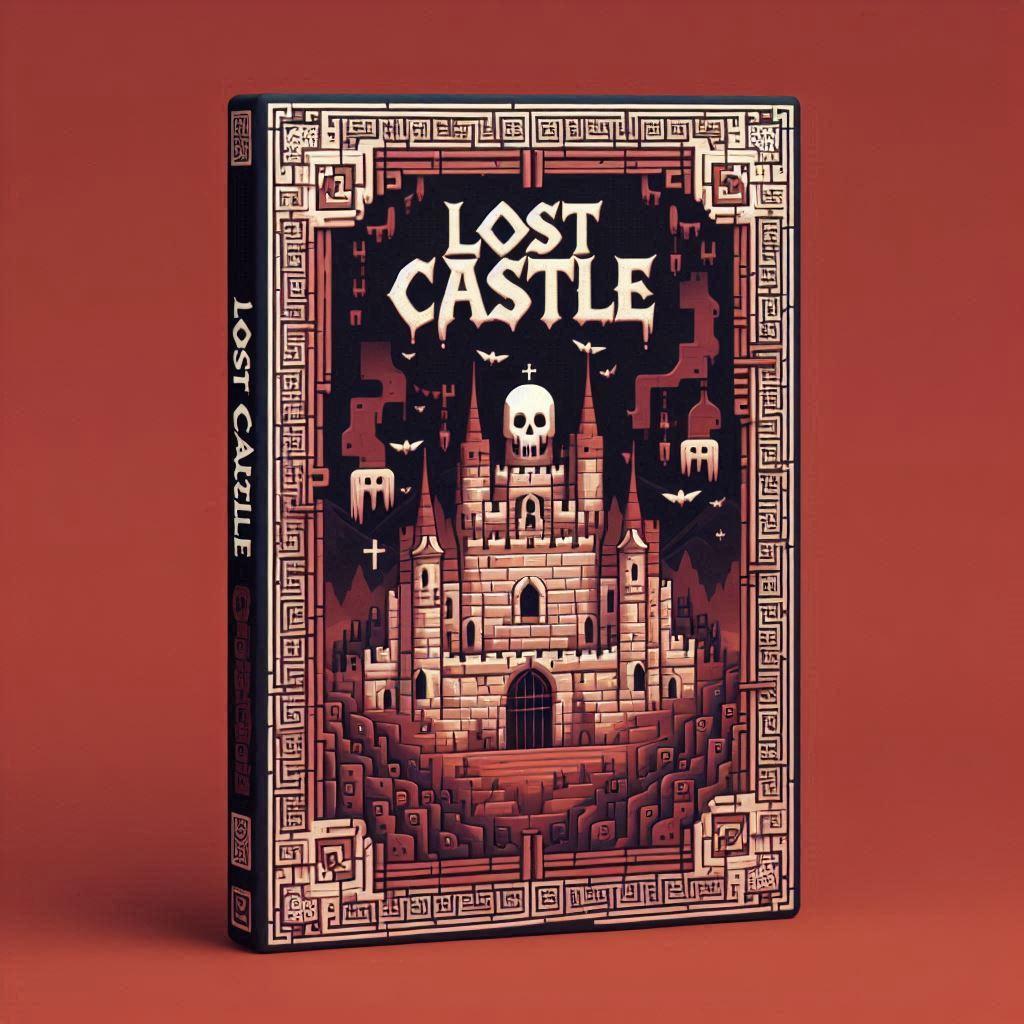
**José Luis González Álamo**

Grado Superior Desarrollo de Aplicaciones Multiplataforma

Lost Castle:  
Descubre secretos, resuelve enigmas. ¡Juega!

**\_\_\_**



[1. INTRODUCCIÓN](#_ezioxomjx0vv)

[2. Agradecimientos](#_7ucxaf5zvszr)

[3. Herramientas](#_9pquhyjvjtpn)

[3.1 Unity](#_aes90vf6q3qh)

[3.2 Microsoft Visual Studio](#_82pplku451pj)

[3.3 Aseprite](#_lamexza58ro6)

[3.4 Trello](#_to3x1neyb7li)

[3.4 Draw.IO](#_z5jor6yls03e)

[4. Análisis](#_lndlcyiqfmgq)

[4.1 Recolección de Influencias](#_e0bnbiulvu3v)

[4.2 Sesión de Brainstorming](#_upo9hqh5n5cf)

[4.3 Organización de Tareas](#_xvhmppgjhold)

[4.4 Búsqueda de recursos](#_eejaozl3n7pa)

[5. Diseño Técnico](#_xthvzz1h9lp6)

[6. Diagramas](#_um7hg79bwzu2)

[6.1 Diagrama Casos de Uso](#_eprvkpnure5p)

[6.2 Diagrama de Clases](#_lnr412vi7cfq)

[6.3 Diagrama Entidad-Relación](#_apd9qgotpho4)

[6.4 Diagrama de Componentes](#_t43o9q1bplwy)

[6.5 Casos de Pruebas](#_c3gx42mb2oan)

[7. Análisis Dafo](#_drdw7lg5wk23)

[8. Resultados y Conclusiones](#_tjgqgihbfzs0)

[9. Análisis de Impacto](#_895qmxbdte49)

[10. Despedida](#_awx8blx27q5c)

# INTRODUCCIÓN

Bienvenido al emocionante mundo de "Lost Castle", un juego de aventuras y misterios que te sumergirá en un viaje épico a través de los pasillos olvidados de un castillo ancestral. En este juego, los jugadores se embarcarán en una búsqueda llena de intriga, exploración y desafíos mientras desentrañan los secretos ocultos del castillo y descubren su fascinante historia.

"Lost Castle" ofrece una experiencia única donde los jugadores tendrán la oportunidad de sumergirse en un mundo cautivador lleno de maravillas y peligros. Con un diseño artístico encantador y una narrativa envolvente, este juego promete llevar a los jugadores a través de una aventura inolvidable llena de giros y vueltas inesperadas.

# Agradecimientos

Agradezco la ayuda a mi tutora María Fernanda Hoyuela de la Cueva por su esfuerzo y dedicación en que alcanzara el mejor trabajo posible, sin su apoyo, colaboración y tutoría durante este largo proyecto no habría sido capaz de alcanzar lo que he logrado hoy.

También quiero agradecer a toda mi familia por el apoyo mostrado durante este tiempo, han sido meses complicados y han estado ahí todos los días apoyándome y ayudándome siempre que lo necesitaba, sin ellos no habría llegado donde estoy a día de hoy.

# Herramientas

La creación de videojuegos se apoya en diversas herramientas que facilitan el proceso creativo. A continuación se detalla cómo funcionan cada una de estas herramientas y para qué se utilizan.

### 

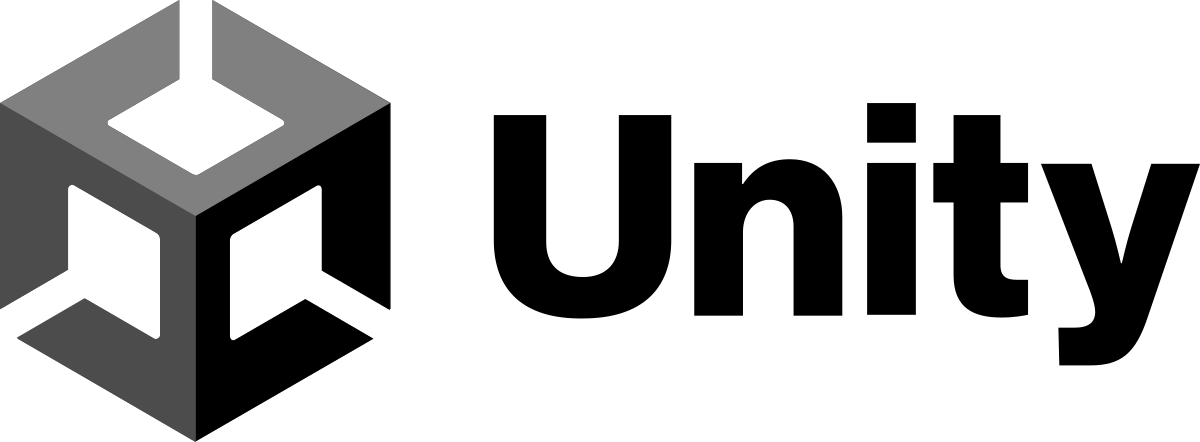
#### 3.1 Unity

Unity es un motor de videojuegos multiplataforma que ha sido la herramienta más utilizada durante el desarrollo. Se eligió este motor por dos razones principales. La primera es que, personalmente, ya tenía experiencia previa con su uso, lo que me permitía trabajar más rápidamente. Además, Unity cuenta con una gran comunidad debido a su popularidad en la industria, lo que facilita encontrar documentación y ayuda cuando es necesario. La versión de Unity se ha actualizado a lo largo del desarrollo, ya que varias de las nuevas versiones han ofrecido mejoras útiles para el ciclo de desarrollo, como un aumento significativo en la velocidad de compilación o el modo oscuro. Actualmente, se está utilizando la versión 2021.3.4f1.

También se ha usado Unity HUB, una herramienta que centraliza todas las versiones de Unity instaladas, simplificando el manejo correcto de los diferentes proyectos en los que se trabaja simultáneamente.

Unity cuenta además con la Asset Store, donde se pueden descargar diversos elementos creados por la comunidad, desde paquetes de texturas hasta módulos que añaden nuevas funcionalidades a Unity. De aquí se descargaron algunos paquetes de iconos para comenzar a programar las mecánicas sin necesidad de crear el arte previamente.

Finalmente, también existe un gestor de paquetes que permite añadir diversas funcionalidades al proyecto. En este caso, se ha utilizado Cinemachine para mejorar el funcionamiento de la cámara del juego.



#### 3.2 Microsoft Visual Studio

La herramienta Visual Studio, desarrollada por Microsoft, ha sido utilizada en su versión Community como el IDE para programar los diversos scripts del proyecto. Aunque Unity utiliza por defecto la versión de 2019, ofreciendo la opción de instalarlo junto con el motor, se optó por la versión de 2022 debido a sus mejoras en el software de autocompletado de código, IntelliCode, y por su mejor rendimiento.



#### **3.3 Aseprite**

Aseprite es un editor de gráficos rasterizados que se ha utilizado para crear partes de las imágenes del videojuego. Este editor es muy popular para crear gráficos en estilo pixel art y se eligió para este proyecto debido a su facilidad de uso y la gran variedad de opciones que ofrece.



#### **3.4 Trello**

Trello es un software de gestión de proyectos basado en la web que se ha utilizado para planificar las tareas del proyecto. Aunque es uno de los más utilizados y está principalmente orientado a equipos, también ha sido útil para trabajar en solitario, ya que ayuda a recordar los detalles de cómo se implementaron ciertas características semanas o meses después.



### 

#### **3.4 Draw.IO**

Draw.io es un software de creación de gráficos que se ha utilizado para elaborar el diagrama de estados del videojuego descrito en esta memoria. Aunque inicialmente se planeó usarlo para crear los diagramas de clases, finalmente se optó por el módulo de Visual Studio, ya que resultó más fácil de usar para ese propósito.



# Análisis

En el desarrollo de un videojuego, es crucial definir su alcance de manera temprana para evitar problemas de diseño en etapas avanzadas. Por ello, se inició con una fase de análisis, en la que se estructuró el diseño general del videojuego.

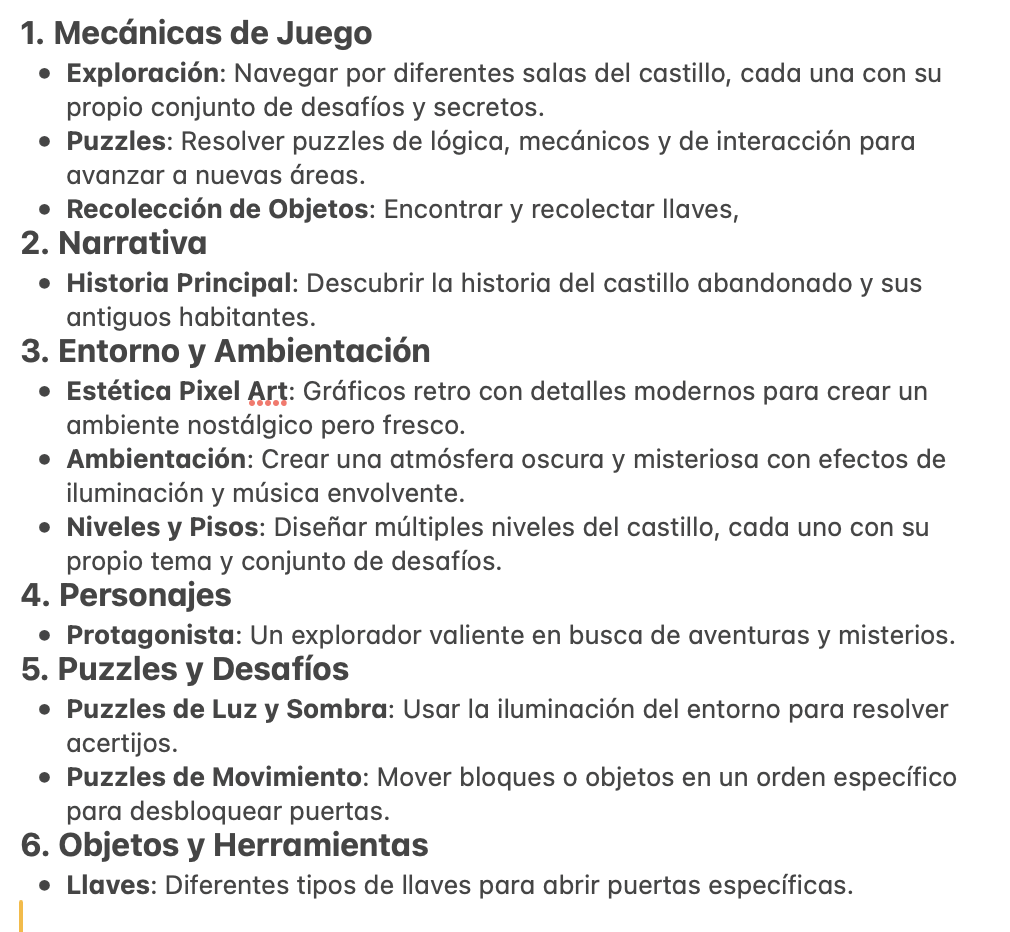
## 4.1 Recolección de Influencias

Lo primero que se hizo, para comprender correctamente el concepto del videojuego a desarrollar, fue experimentar brevemente con sus principales influencias y tomar notas sobre los aspectos relevantes. Se realizaron dos sesiones de alrededor de una hora del videojuego Stardew Valley. Durante estas sesiones, se anotaron los tiempos de los ciclos de día y noche, así como el estilo de la paleta de colores.



## **4.2 Sesión de Brainstorming**

Utilizando las notas previamente tomadas, se creó un documento en el que, durante un tiempo indefinido, se anotaron todas las ideas relacionadas con la imagen mental del videojuego en su versión final, sin ningún tipo de filtro. Luego, se eliminaron las ideas que no encajaban con esta primera versión y se seleccionaron aquellas que se implementarían inicialmente.



## **4.3 Organización de Tareas**

Se creó un tablero de Trello basado en el Brainstorming, donde se incluyeron todas las tareas necesarias para completar el desarrollo del prototipo del videojuego. Cada tarea se asignó a una de cuatro categorías, cada una representada por un color diferente, para una mejor organización durante el desarrollo. Se crearon cuatro listas y se asignaron las tareas a una de ellas:

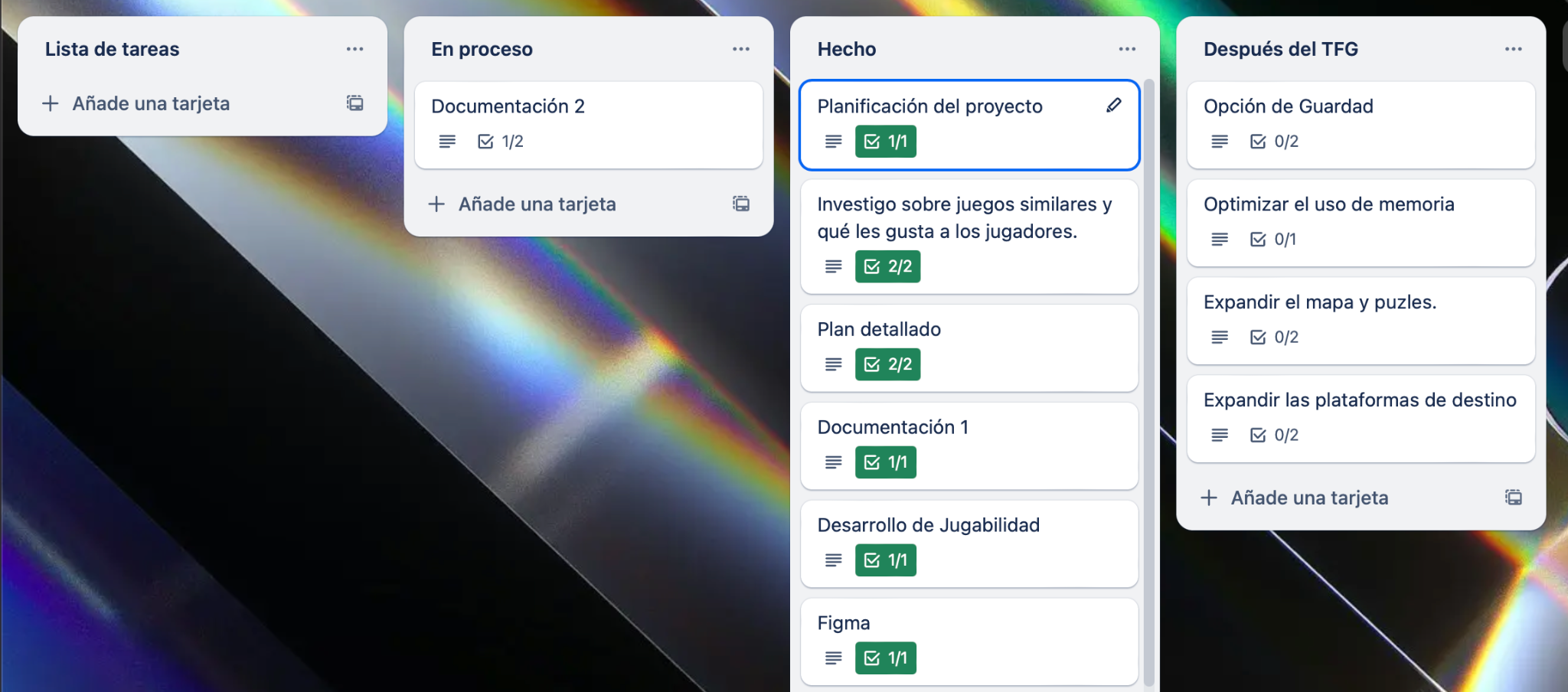
1. Backlog: Tareas pendientes que aún no habían comenzado.

2. To Do: Tareas a realizarse en el corto plazo, aproximadamente en las próximas dos semanas.

3. Doing: Tareas en las que ya se había empezado.

4. Done: Tareas finalizadas.

El uso del tablero variaba durante el desarrollo dependiendo del sistema que se estuviera implementando. Además, dado que se reevaluaron los objetivos del proyecto durante el desarrollo, algunas tarjetas fueron archivadas y reemplazadas por otras. La siguiente ilustración muestra el tablero de Trello utilizado en una etapa casi finalizada del desarrollo del proyecto.



## **4.4 Búsqueda de recursos**

Finalmente, para acelerar la implementación de los diferentes sistemas, se realizó una búsqueda de paquetes de recursos en la Asset Store de Unity. Aunque se descargaron múltiples paquetes, la mayoría no se utilizaron.

# Diseño Técnico

Una vez completada la fase de análisis, se dio paso a la fase de diseño técnico, que sería la más prolongada. En esta etapa, el objetivo principal fue implementar las mecánicas previamente planificadas, siempre teniendo en cuenta la escalabilidad, ya que el objetivo era construir un videojuego completo a partir del prototipo obtenido al finalizar este proyecto.

Antes de continuar, es importante abordar algunos aspectos básicos del funcionamiento de Unity. En este motor, todos los objetos que se encuentran en el videojuego, como la cámara o el jugador, se denominan GameObjects. Sin embargo, estos objetos no pueden realizar ninguna acción por sí mismos, por lo que requieren de Componentes que les otorguen las características necesarias para desempeñar una función específica. Los GameObjects pueden contener un número indefinido de Componentes, que pueden ser tanto los proporcionados por Unity, como un Box Collider 2D para definir los límites con los que la entidad colisiona, o un Mesh Renderer para renderizar su geometría, así como Componentes de creación propia, como los scripts en la mayoría de los casos. El único Componente que se añade automáticamente al crear un GameObject y que no se puede eliminar es Transform, el cual indica la posición que ocupa el objeto en la escena.

# Diagramas

## **6.1 Diagrama Casos de Uso**

****

**Inicio del Juego y Movimiento**

El jugador inicia el juego, lo que desencadena la carga de la primera sala. El jugador se desplaza a una nueva sala, lo que activa al motor del juego para cargar los detalles de la sala actual. Una vez cargada la sala, el motor del juego muestra estos detalles al jugador, permitiéndole visualizar el entorno. Esta interacción inicial sienta las bases para las acciones que el jugador podrá realizar dentro del juego.

**Interacción con Puzzles**

Dentro de la sala, el jugador puede encontrar puzzles. Cuando el jugador interactúa con un puzzle, el motor del juego supervisa esta interacción. Si el jugador resuelve el puzzle, el motor del juego notifica al jugador mostrando un mensaje de resolución. Esta interacción es crucial para el avance del jugador, ya que resolver puzzles puede desbloquear nuevas áreas o proporcionar objetos importantes.

**Recolección de Objetos**

Al explorar la sala y resolver puzzles, el jugador puede recoger objetos útiles. Cuando el jugador recoge un objeto, este se añade automáticamente al inventario. El motor del juego confirma que el objeto ha sido añadido al inventario mediante un mensaje. Esta mecánica permite al jugador acumular herramientas y elementos esenciales para avanzar en el juego y resolver futuros desafíos.

**Transición entre Niveles**

A medida que el jugador avanza en el juego, puede moverse a diferentes niveles o áreas. Cuando el jugador se desplaza a un nuevo nivel, el motor del juego carga los detalles de ese nivel y luego los muestra al jugador. Este proceso permite al jugador explorar nuevas áreas del castillo, cada una con sus propios desafíos y elementos únicos.

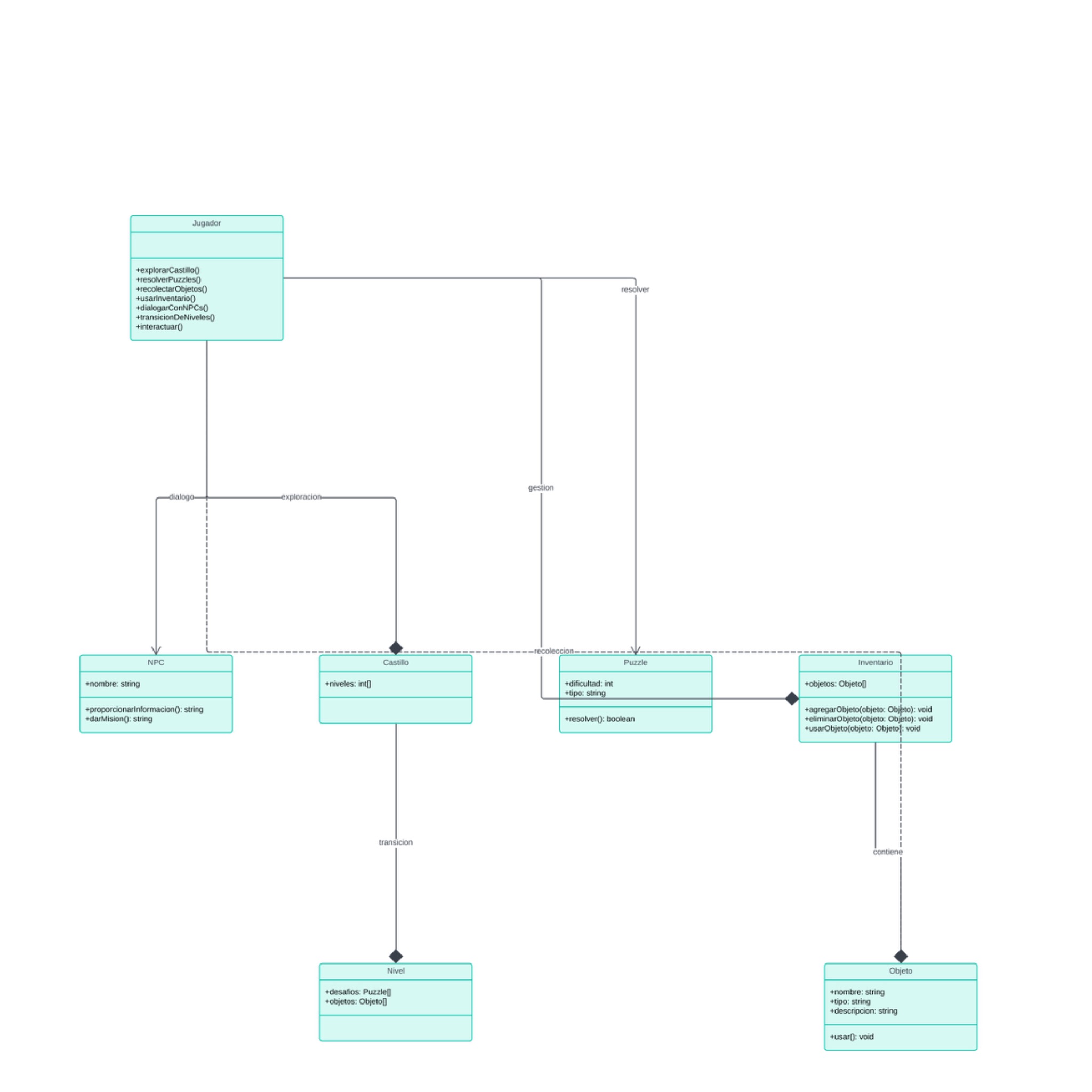
**Interacción con el Entorno**

En los nuevos niveles, el jugador puede interactuar con varios elementos del entorno. Estas interacciones pueden desencadenar eventos específicos. Por ejemplo, activar un mecanismo puede abrir una puerta secreta o iniciar una secuencia de eventos. El motor del juego procesa estas interacciones y muestra el resultado al jugador, manteniendo una experiencia de juego fluida y dinámica.

**Conclusión**

Este diagrama de secuencia ilustra las principales interacciones entre el jugador y los componentes del juego "Lost Castle", proporcionando una visión clara de cómo se desarrollan las acciones y reacciones a lo largo del juego. A través de la exploración, resolución de puzzles, recolección de objetos, el jugador avanza en la historia y descubre los secretos del castillo.

## 6.2 Diagrama de Clases

****

### Jugador

La clase Jugador es el núcleo del juego, representando al usuario que interactúa con el entorno del castillo. El jugador tiene varios métodos clave:

* explorarCastillo(): Permite al jugador moverse y descubrir diferentes áreas del castillo.
* resolverPuzzle(): Maneja la lógica para resolver los puzzles encontrados en el castillo.
* recolectarObjeto(): Permite al jugador recoger objetos útiles durante la exploración.
* usarInventario(): Gestiona el uso de objetos recolectados.
* transicionarNivel(): Maneja la transición entre diferentes niveles del castillo.

### Castillo

La clase Castillo actúa como el entorno principal del juego. Contiene una lista de niveles (niveles: int[]), representando las diferentes áreas que el jugador puede explorar. El castillo también gestiona la exploración del jugador y su interacción con otros elementos del juego, como puzzles.

### Nivel

La clase Nivel representa una subdivisión del castillo. Cada nivel contiene una lista de puzzles (desafíos: Puzzle[]) y una lista de objetos (objetos: Objeto[]) que el jugador puede encontrar y recolectar. Esta clase gestiona la transición entre niveles y la disposición de los elementos dentro de cada uno.

### Puzzle

La clase Puzzle representa los desafíos que el jugador debe resolver para avanzar en el juego. Cada puzzle tiene una dificultad (dificultad: int) y una descripción (reto: string). Además, tiene un atributo booleano resuelto que indica si el puzzle ha sido completado. Esta clase se integra con el jugador a través del método resolver.

### Inventario

La clase Inventario almacena los objetos recolectados por el jugador. Contiene una lista de objetos (objetos: Objeto[]) y varios métodos:

* agregarObjeto(): Añade un nuevo objeto al inventario.

### Objeto

La clase Objeto define los atributos de los elementos que el jugador puede recoger. Cada objeto tiene un nombre, un tipo, una descripción y un método usar() que define su uso en el juego. Estos objetos pueden ser herramientas, armas, llaves u otros ítems necesarios para resolver puzzles o avanzar en la historia.

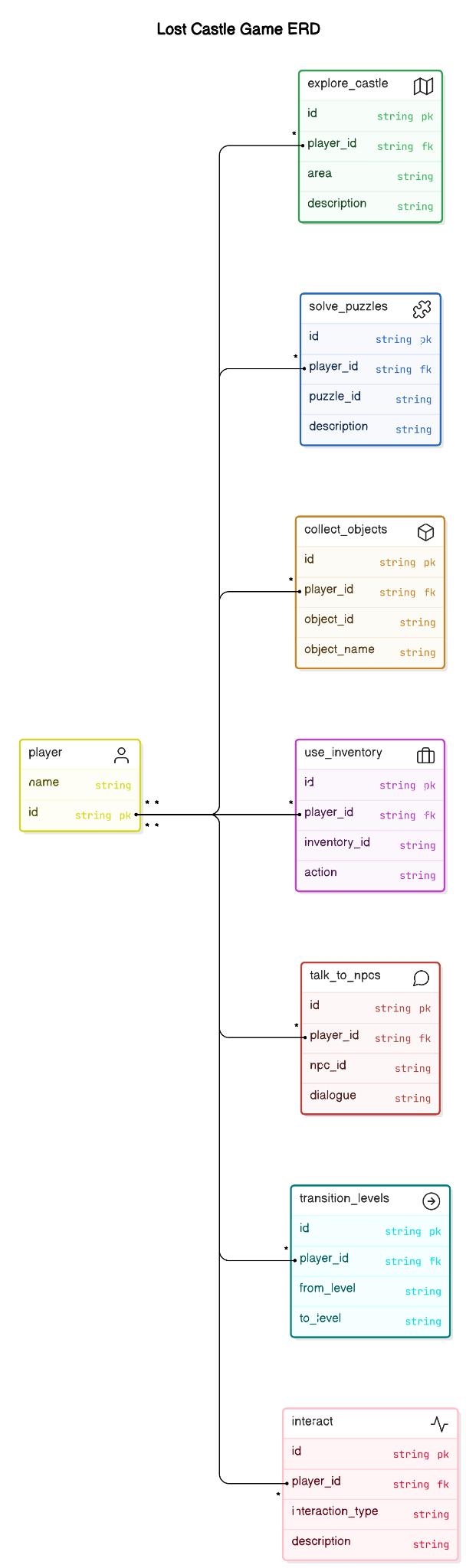
### Relaciones entre Clases

Las relaciones entre las clases están definidas por las interacciones del jugador con el entorno:

* El Jugador explora el Castillo, interactúa con Puzzles, recolecta Objetos y gestiona su Inventario.
* El Castillo contiene múltiples Niveles, cada uno con sus propios Puzzles y Objetos.

Este diagrama de clases ofrece una estructura clara de cómo se interrelacionan las diferentes entidades dentro del juego "Lost Castle", destacando las principales funcionalidades y flujos de interacción del jugador.

## **6.3 Diagrama Entidad-Relación**



**Entidades:**

* **Jugador:** Representa a la persona que está jugando al juego.
  + Atributos:
    - id: Identificador único del jugador.
    - nombre: Nombre del jugador.
    - inventario: Lista de objetos que el jugador tiene en su inventario.
* **Castillo:** Representa el castillo que el jugador debe explorar.
  + Atributos:
    - id: Identificador único del castillo.
    - descripción: Descripción del castillo.
* **Objeto:** Representa los objetos que el jugador puede encontrar en el castillo.
  + Atributos:
    - id: Identificador único del objeto.
    - nombre: Nombre del objeto.
    - descripción: Descripción del objeto.
* **Puzle:** Representa los puzles que el jugador debe resolver para avanzar en el juego.
  + Atributos:
    - id: Identificador único del puzle.
    - descripción: Descripción del puzle.
* **Nivel:** Representa los diferentes niveles del castillo.
  + Atributos:
    - id: Identificador único del nivel.
    - jugador\_id: Identificador del jugador que está en el nivel.
    - nivel\_anterior: Identificador del nivel anterior.
    - nivel\_siguiente: Identificador del siguiente nivel.
* **Interacción:** Representa las interacciones que el jugador tiene con los objetos, puzles y niveles.
  + Atributos:
    - id: Identificador único de la interacción.
    - jugador\_id: Identificador del jugador que realizó la interacción.
    - tipo\_interacción: Tipo de interacción (por ejemplo, "usar objeto", "resolver puzle", "hablar con NPC", "cambiar de nivel").
    - descripción: Descripción de la interacción.

**Relaciones:**

* **Jugador-Castillo:** Un jugador puede explorar un castillo.
* **Jugador-Objeto:** Un jugador puede tener un objeto en su inventario.
* **Jugador-Puzle:** Un jugador puede resolver un puzle.
* **Jugador-Nivel:** Un jugador puede estar en un nivel.
* **Objeto-Puzle:** Un objeto puede ser necesario para resolver un puzle.
* **Nivel-Nivel:** Un nivel puede tener un nivel anterior y un nivel siguiente.

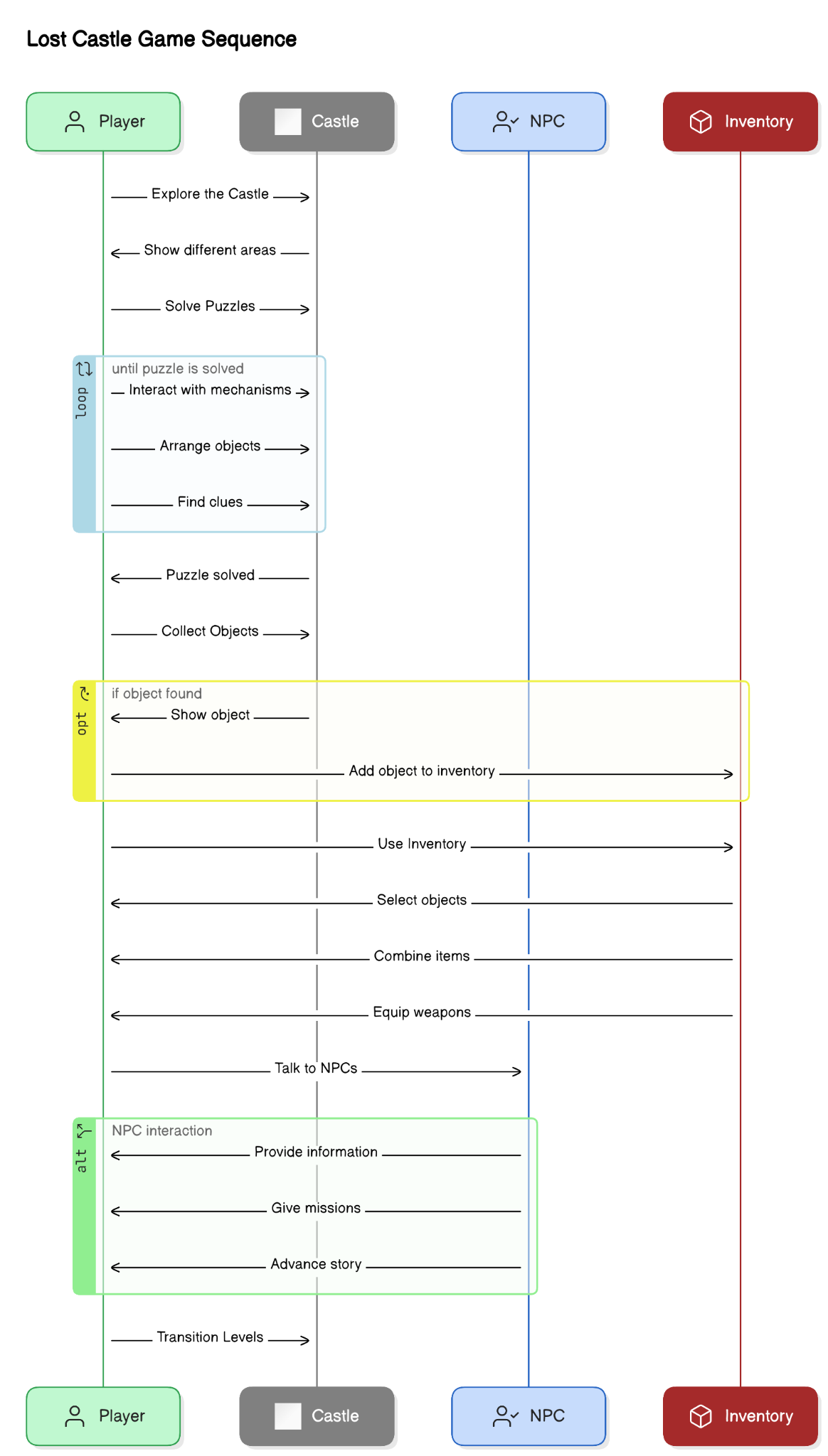
**Cardinalidad:**

* **Jugador-Castillo:** Uno a uno. Un jugador solo puede explorar un castillo a la vez.
* **Jugador-Objeto:** Muchos a muchos. Un jugador puede tener muchos objetos en su inventario, y un objeto puede estar en el inventario de muchos jugadores.
* **Jugador-Puzle:** Muchos a muchos. Un jugador puede resolver muchos puzles, y un puzle puede ser resuelto por muchos jugadores.
* **Jugador-Nivel:** Uno a uno. Un jugador solo puede estar en un nivel a la vez.
* **Objeto-Puzle:** Uno a muchos. Un objeto solo puede ser necesario para resolver un puzle, pero un puzle puede necesitar muchos objetos.
* **Nivel-Nivel:** Uno a uno. Un nivel solo puede tener un nivel anterior y un nivel siguiente.

**Restricciones:**

* Un jugador no puede estar en dos niveles diferentes al mismo tiempo.
* Un objeto no puede estar en dos inventarios diferentes al mismo tiempo.
* Un puzle no puede ser resuelto por dos jugadores al mismo tiempo.

## 6.4 Diagrama de Componentes

****

El diagrama de componentes se trata de un juego de exploración y aventura en el que el jugador debe explorar un castillo y resolver puzles para avanzar en la historia.

**Componentes principales:**

* **Jugador:** Es el componente central del juego y representa al personaje que el jugador controla. El jugador puede moverse por el castillo, interactuar con los objetos y puzles, y hablar con los NPC.
* **Castillo:** Es el escenario principal del juego. El castillo está dividido en diferentes áreas que el jugador debe explorar. Cada área tiene su propia disposición y contiene diferentes objetos y puzles.
* **Objetos:** Son elementos que el jugador puede encontrar en el castillo. Los objetos pueden ser utilizados para resolver puzles, abrir puertas o desbloquear nuevas áreas del castillo.
* **Puzles:** Son desafíos que el jugador debe resolver para avanzar en el juego. Los puzles pueden ser de diferentes tipos, como encontrar objetos, resolver acertijos o activar mecanismos.
* **Inventario:** Es un espacio donde el jugador almacena los objetos que encuentra. El inventario tiene una capacidad limitada, por lo que el jugador debe seleccionar cuidadosamente qué objetos quiere llevar consigo.

**Interacciones entre componentes:**

* **Jugador-Castillo:** El jugador puede moverse por las diferentes áreas del castillo, interactuar con los objetos y puzles, y hablar con los NPC.
* **Jugador-NPC:** El jugador puede hablar con los NPC(Libros) para obtener información, misiones o ayuda para resolver puzles.
* **Jugador-Objetos:** El jugador puede recoger objetos, usarlos para resolver puzles o abrir puertas.
* **Objetos-Puzles:** Algunos objetos pueden ser necesarios para resolver puzles.
* **Puzles-Inventario:** Algunos puzles pueden requerir que el jugador use objetos de su inventario para resolverlos.

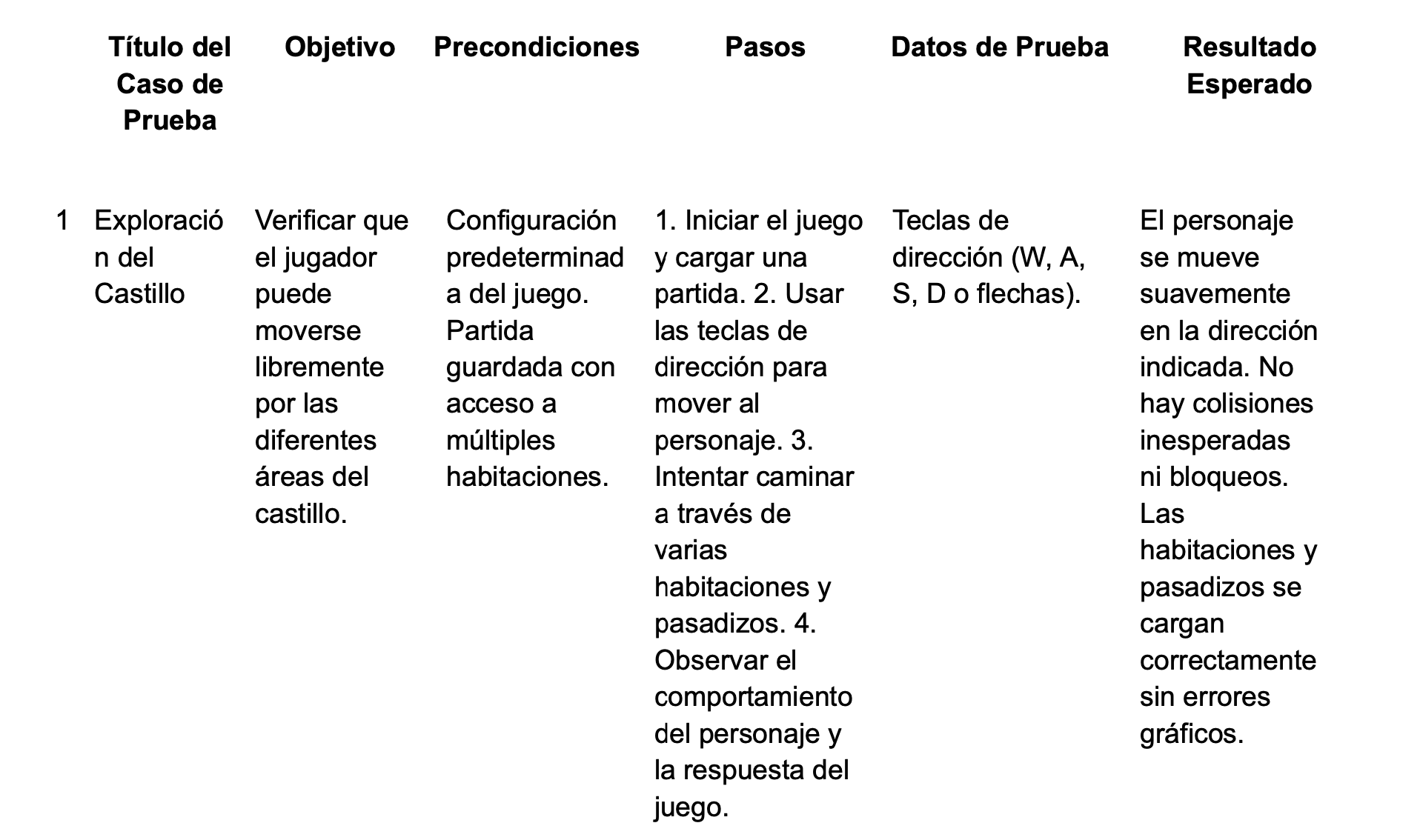
**Características adicionales:**

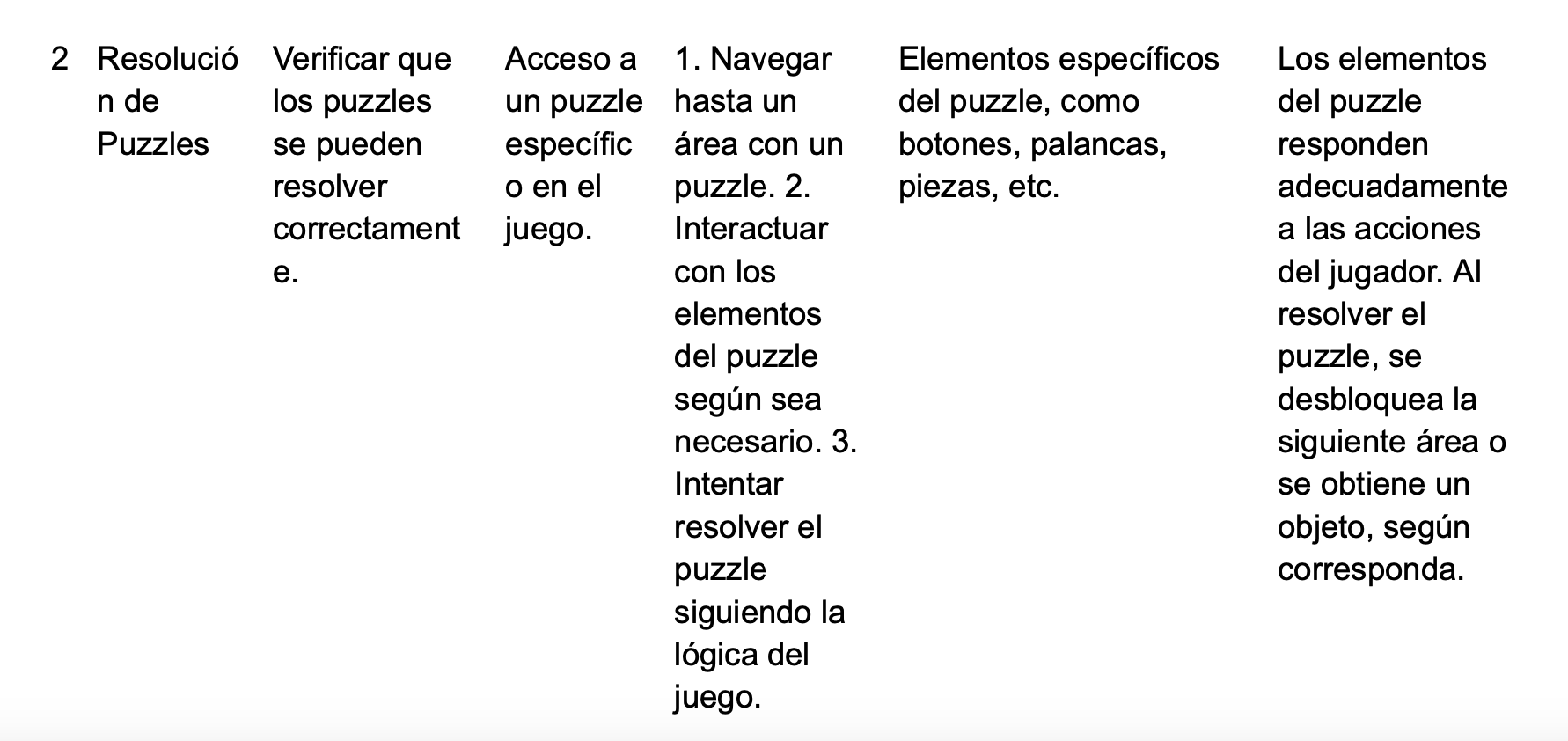
* **Música y efectos de sonido:** El juego tiene una banda sonora y efectos de sonido que ayudan a crear una atmósfera inmersiva.

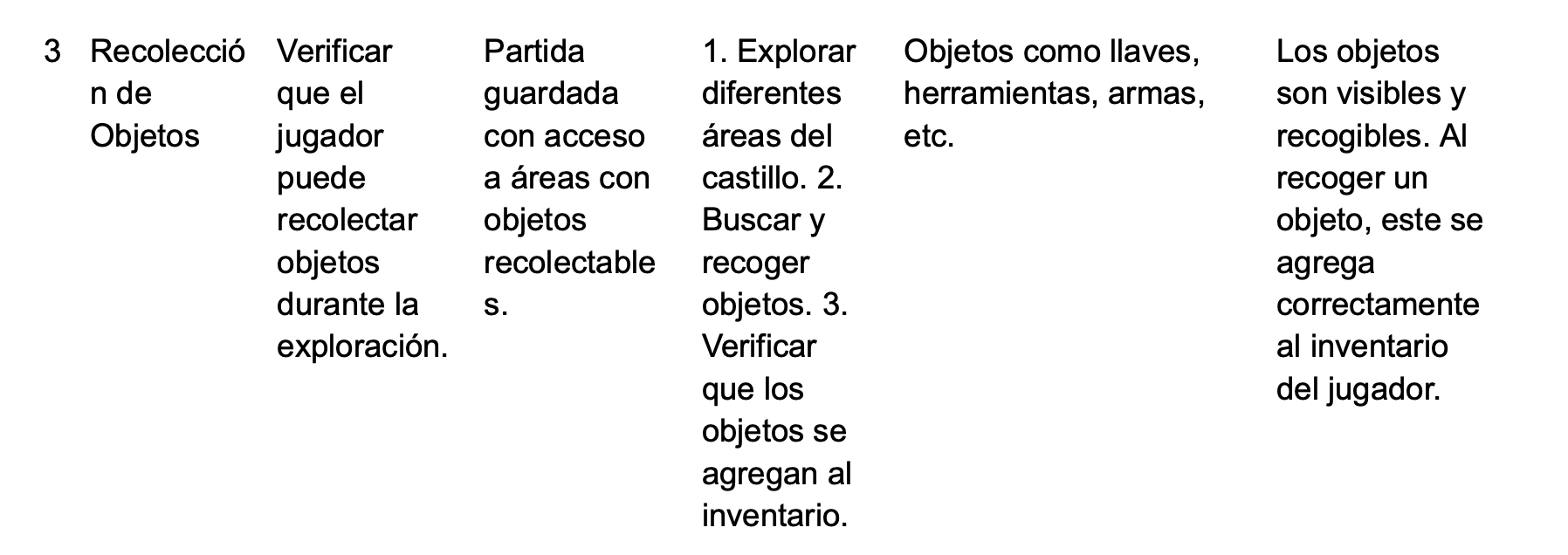
**Beneficios de la arquitectura de componentes:**

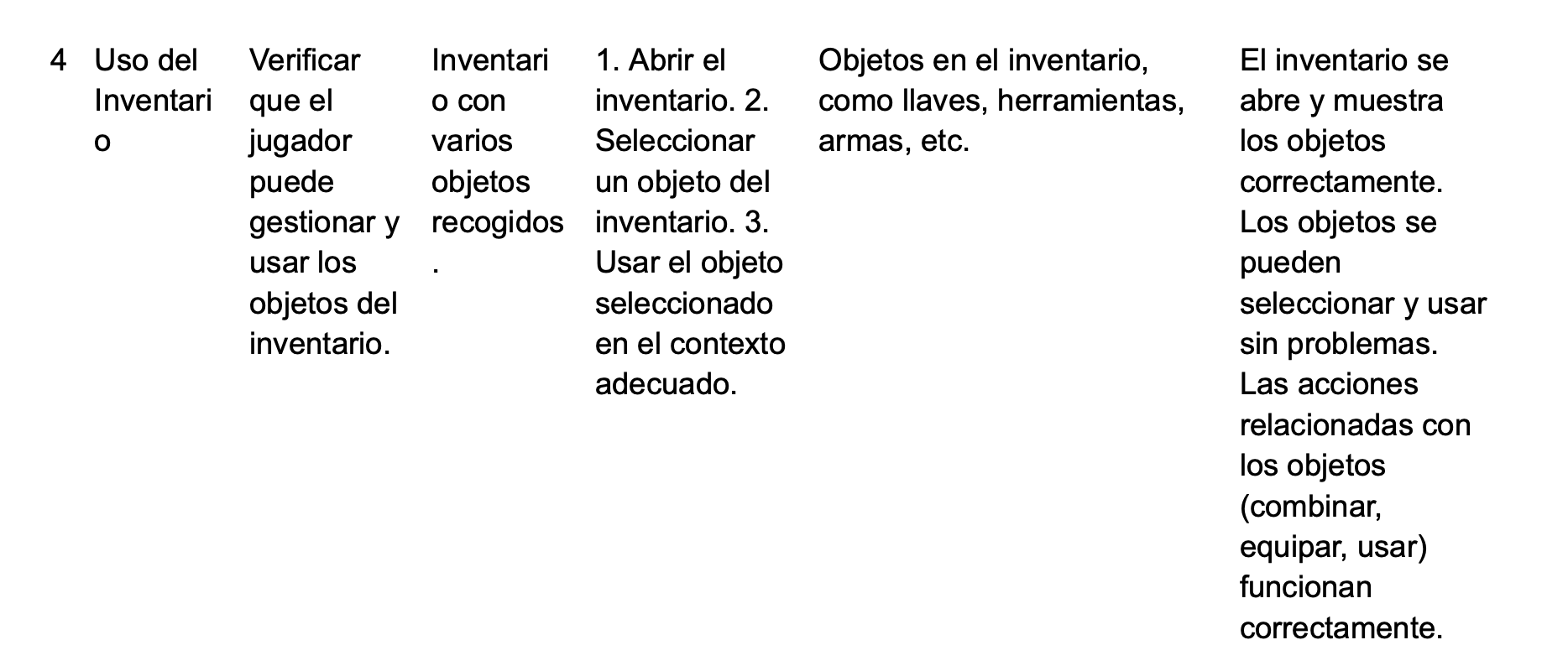
* **Modularidad:** La arquitectura de componentes permite que el juego sea fácilmente ampliable y modificable. Se pueden añadir nuevos componentes o modificar los existentes sin afectar al resto del juego.
* **Reusabilidad:** Los componentes se pueden reutilizar en otros juegos o proyectos.
* **Mantenibilidad:** La arquitectura de componentes facilita el mantenimiento del juego, ya que los componentes se pueden depurar y actualizar de forma individual.

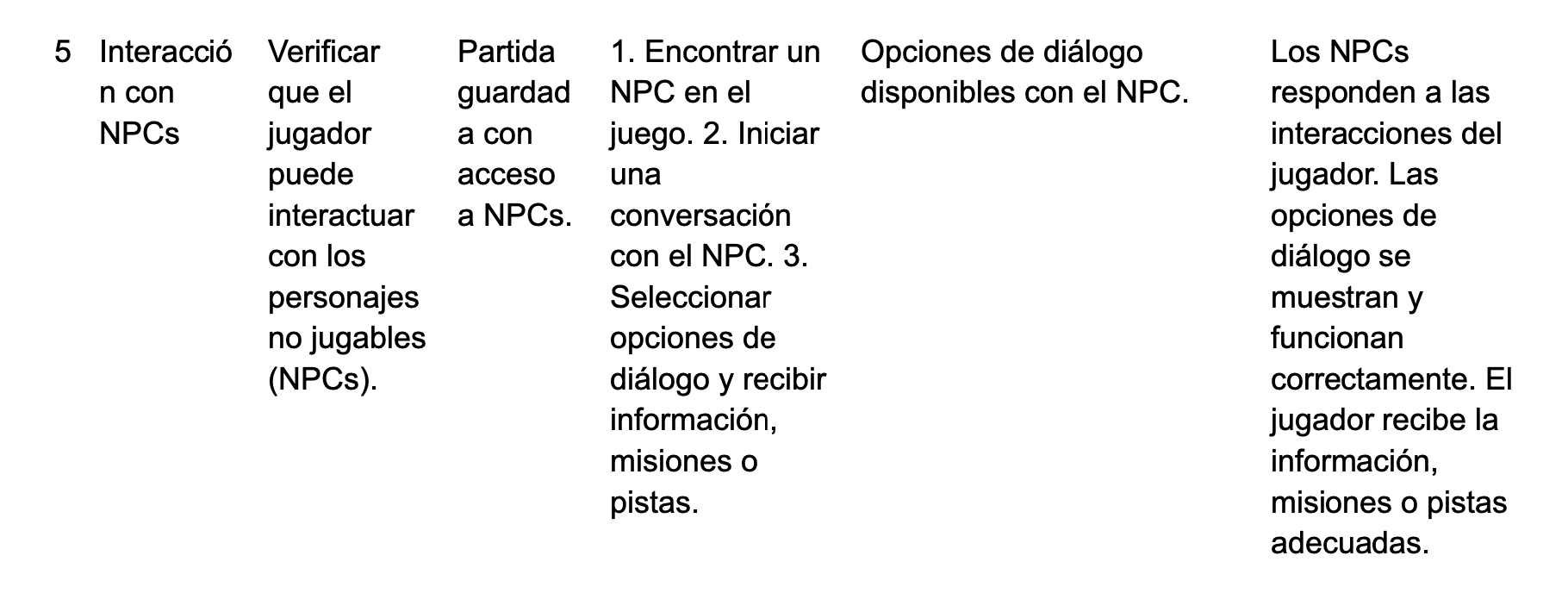
## 6.5 Casos de Pruebas

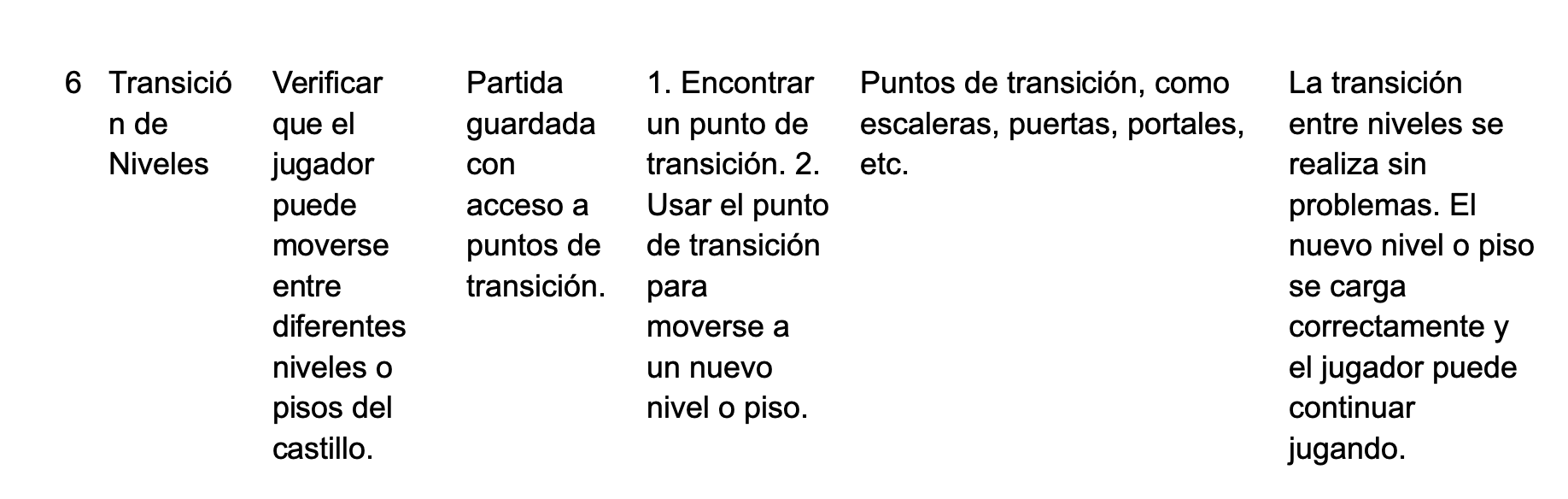


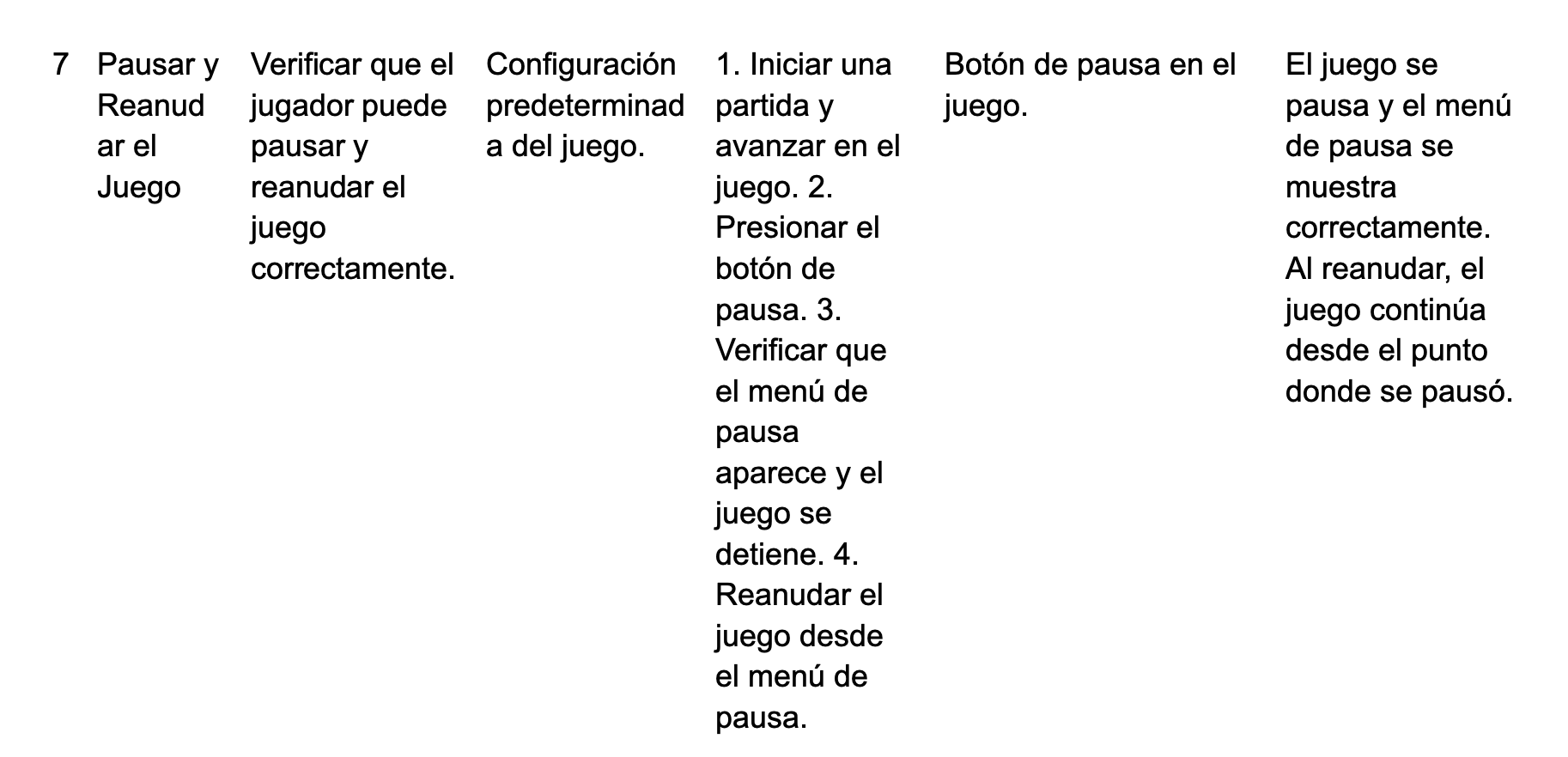


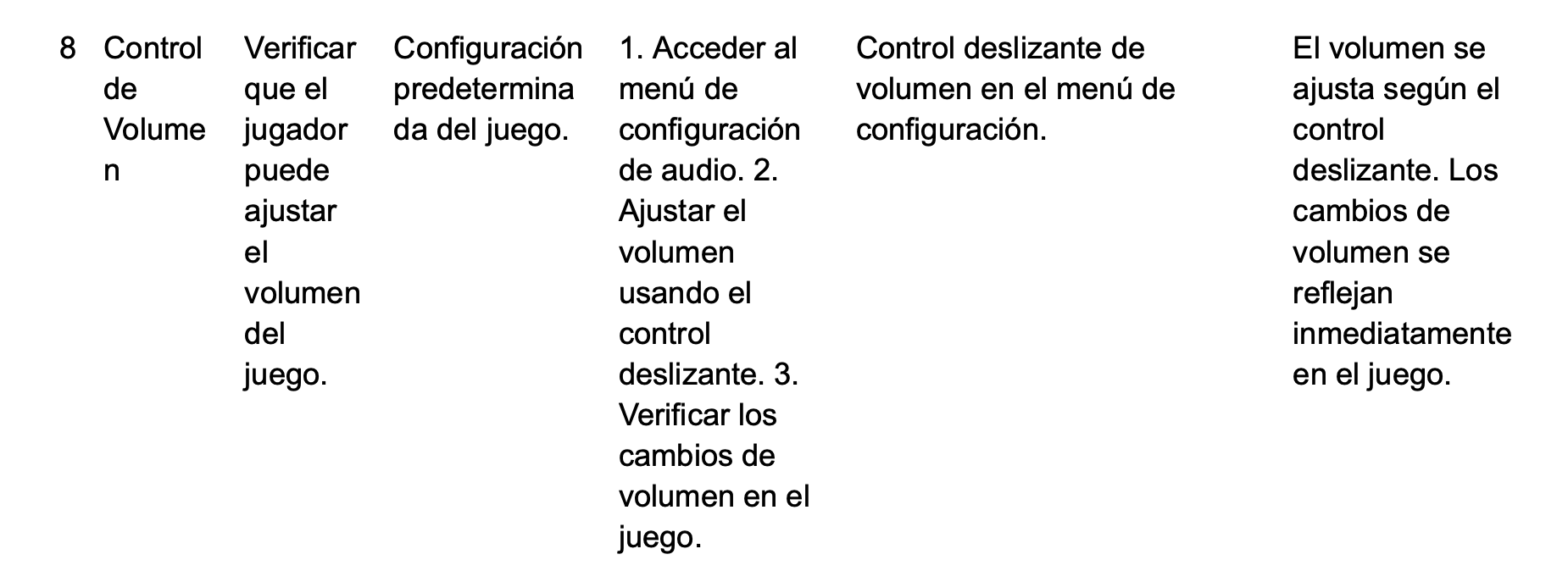














## Análisis Dafo

****

**Fortalezas:**

* **Originalidad del concepto:** "Lost Castle" combina elementos de narrativa y puzzles en un entorno intrigante, lo que puede atraer a un amplio espectro de jugadores.
* **Estilo artístico:** El diseño pixel art es visualmente atractivo y puede apelar a los aficionados de juegos retro y modernos por igual.
* **Experiencia inmersiva:** La combinación de exploración y resolución de puzzles puede mantener a los jugadores interesados y comprometidos.
* **Desarrollo continuado:** El hecho de que el proyecto continuará después de la presentación del TFG permite iteraciones y mejoras basadas en el feedback.

**Debilidades:**

* **Complejidad técnica:** El desarrollo de un juego 2D con elementos de narrativa y puzzles puede ser técnicamente desafiante y requerir un alto nivel de habilidad en programación y diseño.
* **Falta de experiencia:** Este es tu primer gran proyecto de juego como desarrollador, la curva de aprendizaje puede ser empinada y conducir a errores o desafíos inesperados.

**Oportunidades:**

* **Mercado de juegos Indie:** Existe una creciente demanda de juegos Indie únicos y creativos. "Lost Castle" puede captar la atención de este mercado.
* **Feedback de jugadores:** La posibilidad de continuar el desarrollo después del TFG permite la incorporación de feedback de los jugadores para mejorar el juego.
* **Plataformas de distribución:** Plataformas como Steam, itch.io y otras permiten una distribución relativamente fácil y acceso a un público global.

**Amenazas:**

* **Expectativas de los jugadores:** Los jugadores tienen expectativas altas en cuanto a la calidad y la jugabilidad, lo que puede ser dificil de satisfacer sin los recursos adecuados.
* **Errores técnicos:** Bugs y errores técnicos pueden afectar la experiencia del jugador y dañar la reputación del juego.
* **Cambios en el mercado:** Las tendencias en el mercado de videojuegos cambian rápidamente, lo que puede hacer que un concepto popular hoy no sea tan atractivo en el futuro.

**Recomendaciones:**

* **Enfócate en pulir la experiencia central del juego:** Es importante que el juego sea divertido y atractivo de jugar.Céntrate en crear una experiencia de juego sólida antes de agregar características adicionales.
* **Obtén feedback de los jugadores:** Es importante obtener feedback de los jugadores tan pronto como sea posible.Esto te ayudará a identificar áreas del juego que necesitan ser mejoradas.
* **Utiliza herramientas y recursos disponibles:** Hay muchas herramientas y recursos disponibles para desarrolladores de juegos Indie. Aprovecha estos recursos para ayudarte a desarrollar tu juego.
* **Sé paciente y perseverante:** Desarrollar un juego puede llevar mucho tiempo y esfuerzo. Sé paciente y perseverante, y eventualmente lograrás tus objetivos.

# Resultados y Conclusiones

Inicialmente, se hizo una propuesta demasiado optimista sobre lo que se podría lograr en este proyecto, lo cual obligó a reevaluar los objetivos planteados y ajustarlos a algo más realista. Las mayores dificultades en la implementación se encontraron en el sistema de puzzles, requiriendo más tiempo del inicialmente previsto debido a la cantidad de código necesario para su correcto funcionamiento.

Sin embargo, considero que el resultado obtenido, especialmente en cuanto a los sistemas implementados, ha sido muy positivo. Este prototipo constituye una buena base sobre la que se podrá desarrollar un videojuego completo en el futuro, si así se desea. La estructura de clases está diseñada para añadir nuevas funcionalidades de forma rápida y la mayoría de las decisiones de diseño se han tomado con la expansión futura en mente.

# Análisis de Impacto

A nivel personal, este proyecto ha tenido un gran impacto. No solo me ha proporcionado una excelente experiencia de aprendizaje, sino que también me será útil en el futuro al buscar un trabajo en la industria del videojuego, sumando a la experiencia que ya tengo en la creación de estos. Además, creo que me ha aportado un gran desarrollo personal, permitiéndome mejorar múltiples competencias, como mi habilidad para estimar el tiempo necesario para la implementación de ciertos sistemas en este tipo de proyectos y mi capacidad para fijar metas alcanzables.

# Despedida

Quisiera concluir este documento expresando mi más sincero agradecimiento a todos aquellos que han contribuido de una u otra manera en la realización de este proyecto. Este Trabajo de Final de Grado no solo ha sido un desafío académico, sino también una oportunidad invaluable para mi desarrollo profesional y personal.

Agradezco profundamente a mis profesores y tutores por su orientación y apoyo constante, así como a mis compañeros por su colaboración y camaradería durante este viaje. Igualmente, deseo destacar el respaldo de mi familia y amigos, quienes me han motivado a seguir adelante en los momentos más difíciles.

Este proyecto me ha permitido aplicar los conocimientos adquiridos a lo largo de mi formación y explorar nuevas áreas del desarrollo de videojuegos. Espero que el resultado de mi esfuerzo no solo cumpla con las expectativas académicas, sino que también sirva como base para futuros desarrollos y proyectos en mi carrera profesional.

Con esta despedida, cierro un capítulo importante de mi vida académica, pero con la certeza de que es solo el comienzo de una carrera llena de desafíos y oportunidades en el apasionante mundo de los videojuegos.

Gracias a todos por ser parte de este viaje.

Atentamente,

José Luis González Álamo