Jaipur 3D

Tabla de contenidos

[1 Declaración de alcance 3](#_Toc496471029)

[1.1 Objetivo: 3](#_Toc496471030)

[1.2 Requerimientos: 3](#_Toc496471031)

[1.3 Criterio de aceptación: 3](#_Toc496471032)

[1.4 Entregables: 3](#_Toc496471033)

[1.5 WBS inicial: 3](#_Toc496471034)

[1.6 Requerimientos de aprobación: 4](#_Toc496471035)

[1.7 Integrantes del equipo 4](#_Toc496471036)

[2 Work Breakdown Structure 5](#_Toc496471037)

[3 Roles 6](#_Toc496471038)

[4 Calendarización 7](#_Toc496471039)

[5 Análisis de Riesgos 8](#_Toc496471040)

[6 Programación Basada en Componentes 9](#_Toc496471041)

[6.1 ¿Qué es un componente? 9](#_Toc496471042)

[6.2 ¿Cómo y por qué Unity favorece el uso de componentes? 9](#_Toc496471043)

[6.3 Pensando en componentes 10](#_Toc496471044)

[6.4 Conclusión 10](#_Toc496471045)

[7 Proceso de Testing 11](#_Toc496471046)

[7.1 Objetivo 11](#_Toc496471047)

[7.2 Lista de pasos 11](#_Toc496471048)

[7.3 Equipo de Testing 11](#_Toc496471049)

[7.4 Aprobación 11](#_Toc496471050)

# 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:

* Tarjeta de video dedicada o integrada con al menos 512MB de memoria gráfica.
* DirectX 11 o superior.
* Sistema operativo: Windows 10.
* 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.

## Criterio de aceptación:

El juego debe aprobar nuestro 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 y Maestro en Ciencias Juan Francisco Algara Norzagaray.

## Integrantes del equipo

* Christian Ricardo Lugo Arellano
* José David Rochín Cerecer

# Work Breakdown Structure

# 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

|  |  |  |  |
| --- | --- | --- | --- |
| **Riesgo** | **Descripción** | **Probabilidad** | **Costo** |
| El videojuego no es terminado. | El equipo de trabajo no concluye el desarrollo del proyecto debido a que no siguió la calendarización. | 10% | 30 puntos |
| No alcanza el tiempo para resolver todas las Issues. | Los días que se tienen asignados en la calendarización del proyecto para resolver Issues, resultan no ser suficientes y solo se resuelven las Issues con prioridad alta. | 40% | 3 puntos |
| El instalador binario no funciona. | El instalador binario del juego, no funciona correctamente en algunas computadoras. | 10% | 5 puntos |
| El videojuego tiene fallos no detectados. | Después de la entrega, se detectan fallos no críticos en el videojuego que no se detectaron durante el proceso de Testing. | 20% | 5 puntos |
| La caja se daña. | La caja que incluye el disco con el instalador del juego, se entrega al asesor con daño físico. | 5% | 2 puntos |

# 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).

## Lista de pasos

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
* Condiciones específicas que se identificaron

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

## 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 videojuego se pueda calificar como “aprobado” por el proceso de Testing, no deberá tener errores con severidad alta sin resolver.