



UNIVERSITAT
POLITÀCNICA
DE VALÈNCIA



Escola Tècnica
Superior d'Enginyeria
Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Amigurumi

Una historia de terror

Trabajo Fin de Grado
Grado en Ingeniería Informática

Autor: Vidal Tárraga Guillén

Tutor: Ramón Pascual Mollá Vayá

Curso 2019-2020

Dedicatoria

A mis padres, Vidal y Araceli, y a mi hermana Araceli, gracias a ellos, soy quien soy hoy en día y sólo les puedo dar las gracias por haberme apoyado durante mi etapa académica que hoy culmina.

A mi novia Paloma por haberme acompañado durante toda mi carrera académica y por ser mi fuente de inspiración en la vida.

A mis amigos Germán y Sebas, que me han aguantado durante la carrera y ayudado en todo lo posible. Mi gratitud hacia ellos es infinita.

A mi tutor Ramón Mollá, por abrirme el camino de la programación de videojuegos y permitirme poder realizar un juego como trabajo de fin de grado.

Y, sobre todo, dedicar este proyecto al mundo del videojuego, a todos aquellos momentos en los que me hicieron darme cuenta de que era la persona más feliz del mundo con una maquinita y un cartucho de videojuego y, que ahora lo es creando videojuegos.

Resumen

Amigurumi es un juego de terror en el que un bebé tiene que ir derrotando a unos muñecos de lana y avanzando tras las habitaciones de la casa para poder llegar al dormitorio de los padres y de esta manera conseguir la victoria. Todo esto se desarrolla en un mundo exagerado por el bebé. El videojuego parte de un trabajo realizado por un equipo interdisciplinar cuyos componentes eran alumnos de la Facultad de BB.AA. y la ETSINF y que se desarrolló bajo la iniciativa PIME/19-20/167.

Palabras clave: Unity, videojuego, jugador, amigurumi.

Abstract

Amigurumi is a horror game in which a baby has to defeat some wool dolls and advance through the rooms of the house in order to get to the parents' room and so win. All this takes place in a world exaggerated by the baby. The video game is based on a work carried out by an interdisciplinary team whose members were students from the Faculty of Fine Arts and the ETSINF and was developed under the initiative PIME/19-20/167.

Keywords : Unity, videogame, player, amigurumi.

Resum

Amigurumi és un joc de terror en el qual un bebé ha d'anar derrotant a uns ninots de llana i avançant per les habitacions de la casa per poder arribar al dormitori dels pares i d'aquesta manera aconseguir la victòria. Tot això es desenvolupa en un món exagerat per al bebé. El videojoc parteix d'un treball realitzat per un equip interdisciplinar, els components del qual eren alumnes de la Facultat de *BB.*AA. i la *ETSINF i que es va desenvolupar sota la iniciativa *PIME/19-20/167.

Paraules clau : Unity, videojoc, jugador, amigurumi.

Índice

Contenido

Capítulo 1. Introducción	13
1.1 Motivación	14
1.2 Objetivos.....	14
1.3 Impacto esperado	15
1.4 Alcance	15
1.5 Metodología	16
1.6 Estructura.....	16
1.7 Colaboraciones	17
1.8 Convenciones	18
Capítulo 2. Contexto tecnológico	20
2.1 Descripción de las herramientas	20
2.2 Experiencia previa con Unity	21
2.3 Propuesta.....	23
2.4 Juegos de miedo.....	26
Capítulo 3. Análisis del problema.....	31
3.1 Identificación de actores	31
3.2 Especificación de requisitos	31
3.3 Casos de uso.....	37
3.4 Identificación y análisis de soluciones posibles	41
3.5 Solución propuesta.....	43
3.6 Plan de trabajo	44
3.7 Presupuesto	45
Capítulo 4. Diseño de la solución	49
4.1 Arquitectura del sistema	49
4.2 Diagramas de clases	49
4.3 Diagramas de estados	62
4.4 Historia	66
4.5 Estética	66
4.6 Diseño de los controles.....	75
4.7 Diseño de los niveles.....	76
Capítulo 5. Desarrollo e implantación de la solución.....	84
5.1 Organización	84

5.2 Escenas	86
5.3 Implantación del videojuego	94
Capítulo 6. Pruebas	97
Capítulo 7. Conclusiones	103
Capítulo 8. Trabajos futuros.....	105
Capítulo 9. Webgrafía	107
ANEXO	109
1.1 Cambiar cursor predeterminado por otro	109
1.2 Luz encendida y animaciones acabadas	110
1.3 Actualizar cordura y energía	111
2.1 Detectar si un objeto puede ser interactuado y acciones	112
2.2 Movimiento del personaje y aspectos de la cámara	114
3.1 Bocadillos que cuentan la historia	115
3.2 Menú de pausa.....	116
3.3 Cambiar de nivel con transición	117
4.1 Controlador de la visión del amigurumi	118
4.2 Controlador del amigurumi	118
4.3 Enemy.....	119
4.4 Estado patrulla	120
4.5 Estado alerta	121
4.6 Estado persecución.....	121
4.7 Máquina de estados	122
5.1 Ejemplo de objeto con el que se puede interactuar	123
6.1 Ejemplo de susto.....	123
7.1 Sistema de guardado del videojuego	124
8. Estructura de Unity	126

Índice de tablas

Tabla 1. Comparación entre juegos.....	30
Tabla 2. ACT-01 Jugador.....	32
Tabla 3. RQF-01 Movimiento del personaje.....	32
Tabla 4. RQF-02 Iluminación.....	33
Tabla 5. RQF-03 Interactuar con la puerta al final del nivel.....	33
Tabla 6. RQF-04 Creación de las habitaciones.....	33
Tabla 7. RQF-05 Cámara del bebé.....	33
Tabla 8. RQF-06 Barra de cordura.....	33
Tabla 9. RQF-07 Barra de energía.....	33
Tabla 10. RQF-08 Subir.....	33
Tabla 11. RQF-09 Empujar.....	33
Tabla 12. RQF-10 Menú de pausa.....	34
Tabla 13. RQF-11 Inteligencia artificial del amigurumi.....	34
Tabla 14. RQF-12 Menú principal.....	34
Tabla 15. RQF-13 Música.....	34
Tabla 16. RQF-14 Sonidos.....	34
Tabla 17. RQF-15 Explosión de polillas al morir el amigurumi.....	34
Tabla 18. RQF-16 Icono para subir.....	35
Tabla 19. RQF-17 Icono para empujar o interactuar.....	35
Tabla 20. RQF-18 Pantalla Game Over.....	35
Tabla 21. RQF-19 Cinemática enfocando el objetivo del nivel 1.....	35
Tabla 22. RQF-20 Bocadillos para ir contando la historia.....	35
Tabla 23. RQF-21 Neblina.....	35
Tabla 24. RQF-22 Transiciones.....	36
Tabla 25. RQF-23 Perder.....	36
Tabla 26. RQF-24 Salir del juego.....	36
Tabla 27. RQF-25 Guardar partida.....	36
Tabla 28. RQF-26 Gestionar música y sonidos.....	36
Tabla 29. RQF-27 Continuar partida guardada.....	36
Tabla 30. RQF-28 Generar archivo de guardado.....	36
Tabla 31. RQF-29 Linterna.....	37
Tabla 32. RQF-30 Sustos.....	37
Tabla 33. RQF-31 Cursor.....	37
Tabla 34. RQF-32 Animación en el caso de que te pille un amigurumi.....	37
Tabla 35. RQF-33 Animación para cuando el jugador derrote al amigurumi.....	37
Tabla 36. RQF-34 Bloques de bebé.....	37
Tabla 37. RQF-35 Ganar	38
Tabla 38. RQNF-01 Almacenamiento mínimo.....	38
Tabla 39. RQNF-02 Respuesta rápida.....	38
Tabla 40. RQNF-03 Compatibilidad con todos los sistemas operativos.....	38
Tabla 41. RQNF-04 Idioma.....	38
Tabla 42. RQNF-05 Disponer de un ordenador.....	38
Tabla 43. CU-01 Comenzar nueva partida.....	39

Tabla 44. CU-02 Continuar partida guardada.....	39
Tabla 45. CU-03 Opciones.....	39
Tabla 46. CU-04 Mirar controles.....	40
Tabla 47. CU-05 Salir.....	40
Tabla 48. CU-06 Jugar.....	40
Tabla 49. CU-07 Modificar música.....	40
Tabla 50. CU-08 Modificar sensibilidad horizontal.....	40
Tabla 51. CU-09 Modificar sensibilidad vertical.....	40
Tabla 52. CU-10 Subir.....	41
Tabla 53. CU-11 Empujar.....	41
Tabla 54. CU-12 Pulsar / Interactuar.....	41
Tabla 55. CU-13 Pausar juego.....	41
Tabla 56. CU-14 Mover personaje.....	41
Tabla 57. CU-15 Continuar.....	41
Tabla 58. CU-16 Ir al menú principal.....	42
Tabla 59. Presupuesto hardware.....	47
Tabla 60. Presupuesto software.....	47
Tabla 61. Presupuesto recursos humanos.....	48
Tabla 62. Presupuesto total.....	48
Tabla 63. IU-01 Menú principal.....	72
Tabla 64. IU-02 Menú de opciones.....	73
Tabla 65. IU-03 Menú de controles.....	73
Tabla 66. IU-04 Menú de pausa.....	74
Tabla 67. IU-05 Game over.....	74
Tabla 68. IU-06 Pantalla de juego.....	75
Tabla 69. Controles.....	77
Tabla 70. NVL-01 Despacho.....	77
Tabla 71. NVL-02 Baño.....	78
Tabla 72. EDT-01 Pasillo.....	78
Tabla 73. EDT-02 Entrada de la casa.....	79
Tabla 74. NVL-03 Cocina.....	79
Tabla 75. NVL-04 PT1 Salón.....	80
Tabla 76. NVL-04 PT2 Laberinto.....	80
Tabla 77. NVL-04 PT3 Salón fantasma.....	81
Tabla 78. NVL-05 Garaje.....	82
Tabla 79. EDT-03 Entrada de la casa soleada.....	83
Tabla 80. EDT-04 Pasillo soleado.....	83
Tabla 81. Primera encuesta de Amigurumi.....	95
Tabla 82. Segunda encuesta de Amigurumi.....	96
Tabla 83. PR-01 Estado patrulla.....	98
Tabla 84. PR-02 Estado alerta.....	98
Tabla 85. PR-03 Estado persecución.....	98
Tabla 86. PR-04 Perder cordura cuando el jugador mira al amigurumi.....	99
Tabla 87. PR-05 El jugador es teletransportado al utilizar un teletransporte.....	99
Tabla 88. PR-06 Interactuar con un objeto.....	99
Tabla 89. PR-07 Subirse a un objeto.....	99
Tabla 90. PR-08 Empujar un objeto.....	100
Tabla 91. PR-09 Menú principal.....	100



Tabla 92. PR-10 Sustos.....	100
Tabla 93. PR-11 Animaciones adecuadas.....	100
Tabla 94. PR-12 Sistema de guardado.....	101
Tabla 95. PR-13 Prueba final.....	101
Tabla 96. PR-14 Menú de pausa.....	101
Tabla 97. PR-15 Opciones.....	102

Índice de figuras

Figura 1. Ejemplo de amigurumi.....	16
Figura 2. Tablero Kanban utilizado para el desarrollo del videojuego Amigurumi.....	17
Figura 3. Gameplay de Greedy.....	22
Figura 4. Pantalla de selección de nivel de Greedy.....	23
Figura 5. <i>Script</i> del videojuego Greedy.....	23
Figura 6. Logo de Unity.....	24
Figura 7. Gameplay de Cuphead.....	25
Figura 8. Gameplay de Gris.....	25
Figura 9. Logo de Visual Studio.....	26
Figura 10. Ventana de código de Visual Studio.....	26
Figura 11. Portada de “among the sleep”.....	28
Figura 12. Segundo nivel de “among the sleep”.....	28
Figura 13. Portada de “Slender: the eight pages”.....	29
Figura 14. <i>Gameplay</i> de “Slender: the eight pages”.....	29
Figura 15. Imagen del hilo de lana dentro del videojuego.....	42
Figura 16. Rastro que deja el hilo de lana dentro del videojuego.....	43
Figura 17. Bloques de letras utilizados en el videojuego.....	44
Figura 18. Bloque de letra del primer nivel.....	44
Figura 19. Conexión de Vodafone.....	46
Figura 20. Ordenador utilizado.....	46
Figura 21. Bebé.....	67
Figura 22. Oso.....	68
Figura 23. Pájaro.....	69
Figura 24. Oveja.....	69
Figura 25. Mono.....	70
Figura 26. Conejo.....	70
Figura 27. Garaje.....	71
Figura 28. Salón.....	71
Figura 29. Cocina.....	72
Figura 30. Estructura de los niveles de los juegos.....	87
Figura 31. Estructura de MainControl.....	88
Figura 32. Estructura de Player.....	89
Figura 33. Jerarquía de interfaces.....	90
Figura 34. Estructura del enemigo.....	91
Figura 35. Estructura de los sustos.....	93
Figura 36. Escena principal de Unity.....	127
Figura 37. Pestaña de la escena.....	127
Figura 38. Pestaña de animaciones.....	128
Figura 39. Pestaña de jerarquía.....	128
Figura 40. Pestaña de inspector.....	129
Figura 41. Pestaña del proyecto.....	129



Índice de diagramas

Diagrama 1. Diagrama de casos de uso de Amigurumi.....	39
Diagrama 2. Diagrama de clases del controlador del videojuego.....	51
Diagrama 3. Diagrama de clases principal de la inteligencia artificial.....	52
Diagrama 4. Diagrama de clases de la inteligencia artificial del resto de amigurumis.....	53
Diagrama 5. Diagrama de clases de la inteligencia artificial del pájaro.....	54
Diagrama 6. Diagrama de clases del personaje del juego.....	55
Diagrama 7. Diagrama de clases de los bloques recolectables del bebé.....	56
Diagrama 8. Diagrama de clases de las puertas con las que se interactúan.....	57
Diagrama 9. Diagrama de clases de los objetos del nivel 1.....	57
Diagrama 10. Diagrama de clases de los objetos de los niveles 2 y 3.....	58
Diagrama 11. Diagrama de clases de los objetos de los niveles 4 y 5.....	58
Diagrama 12. Diagrama de clases de los objetos de la última escena.....	59
Diagrama 13. Diagrama de clases de varios objetos utilizados a lo largo de varios niveles.....	59
Diagrama 14. Diagrama de clases relacionadas con la interfaz del usuario.....	60
Diagrama 15. Segundo diagrama de clases relacionadas con la interfaz del usuario.....	61
Diagrama 16. Diagrama de clases relacionadas con la función de cargar y guardar el juego.....	61
Diagrama 17. Diagrama de clases relacionadas con los sustos ocasionados por amigurumis.....	62
Diagrama 18. Diagrama de clases relacionadas con los sustos ocasionados por objetos, imágenes o vídeos.....	63
Diagrama 19. Diagrama de estados del bebé.....	64
Diagrama 20. Diagrama de estados de la inteligencia artificial de los amigurumis.....	65
Diagrama 21. Diagrama de estados del oso, oveja y mono.....	65
Diagrama 22. Diagrama de estados del pájaro.....	66
Diagrama 23. Diagrama de estados del conejo.....	66
Diagrama 24. Diagrama de flujo.....	75

Capítulo 1

Introducción

La industria del videojuego es uno de los sectores más complejos y emergentes de estos últimos tiempos. Un videojuego engloba múltiples disciplinas como son el arte de escribir el guion de una historia, crear la música y sonidos adecuados, realizar dibujos y sus respectivos modelados con animaciones y, en mi caso, la programación del videojuego, el hacer que todas estas disciplinas estén coordinadas a la perfección [4].

Este conjunto tan amplio de tantos artes que involucra tanto software como hardware hacen que un videojuego pueda llegar a ser una obra de arte. En muchos lugares del mundo como Estados Unidos, se está empezando a introducir en su cultura los videojuegos incluso se está normalizando el uso de videojuegos en bibliotecas¹. Aunque sea diferente de un libro, de una película o de una serie, el fin es el mismo, generar entretenimiento y experimentar.

Lo que diferencia a un videojuego es que, a la vez que se está disfrutando de la experiencia y te entretienes, estás compitiendo por ganar [3]. Pero ¿qué sucede cuando se habla de juegos de miedo? Es un tipo de experiencia que el jugador sabe que lo va a pasar mal [2], aunque raramente disfruta, sobre todo en adolescentes ya que puede ser visto como una prueba de valor o de ver hasta dónde pueden llegar, incluso a veces suelen jugar en grupo para que la experiencia sea más amena.

¿Por qué atrae el terror? Según Niels Boehnke de la Federación Cultural Digital de Videojuegos alemana "Pasar miedo en un entorno controlado es una técnica cultural ancestral, los juegos de horror tienen una función similar a la de contar historias de miedo junto al fuego o dar un paseo nocturno en el campamento de verano"²). Además, hay que recalcar que hay una gran cantidad de jóvenes prefieren ver este contenido que jugarlo ya que no solo se experimenta el juego de esta manera, sino que se vive junto con el jugador.

Aunque resulte sorprendente, pero los juegos de miedo son actualmente de los más demandados de la industria de los videojuegos [1], especialmente aquellos en los que solo puedes huir o esconderte de aquello que causa terror. Y así es Amigurumi, un juego de terror en el que unos muñecos de lana cobran vida por medio de unas polillas que se introducen en éstos. Antes se hablaba de juegos en el que el personaje está huyendo constantemente del enemigo, es decir que el personaje actúa de una manera indefensa y, que hay menos indefenso que un bebé.

Eso es Amigurumi, un bebé que se despierta en una oscura noche y que tendrá que ir avanzando por los diferentes lugares de la casa recogiendo pistas para poder llegar a la habitación de sus padres y así estar a salvo de los macabros muñecos de lana que le perseguirán hasta atraparlo.

¹ <https://www.infotecarios.com/videojuegos-y-bibliotecas/#.XvdfySgzZPb>

² <https://www.diariolasamericas.com/tecnologia/el-secreto-del-exito-los-videojuegos-terror-n3777818>



1.1 Motivación

Desde que era pequeño, he sido un gran amante de los videojuegos, de pasarme los viajes enteros jugando con la Nintendo DS y la GameBoy Color a contar las horas para que llegasen los fines de semana para poder jugar a las consolas porque mis padres me lo prohibían entre semana.

Quitando el lado de jugador de videojuegos, antes de empezar la carrera me empezó a entrar la curiosidad acerca del funcionamiento interno de los juegos y, desde que empecé siempre he querido tocar programación de videojuegos. No fue hasta el segundo cuatrimestre del tercer curso en el que comencé a programar videojuegos y, supe que era lo que quería hacer durante el resto de mi vida.

Además, se está hablando de un mercado que no para de crecer, tanto en las personas que trabajan en él como las personas que hacen uso de los videojuegos o consolas diariamente.

La oportunidad que se me ha dado en la asignatura de Introducción a la Programación de Videojuegos de poder continuar la realización de un videojuego como trabajo final de grado es maravillosa y, no me he sentido tan cómodo y contento de trabajar en un proyecto en mi vida.

1.2 Objetivos

El objetivo principal de este trabajo es el de la continuación y realización de cinco niveles del videojuego Amigurumi con el fin de poder luego plantear una recaudación de fondos para mejorar el juego y acabarlo, o emplear el proyecto dentro de un portafolio con el fin de obtener trabajo en la industria.

A su vez personalmente se espera alcanzar unos objetivos secundarios tras la finalización de este proyecto.

- Aprendizaje del manejo de la herramienta seleccionada para el desarrollo del videojuego.
- Aprendizaje de las distintas fases de creación, elaboración y despliegue de un videojuego para el entorno escogido.
- Creación de la inteligencia artificial de los muñecos de lana.
- Desarrollar una historia para el juego que entretenga a la vez que provoque terror.
- Permitir al jugador guardar su progreso en la partida, es decir, que pueda volver al momento en el que dejó de jugar.

1.3 Impacto esperado

El juego está pensado para que cause una serie de emociones en el jugador. Provoca **miedo** porque los amigurumis, estos muñecos de lana de origen japonés (Ver Figura 1.1) que aparentan ser muñecos normales durante el día pero que, su percepción cambia totalmente por la noche adoptando una apariencia terrorífica. A esto hay que sumarle la oscuridad en la que se ve inmerso el jugador y que nubla su visión de manera que acarrea una sensación de **desorientación** al jugador. También se busca que haya un poco de **desconocimiento**, es decir, que el jugador tenga que investigar qué es lo que tiene que hacer para poder completar el nivel a pesar de la presión que supone el estar constantemente huyendo de algo.

Una vez que se ha conseguido completar alguno de los niveles, estos factores producen una **satisfacción** mayor a la que se consigue en otro tipo de juegos.



Figura 1: Ejemplo de amigurumi

1.4 Alcance

Este juego está destinado a un público mayor de 12 años (PEGI 12), ya que según la normativa PEGI³, que es la encargada de poner las restricciones de edad en los videojuegos, aunque sea un juego de terror, no hay casi nada de violencia y el terror es más fantástico que real. De ahí que sea para jugadores de 12 o más años, sobre todo jugadores que se quieran iniciar en este tipo de videojuegos o que ya tengan experiencia en juegos de miedo.

³ <https://pegi.info/es/node/59>

1.5 Metodología

Para realizar este trabajo se ha empleado la metodología ágil. En la rama de Ingeniería del Software, sobre todo en estos últimos años se ha adoptado para todo tipo de proyectos el uso de este tipo de metodología. Debido a esto, la manera de trabajar y de organizarse será mucho más sencilla ya que es la realizada durante los últimos dos años.

En resumen, se irán proponiendo unas tareas previamente planteadas y diseñadas que se deberán realizar al cabo de un período de tiempo alrededor de dos semanas con sus respectivas pruebas para que se compruebe que el funcionamiento es el deseado y el correcto. Todo esto gracias a la aplicación Trello.

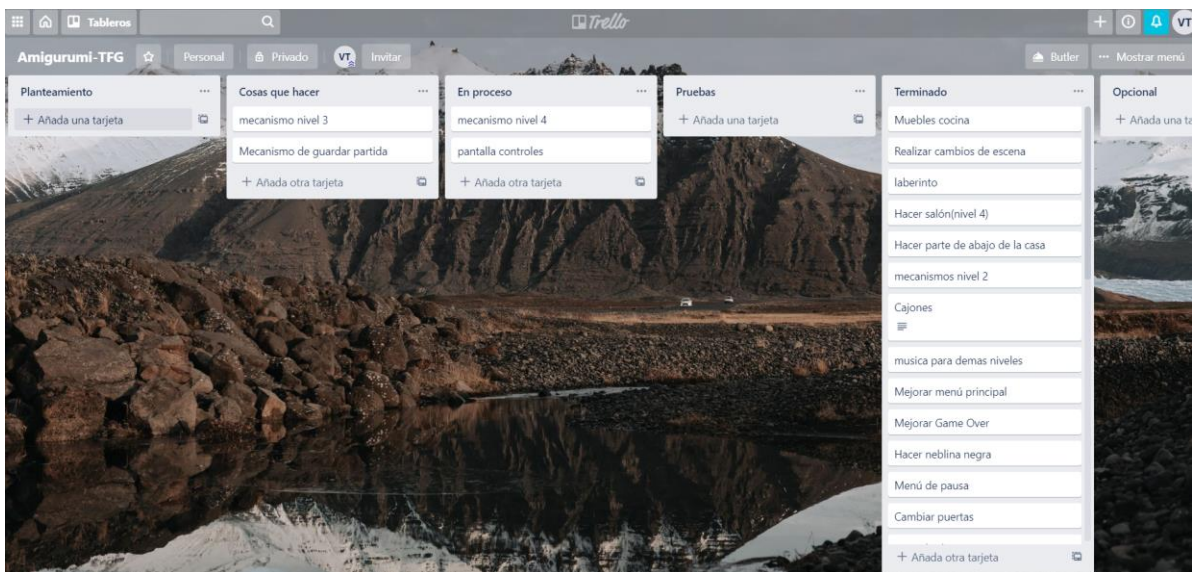


Figura 2: Tablero Kanban utilizado para el desarrollo del videojuego Amigurumi

1.6 Estructura

En cuanto a la estructura del documento, se va a dividir por capítulos con sus respectivos apartados.

- **Capítulo 2. Contexto tecnológico:** en esta sección se habla de las herramientas que se pueden usar para este proyecto y las que finalmente han sido seleccionadas. También se nombrarán a los juegos que han servido de referencia para este juego.
- **Capítulo 3. Análisis del problema:** se hablará de la identificación de los requisitos del videojuego, los respectivos casos de uso y diagramas UML. Después se comentan las posibles soluciones del proyecto, así como la que ha sido propuesta. Finalmente, el plan de trabajo y el presupuesto.

- **Capítulo 4. Diseño de la solución:** este capítulo trata de la arquitectura del sistema y el diseño detallado que engloba el diseño del juego, diseño de la interfaz, etc.
- **Capítulo 5. Desarrollo e implantación de la solución propuesta:** en este apartado se describe todo el desarrollo e implantación del videojuego, con sus problemas o dificultades encontradas.
- **Capítulo 6. Pruebas:** se muestran las pruebas realizadas para el correcto funcionamiento del proyecto.
- **Capítulo 7. Conclusiones:** se realizarán las conclusiones correspondientes a los objetivos puestos al principio de este documento.
- **Capítulo 8. Trabajos futuros:** en esta parte del documento se podrá ver los planes de futuro que tiene el programador con dicho proyecto.
- **Capítulo 9. Bibliografía:** se hablará de las referencias bibliográficas en este capítulo.

Al final del documento habrá un anexo en el que se detallarán partes de código introducidos en el capítulo 5.

1.7 Colaboraciones

Como se ha comentado previamente, este proyecto es la continuación de un trabajo realizado en la asignatura de Introducción a la Programación de Videojuegos donde se formó un grupo junto con estudiantes de bellas artes en el que cada uno realizó una parte del videojuego. A continuación, se detallará la función de cada integrante.

- Silvia Rumi (Artes)
 - Realización de bocetos.
 - Diseño y sonidos del nivel 1.
- Ariel Pascual (Artes)
 - Diseño, modelado y animaciones del amigurumi, bebé y polilla del nivel 1.
- Moisés Mingarro
 - Diseño del menú principal antiguo (actualmente hay otro menú principal).
- Miguel Ángel Zurriaga
 - Realización de la animación de la polilla que sale del amigurumi y va hacia la fuente de luz (actualmente eliminado).
 - Creación del hilo de lana
- Zhihao Zhang:
 - Movimiento del personaje.
 - Habilidades de subir y correr.
 - Objetos con los que el personaje puede interactuar y sus funcionalidades.
 - Barras de energía y de cordura para la interfaz.

- Aspectos de iluminación.
 - Animación de la cámara.
 - Menú de pausa antiguo.
- **Vidal Tárraga**
 - Inteligencia artificial del amigurumi.
 - Creación del nivel 1 (objetos, suelo, paredes, techo).
 - Funcionalidad del menú principal.
 - Música para el videojuego.
 - Funcionalidad de la polilla.
 - Fuentes de la letra.
 - Iconos para la interfaz que indican cuando algo es interactivo, bien sea para subir o por interactuar con un objeto.
 - Pantalla del final del juego.
 - Opción de controles en el menú principal.
 - Cinemática para indicar el objetivo del nivel.
 - Realización de la escena del pasillo.
 - Explosión de polillas al derrotar al amigurumi y su posterior desaparición.

Por último, también se ha realizado una colaboración a través de la página web “Workana”, en la que se contrató a Daniel, un animador profesional de argentina que se encargó del diseño, modelado y animaciones de los cuatro amigurumis restantes.

1.8 Convenciones

En este apartado se especificarán los convencionalismos que aparecerán a lo largo del documento.

- Las palabras extranjeras serán escritas en cursiva.
- Se entrecomillarán las citas textuales externas a la obra, así como títulos de otros videojuegos o de instituciones.
- Las referencias al anexo irán entre paréntesis, subrayadas y con tonos de color violeta, por ejemplo: ([Anexo 3.1](#))
- Las referencias a la bibliografía irán entre corchetes y mediante un número indicarán a que fuente de la bibliografía se corresponde.
- Las notas de pie de páginas van referidas a páginas web que son utilizadas para basarse en pequeños datos utilizados en la memoria. Se indicarán mediante los superíndices^(1,2,etc).

Capítulo 2

Contexto tecnológico

A continuación, se hablará acerca de las herramientas contempladas para la realización de este proyecto, tanto en la parte de motores de videojuegos como en la parte de los entornos de desarrollo para el código. Y finalmente se concluirá que herramientas han sido elegidas para realizar el videojuego

2.1 Descripción de las herramientas

2.1.1 Motores de videojuegos

Para empezar, no se puede realizar ningún videojuego sin disponer de un motor de videojuegos. El concepto motor de videojuegos hace referencia a un software el cual tiene una serie de rutinas de programación que permiten el diseño, la creación y el funcionamiento de un entorno interactivo; es decir, de un videojuego.

Dentro de las funcionalidades típicas que tiene un motor de videojuegos, éstas son las que más destacadas:

- Motor gráfico para renderizar gráficos 2D y 3D
- Motor físico que simule las leyes de la física
- Animaciones
- Sonidos
- Inteligencia Artificial
- Programación o scripting

En cuanto a decidir el motor de videojuegos que se utilizará para este videojuego, en ningún momento se ha contemplado otro motor que no sea Unity ya que, aunque haya opciones más especializadas como Unreal Engine, la curva de aprendizaje es mucho mayor a la de Unity⁴. Además, hay que añadir que el proyecto fue empezado en Unity y portar un proyecto de Unity a otro motor de videojuegos sería una locura ya que habría que implementar el 80% del proyecto de nuevo.

2.1.2 Entornos de desarrollo

Una vez elegido el motor de videojuegos, hay que mirar que lenguajes soporta. En el caso de Unity, básicamente las opciones se reducen a dos: C# y UnityScript, un lenguaje diseñado específicamente para Unity y modelado tras JavaScript. Por sencillez y familiaridad con el lenguaje durante lo largo de la carrera, se ha optado por que el proyecto sea en C#.

⁴ <https://www.avantedigitalinstitute.es/2019/05/13/unity-o-unreal-por-donde-empiezo/>



En cuanto a los entornos de desarrollo integrado para utilizar C#, se han contemplado las opciones de Visual Studio y Monodevelop⁵. Visual Studio es un conjunto de herramientas de desarrollo de software basadas en componentes y otras tecnologías para crear aplicaciones potentes y de alto rendimiento. Por otro lado, MonoDevelop se detalla como "IDE multiplataforma para C #, F # y más ". Permite a los desarrolladores escribir rápidamente aplicaciones de escritorio y web en Linux, Windows y Mac OS X.

Por razones parecidas a las anteriores, se ha decidido apostar por Visual Studio ya que, durante la carrera siempre se ha trabajado en este entorno y ya se empezó a realizar el proyecto en este entorno durante la asignatura de Introducción a la Programación de Videojuegos.

2.2 Experiencia previa con Unity

Antes de que diera a comienzo este proyecto, ya se tenía conocimiento acerca de Unity. En el segundo cuatrimestre del tercer curso se dio la oportunidad de realizar un videojuego con el motor de videojuegos que gustase. Esa fue la primera toma de contacto con Unity. El juego realizado se llamaba "Greedy". Este videojuego es una referencia al clásico "Pac-man" pero con algunas novedades como la aparición de trampas en el mapa o habilidades especiales.

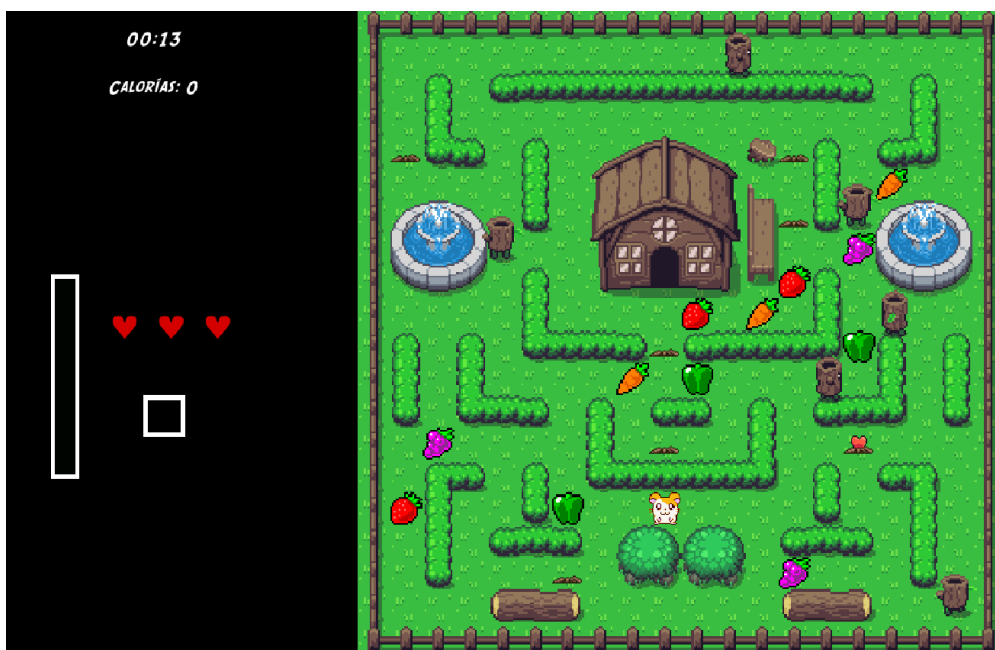


Figura 3: Gameplay de Greedy

Como se puede apreciar, visualmente no es ninguna maravilla, aparte de que la parte en negro de la izquierda no va acorde a lo que viene a ser la pantalla principal, hubiese quedado mucho mejor si todas estas cosas que van a la izquierda estuviesen implementadas en la pantalla del *gameplay*. Otro fallo que tiene el juego es que la inteligencia artificial de los enemigos es de lo más simple que se puede programar, todo lo contrario, a la desarrollada en el videojuego de Amigurumi.

⁵ <https://stackshare.io/stackups/monodevelop-vs-visual-studio#description>



Figura 4: Pantalla de selección de nivel de Greedy

También hay que destacar que antes, todos los textos se realizaban sin utilizar la herramienta llamada “TextMesh Pro” que sirve para mejorar la calidad de los textos y por lo tanto de la interfaz. Todos estos cambios y más se ven reflejados en el desarrollo de Amigurumi.

Por último, recalcar el uso de Visual Studio en este proyecto, que ha hecho que, de cara a nuevos proyectos como Amigurumi, el dominio de C# y los elementos asociados con Unity se vea aumentado de manera notable.

```

MyScriptAssembly
+ EstadoInvulnerable
+ velocidadDeParpadeo
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class EstadoInvulnerable : Estado {
7
8     [Range(0f, 1000f)]
9     float velocidadDeParpadeo = 400f;
10    GameObject soundEffect;
11
12
13    public override void RecibirDaño(int daño, PlayerController player) { }
14
15    private void ParpadeoVisual(float velocidad) {
16        Color colorActual = gameObject.GetComponent<SpriteRenderer>().color;
17        gameObject.GetComponent<SpriteRenderer>().color = new Color(colorActual.r, colorActual.g, colorActual.b, ((Time.time * velocidad) % 255) / 255);
18    }
19
20    public override void FinalizarEstado(PlayerController player) {
21        Color colorActual = gameObject.GetComponent<SpriteRenderer>().color;
22        AcabarEfectoDeSonido();
23        GetComponent<SpriteRenderer>().color = new Color(colorActual.r, colorActual.g, colorActual.b, 1f);
24    }
25
26    private void ComenzarEfectoDeSonido() {
27        soundEffect = Instantiate(Resources.Load<GameObject>("SoundEffectInvincible"));
28    }
29
30    private void AcabarEfectoDeSonido() {
31        Destroy(soundEffect);
32    }
33
34    void Update() {
35        ParpadeoVisual(velocidadDeParpadeo);
36    }
37

```

Figura 5: Script del videojuego Greedy

2.3 Propuesta

Para este proyecto se ha utilizado Visual Studio 2017 para la parte de programación y, Unity 3D para la parte del desarrollo del videojuego. Posteriormente se detallará en profundidad por qué se han elegido ambas herramientas.

UNITY 3D



Figura 6: Logo de Unity

Uno de los puntos fuertes de Unity se centra en la gran comunidad que tiene, lo que permite tener acceso a multitud de documentación y foros en los que se resuelven dudas además de explicar diferentes métodos o técnicas nuevas. Por esto, Unity⁶ es uno de los motores predilectos para aprender a desarrollar videojuegos; ya que supone una puerta de acceso perfecta para aquellos que quieren incursionar en esta industria.

Está disponible como plataforma de desarrollo para Windows, Mac OS y Linux. Gracias a Unity⁷ se pueden crear videojuegos para varias plataformas tales como PC, consolas y móviles a través de un editor visual y programación vía scripting y, pudiendo conseguir resultados totalmente profesionales.

Una ventaja muy importante de Unity es que tiene una tienda llamada “Asset Store”⁸, en la cual se pueden encontrar infinidad de recursos tanto gratuitos como de pago.

Otra facilidad que proporciona este entorno es que es gratuito para el uso personal, en el caso de que sea una empresa habría que adoptar un plan de pago.

Prueba de todo esto que se ha comentado son videojuegos famosos creados con Unity como “Gris” (Ver Figura 2.5) y “Cuphead” (Ver Figura 2.6).

Para ver la estructura de Unity, consultar el anexo ([Anexo 8](#))

6 <https://www.masterd.es/blog/que-es-unity-3d-tutorial/>

7 [https://es.wikipedia.org/wiki/Unity_\(motor_de_videojuego\)](https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego))

8 <https://assetstore.unity.com/>



Figura 7: *Gameplay de Cuphead*



Figura 8: *Gameplay de Gris*



Figura 9: Logo de Visual Studio

Es un entorno de desarrollo integrado para sistemas operativos Windows⁹. Soporta varios lenguajes de programación entre otros Visual C++ y Visual C#, en este caso el proyecto será desarrollado en C#.

Como se puede apreciar en la siguiente figura, se trata de un entorno de programación donde se puede diferenciar entre la parte en la que se sitúa el código y la parte de la derecha que, básicamente engloba el explorador de soluciones.

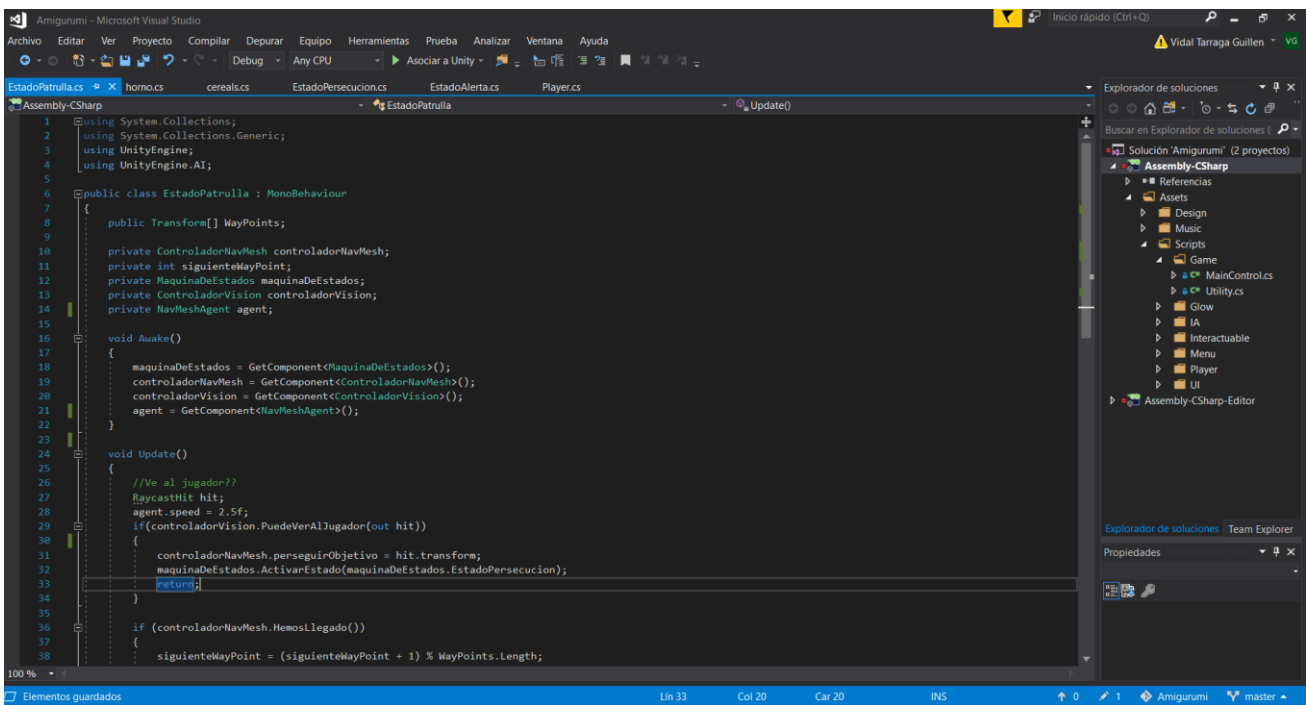


Figura 10: Ventana de código de Visual Studio

⁹ Fuentes consultadas: [https://docs.microsoft.com/en-us/previous-versions/aa700918\(v=msdn.10\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/aa700918(v=msdn.10)?redirectedfrom=MSDN), <https://visualstudio.microsoft.com/es/>



2.4 Juegos de miedo

Todos los juegos de miedo deben de tener unas características en común para que pueda ser considerado como un buen juego de miedo [4].

- **Ambientación:** la atmósfera es determinante para que un juego de terror sea bueno, porque es lo que nos va a atrapar. Los juegos de miedo suelen transcurrir en ambientes claustrofóbicos y angustiosos para transmitir esa hostilidad. A esto hay que sumarle la oscuridad, elemento clave para que el protagonista se sienta indefenso al no tener visión. Por último, una buena gestión de la música y de los sonidos hace que el ambiente sea perfecto para aterrorizar al jugador.
- **Soledad del protagonista:** una de las cosas que más pueden angustiar es la soledad. Casi siempre se repite un mismo patrón: personas normales y corrientes que tienen que desentrañar un misterio y que, por razones desconocidas, acaban en un lugar hostil.
- **Misterio:** otro elemento clave de los juegos de miedo es el misterio, la sensación de desorientación en el entorno, de tener que investigar los alrededores para saber que hay que hacer en el nivel.
- **Enemigos inhumanos:** sin duda uno de los atractivos visuales más importantes de un juego de terror son los enemigos. Los enemigos son la extensión de la atmósfera y tienen que cuadrar con ella. Los enemigos son mayormente antropomórficos. Aunque sean monstruos, fantasmas o muertos vivientes, casi todos tienen forma humana. Pero son seres que rompen las reglas de la naturaleza y eso trastorna al jugador. Son impredecibles, extremadamente fuertes o violentos.
- **Jugabilidad:** el género, especialmente el *survival horror*, nunca ha brillado demasiado por su buena jugabilidad. Los *survival* clásicos podían llegar a ser bastante frustrantes y desesperantes. La capacidad de reacción del personaje que maneja el jugador no suele ser muy buena. Movimientos lentos, poco ágiles y poca defensa frente a los implacables enemigos. También hay que recalcar el uso de puzzles o acertijos durante el transcurso de un juego de miedo y ocasionar esa situación de intentar resolver algo mientras el jugador está aterrorizado.
- **Sustos:** no hay nada peor que el miedo de saber que un monstruo horrible puede estar escondido en cualquier parte y va a asustar al jugador. De esta manera se consigue un miedo directo y rápido, con mucha más tensión para el jugador. Aun así, el miedo es algo personal, a algunos les asustará más los sustos y a otros el terror psicológico.

Posteriormente, se hablará de los videojuegos que han sido una referencia a la hora de hacer este proyecto y, se hablará de sus pros y de sus contras. Adicionalmente se hará una tabla con estos elementos y con la calificación que se les da a estos juegos en estos aspectos.

¹⁰ <http://www.psychologyofgames.com/2015/10/the-psychology-of-horror-games/>

AMONG THE SLEEP



Figura 11: Portada de “among the sleep”

“Among the sleep” es un videojuego de terror del 2014 en el que un niño despierta en una noche junto con su osito de peluche y va recorriendo mundos fantásticos mientras huye de un ser maligno. De este juego gustó la idea de que el protagonista fuese un bebé ya que es un ser frágil e indefenso. Además, en este juego se combinan escenarios de una casa normal con escenarios donde se ve un mundo exagerado o inventado por el bebé, lo que da juego a hacer unos escenarios más fantásticos y complejos. Esto último también es introducido dentro de Amigurumi.



Figura 12: Segundo nivel de “among the sleep”

Lo malo de este juego es que, para ser un juego de terror, el antagonista o el ser que provoca miedo no interviene hasta los últimos instantes del juego. A esto le sumas que los puzles para pasar de pantalla o de nivel no están del todo claros, el jugador puede estar perfectamente estancado en un nivel sin pasar miedo y sin saber que hacer. Por la otra parte, la historia y la manera en la que se realiza la idea de un bebé como protagonista es magistral.

SLENDER: THE EIGHT PAGES

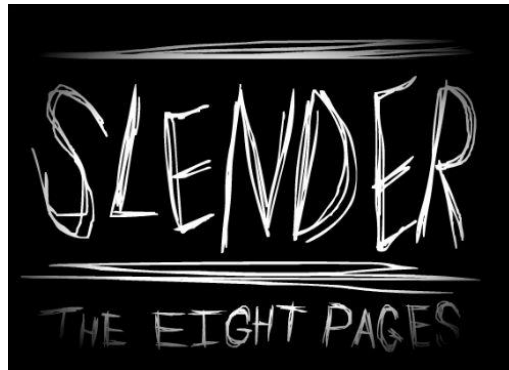


Figura 13: Portada de “Slender: the eight pages”

¿Quién no conoce a Slender? Uno de los famosísimos juegos de miedo que triunfó desde el 2012 hasta los tiempos más actuales. En este videojuego, el jugador tendrá que encontrar 8 páginas en un bosque sombrío para poder completar el juego, todo esto mientras se huye de un señor sin cara y con traje de altura exagerada.



Figura 14: *Gameplay* de “Slender: the eight pages”

Por sacarle alguna contra a este juego, simplemente se tiene que decir que se ha quedado anticuado, no ha habido ninguna actualización de este juego desde el 2013. Es de lo más sencillo en cuanto a juegos de miedo. En cuanto a los pros, es un juego donde se está completamente en tensión ya que, no tienes casi iluminación ni información de donde están las páginas a coleccionar ni de donde se encuentra nuestro enemigo, por lo que hace más agobiante la experiencia y añade ese toque de terror psicológico.

A continuación, se realizará una tabla comparando estos juegos según las características expuestas al principio del apartado. Habrá cinco tipos de calificaciones: muy bueno/a, bueno/a, regular, malo/a y muy malo/a.

Características	Among the sleep	Slender
Ambientación	Muy buena	Buena
Soledad	Buena	Buena
Misterio	Muy bueno	Regular
Enemigo	Muy bueno	Bueno
Jugabilidad	Muy buena	Buena
Susto	Regular	Bueno

Tabla 1: Comparación entre juegos

CONCLUSIÓN

Como se puede ver *Among the sleep* es mejor juego ya que es un juego más moderno y por lo tanto mejor hecho. Además, tiene una historia intrigante, un enemigo que detecta si el jugador hace ruido, una jugabilidad asombrosa, etc. Aun así, *Slender* es un juego muy bueno ya que a pesar de los años que han pasado sigue teniendo rasgos buenos encima de que ha sido el pionero de muchas técnicas de los juegos de miedo.

Para finalizar, se han adaptado mecánicas de estos juegos, como las habilidades que tiene el bebé en “among the sleep” como empujar objetos o abrir cajones, que si el jugador se queda mirando al enemigo pierde en el juego como sucede en “Slender: the eight pages”, y demás ideas que han servido para el desarrollo de este videojuego.

Capítulo 3

Análisis del problema

En esta sección se procederá a analizar aspectos del sistema como la especificación del comportamiento y sus funcionalidades. Para ello, se hará uso de las técnicas aprendidas durante la carrera como vienen a ser la identificación y especificación de requisitos, así como los diagramas de casos de uso.

Seguidamente se van a realizar tablas con la que describir los requisitos, actores y demás componentes del análisis. Dentro de estas tablas se utilizarán unos apartados como los siguientes:

- **ID:** identificador. Es un conjunto de letras y números cuyo objetivo es el de identificar, por ejemplo: el primer actor será ACT-01.
- **Nombre:** nombre del componente.
- **Descripción:** descripción del componente.
- **Prioridad:** la prioridad que se le da la realización de un componente.
- **Esfuerzo:** lo que cuesta realizar dicho componente.

3.1 Identificación de actores

Esta parte del análisis se centra en los actores que utilizan el videojuego. En este caso, dado que solo puede haber un jugador y nadie más, solo habrá un actor.

ID y nombre	ACT-01 Jugador
Descripción	Usuario principal y encargado de jugar el juego.

Tabla 2: ACT-01 Jugador

3.2 Especificación de requisitos

En esta sección se especificarán los requisitos del videojuego. De esta manera se podrá saber las funcionalidades del sistema y la información necesaria para el conocimiento del juego.

3.2.1 Requisitos funcionales

ID y nombre	RQF-01 Movimiento del personaje
Descripción	El sistema deberá poder mover al personaje
Prioridad	Alta
Esfuerzo	Bajo

Tabla 3: RQF-01 Movimiento del personaje

ID y nombre	RQF-02 Iluminación
Descripción	El sistema debe añadir una iluminación acorde a la temática del juego.
Prioridad	Alta
Esfuerzo	Alto

Tabla 4: RQF-02 Iluminación

ID y nombre	RQF-03 Interactuar con la puerta al final de cada nivel
Descripción	Cuando se consiga derrotar al amigurumi, el jugador podrá salir de la habitación interactuando con la puerta.
Prioridad	Media
Esfuerzo	Medio

Tabla 5: RQF-03 Interactuar con la puerta al final del nivel

ID y nombre	RQF-04 Creación de las habitaciones
Descripción	Creación de los escenarios de los niveles con sus respectivos rompecabezas, decoración, elementos interactivos, etc.
Prioridad	Alta
Esfuerzo	Alto

Tabla 6: RQF-04 Creación de las habitaciones

ID y nombre	RQF-05 Cámara del bebé
Descripción	Creación de la cámara del bebé en primera persona con las animaciones correspondientes de ésta.
Prioridad	Media
Esfuerzo	Medio

Tabla 7: RQF-05 Cámara del bebé

ID y nombre	RQF-06 Barra de cordura
Descripción	Se debe mostrar una barra en la esquina superior izquierda que irá descendiendo conforme se mira al enemigo
Prioridad	Media
Esfuerzo	Alto

Tabla 8: RQF-06 Barra de cordura

ID y nombre	RQF-07 Barra de energía
Descripción	El sistema debe mostrar una barra en la esquina superior izquierda que se agota conforme el jugador corre.
Prioridad	Media
Esfuerzo	Alto

Tabla 9: RQF-07 Barra de energía

ID y nombre	RQF-08 Subir
Descripción	El sistema cuando detecte que un objeto puede ser interactuado para subir, el jugador podrá subir a ese objeto.
Prioridad	Media
Esfuerzo	Medio

Tabla 10: RQF-08 Subir

ID y nombre	RQF-09 Empujar
Descripción	El sistema cuando detecte que un objeto puede ser interactuado para empujar, el jugador podrá empujarlo.
Prioridad	Media
Esfuerzo	Medio

Tabla 11: RQF-09 Empujar

ID y nombre	RQF-10 Menú de pausa
Descripción	Cuando el jugador pulsa la tecla “Escape”, el sistema mostrará un menú de pausa en el que tendrás las opciones de salir, reanudar la partida, además de opciones como poder modificar la música, sensibilidad horizontal y vertical, etc. Cuando el jugador vuelve a pulsar la tecla “Escape”, el jugador vuelve al juego automáticamente.
Prioridad	Media
Esfuerzo	Alto

Tabla 12: RQF-10 Menú de pausa

ID y nombre	RQF-11 Inteligencia artificial del amigurumi
Descripción	El amigurumi deberá moverse por la habitación buscando al bebé, una vez que lo vea irá a por él, en caso de no verlo, pero estar a una distancia determinada, el amigurumi girará en torno a sí mismo para ver donde está. Cada amigurumi tendrá una habilidad diferente que les distinga.
Prioridad	Alta
Esfuerzo	Alto

Tabla 13: RQF-11 Inteligencia artificial del amigurumi

ID y nombre	RQF-12 Menú principal
Descripción	Menú principal que aparece una vez iniciado el juego que, permite al jugador empezar una nueva partida, continuar la que estaba en progreso, modificar aspectos del juego como la música y la sensibilidad y, por último, salir del juego. A esta escena también se puede llegar desde el menú de pausa o desde la pantalla de <i>game over</i> .
Prioridad	Media
Esfuerzo	Alto

Tabla 14: RQF-12 Menú principal

ID y nombre	RQF-13 Música
Descripción	El sistema deberá aportar una música que sea acorde a la temática del juego.
Prioridad	Alta
Esfuerzo	Bajo

Tabla 15: RQF-13 Música

ID y nombre	RQF-14 Sonidos
Descripción	El sistema deberá aportar unos sonidos que sean acorde a la temática del juego.
Prioridad	Alto
Esfuerzo	Bajo

Tabla 16: RQF-14 Sonidos

ID y nombre	RQF-15 Explosión de polillas al morir el amigurumi
Descripción	Cuando el amigurumi quede derrotado el sistema mostrará una explosión de polillas.
Prioridad	Media
Esfuerzo	Alto

Tabla 17: RQF-15 Explosión de polillas al morir el amigurumi

ID y nombre	RQF-16 Icono y texto para subir
Descripción	Cuando un objeto pueda ser utilizado para subir, el sistema deberá mostrar un texto informativo y un icono que signifique que puede subir.
Prioridad	Media
Esfuerzo	Bajo

Tabla 18: RQF-16 Icono para subir

ID y nombre	RQF-17 Icono y texto para empujar o interactuar
Descripción	Cuando un objeto pueda ser utilizado para empujar, el sistema deberá mostrar un texto informativo y un icono determinado.
Prioridad	Media
Esfuerzo	Bajo

Tabla 19: RQF-17 Icono para empujar o interactuar

ID y nombre	RQF-18 Pantalla <i>Game Over</i>
Descripción	Cuando el amigurumi te atrape mucho tiempo, te saldrá una pantalla diciéndote que has perdido. Habrá una pantalla diferente de <i>game over</i> en el caso de que te atrape un amigurumi diferente.
Prioridad	Media
Esfuerzo	Medio

Tabla 20: RQF-18 Pantalla *Game Over*

ID y nombre	RQF-19 Cinemática enfocando el objetivo del nivel 1
Descripción	Antes de empezar el nivel, habrá una cinemática que te enseñe el objetivo al cual tienes que llegar para poder activarlo y de esta manera poder pasarte el nivel. Esto entra en el nivel 1 de forma de tutorial, en los demás niveles no aparece ya que es el jugador quien tiene que investigar cual es el objetivo en cada nivel.
Prioridad	Baja
Esfuerzo	Bajo

Tabla 21: RQF-19 Cinemática enfocando el objetivo del nivel 1

ID y nombre	RQF-20 Bocadillos para ir contando la historia
Descripción	En el inicio de algunas escenas irán apareciendo algunos bocadillos o burbujas que vayan contando la historia.
Prioridad	Baja
Esfuerzo	Alto

Tabla 22: RQF-20 Bocadillos para ir contando la historia

ID y nombre	RQF-21 Nieblina
Descripción	Se mostrará una niebla negra que impida ver la habitación y que solo se vea lo que tenga el bebé a corto rango.
Prioridad	Baja
Esfuerzo	Bajo

Tabla 23: RQF-21 Nieblina

ID y nombre	RQF-22 Transiciones
Descripción	Entre las escenas habrá una transición suave de desvanecimiento.
Prioridad	Media
Esfuerzo	Medio

Tabla 24: RQF-22 Transiciones

ID y nombre	RQF-23 Perder
Descripción	Cuando el amigurumi te atrapa o te quedas mirándolo durante mucho tiempo, pierdes.
Prioridad	Alta
Esfuerzo	Alto

Tabla 25: RQF-23 Perder

ID y nombre	RQF-24 Salir del juego
Descripción	En el menú principal se podrá salir del juego.
Prioridad	Baja
Esfuerzo	Bajo

Tabla 26: RQF-24 Salir del juego

ID y nombre	RQF-25 Guardar partida
Descripción	El sistema autoguardará el progreso del juego al inicio de cada escena.
Prioridad	Media
Esfuerzo	Alto

Tabla 27: RQF-25 Guardar partida

ID y nombre	RQF-26 Gestionar música y sonidos
Descripción	Dependiendo de la situación o del momento la música tendrá que cambiar o que haya algún sonido determinado.
Prioridad	Alta
Esfuerzo	Alto

Tabla 28: RQF-26 Gestionar música y sonidos

ID y nombre	RQF-27 Continuar partida guardada
Descripción	Una vez que se tenga una partida guardada, el jugador tendrá la opción de continuar la partida que esté en progreso.
Prioridad	Media
Esfuerzo	Alto

Tabla 29: RQF-27 Continuar partida guardada

ID y nombre	RQF-28 Generar archivo de guardado
Descripción	Tanto para las opciones y para el recorrido del jugador habrá que generar un archivo en el que se almacenará dichos datos para que, aunque el jugador cambie de nivel o incluso apague el ordenador, el progreso de la partida y las opciones queden guardadas.
Prioridad	Alta
Esfuerzo	Alto

Tabla 30: RQF-28 Generar archivo de guardado

ID y nombre	RQF-29 Linterna
Descripción	Para facilitar la visibilidad, el jugador podrá utilizar una especie de sombrero infantil con una linterna.
Prioridad	Baja
Esfuerzo	Bajo

Tabla 31: RQF-29 Linterna

ID y nombre	RQF-30 Sustos
Descripción	Para que la experiencia sea lo más terrorífica posible, se realizarán sustos con imágenes o videos que aparecen al ir por una zona determinada. También habrá sustos de objetos que se mueven como puertas que se cierran, amigurumis corriendo, escondiéndose para salir después para darte un susto. Asimismo, también se jugará con los sonidos incrementando la tensión del ambiente.
Prioridad	Alta
Esfuerzo	Alto

Tabla 32: RQF-30 Sustos

ID y nombre	RQF-31 Cursor
Descripción	El videojuego dispondrá de un cursor especial para el juego.
Prioridad	Baja
Esfuerzo	Bajo

Tabla 33: RQF-31 Cursor

ID y nombre	RQF-32 Animación en el caso de que te pille un amigurumi
Descripción	Cuando el amigurumi te coja, se iniciará una animación en la que todo se vuelve muy oscuro y el amigurumi te coge.
Prioridad	Media
Esfuerzo	Alto

Tabla 34: RQF-32 Animación en el caso de que te pille un amigurumi

ID y nombre	RQF-33 Animación para cuando el jugador derrote al amigurumi
Descripción	Cuando consigas derrotar a cada amigurumi, se iniciará una animación en la que se verá como cae explota el amigurumi y se convierte en un enjambre de polillas que se dispersan por toda la habitación.
Prioridad	Media
Esfuerzo	Alto

Tabla 35: RQF-33 Animación para cuando el jugador derrote al amigurumi

ID y nombre	RQF-34 Bloques de bebé
Descripción	En cada escena habrá uno o dos bloques de letras que el jugador tendrá que obtener para poder completar el nivel.
Prioridad	Media
Esfuerzo	Medio

Tabla 36: RQF-34 Bloques de bebé

ID y nombre	RQF-35 Ganar
Descripción	Una vez llegado a la última escena y haber completado todos los niveles, el jugador podrá completar el juego.
Prioridad	Media
Esfuerzo	Medio

Tabla 37: RQF-35 Ganar

3.2.2 Requisitos no funcionales

ID y nombre	RQNF-01 Almacenamiento mínimo
Descripción	El sistema necesitará 2,48 GB mínimo de almacenamiento.
Prioridad	Alta

Tabla 38: RQNF-01 Almacenamiento mínimo

ID y nombre	RQNF-02 Respuesta rápida
Descripción	El sistema deberá mostrar una respuesta rápida a las transiciones dadas entre niveles y jugabilidad.
Prioridad	Alta

Tabla 39: RQNF-02 Respuesta rápida

ID y nombre	RQNF-03 Compatibilidad con todos los sistemas operativos
Descripción	El juego podrá ser jugado en Linux, Mac OS y Windows.
Prioridad	Alta

Tabla 40: RQNF-03 Compatibilidad con todos los sistemas operativos

ID y nombre	RQNF-04 Idioma
Descripción	El videojuego será en inglés para favorecer la internacionalización del producto.
Prioridad	Media

Tabla 41: RQNF-04 Idioma

ID y nombre	RQNF-05 Disponer de un ordenador y de un ratón
Descripción	El videojuego está realizado para que sea jugado en ordenador. Además, es recomendable tener un ratón para que la experiencia del juego sea lo más idónea.
Prioridad	Alta

Tabla 42: RQNF-05 Disponer de un ordenador

3.3 Casos de uso

En la sección de casos de uso se podrá ver el comportamiento y las interacciones entre el sistema y el actor, que en este caso solo hay uno, el jugador. Todo esto será representado con un diagrama de casos de uso UML. Acto seguido se procederá a la especificación de dichos casos de uso.

3.3.1 Diagrama de casos de uso

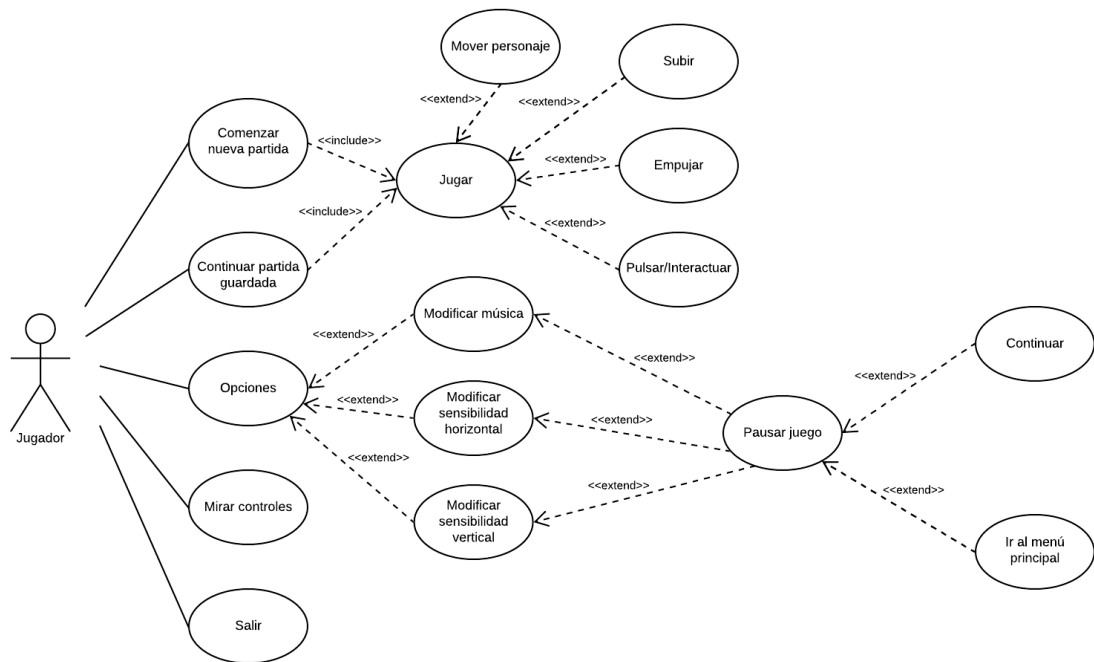


Diagrama 1: Diagrama de casos de uso de Amigurumi

3.3.2 Especificación de los casos de uso

En este caso se añaden los campos de la precondición, frecuencia de uso y el destinado a saber cómo se llega a ese caso de uso.

ID y nombre	CU-01 Comenzar nueva partida
Descripción	Permite empezar el juego de nuevo.
¿Como activarlo?	Dándole al botón de <i>Start new game</i>
Prioridad	Media
Frecuencia de uso	Alta
Precondición	El jugador debe encontrarse en el menú principal.

Tabla 43: CU-01 Comenzar nueva partida

ID y nombre	CU-02 Continuar partida guardada
Descripción	Permite seguir con la partida que estaba en progreso.
¿Como activarlo?	Dándole al botón de <i>Continue</i>
Prioridad	Alta
Frecuencia de uso	Alta
Precondición	El jugador debe encontrarse en el menú principal.

Tabla 44: CU-02 Continuar partida guardada

ID y nombre	CU-03 Opciones
Descripción	Permite entrar en el menú de opciones.
¿Como activarlo?	Dándole al botón de <i>Settings</i>
Prioridad	Media
Frecuencia de uso	Baja
Precondición	El jugador debe encontrarse en el menú principal.

Tabla 45: CU-03 Opciones

ID y nombre	CU-04 Mirar controles
Descripción	Permite entrar en una pantalla donde se muestran los controles del videojuego.
¿Como activarlo?	Dándole al botón de <i>Controls settings</i>
Prioridad	Baja
Frecuencia de uso	Media
Precondición	El jugador debe encontrarse en el menú principal.

Tabla 46: CU-04 Mirar controles

ID y nombre	CU-05 Salir
Descripción	Permite salir del videojuego.
¿Como activarlo?	Dándole al botón de <i>Exit</i>
Prioridad	Baja
Frecuencia de uso	Media
Precondición	El jugador debe encontrarse en el menú principal.

Tabla 47: CU-05 Salir

ID y nombre	CU-06 Jugar
Descripción	Se muestra ya la pantalla donde el jugador puede jugar.
¿Como activarlo?	O bien desde una partida nueva o desde una partida guardada.
Prioridad	Alta
Frecuencia de uso	Alta
Precondición	

Tabla 48: CU-06 Jugar

ID y nombre	CU-07 Modificar música
Descripción	Gestionar el volumen de la música.
¿Como activarlo?	Modificando el <i>slider</i> para subir o bajar la música.
Prioridad	Media
Frecuencia de uso	Media
Precondición	Estar en el menú de opciones.

Tabla 49: CU-07 Modificar música

ID y nombre	CU-08 Modificar sensibilidad horizontal
Descripción	Gestionar la sensibilidad horizontal del ratón.
¿Como activarlo?	Modificando el <i>slider</i> para subir o bajar la sensibilidad horizontal.
Prioridad	Media
Frecuencia de uso	Media
Precondición	Estar en el menú de opciones.

Tabla 50: CU-08 Modificar sensibilidad horizontal

ID y nombre	CU-09 Modificar sensibilidad vertical
Descripción	Gestionar la sensibilidad vertical del ratón.
¿Como activarlo?	Modificando el <i>slider</i> para subir o bajar la sensibilidad vertical.
Prioridad	Media
Frecuencia de uso	Media
Precondición	Estar en el menú de opciones.

Tabla 51: CU-09 Modificar sensibilidad vertical

ID y nombre	CU-10 Subir
Descripción	El jugador subirá encima de un objeto
¿Como activarlo?	Dándole a la tecla “Espacio” una vez que salga el icono en pantalla de que a ese objeto se puede subir.
Prioridad	Alta
Frecuencia de uso	Alta
Precondición	Encontrarse enfrente de un objeto al que se pueda subir.

Tabla 52: CU-10 Subir

ID y nombre	CU-11 Empujar
Descripción	El jugador empuja un objeto.
¿Como activarlo?	Una vez que salga el icono de interactuar con un objeto, se le dará a la tecla “E” junto con algunas de las teclas de dirección “W, A, S, D” para empujar el objeto a esa dirección.
Prioridad	Alta
Frecuencia de uso	Media
Precondición	Encontrarse enfrente de un objeto al que se pueda empujar.

Tabla 53: CU-11 Empujar

ID y nombre	CU-12 Pulsar/Interactuar
Descripción	El jugador pulsa o interactúa con un objeto.
¿Como activarlo?	Una vez que salga el icono de que un objeto puede ser interactuado, se pulsa la tecla “E”.
Prioridad	Alta
Frecuencia de uso	Alta
Precondición	Encontrarse enfrente de un objeto con el que se pueda interactuar.

Tabla 54: CU-12 Pulsar/Interactuar

ID y nombre	CU-13 Pausar juego
Descripción	El jugador pausa el juego.
¿Como activarlo?	Dándole al botón “Escape”.
Prioridad	Alta
Frecuencia de uso	Media
Precondición	

Tabla 55: CU-13 Pausar juego

ID y nombre	CU-14 Mover personaje
Descripción	El jugador mueve al personaje.
¿Como activarlo?	Utilizando las teclas “W, A, S y D” para la dirección del movimiento.
Prioridad	Alta
Frecuencia de uso	Alta
Precondición	

Tabla 56: CU-14 Mover personaje

ID y nombre	CU-15 Continuar
Descripción	El jugador reanuda el juego.
¿Como activarlo?	Dándole al botón <i>Continue</i> .
Prioridad	Media
Frecuencia de uso	Media
Precondición	Estar en el menú de pausa.

Tabla 57: CU-15 Continuar

ID y nombre	CU-16 Ir al menú principal
Descripción	El jugador vuelve al menú principal.
¿Como activarlo?	Dándole al botón <i>Exit</i> .
Prioridad	Media
Frecuencia de uso	Media
Precondición	Estar en el menú de pausa.

Tabla 58: CU-16 Ir al menú principal

3.4 Identificación y análisis de soluciones posibles

Aclarados los requisitos especificados anteriormente, éstos se mantendrían en todas las soluciones que se han barajado. Las únicas dudas que se produjeron fueron en el cómo conseguir llegar al final del videojuego, es decir, que coleccionar cada nivel o que hacer para que se pueda abrir la puerta de la última habitación.

Dos días antes de la entrega final, se tuvo la idea de que la forma de ir avanzando de nivel a nivel sería con el hecho de que los amigurumis fuesen dejando un hilo de lana por su paso que está conectado con un ovillo de lana que a su vez está conectado con la puerta. La clave es que cuando el amigurumi muere, el hilo hace que tire de la puerta y, de esta manera se abra y puedas pasar al siguiente nivel.

Al faltar matices que le diesen sentido a esta idea porque no se entendía muy bien el porqué de que cuando muriese el amigurumi, la puerta se abriese y que el amigurumi tuviese un hilo que era casi infinito.

A esto había que sumarle que su acabado dentro del videojuego no era el mejor ni mucho menos ya que, además de no lucir bien, llenaba el escenario de un hilo de lana el cuál había que hacer que fuese pasable porque si no el jugador se quedaba encerrado. A continuación, se enseñarán algunas imágenes que corroboran estas palabras.

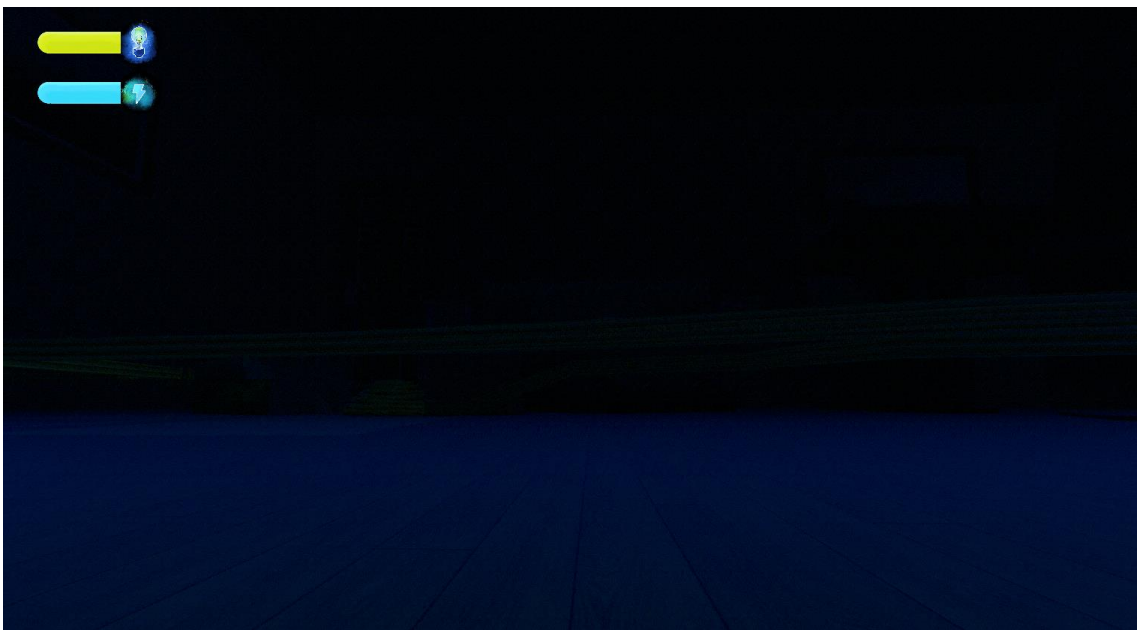


Figura 15: Imagen del hilo de lana dentro del videojuego

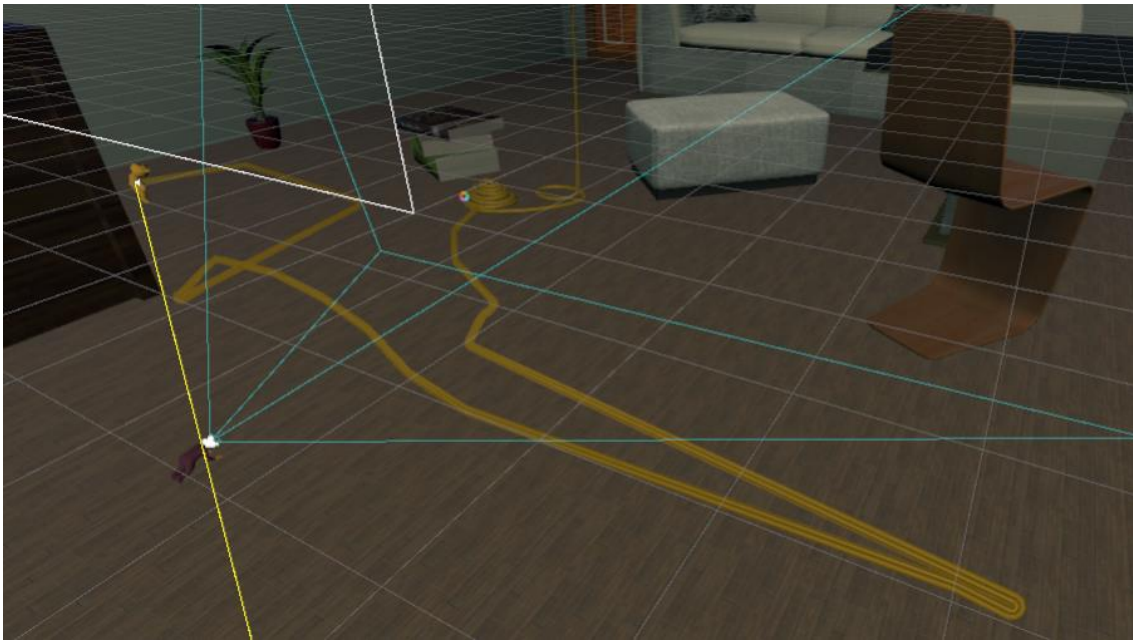


Figura 16: Rastro que deja el hilo de lana dentro del videojuego

Por lo tanto, esta opción fue desechada a los pocos días de continuar el proyecto porque hacía menos estético el videojuego y el argumento no tenía todo el sentido que gustaría que tuviese.

Entonces, aparecieron dos ideas acerca de cómo avanzar por los niveles. Una técnica muy utilizada es la de recolectar trozos de una fotografía para que cuando esté completa poder pasar al final del videojuego. En este caso, la fotografía sería de los padres ya que es a la habitación de los padres a la que hay que llegar para finalizar el videojuego. Dicho esto, al ser tan utilizada esta técnica en videojuegos de miedo y no querer seguir con la misma línea, también fue desechada.

Por último, como el protagonista de este videojuego es un bebé, se pensó que lo mejor sería recolectar bloques de letras para bebés, para que una vez recogido todos y la llave poder hacer una escalera con estos bloques para llegar a la cerradura de la puerta y poder abrirla.

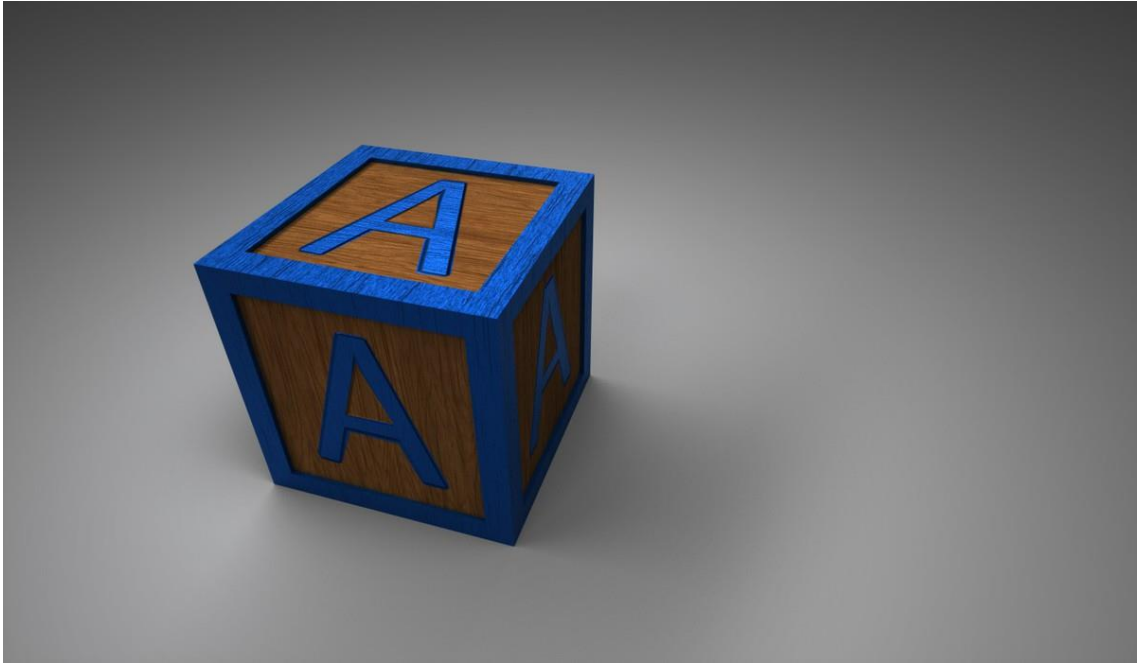


Figura 17: Bloques de letras utilizados en el videojuego

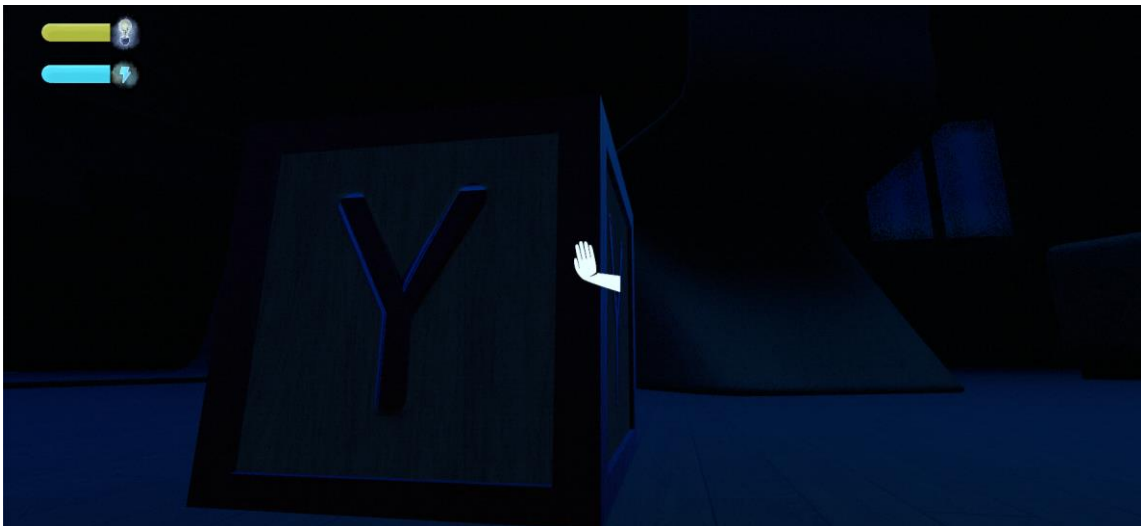


Figura 18: Bloque de letra del primer nivel

3.5 Solución propuesta

En este apartado se hablará de la propuesta final del videojuego, como se va a realizar, las fases en las que se va a llevar a cabo, etc. El videojuego estará compuesto por 5 niveles en los que el jugador tiene como objetivo principal llegar a una fuente de luz o un objeto que pueda dar luz como puede ser una lámpara o un microondas para que, una vez activada la luz, las polillas que están dentro del amigurumi y que hacen que éstos se muevan, saldrán hacia la luz dejando al amigurumi inmóvil y, de esta manera poder completar cada nivel.

En todos los niveles no será sencillo llegar hacia este objeto, pues en cada nivel habrá una especie de rompecabezas o algún conjunto de mecanismos que activar para poder llegar hacia el objetivo. Estos rompecabezas pueden ir desde sacar cajones para subirte a ellos y poder alcanzar superficies más altas o cosas más complicadas como superar un laberinto o introducir un código.

Para que el resultado de todo esto sea el correcto, hay que llevar a cabo determinadas fases. Primero, llega el análisis del videojuego, se hace una sesión de *brainstorming* para sacar ideas de todo tipo, una vez que se hayan sacado las ideas, hay que centrarse en que es lo óptimo para el videojuego dados los recursos disponibles y las limitaciones tanto del programa Unity como del propio programador.

Hecho esto, se pasa a la fase del diseño del videojuego, donde se realiza todo lo relacionado con el diseño del propio videojuego en sí como viene a ser la realización de un menú principal con sus opciones, opción de mirar los controles del videojuego, un menú de pausa, los controles del jugador, los tipos de amigurumi que habrá, que amigurumi habrá en cada nivel, que movimientos o habilidades tendrá cada amigurumi, como estarán organizados los niveles, como será cada escenario, que recorrido debe realizar el jugador para poder completar el nivel, que objetos pueden ser interactuados o no, etc.

Una vez que el diseño del videojuego está claro, hay que pasar al apartado de implementación y programación, dónde se procederá a programar todas las funcionalidades del nivel además de la inteligencia artificial de los amigurumis. También hay que programar la interfaz, es decir, además de lo que se ve en el escenario, es necesario que haya textos informativos o iconos que ayuden al jugador a que la jugabilidad sea más sencilla.

Asimismo, cada vez que una funcionalidad sea programada, se harán sus respectivas pruebas para comprobar que ha sido correctamente implementada y que el comportamiento dentro del videojuego es el esperado. Para finalizar, se documentará todo en el documento de trabajo de fin de grado.

En el siguiente apartado se comentará en profundidad el plan de trabajo que se ha llevado durante estas fases.

3.6 Plan de trabajo

Ahora es el momento de hablar de las técnicas de planificación que se han seguido y la estimación del esfuerzo del desarrollador del videojuego, así como su valor equivalente en el presupuesto. Además, a este presupuesto basado en el esfuerzo realizado, también se comentará los gastos reales realizados en el proyecto.

Como se introdujo en el apartado 1.5, la metodología empleada es ágil, en concreto se optará por la estrategia Kanban. La estrategia Kanban o también conocida como “Tarjeta Visual” consiste en la elaboración de una especie de diagrama/tabla compuesta de una serie de columnas, en mi caso 5 columnas: pendientes, en proceso, pruebas, terminadas y opcional. Como bien dicen el nombre de las columnas, la

primera será utilizada para aquellas tareas pendientes de realizar en el videojuego, la segunda es para las tareas que se están realizando actualmente y una vez acabadas se pasan a la columna de pruebas donde se comprueba que la tarea está correcta y muestra el comportamiento adecuado. Por último, están las columnas de “Terminadas” donde se dejan todas las tareas acabadas y, la columna de opcional es para ideas que van surgiendo con el desarrollo del videojuego y que pueden ser implementadas en caso de que haya tiempo.

Se eligió esta estrategia porque es muy útil ya que permite una buena planificación de tareas, es la más visual porque no deja de ser una tabla con las tareas y la organización de los *sprints* es sencilla porque tras la realización de un número determinado de tareas la entrega es instante, lo que permite trabajar de una manera continua.

3.7 Presupuesto

Respecto al tema del presupuesto se va a dividir en varios apartados, primero se comentará el presupuesto relacionado al material hardware y software utilizados en el proyecto, así como el presupuesto equivalente a los recursos humanos utilizados.

3.7.1 Presupuesto hardware

El hardware utilizado en este proyecto está compuesto de los siguientes componentes:

- **Ordenador:** se necesita un ordenador para poder hacer el videojuego. Si un ordenador tiene una vida media de 4-5 años y este ordenador ha sido utilizado para este proyecto durante 6 meses, se hace la siguiente operación:

100% ----- 60 meses

X ----- 6 meses

$$X = (6 * 100) / 60 = 10\% \text{ de uso}$$

- **Conexión a Internet:** no necesario para el uso del programa encargado de hacer el videojuego, ni para el uso del programa para el apartado de programación, pero importante para la obtención de recursos tales como imágenes, música, sonidos, diseños, etc. Si la conexión a Internet, en este caso de Vodafone está contratada por 10 meses y se ha utilizado un total de 6 meses, se hace la siguiente operación:

100% ----- 10 meses

X ----- 6 meses

$$X = (6 * 100) / 10 = 60\% \text{ de uso}$$



Por lo tanto, hechos los cálculos de los porcentajes de uso de cada material, es hora de relacionarlo con el coste correspondiente.

Componente	% Uso	Coste	(%Uso*Coste)/100
Ordenador	10%	1169,6 €	116,96 €
Conexión de 120 Mb	60%	254,4 €	152,64 €
TOTAL			269,6 €

Tabla 59: Presupuesto hardware

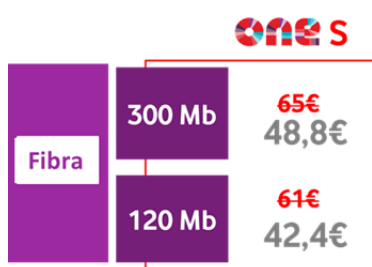


Figura 19: Conexión de Vodafone



Figura 20: Ordenador utilizado

3.7.2 Presupuesto software

El software utilizado en este proyecto está compuesto de los siguientes componentes:

- **Unity:** programa utilizado para la realización del videojuego, al ser gratuito no se realizarán cálculos.
- **Visual Studio 2017:** programa utilizado para la programación del videojuego, al ser gratuito tampoco se realizarán cálculos.
- **Asset Store:** tienda de la que se han sacado recursos para el videojuego, ya fuesen de pago o gratuitos.

Componente	% Uso	Coste	(%Uso*Coste)/100
Unity	-	-	-
Visual Studio 2017	-	-	-
Asset Store	100%	15,95 €	15,95 €
TOTAL			15,95 €

Tabla 60: Presupuesto software

3.7.3 Presupuesto recursos humanos

A continuación, se verán todos los roles que hay dentro de este videojuego, cuanto ha sido el esfuerzo de cada uno y el coste correspondiente a ese trabajo. Dicho esto, de los siguientes cinco roles, una persona ha ocupado cuatro de ellos. Estos son los roles:

- **Analista:** persona encargada de la fase de análisis del videojuego.
- **Diseñador:** persona encargada del diseño de la interfaz, de los escenarios, de la historia y de las funcionalidades del videojuego.
- **Programador:** persona encargada de implementar todas las funcionalidades y de sus correspondientes pruebas.
- **Animador:** persona encargada de realizar los diseños de los amigurumis, modularlos para un entorno 3D y darle animación.
- **Documentador:** persona encargada de documentar el videojuego.

Una vez aclarado los roles hay que ver su aportación en horas y el coste que corresponde, para ello primero habrá que calcular el total de horas que requerirá la realización del videojuego. Para realizar el cálculo del total de horas, se ha trabajado durante 6 meses una media de 8 horas al día descansando 1 día a la semana. A todo esto, hay que sumarle las horas de trabajo que ha realizado el animador contratado para el videojuego que han sido 24 horas totales repartidas en dos semanas.

Analista, diseñador, programador y documentador

$$144 \text{ días} * 8 \text{ horas} = 1152 \text{ horas}$$

$$\text{Total} = 1152 \text{ horas} + 24 \text{ horas (Animador)} = 1176 \text{ horas}$$

En la siguiente tabla se hará el reparto de las horas entre los diferentes roles del videojuego y su coste por hora para poder realizar el coste total del presupuesto de recursos humanos.

Rol	Tiempo	Coste
Analista	176,4 horas	29 € / hora
Diseñador	294 horas	30 € / hora
Programador	588 horas	32 € / hora
Animador	24 horas	25 € / hora
Documentador	93,6 horas	23 € / hora
TOTAL		35504,4 €

Tabla 61: Presupuesto recursos humanos

3.7.4 Presupuesto total

A continuación, se realizará la suma total de todos los presupuestos calculados previamente:

Presupuesto	Coste
Hardware	269,6 €
Software	15,95 €
Recursos humanos	35504,4 €
TOTAL	35789,95 €

Tabla 62: Presupuesto total

Capítulo 4

Diseño de la solución

En este capítulo se hablará del diseño del videojuego. Estará compuesto de dos partes, la arquitectura del sistema y se detallará el diseño en profundidad. Además, se comentarán cosas acerca de la historia, estética del juego, interfaz del usuario, etc.

4.1 Arquitectura del sistema

Este apartado se dividirá en varias secciones ya que dentro de la arquitectura se puede separar la arquitectura software y la arquitectura hardware. Además, se realizarán unos diagramas de clase de los componentes de la solución.

4.1.1 Arquitectura software

La arquitectura software o también llamada arquitectura lógica es el diseño de más alto nivel del sistema y está compuesto por las siguientes capas:

- **Capa de presentación:** es la que interactúa directamente con el usuario por lo tanto se encuentran componentes como vienen a ser la interfaz del usuario (escenarios, menús).
- **Capa de negocio:** contiene los archivos de programación o *scripts* para intercambiar datos entre esta capa y la capa de presentación.
- **Capa de componentes comunes:** es aquella capa que contiene estos componentes a los que puedes acceder desde cualquier parte del videojuego como vienen a ser la configuración del sonido.

4.1.2 Arquitectura hardware

En este proyecto, solamente aparece el ordenador y el ratón como elementos de hardware ya que no se requiere de Internet para su jugabilidad ni tampoco de una base de datos para guardar a los usuarios o algo del estilo.

4.2 Diagramas de clases

Esta sección se enfocará en los diagramas UML de las clases de los componentes que forman parte del videojuego. Hay que aclarar que cuando un *script* es creado en Unity, extiende de `MonoBehaviour`, clase que contiene variables, objetos y funciones que se utilizan normalmente la programación de videojuegos en Unity. Aquellos que no heredan de esta clase será porque no la necesita. Los diagramas se separarán en:

- **Controlador del videojuego:** contiene las clases encargadas del funcionamiento del juego en sí.
- **Inteligencia artificial:** clases relacionadas con la inteligencia artificial de los amigurumis (enemigos del juego).
- **Personaje del juego:** clases relacionadas con el personaje del juego.
- **UI:** clases relacionadas con la interfaz del usuario, menú principal, menú de pausa, etc.
- **Objetos del juego:** contiene las clases acerca de los objetos que pueden ser interactuados por el jugador.
- **Sustos:** son aquellas clases que tienen que ver con la aparición de sustos durante el juego.
- **Cargar y guardar:** contiene las clases relacionadas con el sistema de guardado de opciones y de la partida.

Controlador del videojuego

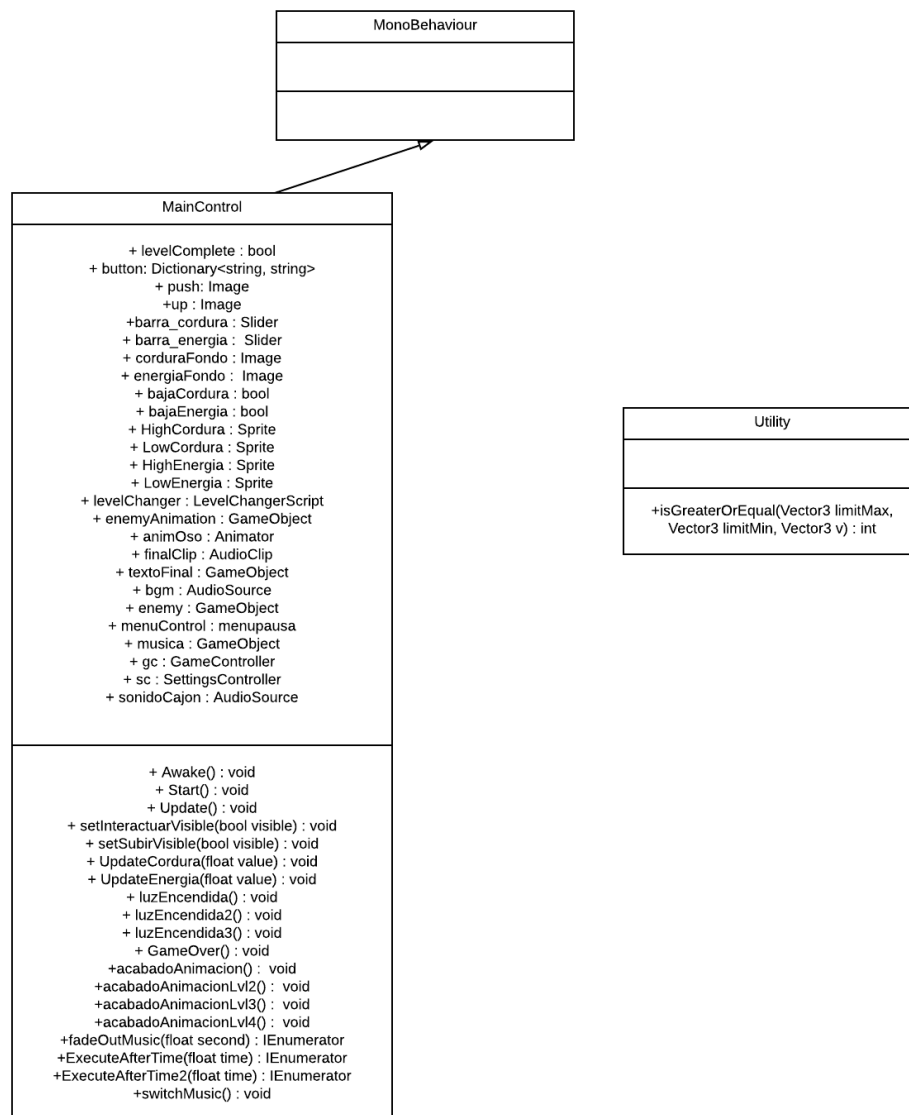


Diagrama 2: Diagrama de clases del controlador del videojuego

En el caso del controlador del videojuego tenemos dos clases, Utility es utilizada para controlar si un cajón del juego está sacado totalmente o no y MainControl es utilizado para gestionar los elementos que aparecen dentro del juego y dependiendo de lo que pase actualizarlos o no. Más tarde se detallará todo en el punto 5.2.1 de la memoria. ([Ver MainControl](#))

Inteligencia artificial

En este caso se va a separar el diagrama de clases en partes para que la visión sea más adecuada al documento. Dicho esto, la primera parte será la principal parte de la inteligencia artificial, el esqueleto y que permite variaciones de otras inteligencias artificiales de los amigurumis.

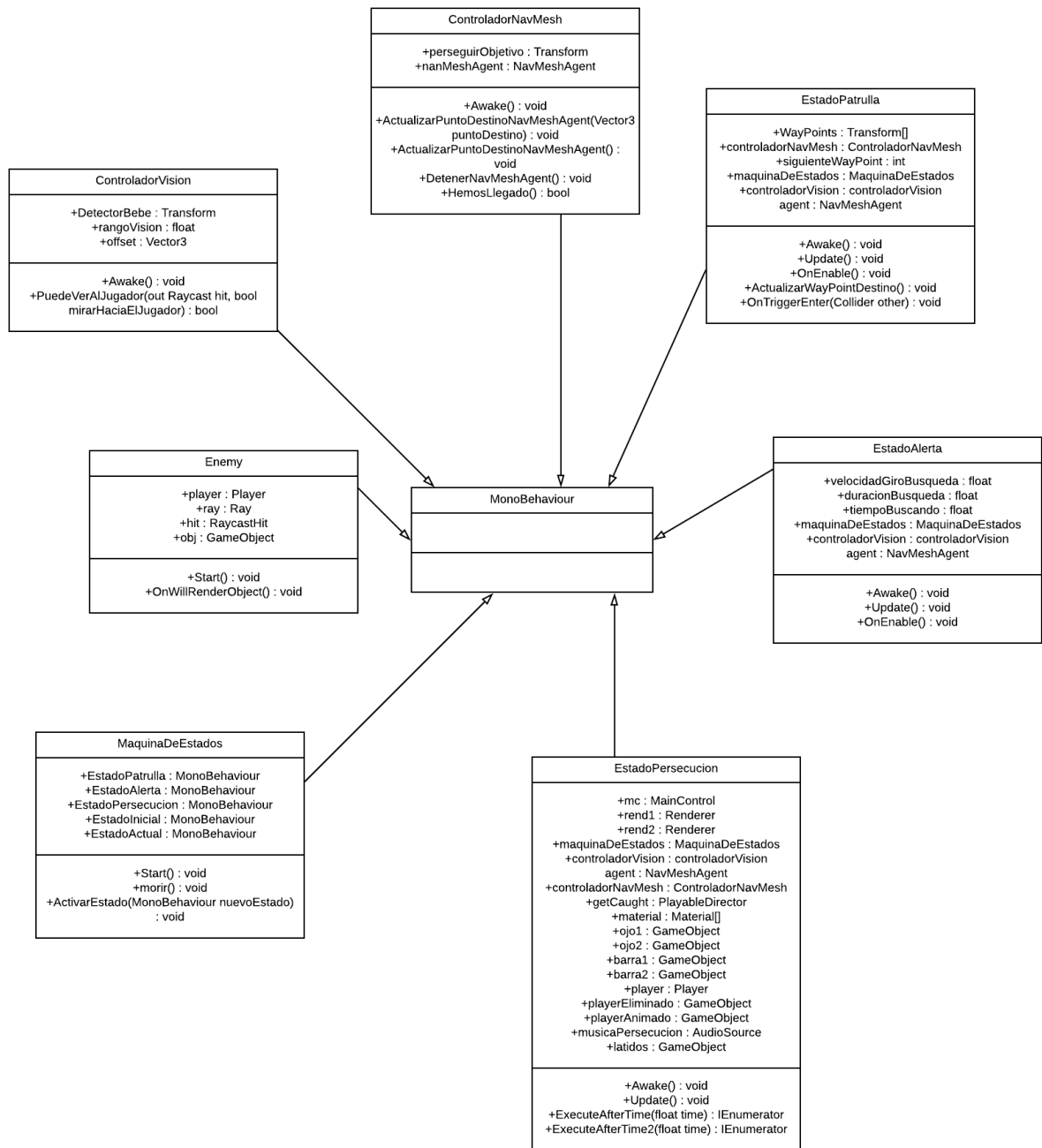


Diagrama 3: Diagrama de clases principal de la inteligencia artificial



Dentro de la parte de la inteligencia artificial están los controladores que, como su nombre indica se encargan de controlar la inteligencia artificial y que se utilizarán para cambiar entre los estados que hay en el diagrama. El controlador de visión se utilizará para saber lo que ve el amigurumi en todo momento y el otro controlador será utilizado para su movimiento dentro del videojuego. Por último, está Enemy que es utilizado para que si el jugador mira al amigurumi entonces pierda vida. Más tarde se detallará todo en el punto 5.2.4 de este documento ([Ver inteligencia artificial](#))

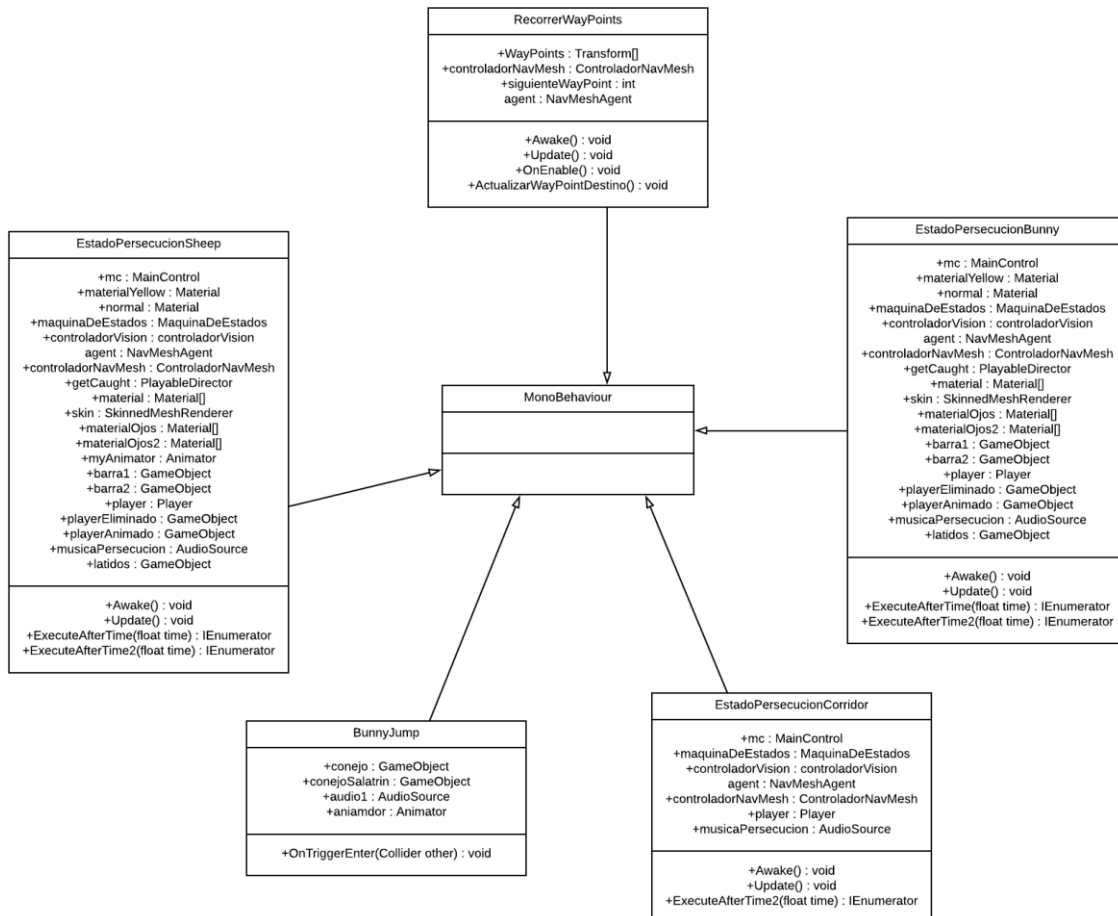


Diagrama 4: Diagrama de clases de la inteligencia artificial del resto de amigurumis

Dentro de la inteligencia artificial se dan variaciones entre los amigurumis ya que no todos se comportan de la misma manera, aunque compartan algunas características. Por ejemplo, el conejo empieza saltando (BunnyJump) hasta que se encuentra con el jugador y pasa a su estado de persecución. También depende del lugar en el que se encuentre se comportará de otra manera distinta como es el caso del estado persecución del pasillo (EstadoPersecucionCorridor)

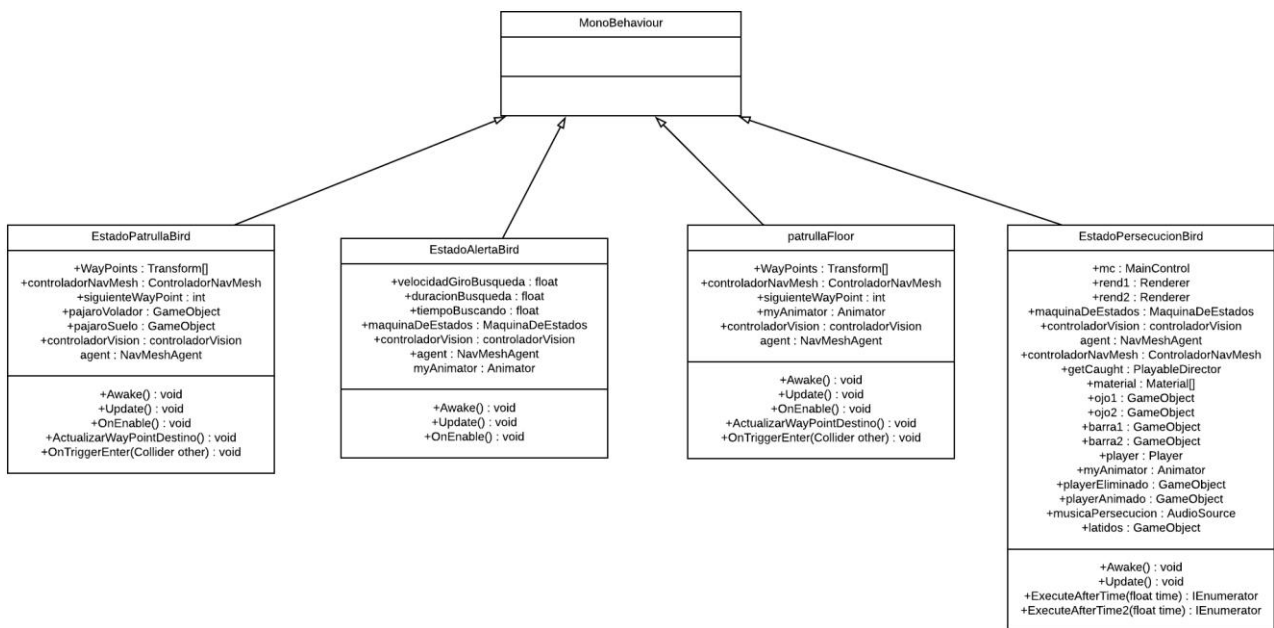


Diagrama 5: Diagrama de clases de la inteligencia artificial del amigurumi pájaro

Para el pájaro también ocurre lo mismo hablado anteriormente, los estados a pesar de seguir un mismo esquema tienen varias diferencias. Dentro de la clase de MonoBehaviour se utilizarán casi siempre en todas las clases los métodos de Update() que sirve para actualizar lo que la clase va haciendo una vez iniciado el juego o el método Awake() que sirve para inicializar algo al inicio del juego.



Personaje del juego

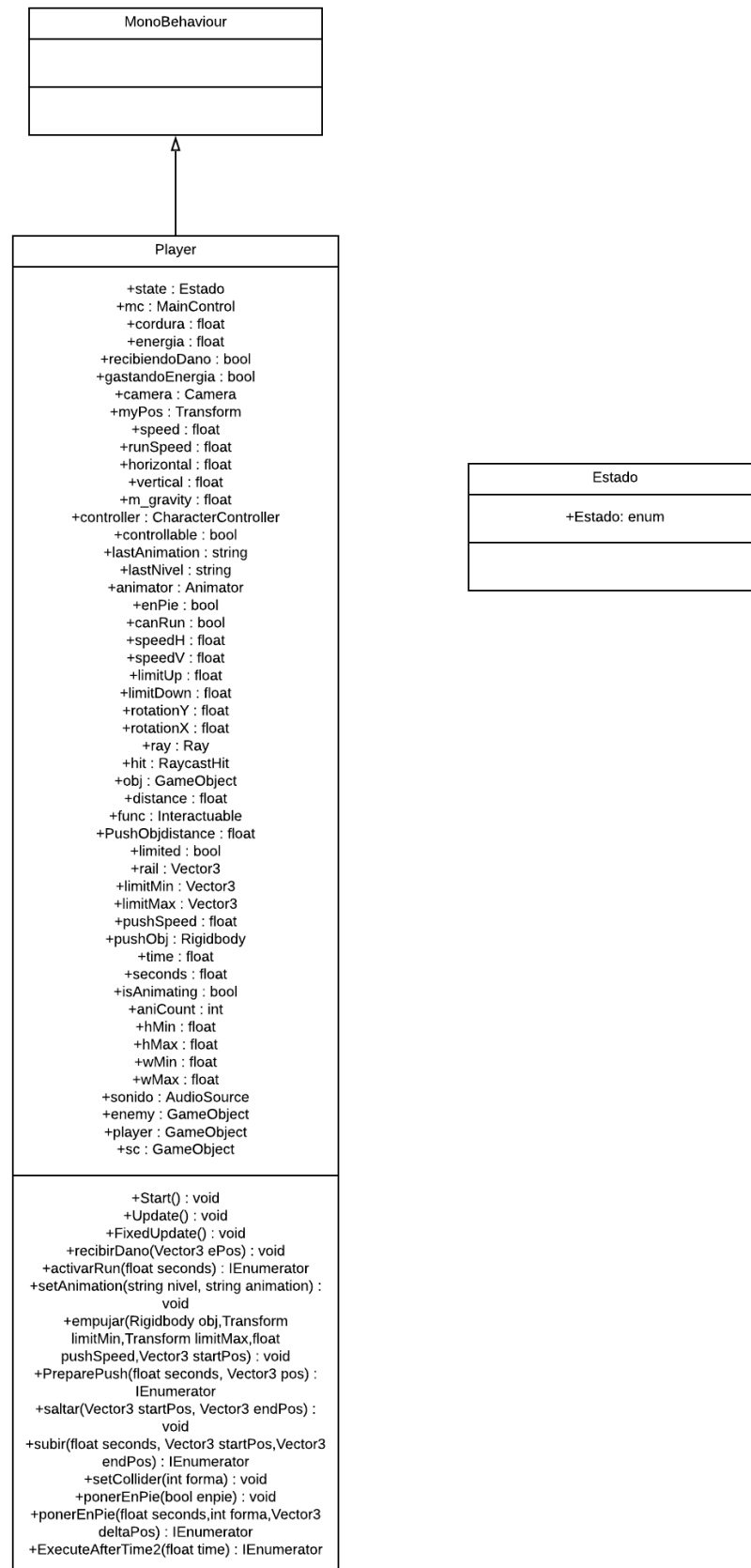


Diagrama 6: Diagrama de clases del personaje del juego

Al igual que en el controlador del juego, aquí también hay dos clases, un enumerador de los estados del jugador, que no necesita de MonoBehaviour y la clase Player destinada al movimiento, acciones, animaciones y demás eventos asociados al jugador. Por ejemplo, en Player se lleva a cabo el sistema para detectar si un objeto puede ser utilizado y las acciones que se pueden realizar con dicho objeto. Más detalles en el apartado 5.2.2 ([Ver player](#))

Objetos del videojuego

Al igual que en la parte de la inteligencia artificial, debido al gran número de clases, se irá separando en partes por temas. En este caso se dispone de la interfaz Interactuabile que tiene 3 métodos que se utilizarán en todos los objetos. Dos métodos son para comprobar si un objeto puede ser utilizado o subido y el otro para decir lo que sucede una vez que se usa el objeto. En casi todos los objetos, una vez usados, se activarán y desactivarán *gameObjects*, que eran los objetos del juego, de ahí la abundancia de este tipo de variables en las clases.

En todos los niveles habrá una clase encargada del objeto que da luz para acabar cada nivel, puede ser el caso de lampara en el nivel 1 o de TvRemote en el nivel 3 y que dan paso a la animación final de cada nivel.

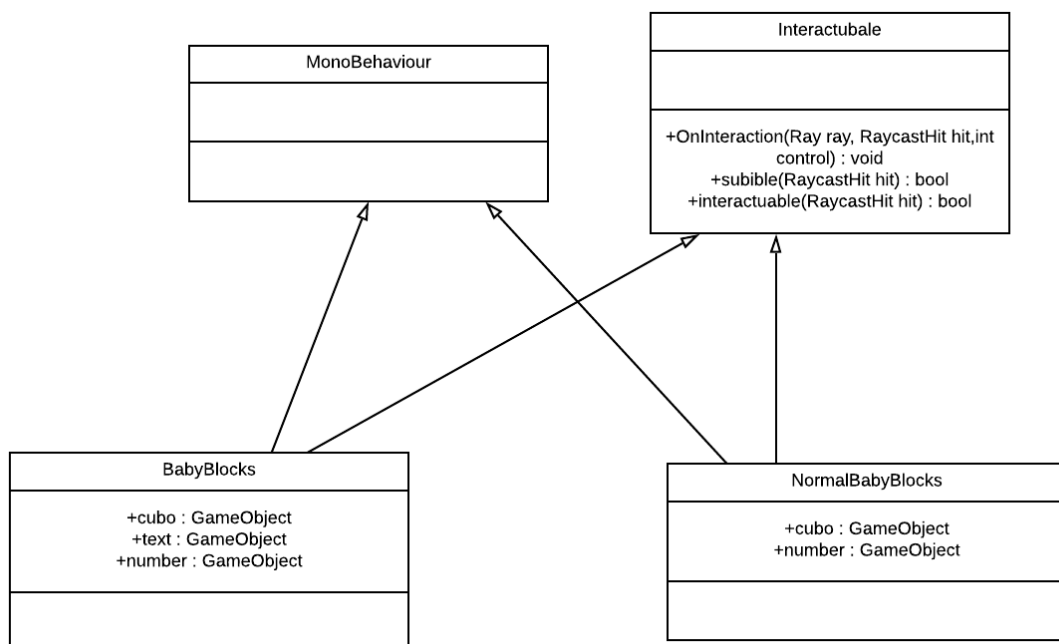


Diagrama 7: Diagrama de clases de los bloques recolectables del bebé

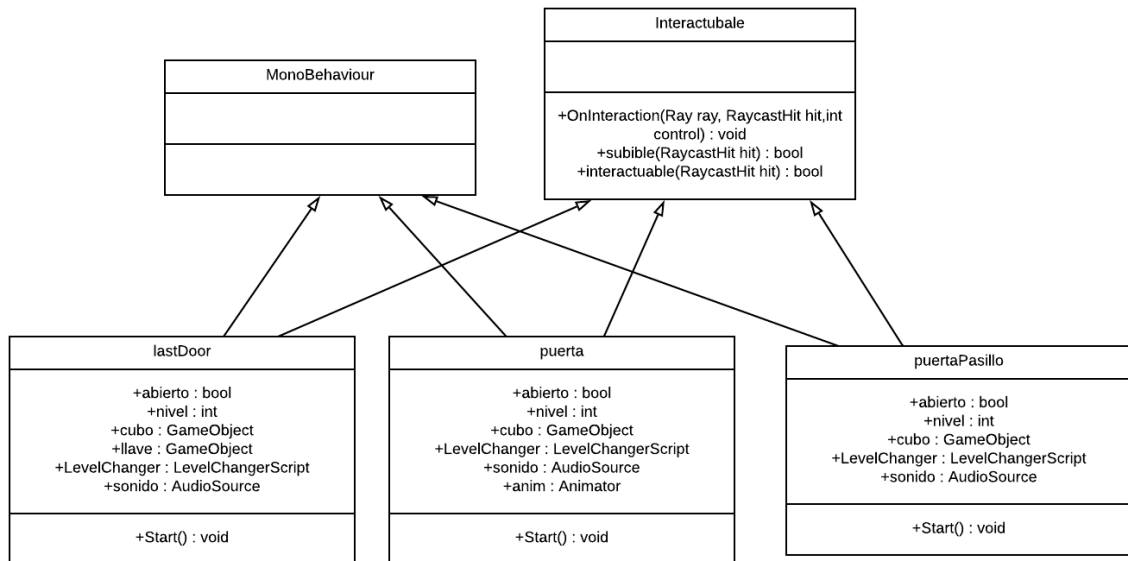


Diagrama 8: Diagrama de clases de las puertas con las que se interactúan

En este caso, para que se puedan utilizar las puertas siempre hay que recoger un cubo y completar el nivel. Activan un sonido de abrir puerta y en la mayoría de las ocasiones permiten al jugador cambiar de escena. Heredan de **Interactuabile** porque se utilizan para cambiar de nivel, pero esto no es posible hasta que el nivel es completado. Para ello se llama a la variable `levelComplete` de `MainControl`.

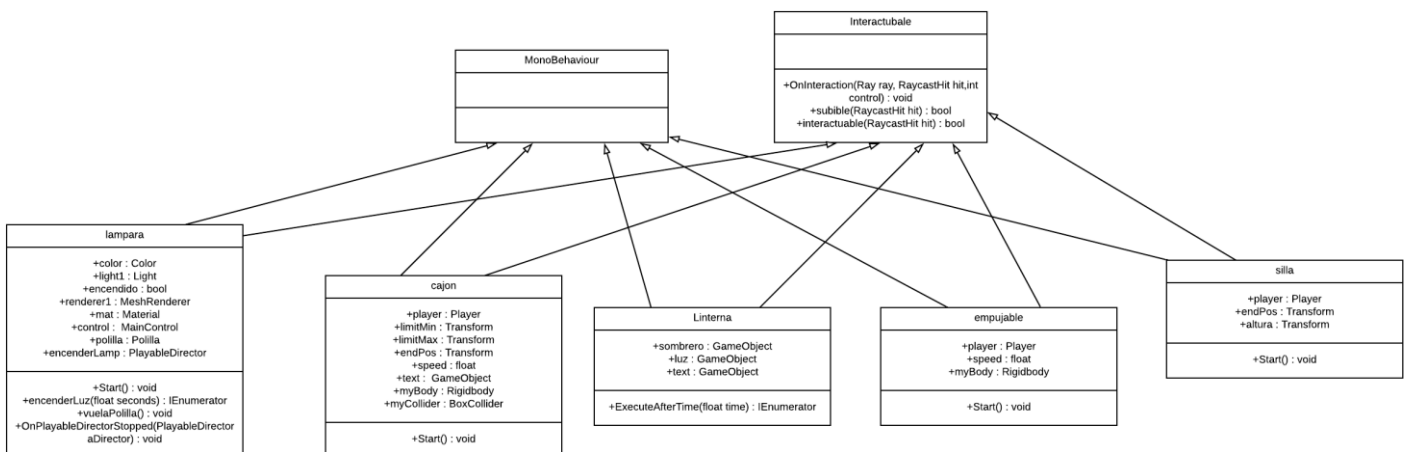


Diagrama 9: Diagrama de clases de los objetos del nivel 1

En este nivel se introduce entre otras muchas la clase **empujable** que sirve para empujar objetos como es el caso de la silla de la habitación. También se encuentra la clase del **cajón** para poder interactuar con ellos y una vez sacados poder subirse a ellos. Hay que destacar también la clase de **lampara** que sirve para activar este objeto y terminar el nivel 1. Muchas de estas clases son utilizadas como base para la creación de otras muchas en los siguientes niveles.

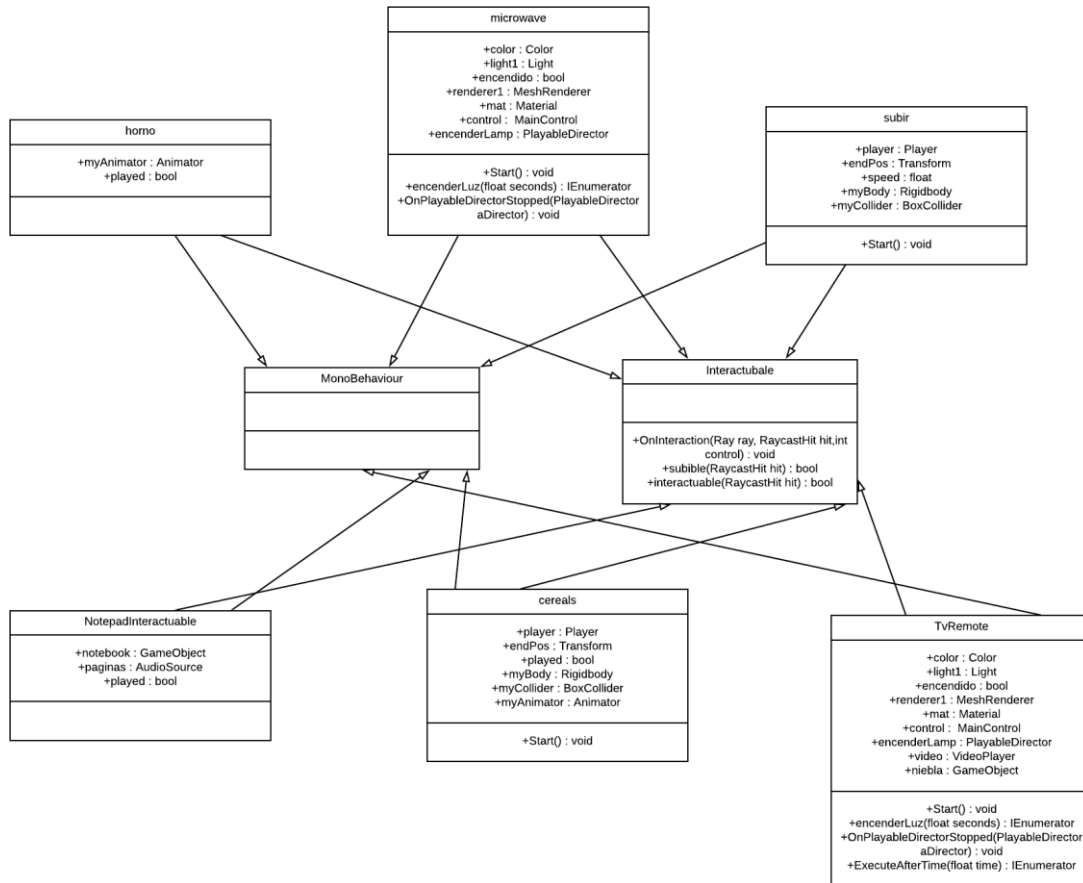


Diagrama 10: Diagrama de clases de los objetos de los niveles 2 y 3

En este caso las clases de microwave y de TvRemote son las utilizadas para finalizar los niveles 2 y 3 respectivamente. Son las clases encargadas de dar luz al objeto referente y provocar la cinemática final de cada nivel.

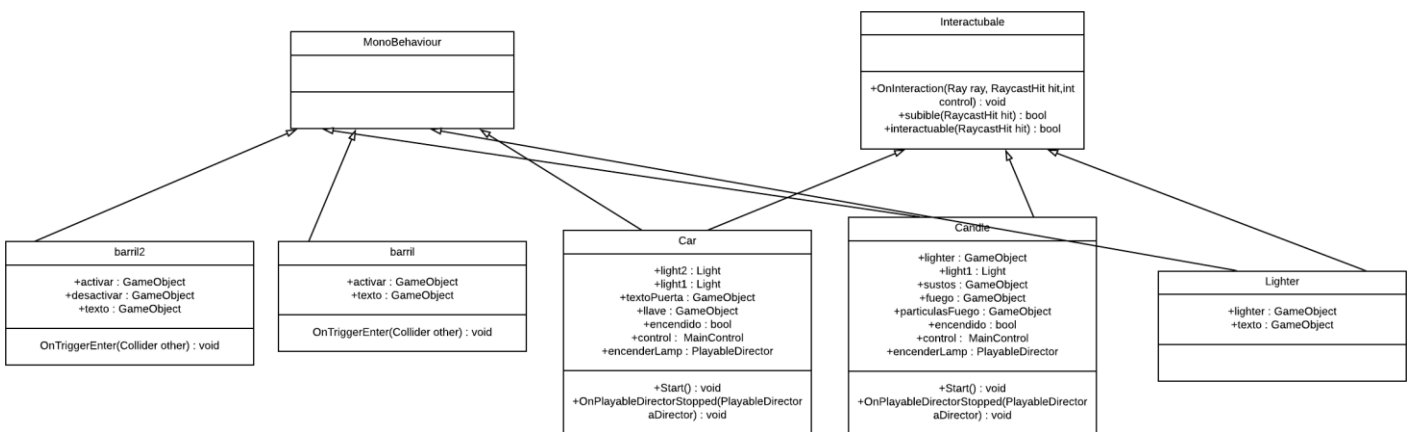


Diagrama 11: Diagrama de clases de los objetos de los niveles 4 y 5

Para el nivel 5 se utilizarán barriles que podrán moverse y que contendrán pistas para resolver los rompecabezas. Por último destacar que en el nivel 4 se necesitará de un mechero (Lighter) para poder encender la vela(Candle) que finalizará dicho nivel.



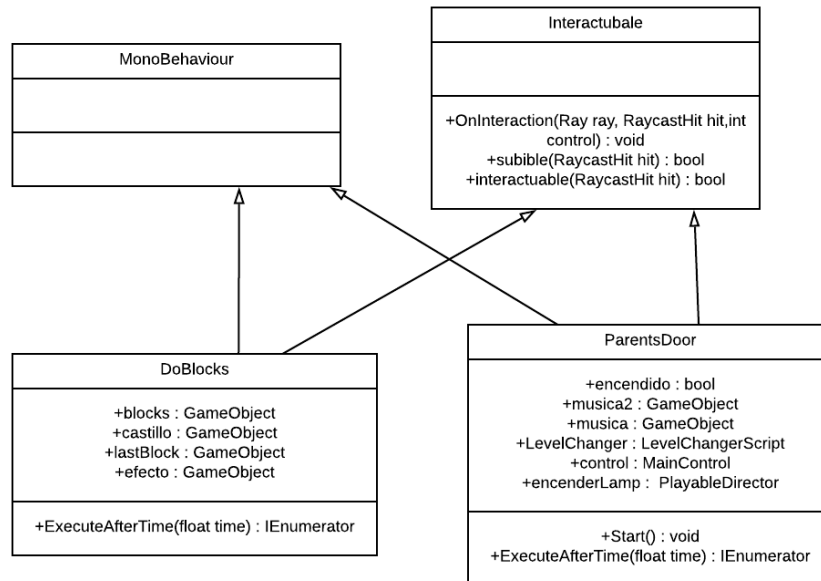


Diagrama 12: Diagrama de clases de los objetos de la última escena

Para la última escena se emplea la clase DoBlocks que sirve para, una vez recogidos todos los bloques de letras del videojuego, poder montar una especie de escalera con ellos para poder llegar a la cerradura de la puerta. Finalmente se utilizará la clase ParentsDoor para abrir la puerta de los padres y finalizar el videojuego con la última cinemática.

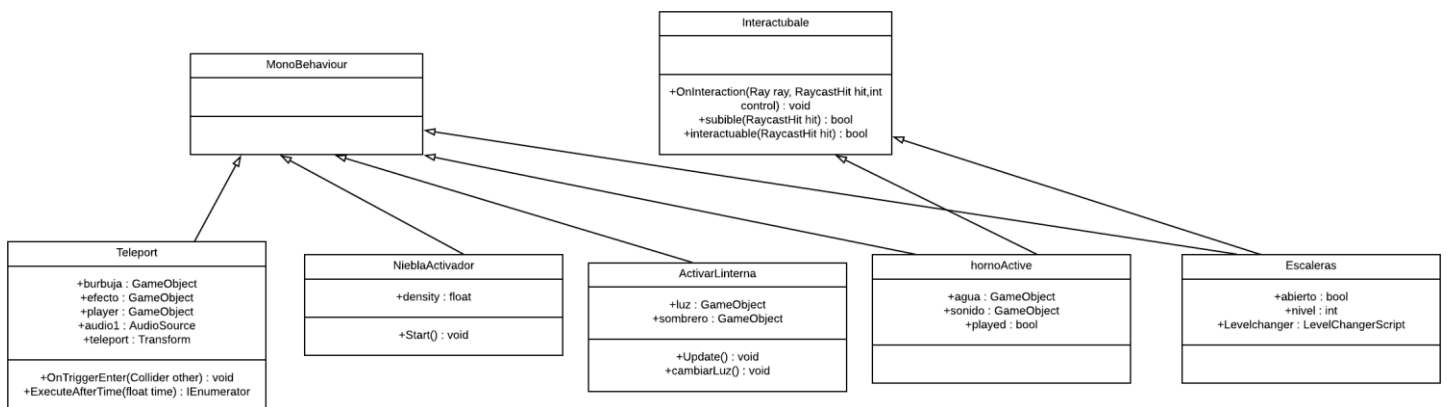


Diagrama 13: Diagrama de clases de varios objetos utilizados a lo largo de varios niveles

Para finalizar con los objetos hay que destacar alguna clase que otra como es el caso de ActivarLinterna para activar un sombrero linterna que lleva el jugador y de esta manera tener mejor visión. Además, existe una clase Teleport que sirve para teletransportar al jugador por la habitación y así poder utilizar todos los rincones de la habitación.

Interfaz del usuario (UI)

Al igual que en las anteriores se separarán en varias partes para que la visión sea mejor. Como se ha dicho ya varias veces, hay una gran cantidad de *gameObjects* que se van a activar o desactivar, en este caso en la interfaz del usuario, bien para informar mediante textos, para adornar o mejorar la interfaz o para el caso del menú de pausa y su funcionamiento.

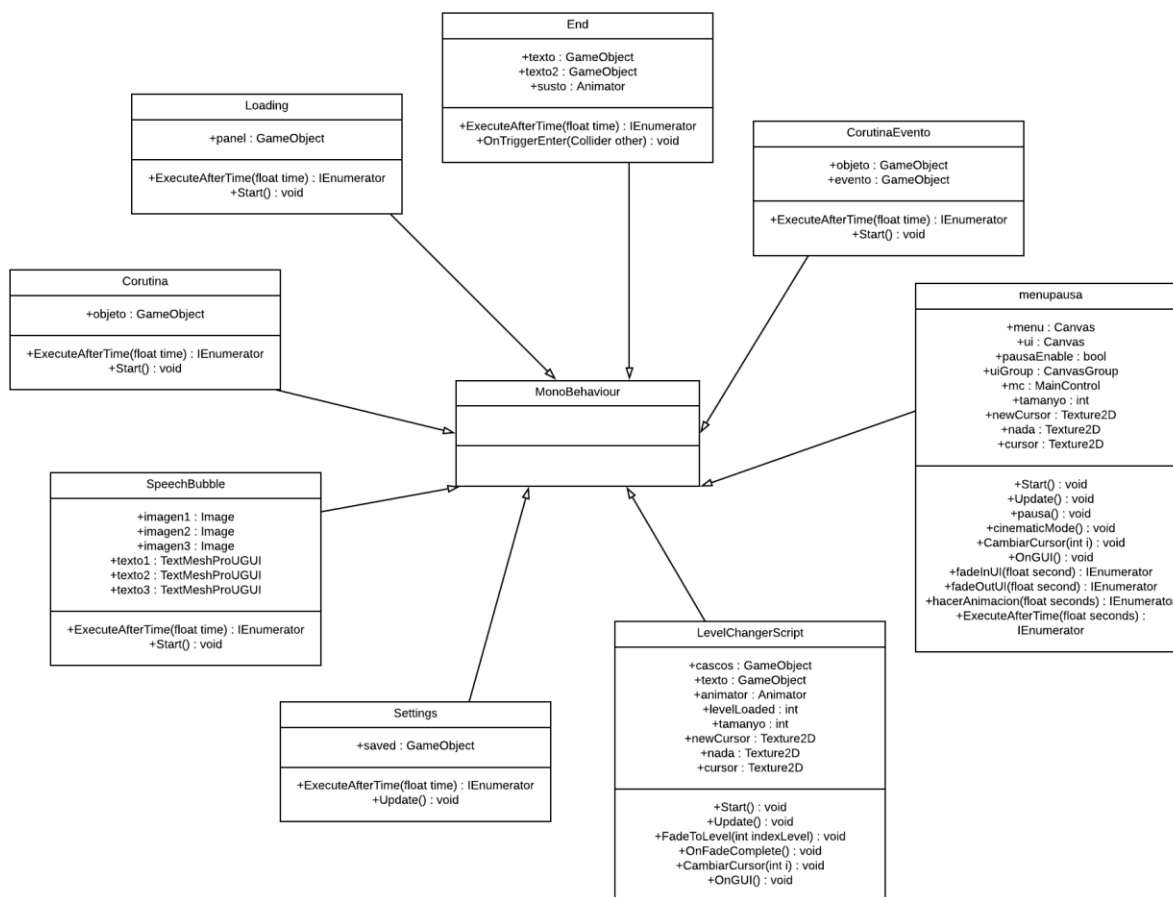


Diagrama 14: Diagrama de clases relacionadas con la interfaz del usuario

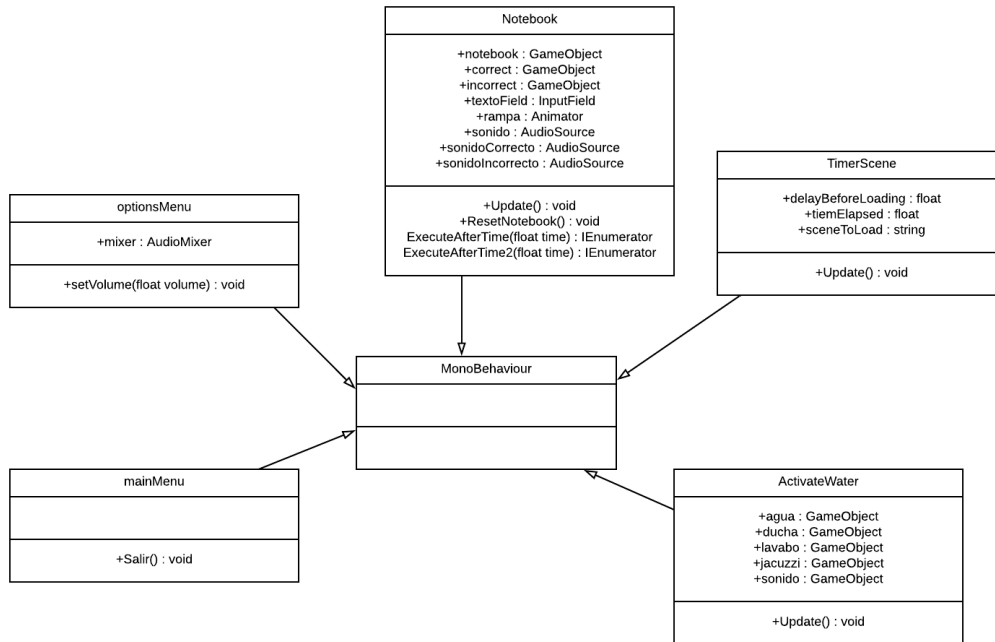


Diagrama 15: Segundo diagrama de clases relacionadas con la interfaz del usuario

De aquí se puede destacar la clase Notebook que es una libreta que al utilizarse, aparece en pantalla una imagen de una hoja en la que habrá que acertar un código de 4 números para poder desbloquear eventos dentro de los niveles y de esta manera avanzar en el transcurso del videojuego.

Cargar y guardar

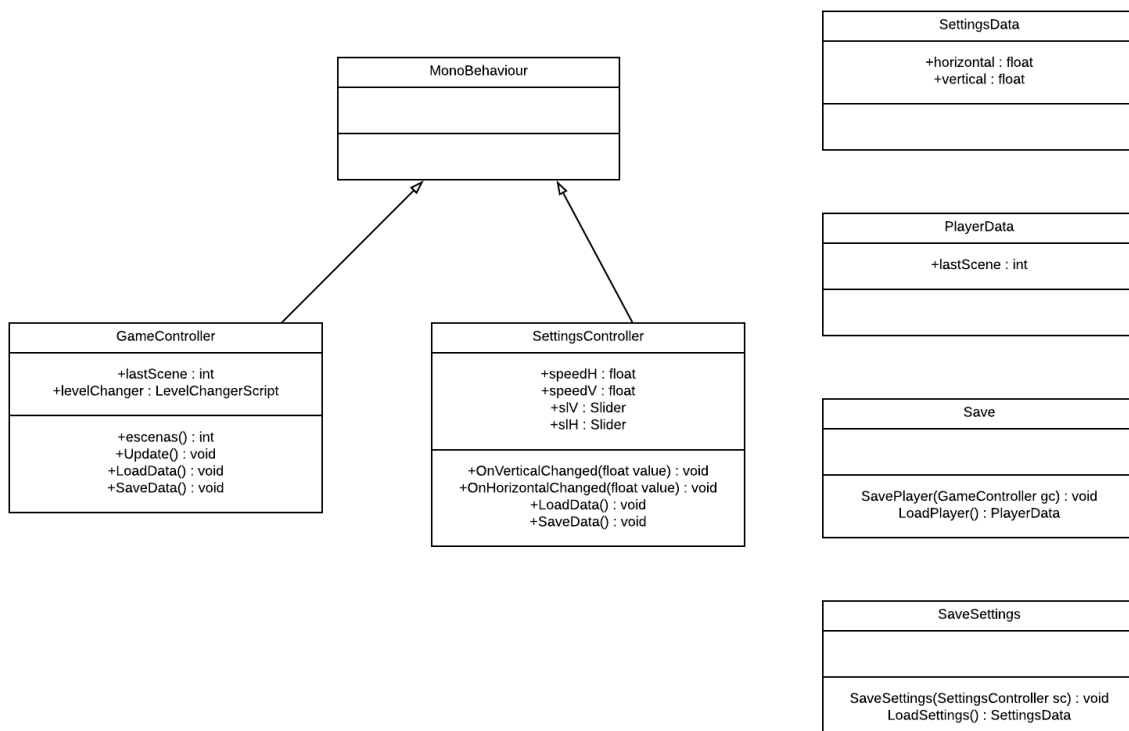


Diagrama 16: Diagrama de clases relacionadas con la función de cargar y guardar el videojuego

Respecto al tema de guardar y cargar el progreso de la partida y sus opciones, hay que hacer unas clases encargadas de controlar los datos que se tienen como pueden ser `PlayerData` o `SettingsData`. Luego otras clases para guardar y cargar estos datos como son `Save` y `SaveSettings` y finalmente las otras dos clases son utilizadas para que las otras 4 clases puedan ser utilizadas dentro del videojuego de manera correcta. Para ver más detalles se puede consultar el apartado 5.2.7. ([Ver GameController y SettingsController](#))

Sustos

En este caso se separará en 2 el diagrama de clases de sustos. Estos dos diagramas de clases hacen referencia al término de videojuego de miedo *jumpscare*, que básicamente es un evento que trata de sorprender y asustar al jugador con el cambio visual repentino de la escena.

El primer diagrama está relacionado cuando el *jumpscare* está ocasionado con la aparición de algún o varios amigurumis, la segunda con objetos, imágenes y vídeos. El `SustoSonido` es utilizado para que una vez que se pase por una zona, suene un sonido determinado, otro como el `SustoDetras` trata de que el amigurumi se teletransporta a la espalda del jugador, etc.

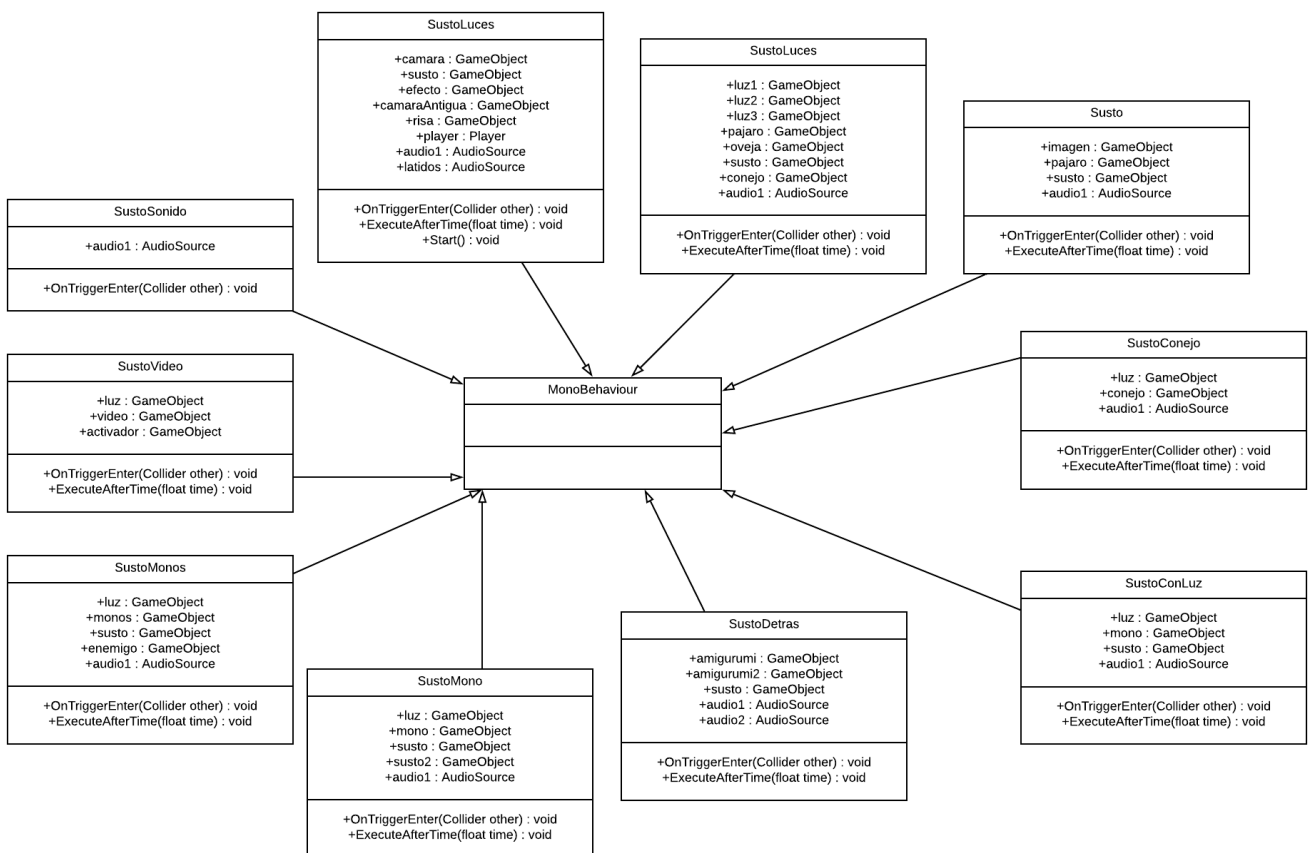


Diagrama 17: Diagrama de clases relacionadas con los sustos ocasionados por amigurumis



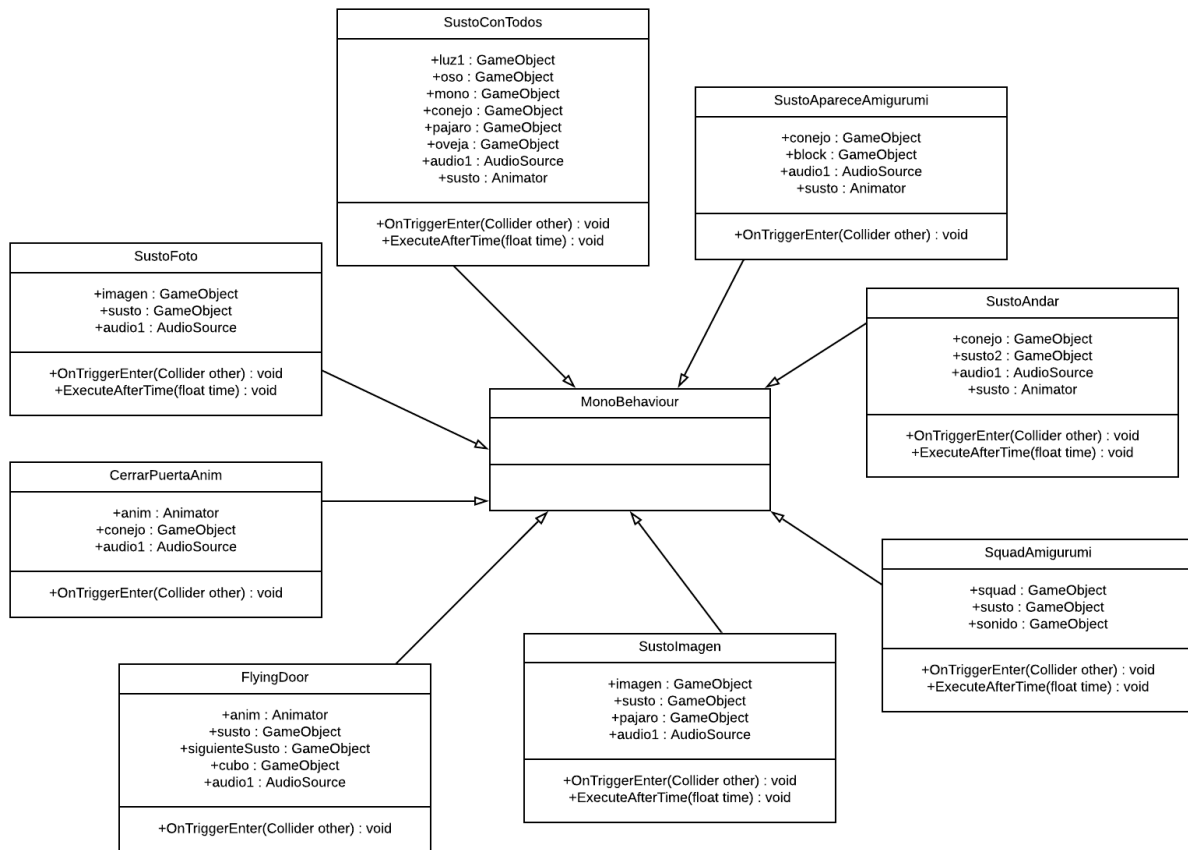


Diagrama 18: Diagrama de clases relacionadas con los sustos ocasionados por objetos, imágenes o vídeos

De esta parte es muy utilizado el SustoFoto cuya función es la de mostrar en la pantalla una foto antigua y terrorífica y, que desaparezca tras un pequeño intervalo de tiempo. También es muy utilizado el CerrarPuertaAnim ya que durante el transcurso del videojuego hay muchas puertas que están abiertas y que cuando el jugador se acerca a ellas misteriosamente son cerradas.

4.3 Diagramas de estados

A pesar de que hay muchos objetos que tienen estados como puertas que están cerradas y pasan a abrirse, en este apartado se centrará en aquellas cosas de las que merezca la pena hablar, como por ejemplo de los personajes del juego: el bebé y los amigurumis.

4.3.1 Bebé

El bebé dispone de los siguientes estados:

- **Parado (de rodillas):** Es la acción por defecto.
- **Gateando:** Es el estado principal, pues se activa cuando te mueves.

- **Parado (de pie):** Cuando se pulsa la tecla shift, pero no te mueves.
- **Corriendo:** Una vez que estas de pie, te mueves y tu velocidad es mayor a la de cuando estas gateando.
- **Empujando:** Si el jugador detecta un objeto con el que se pueda interactuar y es empujable, entonces si avanza hacia delante puede empujar el objeto.
- **Tirando:** Si el jugador detecta un objeto con el que se pueda interactuar y es empujable, entonces si avanza hacia atrás puede empujar el objeto.
- **Subiendo:** Si el jugador detecta un objeto con el que se pueda interactuar y se puede subir, entonces si utiliza la tecla C puede subir al objeto.

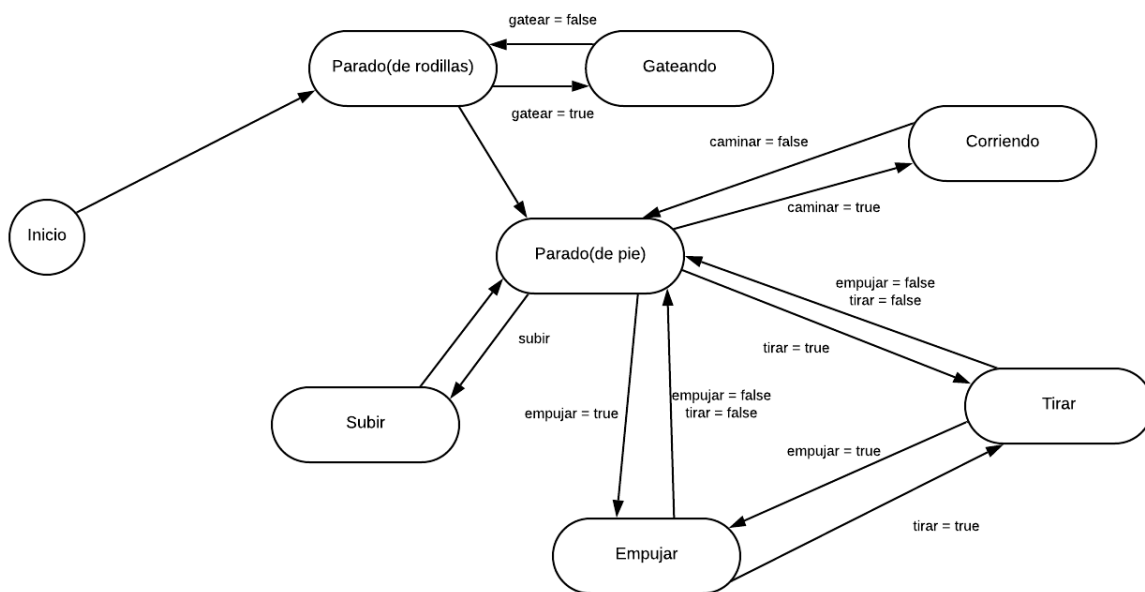


Diagrama 19: Diagrama de estados del bebé

El bebé por defecto está parado de rodillas, si se utilizan las teclas de dirección entonces pasará a gatear. Si se utiliza la tecla SHIFT, el bebé se pondrá de pie y si se mueve correrá. Dependiendo del objeto que tenga enfrente podrá subirse a dicho objeto, empujarlo o tirar de él. Para que se produzca uno de los últimos tres eventos hace falta que se usa alguna tecla especial como la C o la E.

4.3.2 Oso, oveja y mono

Antes de indagar en el funcionamiento de los amigurumis, hay que aclarar que el diagrama de estados de la inteligencia artificial de los enemigos de este videojuego está basado en el siguiente esquema. No obstante, cada amigurumi puede presentar alguna variación respecto al esquema.

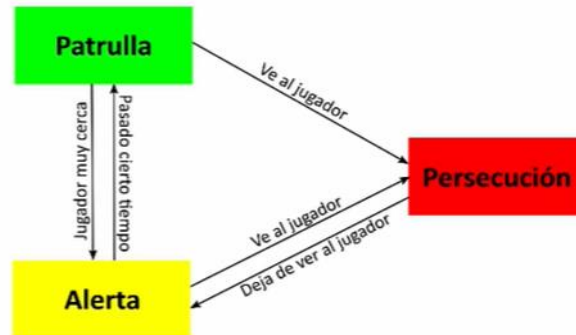


Diagrama 20: Diagrama de estados de la inteligencia artificial de los amigurumis

A continuación, se mostrará el diagrama de estados que vale tanto para el oso como para el mono y la oveja. En este caso, el amigurumi empezará a andar por toda la habitación (estado patrulla), si detecta al jugador cerca, empezará a girar cierto tiempo buscando al jugador (estado alerta). Si el amigurumi ve al jugador empezará a perseguirlo (estado persecución). En caso de perder visión del jugador volverá a girar y si no ve al jugador volverá a su patrulla. Una vez encendida la luz, el amigurumi muera.

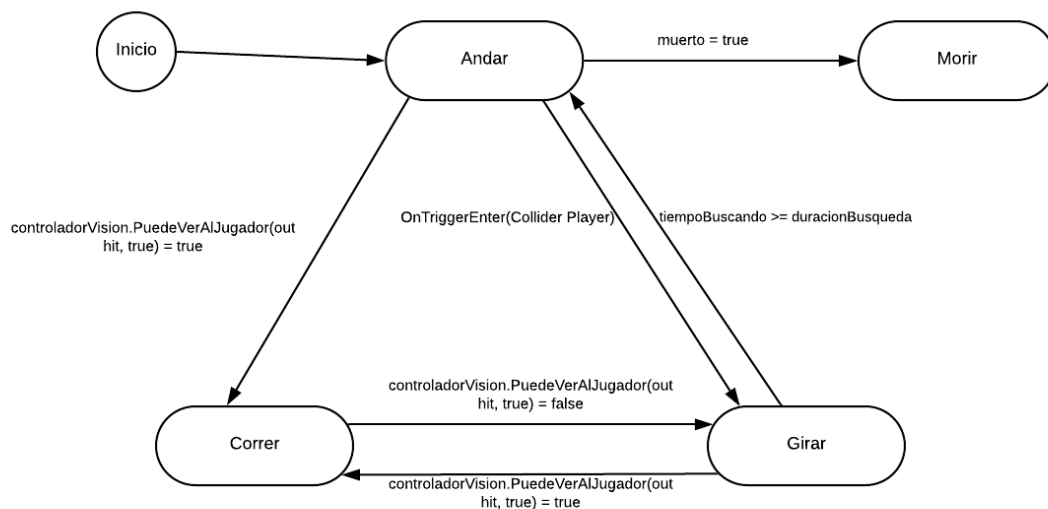


Diagrama 21: Diagrama de estados del oso, oveja y mono

4.3.3 Pájaro y conejo

Aunque sigan teniendo el mismo esquema del estado de patrulla, alerta y persecución, lo que diferencia a estos dos es que tienen dos tipos de patrulla. El pájaro vuela por toda la habitación y si detecta al jugador cerca de su posición vuela hacia un punto del suelo y procede con su búsqueda, pero lo hace andando. Una vez encendido el objeto de luz del nivel del pájaro, el pájaro morirá.

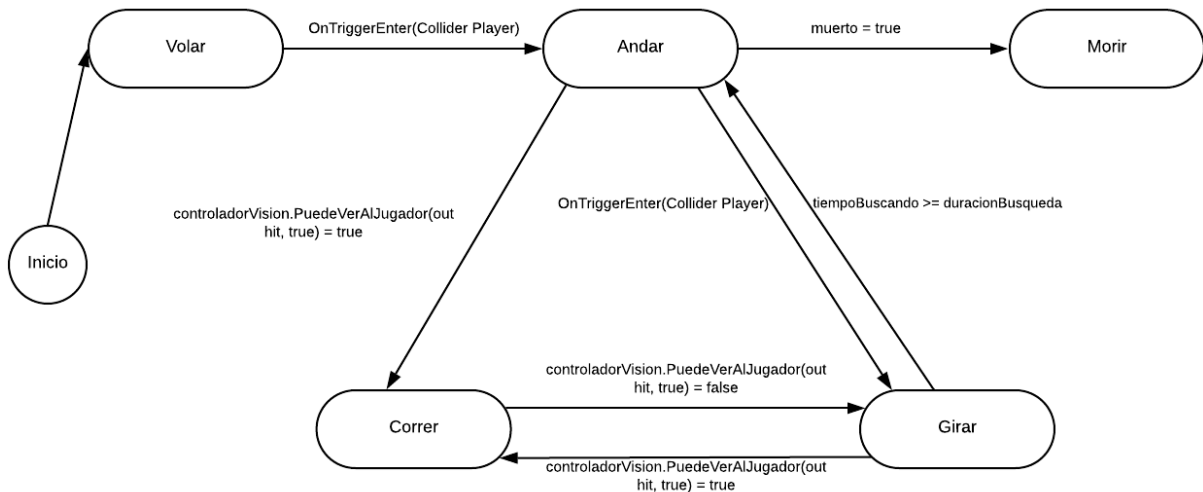


Diagrama 22: Diagrama de estados del pájaro

En cambio, el conejo va saltando por toda la habitación y en el caso de que salte cerca del jugador va directamente a por él, por lo que el jugador tendrá que estar hábil a la hora de esconderse. Una vez que ha avistado al jugador, el conejo irá corriendo a por él. En el laberinto debido a que es muy bajo el techo, el conejo va corriendo todo el rato en busca del jugador.

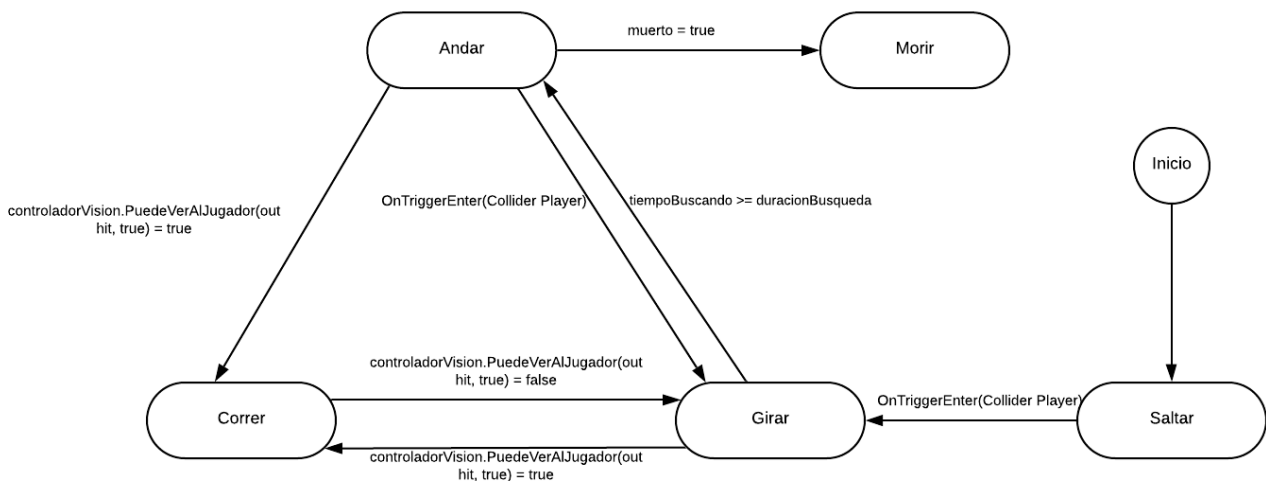


Diagrama 23: Diagrama de estados del conejo

4.4 Historia

Una familia había sido informada de que la casa sufría una plaga de polillas y hasta el siguiente día no iba a poder ser exterminada dicha plaga. A la noche el pequeño de la familia se despierta desorientado y se da cuenta de que el mundo ha aumentado de tamaño mientras que sus amigurumis con los que tantos jugaba ahora se mueven y lo persiguen. Ante esta situación el objetivo del bebé es el de ir a la habitación de sus padres para sentirse a salvo.

4.5 Estética

En cuanto a la estética del juego se busca acercarse lo máximo posible a la realidad dentro del toque de terror fantástico que se da. El videojuego tiene lugar en una mansión de tres pisos (garaje, piso de abajo y piso de arriba). Al transcurrir de noche, dominarán los toques sombríos y oscuros para dar una tónica de desconocimiento del lugar.

4.5.1 Personajes

En cuanto a personajes solo está el bebé que es el protagonista de nuestro videojuego. Dado que este proyecto fue empezado junto con estudiantes de arte, ellos fueron los encargados de realizar el modelado del bebé. Como el juego es realizado en primera persona, nunca se va a alcanzar a ver el rostro del bebé por lo que no se vió necesario la realización de su cara.

Aunque al principio surgió la idea de que el bebé pudiese ser personalizable, se decidió que no era muy adecuado para un juego de miedo así que se deshecho esa idea.



Figura 21: Bebé

4.5.2 Enemigos

En este caso hay que diferenciar entre varios amigurumis. Son cinco los enemigos que hay en este videojuego, todos ellos amigurumis de animales: un conejo, un oso, una oveja, un pájaro y un mono. Cada uno rondará una zona de la casa en específico, pero eso no quiere decir que puedan rondar por otros lugares de la casa haciendo de las suyas.

Como se ha comentado anteriormente, dado que el proyecto fue empezado con estudiantes de artes, el primer amigurumi (oso) es creado por estos estudiantes. Los otros cuatro restantes han sido desarrollados por un animador y modelador profesional ya que el programador de este proyecto no contaba con los conocimientos adecuados para llevar a cabo dicha tarea.

A continuación, se irá detallando las funciones de cada amigurumi.

Oso

Posiblemente el enemigo más sencillo, lo verás al principio, simplemente va andando por la habitación, en el caso de que andes cerca de donde se encuentra, girará en torno a sí mismo para buscar al bebé. En el caso de que lo encuentre irá corriendo a por él, aunque la velocidad de este no sea gran cosa. Se encuentra en la primera habitación y raramente lo verás en otras habitaciones.

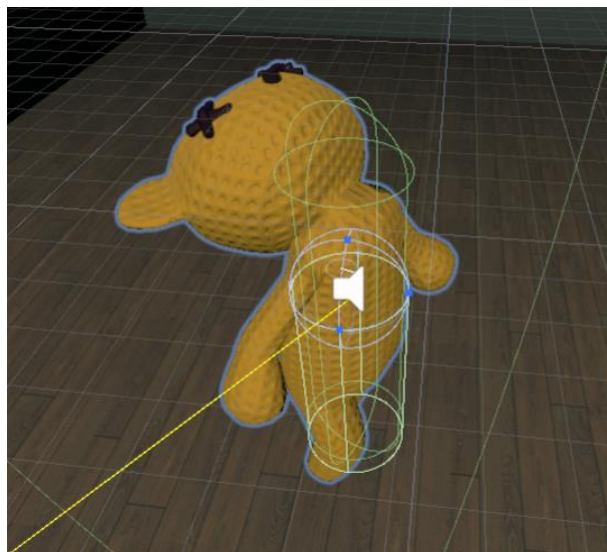


Figura 22: Oso

Pájaro

Este amigurumi se encuentra en la cocina y es algo más complejo que los demás ya que va volando por toda la cocina y si detecta al bebé entonces baja al suelo a por él, además dependiendo de las zonas por las que vayas, el pájaro se puede teletransportar cerca de ti para perseguir al jugador. El jugador se encontrará con este amigurumi varias veces a lo largo del juego, ya sea como susto o persiguiéndole.

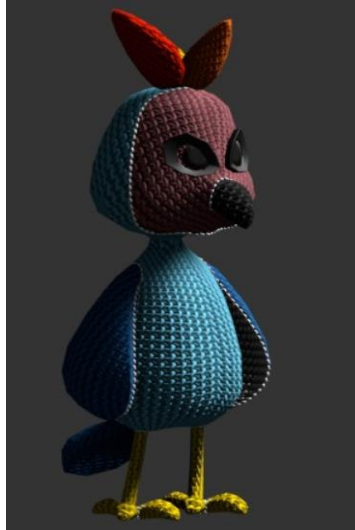


Figura 23: Pájaro

Oveja

La oveja es bastante parecida al funcionamiento del oso, pero es bastante más rápida que el oso así que, el jugador tendrá que ser hábil a la hora de esconderse para que no le vea la oveja. También la oveja produce una cantidad de sustos mayor a la de los dos anteriores. Este amigurumi suele frecuentar el aseo y puede aparecer en el salón.

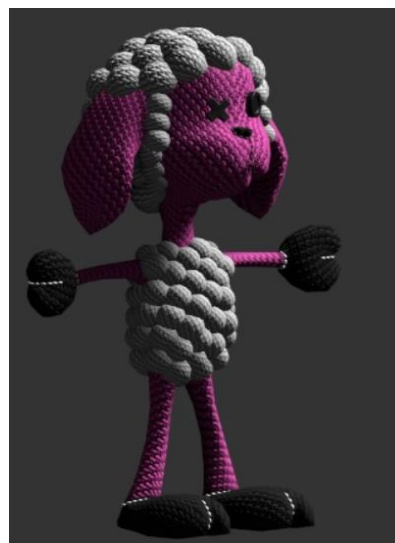


Figura 24: Oveja

Mono

Sin duda, uno de los más terroríficos, es el más rápido de todos, es capaz de escalar por las paredes y, sobre todo, es el que está más presente en todas las habitaciones de la casa dando multitud de sustos. Es en el garaje cuando el jugador tendrá que evitarlo mientras resuelve el rompecabezas de la habitación.

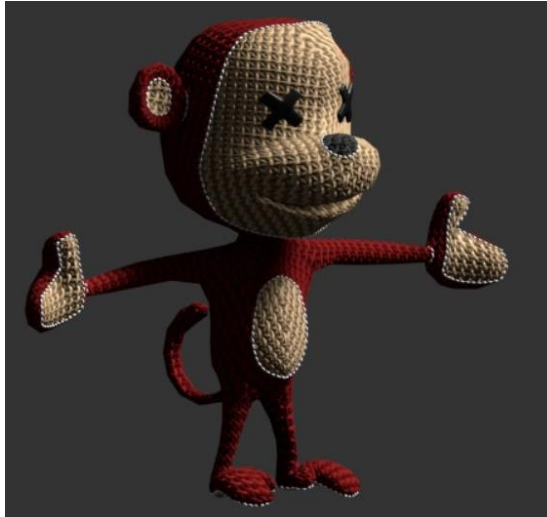


Figura 25: Mono

Conejo

Junto con el mono, el conejo es otro de los más terroríficos amigurumis del juego. El conejo tiene la particularidad de que va saltando por el salón y si aterriza cerca de la localización del jugador irá a por él. Además, aparecerá en varios sitios de la casa corriendo, cerrando puertas, dando sustos, etc.

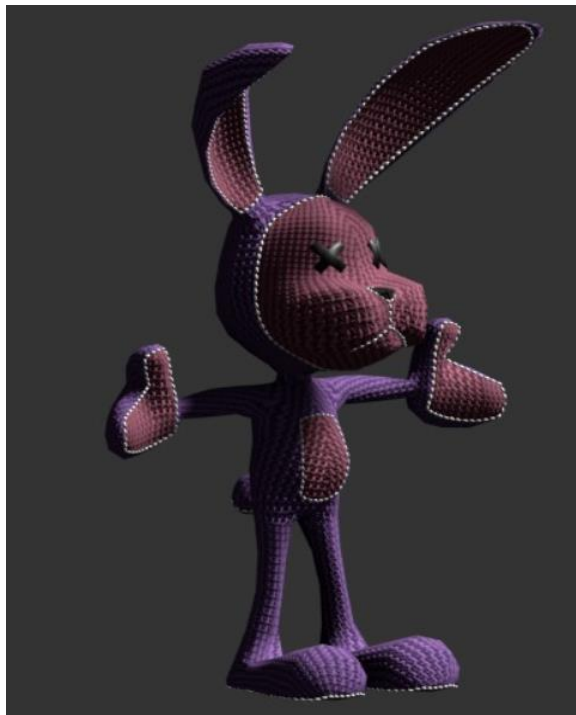


Figura 26: Conejo

4.5.3 Entorno

Como se ha dicho antes en la introducción de este apartado, se ha buscado tener una decoración lo más realista posible, simulando una mansión. Por lo tanto, la mayoría de los objetos han sido sacados de la tienda proporcionada por Unity, el resto han sido creado a mano a través de figuras geométricas. Seguidamente se mostrará unas imágenes en las que se verán algunas escenas del videojuego en las que se puede apreciar la decoración del entorno.

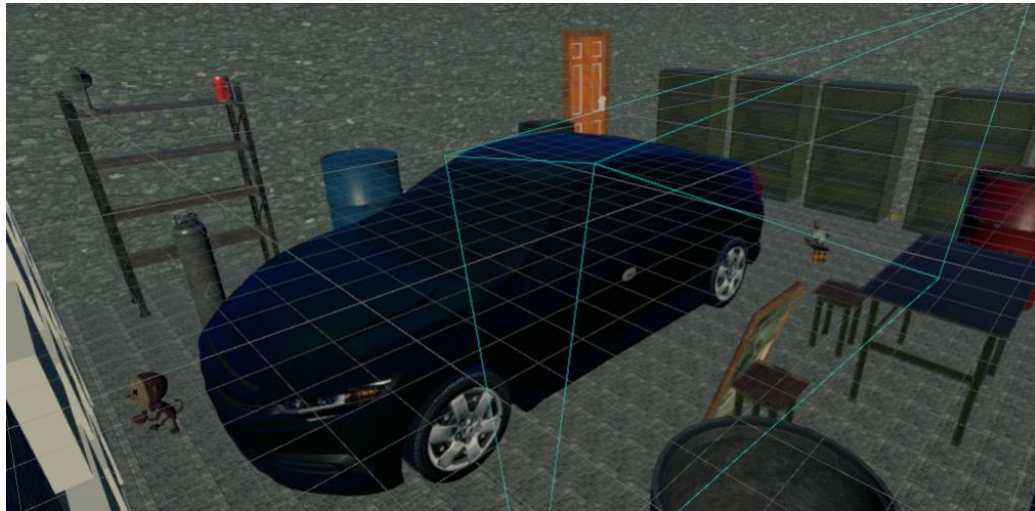


Figura 27: Garaje

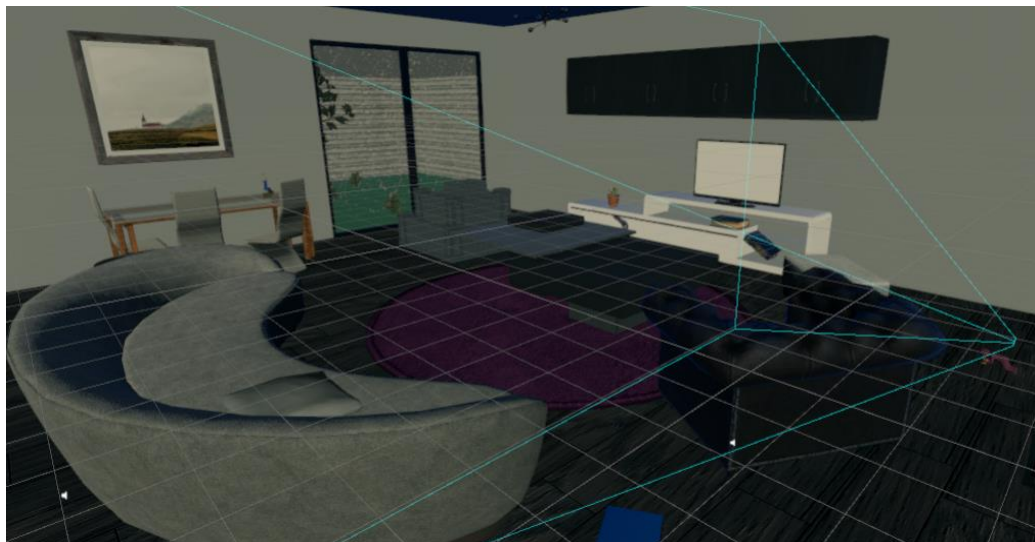


Figura 28: Salón

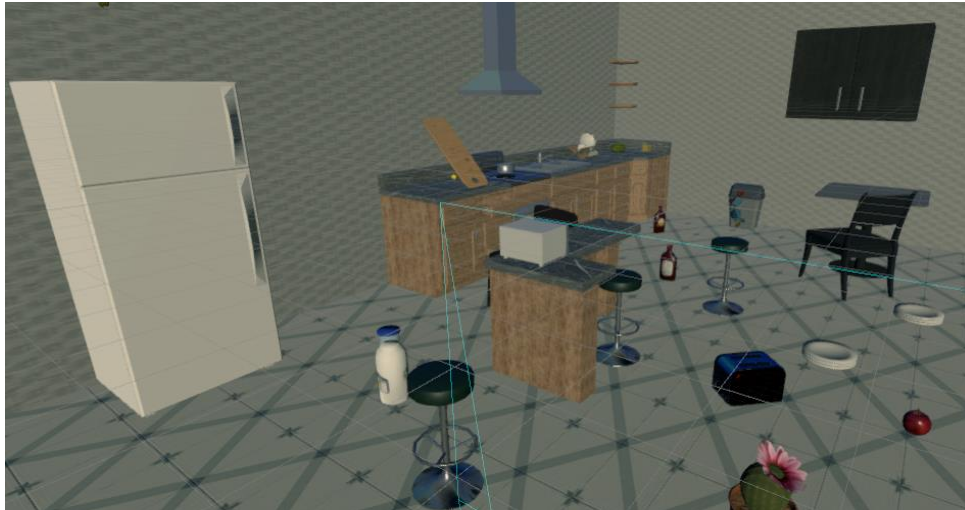


Figura 29: Cocina

4.5.4 Interfaz de usuario

A continuación, para este apartado, se realizarán una serie de tablas en las que se verán las diferentes pantallas o interfaces de usuario del videojuego, como acceder a ellas y un boceto representativo.

ID y nombre	IU-01 Menú principal
Descripción	Es la pantalla inicial del videojuego donde se puede apreciar el título del juego, así como las opciones de empezar partida, continuar con una guardada, entrar en el menú de opciones, mirar los controles o salir del juego.
¿Cómo activarlo?	<ol style="list-style-type: none"> 1.El jugador inicia el videojuego en su ordenador. 2.El jugador tras perder, decide volver al menú principal. 3.El jugador desde el menú de pausa decide salir del nivel.
Boceto	

Tabla 63: IU-01 Menú principal

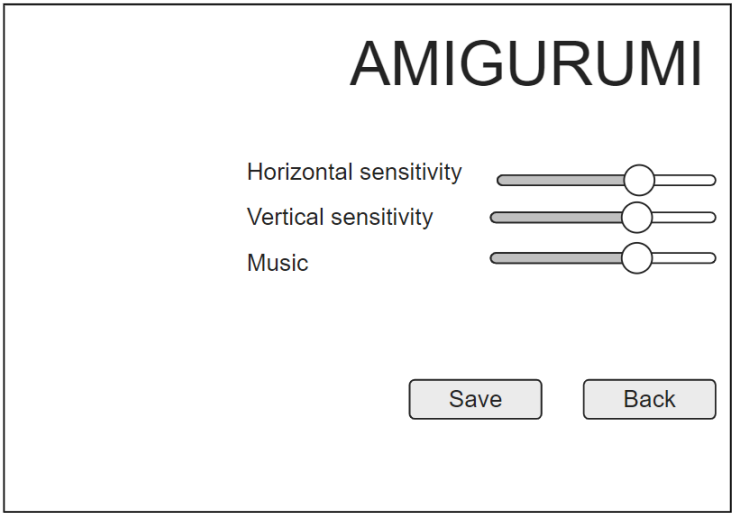
ID y nombre	IU-02 Menú de opciones
Descripción	Es la pantalla en la que el jugador podrá modificar algunas opciones como pueden llegar a ser la sensibilidad horizontal y vertical del ratón para gestionar como de rápido quiere que se mueva la cámara dentro del videojuego, así como también puede modificar la música.
¿Como activarlo?	1.El jugador accede a las opciones desde el menú principal. 2.El jugador accede a las opciones desde el menú pausa desde cualquier nivel.
Boceto	

Tabla 64: IU-02 Menú de opciones


ID y nombre	IU-03 Menú de controles
Descripción	En esta pantalla el jugador puede ver los controles del videojuego.
¿Como activarlo?	1.El jugador accede a los controles desde el menú principal. 2.El jugador accede a los controles desde el menú pausa desde cualquier nivel.
Boceto	

Tabla 65: IU-03 Menú de controles

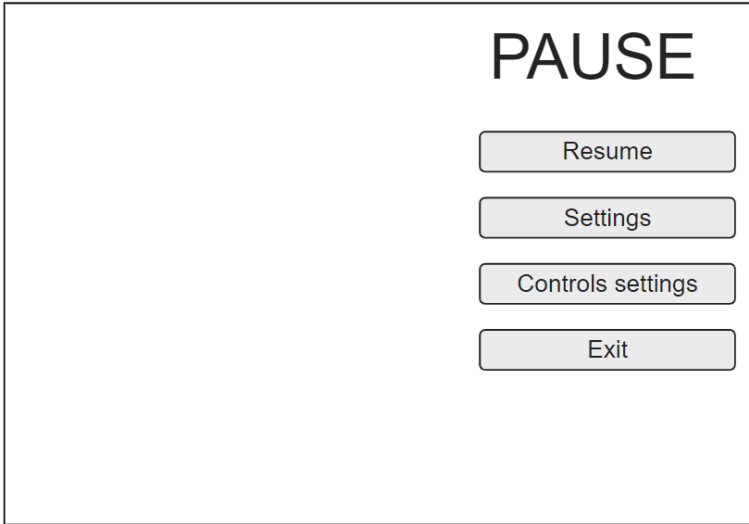
ID y nombre	IU-04 Menú de pausa
Descripción	En esta pantalla el jugador puede parar el juego, acceder a los menús de controles y de opciones, así como reanudar el juego o salir al menú principal.
¿Como activarlo?	1.El jugador accede al menú de pausa desde cualquier nivel con la tecla ESCAPE.
Boceto	

Tabla 66: IU-04 Menú de pausa

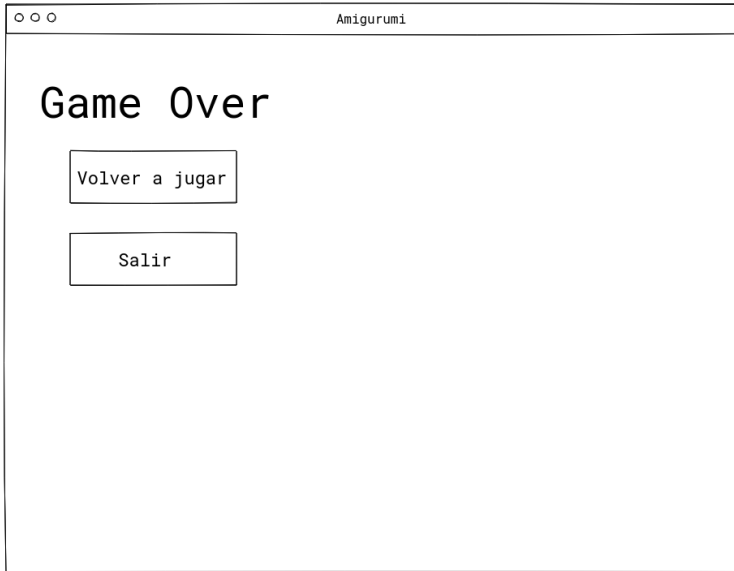
ID y nombre	IU-05 Pantalla de <i>game over</i>
Descripción	Esta pantalla aparece en el caso de que el jugador sea atrapado por algún amigurumi o se quede mirándolo durante mucho tiempo. Solo tienes dos opciones, volver a intentar el nivel desde el principio o volver al menú principal.
¿Como activarlo?	1.El jugador es atrapado por un amigurumi. 2.El jugador mira al amigurumi hasta que la barra de cordura se agota.
Boceto	

Tabla 67: IU-05 *Game Over*

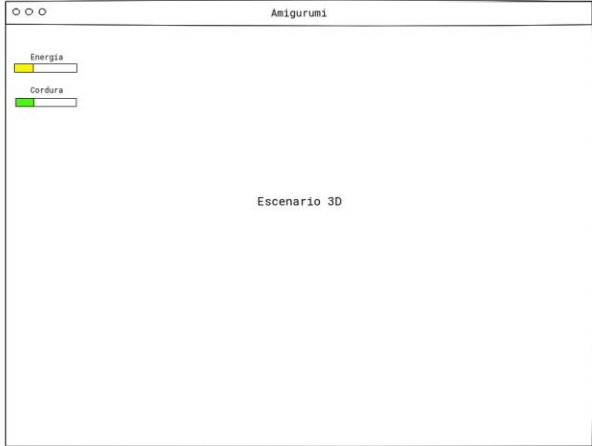
ID y nombre	IU-06 Pantalla de juego
Descripción	En esta pantalla es donde transcurre el videojuego. También estarán las barras de cordura y energía.
¿Como activarlo?	<ol style="list-style-type: none"> 1.El jugador accede a la pantalla de juego tras empezar nueva partida. 2.El jugador accede a la pantalla de juego tras continuar una partida guardada. 3.El jugador accede a la pantalla de juego desde la pantalla de <i>game over</i>. 4. El jugador accede a la pantalla de juego desde el menú de pausa reanudando el juego
Boceto	

Tabla 68: IU-06 Pantalla de juego

4.5.5 Diagrama de flujo

Finalmente, aquí está el diagrama de flujo del videojuego. Básicamente es una simulación de como puede navegar el jugador por el juego entre las diferentes interfaces mostradas previamente.

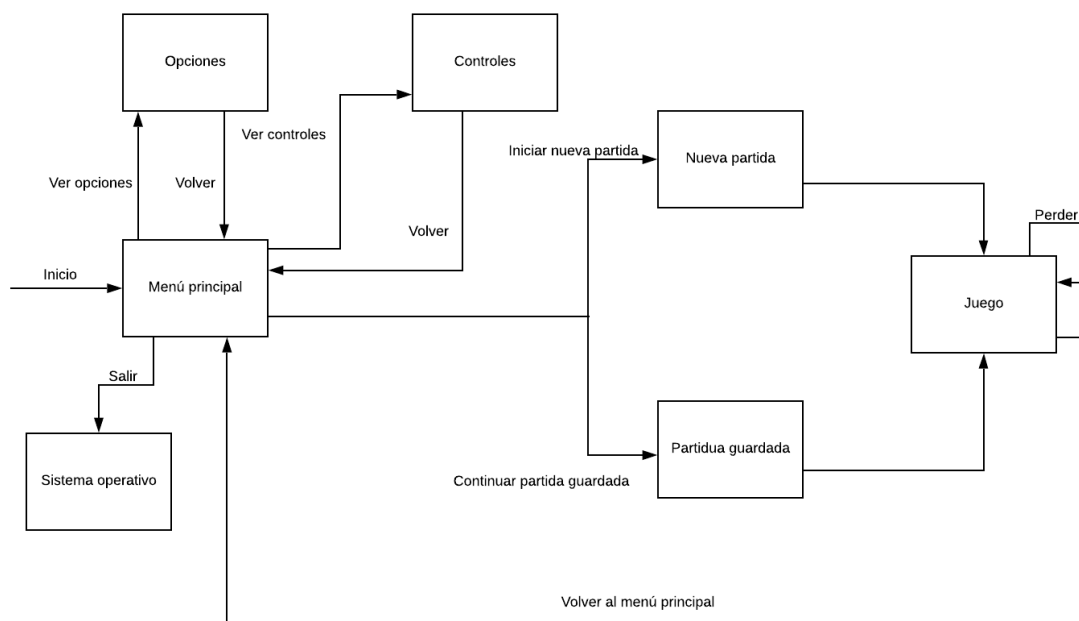


Diagrama 24: Diagrama de flujo

4.6 Diseño de los controles

TECLA	FUNCIÓN	CONTEXTO
W	Mover el bebé hacia delante	El bebé gatea hacia delante.
A	Mover el bebé hacia la izquierda	El bebé gatea hacia la izquierda.
S	Mover el bebé hacia atrás	El bebé gatea hacia atrás.
D	Mover el bebé hacia la derecha	El bebé gatea hacia la derecha.
E	Interactuar	El bebé podrá interactuar con objetos y recogerlos para posteriormente utilizarlos. También se usará para abrir una puerta, empujar, o cualquier acción similar.
C	Subir	Cuando el bebé esté de pie podrá subir a determinados objetos para poder evitar al amigurumi o, para llegar a un sitio determinado.
Shift + dirección (W, A, S y D)	Correr	El bebé podrá gastar su energía para poder salir corriendo.
Escape	Lanzar menú de pausa	En la pantalla aparecerá un menú de pausa, el cual tendrá múltiples opciones para mejorar la experiencia del jugador.
Ratón	Manejar cámara	El jugador podrá manejar la vista del bebé con el ratón.

Ratón	Pulsar sobre opciones	Para elegir entre opciones tales como sensibilidad y música.
-------	-----------------------	--

Tabla 69: Controles

4.7 Diseño de los niveles

En cuanto al diseño de los niveles o escenas hay que distinguir entre dos tipos. El primero es el puro nivel (NVL), en el que tienes que recoger uno o dos bloques además de encontrar una fuente de luz para que el amigurumi muera y así poder pasar de escena. No obstante, también están las escenas de tránsito (EDT) en las que el jugador simplemente tiene que recoger los bloques para pasar de escena. En el último tipo de escena, el amigurumi no puede morir.

Para orientar al lector de este documento se incluyen imágenes simulando un plano de la habitación en el que se utilizarán 4 colores: el **rojo** para el recorrido del bebé para completar el nivel, el **azul** para el recorrido del bebé después de completar el nivel, el **amarillo** para simbolizar los elementos de luz que hay que activar para completar el nivel y el **negro** para simbolizar objetos que hacen la función de teletransportador.


ID y nombre	NIVEL-01 Despacho
¿Como completar el nivel?	El jugador tiene que recoger el bloque que encontrará al aparecer en la escena, después irá hacia el otro extremo de la habitación, sacará los cajones haciendo una escalera y se apoyará en la silla para subir a la mesa. Una vez ahí encenderá la luz y el amigurumi con forma de oso morirá. Tras ello, el jugador podrá abrir la puerta, recoger el bloque del pasillo y avanzar al baño.
Imagen	

Tabla 70: NVL-01 Despacho


ID y nombre	NVL-02 Baño
¿Como completar el nivel?	Este nivel es de los más complejos, la fuente de luz es una tele está sobre una mesa que es inalcanzable para el bebé. El objetivo del bebé es el de activar 3 objetos que expulsan agua de la manera que la habitación quede inundada y el jugador pueda llegar a la televisión. Para el caso de la ducha el jugador simplemente tendrá que empujar un objeto para poder subir a la ducha y activarla. En el caso del grifo del lavabo, hay un cuadro detrás del váter diciendo que vayas ahí, una vez llegado al cuadro te teletransportas hacia el grifo y lo puedes activar. Para el jacuzzi, hay unos números escritos por toda la habitación que hacen un código. Al lado del jacuzzi hay un cuaderno donde poder anotar el código y, en caso de que sea el correcto, aparecerá una rampa con la que se podrá llegar al jacuzzi y activarlo. Después de esto el jugador podrá activar la tele, el amigurumi morirá y se reproducirá un video escalofriante en la televisión. Todo esto con sustos y huyendo del amigurumi en forma de oveja.
Imagen	

Tabla 71: NVL-02 Baño

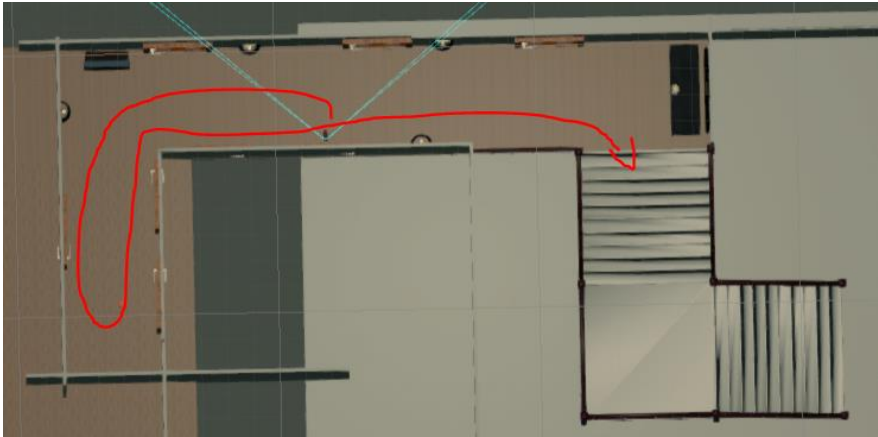
ID y nombre	EDT-01 Pasillo
¿Como completar el nivel?	Cuando sales del baño tendrás que ir hasta el final del pasillo para recoger el bloque de esta escena e ir por las escaleras para poder avanzar.
Imagen	

Tabla 72: EDT-01 Pasillo

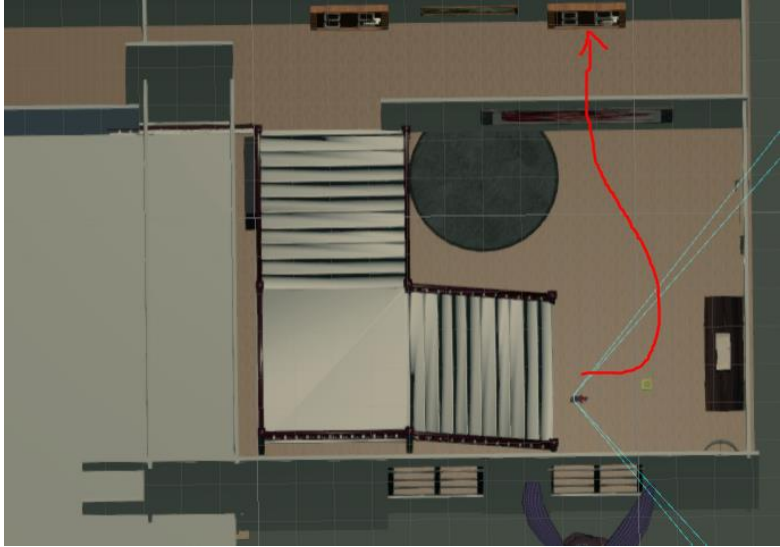
ID y nombre	EDT-02 Entrada de la casa
¿Como completar el nivel?	Una vez bajadas las escaleras, tendrás que recoger el bloque de dicha escena y tras investigar durante un rato el escenario, el jugador podrá avanzar a la cocina.
Imagen	

Tabla 73: EDT-02 Entrada de la casa

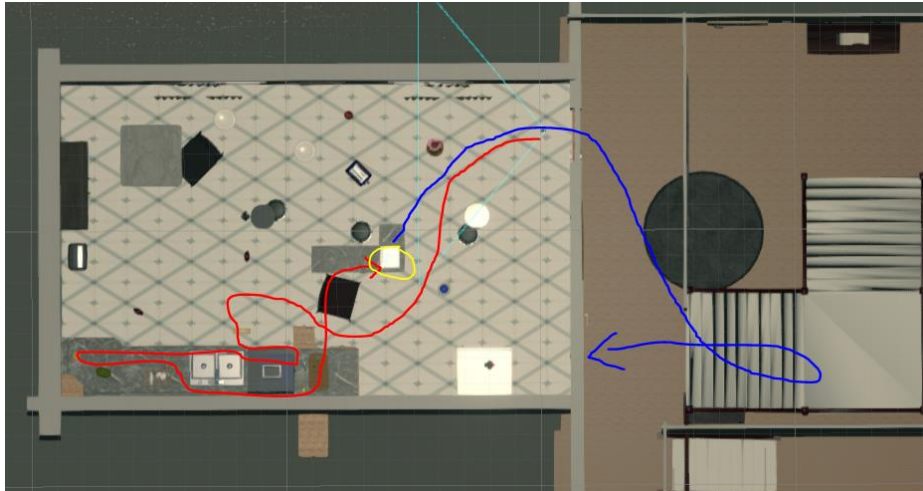
ID y nombre	NVL-03 Cocina
¿Como completar el nivel?	El jugador tendrá que investigar durante un rato la cocina porque hasta cierto tiempo no aparece la caja de cereales, que es clave para el devenir del nivel. El jugador tiene que utilizar dicha caja para subir al horno y de ahí montar otra escalera de cajones parecida a la del nivel 1. Una vez subido a la encimera el jugador recogerá el bloque y utilizará la tabla de cortar para apoyarse en la silla y de ahí moverse hacia la isla de la cocina donde el jugador puede activar el microondas y de esta manera derrotar al amigurumi. Una vez completado el nivel, tendrá que dirigirse hacia la puerta y coger el bloque que hay en la entrada de la casa y dirigirse hacia el salón.
Imagen	

Tabla 74: NVL-03 Cocina

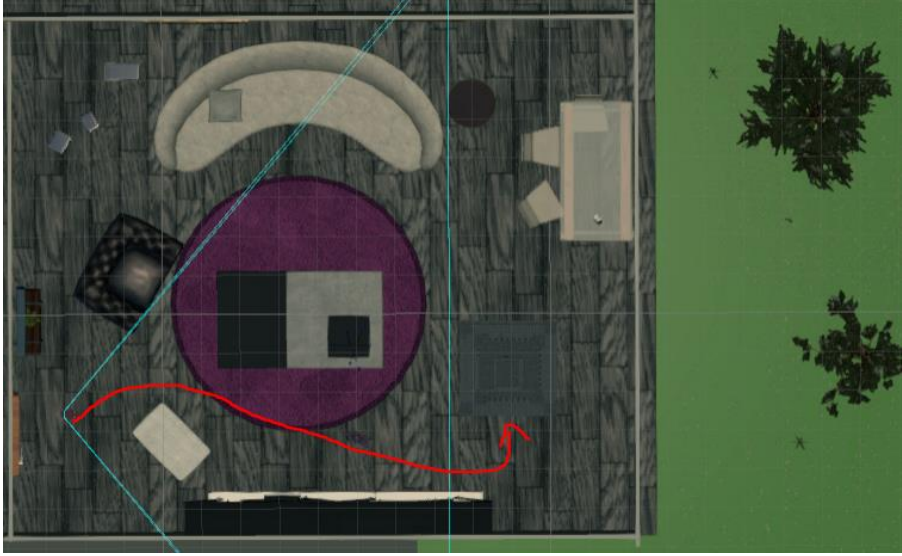
ID y nombre	NVL-04 PT1 Salón
¿Como completar el nivel?	El nivel 4 estará dividido en 3 partes, esta es la primera dónde hay un castillo de juguete al que el jugador tendrá que ir para poder pasar a la siguiente fase.
Imagen	

Tabla 75: NVL-04 PT1 Salón

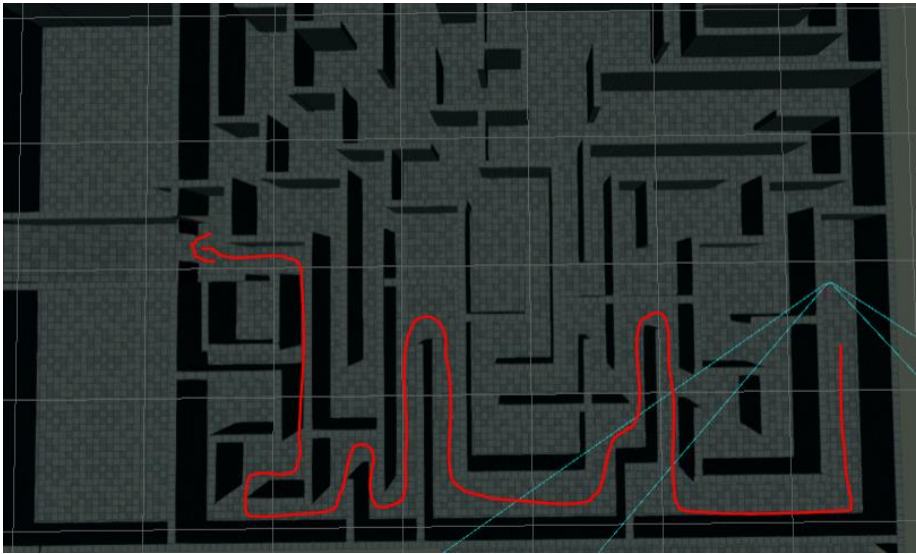
ID y nombre	NVL-04 PT2 Laberinto
¿Como completar el nivel?	Una vez que el jugador entra en el castillo de juguete se introducirá en un laberinto que tendrá que superar para poder avanzar a la última parte del nivel 4.
Imagen	

Tabla 76: NVL-04 PT2 Laberinto


ID y nombre	NVL-04 PT3 Salón fantasma
¿Como completar el nivel?	Cuando el jugador sale del laberinto, se da cuenta de que los objetos del salón están flotando y que todo parece estar fuera de sitio. El bebé recogerá un mechero y aprovechará una serie de objetos flotantes para poder llegar a la vela situada en la mesa para poder completar el nivel. Una vez completado el jugador saldrá de la habitación recogerá el bloque que hay en la entrada de la casa y se irá hacia el garaje de la casa.
Imagen	

Tabla 77: NVL-04 PT3 Salón fantasma

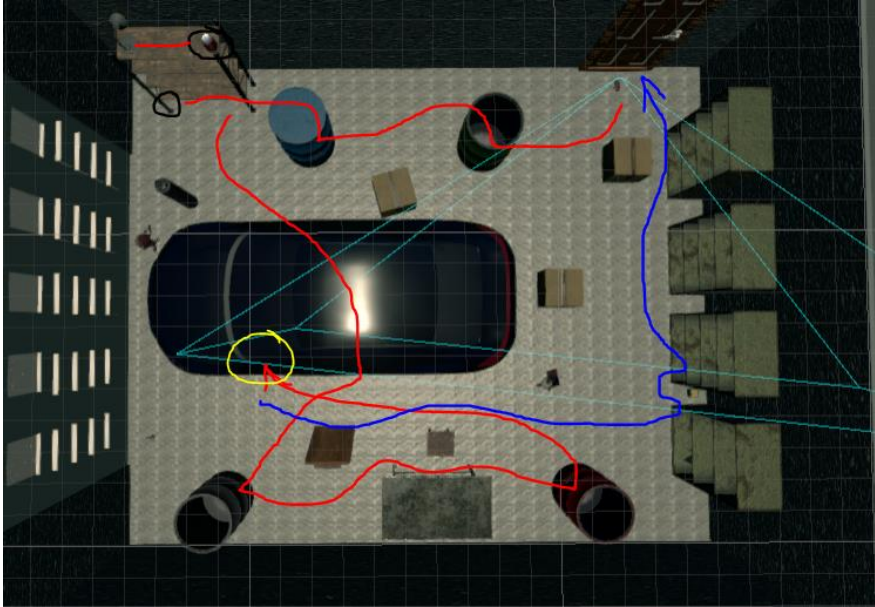
ID y nombre	NVL-05 Garaje
¿Como completar el nivel?	<p>En este nivel hay dos objetivos, uno como es obvio, es acabar con la vida del amigurumi activando las luces del coche y otro es el de encontrar un código para poder obtener la llave que le dé al jugador la posibilidad de ir a la habitación de sus padres y así poder completar el juego. En este nivel hay unos elementos claves y esos son lo barriles de colores, aunque no se aprecie del todo bien en la imagen hay un barril verde, uno rojo, uno azul y uno gris. El caso es que el jugador puede mover los barriles y debajo de estos habrá una pista de a donde tiene que ir para conseguir un número del código. En el caso del verde aparecerá directamente un número del código, el azul te dirá que objeto es teletransportable para poder ir a una localización dónde conseguir otro número. El gris activará un número debajo del coche de la manera que cuando vayas por debajo del coche podrás verlo. En el nivel además hay unas llaves del coche que el jugador deberá recoger para activar las luces del coche, una vez movido el barril rojo, el jugador podrá utilizar un cuadro para subirse al coche y de esta manera activar las luces del coche y que mostrará el número que faltaba del código. Una vez que el jugador haya recogido el bloque del nivel y haya introducido el código aparecerán las llaves de los padres y el jugador podrá salir de la habitación.</p>
Imagen	

Tabla 78: NVL-05 Garaje

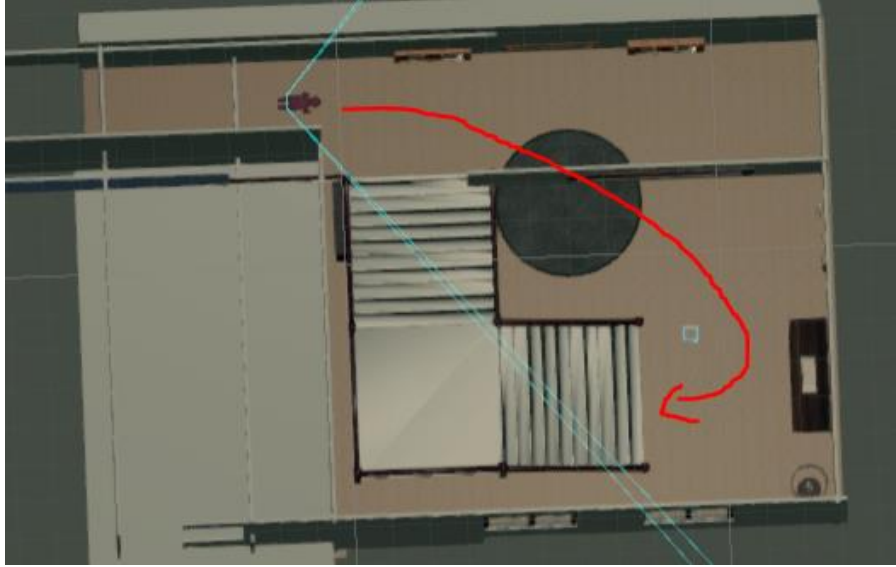
ID y nombre	EDT-03 Entrada de la casa soleada
¿Como completar el nivel?	El jugador se encuentra ahora con un entorno bastante cambiado al que está acostumbrado. Ahora todo es más soleado y no hay peligro por lo que no el bebé recoge el bloque y avanza por las escaleras.
Imagen	

Tabla 79: EDT-03 Entrada de la casa soleada


ID y nombre	EDT-04 Pasillo soleado
¿Como completar el nivel?	El jugador avanza hasta la habitación de sus padres, monta el castillo de bloques de bebé y abre la puerta. Una vez abierta el juego está terminado.
Imagen	

Tabla 80: EDT-04 Pasillo soleado

Capítulo 5

Desarrollo e implantación de la solución

En este capítulo se procede a explicar cómo se ha desarrollado e implementado la estructura y el diseño especificado. Esta parte del documento está dirigida a aquellas personas que quieran entender las funcionalidades desarrolladas internamente, a un nivel más técnico.

5.1 Organización

El proyecto está organizado por capas:

Amigurumi: proyecto Unity3D

- **Assets:** Paquete de recursos
 - **Animations:** *asset* que contiene las animaciones del videojuego.
 - Camera: animaciones relacionadas con la cámara del bebé.
 - Characters: animaciones relacionadas con movimientos de los amigurumis y del bebé.
 - Final: animaciones relacionadas con el nivel final.
 - Level1: animaciones relacionadas con el primer nivel.
 - Level2: animaciones relacionadas con el segundo nivel.
 - Level3: animaciones relacionadas con el tercer nivel.
 - Level4: animaciones relacionadas con el cuarto nivel.
 - Level5: animaciones relacionadas con el quinto nivel.
 - LevelTransition: animaciones relacionadas con las transiciones que hay entre las escenas.
 - Main menu: animaciones relacionadas con el menú principal.
 - Moth: animaciones relacionadas con la polilla.
 - Scary: animaciones relacionadas con los sustos.
 - SpeechBubbles: animaciones relacionadas con las burbujas o bocadillos que van contando la historia durante el videojuego.
 - **Design:** *asset* que contiene aquellos elementos que sirven para la decoración del nivel.
 - Aura: contiene una serie de recursos que mejoran la iluminación por defecto de Unity lo que permite al videojuego tener un acabado mejor.
 - Baby Blocks: son los bloques de letras utilizados para el videojuego.
 - Bathroom: dentro de esta carpeta están todos los recursos utilizados en el nivel del baño.
 - Garage: contiene todos los recursos utilizados en el nivel del garaje.

- Kitchen: contiene todos los recursos utilizados en el nivel de la cocina.
 - Living room: contiene todos los recursos utilizados en el nivel del salón.
 - Materials: son los materiales utilizados para suelos, paredes techos, ojos de los amigurumis, etc.
 - Pictures: cuadros utilizados en los diferentes escenarios del videojuego para decorar.
 - RDL: es un pack de cursores originales que se ha utilizado para diseñar el cursor del videojuego.
 - Sky: contiene los diferentes cielos que hay en el videojuego, tanto los nocturnos como los más soleados.
 - PreTFG: contiene aquellos recursos de decoración utilizados antes de realizar este proyecto como trabajo final de grado.
- **Fonts**: *asset* que contiene las fuentes utilizadas en el videojuego para títulos, textos y botones.
- **Music**: *asset* que contiene la música y sonidos utilizados en el videojuego.
 - Entity: se trata de un pack gratuito de ambientes terroríficos.
 - Level 2: sonidos y música utilizados especialmente en el nivel 2.
 - Level 3: sonidos y música utilizados especialmente en el nivel 3.
 - Level 5: sonidos y música utilizados especialmente en el nivel 5.
 - Objects: sonidos de determinados objetos.
 - Resources: sonidos y música utilizados a lo largo de todo el videojuego.
 - Scary: sonidos y música utilizados para dar sustos.
 - Whispers: susurros utilizados para poner en tensión al jugador.
- **Prefab**: *asset* que contiene los prefabs o también denominados objetos reutilizables que gozan de máxima importancia en el videojuego.
 - Door: diferentes tipos de puertas utilizadas en el videojuego.
 - Enemies: prefabs de amigurumis.
 - Interactive: prefabs de aquellos objetos con los que se pueden interactuar.
 - Kitchen: prefabs utilizados específicamente en la cocina.
 - Level 1: prefabs utilizados en el nivel 1, la mayoría de antes de realizar este videojuego como trabajo final de grado.
 - Materials: materiales que se han reutilizado a lo largo del videojuego.
- **Scenes**: *asset* que contiene las escenas del videojuego
 - Level 1: escena del nivel 1 y ajustes.
 - Level 2: escena del nivel 2 y ajustes.
 - Level 3: escena del nivel 3 y ajustes.
 - Level 4: escenas del nivel 4, escena del laberinto y ajustes.
 - Level 5: escena del nivel 5 y ajustes.
 - NoLevels: escenas de tránsito como viene a ser el pasillo o la entrada de la casa.
 - UI: escenas relacionadas con el menú principal o las pantallas de *game over*.

- **Scripts:** *asset* que contiene los *scripts* del videojuego.
 - Game: *scripts* relacionados con el funcionamiento general del juego.
 - IA: *scripts* relacionados con la inteligencia artificial de los amigurumis.
 - Interactable: *scripts* relacionados con los objetos con los que puede interactuar el jugador.
 - LoadSave: *scripts* relacionados con el sistema de guardado de opciones y del progreso del jugador.
 - Player: *scripts* relacionados con todo aquello que tenga que ver con el jugador.
 - Scary: *scripts* encargados de dar funcionalidad a los sustos.
 - UI: *scripts* encargados de las funcionalidades de los elementos de la interfaz del usuario.
- **Timelines:** *asset* que contiene las cinemáticas del videojuego.
- **UI:** *asset* que contiene los recursos que son utilizados en la interfaz de usuario como vídeos o imágenes.
 - Backgrounds: fondos utilizados en algunas escenas.
 - Level3: elementos de la interfaz de usuario utilizados específicamente en el nivel 3.
 - Materials: materiales utilizados para los elementos de la interfaz de usuario.
 - Menu: elementos relacionados con el menú principal.
 - Pictures: fotos utilizadas en el videojuego.
 - Resources: elementos utilizados a lo largo de todas las escenas del videojuego.
 - Settings: elementos relacionados con las opciones del juego.
 - Videos: videos utilizados en el videojuego.

5.2 Escenas

Todas las escenas del juego siguen la siguiente estructura:

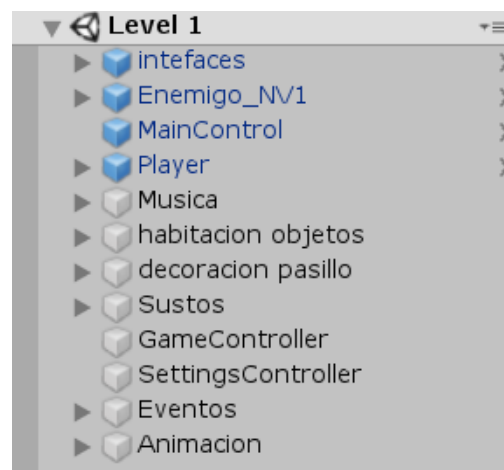


Figura 30: Estructura de los niveles de los juegos

Ahora se pasará a detallar cada elemento de la estructura de cada nivel. En el caso de que haya que añadir una parte de código para enseñarlo al lector, se introducirá en el anexo con su correspondiente explicación.

5.2.1 MainControl

Sin duda es de las partes más importantes del videojuego ya que abarca el funcionamiento general del videojuego. Está compuesto de los siguientes elementos:

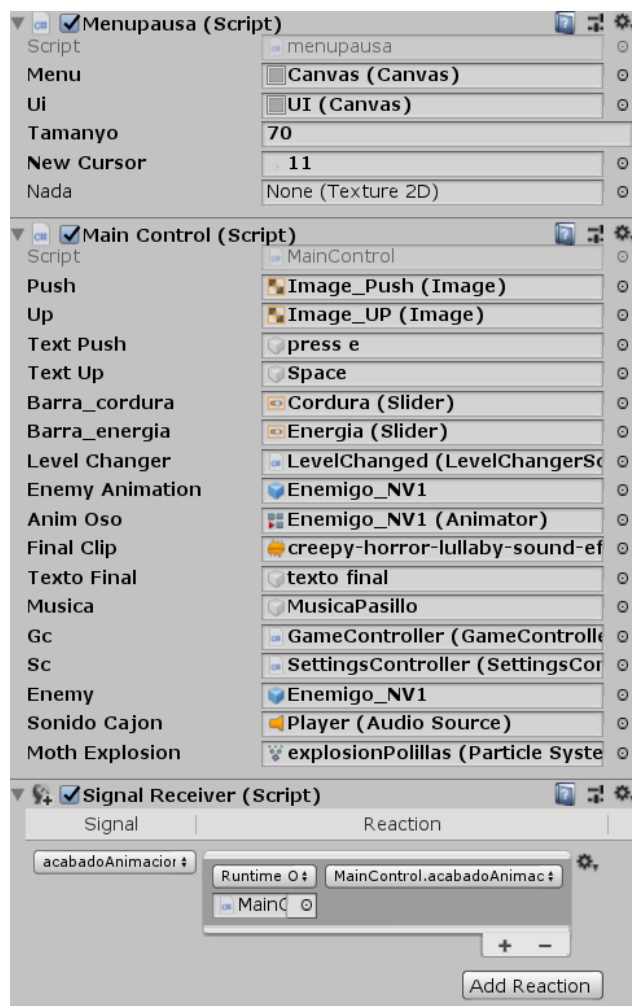


Figura 31: Estructura de MainControl

Dentro de este elemento hay 3 apartados. Primero está el *script* del menú de pausa que tiene relación con la aparición del menú de pausa y sus funciones que se detallarán más tarde en el tema de interfaces. Como aspecto que sí se puede comentar de este *script* es la utilización de un cursor original, de la manera que hay que esconder el predeterminado del ordenador y hacer que el seleccionado pase a ser el encargado de hacer dicha función. ([Anexo 1.1](#))

Después aparece la gran parte del MainControl que es su propio *script*. Este *script* contiene bastantes métodos que luego serán usados en gran parte por el *script* del jugador, por interfaces, etc.

Por ejemplo y como se ve en el final de la imagen, MainControl se encarga de entre muchas cosas del tema de la animación de después de completar el nivel. Para ello utiliza dos métodos: `luzEncendida()` y `acabadoAnimacion()`. ([Anexo 1.2](#))

También se gestiona desde el MainControl el cambio de dibujo de las barras de cordura y energía para simbolizar al jugador de cuando tiene poca energía y cordura. ([Anexo 1.3](#))

5.2.2 Player

Posiblemente el elemento más complicado del videojuego y que más problemas ha ocasionado durante el proyecto. También es el *script* más extenso del videojuego.

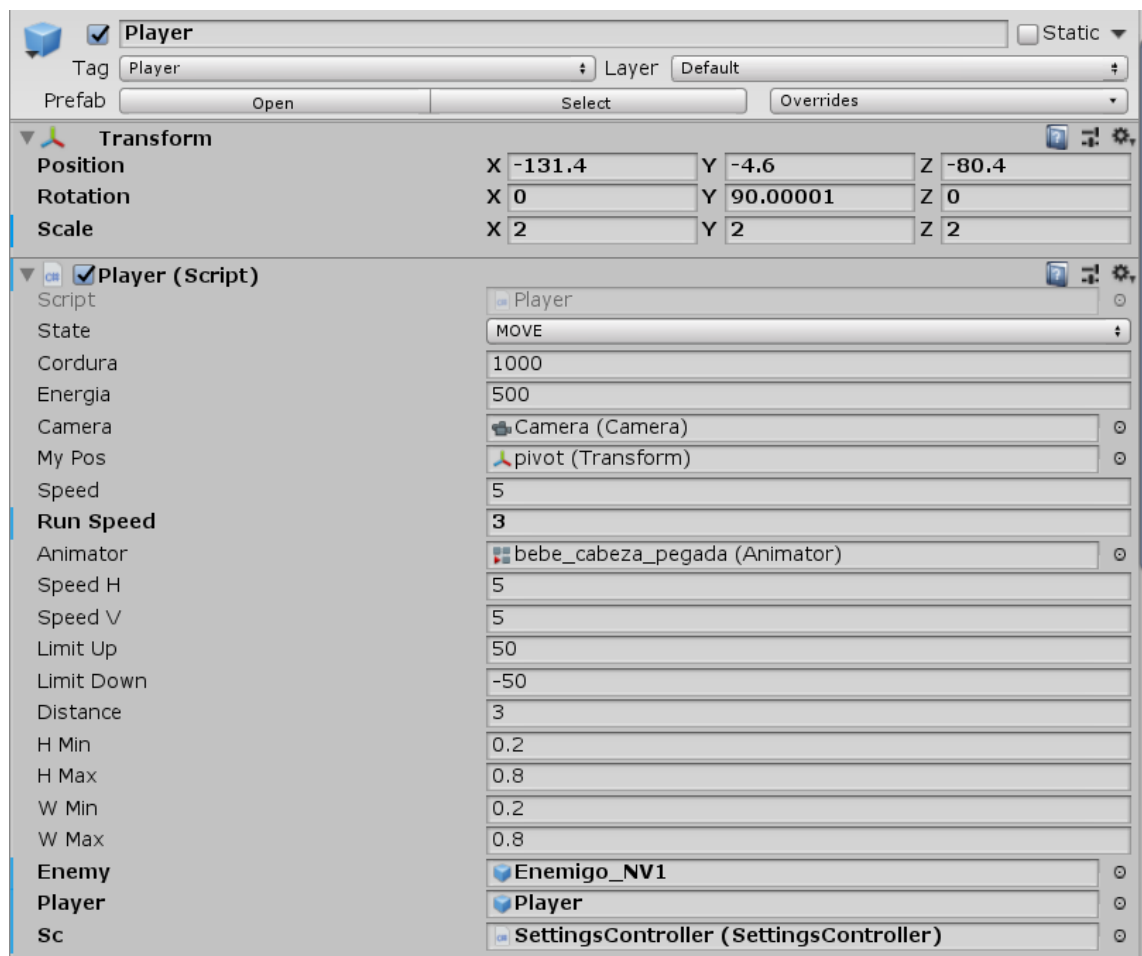


Figura 32: Estructura de Player

Dentro del *script* de *player* dentro del método `Update()` se dan varias funcionalidades tales como las de detectar si un objeto puede ser utilizado o no, si un objeto puede ser subido o no, si puede empujar a ese objeto y las acciones y animaciones correspondientes (subir, empujar, etc) ([Anexo 2.1](#))

También existe un `FixedUpdate()` centrado más en la parte del movimiento del personaje y de la cámara. ([Anexo 2.2](#))

5.2.3 Interfaces

En el apartado de interfaces hay que distinguir entre 3 partes: UI, canvas y LevelChanged.

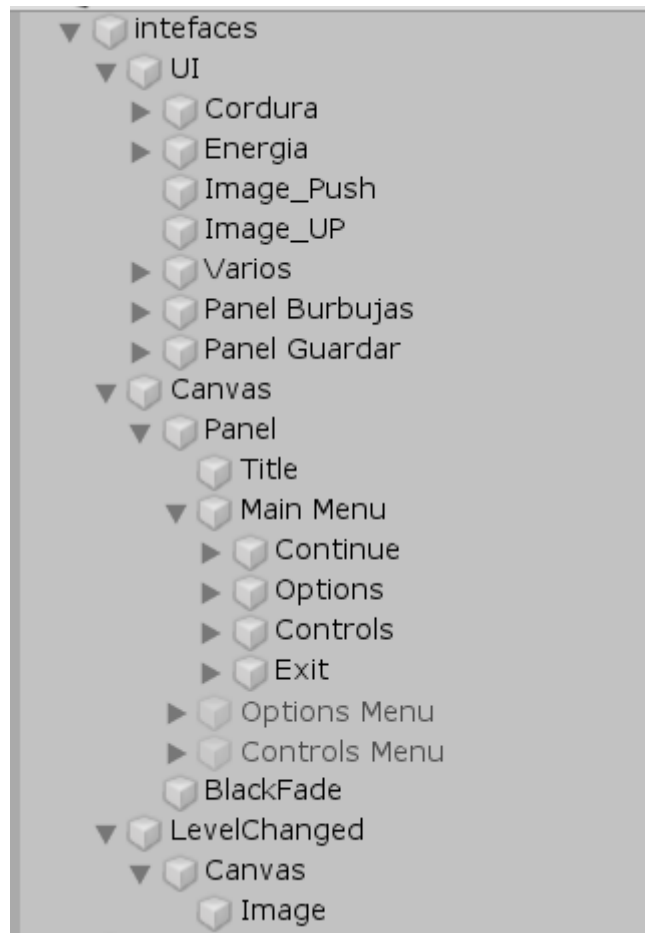


Figura 33: Jerarquía de interfaces

- **UI:** es utilizado para mostrar elementos durante el juego, es decir, mientras el jugador está jugando irán apareciendo elementos tales como:
 - Cordura: se trata de un *slider* o barra que se va rellenando o desgastando, dependiendo si mira al amigurumi o no. Siempre está presente y solo desaparece en caso de que el jugador pause el juego o haya alguna cinemática.
 - Energía: se trata de un *slider* o barra que se va rellenando o desgastando, dependiendo si el jugador corre o no. Siempre está presente y solo desaparece en caso de que el jugador pause el juego o haya alguna cinemática.
 - Image_Push: icono que aparece cuando el jugador puede interactuar con un objeto.
 - Image_UP: icono que aparece cuando el jugador puede subir a un objeto.
 - Varios: la parte de varios va destinada a aquellos textos que son informativos tales como el de informar al jugador que vaya hacia la

puerta una vez terminado el nivel o de que informe cuantos bloques lleva conseguidos en ese momento.

- Panel Burbujas: esta parte trata de los bocadillos que van contando la historia al principio de las escenas. ([Anexo 3.1](#))
- Panel Guardar: panel destinado a enseñar los iconos que informan al jugador que la partida ha sido guardada.
- **Canvas:** utilizado para el aspecto del menú de pausa, engloba las opciones de este menú y su funcionamiento. Dentro del panel principal se pueden encontrar los siguientes elementos.
 - Main Menu: dentro del menú de pausa ([Anexo 3.2](#)) se distinguen 4 botones: continuar, menú de opciones, ver controles y salir.
 - Menú de opciones: una vez activado esté menú y desactivado el anterior se puede gestionar la música y la sensibilidad horizontal y vertical.
 - Menú de controles: simplemente se trata de una serie de imágenes para que el jugador pueda ver los controles del juego.
- **LevelChanged:** Este elemento sirve para realizar el cambio de escena con una transición de desvanecimiento de color negro para dar un efecto suave de transición. ([Anexo 3.3](#))

5.2.4 Enemigo

Los amigurumis a pesar de tener bastantes componentes como se verá a continuación es un elemento bastante sencillo de entender.

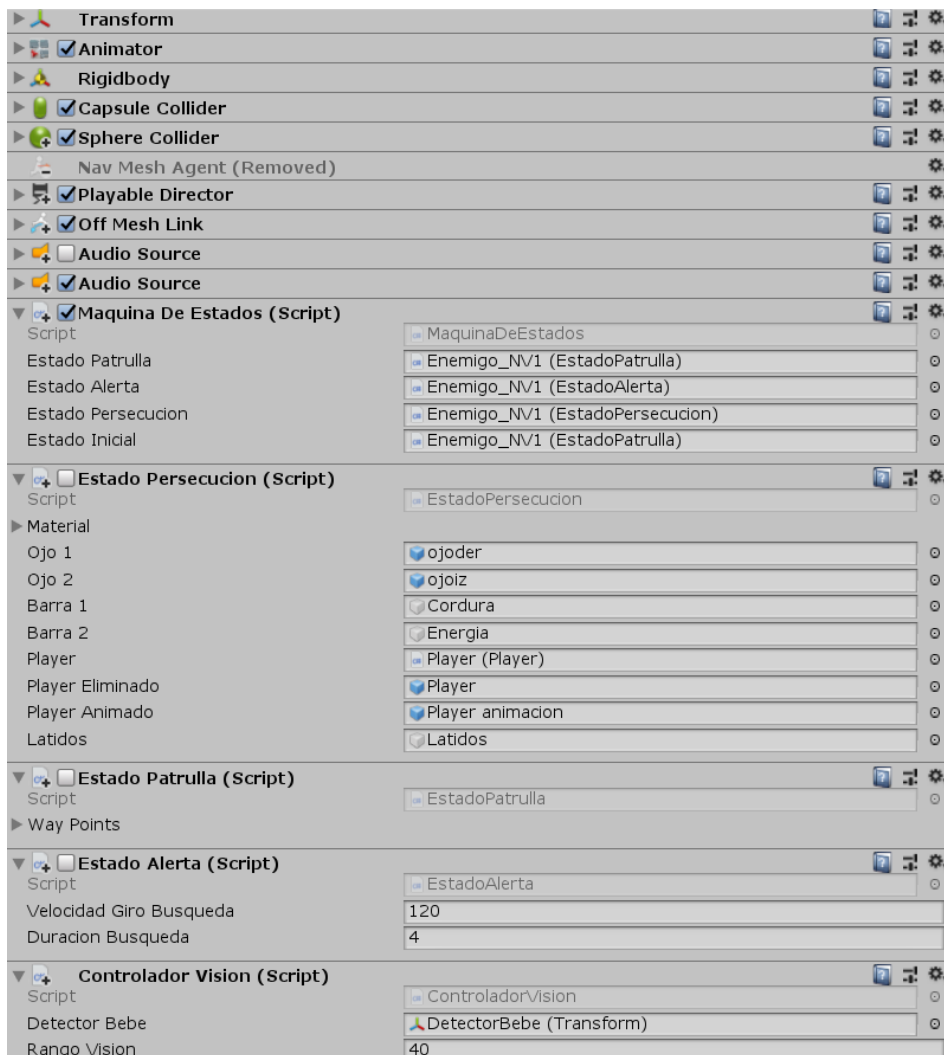


Figura 34: Estructura del enemigo

Un elemento que destacar y que no se ha comentado son los *colliders*, que como su nombre indica sirven para colisionar. Pueden ser de varias formas y hay que jugar con una propiedad que tienen que es *is trigger*, si no se marca está opción el jugador chocará con ese *collider*. En el caso de que se marque esa opción el jugador no choca, pero sin embargo si que se puede detectar que el jugador ha sobrepasado dicho *collider*. El enemigo tiene dos, uno que recubre su cuerpo en forma de capsula y una esfera grande a su alrededor que simplemente sirve para detectar si el jugador ha tocado ese *collider*.

El *playable director* que aparece sirve para en caso de que te pille el amigurumi lanzar una animación para hacer saber al jugador que ha perdido, los *audio sources* sirven para que en caso de que el amigurumi esté persiguiendo al jugador activar la música de persecución.

También hay que remarcar que todo amigurumi lleva un *navMeshAgent* que convierte al amigurumi en un agente que puede moverse por una zona determinada.

En cuanto a los scripts todo amigurumi tendrá mínimo 7: ControladorVision, EstadoPatrulla, EstadoAlerta, EstadoPersecucion, MaquinaDeEstados, ControladorNavMesh y Enemy.

- **ControladorVision:** mediante un detector de bebé que está en el amigurumi, este *script* decide si el amigurumi puede ver al jugador dentro de una distancia considerable. ([Anexo 4.1](#))
- **ControladorNavMesh:** es el *script* encargado de que el amigurumi se mueva dependiendo del estado que se encuentre. ([Anexo 4.2](#))
- **Enemy:** gracias a este *script*, en caso de que el jugador se quede mirando al enemigo, la cordura irá bajando. ([Anexo 4.3](#))
- **EstadoPatrulla:** cuando el amigurumi está en este estado, está realizando una patrulla por toda la habitación hasta que ve al jugador o detecta que está a sus alrededores. ([Anexo 4.4](#))
- **EstadoAlerta:** cuando el amigurumi está en este estado, está girando sobre sí mismo cierto tiempo a ver si encuentra al jugador, si ese tiempo pasa y no lo encuentra volverá al estado patrulla. De lo contrario pasará al estado persecución. ([Anexo 4.5](#))
- **EstadoPersecución:** cuando el amigurumi está en este estado, se inicia una música de persecución, los ojos del amigurumi se iluminan y su velocidad aumenta. En el caso de que el jugador se esconda el amigurumi parará de perseguirle y pasará a estado alerta. ([Anexo 4.6](#))
- **MaquinaDeEstados:** es el *script* encargado de cambiar de estados dependiendo de la situación. ([Anexo 4.7](#))

5.2.5 Decoración

Sobre los objetos que conforman la decoración no hay mucho que decir ya que su función es esa, la de decorar. Si que es verdad que hay objetos que tienen alguna que otra funcionalidad interesante y que se detallarán en el anexo. ([Anexo 5.1](#))

5.2.6 Sustos

En cuanto a los sustos que hay en el juego, todos tienen una misma estructura. Cada susto tiene un activador que se trata de un bloque invisible con la propiedad *is trigger* marcada de la manera que una vez que haya entrado el jugador en dicho bloque entonces es cuando el susto es activado. Casi todos los sustos tendrán adjuntados un sonido o impacto sonoro para asustar al jugador. También se hará uso de luces para que el susto sea mejor visiblemente y como no, la aparición de un amigurumi. Hay gran variedad de sustos en el juego ya sean con amigurumis, imágenes, vídeos o simplemente sonidos. En el anexo se enseñará la estructura con la que se han elaborado la gran mayoría. ([Anexo 6.1](#))

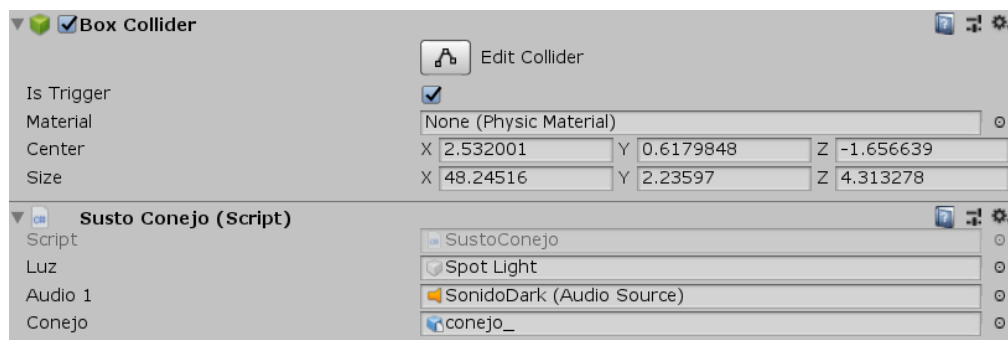


Figura 35: Estructura de los sustos

5.2.7 GameController y SettingsController

Estos dos elementos son los encargados del sistema de guardado del progreso del jugador de la partida y de las opciones. Por ejemplo, si el jugador deja de jugar e iba por el tercer nivel y tenía la sensibilidad y la música al mínimo, cuando el jugador vuelva a entrar al juego tendrá las mismas opciones y si decide continuar con la partida aparecerá en el nivel3. Para más información consulta el anexo. ([Anexo 7.1](#))

5.2.8 Otros

Este apartado va dirigido a aquellos eventos u objetos que suceden para una situación en concreto como puede ser el caso de un intensificador de la niebla del videojuego cuando el amigurumi atrapa al jugador o la aparición de ciertos objetos solamente cuando se da alguna cinemática.

5.3 Implantación del videojuego

Para el apartado de implantación, es decir el poner en marcha el videojuego, se han realizado dos entregas a un gran número de personas para poder recibir *feedback* y ver que fallos salen o qué hace falta reforzar. Todo esto a través de una encuesta que se les proporcionaba a aquellos que jugaban al juego.

En la primera entrega realizada el 8 de mayo del 2020 el juego fue probado por más de 30 personas, pero sólo se obtuvieron 27 encuestas.

En la primera encuesta estos fueron los resultados:

¿Has sentido miedo mientras jugabas?	Mucho: 0% Sí: 33,3% Un poco: 44,4% Nada: 22,2%
¿Visualmente, consideras que se ha conseguido una buena calidad?	Muy buena: 33,3% Buena: 66,6% Mala: 0% Muy mala: 0%
¿Te ha resultado fácil jugar a este juego?	Mucho: 0% Sí: 55,6% Me ha costado: 44,4% Me ha costado bastante: 0%
¿Te has entretenido el rato que has jugado?	Mucho: 11,1% Sí: 88,9% No: 0% Nada: 0%

Tabla 81: Primera encuesta de Amigurumi

Con estos resultados, lo que más chocó fue que un juego que supuestamente estaba hecho para ser un juego de miedo, no daba miedo. Por lo que se trabajó muy duro en el tema del miedo y por eso salieron los sustos. El juego dispone de más de 25 sustos cuando en esta primera entrega tenía solamente 2. En cuanto a la jugabilidad mucha gente se quejó de que le gustaría que apareciese las teclas que tenía que usar para ciertas cosas, porque no sabían cómo utilizar los objetos. También se añadió un apartado dentro del menú de pausa y el menú principal donde se pueden ver los controles.

También había dos preguntas de respuesta abierta en la que se les preguntaba a los jugadores si habían detectado algún fallo y qué mejoras introducirían. Aquí se comunicó sobre todo el problema de que las opciones y el progreso del jugador no se guardaba entre escenas ni mucho menos una vez que se apagaba y encendía el ordenador. También pequeños problemas como que al morir la primera vez por mirar al amigurumi no te podías mover cuando volvías a jugar, etc. Todos esos problemas han sido solucionados y las mejoras han sido integradas en el juego hoy en día.

El día 1 de junio de 2020 se volvió a realizar otra entrega del juego, esta vez ya terminado y en este caso se recibieron un total de 25 respuestas, pero con las mismas preguntas. Estos fueron los resultados:

¿Has sentido miedo mientras jugabas?	Mucho: 88% Sí: 12% Un poco: 0% Nada: 0%
¿Visualmente, consideras que se ha conseguido una buena calidad?	Muy buena: 60% Buena: 40% Mala: 0% Muy mala: 0%
¿Te ha resultado fácil jugar a este juego?	Mucho: 16% Sí: 64% Me ha costado: 20% Me ha costado bastante: 0%
¿Te has entretenido el rato que has jugado?	Mucho: 52% Sí: 48% No: 0% Nada: 0%

Tabla 82: Segunda encuesta de Amigurumi

En cuanto a las preguntas de respuestas abiertas simplemente se habían detectado algunos fallos de que en algún sitio se veía mal o que un objeto atravesaba a otro, etc. Los comentarios fueron muy buenos y los resultados muy positivos.

Capítulo 6

Pruebas

En este capítulo se enseñarán las pruebas que se han realizado durante el transcurso del videojuego. Se hará a través de tablas en las que se especificará el procedimiento de la prueba, el resultado esperado y si el resultado obtenido es correcto o incorrecto.

ID y nombre	PR-01 Estado patrulla
Procedimiento	<ol style="list-style-type: none">1. Observar que el amigurumi realiza el camino correctamente.2. Si ve directamente al jugador pasará a estado persecución.3. Si detecta al jugador en su cercanía pasará a estado alerta.
Resultado esperado	El amigurumi realiza correctamente el camino y pasa a ambos estados en las situaciones probadas.
Resultado obtenido	Correcto

Tabla 83: PR-01 Estado patrulla

ID y nombre	PR-02 Estado alerta
Procedimiento	<ol style="list-style-type: none">1. Observar que el amigurumi da vueltas sobre sí mismo durante el tiempo y a la velocidad determinada.2. Comprobar que el amigurumi pasa a estado patrulla tras el tiempo.3. Comprobar que el amigurumi pasa a estado persecución si ve al jugador.
Resultado esperado	El amigurumi gira correctamente dadas las condiciones de las pruebas y avanza sin problemas a los otros dos estados en los casos probados.
Resultado obtenido	Correcto

Tabla 84: PR-02 Estado alerta

ID y nombre	PR-03 Estado persecución
Procedimiento	<ol style="list-style-type: none">1. Observar que el amigurumi va a por el jugador.2. Comprobar que los ojos se encienden y la música cambia.3. Comprobar que, si el jugador se esconde, el amigurumi deja de perseguirlo y se activa el sonido de los latidos del personaje. El amigurumi pasa a estado alerta.4. Comprobar que si el amigurumi atrapa al jugador salta una cinemática seguida del cambio de escena al <i>game over</i>.
Resultado esperado	El amigurumi avanza a por el jugador con las características que atribuyen a este estado de una manera correcta. En caso de ser atrapado el jugador, se produce una cinemática y seguidamente aparece la pantalla de <i>game over</i> . Cuando el amigurumi pierde de vista al jugador este pasa a estado alerta.
Resultado obtenido	Correcto

Tabla 85: PR-03 Estado persecución



ID y nombre	PR-04 Perder cordura cuando el jugador mira al amigurumi
Procedimiento	<ol style="list-style-type: none"> 1. El jugador se queda mirando al amigurumi. 2. Comprobar que si la cordura llega a 0 se pierde la partida.
Resultado esperado	Mientras el jugador mira al amigurumi la cordura va descendiendo progresivamente hasta llegar a 0, cambiando de escena a la pantalla de <i>game over</i> .
Resultado obtenido	Correcto

Tabla 86: PR-04 Perder cordura cuando el jugador mira al amigurumi

ID y nombre	PR-05 El jugador es teletransportado al utilizar un teletransporte
Procedimiento	<ol style="list-style-type: none"> 1. El jugador va a través del objeto para teletransportarse. 2. El jugador aparece en el sitio indicado sin ningún problema.
Resultado esperado	El jugador se teletransporta de manera correcta sin perder ninguna característica que pueda alterar el funcionamiento del videojuego.
Resultado obtenido	Correcto

Tabla 87: PR-05 El jugador es teletransportado al utilizar un teletransporte

ID y nombre	PR-06 Interactuar con un objeto
Procedimiento	<ol style="list-style-type: none"> 1. El jugador avanza hacia un objeto con el que pueda interactuar. 2. El icono de interactuar y el texto de: <i>press e</i> aparece. 3. Una vez utilizada la tecla E el objeto realiza la función que ha sido programada.
Resultado esperado	Esta prueba ha sido repetida en varias ocasiones con todos los objetos con los que se pueda interactuar. El icono y el texto aparecen y la función realizada por cada objeto es la correcta en todos los casos dados.
Resultado obtenido	Correcto

Tabla 88: PR-06 Interactuar con un objeto

ID y nombre	PR-07 Subirse a un objeto
Procedimiento	<ol style="list-style-type: none"> 1. El jugador avanza hacia un objeto al que se pueda subir. 2. El icono de subir y el texto de: <i>press c</i> aparece. 3. Una vez utilizada la tecla C el jugador se sube al objeto.
Resultado esperado	Esta prueba ha sido repetida en varias ocasiones con todos los objetos con los que se pueda interactuar para subir. El icono y el texto aparecen y el jugador es capaz de subir a todos los objetos a los que pueda subirse de una manera correcta.
Resultado obtenido	Correcto

Tabla 89: PR-07 Subirse a un objeto

ID y nombre	PR-08 Empujar un objeto
Procedimiento	<ol style="list-style-type: none"> 1. El jugador avanza hacia un objeto al que se pueda empujar. 2. El texto de: <i>press s</i> o <i>press w</i> aparece. 3. Una vez utilizada la tecla W o S el jugador empuja al objeto dependiendo de si el jugador quiere empujar el objeto hacia delante o hacia detrás.
Resultado esperado	Esta prueba ha sido repetida en varias ocasiones con todos los objetos con los que se pueda interactuar para empujar. El texto aparece y el jugador empuja todos aquellos objetos que puedan ser empujados sin ningún problema.
Resultado obtenido	Correcto

Tabla 90: PR-08 Empujar un objeto

ID y nombre	PR-09 Menú principal
Procedimiento	<ol style="list-style-type: none"> 1. Comprobar que, si el jugador pulsa el botón de comenzar nueva partida y aparece en el nivel 1. 2. Comprobar que, si el jugador pulsa el botón de continuar y tiene una partida guardada avanza a ese momento de la partida. 3. Comprobar que, si el jugador pulsa el botón de continuar y no tiene una partida guardada no sucede nada. 4. Comprobar que, si el jugador pulsa el botón de opciones aparece el menú de opciones. 5. Comprobar que, si el jugador pulsa el botón de controles aparece el menú de controles. 6. Comprobar que, si el jugador pulsa el botón de salir, la aplicación se cierra
Resultado esperado	Todos los botones del menú principal realizan la función que tienen asignada de una manera correcta.
Resultado obtenido	Correcto

Tabla 91: PR-09 Menú principal

ID y nombre	PR-10 Sustos
Procedimiento	<ol style="list-style-type: none"> 1. El jugador entra en un área que detecta que el jugador ha entrado. 2. Comprobar que, una vez entrado en dicha zona, se activan los elementos correspondientes a cada susto. 3. Comprobar que, tras cierto tiempo el susto es desactivado y no vuelve a ocurrir.
Resultado esperado	Esta prueba ha sido repetida en varias ocasiones con todos los sustos. Todos los sustos se activan correctamente y se desactivan al tiempo especificado en cada <i>script</i> .
Resultado obtenido	Correcto

Tabla 92: PR-10 Sustos

ID y nombre	PR-11 Animaciones adecuadas
Procedimiento	1. Comprobar que, dependiendo del elemento, realiza las animaciones adecuados.
Resultado esperado	Esta prueba ha sido repetida en varias ocasiones con todas las animaciones. Todas las animaciones están hechas con sumo cuidado y su funcionamiento es adecuado.
Resultado obtenido	Correcto

Tabla 93: PR-11 Animaciones adecuadas

ID y nombre	PR-12 Sistema de guardado
Procedimiento	<ol style="list-style-type: none"> 1. El jugador empieza una nueva partida y cambia las opciones, por ejemplo, la sensibilidad vertical. 2. El jugador sale del juego, incluso puede apagar el ordenador. 3. El jugador vuelve a abrir el juego. 4. El jugador le da a continuar y comprueba que ha aparecido en el nivel 1 y que los ajustes realizados a la sensibilidad vertical se han guardado.
Resultado esperado	El juego genera unos archivos del progreso y de opciones que permite que cuando se continua el juego se guarda el progreso de la partida. Incluso en el menú principal, aparecen guardadas las opciones que se han guardado en otros niveles.
Resultado obtenido	Correcto

Tabla 94: PR-12 Sistema de guardado

ID y nombre	PR-13 Prueba final. El videojuego se juega entero sin errores
Procedimiento	1. El jugador completa el juego probando todos los rincones.
Resultado esperado	No se detectan fallos en ningún aspecto del videojuego
Resultado obtenido	Correcto

Tabla 95: PR-13 Prueba final

ID y nombre	PR-14 Menú de pausa
Procedimiento	<ol style="list-style-type: none"> 1. Comprobar que, si el jugador pulsa el botón de volver a la partida, se desactiva el menú de pausa. 2. Comprobar que, si el jugador pulsa el botón de opciones aparece el menú de opciones. 3. Comprobar que, si el jugador pulsa el botón de controles aparece el menú de controles. 4. Comprobar que, si el jugador pulsa el botón de salir, se cambia de escena al menú principal.
Resultado esperado	Todos los botones del menú de pausa realizan la función que tienen asignada de una manera correcta.
Resultado obtenido	Correcto

Tabla 96: PR-14 Menú de pausa

ID y nombre	PR-15 Opciones
Procedimiento	<ol style="list-style-type: none"> 1. Comprobar que, si el jugador baja la sensibilidad horizontal, después se refleja en el juego. 2. Comprobar que, si el jugador baja la sensibilidad vertical, después se refleja en el juego. 3. Comprobar que, si el jugador baja la música, después se refleja en el juego. 4. Comprobar que, si el jugador pulsa el botón de guardar las opciones se quedan guardadas. 5. Comprobar que, si el jugador pulsa el botón de volver, volverá al menú de pausa.
Resultado esperado	Cuando se entra al juego, el movimiento de la cámara es más lento y la música se escucha más baja. Las funcionalidades de los botones se corresponden con su objetivo.
Resultado obtenido	Correcto

Tabla 97: PR-15 Opciones

Capítulo 7

Conclusiones

Cuando se comenzó a trabajar con este videojuego en la asignatura de introducción a la programación de videojuegos, pensaba que al menos varios compañeros iban a seguirme en esta aventura para que el trabajo fuese más fácil y llevadero. Desgraciadamente, este proyecto ha sido desarrollado por una persona, lo que ha hecho que me dé cuenta de lo costoso que es la realización de un videojuego ya que son varios elementos que integrar en un proyecto, arte, música, programación, etc.

No obstante, creo que ha sido una gran decisión el realizar este proyecto como Trabajo de Fin de Grado. Es una gran decisión porque he afrontado este reto con muchísimo ilusión e interés, me ha hecho darme cuenta de que este es el mundo en el que quiero estar trabajando. He estado siempre atento a lo que la gente me sugería o decía y tratar de que todo fuesen mejoras para el juego. También es obvio que este proyecto me ha permitido obtener unos conocimientos sobre Unity3D y C# que nunca pensé que podía llegar a tener.

La realización de una memoria también me ha sido útil para comprender qué debe realizar un ingeniero de software a la hora de desarrollar un proyecto de esta magnitud, permitiendo aprender los pasos a seguir en futuros proyectos, y poniendo en práctica lo aprendido durante la carrera.

Es cierto que ha habido mil problemas tanto con el diseño del juego como con la programación. En el caso de diseño todo ha sido un poco más caótico porque al no estar tan acostumbrado a este tipo de cosas y ser prácticamente un principiante, se ha cambiado bastante de idea en torno a varios elementos del juego. En el caso de la programación no había día con el que me acostase con un nuevo error, pero increíblemente hoy en día me costaría encontrar alguno y, tras haber compartido el juego con más de 30 personas y no haber detectado ningún fallo dudo que pueda surgir algún contratiempo. También me gustaría remarcar que gracias a toda la documentación ofrecida por Unity3D y su increíble comunidad en los foros ha hecho que los fallos en el videojuego durasen escasas horas ocasionando problemas.

Sinceramente, el resultado de este proyecto es algo que me llega a emocionar, es algo que nunca pensé que podía llegar a hacer, es una prueba de que soy capaz de superarme y que en el futuro me volveré a superar.

Capítulo 8

Trabajos futuros

En el futuro, está previsto:

- **Mejorar las animaciones:** dado que solo ha habido un programador y no había gente especializada en el tema de las animaciones hubiese estado bien por contar con la ayuda de profesionales del arte para añadir animaciones o mejorarlas para ser aún más terrorífico.
- **Complicar los niveles con mayores rompecabezas:** la mayoría de los juegos de miedo tienen que resolver una gran cantidad de rompecabezas con el plus del agobio de pensar que en algún momento te puede ocurrir algo. En el videojuego hay muchos rompecabezas, pero nada más difícil de encontrar unos números para un código.
- **Tener sólo una escena y ampliar la casa:** para mejorar el juego todo sería mejor si todo estuviese metido en una escena, es decir si no hubiese cambio de niveles. Todo sería un nivel y tendrías que ir derrotando a todos los amigurumis. Daría mucho más juego a la hora de dar sustos o de manejar a los amigurumis por la casa.
- **Lanzar el juego a una tienda online:** si el juego sigue adelante y se puede buscar un equipo para mejorar el juego, seguramente el juego sea lanzado en la tienda de Steam, ya que es de las mejores tiendas *online* de videojuegos y el hecho de subir un juego no tiene más complicación que abonar un precio determinado.
- **Poder trabajar con un equipo:** Para que todo lo comentado anteriormente se pueda llevar a cabo, es necesario un equipo compuesto por gente de programación y de arte. Si algo se ha aprendido de este proyecto es que, un videojuego difícilmente puede ser llevado por una persona a no ser que se tengan los conocimientos necesarios y tiempo de sobra para dedicárselo.



Capítulo 9

Bibliografía

Este capítulo contiene la bibliografía a la que se ha recurrido para realizar el proyecto:

- [1] Llorente y Cuenca (AEVI). *El sector de los videojuegos en España: impacto económico y escenarios fiscales*. 2018.
Disponible en: http://www.aevi.org.es/web/wp-content/uploads/2018/01/1801_AEVI_EstudioEconomico.pdf
- [2] Cruz-Palacios, E.; Marzal García-Quismondo, M. A. *Gaming para las Alfabetizaciones Múltiples: Videojuegos en la Educación del Siglo XXI*. 2017. V Congreso Internacional de Videojuegos y Educación (Puerto de la Cruz, Santa Cruz de Tenerife, España, 7-9 de junio de 2017).
- [3] Asociación Española de Distribuidores y Editores de Software de Entretenimiento. *El videojugador español: perfil, hábitos e inquietudes de nuestros gamers*. 2011. Disponible en: http://www.aevi.org.es/pdf/EstilodeVidayvaloresdelosjugadoresdevideojuegos_resumenpresentacion.pdf
- [4] Schell, J. *The Art of Game Design: A book of lenses*. CRC Press (2017).

ANEXO

Aquí se procederá a explicar y detallar esas partes de código expuestas en el apartado de desarrollo e implantación del videojuego.

1.1 Cambiar cursor predeterminado por otro

Primero para desaparecer al cursor predeterminado del ordenador y mantenerlo en el centro para manejar de una manera correcta la cámara se realizará lo siguiente:

```
Cursor.lockState = CursorLockMode.Locked;  
Cursor.visible = false;
```

Una vez que está oculto el cursor, hay que crear el cursor original a través de una textura 2D y asignarle los valores del cursor para que se comporte como tal.

```
private void OnGUI()  
{  
    GUI.DrawTexture(new Rect(Input.mousePosition.x - 30f, Screen.height -  
        Input.mousePosition.y - 30f, tamaño, tamaño), cursor);  
}
```

Para activar el cursor una vez que se haya activado el menú de pausa simplemente se llama a un método que cambia la textura del cursor a uno que se le pasa y cuando se desactiva el cursor se cambia a nada. De manera que cuando el jugador esté fuera del menú de pausa no verá el cursor.

```
public void CambiarCursor(int i)  
{  
    if (i == 0)  
    {  
        cursor = newCursor;  
    }  
    else if (i == 1)  
    {  
        cursor = nada;  
    }  
}
```

1.2 Luz encendida y animaciones acabadas

Para el final de todos los niveles se deberá llamar a los métodos `luzEncendida()` y `acabadoAnimacion()`. El primero se utiliza para encender la luz del objeto que hará que se derrote al amigurumi.

```
public void luzEncendida()
{
    Player.controllable = false;
    levelComplete = true;
    enemy.GetComponent<MaquinaDeEstados>().morir();
    animOso.SetBool("muerto", true);
    StartCoroutine("fadeOutMusic", 2f);
}
```

El jugador no podrá moverse ya que se iniciará una cinemática y como se ha derrotado al amigurumi, el nivel pasa a estar completado y el enemigo pasa al estado de morir.

```
public void acabadoAnimacion()
{
    Player.controllable = true;
    menuControl.cinematicMode(false);
    mothExplosion.Play();
    StartCoroutine(ExecuteAfterTime(6.0f));
    GameObject amigurimi = GameObject.Find("/Enemigo_NV1");
    amigurimi.SetActive(false);
}
```

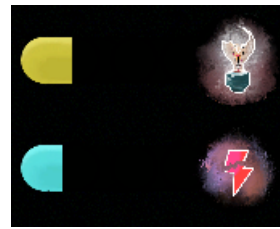
Una vez acabada la animación habrá que volver a activar el jugador y activar la explosión de polillas. Se elimina al amigurumi de la escena y se inicia una corutina en la que al cabo de un tiempo se cambia la música y sale un texto informando al jugador que tiene que dirigirse hacia la puerta. El término corutina¹¹ hace referencia a un trozo de código que el programador quiere que se ejecute al cabo de un tiempo determinado, en este caso 6 segundos.

¹¹ <https://docs.unity3d.com/es/2018.4/Manual/Coroutines.html>

1.3 Actualizar cordura y energía

Este método está escrito en MainControl pero es llamado desde Player. Básicamente se verá cómo se actualiza el valor y el dibujo de la barra en el caso de la cordura. En el caso de la energía es exactamente lo mismo.

```
public void UpdateCordura(float value)
{
    if(value > 0.3 && bajaCordura)
    {
        corduraFondo.sprite = HighCordura;
        bajaCordura = false;
    }
    else if(value <= 0.3 && !bajaCordura)
    {
        corduraFondo.sprite = LowCordura;
        bajaCordura = true;
    }
    float v = value * 0.9f;
    barra_cordura.value = v;
}
```



2.1 Detectar si un objeto puede ser interactuado y acciones

Esto es de lo más complejo de todo el proyecto, el hecho de que el bebé detecte que puede interactuar con un objeto y realizar la acción correspondiente.

```
ray = camera.ScreenPointToRay(Input.mousePosition);
if (Physics.Raycast(ray, out hit, distance))
{
    obj = hit.collider.gameObject;
```

Para empezar con la variable ray se guarda lo que ve el jugador a través de la cámara, entonces si colisiona a cierta distancia con un objeto, este es guardado en la variable obj.

```
if (obj.CompareTag("Interactable"))
{
    if (func == null)
    {
        func = obj.GetComponent<Interactable>();
    }

    if (func.interactable(hit))
    {
        mc.setInteractuarVisible(true);
        if (Input.GetButtonDown("interactuar") &&
            state.Equals (Estado.MOVE) && controllable)
        {
            func.OnInteraction(ray, hit, 0);
        }
    }
}
```

En esta parte se comprueba que el objeto puede utilizarse y se comprueba que la función que tiene es la interactuar con ese objeto. Por lo tanto, se activará el icono de que se puede interactuar con dicho objeto y una vez que se utilice el botón para interactuar comenzará la interacción.

```
else
{
    mc.setInteractuarVisible(false);
}
if (func.subible(hit))
{
    mc.setSubirVisible(true);
    if (Input.GetKey(KeyCode.C) && controllable)
    {
        func.OnInteraction(ray, hit, 1);
    }
}
else
{
    mc.setSubirVisible(false);
}
```

Aquí lo mismo para el caso de subir.


```

public void saltar( Vector3 startPos,Vector3 endPos)
{
    sonido.Stop();
    controllable = false;
    if (state == Estado.PUSH)
    {
        animator.speed = 1;
        cameraAnimator.speed = 1;
        pushObj.isKinematic = true;
        state = Estado.MOVE;
    }
    if (!enPie)
    {
        enPie = true;
        ponerEnPie(true);
        Debug.Log(enPie);
    }

    setAnimation("enPie", null);
    StartCoroutine(subir(1f, startPos,endPos));
}

```

Para las acciones se llevará un esquema bastante similar, en el que primero se utiliza un método como el de arriba para comprobar el estado y la situación en la que está el jugador para luego proceder con la corutina de la acción, en este caso subir.

```

IEnumerator subir(float seconds, Vector3 startPos,Vector3 endPos)
{
    aniCount++;
    while (isAnimating)
    {
        yield return new WaitForSeconds(this.seconds - time);
    }
    isAnimating = true;
    time = 0;
    this.seconds = seconds;
    Vector3 movement = Vector3.zero;
    movement = startPos - transform.position;
    movement.y -= 1f;
    Vector3 offset =
        transform.InverseTransformPoint(camera.transform.position);
    camera.transform.eulerAngles = new Vector3(0, 0, 0);
    camera.transform.parent = myPos.parent;
    animator.SetTrigger("subir");
    while (time < 0.5)
    {
        time += Time.deltaTime;
        controller.Move(movement * (Time.deltaTime / 0.5f));

        yield return null;
    }
    movement = endPos;
    movement.y = 1f;
    time = 0;
    seconds -= 0.5f;
    while (time < seconds)
    {
        time += Time.deltaTime;

```



```

        controller.Move(movement * (Time.deltaTime / seconds));

        yield return null;
    }
    movement.y = 0;
    movement *= 2f;
    camera.transform.parent = cameraAnimator.transform;
    camera.transform.position = transform.TransformPoint(offset);
    camera.transform.eulerAngles = new Vector3(0, 0, 0);
    controller.Move(movement);

    isAnimating = false;
    if(--aniCount == 0)
    {
        controllable = true;
    }
}

```

Lo que se realiza en este método por muy extenso que parezca es ajustar todas las posiciones tanto de la cámara como del jugador para que una vez el jugador haya subido a un lugar no haya ningún fallo de posición y el jugador no sufra ninguna irregularidad.

2.2 Movimiento del personaje y aspectos de la cámara

Para el tema de la cámara se buscará que el jugador tenga unos límites para girar la cámara de arriba abajo, pues ningún ser humano es capaz de hacer un giro completo de arriba abajo.

```

rotationy = speedH * Input.GetAxis("Mouse X");
rotationx = -speedV * Input.GetAxis("Mouse Y");
float rotationxNow = camera.transform.eulerAngles.x;
if (rotationxNow > 60) rotationxNow -= 360;
if (rotationxNow + rotationx < limitDown)
{
    rotationx = 0;
}
else if (rotationxNow + rotationx > limitUp)
{
    rotationx = 0;
}

```

Las variables speedH y speedV son las velocidades de la sensibilidad horizontal y vertical. Esto se hace para que la velocidad de la cámara vaya ajustada a la sensibilidad que elija el jugador dentro del menú de opciones.

```

if (controllable)
{
    horizontal = Input.GetAxis("Horizontal");
    vertical = Input.GetAxis("Vertical");
}

```

De esta manera el jugador ya puede empezar a moverse con las teclas W, A, S y D.

```

if (Input.GetButton("correr")&&canRun && energia >= 0)
{
    if (!enPie)
    {
        ponerEnPie(true);
    }
    enPie = true;
    gastandoEnergia = true;
}
else
{
    if (enPie)
    {
        ponerEnPie(false);
    }
    enPie = false;
    gastandoEnergia = false;
    if(energia <= 0)
    {
        canRun = false;
        StartCoroutine(activarRun(3.0f));
    }
}

```

Aquí se puede ver como se comprueba que si el jugador está en movimiento y pulsa el botón de correr y tiene energía se pondrá de pie y podrá correr, de lo contrario volverá a gatear.

Este script se basa en esta misma estructura para todos los tipos de movimiento. Desde el método Update() o FixedUpdate() se llamará a una acción que tiene su propio método que a su vez llama al método para realizar la animación.

3.1 Bocadillos que cuentan la historia

Este script es bastante sencillo, simplemente se realizan unas corutinas como se verá con el apartado de los sustos.

```

void Start()
{
    imagen1.enabled = false;
    text1.enabled = false;
    imagen2.enabled = false;
    text2.enabled = false;
    imagen3.enabled = false;
    text3.enabled = false;
    StartCoroutine(ExecuteAfterTime(1.0f));
}

```



En el método Start() se ocultan todos los elementos y se inicia la corutina.

```
IEnumerator ExecuteAfterTime(float time)
{
    yield return new WaitForSeconds(time);

    // Code to execute after the delay
    imagen1.enabled = true;
    text1.enabled = true;

    yield return new WaitForSeconds(time + 5.0f);
    imagen1.enabled = false;
    text1.enabled = false;

    yield return new WaitForSeconds(time);
    imagen2.enabled = true;
    text2.enabled = true;

    ...
}
```

En la corutina se activa el primer texto con la primera burbuja y al cabo de un tiempo se oculta a la vez que se empieza a activar la segunda burbuja y así con todos los elementos.

3.2 Menú de pausa

Para que el menú de pausa se active primero habrá que pulsar la tecla ESCAPE.

```
void Update()
{
    if (!menu.enabled)
    {
        if (Input.GetKey(KeyCode.Escape))
        {
            CambiarCursor(1);
            pausa();
        }
    }
    if(menu.enabled)
    {
        if (Input.GetKey(KeyCode.Escape))
        {
            pausa();
            CambiarCursor(0);
        }
    }
}
```

Como se ha visto antes, se hace uso del método CambiarCursor(int i) cuando se activa y se desactiva el menú de pausa.

```
public void pausa()
{
    if (pausaEnable)
    {
        menu.enabled = !menu.enabled;
    }
}
```

```

        ui.enabled = !ui.enabled;
        pausaEnable = false;
        if (menu.enabled)
        {
            Cursor.lockState = CursorLockMode.Confined;
            Time.timeScale = 0;
        }
        else
        {
            Cursor.lockState = CursorLockMode.Locked;
            Cursor.visible = false;
            CambiarCursor(1);
            Time.timeScale = 1;
        }

        StartCoroutine("hacerAnimacion", 1);
    }
}

```

Con este método se activa y se desactiva el menú de pausa.

3.3 Cambiar de nivel con transición

Este proceso es bastante sencillo, Unity tiene clases específicas para poder realizar este tipo de cambios de escenas.

```

public void FadeToLevel(int indexLevel)
{
    Scene currentScene = SceneManager.GetActiveScene();
    int buildIndex = currentScene.buildIndex;
    if(buildIndex == 0)
    {
        cascos.SetActive(false);
        texto.SetActive(false);
    }
    levelLoaded = indexLevel;
    animator.SetTrigger("fadeOut");
    Time.timeScale = 1;
}

```

Cada escena tiene un índice que se puede cambiar desde el programa de Unity. Lo que en este método se hace es coger el índice de la escena y si coincide con la del menú principal se ocultan dos elementos al pasar de escena. Quitando este detalle, la función general es cuando obtiene el nivel al que cambiar y aplica la animación fadeOut.

```

public void OnFadeComplete()
{
    Cursor.lockState = CursorLockMode.None;

    SceneManager.LoadScene(levelLoaded);
}

```

Entonces se activa este método y se carga la escena que se le había pasado en el anterior método.

4.1 Controlador de la visión del amigurumi

Para el controlador de la visión, el amigurumi tendrá un objeto llamado `DetectorBebe` que es invisible pero que está a la altura del bebé y se utiliza como bien dice su nombre para detectar al bebé.

```
public bool PuedeVerAlJugador(out RaycastHit hit, bool mirarHaciaElJugador =
false)
{
    Vector3 vectorDireccion;
    if (mirarHaciaElJugador)
    {
        vectorDireccion = (controladorNavMesh.perseguirObjetivo.position +
offset) - DetectorBebe.position;
    }
    else
    {
        vectorDireccion = DetectorBebe.forward;
    }
    return Physics.Raycast(DetectorBebe.position, vectorDireccion, out hit,
rangoVision) && hit.collider.CompareTag("Player");
}
```

Este método se utiliza para comprobar si el amigurumi puede ver al jugador, para ello se calcula el vector de la dirección entre el amigurumi y el jugador. En caso de que el amigurumi mire al jugador dará true.

4.2 Controlador del amigurumi

Este script está diseñado para poder manejar desde código al agente del amigurumi.

```
public void ActualizarPuntoDestinoNavMeshAgent(Vector3 puntoDestino)
{
    navMeshAgent.destination = puntoDestino;
    navMeshAgent.Resume();
}

public void ActualizarPuntoDestinoNavMeshAgent()
{
    ActualizarPuntoDestinoNavMeshAgent(perseguirObjetivo.position);
}
```

El primer método sirve para actualizar el punto destino que se le pase como puede ser un punto del camino que tiene que hacer, el segundo método sirve para indicarle que su siguiente punto de destino es el jugador.

```

public void DetenerNavMeshAgent()
{
    navMeshAgent.Stop();
}

public bool HemosLlegado()
{
    return navMeshAgent.remainingDistance <= navMeshAgent.stoppingDistance &&
!navMeshAgent.pathPending;
}

```

También habrá un método para parar al amigurumi y otro para hacerle saber al amigurumi que ya ha llegado hacia dónde quería ir y que hay que actualizar el punto de destino.

4.3 Enemy

Este script se utiliza para que en el caso de que el jugador tenga contacto visual con el amigurumi entonces empiece a perder cordura.

```

void OnWillRenderObject()
{
    if (Physics.Raycast(transform.position, player.myPos.position -
transform.position , out hit))
    {
        Debug.DrawLine(transform.position, (player.myPos.position -
transform.position) * hit.distance, Color.yellow);
        obj = hit.collider.gameObject;
        if (obj.tag == "Player" && !MainControl.levelComplete)
        {
            player.recibirDano(transform.position);
        }
    }
}

```

El método `OnWillRenderObject()`¹² se utiliza para captar aquello que ves por la cámara del jugador de la manera que en el momento que el jugador vea al amigurumi a una distancia considerable empezará a perder cordura.

```

public void recibirDano(Vector3 ePos)
{
    float dist = Vector3.Distance(player.transform.position,
enemy.transform.position);
    Vector3 pos = camera.WorldToViewportPoint(ePos);
    if(dist < 40f)
    {
        if (pos.x < wMax && pos.x > wMin && pos.y > hMin && pos.y < hMax &&
pos.z >= 0)
        {
            recibiendoDano = true;
        }
    }
}

```

¹²

<https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnWillRenderObject.html>



Aquí arriba está el método en Player por el cual se activa la propiedad `rebiendoDano` que desencadena el perder cordura dentro del videojuego.

4.4 Estado patrulla

Como ya se ha comentado varias veces en este documento el estado patrulla consiste en el camino del amigurumi por la habitación hasta que detecta al jugador cerca suyo o ve directamente al jugador.

```
void Update()
{
    //Ve al jugador??
    RaycastHit hit;
    agent.speed = 8f;
    if(controladorVision.PuedeVerAlJugador(out hit))
    {
        controladorNavMesh.perseguirObjetivo = hit.transform;
        maquinaDeEstados.ActivarEstado(maquinaDeEstados.EstadoPersecucion);
        return;
    }

    if (controladorNavMesh.HemosLlegado())
    {
        siguienteWayPoint = (siguienteWayPoint + 1) % WayPoints.Length;
        ActualizarWayPointDestino();
    }
}
```

Si el controlador de la visión choca con el jugador activará el estado persecución, de lo contrario seguirá su camino e irá actualizando los puntos en el camino o *waypoints*.

```
void ActualizarWayPointDestino()
{
    controladorNavMesh.ActualizarPuntoDestinoNavMeshAgent(WayPoints[siguienteWayPoint].position);
}

public void OnTriggerEnter(Collider other)
{
    if (other.gameObject.CompareTag("Player") && enabled)
    {
        maquinaDeEstados.ActivarEstado(maquinaDeEstados.EstadoAlerta);
    }
}
```

Este script tendrá un método para ir actualizando los puntos del camino utilizando el controlador del agente. Además, como también se ha dicho en este documento, el amigurumi tiene una esfera invisible con la que detecta si el jugador está

cerca. Entonces el método `OnTriggerEnter` sirve para que en caso de que el jugador entre en esta esfera invisible, el amigurumi pasará al estado alerta.

4.5 Estado alerta

El estado más sencillo sin duda, el amigurumi gira sobre sí mismo, si encuentra al jugador lo persigue, de lo contrario pasado un tiempo volverá al estado patrulla.

```
void Update()
{
    RaycastHit hit;
    if (controladorVision.PuedeVerAlJugador(out hit))
    {
        controladorNavMesh.perseguirObjetivo = hit.transform;
        maquinaDeEstados.ActivarEstado(maquinaDeEstados.EstadoPersecucion);
        return;
    }

    transform.Rotate(0f, velocidadGiroBusqueda * Time.deltaTime, 0f);
    tiempoBuscando += Time.deltaTime;
    if(tiempoBuscando >= duracionBusqueda)
    {
        maquinaDeEstados.ActivarEstado(maquinaDeEstados.EstadoPatrulla);
    }
}
```

4.6 Estado persecución

El estado persecución es el más complejo debido a que cuando entra en persecución, los ojos del amigurumi se iluminan, la velocidad aumenta, se activa una música de persecución, cabe la posibilidad de que te coja y se pierda, etc. A continuación, se irá parte por parte explicando los detalles.

```
void Update()
{
    myAnimator.SetBool("persecucion", true);

    materialOjos = new Material[] {skin.material, materialYellow };
    skin.materials = materialOjos;
    latidos.SetActive(false);
    RaycastHit hit;
    agent.speed = 13f;
    musicaPersecucion.enabled = true;

    float dist = Vector3.Distance(player.transform.position,
    agent.transform.position);
}
```

Cuando se entra en el estado persecución, el controlador de las animaciones del amigurumi avisa a este de que tiene que cambiar de animación a perseguir. Además de que se produzcan todos los eventos pronunciados en el inicio de este punto.



```

if (dist < 5f)
{
    Player.controllable = false;
    playerEliminado.SetActive(false);
    playerAnimado.SetActive(true);
    barra1.SetActive(false);
    barra2.SetActive(false);
    getCaught.Play();
    StartCoroutine(ExecuteAfterTime2(5.4f));
}

if (!controladorVision.PuedeVerAlJugador(out hit, true))
{
    materialOjos2 = new Material[] { skin.material, normal };
    skin.materials = materialOjos2;
    myAnimator.SetBool("persecucion", false);
    maquinaDeEstados.ActivarEstado(maquinaDeEstados.EstadoAlerta);
    musicaPersecucion.enabled = false;
    StartCoroutine(ExecuteAfterTime(1.0f));

    return;
}

```

Una vez que la distancia entre jugador y amigurumi es mínima, el jugador queda inmovilizado, se utiliza un bebé especial para facilitar la animación y una vez quitados los elementos de la interfaz de usuario se procede con la animación. Finalizada la animación se inicia una corutina que cambia de escena a la pantalla de *game over*.

En caso de que el jugador se esconda del amigurumi y este pierda la visión, los ojos volverán a su aspecto normal y su controlador de animaciones avisará al amigurumi para cambiar su movimiento. Además, se quita la música de persecución y se activa un sonido de latidos haciendo sentir al jugador que ha pasado un mal rato. Por último, se activa el estado de alerta.

4.7 Máquina de estados

Por último, dentro del amigurumi se encuentra la máquina de estados, el gestor de todos los estados y de los cambios entre ellos. En el Start() se activa el estado inicial, que siempre será el estado patrulla.

```

void Start() { ActivarEstado(EstadoInicial); }

public void ActivarEstado(MonoBehaviour nuevoEstado)
{
    if (estadoActual != null)
    {
        estadoActual.enabled = false;
    }
    estadoActual = nuevoEstado;
    estadoActual.enabled = true;
}

```

5.1 Ejemplo de objeto con el que se puede interactuar

Para empezar, para que el jugador pueda interactuar con un objeto. El objeto tiene que heredar de Interactuable.

```
public interface Interactuable
{
    void OnInteraction(Ray ray, RaycastHit hit, int control);
    bool subible(RaycastHit hit);
    bool interactuable(RaycastHit hit);
}
```

Tendrá 3 métodos, el primero sirve para programar lo que sucede después de que se haya interactuado con el objeto. Los otros dos sirven para comprobar cuando un objeto puede ser utilizado o puede ser subido. A continuación, se expondrá el script de una puerta del videojuego cuya función es la de una vez completado el nivel y recogidos el cubo y las llaves, el jugador podrá interactuar con la puerta y cambiar de nivel.

```
public bool interactuable(RaycastHit hit)
{
    return MainControl.levelComplete && !cubo.activeInHierarchy &&
!llave.activeInHierarchy;
}
public void OnInteraction(Ray ray, RaycastHit hit, int control)
{
    sonido.Play();
    Levelchanger.FadeToLevel(nivel);
}

public bool subible(RaycastHit hit)
{
    return false;
}
```

6.1 Ejemplo de susto

En cuanto a los sustos, es de los *scripts* más sencillos. Se hará uso de las corutinas y se jugará con el método setActive(bool) para activar elementos una vez que el jugador entra en una zona determinada (OnTriggerEnter()). Obviamente hay variaciones y cosas que se añaden en algunos *scripts* pero casi todos tienen estos elementos en común ya sean sonidos, luces, objetos, imágenes, etc.

```
public void OnTriggerEnter(Collider other) {
    if (other.gameObject.CompareTag("Player") && enabled) {
        audio1.enabled = true;
        efecto.SetActive(true);
        camaraAntigua.SetActive(false);
        camara.SetActive(true);
        Player.controllable = false;
        StartCoroutine(ExecuteAfterTime(0.45f));
    }
}
```



Este susto en concreto tiene que ver con la aparición de un susto del amigurumi en la cámara del jugador. Por lo que una vez entrado en la zona se activa un impacto sonoro, se activa una cámara de la que sale un amigurumi y el jugador queda inmóvil. Además, se produce un efecto en la pantalla de pestañear debido al susto y se inicia una corutina.

```
IEnumerator ExecuteAfterTime(float time)
{
    yield return new WaitForSeconds(time);
    camaraAntigua.SetActive(true);
    camara.SetActive(false);
    Player.controllable = true;
    susto.SetActive(false);
    yield return new WaitForSeconds(time + 3f);
    audio1.enabled = false;
    efecto.SetActive(false);
    risa.SetActive(true);
}
```

Pasado un tiempo se vuelve a activar la cámara que tenía el jugador por defecto y el jugador vuelve a poder ser controlado. Se desactiva el susto, los audios y demás y se activa una risa maléfica.

7.1 Sistema de guardado del videojuego

Tanto para el progreso del jugador como para el guardado de las opciones se utilizará el mismo esquema de *script*. En este caso se expondrá el ejemplo del guardado del progreso del jugador. A continuación, se muestra la clase `PlayerData` que guardará la última escena en la que estuvo el jugador.

```
public class PlayerData
{
    public int lastScene;
    public PlayerData (GameController gc)
    {
        lastScene = gc.lastScene;
    }
}
```

Para empezar Unity no dispone de ningún sistema de guardado automático a no ser que el jugador no salga del juego, es decir, que en el momento que el jugador sale del juego todo progreso se pierde. Para que esto no suceda, se ha de generar un archivo binario dentro de la carpeta del juego para guardar su progreso.

```
public static void SavePlayer(GameController gc)
{
    BinaryFormatter formatter = new BinaryFormatter();
    string path = Application.persistentDataPath + "/player.fun";
    FileStream stream = new FileStream(path, FileMode.Create);
    PlayerData data = new PlayerData(gc);

    formatter.Serialize(stream, data);
    stream.Close();
}
```

Como se puede apreciar en el código de arriba, se crea un archivo binario con los datos proporcionados de `PlayerData` en el *path* correspondiente.

```
public static PlayerData LoadPlayer()
{
    string path = Application.persistentDataPath + "/player.fun";
    if (File.Exists(path))
    {
        BinaryFormatter formatter = new BinaryFormatter();
        FileStream stream = new FileStream(path, FileMode.Open);

        PlayerData data = formatter.Deserialize(stream) as PlayerData;
        stream.Close();

        return data;
    }
    else
    {
        return null;
    }
}
```

Para el caso de cargar datos, simplemente busca en el *path* correspondiente si existe tal archivo y en el caso de que exista se cargan los datos. Y de la misma manera para el caso de las opciones. Estos métodos de cargar y guardar se llamarán bien desde el método `Update()` del `MainControl` o bien utilizando botones desde la propia interfaz de usuario.



8. Estructura de Unity

Seguidamente se enseñará la estructura de Unity. En la figura 2.4 se puede apreciar lo que se vería al iniciar el programa.

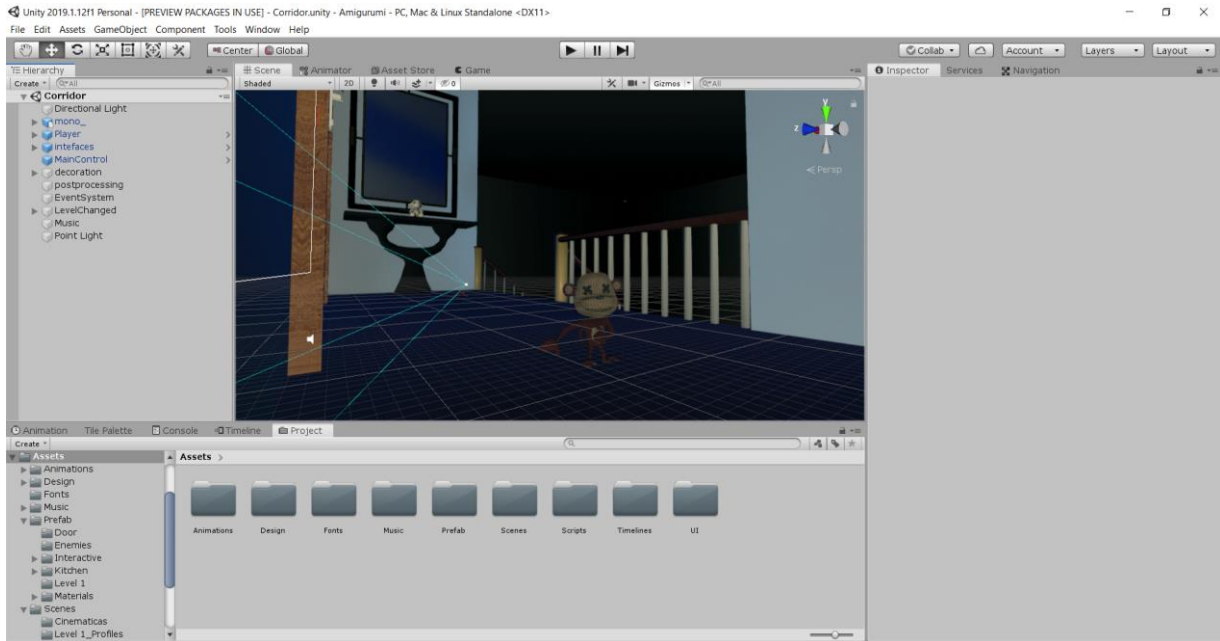


Figura 36: Escena principal de Unity

En el centro de la pantalla donde se puede ver el juego, se encuentra la pestaña de la **escena**, desde ahí es donde se insertan los elementos del juego llamados “GameObject” y se pueden modificar sus atributos o mover.

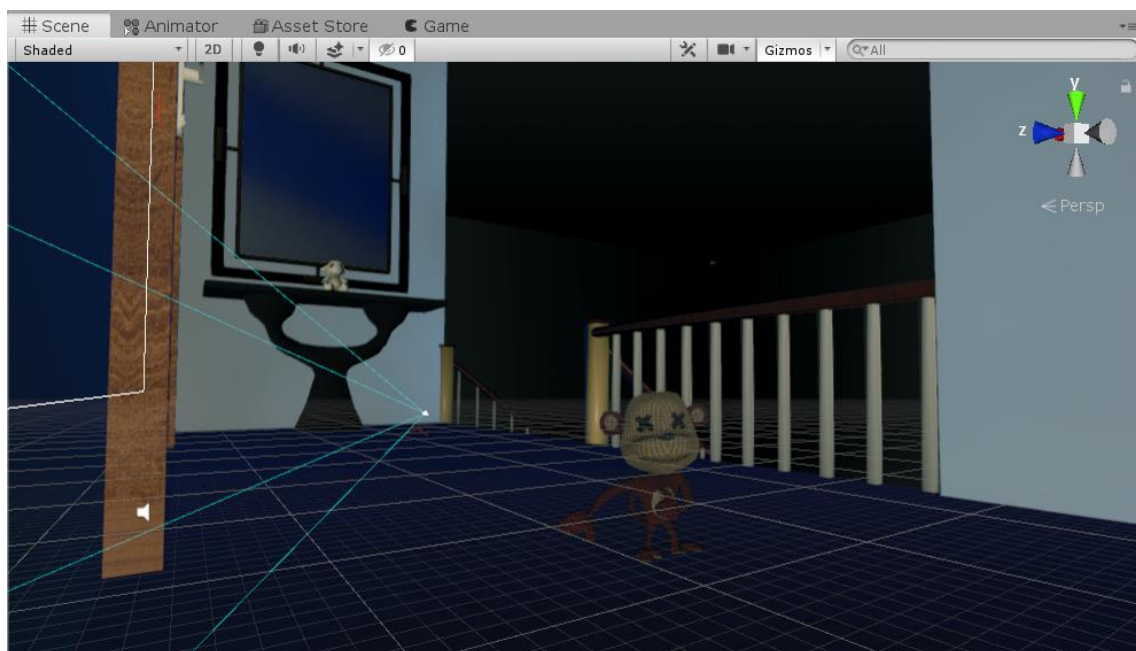


Figura 37: Pestaña de la escena

La ventana de **animación** sirve para crear y modificar animaciones para utilizarlas en la escena principal.

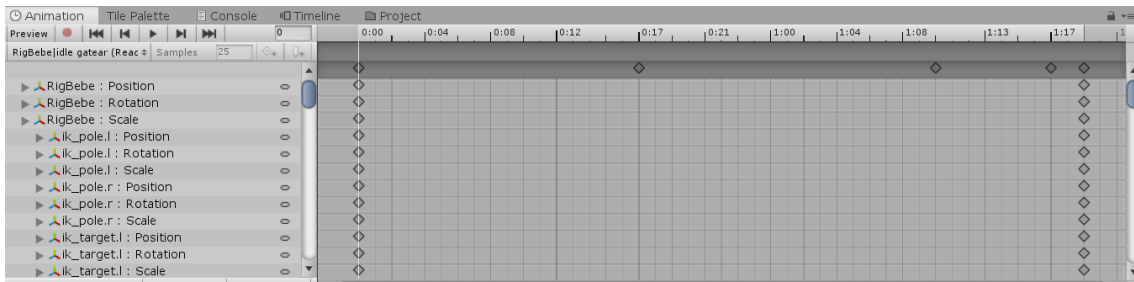


Figura 38: Pestaña de animaciones

La ventana de **jerarquía** enseña todos los “Game Objects” que hay en la escena del juego y permite acceder a ellos de una manera más rápida y sencilla.

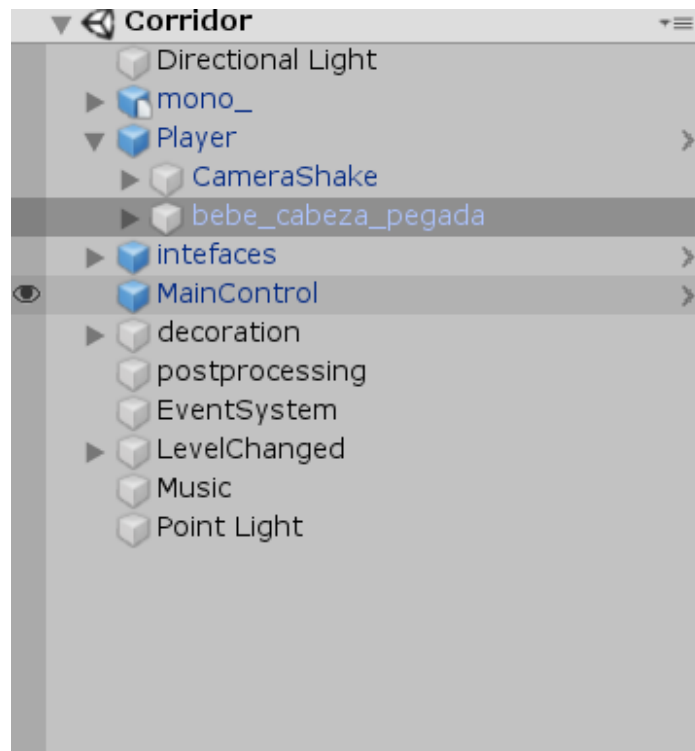


Figura 39: Pestaña de jerarquía

La parte más importante de todas es la del inspector, donde se pueden ver las propiedades del objeto seleccionado, así como su posición, *scripts*, etc.

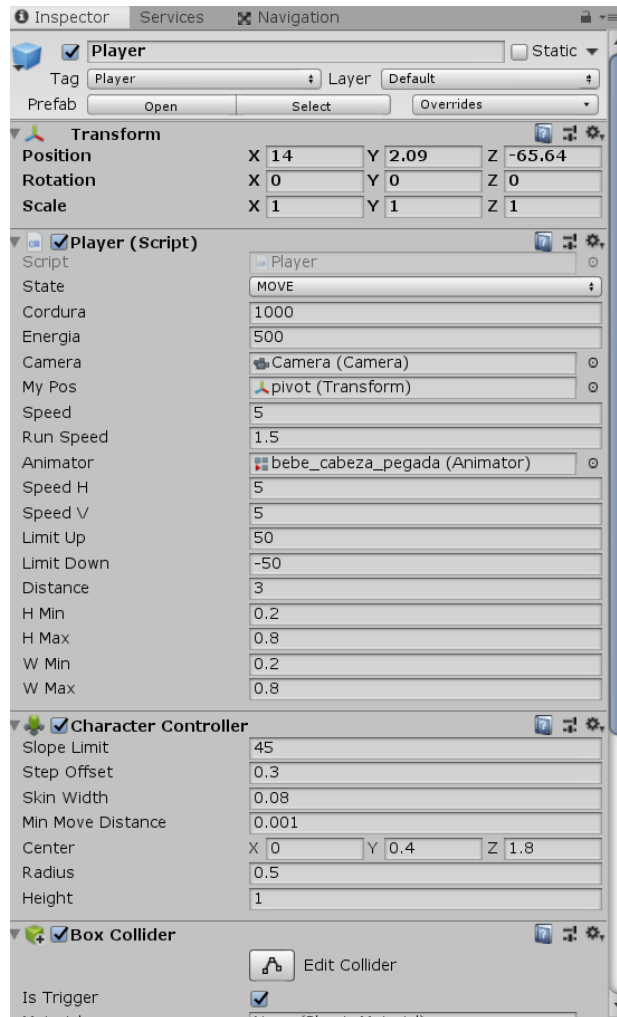


Figura 40: Pestaña de inspector

Por último, la ventana del proyecto donde se encuentra todas las carpetas de este tales como las animaciones, objetos varios, música, *scripts*, escenas, etc.

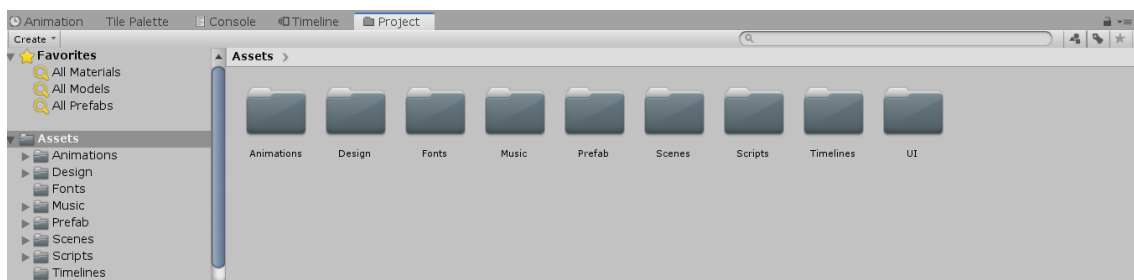


Figura 41: Pestaña del proyecto

