



Zetta's Bullet System

Documento de Requerimientos

Índice

Índice.....	1
1: Introducción.....	2
1.1: Propósito.....	2
1.2: Alcance.....	2
1.3: Requerimientos Técnicos.....	2
2: Requerimientos.....	2
2.1: Funcionales.....	2
2.1.1: Generador de Balas.....	2
2.1.2: Escala de Tiempo Local.....	3
2.1.3: Patrón de Disparo con Objetivo.....	3
2.1.4: Patrón Circular.....	4
2.1.5: Patrón de una Espiral.....	4
2.1.6: Patrón de Múltiple Espiral.....	5
2.1.7: Patrón de Polígono.....	5
2.1.8: Patrón de Estrella.....	6
2.1.9: Control del Generador.....	6
2.1.10: Editor General.....	7
2.1.11: Escena de Ejemplo Avanzada.....	7
2.1.12: Escena de Ejemplo Simple.....	8
2.2: No Funcionales.....	8
2.2.1: Editor Personalizado Intuitivo.....	8
2.2.2: Buen Rendimiento.....	8
2.2.3: Comentarios.....	8
2.2.4: Manual de Uso.....	9
3: Matriz de Requerimientos.....	9
4: Casos de Uso.....	10
4.1: Patrón Circular en un Juego Bullet Hell.....	10
4.2: Patrón de Espiral en un Minijuego de Precisión.....	10
4.3: Generación de Estrellas para Efectos Visuales en un Juego de Puzzles.....	11
5: Glosario.....	11
6: Referencias y Recursos.....	12

1: Introducción

1.1: Propósito

Proveer una herramienta intuitiva y fácil de usar para la generación de patrones de balas (proyectiles) en Unity. Diseñada para ser modular, altamente personalizable y de fácil integración en proyectos de videojuegos, la herramienta ofrece una solución eficaz para implementar mecánicas de disparo complejas. Además, permite una personalización avanzada del código, brindando libertad creativa a los usuarios más experimentados.

1.2: Alcance

La herramienta está dirigida a desarrolladores de videojuegos que deseen incorporar mecánicas de disparo estilo *bullet hell* en sus proyectos 2D. Asimismo, resulta útil para quienes busquen implementar patrones de disparo personalizados en otras mecánicas relacionadas con proyectiles o similares, así como efectos visuales en proyectos 3D.

1.3: Requerimientos Técnicos

- **Versión de Unity:** La herramienta se desarrollará utilizando la versión **6000.0.30f1** de Unity. Aunque es probable que funcione correctamente en versiones anteriores, se recomienda usar esta versión o posteriores para garantizar la mejor compatibilidad y rendimiento.

2: Requerimientos

2.1: Funcionales

2.1.1: Generador de Balas

ID: 01

Dependencias:

- Ninguna.

Descripción:

Este módulo será responsable de la creación e inicialización del generador de partículas (balas). Se estructurará mediante una jerarquía en la que un objeto *spawner* crea objetos hijos para contener a los generadores, y estos, a su vez, contienen los sistemas de partículas. El generador de balas se encargará de asignar las texturas, configuraciones, colisiones, colores y la forma de disparo de los proyectiles, así como de establecer los nombres de los sistemas de partículas según el ángulo de rotación.

Datos Específicos:

- Jerarquía: El sistema debe seguir una jerarquía objeto -> generador -> sistema de partículas.
- Forma de Disparo: El generador debe iniciar los sistemas de disparo según tres formas predefinidas del sistema de partículas de Unity: Línea recta, Rectangular y Cono.
- Inicialización: Los proyectiles deben ser inicializados con parámetros preestablecidos, tales como velocidad, tiempo de vida y cadencia de disparo.
- Textura: El sistema debe permitir asignar la textura del proyectil. Si no se asigna ninguna textura, el sistema debe detener su ejecución y generar un error.
- Colisión: El sistema debe permitir la selección de la capa de colisión para los proyectiles.
- Rotación: El sistema debe permitir el ajuste del ángulo de rotación para los proyectiles y asignar un nombre a cada sistema en función de dicho ángulo.
- Manejo de Errores: El sistema no debe permitir utilizar valores imposibles, tales como cantidades negativas, objetivos nulos, polígonos de menos de tres lados, entre otras cosas.

Prioridad: Obligatorio.

2.1.2: Escala de Tiempo Local

ID: 02

Dependencias:

- Ninguna

Descripción:

La herramienta proporcionará una escala de tiempo local, independiente de la escala global predeterminada de Unity, la cual permitirá pausar o reanudar el generador. La escala local no deberá interferir con la escala global, permitiendo al usuario usar la que mejor se adecúe a su proyecto.

Datos Específicos:

- Independencia: Si el usuario lo desea, debe poder ser capaz de usar la escala de Unity sin que eso afecte al funcionamiento del sistema.

Prioridad: Obligatorio.

2.1.3: Patrón de Disparo con Objetivo

ID: 03

Dependencias:

- 01: El generador de balas debe estar implementado y operativo para poder usar este patrón.

Descripción:

Este patrón permitirá que los proyectiles sean disparados hacia un objetivo preestablecido (*target*), el cual puede estar en movimiento. El sistema permitirá personalizar la cadencia de disparo, seleccionar el objetivo al que se disparará y definir el tiempo que tardará en seguir al objetivo.

Datos Específicos:

- Objetivo: El sistema debe permitir asignar un objeto como objetivo para que los proyectiles se dirijan hacia él. Si no se asigna ningún objetivo, el sistema debe detener su ejecución y generar un error.
- Seguimiento: El sistema debe ajustar continuamente su rotación para dirigir los proyectiles hacia el objetivo, con un control de enfriamiento preestablecido para la rotación.

Prioridad: Obligatorio

2.1.4: Patrón Circular

ID: 04

Dependencias:

- 01: El generador de balas debe estar implementado y operativo para poder usar este patrón.

Descripción:

Este patrón permitirá que los proyectiles sean disparados en forma circular, ofreciendo la capacidad de personalizar la cadencia de disparo y la cantidad de proyectiles que conformarán el círculo (precisión).

Datos Específicos:

- Precisión: El sistema debe permitir determinar de cuántos proyectiles estará formado el círculo del patrón.

Prioridad: Obligatorio

2.1.5: Patrón de una Espiral

ID: 05

Dependencias:

- 01: El generador de balas debe estar implementado y operativo para poder usar este patrón.

Descripción:

Este patrón permitirá disparar los proyectiles en una sola hilera, controlando la velocidad de rotación para dar la ilusión de una espiral. Además, contará con la posibilidad de activar el modo Espiral Invertida.

Datos Específicos:

- Espiral Invertida: El sistema debe ser capaz de generar una segunda hilera, con las mismas características que la original y que rote en sentido contrario, si así lo desea el usuario.

Prioridad: Obligatorio.

2.1.6: Patrón de Múltiple Espiral

ID: 06

Dependencias:

- 05: El patrón de una espiral debe estar implementado y operativo para poder extenderse a la forma de múltiples espirales.

Descripción:

Este patrón permitirá disparar los proyectiles en varias hileras, controlando la velocidad de rotación para dar la ilusión de una espiral. Además, contará con la posibilidad de activar el modo Espiral Invertida.

Datos Específicos:

- Espiral Invertida: El sistema debe ser capaz de generar una nueva hilera por cada hilera original, con las mismas características y que rote en sentido contrario, si así lo desea el usuario.

Prioridad: Obligatorio.

2.1.7: Patrón de Polígono

ID: 07

Dependencias:

- 01: El generador de balas debe estar implementado y operativo para poder usar este patrón.

Descripción:

Este patrón permitirá disparar los proyectiles en varias hileras, cada una con una velocidad distinta para dar la ilusión de un polígono regular. El usuario será capaz de controlar el número de lados, la cadencia de disparo, la cantidad de proyectiles que conformarán el polígono y la velocidad de rotación.

Datos Específicos:

- Precisión: El sistema debe permitir determinar cuántos proyectiles habrá entre cada lado del polígono.

Prioridad: Obligatorio.

2.1.8: Patrón de Estrella

ID: 08

Dependencias:

- 07: El patrón de polígono debe estar implementado y operativo para poder extenderse a su variación de estrella.

Descripción:

Este patrón permitirá disparar los proyectiles en varias hileras, cada una con una velocidad distinta para dar la ilusión de una estrella. El usuario será capaz de controlar el número de lados, la cadencia de disparo, la cantidad de proyectiles que conformarán el polígono, la velocidad de rotación y la profundidad de la estrella.

Datos Específicos:

- Precisión: El sistema debe permitir determinar cuántos proyectiles habrá entre cada lado de la estrella.
- Profundidad: El sistema debe permitir determinar qué tan cóncavos o convexos serán los lados interiores de la estrella. (Deseable)

Prioridad: Obligatorio.

2.1.9: Control del Generador

ID: 09

Dependencias:

- 01: El generador de balas debe estar implementado y operativo para poder llevar a cabo su control.
- 02: La escala de tiempo local debe estar implementada y operativa para poder llevar a cabo el control del generador.

Descripción:

La zona de control del generador permitirá pausar el generador, haciendo uso de la escala de tiempo local; detener el generador, activarlo e invertir la rotación.

Datos Específicos:

- Detención: El sistema debe ser capaz de detener el generador sin que este sea eliminado por completo, permitiendo a los proyectiles que ya han sido disparados continuar con su camino. Al hacer esto, debe detener a todos los generadores activos.

- Activación: El sistema debe ser capaz de crear y activar nuevos generados independientemente de cuántos generados activos haya.

Prioridad: Obligatorio.

2.1.10: Editor General

ID: 10

Dependencias:

- 03, 04, 06, 08: Todos los patrones deben estar implementados y operativos para poder ser modificado desde el editor.
- 09: El control del generador debe estar implementado y operativo para poder ser accedido desde el editor.

Descripción:

La herramienta contará con una interfaz única que permita modificar los elementos del generador. Este editor personalizado contendrá las secciones de interés sobre distintas características y opciones del generador.

Datos Específicos:

- Compatibilidad: El editor personalizado no debe interferir con otros *scripts* del proyecto que también implementen su propio editor.
- Configuración del Generador: El editor debe permitir seleccionar el patrón que se desea usar y cambiar sus opciones de personalización con base en eso.
- Configuración del Proyectil: El editor debe permitir cambiar las características de los proyectiles, tales como la velocidad, el tiempo de vida, su textura y color.
- Configuración del Control: El editor debe contar con botones personalizados que permitan realizar las acciones de inicio, pausa y de invertir la rotación.
- Manejo de Errores: El editor no debe permitir ingresar valores imposibles, tales como cantidades negativas, objetivos nulos, polígonos de menos de tres lados, entre otras cosas.

Prioridad: Obligatorio.

2.1.11: Escena de Ejemplo Avanzada

ID: 11

Dependencias:

- 03, 04, 06, 08: Todos los patrones deben estar implementados y operativos para poder ser modificado desde el editor.
- 09: El control del generador debe estar implementado y operativo para poder ser accedido desde el editor.

Descripción:

La herramienta, desde el proyecto de Unity, contará con una escena de ejemplo, que muestre cómo controlar el generador de manera profunda desde código.

Datos Específicos:

- Documentación: El código debe estar propiamente documentado, explicando a detalle cómo hacer uso de cada público del generador

Prioridad: Deseable.

2.1.12: Escena de Ejemplo Simple

ID: 12

Dependencias:

- 10: El editor personalizado debe estar implementado y operativo para poder mostrar su uso.

Descripción:

La herramienta, desde el proyecto de Unity, contará con una escena de ejemplo, que muestre cómo controlar el generador con una sola línea de código, personalizando el patrón desde su editor.

Datos Específicos:

- Documentación: El código debe estar propiamente documentado, explicando a detalle cómo activar correctamente el generador.

Prioridad: Deseable.

2.2: No Funcionales

2.2.1: Editor Personalizado Intuitivo

ID: NF1

Descripción:

El editor de la herramienta debe ser fácil de usar y accesible para personas sin conocimiento o interés en el apartado técnico del sistema.

2.2.2: Buen Rendimiento

ID: NF2

Descripción:

La herramienta debe ser capaz de generar miles de proyectiles que perduren por un tiempo considerable sin que esto lleve a pérdidas significativas en el rendimiento.

2.2.3: Comentarios

ID: NF3

Descripción:

El código debe estar propiamente comentado para ayudar a su comprensión, además, los comentarios deben estar escritos en inglés, con el fin de que la mayor cantidad de personas puedan entenderlos.

2.2.4: Manual de Uso

ID: NF4

Descripción:

La herramienta debe contar con un manual que explique paso a paso su instalación y uso, así como limitaciones y problemas conocidos. Dicho manual debe estar redactado en inglés.

3: Matriz de Requerimientos

	01	02	03	04	05	06	07	08	09	10	11	12
01												
02												
03	x											
04	x											
05	x											
06					x							
07	x											
08							x					
09	x	x										
10			x	x		x		x	x			
11			x	x		x		x	x			
12										x		

4: Casos de Uso

4.1: Patrón Circular en un Juego Bullet Hell

Descripción:

Un desarrollador desea implementar un jefe en un juego que dispare proyectiles en un patrón circular. Los proyectiles deben salir desde el centro del enemigo, con una cadencia ajustable a lo largo de la batalla.

Pasos:

1. **Inicialización:** Crear un objeto en Unity que contenga el componente “BulletSystem” de ZBS.
2. **Configuración:** Desde el editor personalizado de ZBS, seleccionar el patrón circular y editar sus características de la forma que se desee.
3. **Personalización Profunda:** En un script personalizado, donde se controlará la batalla, ajustar la cadencia de disparo dependiendo de la fase de la pelea.
4. **Prueba y Ajustes:** Ejecutar la escena y ajustar parámetros según sea necesario para equilibrar la dificultad.

Resultado Esperado:

Los proyectiles deben dispararse de manera uniforme en todas direcciones, generando un círculo perfecto alrededor del jefe, aumentando o disminuyendo su velocidad de disparo en función de la fase de la batalla.

4.2: Patrón de Espiral en un Minijuego de Precisión

Descripción:

Un usuario está desarrollando un minijuego donde el jugador debe esquivar proyectiles disparados en forma de espiral. La velocidad de rotación y el número de hileras deben poder configurarse.

Pasos:

1. **Inicialización:** Crear un objeto en Unity que contenga el componente “BulletSystem” de ZBS.
2. **Configuración:** En el editor, seleccionar el disparo de espiral múltiple y habilitar la opción de espiral invertida.
3. **Ajustes de Velocidad:** En un script personalizado, configurar la velocidad de rotación y el tiempo de vida de los proyectiles. Si se desea, también variar la cantidad de espirales y cambiar el sentido de rotación cada cierto tiempo.
4. **Pruebas:** Ejecutar la escena y observar el comportamiento de los proyectiles para realizar ajustes.

Resultado Esperado:

Los proyectiles deben formar una espiral, con movimientos fluidos y configuraciones que se ajusten a la dificultad deseada; variando su velocidad, rotación y cantidad según se haya establecido.

4.3: Generación de Estrellas para Efectos Visuales en un Juego de Puzzles

Descripción:

Un desarrollador busca implementar un efecto visual con forma de estrella al completar un nivel en un juego de puzzles. Las puntas de la estrella deben aumentar en función de la dificultad del puzzle.

Pasos:

1. **Inicialización:** Crear un objeto en Unity que contenga el componente “BulletSystem” de ZBS.
2. **Configuración:** Seleccionar el patrón de estrella en el editor de ZBS y ajustar los parámetros generales de la estrella.
3. **Personalización de Puntas:** En el script encargado de controlar el final del nivel, añadir las puntas que la estrella tendrá, así como otros valores de interés, tales como la rotación, velocidad y tiempo de vida.
4. **Prueba de Transición:** Ejecutar el patrón y verificar que el efecto sea el adecuado y coincida con el término del nivel.

5: Glosario


- **Bullet Hell:** Subgénero de videojuegos caracterizado por un alto volumen de proyectiles en la pantalla y su alta dificultad.
- **Jerarquía:** Estructura organizativa en Unity donde los objetos están relacionados en niveles padre-hijo, lo que define dependencias y transformaciones relativas.
- **Sistema de Partículas:** Componente de Unity utilizado para simular diversos efectos visuales.
- **Spawner:** Objeto que genera otros objetos.
- **Target:** Objeto o entidad hacia el cual se dirigen los proyectiles disparados por un patrón con objetivo.
- **Unity:** Motor de desarrollo de videojuegos utilizado para crear experiencias interactivas en 2D, 3D y realidad virtual.
- **ZBS:** Zetta’s Bullet System, la herramienta a desarrollar.

6: Referencias y Recursos

Para obtener más información sobre los conceptos utilizados en el desarrollo de Zetta's Bullet System y para complementar el uso de esta herramienta, así como realizar modificaciones/expansiones de la misma, se recomienda ver los siguientes recursos:

1. How to Make a Bullet Hell Engine in Unity Using Particles (Tutorial)

- a. **Descripción:** ZBS está fuertemente inspirado en el siguiente video tutorial, donde se utiliza el sistema de partículas de Unity para crear una lluvia de proyectiles.

- b. **Enlace:**  [How to Make a Bullet Hell Engine in Unity Using Particles \(Tutor...](#)

2. ParticleSystem

- a. **Descripción:** Guía oficial de Unity sobre el uso y configuración de sistemas de partículas, que incluye ejemplos y referencias detalladas.

- b. **Enlace:** [Scripting API: ParticleSystem](#)