Desarrollo e Implementación

Una visión general de nuestro proceso de desarrollo y las características clave implementadas.

Paradigma de Desarrollo Ágil

Iteración Continua

Sistemas añadidos progresivamente: daño, jugador, enemigo, proyectil. Funcionalidad mínima que se amplía.

Modularidad

Diseño independiente y reutilizable: daño, IA, proyectiles separados. Facilita pruebas y modificaciones.

Pruebas y Ajustes Rápidos

Lógica de depuración y debug prints en eventos clave. Validación y ajustes inmediatos.

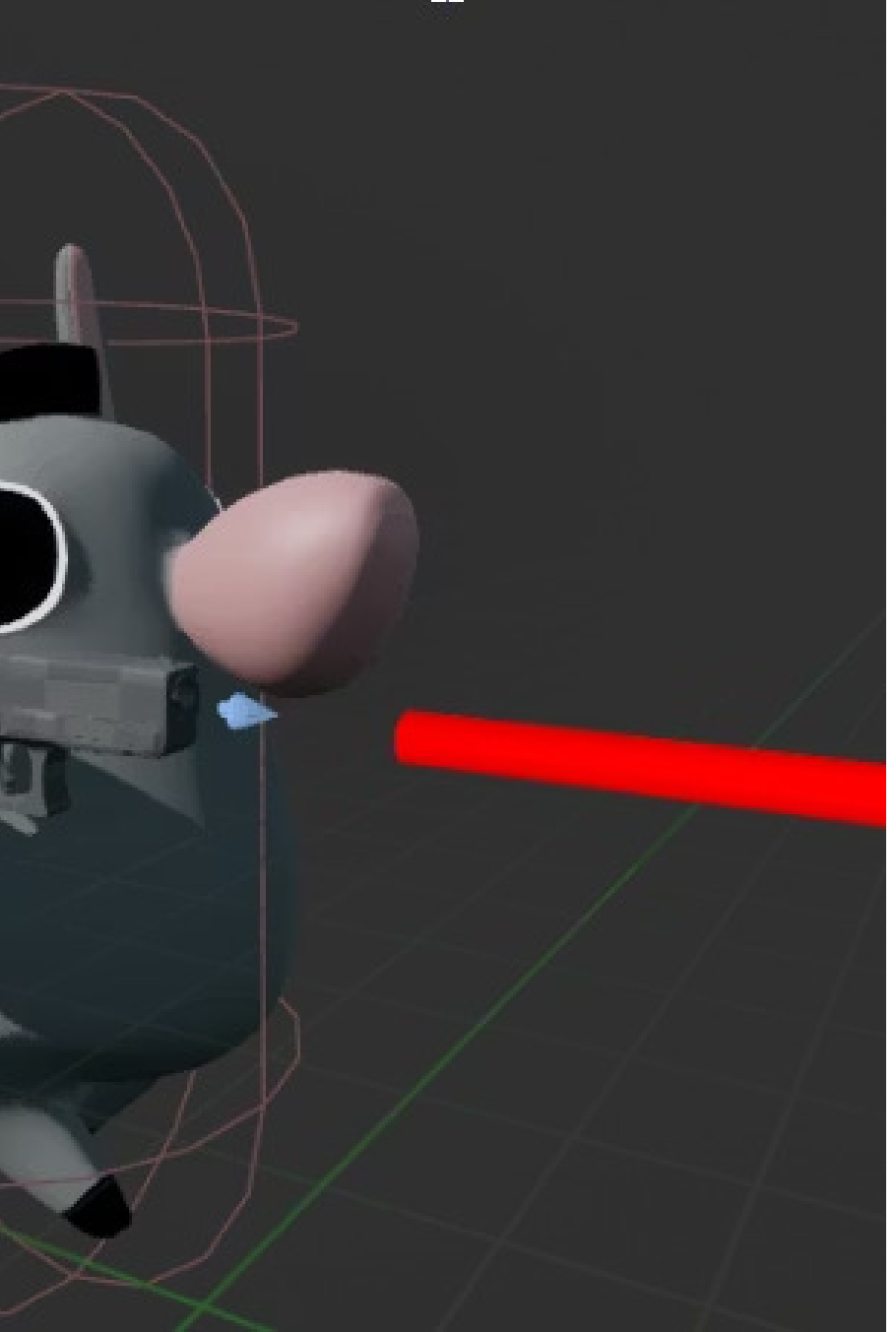
BP_Player: Funcionalidades Clave

Disparo de Proyectil

- Retardo (0.1s) y selección aleatoria de clase.
- SpawnActor con parámetros de velocidad y gravedad.
- Ignora colisiones con el jugador.
- Aplica 10 de daño de tipo "Melee" al impactar.

Mapeo de Inputs y HUD de Vida

- Carga el mapping de Enhanced Input al inicio.
- Crea y asigna el widget de barra de vida al personaje.



BP_Player: Interacción con la UI y Movimiento

Toggle de Menú de Settings

- Alterna visibilidad del menú de ajustes.
- Ajusta el modo de entrada y el cursor para UI.

Pausa del Juego

- Crea/usa el widget de pausa.
- Congela el juego y alterna modo de entrada/ratón.

Movimiento y Salto

La locomoción utiliza ControlRotation para el movimiento en plano X/Y y maneja el salto con IA_Jump.

Sistema de Daño: BPI_Damagable y BPC_DamageSystem

BPI_Damagable

Define el contrato para actores "dañables": GetCurrentHealth, GetMaxHealth, Heal, TakeDamage.

S_DamageInfo

Estructura de datos por ataque: Cantidad, Tipo, Respuesta, Invulnerabilidad, Bloqueo, Parry, Interrupción.

Enumeraciones de Apoyo

E_DamageResponse (reacción) y E_DamageType (origen del daño).

Función TakeDamage y Macro CanBeDamage

Función TakeDamage

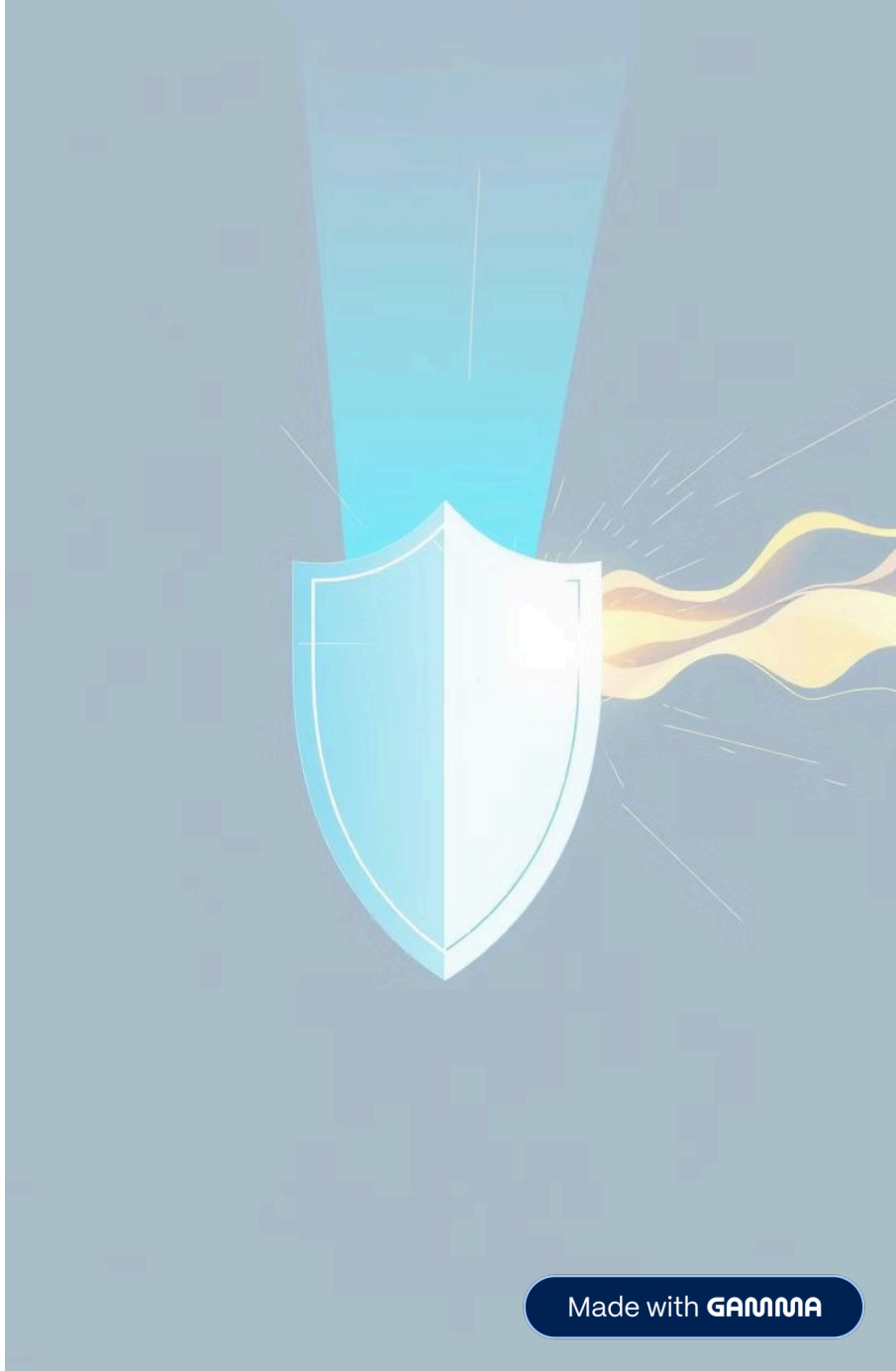
- Gestiona la lógica principal al recibir impacto.
- Evalúa con CanBeDamage si el daño se aplica, bloquea o ignora.
- Dispara eventos OnBlocked, OnDeath, OnDamageResponse.

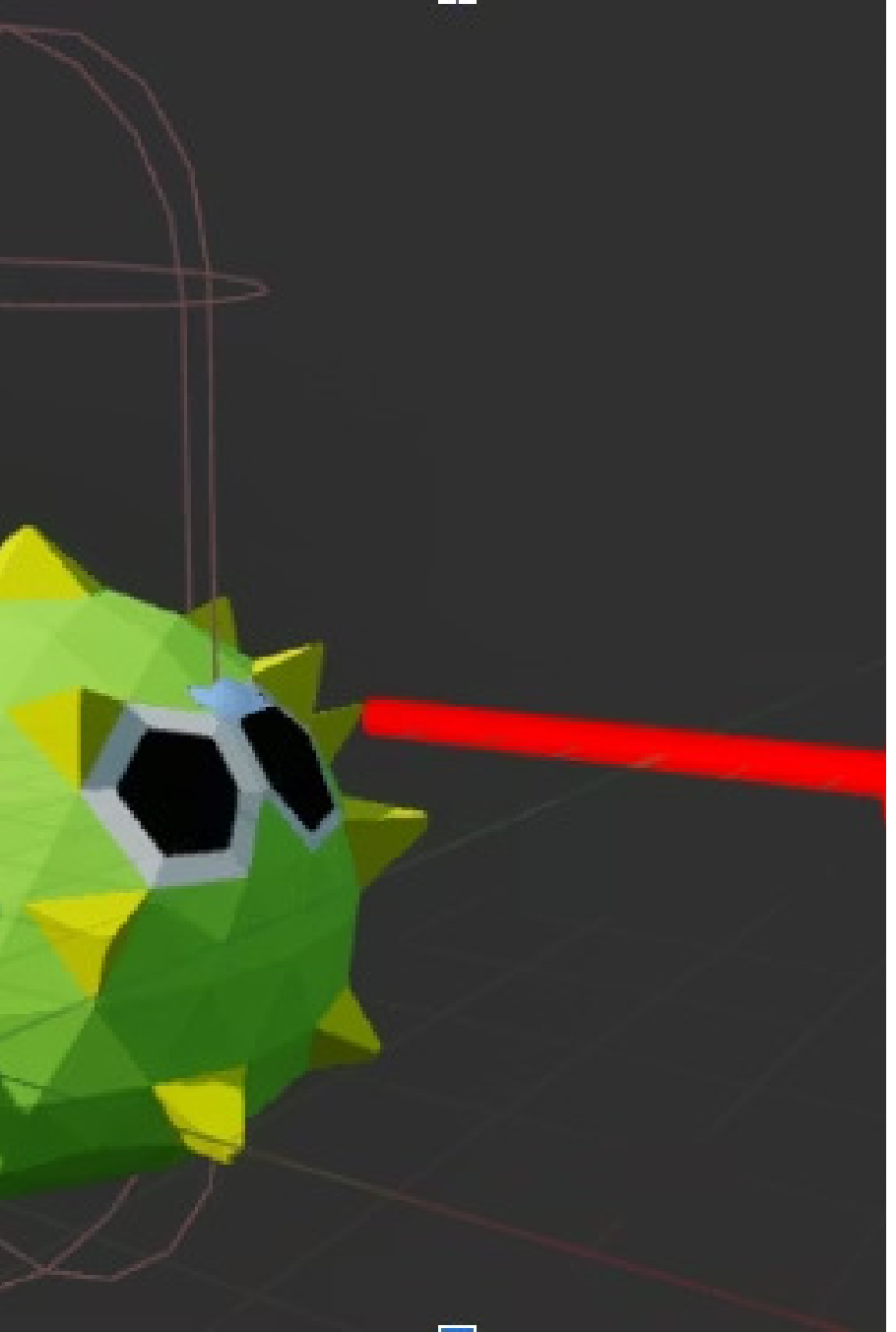
Macro CanBeDamage

- Filtro previo antes de aplicar daño.
- Comprueba estado (muerto, invencible) y bloqueo.
- Salidas: Block Damage, Do Damage, No Damage.

Función Heal

Gestiona la curación del actor, sumando el monto a la salud y ajustando el valor entre 0 y MaxHealth.





BP_Enemy: Comportamiento y Lógica

Movimiento e Input

Movimiento básico con EnhancedInputAction IA_Move. Añade input mapping al PlayerController.

Inicialización y HUD

Registra al enemigo en GameMode, configura timer para ataques y crea widget de barra de vida.

Lógica de Ataque y Daño

Detecta actores cercanos con Sphere Trace. Aplica daño (10 de tipo Melee) a través de BPI_Damagable. Reproduce efectos y se destruye al morir.

BP_Projectile: Configuración y Efectos

1

Configuración Inicial

Ignora colisiones con el instigador. Rota hacia el objetivo si es válido.

2

Rotación y Trayectoria

Calcula dirección y velocidad para el movimiento hacia el objetivo.

3

Impacto y Efectos

Dispara evento de impacto, genera efectos visuales/sonoros y se destruye.

4

Comportamiento Homing

Habilita modo Homing y asigna objetivo si la propiedad está activada.

WB_MainMenu: Navegación Principal

Botón New Game

Reproduce sonido, muestra pantalla de carga (2s), abre Nivel 1, y ajusta control a Game Only.

Botón Credits

Reproduce sonido, elimina menú principal y crea/añade WB_Credits.

Botones Quit Game y Settings

Quit Game finaliza la aplicación. Settings reproduce sonido, elimina menú principal y crea/añade WB_Settings.

WB_Pause y WB_Settings: Menús de Juego

WB_Pause

- Reanudar: despausa, oculta cursor, Game Only.
- Quit: alterna menú de confirmación.
- Main Menu: carga nivel principal.
- Desktop: cierra aplicación.
- Credits/Settings: crea y muestra widgets correspondientes.

WB_Settings

- Cierre: con IA_Pause, elimina del viewport.
- Navegación: conmutación entre secciones (Back, Screen, Volume, Control).
- Ajustes gráficos: aplica comandos de consola (resolución, texturas, sombras, FPS).
- Key Mappings: muestra bindings de teclas.
- Volúmenes: carga/guarda desde SaveGame, aplica con SoundMix overrides.