Jaipur 3D

Tabla de contenidos

[1 Declaración de alcance 4](#_Toc497758031)

[1.1 Objetivo: 4](#_Toc497758032)

[1.2 Requerimientos: 4](#_Toc497758033)

[1.3 Criterio de aceptación: 4](#_Toc497758034)

[1.4 Entregables: 4](#_Toc497758035)

[1.5 WBS inicial: 4](#_Toc497758036)

[1.6 Requerimientos de aprobación: 4](#_Toc497758037)

[1.7 Integrantes del equipo 5](#_Toc497758038)

[2 Work Breakdown Structure 6](#_Toc497758039)

[3 Roles 7](#_Toc497758040)

[4 Calendarización 8](#_Toc497758041)

[5 Análisis de Riesgos 9](#_Toc497758042)

[5.1 General 9](#_Toc497758043)

[5.2 Diseño de manejador del juego 9](#_Toc497758044)

[5.3 Diseño de otros componentes 9](#_Toc497758045)

[5.4 Diseño de interfaz de usuario 10](#_Toc497758046)

[5.5 Diseño de servidor 10](#_Toc497758047)

[5.6 Diseño de cliente 10](#_Toc497758048)

[5.7 Implementación del menú de juego 11](#_Toc497758049)

[5.8 Implementación del menú principal 11](#_Toc497758050)

[5.9 Programación de otros componentes 12](#_Toc497758051)

[5.10 Programación del cliente-servidor 12](#_Toc497758052)

[5.11 Programación del manejador del juego 13](#_Toc497758053)

[5.12 Integración 13](#_Toc497758054)

[5.13 Recopilación de referencias 13](#_Toc497758055)

[5.14 Diseño del manual de usuario 14](#_Toc497758056)

[5.15 Impresión del manual de usuario 14](#_Toc497758057)

[5.16 Modelado de cartas y fichas 14](#_Toc497758058)

[5.17 Texturizado de cartas y fichas 14](#_Toc497758059)

[5.18 Proceso de Testing 15](#_Toc497758060)

[5.19 Compra o adquisición de la caja 15](#_Toc497758061)

[5.20 Diseño de la portada de la caja 15](#_Toc497758062)

[5.21 Impresión de la portada de la caja 15](#_Toc497758063)

[5.22 Elaboración de instalador binario 16](#_Toc497758064)

[5.23 Preparación de entregables 16](#_Toc497758065)

[5.24 Resolución de Issues 16](#_Toc497758066)

[5.25 Entrega 16](#_Toc497758067)

[6 Programación Basada en Componentes 17](#_Toc497758068)

[6.1 ¿Qué es un componente? 17](#_Toc497758069)

[6.2 ¿Cómo y por qué Unity favorece el uso de componentes? 17](#_Toc497758070)

[6.3 Pensando en componentes 17](#_Toc497758071)

[6.4 Conclusión 18](#_Toc497758072)

[7 Proceso de Testing 19](#_Toc497758073)

[7.1 Objetivo 19](#_Toc497758074)

[7.2 Lista de pasos 19](#_Toc497758075)

[7.3 Equipo de Testing 19](#_Toc497758076)

[7.4 Aprobación 19](#_Toc497758077)

# Declaración de alcance

## Objetivo:

Producir el videojuego “Jaipur 3D” en 5 semanas.

## Requerimientos:

El juego debe tener gráficos con perspectiva tridimensional y multijugador en red (máximo 2 jugadores). Las reglas del juego deben de ser las mismas del juego de mesa “Jaipur” de Sébastien Pauchon.

Debe poder instalarse y jugarse en una computadora que tenga las siguientes especificaciones (para poder jugarse en la configuración de gráficos “baja”):

* Tarjeta de video dedicada o integrada con al menos 512MB de memoria gráfica.
* DirectX 11 o superior.
* Sistema operativo: Windows 10 de 64 bits.
* CPU: Que soporte set de instrucciones SSE2.
* RAM: Al menos 2GB de memoria.
* Monitor que soporte resolución de 1366 x 768 o 1920 x 1080.
* Teclado y mouse.

## Criterio de aceptación:

El juego debe aprobar el proceso de Testing. El desarrollo del proyecto debe de mantener su calendarización para poder ser terminado en el plazo de 5 semanas.

## Entregables:

Instalador del juego binario, código fuente del videojuego, manual de usuario, caja, portada de la caja y disco.

* Instalador del juego
* Código fuente
* Manual de usuario
* Caja
* Portada de la caja
* Disco con el instalador
* Calendarización
* Documentos de diseño

## WBS inicial:

El desarrollo del proyecto estará dividido en 4 fases: programación, arte, pruebas y manejo de entregables.

## Requerimientos de aprobación:

“Jaipur 3D” será entregado después de ser aprobado por los mismos desarrolladores y por el asesor del proyecto, el Maestro en Ciencias Juan Francisco Algara Norzagaray.

## Integrantes del equipo

* Christian Ricardo Lugo Arellano
* José David Rochín Cerecer

# Work Breakdown Structure

**Nota**

“Otros componentes” se refiere a todos los componentes del juego que no son Cliente, Servidor y Manejador del Juego.

# Roles

|  |  |
| --- | --- |
| **Persona** | **Roles** |
| Christian Ricardo Lugo Arellano | Programador de Netcode, Testing. |
| José David Rochín Cerecer | Programador del Juego, Testing. |
| M.C. Juan Francisco Algara Norzagaray | Asesor del Proyecto |

# Calendarización



# Análisis de Riesgos

## General

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El videojuego no es terminado. | El equipo de trabajo no concluye el desarrollo del videojuego; queda en un estado inutilizable. | 10% | 40 puntos | Aceptar. |
| El videojuego es parcialmente terminado. | El equipo de trabajo logra tener un videojuego usable pero no cuenta con todos los requerimientos y características definidas en la planeación. | 10% | 15 puntos | Mitigar. Durante cada tarea, se analizará si el tiempo determinado para terminar el proyecto es apropiado, y de no ser así, se contratará a alguien externo para ayudar con el desarrollo. |
| Faltas ortográficas | Los textos de la interfaz gráfica del videojuego tienen faltas ortográficas. | 5% | 3 puntos | Mitigar. Se verificará la ortografía de los textos del videojuego en la etapa de Testing. |

## Diseño de manejador del juego

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El diseño no se termina, por lo tanto, no se implementa. | 20% | 30 puntos | Mitigar. Durante la tarea de diseño, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |
| El diseño es incorrecto. | El diseño del Manejador del Juego es calificado por el asesor como “incorrecto”. | 60% | 1 punto | Mitigar. Se revisará el diseño con el asesor siempre que sea posible. |

## Diseño de otros componentes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El diseño no se termina, por lo tanto, no se implementa. | 20% | 30 puntos | Mitigar. Durante la tarea de diseño, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |
| El diseño es incorrecto. | El diseño de la estructura de componentes del Juego es calificado por el asesor como “incorrecto”. | 60% | 1 punto | Mitigar. Se revisará el diseño con el asesor siempre que sea posible. |

## Diseño de interfaz de usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El diseño no se termina, por lo tanto, no se implementa. | 30% | 30 puntos | Mitigar. Durante la tarea de diseño, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |
| El diseño es incorrecto. | El diseño de la interfaz de usuario es calificado por el asesor como “incorrecto”. | 60% | 1 punto | Mitigar. Se revisará el diseño con el asesor siempre que sea posible. |

## Diseño de servidor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El diseño no se termina, por lo tanto, no se implementa. | 30% | 30 puntos | Mitigar. Durante la tarea de diseño, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |
| El diseño es incorrecto. | Los diseños del cliente y el servidor son calificados por el asesor como “incorrectos”. | 60% | 1 punto | Mitigar. Se revisará el diseño con el asesor siempre que sea posible. |

## Diseño de cliente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El diseño no se termina, por lo tanto, no se implementa. | 30% | 30 puntos | Mitigar. Durante la tarea de diseño, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |
| El diseño es incorrecto. | Los diseños del cliente y el servidor son calificados por el asesor como “incorrectos”. | 60% | 1 punto | Mitigar. Se revisará el diseño con el asesor siempre que sea posible. |

## Implementación del menú de juego

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Es implementado de una manera muy diferente a su diseño. | La implementación no sigue las partes básicas e importantes del diseño. | 20% | 1 punto | Aceptar. Se tiene poca experiencia en diseño de interfaces, así que es muy probable que la implementación no siga algunas partes básicas del diseño. |
| No se termina de implementar. | No se termina de implementar, por lo tanto, el juego no funciona parcial o totalmente. | 20% | 30 puntos | Mitigar. Durante la tarea de implementación, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |

## 

## Implementación del menú principal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Es implementado de una manera muy diferente a su diseño. | La implementación no sigue las partes básicas e importantes del diseño. | 20% | 1 punto | Aceptar. Se tiene poca experiencia en diseño de interfaces, así que es muy probable que la implementación no siga algunas partes básicas del diseño. |
| No se termina de implementar. | No se termina de implementar, por lo tanto, el juego no funciona parcial o totalmente. | 20% | 30 puntos | Mitigar. Durante la tarea de implementación, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |

## Programación de otros componentes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Es programado de una manera muy diferente a su diseño. | La programación no sigue las partes básicas e importantes del diseño. | 20% | 1 punto | Aceptar. Se tiene poca experiencia en diseño de software, así que es muy probable que la programación no siga algunas partes básicas del diseño. |
| No se termina de programar. | No se termina de programar, por lo tanto, el juego no funciona parcial o totalmente. | 20% | 30 puntos | Mitigar. Durante la tarea de programación, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |

## 

## Programación del cliente y programación del servidor

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Es programado de una manera muy diferente a su diseño. | La programación no sigue las partes básicas e importantes del diseño. | 20% | 1 punto | Aceptar. Se tiene poca experiencia en diseño de software, así que es muy probable que la programación no siga algunas partes básicas del diseño. |
| No se termina de programar. | No se termina de programar, por lo tanto, el juego no funciona parcial o totalmente. | 20% | 30 puntos | Mitigar. Durante la tarea de programación, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |

## Programación del manejador del juego

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Es programado de una manera muy diferente a su diseño. | La programación no sigue las partes básicas e importantes del diseño. | 20% | 1 punto | Aceptar. Se tiene poca experiencia en diseño de software, así que es muy probable que la programación no siga algunas partes básicas del diseño. |
| No se termina de programar. | No se termina de programar, por lo tanto, el juego no funciona parcial o totalmente. | 20% | 30 puntos | Mitigar. Durante la tarea de programación, el administrador del proyecto verificará que su avance sea suficiente para terminarla. |

## 

## Integración

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se termina la integración. | No se termina la integración de los componentes, por lo tanto, el juego funciona parcial o totalmente. | 10% | 30 punto | Mitigar. En cuanto se detecte que el tiempo de integración no es suficiente, se recalendarizará la tarea. |

## Recopilación de referencias

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se encuentran todas las referencias necesarias. | La búsqueda de referencias no concluye debido a que las fuentes consultadas no contienen la información necesaria para hacerlo. | 20% | 1 punto | Mitigar. Si se detecta que no hay mucha información, se conseguirá una copia física del juego de mesa. |
| Se pierden los archivos. | Los archivos resultantes de la recopilación son extraviados o dañados. | 5% | 5 puntos | Mitigar. Los archivos se guardarán en el repositorio remoto siempre que se tenga la oportunidad. |

## Diseño del manual de usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se termina. | El diseño del manual de usuario no es terminado. | 10% | 5 punto | Aceptar. |

## Impresión del manual de usuario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El manual no se imprime. | Debido a dificultades de cualquier tipo, la impresión del manual no se lleva a cabo. | 10% | 5 punto | Mitigar. Se asignarán dos días completos para la impresión del manual. |
| Las medidas de la impresión son incorrectas. | Ya sea por error del responsable o por problema técnico, la impresión resulta tener el tamaño incorrecto. | 20% | 1 punto | Mitigar. Se verificará la configuración de las medidas de la impresión en una impresora casera antes de ir a imprimirlo a color a un negocio. |

## Modelado de cartas y fichas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se terminan de modelar los objetos. | La persona encargada de modelar los objetos no termina de hacerlo; por lo tanto, el videojuego se queda sin modelos. | 5% | 20 puntos | Mitigar. En caso de que el tiempo no alcance para modelar, se usaran modelos primitivos que vienen incluidos con Unity. |
| Se pierden los archivos. | Los archivos resultantes del modelado son extraviados o dañados. | 5% | 5 puntos | Mitigar. Los archivos se guardarán en el repositorio remoto. |

## 

## Texturizado de objetos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se terminan de texturizar los objetos. | La persona encargada de texturizar los objetos no termina de hacerlo. | 5% | 1 punto | Aceptar. |
| Se pierden los archivos. | Los archivos resultantes del texturizado son extraviados o dañados. | 5% | 1 punto | Mitigar. Los archivos se guardarán en el repositorio remoto siempre que se tenga la oportunidad. |
| Se encuentran texturas libres de uso en internet. | Nos ahorramos trabajo de texturizado debido a que encontramos texturas de uso libre en internet que sirven para el juego. | 90% | 1 punto | Aceptar. |

## Proceso de Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se detectan errores importantes durante el proceso de Testing. | Se encuentran problemas importantes en el juego después (y fuera) del proceso de Testing. | 20% | 1 punto | Mitigar. Además de llevar a cabo el proceso de Testing interno, se le pedirá informalmente a amigos y familiares que prueben el juego y ayuden a encontrar errores. |

## Compra o adquisición de la caja

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se encuentra la caja. | No se encuentra una caja adecuada para contener el disco del instalador del juego. | 5% | 1 punto | Aceptar. Se usará una caja usada que ya posee el equipo. |

## Diseño de la portada de la caja

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El diseño no es terminado. | El responsable de elaborar el diseño de la caja no lo termina antes de la fecha especificada. | 10% | 2 puntos | Mitigar. Se elaborará un diseño básico que solo incluya el nombre del juego. |

## 

## Impresión de la portada de la caja

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se imprime la portada. | La portada de la caja no se imprime. | 10% | 1 punto | Mitigar. La tarea de la impresión de la portada se calendarizará días antes de la fecha de entrega. |
| Las medidas de la impresión son incorrectas. | Ya sea por error del responsable o por problema técnico, la impresión resulta tener el tamaño incorrecto. | 20% | 1 punto | Mitigar. Se verificarán las medidas del archivo imprimiéndolo antes en una impresora casera en blanco y negro. |

## Elaboración de instalador binario

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se elabora el instalador. | Debido a problemas de cualquier tipo, el responsable de elaborar el instalador del juego no lo hace. | 5% | 1 punto | Mitigar. Si el responsable no termina, se recalendarizará la tarea y se cambiará de responsable. |

## Preparación de entregables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| El disco pasa a ser inservible. | Por problemas técnicos, el disco que llevará el instalador del juego “se echa a perder”; o sea, pasa a ser inservible. | 25% | 1 punto | Mitigar. Se comprarán al menos 3 discos de diferentes marcas. |

## Resolución de Issues

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| No se resuelven Issues de prioridad alta. | El tiempo no alcanza para resolver todas las Issues con prioridad alta. | 30% | 1 punto | Mitigar. Se asignarán a todos los miembros del equipo a resolver Issues para aumentar las probabilidades de terminar. |

## Entrega

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** | **Acción a tomar** |
| Los entregables no están listos. | Los entregables no están listos para el día de su entrega. | 5% | 20 puntos | Aceptar. |

# Programación Basada en Componentes

Mientras que la clásica Programación Orientada a Objetos (POO) puede ser, y es usada, el flujo de trabajo de Unity se construye alrededor de la estructura de componentes—lo cual requiere pensamiento basado en componentes.

## ¿Qué es un componente?

En el mundo de la programación, el concepto de “componentes” y “desacoplamiento” van de la mano. Se puede decir que un componente es una pieza pequeña de una maquina más grande. Cada componente tiene su propio trabajo específico, y pueden (óptimamente) cumplir su tarea o propósito sin la ayuda de fuentes externas. Adicionalmente, los componentes rara vez pertenecen a una sola máquina, y pueden ser unidos con varios sistemas para cumplir con su tarea, pero generar otro resultado. Esto se debe a que los componentes no solo no les importa saber sobre el sistema al que pertenecen, sino que tampoco saben que existe.

Considere un mando de Xbox 360. Tiene 2 sticks análogos, varios botones, gatillos, etc. No solo el mando entero es un componente, sino que también, cada aspecto individual del control es un componente.

El botón X puede: ser presionado; informar que fue presionado; ser liberado; informar que fue liberado. No tiene idea de que hay varios otros botones al lado de él y no le importa.

El mando en sí es un componente, compuesto de otros componentes (todos los botones, joysticks y gatillos), porque puede mandar datos sin importar donde esté conectado, y que tipo de objeto es (Xbox, PC, alguna creación de Arduino u otra cosa). El botón X y el mando en sí, no necesitan saber que juego está jugando, y hará su trabajo sin importarle el receptor de la información que mandan.

La función del mando es unidireccional, y su tarea nunca va a cambiar de receptor a receptor. Esto es lo que lo convierte en un componente exitoso; puede hacer su trabajo él solo, y puede hacer su trabajo conectado a múltiples dispositivos.

## ¿Cómo y por qué Unity favorece el uso de componentes?

Unity se construyó teniendo a los componentes en mente, y lo demuestra. Uno de los aspectos más valorados y distintivos de Unity, es que es un programa muy visual. Unity te deja ver todo en lo que estás trabajando en tiempo real. Esto significa que puedes probar tu proyecto, ver tu proyecto corriendo en una ventana separada, hacer cambios al código y ver esos cambios reflejados en tiempo real. La cantidad de poder que este sistema le da al desarrollador es inmensa. Todo esto fue posible gracias a la arquitectura basada en componentes de Unity.

## Pensando en componentes

La parte más difícil de trabajar con componentes, es aprender como estructurar tus proyectos al usarlo. Para la mayoría de los programadores, esto probablemente significa crear un montón de scripts que harán tareas más pequeñas y específicas.

La comunicación entre scripts también es un obstáculo decente, ya que tendrás más piezas y menos clases gigantes en las que cada objeto conoce a todos los demás. Obviamente, hay maneras de superar este obstáculo, como variables estáticas (para partes clave del juego como jugadores, puntuación, etc.), pero eso raramente funciona para cualquier cosa. También hay métodos avanzados para estructurar correctamente los componentes y mantenerlos desacoplados.

Por suerte, como Unity fue construido pensando en componentes, tiene una variedad de funciones integradas que nos ayudan a lograr esto. Hay funciones para obtener referencias a un componente especifico, para revisar todos los objetos y ver si contienen un componente especifico, etc. Con estas funciones uno puede fácilmente conseguir la información necesaria para crear una comunicación correcta entre componentes.

## Conclusión

Pensar en componentes puede ser difícil, pero definitivamente tiene sus beneficios. Al usar Unity, uno puede elegir el método que prefiera, pero definitivamente, pensar en componentes es la mejor opción.

# Proceso de Testing

## Objetivo

La función principal del proceso de testeo es el descubrimiento y documentación de los defectos en el videojuego (bugs).

## Metodología

Para empezar el proceso de Testing, se verificará mediante la prueba del videojuego que todos los casos de uso elaborados se cumplan. Posteriormente se revisará que todas las reglas oficiales del juego de mesa original sean aplicadas en el videojuego.

Todos los errores que sean identificados se documentaran, de tal manera que sea posible reproducir el error para su posterior estudio y corrección. El equipo de Testing documentará los siguientes aspectos:

* Acción que se realizó
* Resultado esperado
* Resultado obtenido
* Pasos para reproducirse.

Posteriormente se le asignará un grado de severidad y una prioridad.

Durante todo el proceso de Testing, se verificará constantemente la ortografía de los textos mostrados en el videojuego.

## Equipo de Testing

Debido a que solo habrá 2 personas trabajando en el proyecto, el equipo de testeo estará conformado por los mismos desarrolladores del videojuego.

## Aprobación

Para que el proceso de Testing pueda ser calificado como “terminado”, deberá haber cumplido con lo siguiente:

1. Se deberá haber revisado el cumplimiento de todos los casos de uso.
2. Se deberá haber verificado el cumplimiento de todas las reglas del juego original.
3. Se deberá haber comprobado la orografía de todos los textos del juego.