro 4.16: Requisito coger objeto

FRQ-16	Cerrar inventario
Descripción	El sistema debe ser capaz de permitir que el jugador salga del inventario.

Cuadro 4.17: Requisito cerrar inventario

FRQ-17	Combinar objetos
Descripción	El sistema debe ser capaz de permitir que el jugador combine objetos que haya recogido.

Cuadro 4.18: Requisito combinar objetos

FRQ-18	Cancelar combinación
Descripción	El sistema debe ser capaz de permitir que el jugador cancele la combinación de objetos.

Cuadro 4.19: Requisito cancelar combinación

FRQ-19	Mostrar descripción
Descripción	El sistema debe ser capaz de mostrar la descripción de un objeto que el jugador haya seleccionado.

Cuadro 4.20: Requisito mostrar descripción

FRQ-20	Mostrar objetivo
Descripción	El sistema debe ser capaz de mostrar el próximo objetivo del jugador.
Comentario	Se entiende como objetivo el próximo paso que tiene que dar el jugador en el juego para llegar a completarlo. Esta funcionalidad proporciona al jugador una pista sobre el puzle que está realizando.

Cuadro 4.21: Requisito mostrar objetivo

FRQ-21	Actualizar objetivo
Descripción	El sistema debe ser capaz de actualizar el objetivo del jugador.

Cuadro 4.22: Requisito actualizar objetivo

FRQ-22	Elegir asesino
Descripción	El sistema debe ser capaz de permitir que el jugador identifique al asesino al final del juego.

Cuadro 4.23: Requisito elegir asesino

FRQ-23	Elegir arma
Descripción	El sistema debe ser capaz de permitir que el jugador identifique al arma homicida al final del juego.

Cuadro 4.24: Requisito elegir arma

FRQ-24	Sumar puntos
Descripción	El sistema debe ser capaz de sumar puntos al jugador.

Cuadro 4.25: Requisito sumar puntos

FRQ-25	Actualizar fichero de Log
Descripción	El sistema debe ser capaz de registrar las acciones realizadas por el jugador en un fichero de log.

Cuadro 4.26: Requisito actualizar fichero de log

FRQ-26	Subir fichero de Log
Descripción	El sistema debe ser capaz de, una vez terminada la partida, subir el fichero de log a un servidor de almacenamiento para su posterior examen por parte de las personas implicadas en el experimento.

Cuadro 4.27: Requisito subir fichero de log

FRQ-27	Ver créditos
Descripción	El sistema debe ser capaz de permitir que el jugador pueda visualizar los títulos de crédito del juego.

Cuadro 4.28: Requisito ver créditos

FRQ-28	Abrir diccionario
Descripción	El sistema debe ser capaz de permitir que el jugador pueda visualizar el diccionario de términos.

Cuadro 4.29: Requisito abrir diccionario

FRQ-29	Cerrar diccionario
Descripción	El sistema debe ser capaz de permitir que el jugador pueda salir del diccionario y volver al pasillo.

Cuadro 4.30: Requisito cerrar diccionario

4.2.1. Diagramas de casos de uso

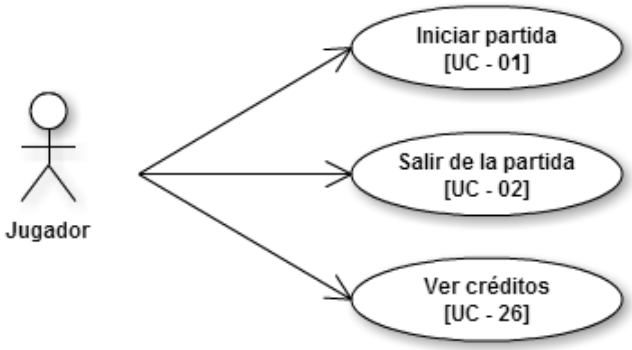


Figura 4.1: Casos de uso de la pantalla de inicio.

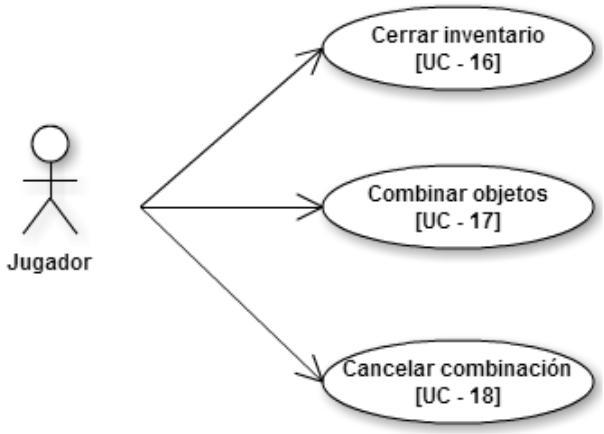


Figura 4.2: Casos de uso de la pantalla inventario.

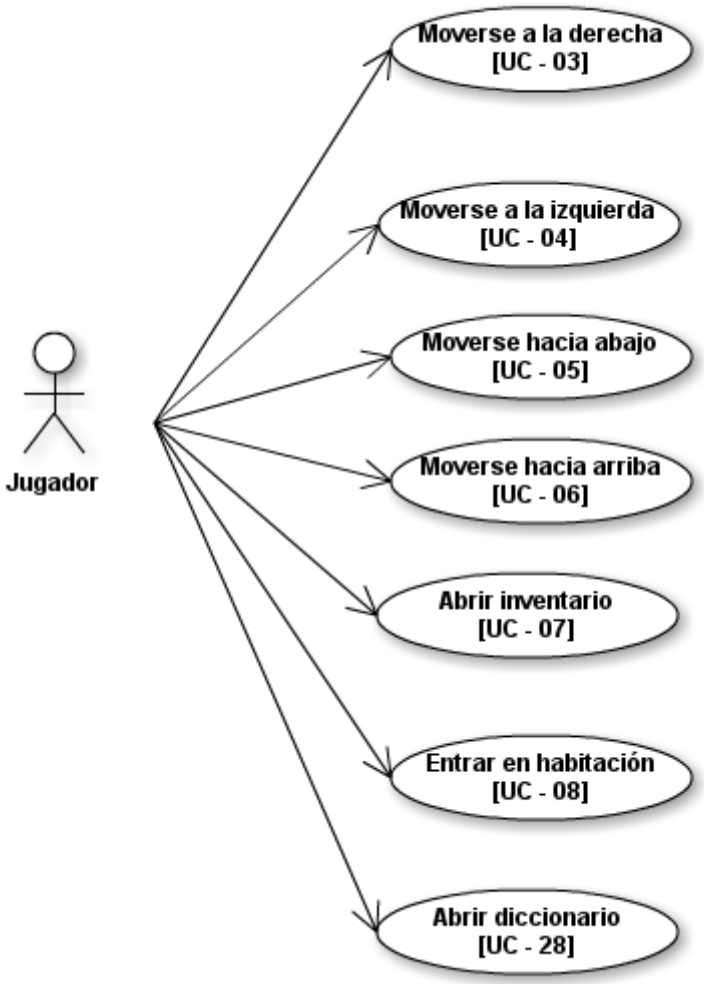


Figura 4.3: Casos de uso de la pantalla pasillo.

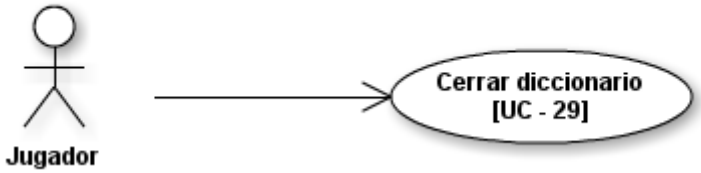


Figura 4.4: Casos de uso de la pantalla diccionario.

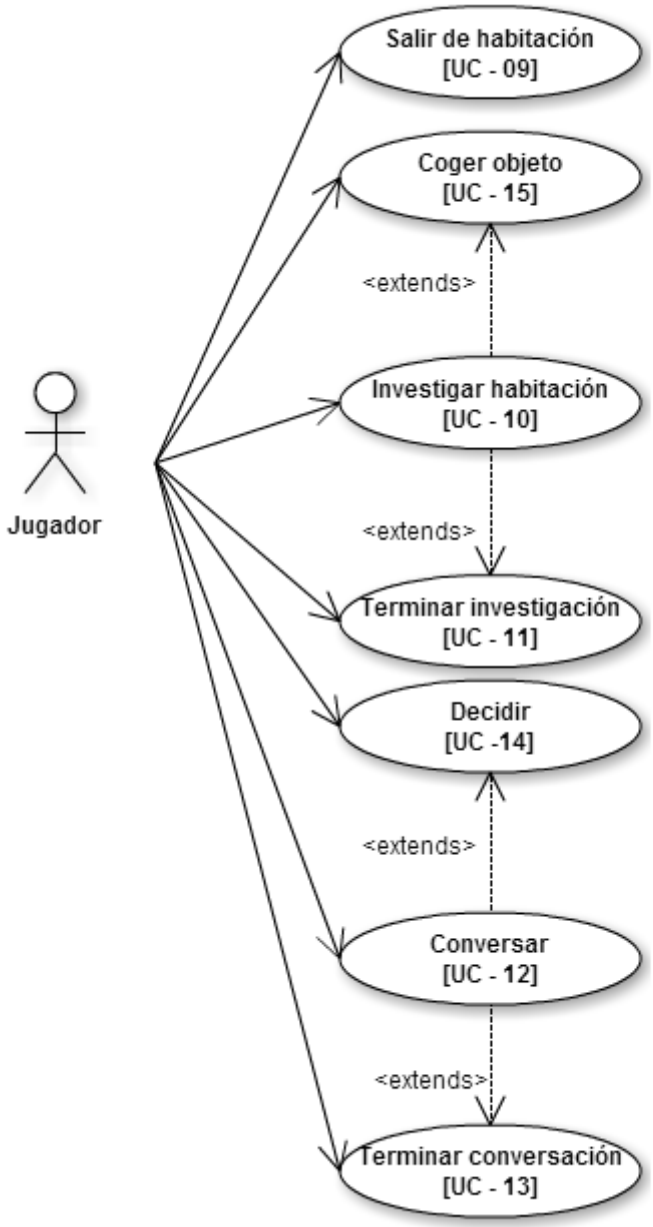


Figura 4.5: Casos de uso de la pantalla habitación.

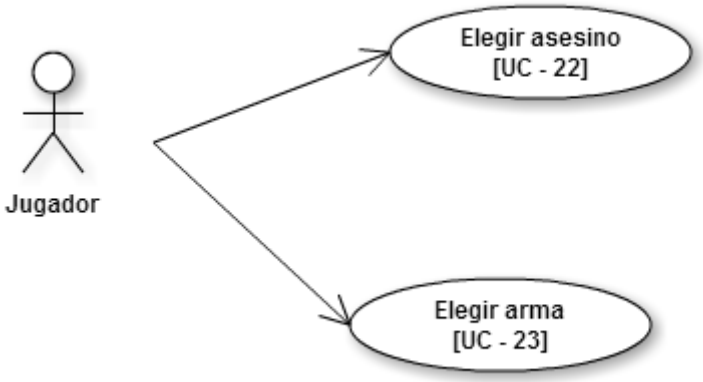


Figura 4.6: Casos de uso de las pantallas Selección Asesino y Selección Arma.

4.2.2. Descripción de escenarios

UC-01	Iniciar partida	
Descripción	El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-01.	
Precondición	No hay precondiciones.	
Secuencia	Pasos	Acción
	1	El jugador inicia el juego.
	2	El sistema muestra la pantalla de título.
	3	El jugador pulsa el botón *'Inicio'*.
	4	El sistema muestra la pantalla de presentación.
	5	El jugador introduce su nombre y apellidos y pulsa el botón aceptar.
	6	El jugador pulsa el botón Continuar.
	7	El sistema muestra la pantalla Pasillo y el jugador puede empezar la partida.
Postcondicion	El jugador inicia la partida.	
Excepciones	Pasos	Acción
	3	El jugador pulsa el botón *'Salir'*, el juego termina y el caso de uso queda sin efecto.
	3	El jugador pulsa el botón *'Créditos'*, el juego muestra la pantalla de créditos.
	*	El jugador cierra el juego y el caso de uso queda sin efecto.

Cuadro 4.31: Descripción del caso de uso Iniciar Partida.

<table>
<tr><td>UC-02</td><td colspan="2">Salir de la partida</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-02.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado el juego.</td></tr>
<tr><td rowspan="3">Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>El jugador pulsa el botón 'Salir'.</td></tr>
<tr><td>2</td><td>El sistema cierra el juego.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador cierra la aplicación.</td></tr>
<tr><td rowspan="4">Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>El jugador pulsa el botón 'Inicio'. Se ejecutaría el caso de uso Iniciar partida.</td></tr>
<tr><td>1</td><td>El jugador pulsa el botón 'Créditos', el juego muestra la pantalla de créditos.</td></tr>
<tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.32: Descripción del caso de uso Salir de la Partida.

<table>
<tr><td>UC-03</td><td colspan="2">Moverse a la derecha</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-03.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td rowspan="3">Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>El jugador pulsa el botón de movimiento lateral a la derecha.</td></tr>
<tr><td>2</td><td>El sistema hace que el protagonista se mueva a la derecha.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El protagonista se mueve a la derecha.</td></tr>
<tr><td rowspan="3">Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td>2</td><td>Hay una pared o el jugador deja de pulsar el botón de movimiento lateral a la derecha. El protagonista se para.</td></tr>
<tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.33: Descripción del caso de uso Moverse a la Derecha.

<table>
<tr><td>UC-04</td><td colspan="2">Moverse a la izquierda</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-04.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa el botón de movimiento lateral a la izquierda.</td></tr><tr><td>2</td><td>El sistema hace que el protagonista se mueva a la izquierda.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El protagonista se mueve a la izquierda.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>2</td><td>Hay una pared o el jugador deja de pulsar el botón de movimiento lateral a la izquierda. El protagonista se para.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.34: Descripción del caso de uso Moverse a la Izquierda.

<table>
<tr><td>UC-05</td><td colspan="2">Moverse hacia abajo</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-05.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa el botón de movimiento vertical hacia abajo.</td></tr><tr><td>2</td><td>El sistema hace que el protagonista se mueva hacia abajo.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El protagonista se mueve hacia abajo.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>2</td><td>Hay una pared o el jugador deja de pulsar el botón de movimiento vertical hacia abajo. El protagonista se para.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.35: Descripción del caso de uso Moverse Abajo.

<table>
<tr><td>UC-06</td><td colspan="2">Moverse hacia arriba</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-06.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa el botón de movimiento vertical hacia arriba.</td></tr><tr><td>2</td><td>El sistema hace que el protagonista se mueva hacia arriba.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El protagonista se mueve hacia arriba.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>2</td><td>Hay una pared o el jugador deja de pulsar el botón de movimiento vertical hacia arriba. El protagonista se para.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.36: Descripción del caso de uso Moverse Arriba.

<table>
<tr><td>UC-07</td><td colspan="2">Abrir inventario</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-07.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa el botón 'Inventario'.</td></tr><tr><td>2</td><td>El sistema muestra la pantalla Inventario.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">Se muestra el inventario del jugador.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.37: Descripción del caso de uso Abrir Inventario.

<table>
<tr><td>UC-08</td><td colspan="2">Entrar en habitación</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-08.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en la pantalla Pasillo.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador está cerca de una puerta.</td></tr><tr><td>2</td><td>El sistema activa el botón 'Puerta'.</td></tr><tr><td>3</td><td>El jugador pulsa el botón 'Puerta'.</td></tr><tr><td>4</td><td>El sistema muestra la pantalla de la habitación a la que corresponda la puerta.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador entra en la habitación.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador está lejos de una puerta. El botón 'Puerta' está desactivado y no puede usarse.</td></tr><tr><td>3</td><td>El jugador se aleja de la puerta y el botón 'Puerta' vuelve a desactivarse.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.38: Descripción del caso de uso Entrar en Habitación.

<table>
<tr><td>UC-09</td><td colspan="2">Salir de habitación</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-09.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado NORMAL.</td></tr><tr><td>2</td><td>El jugador pulsa el botón 'Puerta'.</td></tr><tr><td>3</td><td>El sistema muestra la pantalla Pasillo.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador sale al pasillo.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en otro estado distinto a NORMAL. El botón 'Puerta' está desactivado y no puede usarse.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.39: Descripción del caso de uso Salir Habitación.

<table>
<tr><td>UC-10</td><td>Investigar habitación</td></tr>
<tr><td>Descripción</td><td>El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-10.</td></tr>
<tr><td>Precondición</td><td>El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>La habitación se encuentra en el estado NORMAL.</td></tr>
<tr><td>2</td><td>El jugador pulsa el botón 'Investigar'.</td></tr>
<tr><td>3</td><td>El sistema hace que la habitación pase al estado INVESTIGAR.</td></tr>
</table></td></tr>
<tr><td>Postcondicion</td><td>El jugador puede investigar la habitación en la que se encuentra.</td></tr>
<tr><td>Excepciones</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>La habitación se encuentra en el estado CONVERSAR. El botón 'Investigar' está desactivado y no puede usarse.</td></tr>
<tr><td>2</td><td>La habitación se encuentra en el estado INVESTIGAR. Se ejecuta el caso de uso UC-11.</td></tr>
<tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table></td></tr>
</table>

Cuadro 4.40: Descripción del caso de uso Investigar Habitación.

<table>
<tr><td>UC-11</td><td colspan="2">Terminar investigación</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-11.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado INVESTIGAR.</td></tr><tr><td>2</td><td>El jugador pulsa el botón 'Investigar'.</td></tr><tr><td>3</td><td>El sistema hace que la habitación pase al estado NORMAL.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador deja de investigar la habitación en la que se encuentra.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado CONVERSAR. El botón 'Investigar' está desactivado y no puede usarse.</td></tr><tr><td>2</td><td>La habitación se encuentra en el estado NORMAL. Se ejecuta el caso de uso UC-10.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.41: Descripción del caso de uso Terminar Investigación.

<table>
<tr><td>UC-12</td><td colspan="2">Conversar</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-12.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado NORMAL.</td></tr><tr><td>2</td><td>El jugador pulsa el botón 'Conversar'.</td></tr><tr><td>3</td><td>El sistema hace que la habitación pase al estado CONVERSAR. Se inicia la conversación con el personaje que se encuentre en la habitación.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador inicia una conversación</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado otro estado distinto a NORMAL. El botón 'Conversar' está desactivado y no puede usarse.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.42: Descripción del caso de uso Conversar.

<table>
<tr><td>UC-13</td><td>Terminar conversación</td></tr>
<tr><td>Descripción</td><td>El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-13.</td></tr>
<tr><td>Precondición</td><td>El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>La habitación se encuentra en el estado CONVERSAR.</td></tr>
<tr><td>2</td><td>Si la conversación a terminado el sistema muestra el botón 'Fin Conversación'.</td></tr>
<tr><td>3</td><td>El jugador pulsa el botón 'Fin Conversación'.</td></tr>
<tr><td>4</td><td>El sistema hace que la habitación pase al estado NORMAL.</td></tr>
</table></td></tr>
<tr><td>Postcondicion</td><td>El jugador deja de conversar con el personaje que se encuentra la habitación.</td></tr>
<tr><td>Excepciones</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>2</td><td>La conversación aún no ha terminado. El sistema muestra el botón Siguiente conversación y la conversación continua.</td></tr>
<tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table></td></tr>
</table>

Cuadro 4.43: Descripción del caso de uso Terminar Conversación.

<table>
<tr><td>UC-14</td><td colspan="2">Decidir</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-14.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado CONVERSAR.</td></tr><tr><td>2</td><td>El sistema hace que la habitación pase al estado DECISIÓN.</td></tr><tr><td>3</td><td>El jugador pulsa sobre una de las opciones que se le presentan.</td></tr><tr><td>4</td><td>El sistema hace que la habitación pase al estado CONVERSAR.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador escoge una respuesta y vuelve a conversar con el personaje de la habitación.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.44: Descripción del caso de uso Decidir.

<table>
<tr><td>UC-15</td><td colspan="2">Coger objeto</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-15.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Habitación.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>La habitación se encuentra en el estado INVESTIGAR.</td></tr><tr><td>2</td><td>El jugador pulsa sobre uno de los objetos que están en la habitación.</td></tr><tr><td>3</td><td>Si el sistema permite coger ese objeto, este pasa de la habitación al inventario del jugador.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador deja de conversar con el personaje que se encuentra la habitación.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>3</td><td>El sistema no permite coger ese objeto. El número de errores crece y la puntuación recibida al final del puzle disminuye.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.45: Descripción del caso de uso Coger Objeto.

<table>
<tr><td>UC-16</td><td colspan="2">Cerrar inventario</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-16.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Inventario.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El inventario se encuentra en el estado NORMAL.</td></tr><tr><td>2</td><td>El jugador pulsa sobre el botón 'Cerrar Inventario'.</td></tr><tr><td>3</td><td>El sistema muestra la textura del pasillo.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador pasa del inventario al pasillo.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El inventario se encuentra en el estado COMBINANDO o COMBINACIÓN_PREPARADA. El botón 'Cerrar Inventario' está desactivado y no puede usarse.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.46: Descripción del caso de uso Cerrar Inventario.

<table>
<tr><td>UC-17</td><td>Combinar objetos</td></tr>
<tr><td>Descripción</td><td>El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-17.</td></tr>
<tr><td>Precondición</td><td>El jugador ha iniciado una partida y se encuentra en una pantalla Inventario.</td></tr>
<tr><td>Secuencia</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>El inventario se encuentra en el estado NORMAL.</td></tr>
<tr><td>2</td><td>El jugador pulsa sobre el botón 'Combinar'.</td></tr>
<tr><td>3</td><td>El sistema hace que el inventario pase al estado COMBINANDO.</td></tr>
<tr><td>4</td><td>El jugador selecciona el primer objeto a combinar.</td></tr>
<tr><td>5</td><td>El sistema registra el objeto.</td></tr>
<tr><td>6</td><td>El jugador selecciona el segundo objeto a combinar.</td></tr>
<tr><td>7</td><td>El sistema registra el objeto y comprueba si se pueden combinar ambos objetos. Si se puede hace que el inventario pase al estado COMBINACIÓN_PREPARADA y se activa el botón Aceptar Combinación'.</td></tr>
<tr><td>8</td><td>El jugado pulsa el botón 'Aceptar Combinación'.</td></tr>
<tr><td>9</td><td>El sistema borra ambos objetos del inventario y añade el resultado de la combinación de ambos.</td></tr>
</table></td></tr>
<tr><td>Postcondicion</td><td>El jugador recibe el objeto resultado de la unión de ambos objetos y pierde estos últimos.</td></tr>
<tr><td>Excepciones</td><td><table>
<tr><td>Pasos</td><td>Acción</td></tr>
<tr><td>1</td><td>El inventario se encuentra en el estado COMBINAR o COMBINACIÓN_PREPARADA. El botón 'Combinar' está desactivado y no puede usarse.</td></tr>
<tr><td>6</td><td>El jugador vuelve a pulsar sobre el mismo objeto. El sistema elimina el objeto del proceso de combinación.</td></tr>
<tr><td>7</td><td>Ambos objetos no pueden combinarse, el inventario permanece en el estado COMBINANDO y no se activa el botón 'Aceptar Combinación'.</td></tr>
<tr><td>8</td><td>El jugador vuelve a pulsar uno de los objetos seleccionados. El sistema lo elimina del proceso de combinación y el inventario vuelve al estado COMBINANDO.</td></tr>
<tr><td>4-8</td><td>El jugador pulsa el botón 'Cancelar Combinación' y el inventario vuelve al estado NORMAL.</td></tr>
<tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table></td></tr>
</table>

Cuadro 4.47: Descripción del caso de uso Combinar Objetos.

<table>
<tr><td>UC-18</td><td colspan="2">Cancelar combinación</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-18.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y se encuentra en una pantalla Inventario.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El inventario se encuentra en el estado COMBINANDO o COMBINACIÓN_PREPARADA.</td></tr><tr><td>2</td><td>El jugador pulsa sobre el botón 'Cancelar Combinación'.</td></tr><tr><td>3</td><td>El sistema hace que el inventario pase al estado NORMAL.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador pasa del inventario al pasillo.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.48: Descripción del caso de uso Cancelar Combinación.

<table>
<tr><td>UC-19</td><td colspan="2">Mostrar descripción</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-19.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida, se encuentra en una pantalla Inventario y el contiene algún objeto.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El inventario se encuentra en el estado NORMAL.</td></tr><tr><td>2</td><td>El jugador pulsa un objeto que contenga el inventario.</td></tr><tr><td>3</td><td>El sistema muestra la descripción del objeto en el cuadro de descripciones.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador puede visualizar la descripción del objeto seleccionado.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.49: Descripción del caso de uso Mostrar descripción.

<table>
<tr><td>UC-20</td><td colspan="2">Mostrar objetivo</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-20.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida, se encuentra en una pantalla Inventario.</td></tr>
<tr><td>Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El sistema muestra el objetivo actual del jugador en el cuadro de objetivos.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador puede visualizar una descripción de su próximo objetivo.</td></tr>
<tr><td>Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.50: Descripción del caso de uso Mostrar objetivo.

<table>
<tr><td>UC-21</td><td colspan="2">Actualizar objetivo</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-21.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida y ha completado un subestado del puzle que está resolviendo.</td></tr>
<tr><td>Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El sistema actualiza el texto del objetivo con el siguiente paso que tiene que dar el jugador.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El sistema actualiza el objetivo.</td></tr>
<tr><td>Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.51: Descripción del caso de uso Actualizar Objetivo.

<table>
<tr><td>UC-22</td><td colspan="2">Elegir asesino</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-22.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida se encuentra en la pantalla de selección de asesino.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa sobre uno de los posibles asesinos.</td></tr><tr><td>2</td><td>El sistema comprueba si el personaje escogido es el culpable. Si lo es, el sistema muestra la pantalla de selección de arma.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El jugador pasa a la pantalla de selección de arma.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>2</td><td>Si el personaje elegido no es el asesino se restan puntos a la puntuación final del jugador, se añade un error más y el jugador continúa en la pantalla de selección de asesino.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.52: Descripción del caso de uso Elegir Asesino.

<table>
<tr><td>UC-23</td><td colspan="2">Elegir arma</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-23.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha iniciado una partida se encuentra en la pantalla de selección de arma.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa sobre una de las posibles armas.</td></tr><tr><td>2</td><td>El sistema comprueba si el arma escogida es la usada por el asesino. Si lo es, el juego termina y se registra la puntuación y el número de errores en un fichero.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">El juego termina y se registran los resultados.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>2</td><td>Si el arma elegida no es la usada por el asesino, se restan puntos a la puntuación final del jugador, se añade un error más y el jugador continúa en la pantalla de selección de arma.</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.53: Descripción del caso de uso Elegir Arma.

<table>
<tr><td>UC-24</td><td colspan="2">Sumar puntos</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-24.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha completado un puzle.</td></tr>
<tr><td>Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El sistema añade puntos al contador del jugador siguiendo la siguiente fórmula. PuntosPorRonda = 1000 - ErroresPorRonda * 100. Sin ser menor que 0.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El sistema añade puntos al jugador.</td></tr>
<tr><td>Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.54: Descripción del caso de uso Sumar Puntos.

<table>
<tr><td>UC-25</td><td colspan="2">Actualizar fichero de log</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-25.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador ha realizado una acción que se ha de registrar en el fichero de log.</td></tr>
<tr><td>Secuencia</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>1</td><td>El sistema, siguiendo una leyenda propuesta anteriormente, actualiza el fichero de log con la acción que el jugador acaba de realizar.</td></tr>
<tr><td>Postcondicion</td><td colspan="2">El sistema actualiza el fichero de log.</td></tr>
<tr><td>Excepciones</td><td>Pasos</td><td>Acción</td></tr>
<tr><td></td><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr>
</table>

Cuadro 4.55: Descripción del caso de uso Actualizar Fichero de Log.

UC-26	Subir fichero de log	
Descripción	El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-26.	
Precondición	El jugador se encuentra en la pantalla Final y tiene conexión a Internet.	
Secuencia	Pasos	Acción
	1	El sistema, sube el fichero de log al servidor de Dropbox.
Postcondicion	El fichero de log se sube correctamente a Dropbox.	
Excepciones	Pasos	Acción
	*	El jugador cierra el juego y el caso de uso queda sin efecto.

Cuadro 4.56: Descripción del caso de uso Subir Fichero de Log.

UC-27	Ver créditos	
Descripción	El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-27.	
Precondición	El jugador se encuentra en la pantalla de inicio.	
Secuencia	Pasos	Acción
	1	El jugador pulsa el botón Créditos.
	2	El sistema muestra la pantalla de créditos.
Postcondicion	Se muestra los créditos del juego.	
Excepciones	Pasos	Acción
	*	El jugador cierra el juego y el caso de uso queda sin efecto.

Cuadro 4.57: Descripción del caso de uso Ver Créditos.

UC-28	Abrir diccionario	
Descripción	El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-28.	
Precondición	El jugador se encuentra en la pantalla Pasillo.	
Secuencia	Pasos	Acción
	1	El jugador pulsa el botón *'Abrir Diccionario'*.
	2	El sistema muestra la pantalla Diccionario.
Postcondicion	Se muestra el diccionario.	
Excepciones	Pasos	Acción
	*	El jugador cierra el juego y el caso de uso queda sin efecto.

Cuadro 4.58: Descripción del caso de uso Abrir diccionario.

<table>
<tr><td>UC-29</td><td colspan="2">Cerrar diccionario</td></tr>
<tr><td>Descripción</td><td colspan="2">El sistema se debe comportar tal y como se explica en el requisito funcional FRQ-29.</td></tr>
<tr><td>Precondición</td><td colspan="2">El jugador se encuentra en la pantalla Diccionario.</td></tr>
<tr><td>Secuencia</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>1</td><td>El jugador pulsa el botón 'Cerrar Diccionario'.</td></tr><tr><td>2</td><td>El sistema muestra la pantalla Pasillo.</td></tr></table></td></tr>
<tr><td>Postcondicion</td><td colspan="2">Se cierra el diccionario y permite al jugador seguir con la partida.</td></tr>
<tr><td>Excepciones</td><td colspan="2"><table><tr><td>Pasos</td><td>Acción</td></tr><tr><td>*</td><td>El jugador cierra el juego y el caso de uso queda sin efecto.</td></tr></table></td></tr>
</table>

Cuadro 4.59: Descripción del caso de uso Cerrar diccionario.

4.3. Requisitos no funcionales

NFR-01	Adecuación funcional
Descripción	El sistema debe cumplir con las necesidades del cliente, en este caso el jugador. Este requisito solo se satisface si las funcionalidades del sistema son adecuadas con lo que se ha pedido y si cubren todos los objetivos propuestos.

Cuadro 4.60: Requisito no funcional Adecuación funcional

NFR-02	Compatibilidad
Descripción	El sistema debe ser compatible con los dispositivos Android. También con los sistemas operativos Windows y Linux.

Cuadro 4.61: Requisito no funcional Compatibilidad

NFR-03	Usabilidad
Descripción	El sistema tener una interfaz de usuario fácil de usar e intuitiva.

Cuadro 4.62: Requisito no funcional Usabilidad

4.4. Diagrama de clases

A continuación se muestra el diagrama ER, donde se puede ver la relación entre las clases que forman el videojuego. Además se añade una pequeña descripción de las entidades y las relaciones que existen entre ellas.

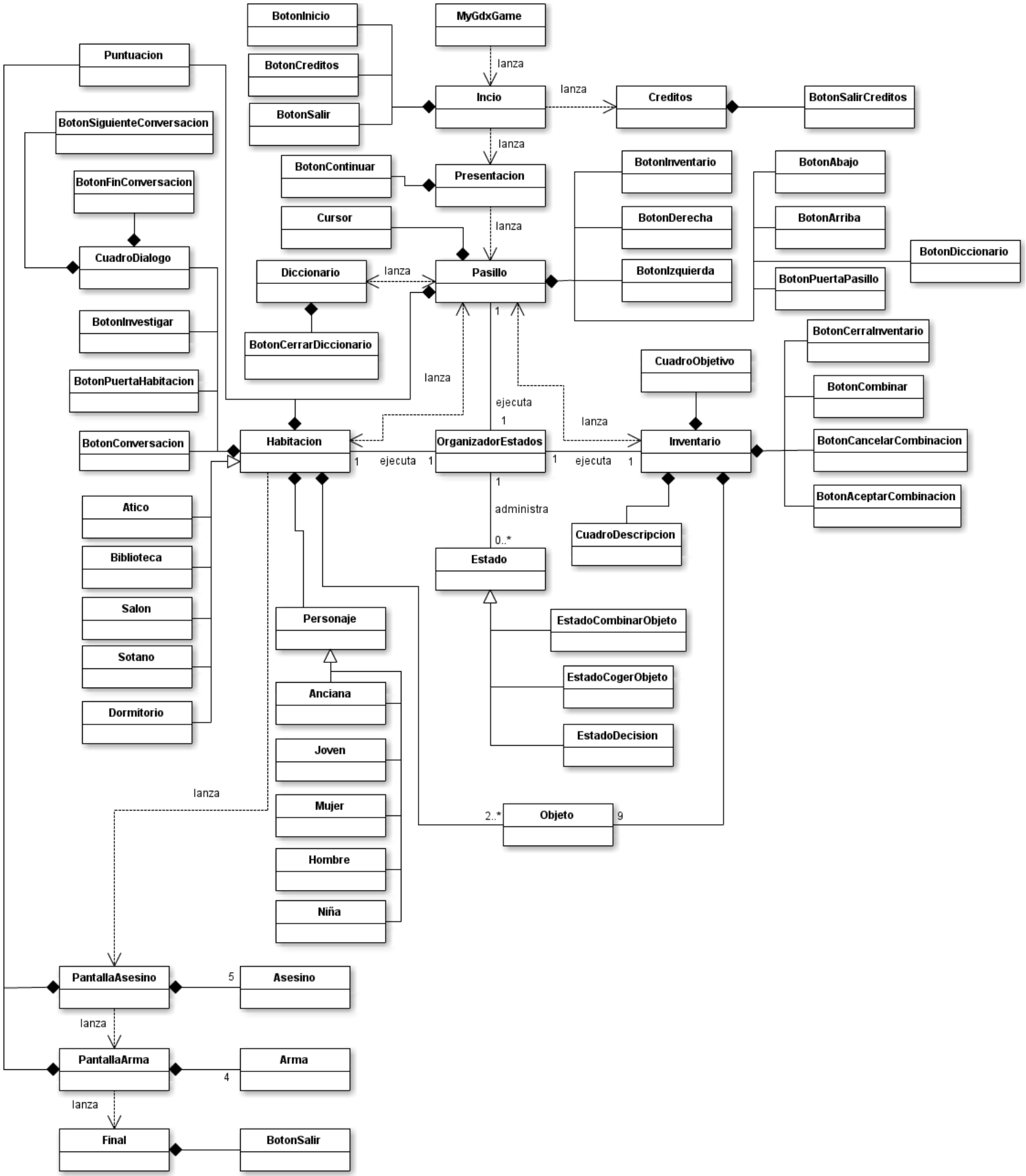


Figura 4.7: Diagrama de clases.

4.5. Modelo de comportamiento del sistema

El modelo de comportamiento describe cómo actúa el sistema. Consta del diagrama de secuencia, que presenta la secuencia de eventos entre los actores y el sistema. Y el contrato de las operaciones, que describe el efecto de dichas operaciones.
A continuación se muestran los diagrama de secuencia y operaciones del sistema mediante notación UML.
Caso de uso: Iniciar Partida

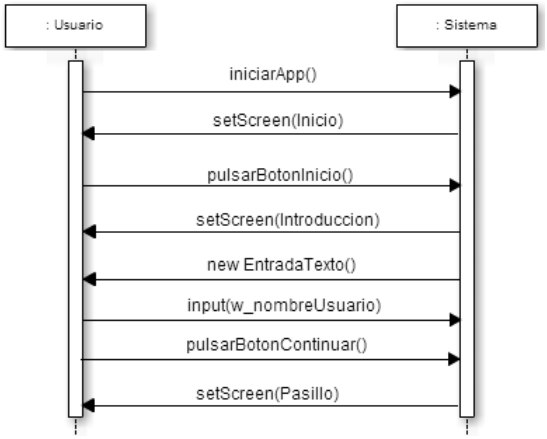


Figura 4.8: Diagrama de secuencia: Iniciar Partida

- Operación: iniciarApp()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario inicia el juego.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: Ninguna.
  - Postcondiciones: Se inicia el juego.
- Operación: setScreen(Inicio)
  - Actores: Usuario, Sistema.
  - Responsabilidades: Mostrar la pantalla de inicio.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario ha iniciado el juego.
  - Postcondiciones: Se muestra la pantalla de inicio.
- Operación: pulsarBotonInicio()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón de inicio.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario visualiza la pantalla de inicio.

- Postcondiciones: El sistema cambia de pantalla.

- Operación: setScreen(Introduccion)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema muestra la pantalla de introducción.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario ha pulsado el botón de inicio.
  - Postcondiciones: El sistema muestra la pantalla de introducción.
- Operación: newEntradaTexto()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema crea un nuevo cuadro de texto.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario visualiza la pantalla de presentación.
  - Postcondiciones: El usuario puede escribir en el nuevo cuadro de texto.
- Operación: input(w_nombreUsuario)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario escribe su nombre y apellidos en el cuadro de texto.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario visualiza el cuadro de texto.
  - Postcondiciones: El usuario introduce su nombre y apellidos.
- Operación: pulsarBotonContinuar()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón continuar.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario visualiza la pantalla de presentación y ha introducido su nombre y apellidos.
  - Postcondiciones: El sistema cambia de pantalla.
- Operación: setScreen(Pasillo)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema muestra la pantalla Pasillo.
  - Referencias cruzadas: Caso de uso Iniciar Partida.
  - Precondiciones: El usuario ha pulsado el botón Continuar.
  - Postcondiciones: El sistema muestra la pantalla Pasillo.

Caso de uso: Salir de la Partida

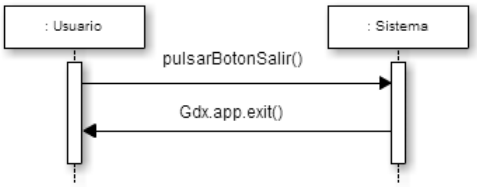


Figura 4.9: Diagrama de secuencia: Salir de la partida

- Operación: pulsarBotonSalir()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Salir.
  - Referencias cruzadas: Caso de uso Salir de la Partida.
  - Precondiciones: El usuario visualiza la pantalla Inicio o la pantalla Final.
  - Postcondiciones: Se activa la lógica de botón Salir.
- Operación: Gdx.app.exit()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cierra la partida.
  - Referencias cruzadas: Caso de uso Salir de la partida.
  - Precondiciones: El usuario ha pulsado el botón Salir.
  - Postcondiciones: El sistema cierra la aplicación.

Caso de uso: Moverse a la derecha

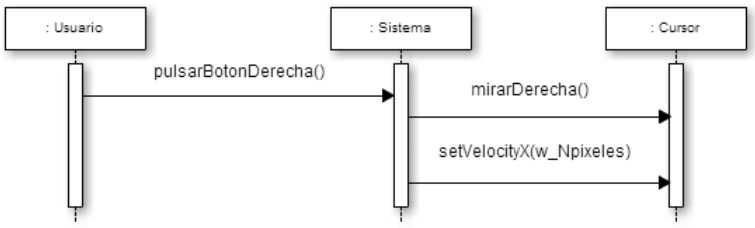


Figura 4.10: Diagrama de secuencia: Moverse a la derecha

- Operación: pulsarBotonDerecha()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón de desplazamiento lateral derecho.
  - Referencias cruzadas: Caso de uso Moverse a la derecha.

- Precondiciones: El usuario está en la pantalla Pasillo y visualiza el botón Derecha.
- Postcondiciones: Se activa la lógica del botón.

- Operación: mirarDerecha()
  - Actores: Sistema, Cursor.
  - Responsabilidades: El sistema carga la textura en la que el cursor mira a la derecha.
  - Referencias cruzadas: Caso de uso Moverse a la derecha.
  - Precondiciones: El usuario ha pulsado el botón Derecha.
  - Postcondiciones: El sistema carga la textura correcta.
- Operación: setVelocityX(w_Npixeles)
  - Actores: Sistema, Cursor.
  - Responsabilidades: El cursor aumenta su variable X de posición.
  - Referencias cruzadas: Caso de uso Moverse a la derecha.
  - Precondiciones: El usuario ha pulsado el botón Derecha.
  - Postcondiciones: El cursor se desplaza a la derecha.

Caso de uso: Moverse a la izquierda

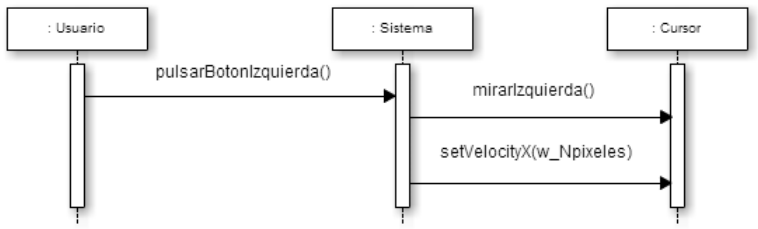


Figura 4.11: Diagrama de secuencia: Moverse a la izquierda

- Operación: pulsarBotonIzquierda()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón de desplazamiento lateral izquierdo.
  - Referencias cruzadas: Caso de uso Moverse a la izquierda.
  - Precondiciones: El usuario está en la pantalla Pasillo y visualiza el botón Izquierda.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: mirarIzquierda()
  - Actores: Sistema, Cursor.
  - Responsabilidades: El sistema carga la textura en la que el cursor mira a la izquierda.
  - Referencias cruzadas: Caso de uso Moverse a la izquierda.
  - Precondiciones: El usuario ha pulsado el botón Izquierda.

- Postcondiciones: El sistema carga la textura correcta.

- Operación: setVelocityX(w_Npixeles)
  - Actores: Sistema, Cursor.
  - Responsabilidades: El cursor disminuye su variable X de posición.
  - Referencias cruzadas: Caso de uso Moverse a la izquierda.
  - Precondiciones: El usuario ha pulsado el botón Izquierda.
  - Postcondiciones: El cursor se desplaza a la izquierda.

Caso de uso: Moverse hacia abajo

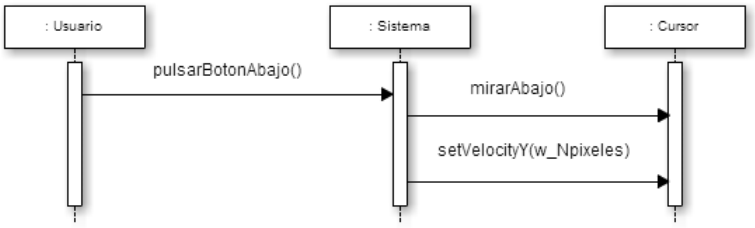


Figura 4.12: Diagrama de secuencia: Moverse hacia abajo

- Operación: pulsarBotonAbajo()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón de desplazamiento vertical hacia abajo.
  - Referencias cruzadas: Caso de uso Moverse hacia abajo.
  - Precondiciones: El usuario está en la pantalla Pasillo y visualiza el botón Abajo.
  - Postcondiciones: Se activa la lógica del botón.

- Operación: mirarAbajo()
  - Actores: Sistema, Cursor.
  - Responsabilidades: El sistema carga la textura en la que el cursor mira hacia Abajo.
  - Referencias cruzadas: Caso de uso Moverse hacia abajo.
  - Precondiciones: El usuario ha pulsado el botón Abajo.
  - Postcondiciones: El sistema carga la textura correcta.

- Operación: setVelocityY(w_Npixeles)
  - Actores: Sistema, Cursor.
  - Responsabilidades: El cursor disminuye su variable Y de posición.
  - Referencias cruzadas: Caso de uso Moverse hacia abajo.
  - Precondiciones: El usuario ha pulsado el botón Abajo.
  - Postcondiciones: El cursor se desplaza hacia abajo.

Caso de uso: Moverse hacia arriba

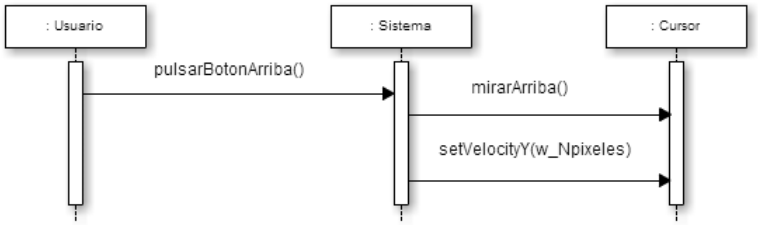


Figura 4.13: Diagrama de secuencia: Moverse hacia arriba

- Operación: pulsarBotonArriba()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón de desplazamiento vertical hacia arriba.
  - Referencias cruzadas: Caso de uso Moverse hacia arriba.
  - Precondiciones: El usuario está en la pantalla Pasillo y visualiza el botón Arriba.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: mirarArriba()
  - Actores: Sistema, Cursor.
  - Responsabilidades: El sistema carga la textura en la que el cursor mira hacia Arriba.
  - Referencias cruzadas: Caso de uso Moverse hacia arriba.
  - Precondiciones: El usuario ha pulsado el botón Arriba.
  - Postcondiciones: El sistema carga la textura correcta.
- Operación: setVelocityY(w_Npixeles)
  - Actores: Sistema, Cursor.
  - Responsabilidades: El cursor aumenta su variable Y de posición.
  - Referencias cruzadas: Caso de uso Moverse hacia arriba.
  - Precondiciones: El usuario ha pulsado el botón Arriba.
  - Postcondiciones: El cursor se desplaza hacia arriba.

Caso de uso: Abrir inventario

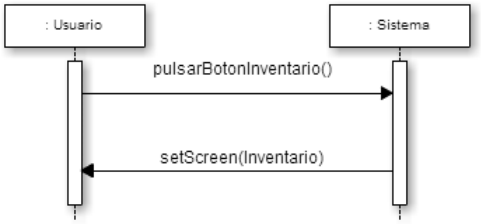


Figura 4.14: Diagrama de secuencia: Abrir inventario

- Operación: pulsarBotonInventario()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Inventario.
  - Referencias cruzadas: Caso de uso Abrir inventario.
  - Precondiciones: El usuario está en la pantalla Pasillo y visualiza el botón Inventario.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Inventario)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Abrir inventario.
  - Precondiciones: El usuario ha pulsado el botón Inventario.
  - Postcondiciones: El sistema muestra la pantalla Inventario.

Caso de uso: Entrar en habitación

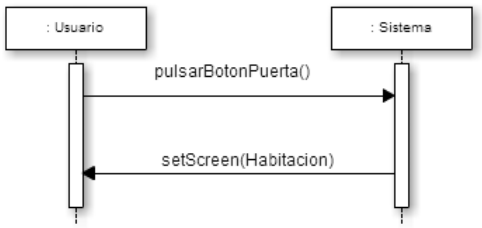


Figura 4.15: Diagrama de secuencia: Entrar en habitación

- Operación: pulsarBotonPuerta()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Puerta que hay en el pasillo.
  - Referencias cruzadas: Caso de uso Entrar en habitación.
  - Precondiciones: El jugador está cerca de una puerta y el botón está activado.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Habitacion)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Entrar en habitación .
  - Precondiciones: El usuario ha pulsado el botón Puerta del pasillo.
  - Postcondiciones: El sistema muestra la habitación que corresponde con la puerta por la que ha entrado el usuario.

Caso de uso: Salir de habitación

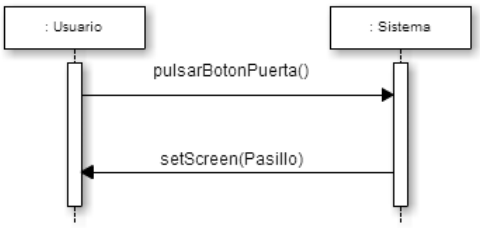


Figura 4.16: Diagrama de secuencia: Salir de habitación

- Operación: pulsarBotonPuerta()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Puerta que hay en la habitación.
  - Referencias cruzadas: Caso de uso Salir de habitación.
  - Precondiciones: El jugador está dentro de una habitación.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Pasillo)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Salir de habitación.
  - Precondiciones: El usuario ha pulsado el botón Puerta de la habitación.
  - Postcondiciones: El sistema muestra la pantalla Pasillo.

Caso de uso: Investigar habitación

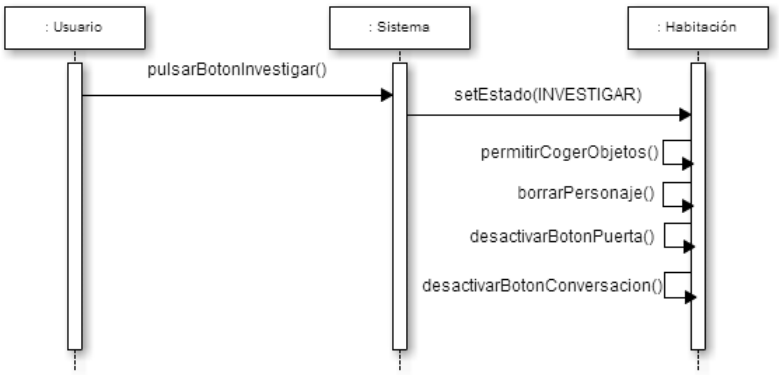


Figura 4.17: Diagrama de secuencia: Investigar habitación

- Operación: pulsarBotonInvestigar()
  - Actores: Usuario, Sistema.

- Responsabilidades: El usuario pulsa el botón Investigar.
- Referencias cruzadas: Caso de uso Investigar habitación.
- Precondiciones: El usuario está dentro de una habitación. La habitación se encuentra en el estado NORMAL.
- Postcondiciones: Se activa la lógica del botón.

- Operación: setEstado(INVESTIGAR)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable de estado de la habitación.
  - Referencias cruzadas: Caso de uso Investigar habitación.
  - Precondiciones: El usuario ha pulsado el botón Investigar.
  - Postcondiciones: La habitación pasa al estado INVESTIGAR.
- Operación: permitirCogerObjetos()
  - Actores: Habitación.
  - Responsabilidades: La habitación permite al usuario interactuar con los objetos que esta contenga.
  - Referencias cruzadas: Caso de uso Investigar habitación.
  - Precondiciones: La habitación se encuentra en el estado INVESTIGAR.
  - Postcondiciones: El usuario puede interactuar con los objetos de la habitación.
- Operación: borrarPersonaje()
  - Actores: Habitación.
  - Responsabilidades: La habitación hace que el personaje que haya en ella desaparezca de la pantalla.
  - Referencias cruzadas: Caso de uso Investigar habitación.
  - Precondiciones: La habitación se encuentra en el estado INVESTIGAR.
  - Postcondiciones: La textura del personaje desaparece de la pantalla.
- Operación: desactivarBotonPuerta()
  - Actores: Habitación.
  - Responsabilidades: El botón Puerta pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Investigar habitación.
  - Precondiciones: La habitación se encuentra en el estado INVESTIGAR.
  - Postcondiciones: El usuario no puede salir de la habitación.
- Operación: desactivarBotonConversación()
  - Actores: Habitación.
  - Responsabilidades: El botón Conversación pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Investigar habitación.

- Precondiciones: La habitación se encuentra en el estado INVESTIGAR.
- Postcondiciones: El usuario no puede conversar con el personaje.

Caso de uso: Terminar investigación

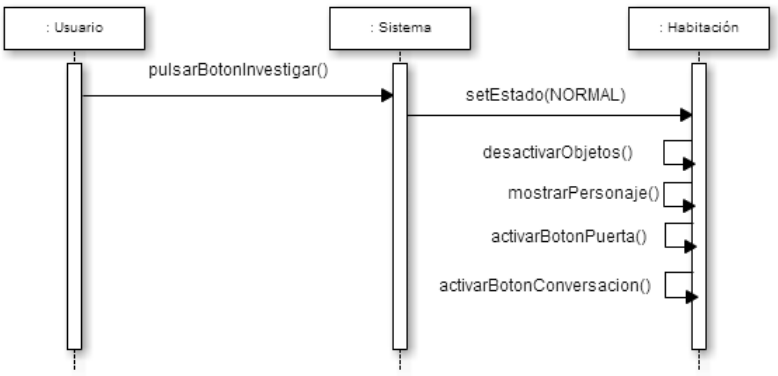


Figura 4.18: Diagrama de secuencia: Terminar investigación

- Operación: pulsarBotonInvestigar()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Investigar.
  - Referencias cruzadas: Caso de uso Terminar investigación.
  - Precondiciones: El usuario está dentro de una habitación. La habitación se encuentra en el estado INVESTIGAR.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setEstado(NORMAL)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable de estado de la habitación.
  - Referencias cruzadas: Caso de uso Terminar investigación.
  - Precondiciones: El usuario ha pulsado el botón Investigar.
  - Postcondiciones: La habitación pasa al estado NORMAL.
- Operación: desactivarObjetos()
  - Actores: Habitación.
  - Responsabilidades: La habitación no permite que el usuario interactúe con los objetos que esta contenga.
  - Referencias cruzadas: Caso de uso Terminar investigación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario deja de poder pulsar sobre los objetos.
- Operación: mostrarPersonaje()

- Actores: Habitación.
- Responsabilidades: El personaje vuelve a añadirse al stage.
- Referencias cruzadas: Caso de uso Terminar investigación.
- Precondiciones: La habitación se encuentra en el estado NORMAL.
- Postcondiciones: La textura del personaje vuelve a aparecer en la pantalla.

▪ Operación: activarBotonPuerta()

- Actores: Habitación.
- Responsabilidades: El botón Puerta pasa a estar activo.
- Referencias cruzadas: Caso de uso Terminar investigación.
- Precondiciones: La habitación se encuentra en el estado NORMAL.
- Postcondiciones: El usuario puede salir de la habitación.

▪ Operación: activarBotonConversacion()

- Actores: Habitación.
- Responsabilidades: El botón Conversación pasa a estar activo.
- Referencias cruzadas: Caso de uso Terminar investigación.
- Precondiciones: La habitación se encuentra en el estado NORMAL.
- Postcondiciones: El usuario puede conversar con el personaje.

Caso de uso: Conversar

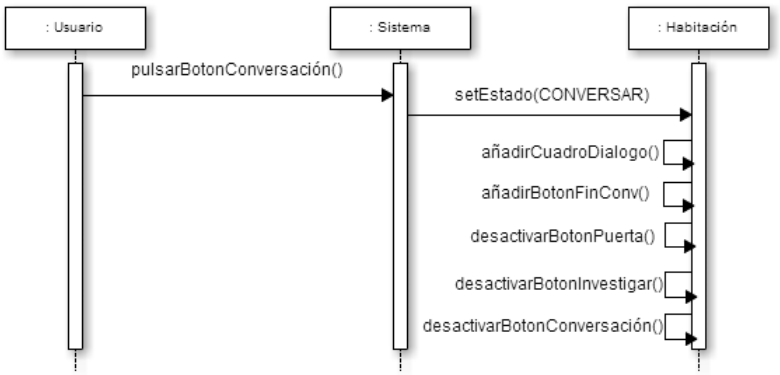


Figura 4.19: Diagrama de secuencia: Conversar

▪ Operación: pulsarBotonConversacion()

- Actores: Usuario, Sistema.
- Responsabilidades: El usuario pulsa el botón Conversación.
- Referencias cruzadas: Caso de uso Conversar.
- Precondiciones: El usuario está dentro de una habitación. La habitación se encuentra en el estado NORMAL.

- Postcondiciones: Se activa la lógica del botón.

- Operación: setEstado(CONVERSAR)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable estado de la habitación.
  - Referencias cruzadas: Caso de uso Conversar.
  - Precondiciones: El usuario ha pulsado el botón Conversar.
  - Postcondiciones: La habitación pasa al estado CONVERSAR.
- Operación: añadirCuadroDialogo()
  - Actores: Habitación.
  - Responsabilidades: Se añade el cuadro de diálogo al stage de la habitación.
  - Referencias cruzadas: Caso de uso Conversar.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario visualiza el cuadro de diálogo.
- Operación: añadirBotonFinConv()
  - Actores: Habitación
  - Responsabilidades: Se añade el botón Fin de conversación al stage de la habitación.
  - Referencias cruzadas: Caso de uso Conversar.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario visualiza el botón Fin de conversación.
- Operación: desactivarBotonPuerta()
  - Actores: Habitación
  - Responsabilidades: El botón Puerta pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Conversar.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario no puede salir de la habitación.
- Operación: desactivarBotonInvestigacion()
  - Actores: Habitación
  - Responsabilidades: El botón Investigación pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Conversar.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario no puede investigar la habitación.
- Operación: desactivarBotonConversar()
  - Actores: Habitación
  - Responsabilidades: El botón Conversación pasa a estar inactivo.

- Referencias cruzadas: Caso de uso Conversar.
- Precondiciones: La habitación se encuentra en el estado CONVERSAR.
- Postcondiciones: El usuario no puede iniciar una nueva conversación hasta que no termine la actual.

Caso de uso: Terminar conversación

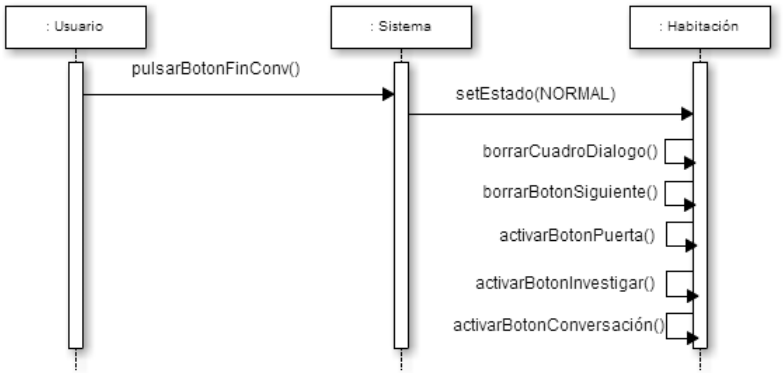


Figura 4.20: Diagrama de secuencia: Terminar conversación

- Operación: pulsarBotonFinConv()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Fin conversación.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: El usuario esta dentro de una habitación. La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setEstado(NORMAL)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable estado de la habitación.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: El usuario ha pulsado el botón Fin conversación.
  - Postcondiciones: La habitación pasa al estado NORMAL.
- Operación: borrarCuadroDialogo()
  - Actores: Habitación.
  - Responsabilidades: Se borra el cuadro de diálogo del stage de la habitación.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El cuadro de diálogo desaparece de la pantalla.

- Operación: borrarBotonFinConv()
  - Actores: Habitación.
  - Responsabilidades: Se borra el botón de Fin de conversación del stage de la habitación.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El botón Fin de conversación desaparece de la pantalla.
- Operación: activarBotonPuerta()
  - Actores: Habitación.
  - Responsabilidades: El botón Puerta pasa a estar activo.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede salir de la habitación.
- Operación: activarBotonInvestigar()
  - Actores:Habitación.
  - Responsabilidades: El botón Investigar pasa a estar activo.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede investigar la habitación.
- Operación: activarBotonConversacion()
  - Actores: Habitación.
  - Responsabilidades: El botón Conversación pasa a estar activo.
  - Referencias cruzadas: Caso de uso Terminar conversación.
  - Precondiciones: La habitación se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede conversar con el personaje.

Caso de uso: Decidir

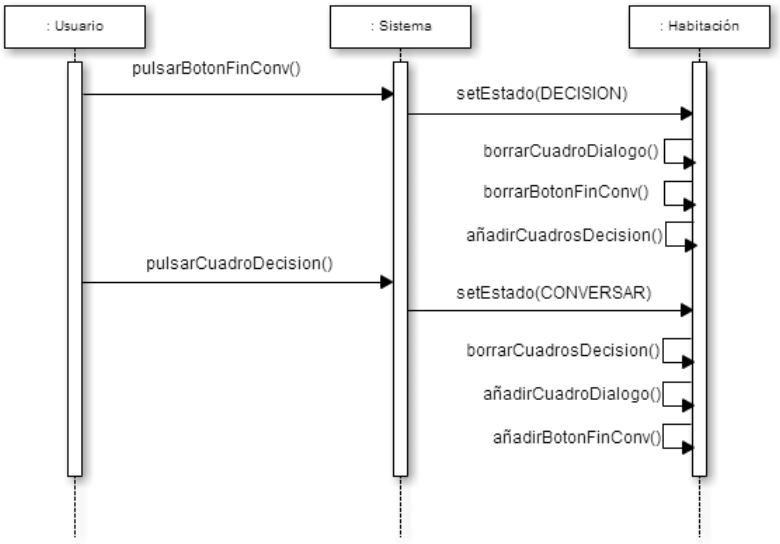


Figura 4.21: Diagrama de secuencia: Decidir

- Operación: pulsarBotonFinConv()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Fin conversación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: El usuario esta dentro de una habitación. La habitación está en el estado CONVERSAR.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setEstado(DECISION)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable estado de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: El usuario ha pulsado el botón Fin conversación.
  - Postcondiciones: La habitación pasa al estado DECISIÓN.
- Operación: borrarCuadroDialogo()
  - Actores: Habitación.
  - Responsabilidades: Se borra el cuadro de diálogo del stage de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado DECISIÓN.
  - Postcondiciones: El cuadro de diálogo desaparece de la pantalla.
- Operación: borrarBotonFinConv()
  - Actores: Habitación.

- Responsabilidades: Se borra el botón de Fin de conversación del stage de la habitación.
- Referencias cruzadas: Caso de uso Decidir.
- Precondiciones: La habitación se encuentra en el estado DECISIÓN.
- Postcondiciones: El botón Fin de conversación desaparece de la pantalla.

- Operación: añadirCuadrosDecision()
  - Actores: Habitación.
  - Responsabilidades: Se añade al stage de la habitación los cuatro cuadros de decisión.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado DECISIÓN.
  - Postcondiciones: El usuario visualiza los cuadros de decisión.
- Operación: pulsarCuadroDecisión()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa uno de los cuadros de decisión.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado DECISIÓN.
  - Postcondiciones: Se elige una de las cuatro respuestas.
- Operación: setEstado(CONVERSAR)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema modifica la variable estado de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: El usuario ha elegido una de las respuestas.
  - Postcondiciones: La habitación pasa al estado CONVERSACIÓN.
- Operación: borrarCuadrosDecision()
  - Actores: Habitación.
  - Responsabilidades: Los cuadros decisión se borrar de la stage de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: Los cuadros decisión desaparecen de la pantalla.
- Operación: añadirCuadroDialogo()
  - Actores: Habitación.
  - Responsabilidades: Se añade el cuadro de diálogo al stage de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario visualiza el cuadro de diálogo.

- Operación: añadirBotonFinConv()
  - Actores: Habitación.
  - Responsabilidades: Se añade el botón Fin de conversación al stage de la habitación.
  - Referencias cruzadas: Caso de uso Decidir.
  - Precondiciones: La habitación se encuentra en el estado CONVERSAR.
  - Postcondiciones: El usuario visualiza el botón Fin de conversación.

Caso de uso: Coger objeto

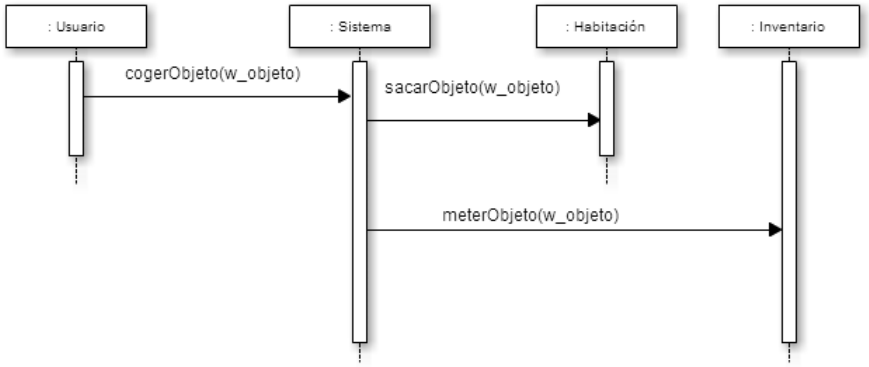


Figura 4.22: Diagrama de secuencia: Coger objeto

- Operación: cogerObjeto(w_objeto)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa sobre un objeto.
  - Referencias cruzadas: Caso de uso Coger objeto.
  - Precondiciones: La habitación se encuentra en el estado INVESTIGACIÓN y el usuario tiene permiso para coger ese objeto.
  - Postcondiciones: El objeto activa su lógica.
- Operación: sacarObjeto(w_objeto)
  - Actores: Sistema, Habitación.
  - Responsabilidades: El sistema debe eliminar el objeto de la habitación
  - Referencias cruzadas: Caso de uso Coger objeto.
  - Precondiciones: El usuario ha pulsado sobre el objeto y puede cogerlo.
  - Postcondiciones: El objeto desaparece de la habitación.
- Operación: meterObjeto(w_objeto)
  - Actores: Sistema Inventario.
  - Responsabilidades: El sistema mete el objeto en el inventario del usuario.
  - Referencias cruzadas: Caso de uso Coger objeto.

- Precondiciones: El usuario ha pulsado sobre el objeto y puede cogerlo.
- Postcondiciones: El objeto aparece en el inventario del usuario.

Caso de uso: Cerrar inventario

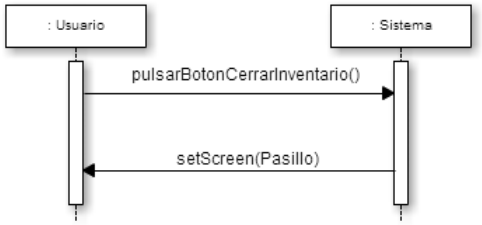


Figura 4.23: Diagrama de secuencia: Cerrar inventario

- Operación: pulsarBotonCerrarInventario()
  - Actores: Usuario, Sistema
  - Responsabilidades: El usuario pulsa sobre el botón Cerrar inventario.
  - Referencias cruzadas: Caso de uso Cerrar inventario.
  - Precondiciones: El usuario se encuentra en la pantalla Inventario. El inventario está en el estado NORMAL.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Pasillo)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Cerrar inventario.
  - Precondiciones: El usuario ha pulsado el botón Cerrar inventario.
  - Postcondiciones: El sistema muestra la pantalla Pasillo.

Caso de uso: Combinar objetos

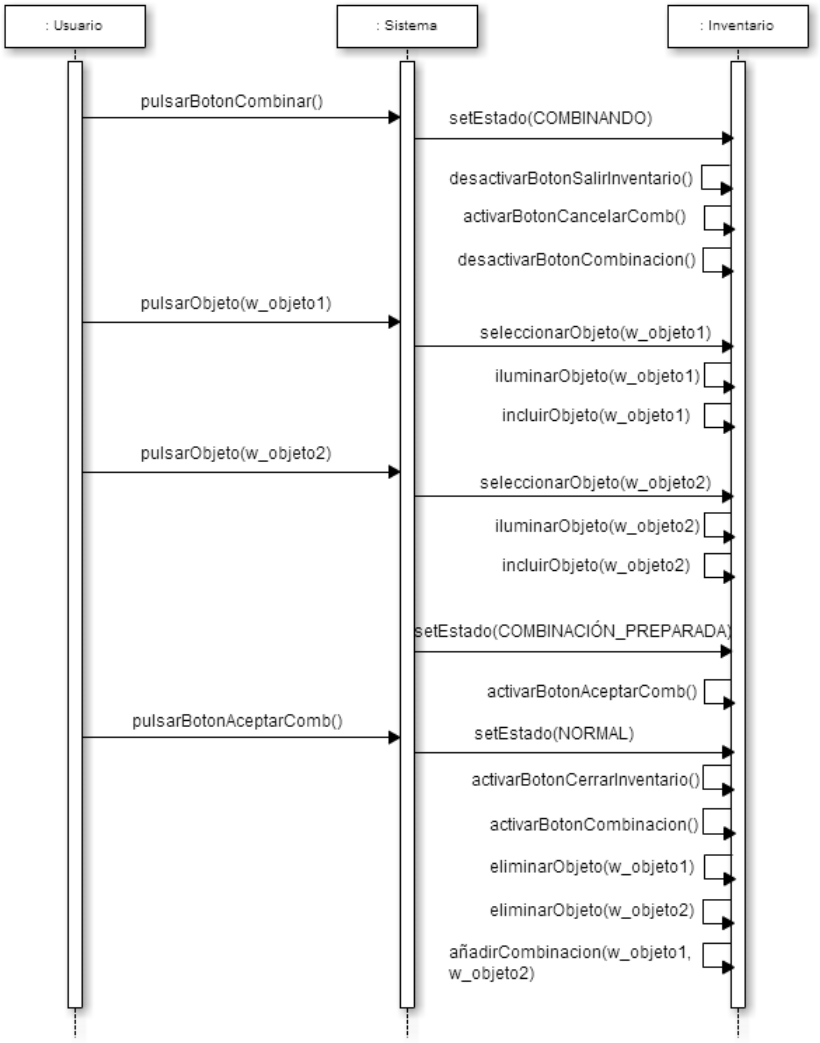


Figura 4.24: Diagrama de secuencia: Combinar objetos

- Operación: pulsarBotonCombinar()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Combinar.
  - Referencias cruzadas: Caso de uso Combinar objetos.
  - Precondiciones: El usuario está en la pantalla Inventario.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setEstado(COMBINANDO)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema cambia la variable estado del inventario.
  - Referencias cruzadas: Caso de uso Combinar objetos.
  - Precondiciones: El usuario ha pulsado el botón Combinar.

- Postcondiciones: El inventario pasa al estado COMBINANDO.

- Operación: desactivarBotonSalirInventario()
  - Actores: Sistema, Inventario.
  - Responsabilidades: El botón Inventario pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Combinar objetos.
  - Precondiciones: El inventario se encuentra en el estado COMBINANDO.
  - Postcondiciones: El usuario no puede salir del inventario.
- Operación: activarBotonCancelarComb()
  - Actores: Sistema, Inventario.
  - Responsabilidades: El botón Cancelar combinanción pasa a estar activo.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINANDO.
  - Postcondiciones: El usuario puede cancelar una combinación.
- Operación: desactivarBotonCombinacion()
  - Actores: Sistema, Inventario
  - Responsabilidades: El botón Combinar pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINANDO.
  - Postcondiciones: El usuario no puede volver a combinar objetos hasta que no termine la combinación actual.
- Operación: pulsarObjeto(w_objeto1)
  - Actores: Usuario, Sistema
  - Responsabilidades: El usuario selecciona el primer objeto para combinar.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINANDO.
  - Postcondiciones: El sistema selecciona el objeto.
- Operación: seleccionarObjeto(w_objeto1)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema selecciona el objeto.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado el objeto.
  - Postcondiciones: El objeto activa su lógica.
- Operación: iluminarObjeto(w_objeto1)
  - Actores: Inventario.

- Responsabilidades: El objeto cambia su textura.
- Referencias cruzadas: Caso de uso Combinar
- Precondiciones: El usuario ha seleccionado el objeto.
- Postcondiciones: El objeto se ilumina.

- Operación: incluirObjeto(w_objeto1)
  - Actores: Inventario.
  - Responsabilidades: El inventario almacena el objeto en el vector de combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado el objeto.
  - Postcondiciones: El objeto se almacena en el vector de combinación.
- Operación: pulsarObjeto(w_objeto2)
  - Actores: Usuario, Sistema
  - Responsabilidades: El usuario selecciona el segundo objeto para combinar.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINANDO.
  - Postcondiciones: El sistema selecciona el objeto.
- Operación: seleccionarObjeto(w_objeto2)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema selecciona el objeto.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado el objeto.
  - Postcondiciones: El objeto activa su lógica.
- Operación: iluminarObjeto(w_objeto2)
  - Actores: Inventario.
  - Responsabilidades: El objeto cambia su textura.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado el objeto.
  - Postcondiciones: El objeto se ilumina.
- Operación: incluirObjeto(w_objeto2)
  - Actores: Inventario.
  - Responsabilidades: El inventario almacena el objeto en el vector de combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado el objeto.
  - Postcondiciones: El objeto se almacena en el vector de combinación.

- Operación: setEstado(COMBINACIÓN_PREPARADA)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema cambia la variable estado del inventario.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha seleccionado dos objetos.
  - Postcondiciones: El inventario pasa al estado COMBINACIÓN_PREPARADA.
- Operación: activarBotonAceptarComb()
  - Actores: Sistema, Inventario.
  - Responsabilidades: El botón aceptar combinación pasa a estar activo.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINACIÓN_PREPARADA.
  - Postcondiciones: El usuario puede completar la combinación.
- Operación: pulsarBotonAceptarComb()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Aceptar combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado COMBINACIÓN_PREPARADA.
  - Postcondiciones: El botón Aceptar combinación activa su lógica.
- Operación: setEstado(NORMAL)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema cambia el valor de la variable estado del inventario.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha pulsado el botón Aceptar combinación.
  - Postcondiciones: El inventario pasa al estado NORMAL.
- Operación: activarBotonCerrarInventario()
  - Actores: Inventario.
  - Responsabilidades: El botón cerrar inventario pasa a estar activo.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El inventario se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede salir del inventario.
- Operación: activarBotonCombinacion()
  - Actores: Inventario.
  - Responsabilidades: El botón Combinación pasa a estar activo.
  - Referencias cruzadas: Caso de uso Combinar

- Precondiciones: El inventario se encuentra en el estado NORMAL.
- Postcondiciones: El usuario puede volver a combinar.

- Operación: eliminarObjeto(w_objeto1)
  - Actores: Inventario.
  - Responsabilidades: El inventario elimina el objeto utilizado en la combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha pulsado el botón Aceptar combinación.
  - Postcondiciones: Se elimina w_objeto1 del inventario.
- Operación: eliminarObjeto(w_objeto2)
  - Actores: Inventario.
  - Responsabilidades: El inventario elimina el objeto utilizado en la combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha pulsado el botón Aceptar combinación.
  - Postcondiciones: Se elimina w_objeto2 del inventario.
- Operación: añadirCombinacion(w_objeto1, w_objeto2)
  - Actores: Inventario.
  - Responsabilidades: El inventario añade el resultado de la combinación.
  - Referencias cruzadas: Caso de uso Combinar
  - Precondiciones: El usuario ha pulsado el botón Aceptar combinación.
  - Postcondiciones: Se añade el resultado de la combinación al inventario.

Caso de uso: Cancelar combinación

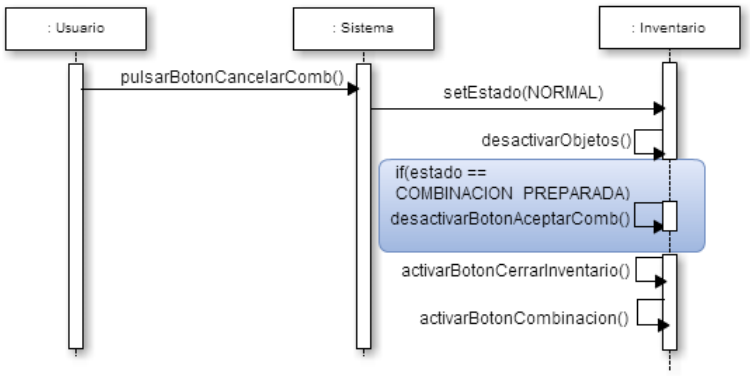


Figura 4.25: Diagrama de secuencia: Cancelar combinación

- Operación: pulsarBotonCancelarComb()
  - Actores: Usuario, Sistema.

- Responsabilidades: El usuario pulsa el botón Cancelar combinación.
- Referencias cruzadas: Caso de uso Cancelar combinación.
- Precondiciones: El usuario se encuentra en el inventario. El inventario está en el estado COMBINANDO o COMBINACIÓN_PREPARADA.
- Postcondiciones: Se activa la lógica del botón.

- Operación: setEstado(NORMAL)
  - Actores: Sistema, Inventario.
  - Responsabilidades: El sistema modifica la variable estado del inventario.
  - Referencias cruzadas: Caso de uso Cancelar combinación.
  - Precondiciones: El usuario ha pulsado del botón Cancelar combinación.
  - Postcondiciones: El inventario pasa al estado NORMAL.
- Operación: desactivarObjetos()
  - Actores: Inventario.
  - Responsabilidades: Se deseleccionan todos los objetos.
  - Referencias cruzadas: Caso de uso Cancelar combinación.
  - Precondiciones: El inventario se encuentra en el estado NORMAL.
  - Postcondiciones: Todos los objetos se deseleccionan.
- Operación: desactivarBotonAceptarComb()
  - Actores: Inventario.
  - Responsabilidades: El botón Aceptar combinación pasa a estar inactivo.
  - Referencias cruzadas: Caso de uso Cancelar combinación.
  - Precondiciones: El inventario se encontraba en el estado COMBINACIÓN_PREPARADA.
  - Postcondiciones: El botón se desactiva.
- Operación: activarBotonCerrarInventario()
  - Actores: Inventario.
  - Responsabilidades: El botón Cerrar inventario pasa a estar activo.
  - Referencias cruzadas: Caso de uso Cancelar combinación.
  - Precondiciones: El inventario se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede salir del inventario.
- Operación: activarBotonCombinacion()
  - Actores: Inventario.
  - Responsabilidades: El botón Combinacion pasa a estar activo.
  - Referencias cruzadas: Caso de uso Cancelar combinación.
  - Precondiciones: El inventario se encuentra en el estado NORMAL.
  - Postcondiciones: El usuario puede combinar objetos.

Caso de uso: Mostrar descripción

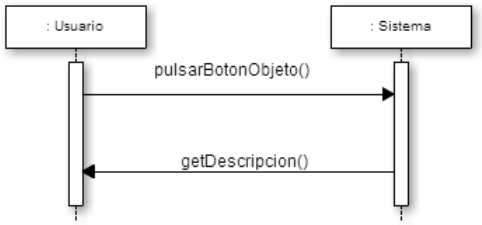


Figura 4.26: Diagrama de secuencia: Mostrar descripción

- Operación: pulsarBotonObjeto()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa sobre un objeto.
  - Referencias cruzadas: Caso de uso Mostrar descripción.
  - Precondiciones: El usuario está en la pantalla Inventario.
  - Postcondiciones: El objeto se ilumina y se activa su lógica.
- Operación: getDescripcion()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema muestra por pantalla la descripción del objeto.
  - Referencias cruzadas: Caso de uso Mostrar descripción.
  - Precondiciones: El usuario ha pulsado un objeto de su inventario.
  - Postcondiciones: El usuario visualiza la descripción del objeto.

Caso de uso: Mostrar objetivo

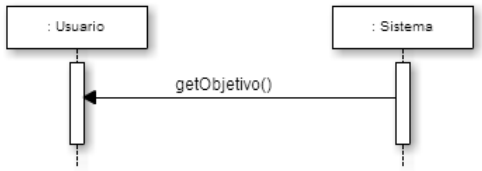


Figura 4.27: Diagrama de secuencia: Mostrar objetivo

- Operación: getObjetivo()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema muestra por pantalla el próximo objetivo del usuario.
  - Referencias cruzadas: Caso de uso Mostrar objetivo.
  - Precondiciones: El usuario está en la pantalla Inventario.
  - Postcondiciones: El usuario visualiza su próximo objetivo.

Caso de uso: Actualizar objetivo

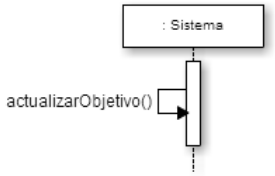


Figura 4.28: Diagrama de secuencia: Actualizar objetivo

- Operación: actualizarObjetivo()
  - Actores: Sistema.
  - Responsabilidades: El sistema cambia el objetivo antiguo por el actual.
  - Referencias cruzadas: Caso de uso Actualizar objetivo.
  - Precondiciones: El usuario ha iniciado una partida.
  - Postcondiciones: El sistema actualiza el objetivo que se muestra en la pantalla Inventario.

Caso de uso: Elegir asesino

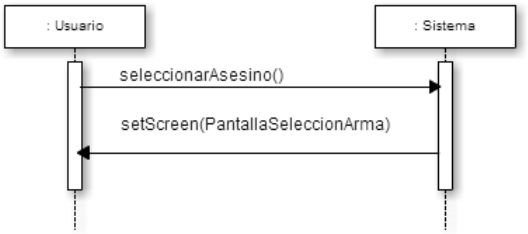


Figura 4.29: Diagrama de secuencia: Elegir asesino

- Operación: seleccionarAsesino()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario selecciona un personaje.
  - Referencias cruzadas: Caso de uso Elegir asesino.
  - Precondiciones: El usuario se encuentra en la pantalla de selección de asesino.
  - Postcondiciones: El usuario pulsa sobre un personaje.
- Operación: setScreen(PantallaSeleccionArma)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Elegir asesino.

- Precondiciones: El usuario ha elegido al personaje correcto.
- Postcondiciones: El sistema muestra la pantalla de selección de arma.

Caso de uso: Elegir arma

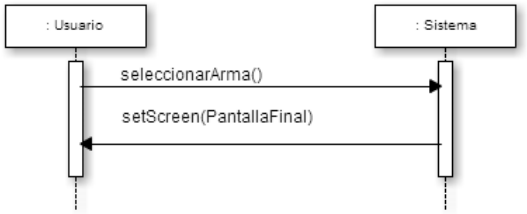


Figura 4.30: Diagrama de secuencia: Elegir arma

- Operación: seleccionarArma()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario selecciona un arma.
  - Referencias cruzadas: Caso de uso Elegir arma.
  - Precondiciones: El usuario se encuentra en la pantalla de selección de arma.
  - Postcondiciones: El usuario pulsa sobre un arma.
- Operación: setScreen(PantallaFinal)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Elegir arma.
  - Precondiciones: El usuario ha elegido el arma correcta.
  - Postcondiciones: El sistema muestra la pantalla Final.

Caso de uso: Sumar puntos

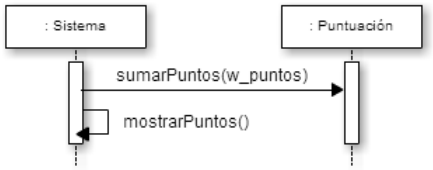


Figura 4.31: Diagrama de secuencia: Sumar puntos

- Operación: sumarPuntos(w_puntos)
  - Actores: Sistema, Puntuación.
  - Responsabilidades: El sistema suma w_puntos a la puntuación total del usuario.

- Referencias cruzadas: Caso de uso Sumar puntos.
- Precondiciones: El usuario ha iniciado una partida.
- Postcondiciones: La puntuación del usuario aumenta w_puntos.

- Operación: mostrarPuntos()
  - Actores: Sistema
  - Responsabilidades: El sistema debe mostrar los puntos actuales del usuario.
  - Referencias cruzadas: Caso de uso Sumar puntos.
  - Precondiciones: El usuario ha iniciado una partida.
  - Postcondiciones: El usuario visualiza sus puntos durante toda la partida.

Caso de uso: Actualizar fichero de log

Figura 4.32: Diagrama de secuencia: Actualizar fichero de log

- Operación: añadirLinea(w_cadena)
  - Actores: Sistema, Archivo Log.
  - Responsabilidades: El sistema añade una nueva línea al fichero.
  - Referencias cruzadas: Caso de uso Actualizar fichero de log.
  - Precondiciones: El usuario ha realizado una acción que debe ser registrada en el fichero.
  - Postcondiciones: Se añade una línea al fichero de log.

Caso de uso: Subir fichero de log

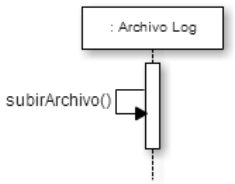


Figura 4.33: Diagrama de secuencia: Subir fichero de log

- Operación: subirArchivo()
  - Actores: Archivo Log.

- Responsabilidades: El fichero debe lanzar una orden para conectarse a Dropbox y subirse al servidor.
- Referencias cruzadas: Subir fichero de log.
- Precondiciones: El usuario ha llegado a la pantalla Final.
- Postcondiciones: El fichero se sube a Dropbox.

Caso de uso: Ver créditos

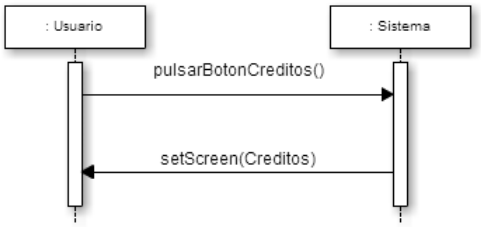


Figura 4.34: Diagrama de secuencia: Ver créditos

- Operación: pulsarBotonCreditos()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Creditos.
  - Referencias cruzadas: Caso de uso Ver créditos.
  - Precondiciones: El usuario se encuentra en la pantalla Inicio.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Creditos)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Ver créditos.
  - Precondiciones: El usuario ha pulsado el botón Créditos.
  - Postcondiciones: Se muestra la pantalla Créditos.

Caso de uso: Abrir diccionario

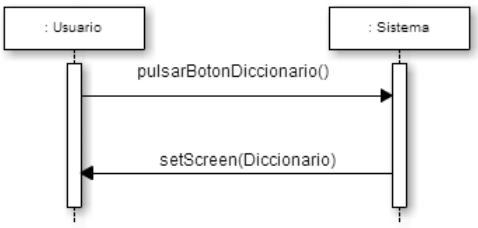


Figura 4.35: Diagrama de secuencia: Abrir diccionario

- Operación: pulsarBotonDiccionario()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa el botón Diccionario.
  - Referencias cruzadas: Caso de uso Abrir diccionario.
  - Precondiciones: El usuario se encuentra en la pantalla Pasillo.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Diccionario)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Abrir diccionario.
  - Precondiciones: El usuario ha pulsado el botón Abrir diccionario.
  - Postcondiciones: Se muestra la pantalla Diccionario.

Caso de uso: Cerrar diccionario

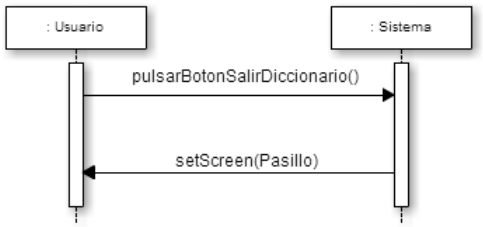


Figura 4.36: Diagrama de secuencia: Cerrar diccionario

- Operación: pulsarBotonSalirDiccionario()
  - Actores: Usuario, Sistema.
  - Responsabilidades: El usuario pulsa sobre el botón Cerrar diccionario.
  - Referencias cruzadas: Caso de uso Cerrar diccionario.
  - Precondiciones: El usuario se encuentra en la pantalla Diccionario.
  - Postcondiciones: Se activa la lógica del botón.
- Operación: setScreen(Pasillo)
  - Actores: Usuario, Sistema.
  - Responsabilidades: El sistema cambia de pantalla.
  - Referencias cruzadas: Caso de uso Cerrar diccionario.
  - Precondiciones: El usuario ha pulsado el botón Cerrar diccionario.
  - Postcondiciones: Se muestra la pantalla Pasillo.

Capítulo 5

Diseño del Sistema

En esta sección se explicarán todas pantallas que intervienen en el juego como también, los elementos gráficos que contiene cada una de ellas. También se ofrecerá una breve explicación de las herramientas utilizadas durante el desarrollo del proyecto.

Desde el primer momento se decidió usar la librería LibGDX, por esta razón, era necesario utilizar el lenguaje de programación Java. Se ha elegido esta librería por varios motivos: su facilidad de uso, la extensa documentación que existe de la misma, la facilidad de exportar el código a otros lenguajes como Android o IOS y porque ya se había trabajado anteriormente con ella, lo cual supuso una etapa de aprendizaje más corta. También se ha usado XML para almacenar los diálogos del juego debido a que Java nos proporciona métodos para trabajar cómodamente con este lenguaje.

El IDE utilizado para desarrollarlo ha sido Eclipse Luna. Se elige Eclipse porque, durante la carrera, de entre todos los IDEs utilizados, esta herramienta ha sido la más usada tanto por alumnos como por profesores. Se requiere la versión Luna porque la nueva versión de LibGDX es mucho más estable si se usa esta versión.

Como servidor de almacenamiento de los ficheros de log generados durante la partida se ha decidido usar Dropbox. Esto se debe a que Java es compatible con su API, lo cual da como resultado que sea muy sencillo subir un archivo. Así no es necesario que el usuario de permiso para que la

aplicación se conecte a la cuenta de Dropbox habilitada anteriormente para el almacenamientos de estos ficheros.

5.1. Herramientas utilizadas

5.1.1. Gráficos

Para el retoque de fotos, tanto de objetos, personajes, habitaciones y para la creación de los botones que se representan mediante palabras se ha usado la herramienta de retoque fotográfico Gimp. Se ha optado por esta herramienta porque es la mejor alternativa gratuita a Photoshop y también la más parecida.

Para crear elementos desde cero se ha optado por usar otra herramienta libre, llamada Kolour-Paint. De esta forma se han creado los botones, algunos elementos del pasillo y el personaje principal. Al no tratarse de elementos muy grandes no hubo necesidad de usar una herramienta de diseño vectorial. Se ha optado por esta herramienta debido a su sencillez y por ser la opción más parecida a Paint.

Las imágenes utilizadas tienen licencia libre para usar y modificarlas. Las imágenes se han obtenido de diversas webs como `http://www.freedigitalphotos.net`, `https://es.wikipedia.org`, `http://es.freeimages.com/` y `http://photobucket.com/`.

5.1.2. Sonido

Al no poseer conocimientos sobre composición y retoques de sonido, la música y los efectos de sonido se han tenido que descargar desde las webs `https://www.jamendo.com/es` y `https://freesound.org/`. Ambas con licencia de libre uso, excepto para fines comerciales.

5.1.3. Redacción de la memoria

Para redactar la memoria del proyecto se ha usado el editor de textos LaTeX, concretamente la web Sharelatex. Esta permite redactar textos y compartirlos entre varios usuarios, además posee un compilador que transforma el código LaTeX a formato PDF. Para diseñar los diagramas se ha usado la herramienta online Cacoo dada su facilidad de uso y que además ya estaba familiarizado con ella.

LaTeX

5.2. Interfaz gráfica

5.2.1. Pantallas Inicio, Presentación y Créditos

Figura 5.1: Pantalla de Inicio de la versión con ambiente de suspense.

Estas pantallas representan la parte inicial del juego. En la pantalla de inicio el jugador puede optar entre, empezar una partida nueva, salir del juego o ver los títulos de crédito. Cada una de estas opciones está representada por un botón.

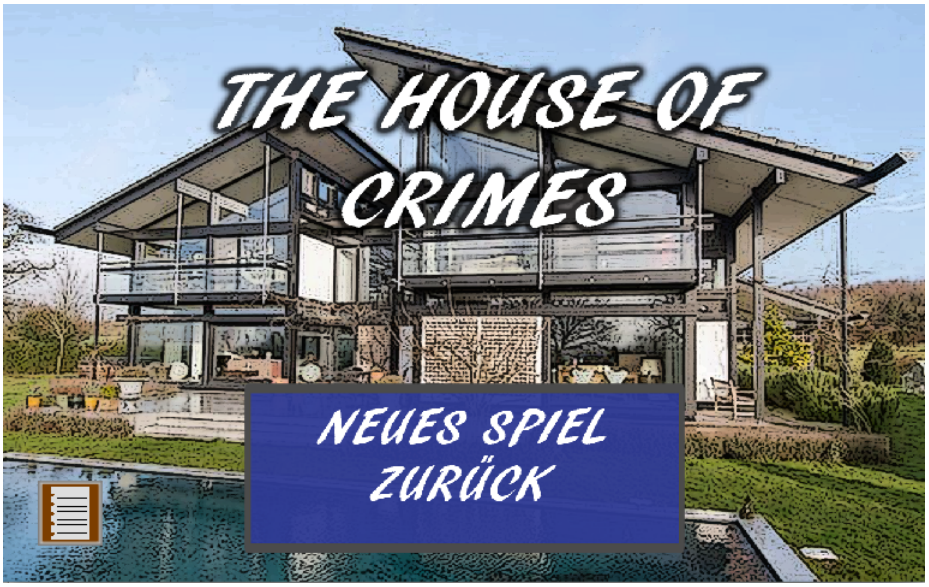


Figura 5.2: Pantalla de Inicio de la versión con ambiente neutral.

Si el jugador pulsa el botón inicio se le mostrará la pantalla de presentación. En esta pantalla se dan indicaciones sobre lo que el alumno debe de hacer durante el juego. La idea es poner al jugador en el contexto del juego. Además en esta pantalla se pide que el jugador introduzca su nombre. Si el usuario no modifica este campo nombre de usuario es Usuario 1. De esta forma se completará la información del fichero de log. Esta pantalla cuenta con un botón de empezar, que una vez pulsado este botón, comienza la partida.

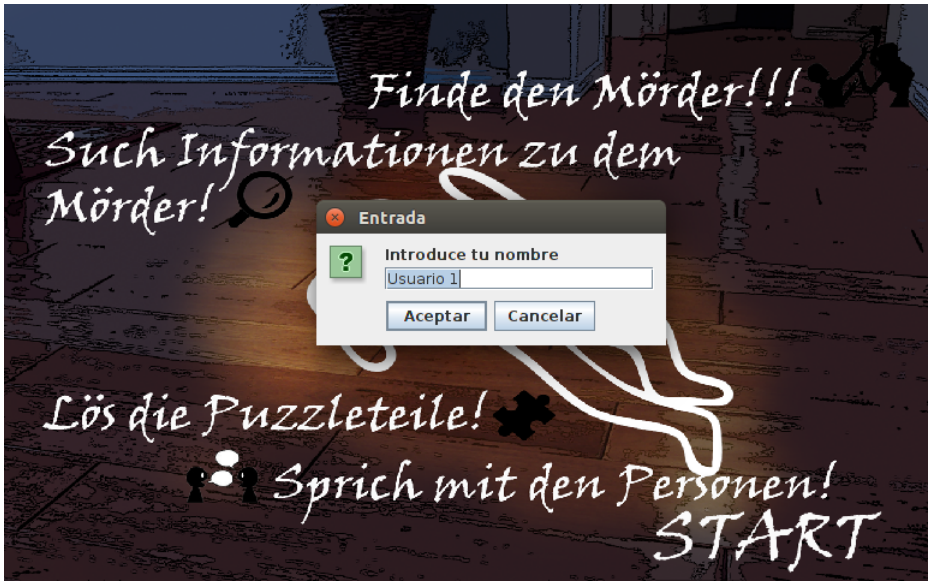


Figura 5.3: Pantalla de presentación.

En la pantalla de créditos se muestran los nombres de los participantes en el proyecto, así como

una lista de las licencias de todo el material no original. Esta pantalla cuenta con un botón para volver a la pantalla de inicio.

Figura 5.4: Pantalla de créditos.

5.2.2. Pantalla Pasillo

La pantalla pasillo es la pantalla principal del juego. El jugador la usará para ir de una habitación a otra. Utilizando los botones de dirección el jugador puede moverse por ella de arriba a abajo y de izquierda a derecha. La pantalla pasillo es la única pantalla en la que el jugador puede acceder, mediante diferentes botones al inventario y al diccionario que nos proporciona el juego. También hay un botón que permite al jugador entrar en las habitaciones, este solo se activa si el protagonista está cerca de una puerta, si no, permanece desactivado. En esta pantalla se muestran los puntos que ha conseguido el jugador durante la partida, así como el total de puntos que puede conseguir.

Figura 5.5: Pantalla Pasillo.

Figura 5.6: Pantalla Pasillo con el botón Puerta activado.

En esta pantalla el fichero de log se actualizará cada vez que el jugado realiza una de las siguientes acciones:

- **Pulsa el botón Inventario para acceder a su inventario**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber que entra en el inventario.
- **Pulsa el botón Puerta para entrar en una habitación**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber en que habitación entra.

- **Pulsa el botón Diccionario para acceder al diccionario**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber que entra en el diccionario.

5.2.3. Pantalla Habitación

Esta pantalla representa cada una de las habitaciones que hay en el juego. Hay un total de cinco habitaciones, que se dividen en, salón, dormitorio, biblioteca, ático y sótano. Todas las habitaciones tienen los mismos elementos, un botón para salir de la habitación, un botón para investigarla y un botón para hablar con el personaje, que se encuentra en la habitación. Cabe señalar que los personajes y objetos varían de una habitación a otra. En esta pantalla, al igual que en el pasillo, también aparecen los puntos del jugador sobre el máximo de puntos.

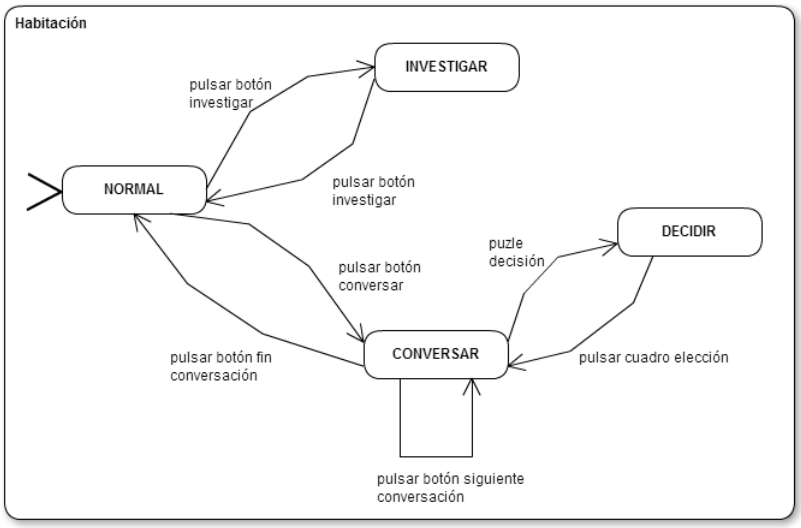


Figura 5.7: Diagrama de estados de la Pantalla Habitación.

En este diagrama de estados se puede ver el funcionamiento de la lógica de una habitación. Al entrar, la habitación se encuentra en el estado Normal. A partir de ahí podemos ir al estado Investigar, al estado conversar o salir de la habitación.

Figura 5.8: Pantalla Habitación en estado Normal con ambiente neutro y objetos con suspense.

Figura 5.9: Pantalla Habitación en estado Normal con ambiente y objetos con suspense.

Si nos encontramos en el estado Investigar solo podemos volver al estado Normal. Lo mismo ocurre en el estado Conversar. Durante la investigación podemos interactuar con los objetos que están repartidos por la habitación. Solo volvemos al estado Normal si se pulsa el botón Investigar una segunda vez.

Figura 5.10: Pantalla Habitación en estado Investigar.

En el estado Conversar podemos entablar una conversación con el personaje que se encuentra en la habitación. Solo cuando pulsamos el botón Finalizar conversación volvemos al estado Normal. En un momento dado, cuando nos encontremos en el estado conversar, puede ser que un personaje nos plantee una pregunta a la que tenemos que contestar. Si esto ocurre pasaremos al estado Decisión. En este estado tenemos la opción de elegir una respuesta de entre cuatro respuestas que se nos ofrecen. Una vez elegida una respuesta, indiferentemente si hemos acertado o no, volveremos al estado Conversar.

Figura 5.11: Pantalla Habitación en estado Conversar.

Figura 5.12: Pantalla Habitación en estado Decisión.

En esta pantalla, el fichero de log se actualiza cuando:

- **El jugador pulsa el botón Puerta para salir de la habitación**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber que sale al pasillo.
- **El jugador pulsa el botón Investigar**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, la habitación en la que se encuentra y un indicador para saber si activa o desactiva el modo investigación.
- **El jugador pulsa el botón Conversación**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, la habitación en la que se encuentra, con que personaje conversa y un indicador para saber si consigue hablar o no con el personaje.
- **El jugador comienza una misión**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, que personaje le manda el recado, la habitación en la que se encuentra y el objeto implicado en la misión.
- **El jugador finaliza una misión**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, que personaje le manda el recado, la habitación en la que se encuentra y el objeto implicado en la misión.
- **El jugador coge un objeto**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, el objeto que intenta, la habitación en la que se encuentra y un indicador que muestra si el jugador ha podido o no coger el objeto.
- **El jugador pulsa sobre una elección**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, que personaje le hace

la pregunta, la habitación en la que se encuentra, el objeto implicado en la misión y un indicador que muestra si el jugador ha acertado la pregunta o no.

- **El jugador recibe una pista**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, que personaje le da la pista, la habitación en la que se encuentra, un indicador que muestra si la pista es sobre el arma o el asesino y un pequeño resumen de la pista.

5.2.4. Pantalla Inventario

Esta pantalla representa el inventario del jugador. El inventario, al igual que las habitaciones, tiene diferentes estados, en cada uno de ellos se pueden realizar diferentes acciones. El inventario está compuesto por varios elementos: una lista de los objetos conseguidos, una sección donde se muestra la descripción de un objeto seleccionado, una sección donde se muestra una breve explicación de lo que el jugador tiene que hacer en ese momento, un botón para salir del inventario, un botón para entrar en el estado Combinando, un botón para cancelar la combinación de objetos y un botón para aceptar la combinación y conseguir un nuevo objeto, resultado de la combinación de los dos objetos seleccionados.

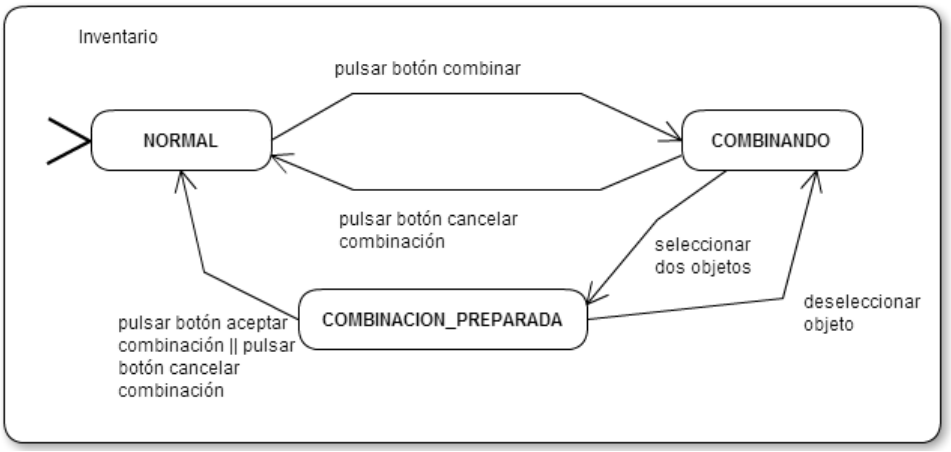


Figura 5.13: Diagrama de estados de la pantalla Inventario.

Siempre que se abre el inventario, este se encuentra en el estado Normal. Dentro de este estado podemos salir del inventario, pulsar sobre los objetos conseguidos para leer una breve descripción del objeto o, si pulsamos sobre el botón Combinar, entrar en el estado Combinando. En este último estado podemos seleccionar los objetos que hayamos recogido para combinarlos entre ellos y crear nuevos objetos. Para ello solo hay que seleccionar dos objetos que estén en el inventario. Tras esto, y si ambos objetos se pueden combinar, se pasa al estado Combinación Preparada. Desde este estado podemos aceptar la combinación, con lo que se creará un nuevo objeto y se volverá al estado Normal. También se puede cancelar la combinación. De esta forma solo se vuelve al estado Inicial o deseleccionar uno o ambos objetos y volver al estado Combinando. La información sobre la misión siempre permanece en pantalla.

Figura 5.14: Pantalla Inventario.

Figura 5.15: Pantalla Inventario en estado Combinando.

En esta pantalla el fichero de log se actualiza cuando:

- **El jugador sale del inventario**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber si sale al pasillo.
- **El jugador realiza una combinación**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos , un indicador para saber

en qué pantalla se encuentra, el primer objeto seleccionado, el segundo objeto seleccionado y el resultado de la combinación.

5.2.5. Pantalla diccionario

Esta pantalla representa al diccionario. En él están recogidas todas aquellas palabras que aparecen durante el juego y que puedan ser palabras desconocidas para los alumnos y, por tanto, difíciles de entender sin ayuda de un diccionario. Junto a estas palabras, aparecen imágenes que nos ayudan a entender su significado. El diccionario consta de un botón que permite al jugador volver, después de la consulta, al pasillo.

Figura 5.16: Pantalla Diccionario.

El fichero de log se actualiza cuando:

- **El jugador sale del diccionario**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos y un indicador para saber si sale al pasillo.

5.2.6. Pantallas Selección de Asesino, Selección de Arma y Final

Estas pantallas representan la última parte del juego. En la pantalla Selección de Asesino el jugador, con las pistas que ha obtenido a lo largo de la partida, debe elegir entre los personajes al asesino. Esta pantalla muestra a los personajes del juego. Para pasar a la siguiente pantalla, el jugador debe elegir al personaje correcto.

Figura 5.17: Pantalla Selección Asesino.

De manera similar, en la pantalla Selección de Arma, el jugador debe elegir el arma que se usó en el asesinato basándose en las pistas obtenidas. Esta pantalla muestra algunos objetos que han podido ser usados como armas para el asesinato. Una vez elegido el objeto correcto se pasa a la pantalla Final. Tanto en esta pantalla como en la anterior, se muestran los puntos del jugador.

Figura 5.18: Pantalla Selección Arma.

La pantalla Final es la última pantalla del videojuego. En ella se felicita al jugador por haber completado el juego. La pantalla Final dispone de un botón para cerrar la aplicación. Es en este momento cuando el fichero de log se sube al servidor de Dropbox.

Figura 5.19: Pantalla Final.

El fichero de log se actualiza en las pantallas Selección de Asesino y Selección de Arma cuando:

- **El jugador señala a un personaje como el asesino**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, un indicador para saber que se está eligiendo al asesino, el personaje que el jugador selecciona y al asesino.
- **El jugador señala a un objeto como el arma utilizada**. El fichero registra el nombre de usuario, la fecha y la hora de la acción, la puntuación conseguida, los puntos perdidos, un indicador para saber que se está eligiendo al arma, el objeto que el jugador selecciona y al arma usada.

Capítulo 6

Implementación

Podemos afirmar que, de todas las fases del proceso de desarrollo de un software, la fase de implementación es, normalmente, la más larga y en la que más esfuerzos hay que realizar. Es en este momento en el que hay que volcar todas las ideas y objetivos en el código fuente del programa.

A continuación se describe como se ha realizado el proceso de desarrollo y las fases en las que se ha dividido.

6.1. Etapa de desarrollo

Esta etapa se ha dividido en varias sub-etapas. En cada una de ellas se ha implementado una de las pantallas del videojuego con todas sus funcionalidades. A continuación se describen, por orden temporal, las etapas que la forman:

1. El desarrollo comenzó con la implementación de la pantalla Pasillo, porque es la pantalla principal del juego. Se empezó dibujando el fondo, que previamente fue creado usando una de las herramientas de diseño gráfico. Una vez realizado este paso, se introdujo al personaje principal y se le dotó de movimiento. En esta fase nos encontramos con un problema, el escenario no se refrescaba con la suficiente rapidez y el personaje no se movía de forma fluida. Esto ocasionaba una molestia importante al jugador. Este problema se resolvió disminuyendo el tamaño de la textura del pasillo y haciendo un zoom en la pantalla para que hubiera menos píxeles que refrescar. Una vez resuelto este problema se implementaron las colisiones con las paredes y con las puertas.

2. Seguidamente se pasó a implementar las habitaciones. Para entrar en ellas hizo falta desarrollar la lógica del botón Puerta, que se encuentra en la pantalla Pasillo. Seguidamente se decidió el diseño de cada habitación, y el personaje y los objetos que iba a haber en cada una de ellas. A continuación se implementó la lógica de los botones Investigar y se amplió la funcionalidad del botón Puerta para permitir que el jugador pudiera salir de una habitación. Además se implementó la función de coger objetos. Llegados a este punto surgió otro problema, cada vez que se pasaba de una pantalla a otra LibGDX nos obligaba a hacer una llamada al constructor de la pantalla que se quería mostrar. Esto tuvo como consecuencia que los cambios que se producían en una habitación, como coger un objeto, desaparecían una vez que el jugador hubiera salio de la habitación y volviese a entrar. Para solventar este problema fue necesario dotar al juego de "memoria". Esto se consiguió recurriendo a

los patrones de diseño, concretamente al patrón Singleton, única instancia. De esta forma cada vez que se llama al constructor de una habitación, si ya existe una instancia de esta, no se crea una nueva, sino que se devuelve la que ya existe.[Freeman et al., 2004]

3. El próximo paso fue implementar el inventario. Hace falta añadir un botón Inventario a la pantalla Pasillo para poder acceder a él. El inventario presenta el mismo problema que las habitaciones, es decir, no mantiene la información una vez que hemos salido de esta pantalla. Por ello tuvimos que volver a recurrir al patrón Singleton. Una vez que conseguimos resolver este problema se pasó a completar la implementación de la funcionalidad de recoger objetos. Hasta aquel momento, cuando se pulsaba sobre un objeto, éste desaparecía de la habitación, pero no se guardaba en el inventario. Mientras se desarrollaba esta funcionalidad descubrimos otro problema. Cuando se pulsaba sobre un objeto y se recoge, se creaba una instancia por cada frame que duraba la acción de cogerlo. Esto que suponía que en el inventario aparecían decenas de objetos del mismo tipo. Usando valores booleanos se consiguió que la acción de coger el objeto solo ocurriera durante el primer frame de la acción. Más tarde se implementó la opción de visualizar la descripción de un objeto y la combinación de dos de ellos. En esta fase volvió a ocurrir el problema anterior, el objeto se seleccionaba y se deseleccionaba en cada frame. Esto hacía que, si el número de frames, desde que se pulsaba sobre el objeto hasta que se dejara de pulsar era par, el objeto aparecía deseleccionado. Dado que volvíamos a necesitar que la acción ocurriera solo en el primer frame, volvimos a aplicar la misma solución anteriormente explicada.

4. A continuación se implementaron los diálogos de los personajes. En esta etapa se presentó un problema, los diálogos y las tareas a realizar cambiaron de una partida a otra. Por lo que se planteó un sistema de estados para organizar la información. Una partida está formada por siete estados, ampliable si la situación lo requiere. Cada estado contiene un número de puzles del mismo tipo, decisión, recoger objetos o combinación. Cada puzle posee los datos necesarios para que la lógica implementada pueda funcionar correctamente. Para cada estado se elige un puzle al azar. Y gracias a unos valores booleanos cada estado pasa por otros sub-estados que hacen variar las conversaciones según el desarrollo del puzle. Para que los diálogos funcionaran tuvimos que implementar la lógica del botón Conversar, disponible en las habitaciones, los cuadros de texto y de decisión, que se muestran cada vez que se habla con un personaje y el cuadro de objetivos, que se muestra en el inventario y contiene un resumen actualizado de la misión en curso. En esta etapa se implementaron las pistas que facilitan los diferentes personajes al jugador conforme que este vaya avanzando en el juego. La idea es que con la ayuda de las pistas y la información que va encontrando sea capaz de resolver el crimen cometido. Para evitar que el jugador pudiera descubrir un patrón hemos configurado el juego de tal manera que las pistas no aparezcan siempre en el mismo orden. Además, en el caso de que el jugador hubiera cometido muchos errores en un estado, al completar el puzle, el personaje no le dará la pista correspondiente.

5. Ya bien avanzado el desarrollo se observó que había muchas palabras que el jugador objeto (estudiante del nivel A1 de alemán) probablemente no conoce, ya que desconoce el significado. Para resolver este problema se implementó un pequeño diccionario donde se recogen todas estas palabras. En un principio se pensó implementar un sistema que permitiera al jugador consultar una palabra, durante una conversación, de modo que al pulsar sobre una palabra, señalada de otro color, se mostrase su significado. Esta idea acabó descartándose por falta de tiempo que habría requerido su desarrollo. A cambio se desarrolló la actual pantalla Diccionario, a la cual el jugador puede acceder desde el pasillo

6. Por último se implementaron las pantallas de selección de Asesino y selección de Arma. En esta etapa no surgieron problemas importantes. En estas pantallas se comprueba si las elecciones hechas por el jugador son correctas o no, usando la información generada anteriormente.

7. Esta etapa se desarrolló durante todo el proceso. Consistió en la implementación de pantallas intermedias del juego: Inicio, Presentación y Final. La implementación de estas pantallas es trivial y, a priori, no deberían surgir obstáculos durante su desarrollo. Por esta razón cada vez que surgía una dificultad para la que no se tenía una solución inmediata, y con idea de no interrumpir el desarrollo, se pasaba a implementar estas pantallas.

6.2. Gestión de diálogos

Organizar los diálogos fue una decisión complicada. No solo no podíamos almacenarlos junto con el código (si se hubiera hecho así añadir otros idiomas sería una tarea muy complicada) sino que además los diálogos variaban de una partida a otra, lo cual dificultaba aún más esta tarea.

Al final se decidió usar ficheros XML para almacenar los diálogos. Están organizados por estados y estos contienen puzles. Cada puzle contiene todos los datos necesarios para su correcto funcionamiento, entre ellos los diálogos de los personajes.

Cada vez que se inicia una partida se elige, aleatoriamente, un puzle de cada estado. Ya dentro del juego, cada estado pasa por varios sub-estados: aún no se ha empezado la misión, misión en curso y misión finalizada. Cada vez que se pasa de un sub-estado a otro se actualiza el diálogo del personaje implicado en el puzle.
A continuación se presentan extractos de uno de los ficheros de configuración XML. En él se pueden ver varios ejemplos de cada tipo de puzle:

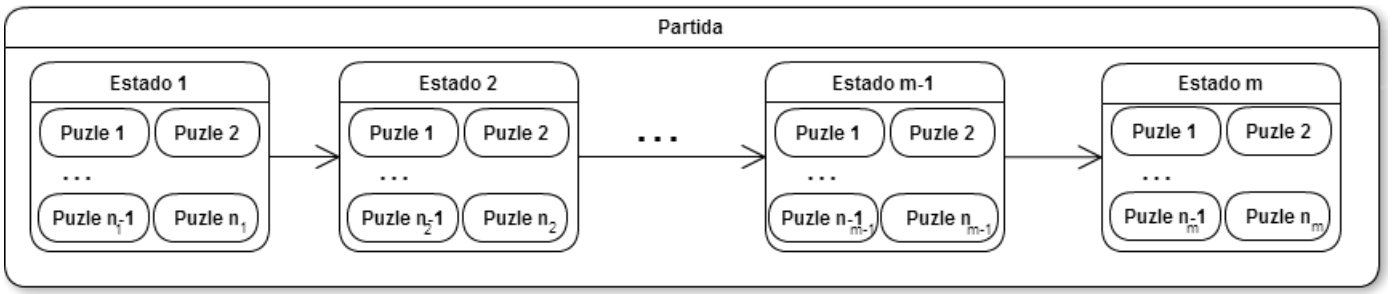
```
<?xml version="1.0" encoding="UTF-8"?>
<estados>
    <!--================================================================================-->
    <!--============================NUEVO ESTADO DECISION===============================-->
    <!--================================================================================-->
    <estado fase = "1">
        <puzzle id = "1">
            <habitacion tipoHabitacionInicio = "Atico"/>
            <dialogo texto = "Geh ins Wohnzimmer und such den Schlüssel! Komm dann zurück
            und sag mir, welche Farbe der Schlüssel hat!"/>
            <opcion1 texto = "gelb" correcto = "no"/>
            <opcion2 texto = "braun" correcto = "si"/>
            <opcion3 texto = "blau" correcto = "no"/>
            <opcion4 texto = "rot" correcto = "no"/>
            <objetivo1 texto = "Geh bitte auf den Dachboden und sprich mit dem Kind!"/>
            <objetivo2 texto = "Such die Farbe des Schlüssels! Sag dann dem Kind auf dem
            Dachboden, welche Farbe der Schlüssel hat!"/>
            <fallo texto = "Deine Antwort ist leider nicht korrekt! Versuch es noch einmal!"/>
            <prePuzzle texto = "Und, hast du die Information?"/>
            <agradecimiento texto = "Danke für die Information!"/>
            <objeto texto = "Llave"/>
        </puzzle>

        <puzzle id = "2">
```

```
        <habitacion tipoHabitacionInicio = "Atico"/>
        <dialogo texto = "Geh in die Bibliothek und such das Bonbon! Komm dann zurück
        und sag mir, wo das Bonbon ist!"/>
        <opcion1 texto = "links auf dem Tisch" correcto = "no"/>
        <opcion2 texto = "rechts über dem Tisch" correcto = "no"/>
        <opcion3 texto = "links über dem Tisch" correcto = "no"/>
        <opcion4 texto = "rechts auf dem Tisch" correcto = "si"/>
        <objetivo1 texto = "Geh bitte auf den Dachboden und sprich mit dem Kind!"/>
        <objetivo2 texto = "Such das Bonbon! Sag dann dem Kind auf dem Dachboden, wo
        das Bonbon ist!"/>
        <fallo texto = "Deine Antwort ist leider nicht korrekt! Versuch es noch einmal!"/>
        <prePuzzle texto = "Und, hast du die Information?"/>
        <agradecimiento texto = "Danke für die Information!"/>
        <objeto texto = "Caramelo"/>
    </puzzle>
</estado>

<!--=============================================================================-->
<!--=======================================NUEVO ESTADO COGER OBJETO==============-->
<!--=============================================================================-->

<estado fase = "3">
    <puzzle id = "1">
        <habitacion tipoHabitacionInicio = "Dormitorio"
        tipoHabitacionFinal = "Biblioteca"/> <!--Habitacion donde se inicia
        el puzzle y termina-->
        <objeto tipoObjeto = "Caramelo"/> <!--Objeto que participa en el puzzle-->
        <personaje tipoPersonaje = "hombre"/> <!--Personaje que inicia el puzzle-->
        <dialogo texto = "Geh bitte in die Bibliothek! Such und bring
        mir das Bonbon!"/> <!--Texto del personaje-->
        <objetivo1 texto = "Geh bitte ins Schlafzimmer und sprich
        mit dem Mann!"/> <!--Objetivo que aparece en el inventario-->
        <objetivo2 texto = "Nimm das Bonbon aus der Bibliothek! Bring dann das
        Bonbon dem Mann im Schlafzimmer!"/>
        <postPuzzle texto = "Was ist los? Zackibacki! Komm, jetzt geh
        schon in die Bibliothek!!!"/> <!--Date prisa y haz el puzzle-->
        <agradecimiento texto = "Danke für das Bonbon!"/>
    </puzzle>

    <puzzle id = "2">
        <habitacion tipoHabitacionInicio = "Sotano" tipoHabitacionFinal = "Salon"/>
        <!--Habitacion donde se inicia el puzzle y termina-->
        <objeto tipoObjeto = "Llave"/> <!--Objeto que participa en el puzzle-->
        <personaje tipoPersonaje = "hombre"/> <!--Personaje que inicia el puzzle-->
        <dialogo texto = "Geh bitte ins Wohnzimmer!
        Such und bring mir den Schlüssel!"/> <!--Texto del personaje-->
        <objetivo1 texto = "Geh bitte in den Keller und sprich
        mit dem jungen Mann!"/> <!--Objetivo que aparece en el inventario-->
        <objetivo2 texto = "Nimm den Schlüssel aus dem Wohnzimmer! Bring dann den Schlüssel
        dem jungen Mann im Keller!"/>
        <postPuzzle texto = "Was ist los? Zackibacki! Komm, jetzt geh
        schon ins Wohnzimmer!!!"/> <!--Date prisa y haz el puzzle-->
        <agradecimiento texto = "Danke für den Schlüssel!"/>
    </puzzle>
</estado>
<!--=============================================================================-->
<!--============================NUEVO ESTADO COMBINAR OBJETOS====================-->
<!--=============================================================================-->
```

```
    <estado fase = "7">
        <puzzle id = "1">
            <habitacion tipoHabitacionInicio = "Dormitorio" tipoHabitacionFinal1 = "Sotano"
            tipoHabitacionFinal2 = "Sotano"/> <!--Habitacion donde se inicia el puzzle-->
            <objeto tipoObjetoFinal = "CafeAzucar" tipoObjeto1 = "Cafe"
            tipoObjeto2 = "Azucar"/> <!--Objeto que participa en el puzzle-->
            <personaje tipoPersonaje = "hombre"/> <!--Personaje que inicia el puzzle-->
            <dialogo texto = "Such bitte den Kaffee! Aber bitte Zucker!" />
            <!--Texto del personaje-->
            <objetivo1 texto = "Geh bitte ins Schlafzimmer und sprich mit dem Mann!"/>
            <objetivo2 texto = "Such einen Kaffee mit Zucker!"/>
            <!--Objetivo que aparece en el inventario-->
            <postPuzzle texto = "Was ist los? Zackibacki! Komm, jetzt geh
            schon und such einen Kaffee mit Zucker!!!"/> <!--Date prisa y haz el puzzle-->
            <agradecimiento texto = "Danke für deine Hilfe!"/>
        </puzzle>

        <puzzle id = "2">
            <habitacion tipoHabitacionInicio = "Sotano" tipoHabitacionFinal1 = "Dormitorio"
            tipoHabitacionFinal2 = "Atico"/> <!--Habitacion donde se inicia el puzzle-->
            <objeto tipoObjetoFinal = "LibroPintado" tipoObjeto1 = "Libro"
            tipoObjeto2 = "Boligrafo"/> <!--Objeto que participa en el puzzle-->
            <personaje tipoPersonaje = "joven"/> <!--Personaje que inicia el puzzle-->
            <dialogo texto = "Such das Buch und notier bitte deinen
            Namen in dem Buch!" /> <!--Texto del personaje-->
            <objetivo2 texto = "Notier etwas in
            dem Buch!"/> <!--Objetivo que aparece en el inventario-->
            <postPuzzle texto = "Was ist los? Zackibacki! Komm, jetzt geh schon und notier
            etwas in dem Buch!!!"/> <!--Date prisa y haz el puzzle-->
            <agradecimiento texto = "Danke für deine Hilfe!"/>
        </puzzle>
    </estado>
</estados>
```

6.3. Generación de partida

Hemos afirmado en varias de ocasiones que una partida se genera aleatoriamente y que es muy difícil que un mismo alumno juegue dos partidas iguales. También hemos hablado de que una partida está formada por estados y que cada estado contiene una serie de puzles.
En esta sección se explicará la estructura de una partida y qué ocurre cuando un jugador inicia una partida nueva. Primero echemos un vistazo al siguiente diagrama:



Cada vez que el usuario pulsa el botón Empezar en la pantalla de introducción el código el

videojuego genera una nueva partida. Se crea un vector de estados vacío donde cada celda se rellenará con un tipo de estado. Cada vez que se genera un estado se elige aleatoriamente, de entre los puzles que lo forman, uno, que será el que aparezca en la partida.

De esta forma podemos generar X partidas distintas, siendo:

$$\prod_{i=1}^{m} n_i = X$$

El asesino y el arma también se eligen aleatoriamente. Contamos con cinco personajes y cuatro armas. Esto nos da como resultado veinte finales distintos.

6.4. Fichero de log

Este fichero se crea al principio de cada partida. En un principio está vacío, pero a medida que el jugador realiza acciones se va completando con lineas que, posteriormente, nos indicarán todos los pasos, que se han dado para completar el juego.
La estructura de cada línea depende de la acción que se realice. A continuación se listan todas las posibles acciones y cómo se leen:

- Transición (T): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos transita a [Pantalla a la que se transita].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];T;[Pantalla a la que se transita].

- Uso de lupa (L): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos [0 = desactiva, 1 = activa] la lupa en [Pantalla actual].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];L;[Pantalla actual];[0 o 1].

- Conversación (V): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos [0 = no consigue, 1 = consigue] hablar con [Personaje] en [Pantalla actual].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];V;[Personaje];[Pantalla actual];[0 o 1].

- Inicio de misión (I): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos recibe la misión una misión relacionada con [Objeto] del [Personaje] en [Pantalla actual].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];I;[Personaje];[Pantalla actual];[Objeto].

- Adquisición (A): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos [0 = no consigue, 1 = consigue] adquirir [Objeto] en [Pantalla actual].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];A;[Objeto];[Pantalla actual];[0 o 1].

- Combinación (B): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos combina [Objeto 1]; [Objeto 2] para crear [Objeto] en [Pantalla actual].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];B;[Pantalla actual]; [Objeto 1];[Objeto 2];[Objeto].

- Fin de misión (F): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos cumple la misión relacionada con [Objeto] encargada por [Personaje] en [Habitación].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];F;[Personaje]; [Habitación];[Objeto].

- Acertijo (J): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos [0 = no acierta, 1 = acierta] la pregunta sobre [Objeto] realizada por [Personaje] en [Habitación].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];J;[Personaje]; [Habitación];[Objeto];[0 o 1].

- Pista (P): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos es informado de que el [personaje = el asesino, arma = el arma] [Resumen pista] por parte de [Personaje] en [Habitación].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];P;[Personaje]; [Habitación];[personaje o arma];[Resumen pista].

- Hipótesis (H): El usuario [Nombre de usuario], el [Fecha y hora], con [puntos restados] puntos restados y [Puntos totales] puntos conseguidos asegura que [asesino = el asesino, arma = el arma] es [Hipótesis] siendo [Respuesta correcta]].

  [Nombre de usuario];[Fecha y hora];[puntos restados];[Puntos totales];H;[personaje o arma][Hipótesis];[Respuesta correcta];[0 o 1].

El resultado es un fichero de log parecido a este:

```
Mon Oct 12 13:13:43 CEST 2015;Suspense
mercedes paez;Mon Oct 12 13:15:24 CEST 2015;0;0;T;pasillo.
mercedes paez;Mon Oct 12 13:15:31 CEST 2015;0;0;T;Dormitorio.
mercedes paez;Mon Oct 12 13:15:32 CEST 2015;0;0;V;Hombre;Dormitorio;1
mercedes paez;Mon Oct 12 13:15:38 CEST 2015;0;0;I;Hombre;Dormitorio;Serpiente
mercedes paez;Mon Oct 12 13:15:39 CEST 2015;0;0;T;Pasillo.
mercedes paez;Mon Oct 12 13:15:50 CEST 2015;0;0;T;Biblioteca.
mercedes paez;Mon Oct 12 13:16:05 CEST 2015;0;0;T;Pasillo.
mercedes paez;Mon Oct 12 13:16:16 CEST 2015;0;0;T;Dormitorio.
mercedes paez;Mon Oct 12 13:16:16 CEST 2015;0;0;V;Hombre;Dormitorio;1
mercedes paez;Mon Oct 12 13:16:22 CEST 2015;0;0;J;Hombre;Dormitorio;Serpiente;1
mercedes paez;Mon Oct 12 13:17:09 CEST 2015;0;0;P;Hombre;Dormitorio;arma;
arma normal
mercedes paez;Mon Oct 12 13:17:09 CEST 2015;0;1000;F;Hombre;Dormitorio;
Serpiente
mercedes paez;Mon Oct 12 13:26:22 CEST 2015;0;7000;F;Hombre;Dormitorio;
```

```
SerpienteEnjaulada
mercedes paez;Mon Oct 12 13:26:24 CEST 2015;0;7000;H;asesino;JOVEN;JOVEN;1
mercedes paez;Mon Oct 12 13:26:25 CEST 2015;0;7000;H;arma;RIFLE;RIFLE;1
```

6.5. Sistema de control de versiones

Dada la dimensión y la importancia de este proyecto, se ha usado una herramienta para el control de versiones durante todo el desarrollo del mismo. Aunque se tenga una copia en local, cualquier problema con esta copia, no funcionamiento o perdida de datos, puede solventarse usando la copia almacenada en este sistema.

El sistema elegido para este proyecto ha sido Git. Se ha decido usar este por diversas razones, entre ellas, porque es uno de los más potentes que existe actualmente, porque se ha enseñado a manejarlo durante el transcurso de la carrera y porque el plugin de Eclipse de GitHub facilita las operaciones con el repositorio. El código está disponible en la siguiente dirección:

https://github.com/Gandio/ProjectRiddle

6.6. Documentación del código

A lo largo del desarrollo el código ha sido comentado internamente siempre que se ha estimado oportuno. Con esto se pretende que el código pueda ser leído por otras personas que quieran ampliar el proyecto con mayor facilidad.

Además se ha generado un Javadoc, que se integra dentro del propio proyecto en Eclipse. En él encontramos una explicación de todas las clases del juego, así como de sus atributos y métodos.

Capítulo 7

Pruebas del Sistema

En todos los sistemas software, que se desarrollan, la fase de pruebas es tan importante como las demás. Aunque hace unos años esta fase se consideraba un mero trámite que había que superar, recientemente la metodología de desarrollo ha ido evolucionando para colocar las pruebas al mismo nivel de importancia que el resto de tareas.

Cometer fallos a la hora de programar es muy fácil. En proyectos de pequeña magnitud es fácil localizar el error y corregirlo. Pero en grandes o medianos proyectos, como el que se está presentando, un error puede ser muy difícil de localizar y resolver. Pero si se realizan pruebas cada vez que una funcionalidad es implementada la posibilidad de cometer estos errores se reduce. Por esta razón es imprescindible realizar pruebas, para asegurarnos que nuestro código está limpio de errores.

7.1. Entorno de Pruebas

Dado que el proyecto ha sido desarrollado en Java el equipo, en el cual se lleven a cabo las pruebas debe disponer, como mínimo de la versión 1.7 de Java.

Además la tarjeta gráfica del equipo debe ser compatible con Open GL2.0 en adelante. En anteriores versiones de LibGDX esta compatibilidad era posible hasta con Open GL1.0, pero en la versión que hemos utilizado nosotros para este proyecto, esta opción ha sido retirada, por lo cual ahora es incompatible.

7.2. Niveles de Pruebas

En esta sección se documentan los diferentes tipos de pruebas que se han llevado a cabo.

7.2.1. Pruebas Unitarias

Las pruebas unitarias se han ido realizando durante todo el desarrollo del juego. Esto significa que, cada vez que se ha implementado una nueva funcionalidad o un nuevo módulo, se ha modificado el código para que el juego empezara en la pantalla donde se incluía la funcionalidad. De este modo este nuevo módulo se aislaba del resto del juego.

7.2.2. Pruebas de Integración

Estas pruebas se realizaron durante el desarrollo del juego. Cada vez que las pruebas unitarias de una funcionalidad o de un nuevo módulo resultaron aprobadas se volvió a modificar el código uniendo la nueva funcionalidad con el resto del juego ya implementado y probado. Tras esta modificación se volvieron a realizar las pruebas de integración anteriores y las que se correspondieron con el nuevo módulo.

7.2.3. Pruebas de Sistema

En esta actividad se realizaron las pruebas de sistema a fin de asegurar que el sistema cumpliera con todos los requisitos establecidos: funcionales y no funcionales. Este tipo de pruebas se suelen desarrollar en un entorno específico para pruebas.

Pruebas Funcionales

Las pruebas funcionales se realizaron cada vez que se terminó de implementar la lógica de los botones de una pantalla. Se probaron todos los botones para comprobar que efectivamente nos llevaran a la pantalla correspondiente o que se activara el evento correspondiente. De esta forma conseguimos comprobar que la interacción con el usuario fuera completa y que su funcionamiento se adecuara con el requisito funcional que le corresponde.

Pruebas No Funcionales

El sistema se ha probado en diferentes equipos, cada uno de ellos con diferentes sistemas operativos, Ubuntu 14.04 Ubuntu 15.04, Windows 7, Windows 8.1 y Windows 10. En todos ellos el sistema ha funcionado satisfactoriamente.

Para realizar las pruebas de usabilidad se ha contado con la ayuda de la profesora Anke Berns, que ha nos proporcionado información importante sobre el punto de vista de un usuario que normalmente no juega a videojuegos y que, por lo tanto, podría tener algunas dificultades con determinadas tareas, del juego. Una de las mejoras, que realizamos a partir del *feedback* de Anke Berns, fue diseñar una interfaz más intuitiva.

7.2.4. Pruebas de Aceptación

En cada reunión que se ha mantenido, se enseña el videojuego al resto de integrantes del proyecto. Se muestran las mejoras acordadas en reuniones anteriores y las nuevas funcionalidades. Una vez ha terminado la presentación los asistentes expresan su opinión y se apuntan los fallos de diseño que surjan, las mejoras y se habla sobre las siguientes funcionalidades a implementar.

Una vez que el juego fue aceptado por los integrantes del proyecto, se empezó a distribuir entre varios compañeros del grado de Ingeniería Informática para localizar posibles fallos o realizar mejoras, siempre bajo la aceptación de los primeros.

7.3. Análisis del feedback

Durante las últimas semanas de desarrollo la alumnas colaboradoras Mercedes Páez Piña, Andrea Calderón Márquez y Alicia Garrido Guerrero se ofrecieron voluntarias para probar el videojuego.

Su colaboración resultó crucial para descubrir errores de idioma y mejorar el diseño de la interfaz de usuario. A continuación se detallan todos los errores que se encontraron durante esta etapa.

Botones nueva partida y salir

En la versión con ambiente neutro, los botones de nueva partida y salir tienen muy poca separación entre ellos. Esto daba lugar a que las jugadoras pensaran que era un solo botón y, a veces, pulsaban sobre el botón salir creyendo que iban a iniciar una partida.
Este fallo en la interfaz de usuario se soluciona ampliando la separación entre los botones. Las jugadoras confirman que de esta forma se ve mucho más claro que son dos botones diferentes.

Textura del desván

Se descubre que siempre que se intenta entrar en el desván el juego se cierra inesperadamente. Un examen más exhaustivo revela que hay problema con la textura de esta habitación, el archivo está corrompido. Se vuelve a importar el archivo y continua fallando. De esto se deduce que no es un problema de la importación, sino del propio archivo.
Este se problema se resuelve volviendo a crear la textura del pasillo a partir de la imagen original. Todas las imágenes originales se han guardado junto a las modificadas por si surgieran fallos de este tipo.

Fallos lingüísticos

Ya que el juego está desarrollado en un idioma extranjero con el que no se está relacionado, es muy común que ocurran este tipo de fallos. Normalmente se tratan de palabras con faltas de ortografía o el uso erróneo de artículos. Pero en otras ocasiones se ha detectado errores de diálogos que dan al jugador una información que no se corresponde con la realidad del videojuego.
Las alumnas envía capturas de pantallas para saber exactamente en qué dialogo está el error y se corrige el fichero XML.

Parte III

Epílogo

Capítulo 8

Manual de usuario

A continuación se presenta un manual de usuario que pretende resolver cualquier duda que pueda tener el usuario sobre el juego o los controles del mismo.

8.1. Instalación

Para poder jugar debes tener, como mínimo la versión 1.7 de Java. Para comprobarlo, abre una consola y escribe lo siguiente:

Si aparece algo así es que todo está listo para jugar. Si no puedes descargar la última versión desde el siguiente enlace:
http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Para jugar, basta con acceder a la siguiente dirección web:

https://github.com/Gandio/ProjectRiddle/releases

En ella podrás descargar las cuatro versiones del videojuego. Una vez se hayan descargado, simplemente haz doble clic sobre el archivo para ejecutarlo.

8.2. El juego

8.2.1. Antes de jugar

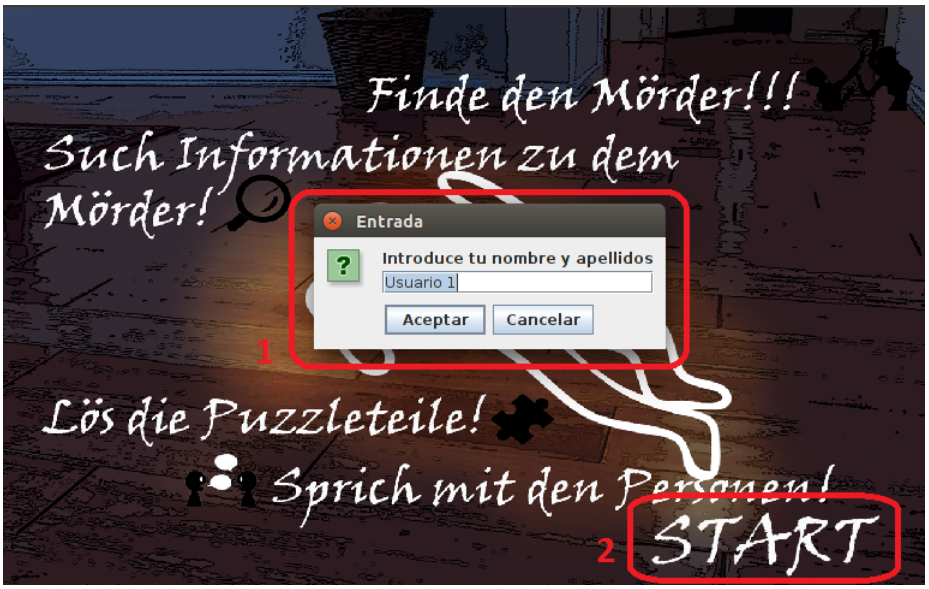
Una vez que hayas ejecutado el juego aparecerá ante ti la pantalla de título. Esta pantalla tiene dos versiones, pero los elementos son los mismos.

1. Pulsa este botón para empezar una partida nueva.
2. Pulsa este botón para salir de la partida.
3. Pulsa este botón para acceder a los títulos de créditos.

Si eliges ver los créditos del juego, aparecerá la siguiente pantalla.

1. Pulsa este botón para volver a la pantalla de título.

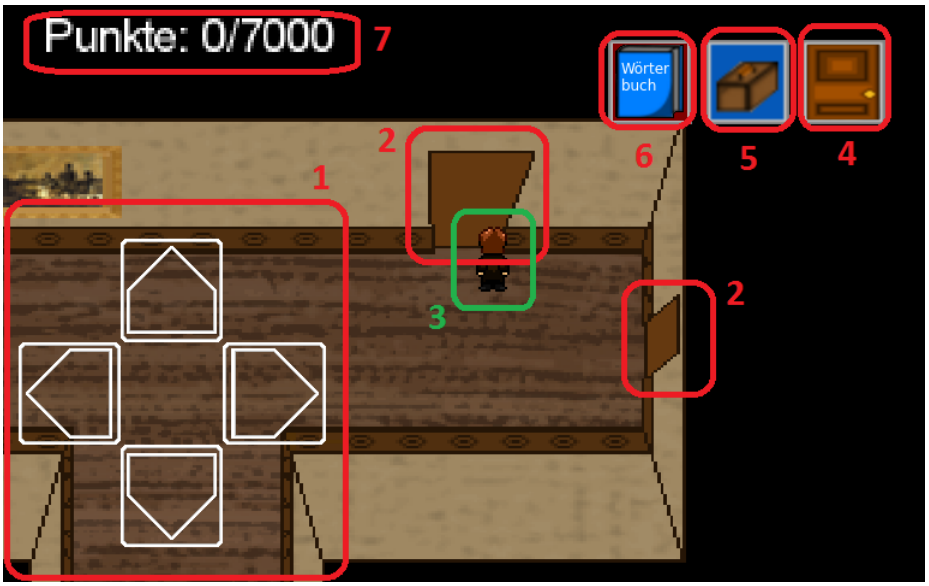
Una vez que hayas pulsado el botón para empezar una partida verás una pequeña introducción donde se te explicarán los objetivos y podrás poner nombre a tu personaje.



1. En este cuadro podrás escribir el nombre de tu personaje. Pulsa el botón Aceptar para confirmarlo.
2. Pulsa este botón para empezar la partida.

8.2.2. El pasillo

Acabas de empezar la partida. Tu personaje estará en el pasillo de una casa. Muévete por esta pantalla para ir de una habitación a otra.

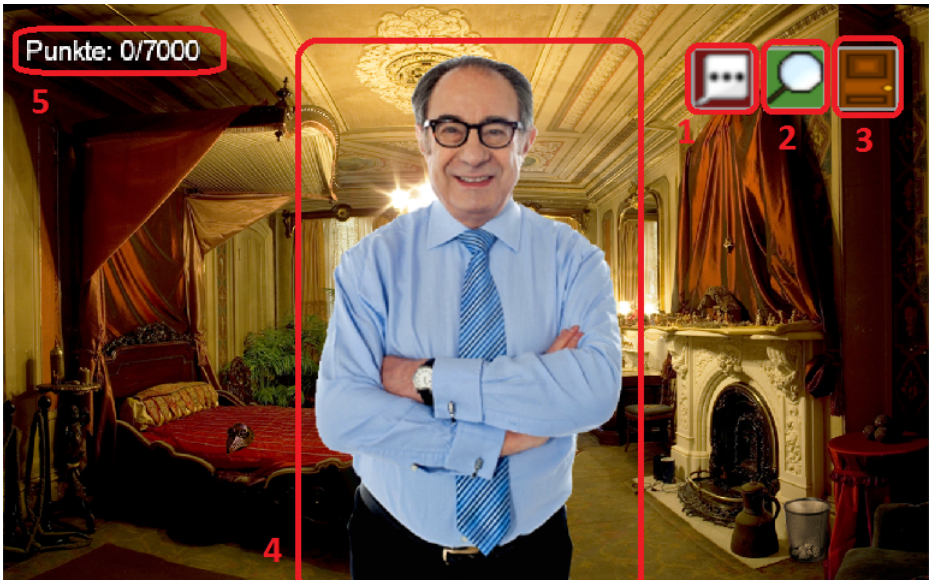


1. Pulsa estos cuatro botones para moverte por los pasillos. Puedes moverte de arriba a abajo y de izquierda a derecha.
2. Las puertas de las habitaciones, acércate una de las puertas de las habitaciones para poder entrar en ella.

3. Tu personaje, pulsando sobre los botones de dirección puede moverte con tu personaje por el pasillo.
4. Pulsa este botón cuando estés cerca de una puerta para entrar en una habitación. Si estás muy lejos de una puerta el botón no aparecerá coloreado y no podrás usarlo.
5. Pulsa este botón para abrir tu maleta y entrar en el inventario.
6. Pulsa este botón para echar un vistazo a tu diccionario.
7. Muestra los puntos que has conseguido durante tu partida.

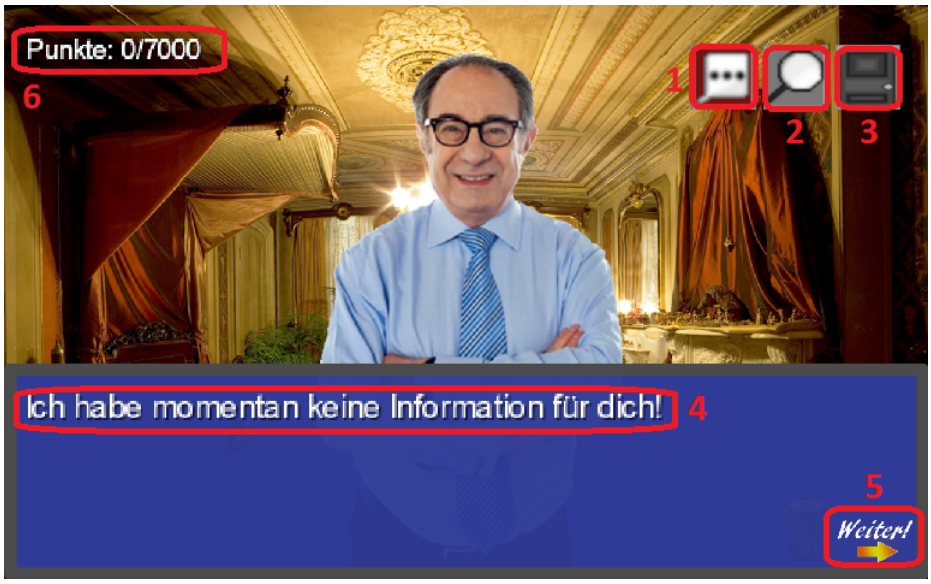
8.2.3. Las habitaciones

Cuando entres en una habitación aparecerá una nueva pantalla y podrás realizar otras acciones.



1. Este botón sirve para empezar una conversación con el personaje.
2. Este botón sirve para investigar la habitación. Durante la investigación el personaje se echará a un lado para que puedas ver la habitación al completo y recoger algunos de los objetos que se encuentran en ella.
3. Este botón sirve para salir de la habitación.
4. El personaje que se encuentra en la habitación. Puede que durante la partida te dé alguna pista sobre el asesinato, pero antes tendrás que realizar una tarea para él.
5. Muestra los puntos que has conseguido durante tu partida.

Si eliges conversar con el personaje la pantalla cambiará y aparecerán nuevos botones. Otros, en cambio, quedarán desdibujados hasta que no termines la conversación.

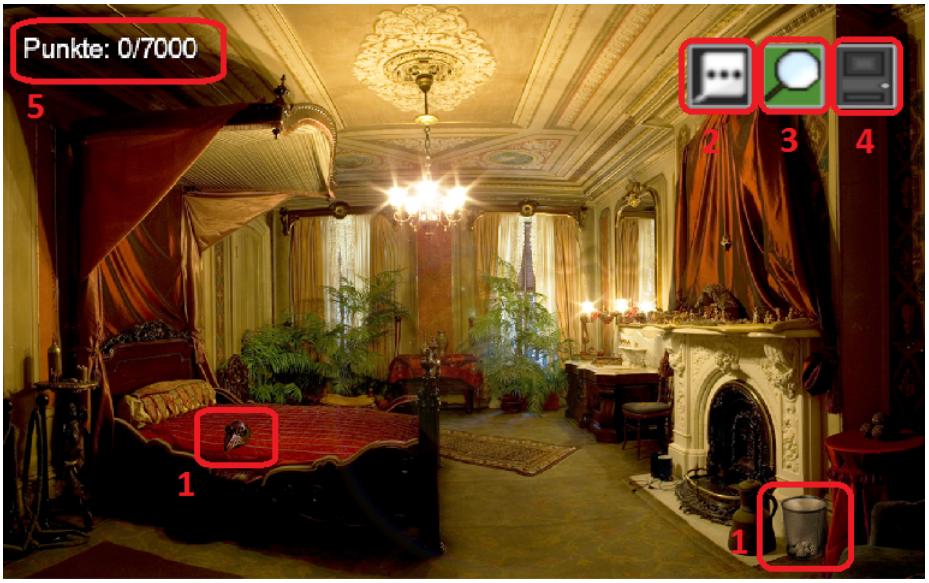


1. El botón para conversar permanece coloreado para indicarte que se está conversando, pero no podrás usarlo hasta que no termines la conversación actual.
2. Mientras estés conversando no podrás investigar la habitación.
3. Mientras estés conversando no podrás salir de la habitación.
4. Aquí se muestra lo que te ha dicho el personaje.
5. Pulsa este botón para terminar una conversación.
6. Muestra los puntos que has conseguido durante tu partida.

Durante una conversación, un personaje puede hacerte una pregunta y tendrás que elegir la respuesta correcta entre cuatro posible opciones que se te ofrecen. Pulsa sobre una opción para seleccionar aquella que crees correcta.

1. Posibles respuestas. Pulsa sobre una de ellas para seleccionarla.
2. El botón conversación permanece inactivo hasta que no respondas a la pregunta.
3. Mientras estés respondiendo a una pregunta no podrás investigar la habitación.
4. Mientras estés respondiendo a una pregunta no podrás salir de la habitación.
5. Muestra los puntos que has conseguido durante tu partida.

Si eliges investigar una habitación podrás recoger algunos objetos. Solo podrás coger un objeto si algún personaje te lo ha encargado antes.



1. Objetos con los que se puede interactuar.
2. El botón para conversar está desdibujado porque no puedes hablar mientras estás investigando la habitación.
3. El botón para investigar sigue activo. Si lo vuelves a pulsar terminarás la investigación.
4. Hasta que no termines la investigación no podrás abandonar la habitación.
5. Muestra los puntos que has conseguido durante tu partida.

8.2.4. Tu maleta

Cuando abras el inventario verás una pantalla donde se muestran los objetos que has recogido y el próximo paso de tu misión.

1. Con este botón podrás salir del inventario.
2. Pulsa este botón para confirmar la combinación de dos objetos. Hasta que no elijas una combinación válida no se iluminará.
3. Este botón cancela la combinación de objetos. No puedes usarlo hasta que no pulses el botón que permite combinar objetos.
4. Este botón permite combinar dos objetos.
5. Objetos recogidos durante los encargos.
6. Descripción del objeto. Para que aparezca la descripción hay que pulsar sobre el objeto.
7. Descripción del objetivo actual del encargo. Léelo, si no sabes como continuar, puede ser de mucha ayuda.

En algún momento de la partida, un personaje te pedirá un objeto que no se encuentra en ninguna de las habitaciones. Pero podrás crearlo a partir de otros dos objetos que encuentres en la casa. Cuando hayas conseguido los dos objetos necesarios pulsa el botón para combinar los objetos.

1. Durante la combinación no podrás salir del inventario. Deberás realizar una combinación con éxito o bien cancelarla.
2. Pulsa este botón para confirmar la combinación de dos objetos. Hasta que no elijas una combinación válida no se iluminará.
3. Este botón cancela la combinación de objetos.
4. Este botón indica que estás combinando objetos.
5. Objetos recogidos durante los encargos. Si la opción de combinar está activada pulsa sobre ellos para seleccionarlos como materiales para la combinación, si se pueden combinar el botón 2 se iluminará y podrás usarlo.
6. Descripción del objetivo actual del encargo. Léelo si no sabes como continuar, puede ser de mucha ayuda.

8.2.5. El diccionario

En cualquier momento, durante la partida, podrás acceder a tu diccionario. En él aparecerán palabras con una dificultad más alta que la media acompañadas por una imagen que la representa.

1. Pulsa este botón para cerrar el diccionario y volver al pasillo.

8.2.6. Resolviendo el caso

Una vez que hayas conseguido todas las pistas, tendrás todo lo necesario para descubrir al asesino y el arma homicida. Primero tendrás que desenmascarar al asesino.

1. Posibles asesinos, pulsa sobre quien creas que es el culpable, basándote en las pistas que has conseguido.

2. Muestra los puntos que has conseguido durante tu partida. Si te equivocas en este punto perderás la mitad.

Cuando hayas descubierto al asesino, tendrás que hacer lo mismo con el arma.

1. Posibles armas usadas en el asesinato, pulsa sobre la que creas que se usó para cometer el crimen, basándote en las pistas que has conseguido.

2. Muestra los puntos que has conseguido durante tu partida. Si te equivocas en este punto perderás la mitad.

Una vez hayas resuelto el caso, la partida terminará, no sin antes felicitarte por tu trabajo.

1. Pulsa este botón para terminar la partida.

Capítulo 9

Manual de ampliación

9.1. Introducción

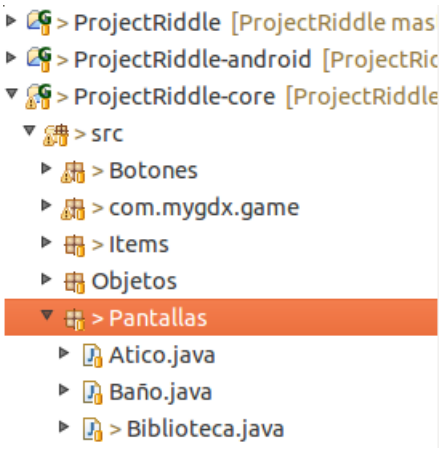
Este proyecto ha sido un primer intento de realizar un experimento para probar que, efectivamente, el nivel de aprendizaje de un jugador aumenta en situaciones de suspense. En consecuencia se ha creado un mundo pequeño, pero esto no impide que se pueda ampliar en un futuro. Entendemos por ampliar, aumentar el número de objetos, habitaciones y personajes. Indirectamente también se ampliará el número de puzles y el número de partidas distintas que se pueden jugar.

Esta sección se ha desarrollado con objeto de ayudar a los futuros desarrolladores a ampliar el mundo. A continuación se dan las instrucciones necesarias para realizar con éxito cada una de las ampliaciones.

9.2. Ampliación de habitaciones

En esta sección se explica cómo crear una nueva habitación. Para ello primero tenemos que modificar la textura del pasillo, usando alguna herramienta de edición de imágenes y añadir una nueva puerta como la que se muestra a continuación:

A continuación creamos una nueva clase que herede de la clase padre Habitación. Esta clase se debe de crear en el paquete Pantallas.

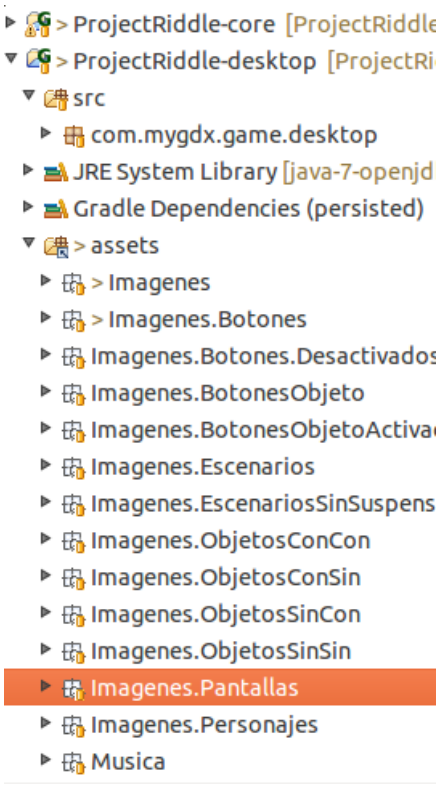


Por ejemplo, si queremos crear una cocina, la estructura de la clase será la siguiente:

```
public final class Cocina extends Habitacion {

    private static MyGdxGame game;

    //AQUI INICIALIZAMOS LOS OBJETOS

    private static Cocina unicaInstancia;

    private Cocina(MyGdxGame game, Cursor c) {
        super(Inicio.game, c);
        objetos = new Array<Objeto>();

        Iterator<Objeto> iter = objetos.iterator();

        if(TheHouseOfCrimes.SUSPENSE_OBJETOS){

            //INCLUIMOS LOS OBJETOS DE LA VERSION DE
            //SUSPENSE CREADOS ANTERIORMENTE EN EL ARRAY

            //INICIALIZAMOS LAS COORDENADAS DE LOS OBJETOS DE SUSPENSE
        }else{

            //INCLUIMOS LOS OBJETOS DE LA VERSION
            //NEUTRA CREADOS ANTERIORMENTE EN EL ARRAY

            //INICIALIZAMOS LAS COORDENADAS DE LOS OBJETOS NEUTROS

        }
        while(iter.hasNext()){
            iter.next().setTouchable(Touchable.enabled);
        }

        iter = objetos.iterator();

        while(iter.hasNext()){
            stage.addActor(iter.next());
```

```
        }

        //INICIALIZAMOS AL PERSONAJE
        personaje.setCoordenadas(0, 0);

        stage.addActor(personaje);

        stage.addActor(botonConversacion);
        stage.addActor(botonInvestigar);
        stage.addActor(botonPuerta);
    }

    public void render(float delta) {
        super.render(delta);

        Iterator<Objeto> iterObjetos = objetos.iterator();
        while(iterObjetos.hasNext()){
            iterObjetos.next().seSelecciona();
        }

        Gdx.input.setInputProcessor(stage);
        stage.draw();
    }

    public void show() {
        if(TheHouseOfCrimes.SUSPENSE_AMBIENTE)
            pantalla = new Texture(Gdx.files.internal(URL_IMAGEN_SUSPENSE));
        else
            pantalla = new Texture(Gdx.files.internal(URL_IMAGEN_NEUTRA));
    }

    public void pause() {}

    public void resume() {}

    public static Cocina getInstancia(){
        if(unicaInstancia == null){
            unicaInstancia = new Cocina(game, c);
        }

        return unicaInstancia;
    }
}
```

Ya que los objetos Habitación deben ser los mismos durante toda la partida tenemos que usar el patrón de diseño Singleton. Por esa razón el constructor debe ser privado. Las texturas de las habitaciones hay que importarlas en la carpeta *assets* del proyecto, concretamente en la carpeta *Pantallas*



Una vez que hemos creado la nueva habitación, hay que modificar la clase Pasillo.

```
public class Pasillo implements Screen{
    //........

    private boolean debug = false;

    //.........

    private Array<Rectangle> colisionesPuertas = new Array<Rectangle>();

    //...........

    //Habitaciones
    public static Sotano sotano = Sotano.getInstancia();
    public static Dormitorio dormitorio = Dormitorio.getInstancia();
    public static Habitacion salon = Salon.getInstancia();
    public static Biblioteca biblioteca = Biblioteca.getInstancia();
    public static Atico atico = Atico.getInstancia();

    //AQUI SE INICIALIZAN LAS HABITACIONES

    //.......


    public Pasillo(TheHouseOfCrimes game) {

        //...................

        colisionesPuertas.add(new Rectangle(546, 385, 75, 75)); //primera superior
```

```
        colisionesPuertas.add(new Rectangle(1060, 385, 65, 65)); //segunda superior
        colisionesPuertas.add(new Rectangle(1230, 290, 65, 65)); //derecha
        colisionesPuertas.add(new Rectangle(255, 200, 75, 75)); //inferior
        colisionesPuertas.add(new Rectangle(-20, 490, 75, 65)); //izquierda

        //SE CREAN LOS RECTANGULOS

        //............
    }
}
```

Como no podemos usar el constructor de las habitaciones tenemos que crearlas de esta forma. Los rectángulos se usan para las colisiones. Cuando nuestro personaje colisione con uno de ellos significa que está lo suficientemente cerca de una puerta como para abrirla. Para que se muestren los rectángulos hay que poner la variable *debug* a *true*. De esta forma podemos ver si los rectángulos están bien situados.
Por último tenemos que modificar la clase *BotonPuertaPasillo* para que, cuando pulsemos el botón, entremos en la nueva habitación.

```
public class BotonPuertaPasillo extends Boton{
    private Texture botonActivado, botonDesactivado;
    private Sound sonido;
    private Cursor cursor = Pasillo.getCursor();
    /*
     * Cada puerta tiene un entero asociado por el cual podemos
     *identificar por que habitacion
     * entramos. Los valores son los siguientes.
     *
     * 0 ------- Salon
     * 1 ------- Dormitorio
     * 2 ------- Atico
     * 3 ------- Biblioteca
     * 4 ------- Sotano
     */
    private int numPuerta = -1;


    //.........

    private void cambiarHabitacion(int numPuerta){
        if(numPuerta == 0){ //es el salon
            game.setScreen(Salon.getInstancia());

            TheHouseOfCrimes.getArchivoLog().escribirLinea(
            new LineaLog(TheHouseOfCrimes.getUsuario() + ";" +
                    TheHouseOfCrimes.getFecha() + ";" +
                    Puntuacion.getError() * (-100) + ";" +
                    Puntuacion.getPuntos() + ";" +  "T" + ";" + "Salon."));

        }else if(numPuerta == 1){ //es el dormitorio
            game.setScreen(Dormitorio.getInstancia());

            TheHouseOfCrimes.getArchivoLog().escribirLinea(
            new LineaLog(TheHouseOfCrimes.getUsuario() + ";" +
                    TheHouseOfCrimes.getFecha() + ";" +
                    Puntuacion.getError() * (-100) + ";" +
```

```
                    Puntuacion.getPuntos() + ";" +  "T" + ";" + "Dormitorio."));
        }else if(numPuerta == 2){// El atico
                game.setScreen(Atico.getInstancia());

                TheHouseOfCrimes.getArchivoLog().escribirLinea(
                new LineaLog(TheHouseOfCrimes.getUsuario() + ";" +
                        TheHouseOfCrimes.getFecha() + ";" +
                        Puntuacion.getError() * (-100) + ";" +
                        Puntuacion.getPuntos() + ";" +  "T" + ";" + "Atico."));

        }else if(numPuerta == 3){// La biblioteca
                game.setScreen(Biblioteca.getInstancia());

                TheHouseOfCrimes.getArchivoLog().escribirLinea(
                new LineaLog(TheHouseOfCrimes.getUsuario() + ";" +
                        TheHouseOfCrimes.getFecha() + ";" +
                        Puntuacion.getError() * (-100) + ";" +
                        Puntuacion.getPuntos() + ";" +  "T" + ";" + "Biblioteca."));

        }else if(numPuerta == 4){// El sotano
                game.setScreen(Sotano.getInstancia());

                TheHouseOfCrimes.getArchivoLog().escribirLinea(
                new LineaLog(TheHouseOfCrimes.getUsuario() + ";" +
                        TheHouseOfCrimes.getFecha() + ";" +
                        Puntuacion.getError() * (-100) + ";" +
                        Puntuacion.getPuntos() + ";" +  "T" + ";" + "Sotano."));
        }

        //INCLUIMOS LA NUEVA HABITACION
    }
}
```

Seguimos añadiendo condicionales con la misma estructura. Identificamos cada habitación por la variable *numPuerta*, esta representa el índice donde se ha almacenado el rectángulo de colisión con la habitación en la clase Pasillo. Por lo tanto hay que tener mucho cuidado con el orden, si no queremos acabar entrando en una habitación que no corresponde. Además tenemos que registrar la entrada en la habitación en el fichero de log. Para ello basta con cambiar la última parte de *LineaLog* con el nombre de la habitación.

9.3. Ampliación de objetos

En esta sección se explica cómo crear un nuevo objeto y colocarlo en una habitación ya creada.

Primero veremos la clase padre, Objeto. Se hablará solo de las variables que intervienen cuando se crea un objeto nuevo y se sitúa en una habitación.

```
public abstract class Objeto extends Actor{
    private TheHouseOfCrimes game = Pasillo.game;
    protected Texture textura, botonObjeto, botonObjetoActivado, texturaActualBoton;
    protected Vector2 coordenadas;
    protected Array<Identificador> combinables;
    private boolean sePuedeCoger, investigando, seleccionado;
    private boolean tocadoUnaVez = false, control1 = false, control2 = false;
```

```
protected Identificador identificador;

protected XmlReader reader = new XmlReader();
protected Element raiz;
protected Array<Element> objetos;
protected String descripcionObjeto;

private Sound sonido, error;

public Objeto(TheHouseOfCrimes game){
    sePuedeCoger = false;
    investigando = false;
    seleccionado = false;

    sonido = Gdx.audio.newSound(Gdx.files.internal("Sonido/cogerObjeto.wav"));
    error = Gdx.audio.newSound(Gdx.files.internal("Sonido/Error.wav"));

    try{
        raiz = reader.parse(Gdx.files.internal("xml/objetosAleman.xml"));
    }catch(IOException e){}

    objetos = raiz.getChildrenByName("objeto");
}

public void setCoordenadas(float x, float y){
    coordenadas.x = x;
    coordenadas.y = y;
}

public void draw(Batch batch, float parentAlpha) {
    game.getClass();
    if(game.getScreen().getClass() == Inventario.class){
        batch.draw(texturaActualBoton, coordenadas.x, coordenadas.y);
    }else{
        batch.draw(textura, coordenadas.x, coordenadas.y);
    }
}
```

Un objeto está compuesto por cuatro tipos diferentes de texturas. *Textura*, es la imagen que adopta un objeto cuando se encuentra en una habitación, *botonObjeto* es la textura que adquiere el objeto cuando está en el inventario. *BotonObjetoActivado* es la textura que tiene el botón objeto cada vez que se pulsa. Estas dos texturas tienen una particularidad, el fichero usado por ambas debe ser obligatoriamente de 100x100 píxeles. Si son más pequeños o más grandes otros objetos que haya en el inventario pueden adquirir una posición errónea, llegando a situarse encima de otros elementos e impidiendo o dificultando su uso. Por último *texturaActualBoton* es la textura que tiene en ese momento el objeto.

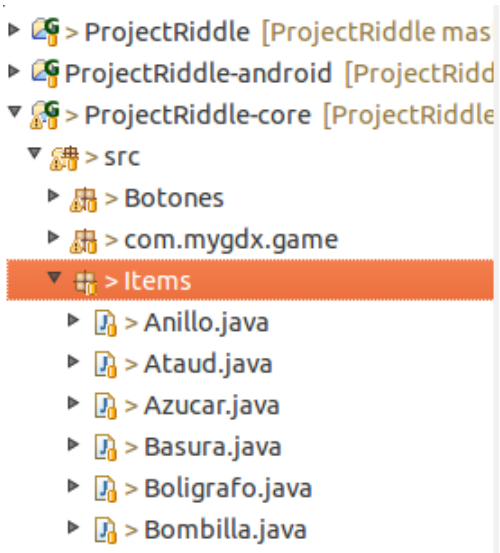
La variable *coordenadas* representa las coordenadas donde se localiza el objeto dentro de la habitación. Esta variable se tiene que inicializar en el constructor del nuevo objeto que se quiere crear y se le da valores dentro del código de la habitación donde se quiera colocar el objeto.
En el vector *combinables* se almacenan los identificadores de los objetos con los que el nuevo objeto se puede combinar. Si no tiene ninguno se inicializa a *null* cuando se crea el nuevo objeto. Cada objeto tiene su propio identificador, el nombre del objeto. Cada vez que se cree un objeto hay que añadir un nuevo valor al enumerado *Identificador*

```
package Items;

public enum Identificador {
    Anillo, Azucar, Boligrafo, Bombilla, Botella, Cafe, CafeAzucar, Caramelo, Espejo,
    Joya, Libro, LibroPintado, Daga, Bala, Llave, Moneda, Mascara, Pistola, Reloj, Veneno,
    Zapato, Jaula, Basura, Rifle, Tabaco, Ataud, Navaja, Cuadro, Calavera, Serpiente,
    SerpienteEnjaulada;
}
```

La variable *descripcionObjeto* se usa para almacenar la descripción del objeto que se coge del fichero XML *objetos[idioma]*. Este fichero está formado por varias entradas con el siguiente formato. Con añadir una nueva entrada es suficiente. Es muy importante que el campo nombre sea igual que el identificador que se ha creado anteriormente.

```
    <objeto nombre = "Cafe">
        <descripcion texto = "Oh, ich liebe Kaffee!"/>
    </objeto>

    <objeto nombre = "Calavera">
        <descripcion texto = "Oh, ich glaube der Totenkopf ist authentisch."/>
    </objeto>
```

Ahora pasaremos a crear un nuevo objeto. Lo crearemos en el paquete *Items*, donde se almacenan todos los objetos. Crearemos una nueva clase cuyo nombre será el objeto que queramos crear y será hija de la clase *Objeto*.

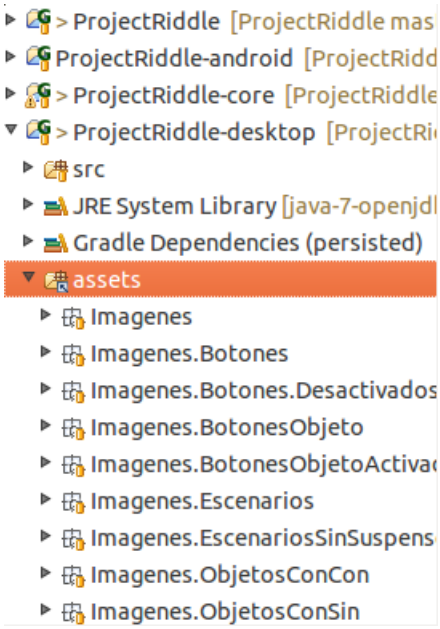


Para este manual vamos a crear una clase Botella. Así es como quedaría el código una vez inicializados todos los atributos.

```
public class Botella extends Objeto{
```

```
    /**
     * Constructor de la clase Azucar
     * @param game
     */

    public Botella(TheHouseOfCrimes game) {
        super(game);
        botonObjeto = new Texture(Gdx.files.internal(URL_IMAGEN_BOTON));
        botonObjetoActivado = new Texture(Gdx.files.internal(
        URL_IMAGEN_BOTON_ACTIVADO));
        texturaActualBoton = botonObjeto;

        /*Si no es combinable se escribe
        combinables = null;
        si lo es se escribe
        combinables = new Array<Identificador>();
        y se incluyen los identificadores de los objetos con los que se puede combinar
        combinables.add(Identificador.NOMBRE_IDENTIFICADOR);
        */

        identificador = Identificador.Azucar;

        if(TheHouseOfCrimes.SUSPENSE_AMBIENTE)
            textura = new Texture(Gdx.files.internal(URL_IMAGEN_SUSPENSE));
        else
            textura = new Texture(Gdx.files.internal(NEUTRO));

        coordenadas = new Vector2(Tools.centrarAncho(game, textura),
        Tools.centrarAlto(game, textura));


        //Descripcion del objeto
        for (Element child : objetos){
            if(identificador.name().equals(child.getAttribute("nombre")))
                descripcionObjeto = child.getChildByName("descripcion").getAttribute("texto");
        }
    }
}
```

Las imágenes se deben importar en la carpeta *assets* del proyecto.



Una vez hecho esto ya tenemos creado nuestro objeto, después solo queda colocarlo en la habitación que queramos. Para ello tenemos que crear un nuevo objeto, añadirlo al array de objetos de la habitación y especificar las coordenadas apropiadas para el objeto. Hay que tener en cuenta para qué versión queremos el objeto para luego escribir en una parte o en otra del condicional. Si un mismo objeto no se adapta de la misma forma en una versión u otra basta con utilizar el condicional interior.

```
public final class Biblioteca extends Habitacion {

    private static TheHouseOfCrimes game;
    private static Biblioteca unicaInstancia;

    private static Objeto serpiente = new Serpiente(game);
    private static Objeto calavera = new Calavera(game);

    private static Objeto caramelo = new Caramelo(game);
    private static Objeto moneda = new Moneda(game);

    //AQUI CREAMOS MAS OBJETOS
    private Biblioteca(TheHouseOfCrimes game, Cursor c) {
        super(Pasillo.game, c);
        objetos = new Array<Objeto>();

        //Objetos
        Iterator<Objeto> iter = objetos.iterator();


        //AQUI INCLUIMOS OBJETOS EN EL VECTOR DE OBJETOS
        if(TheHouseOfCrimes.SUSPENSE_OBJETOS){
            objetos.add(calavera);
            objetos.add(serpiente);
```

```
            //AQUI ESPECIFICAMOS LAS COORDENADAS
            calavera.setCoordenadas(530, 220);
            if(TheHouseOfCrimes.SUSPENSE_AMBIENTE)
                serpiente.setCoordenadas(40, 10);
            else
                serpiente.setCoordenadas(60, 10);
        }else{
            //TAMBIEN PODEMOS INCLUIR OBJETOS AQUI
            objetos.add(caramelo);
            objetos.add(moneda);

            //ESPECIFICAMOS COORDENADAS AQUI
            caramelo.setCoordenadas(510, 210);
            moneda.setCoordenadas(40, 10);
        }


        //........

    }
    //.............
}
```

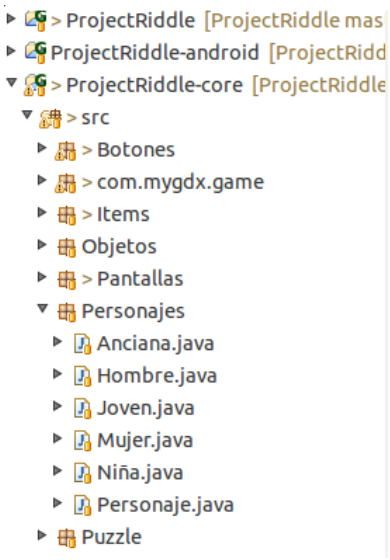
9.4. Ampliación de personajes

En esta sección se explica cómo crear un nuevo personaje y colocarlo en una habitación ya creada. Veamos primero la clase padre. Esta clase está formada por la imagen del personaje y por las coordenadas donde se sitúa en la habitación. Como podemos ver, Personaje es una clase abstracta por lo cual no podemos crear instancias de ella.

```
public abstract class Personaje extends Actor{
    protected Texture personaje;
    protected Vector2 coordenadas;

    /**
     * Constructor de la clase personaje
     * @param game
     */
    public Personaje(TheHouseOfCrimes game){}

    /**
     * Dibuja al actor en el stage
     */
    public void draw(Batch batch, float parentAlpha) {
        batch.draw(personaje, coordenadas.x, coordenadas.y);
    }

    /**
     * Este metodo se usa para modificar las coordenadas del personaje
     * @param x
     * @param y
     */
    public void setCoordenadas(float x, float y){
```

```
        coordenadas.x = x;
        coordenadas.y = y;
    }
}
```

Para crear un nuevo personaje hay que crear una nueva clase que herede de esta. La añadiremos en el paquete *personajes*.

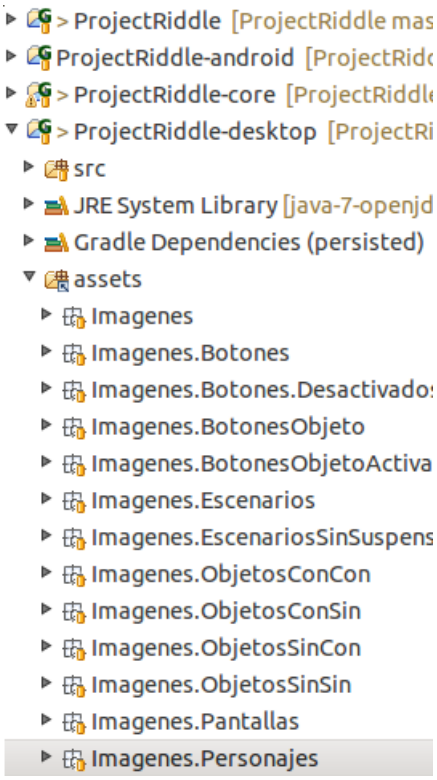


Solo habrá un objeto de este nuevo personaje durante toda la partida, por lo cual usaremos el patrón Singleton. Basta con declarar un atributo de esta misma clase llamado *unicaInstancia*, hacer que el constructor sea privado y añadir el método *getInstancia*.

```
public class Mujer extends Personaje{

    private static TheHouseOfCrimes game;
    private static Mujer unicaInstancia;

    private Mujer(TheHouseOfCrimes game) {
        super(game);
        Mujer.game = game;

        if(TheHouseOfCrimes.SUSPENSE_AMBIENTE)
            personaje = new Texture(Gdx.files.internal(URL_TEXTURA_SUSPENSE));
        else
            personaje = new Texture(Gdx.files.internal(URL_TEXTURA_NEUTRA));

        coordenadas = new Vector2(Tools.centrarAncho(game, personaje),
        Tools.centrarAlto(game, personaje));
    }


    public static Mujer getInstancia(){
```

```
        if(unicaInstancia == null)
            unicaInstancia = new Mujer(game);

        return unicaInstancia;
    }
}
```

Las texturas de los personajes se añadirán a la carpeta *assets* del proyecto, concretamente a la carpeta *personajes*.



Una vez que hemos creado el nuevo personaje tenemos que añadirlo a una habitación.

```
public final class Salon extends Habitacion {
    //............

    private Salon(TheHouseOfCrimes game, Cursor c) {
        super(Pasillo.game, c);

        //.....................

        //Actores
        personaje = Anciana.getInstancia();
        personaje.setCoordenadas(0, 0);

        //incluimos los actores
```

```
        stage.addActor(personaje);

        //.............
    }

    //.............
}
```

Capítulo 10

Conclusiones

10.1. Experiencia personal

Sin duda alguna este ha sido el proyecto más grande que he realizado y en el que más tiempo he empleado. No ha sido fácil compaginar su desarrollo con otras obligaciones, debido a que me restaban mucho tiempo. Por ello una buena organización del tiempo ha sido esencial para poder continuar y que el desarrollo no se estancara durante mucho tiempo.

Este videojuego me ha enseñado muchas cosas que es muy posible que tenga que aplicar a lo largo de mi vida laboral. Entre ellas cabe mencionar, la importancia de una buena planificación, a cumplir con los plazos exigidos, a trabajar bajo presión como también la importancia de las etapas de desarrollo y la revisión constante del trabajo a fin de conseguir un producto final satisfactorio.

En mi opinión el peor momento que he vivido durante todo este tiempo ha sido cuando, debido a un error en el diagrama de clases inicial, tuve que hacer una reestructuración de gran parte del código, lo cual retrasó cerca de dos meses el progreso. Al principio fue muy duro ver como, casi todo lo que había implementado hasta la fecha estuvo mal. Sin embargo, incluso de esa mala experiencia, aprendí a sobreponerme, y al final pude conseguir un código que funcionaba.

Lo mejor de este proyecto ha sido poder dedicarme a lo que realmente me gusta: el desarrollo de videojuegos. Desde mi punto de vista no hay mejor satisfacción personal que ver como lo que has programado cobra vida en la pantalla, justo delante de tus ojos. Y sobre todo desarrollar algo que las personas de a pie pueden apreciar y pueden imaginar el gran trabajo que hay detrás. A diferencia de otros programas en los que se introducen datos por teclado y, tras realizar un complejo algoritmo que solo los que hemos estudiado alguna ingeniería conocemos, un resultado aparece en pantalla.

10.2. Trabajo futuro

Aunque se han cumplido los objetivos propuestos y estoy satisfecho con la versión final, hay algunas mejoras que, por la necesidad de terminar el proyecto lo antes posible no se han podido incluir. Estas podrían ser las siguientes:

- Mejorar el rendimiento de la versión de Android para poder desplegar el jugo también en dispositivos móviles.Dado que el experimento previsto se realizará, de momento, en

ordenadores de la universidad, la versión Android quedó en un segundo plano. Aunque juego funciona en dispositivos móviles, tiene problemas que impiden una experiencia satisfactoria. Uno de los problemas es, por ejemplo, el tiempo de carga de la partida, o que en algunas pantallas no se reproduce la música.

- Más tipos de puzles. Aunque ahora mismo solo hay tres tipos, este número puede ampliarse, lo cual enriquecería la experiencia del juego.
- Más habitaciones, personajes y objetos. Esto ampliaría el mundo que nos muestra el juego y, por consiguiente, aumentaría el número de puzles.
- Incluir más idiomas. De esta manera se puede ampliar el experimento e implementarlo en otros cursos de idiomas, aparte del curso de alemán.

Bibliografía

[Álvarez, 2009] Álvarez, R. B. (2009). El e-learning, una respuesta educativa a las demandas de las sociedades del siglo xxi. *Pixel-Bit: Revista de medios y educación*, (35):87–96.

[Bradley and Lang, 1999] Bradley, M. M. and Lang, P. J. (1999). Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Technical Report C-1, The Center for Research in Psychophysiology, University of Florida.

[Delatorre et al., 2015] Delatorre, P., Berns, A., Palomo-Duarte, M., Gervás, P., and Madueño, F. (2015). Diseño de un juego serio basado en el suspense. In *Proceedings 2st Congreso de la Sociedad Española para las Ciencias del Videojuego, Barcelona, Spain, June 24, 2015.*, pages 102–111.

[Freeman et al., 2004] Freeman, E., Robson, E., Bates, B., and Sierra, K. (2004). *Head first design patterns*. .ºReilly Media, Inc.".

[Gerdes et al., 2010] Gerdes, K. E., Segal, E. A., and Lietz, C. A. (2010). Conceptualising and measuring empathy. *British Journal of Social Work*, 40(7):2326–2343.

[Hunter, 2005] Hunter, D. (2005). *XML : curso de iniciación*. Barcelona:Inforbook's.

[Joosten et al., 2010] Joosten, E., Lankveld, G., and Spronck, P. (2010). Colors and emotions in video games. In *11th International Conference on Intelligent Games and Simulation GAME-ON*, pages 61–65.

[Marcano, 2008] Marcano, B. (2008). Juegos serios y entrenamiento en la sociedad digital. *Teoría de la Educación: Educación y Cultura en la Sociedad de la Información*, 9(3):5.

[Márquez and Sánchez, 2014] Márquez, D. S. and Sánchez, A. C. (2014). *Libgdx Cross-platform Game Development Cookbook*. Packt Publishing Ltd.

[Paiva et al., 2005] Paiva, A., Dias, J., Sobral, D., Aylett, R., Woods, S., Hall, L., and Zoll, C. (2005). Learning by feeling: Evoking empathy with synthetic characters. *Applied Artificial Intelligence*, 19(3-4):235–266.

[Peinado et al., 2005] Peinado, F., Gervás, P., and Moreno-Ger, P. (2005). Interactive storytelling in educational environments. In *3rd International Conference on Multimedia and ICTs in Education, Badajoz, Spain*. Citeseer.

[Santamas, 2014] Santamas, J. M. (2014). El mercado global de e-learning. Technical report, Online Business School.

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.
You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the

same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.
The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.
The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.
In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.
You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.
If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.
If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © 2015 Francisco Madueño Chulián. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.
If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.