
BANANARANG

# ÍNDICE

***Game Design Document · v1.1 · Materia: Diseño de Sistemas de Juegos · Prof. Nicolás Lince***

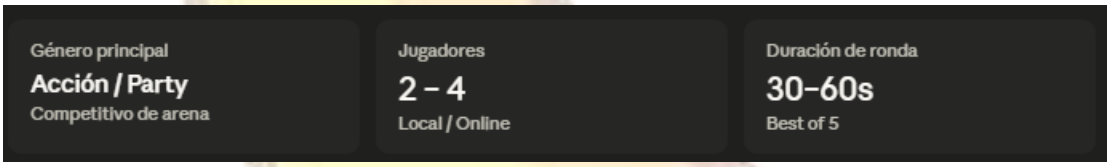
# PARTE 1 — PITCH / OVERVIEW

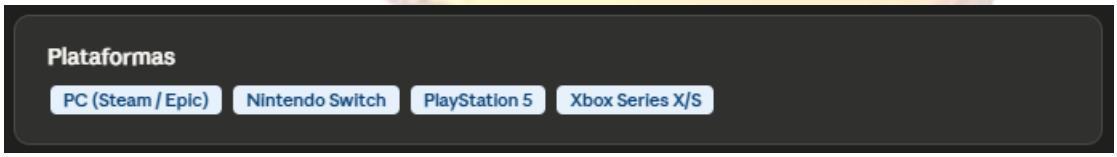
## Ficha técnica

- Party Game Arena.
- Brawler.
- PC · Switch · PS5 · XSX
- 2 – 4 Jugadores.
- Pixel Art · Retro.

## One-sentence pitch

- *"Un party game competitivo de 2 a 4 jugadores donde monos equipados con plátanos-búmeran se enfrentan en dinámicas arenas retro, combinando la estrategia espacial de rebotes con la recuperación activa del arma."*




## Audiencia objetivo

- *Jugadores casuales de 13–30 años, fanáticos del party game local (Boomerang Fu, TowerFall). Ideal para sesiones cortas con amigos. No requiere experiencia previa en brawlers.*

### Propuesta de valor única

- *A diferencia de la competencia, el proyectil **nunca desaparece**: rebota, se queda en el escenario como peligro latente, y puede ser invocado de vuelta (**Recall**) convirtiendo el retorno en un arma adicional. El mapa se convierte en una red táctica viva.*

# PARTE 2 — DISEÑO DE SISTEMAS Y MECÁNICAS

## ¿Qué es un sistema en Bananarang?

- *Un sistema es un conjunto de elementos relacionados que funcionan como un todo. Bananarang está compuesto por sistemas simples que se interconectan para formar sistemas complejos que soportan el core loop.*

***Sistemas simples (cada uno cumple una función específica)***

- *Movimiento del mono.*
- *Lanzamiento de banana.*
- *Física de rebote.*
- *Recall (retorno).*
- *Detección de impacto.*
- *Cooldown de Dash.*
- *Vidas por ronda.*
- *Puntuación.*

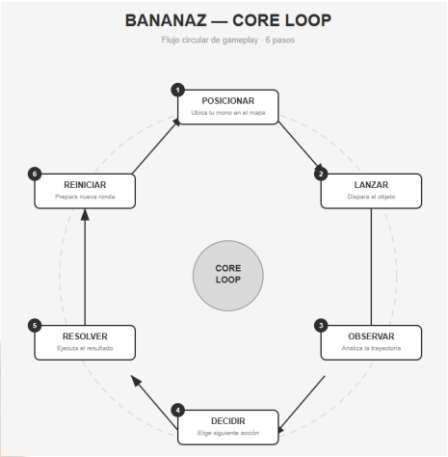
***Sistemas complejos (varios sistemas simples interconectados)***

- *Sistema de proyectil persistente.*
- *Sistema de control del jugador.*
- *Sistema de variantes de banana.*
- *Sistema de power-ups.*
- *Sistema de puntuación y rondas.*

*El conjunto de nuestros sistemas permite soportar el CORE LOOP.*

### Core Loop: Posicionar → Lanzar → Observar → Decidir → Resolver → Reiniciar

***(Loop circular no lineal)***




***Nuestros sistemas cumplen con las reglas y pautas esperadas, las cuales son:***

### *Reglas de diseño de los sistemas*

- ***Comprensible***

*El jugador entiende intuitivamente que la banana no desaparece: el comportamiento visual (trail de color) comunica al dueño de cada proyectil.*

- ***Consistente***

*Las reglas de rebote se aplican igual para todas las variantes de banana y en todos los mapas. Ninguna superficie cambia las reglas base (salvo los modificadores que podrían aparecer en futuras versiones)*

- ***Predecible***

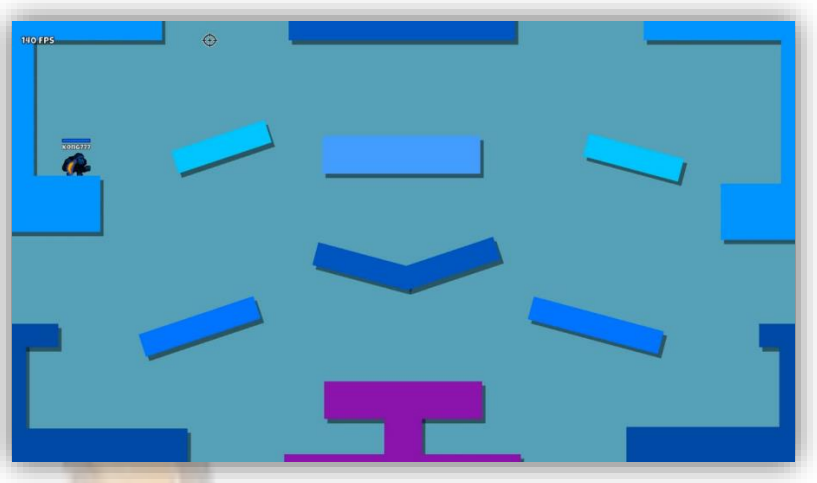
*La trayectoria de rebote usa vectores reflejos sobre normales de superficie. El diseñador puede calcular y prever el resultado de cualquier ángulo de lanzamiento en cualquier mapa.*

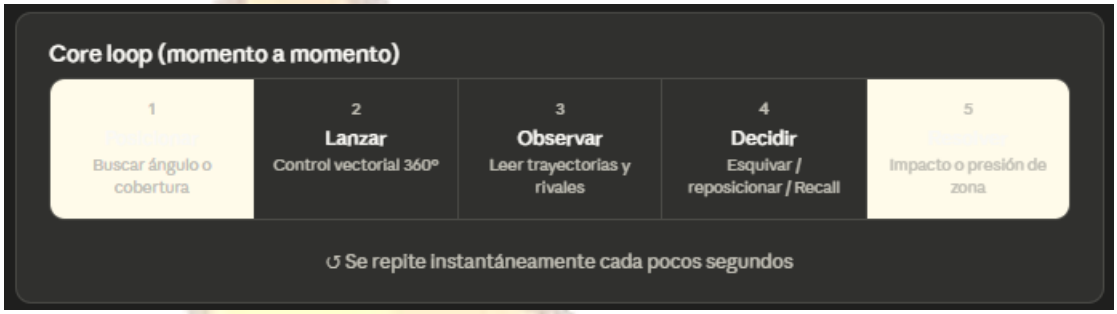
- ***Extensible***

*Agregar una nueva variante de banana (ej: Banana de Hielo) solo requiere definir nuevos modificadores sobre el sistema base sin romper los existentes.*

- ***Elegante***

*Un solo botón (Recall) cumple tres funciones tácticas: recuperar el arma, matar en el regreso, y limpiar el escenario de proyectiles propios. Máximo impacto con mínima complejidad de input.*






***Sub-loops dentro del core loop***

- ***Sub-loop A : Combate***

*Lanzar → Rebotar → Recall · Este loop de duración ~2-5 segundos es el más frecuente y define el ritmo arcade.*

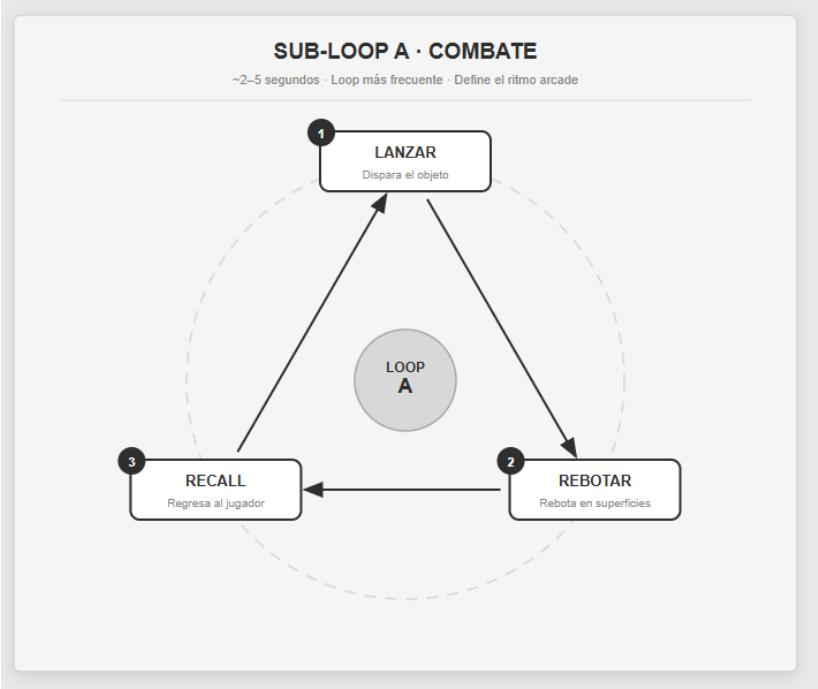
- ***Sub-loop B : Ronda***

*Spawn → Combate (sub-loop A) → Eliminación → Ganador de ronda. Duración: 30–60 segundos.*

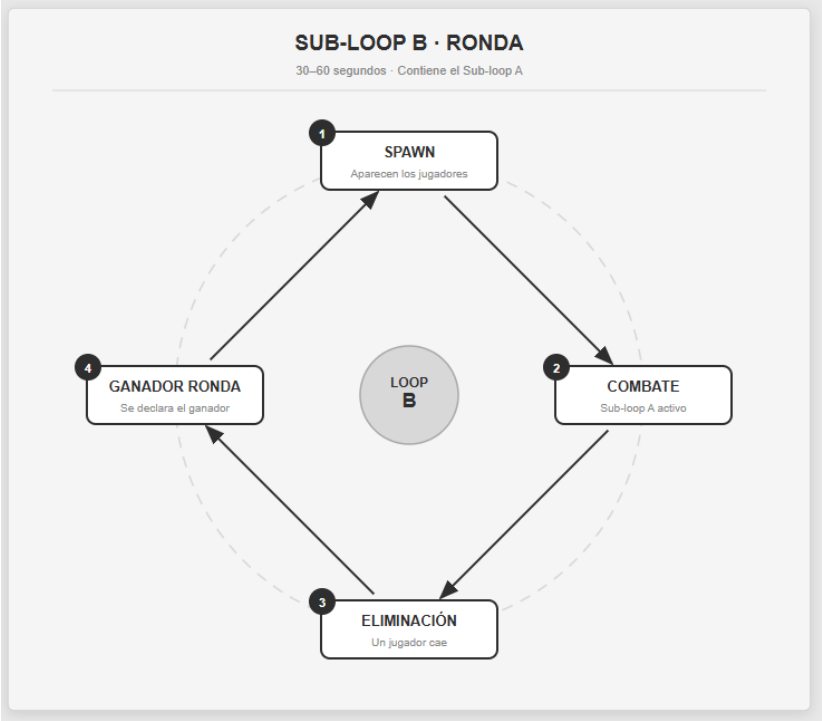
- ***Sub-loop C : Partida***

*Jugar rondas → Acumular puntos → Mejor de 3 → Pantalla de galardones → Nueva partida.*

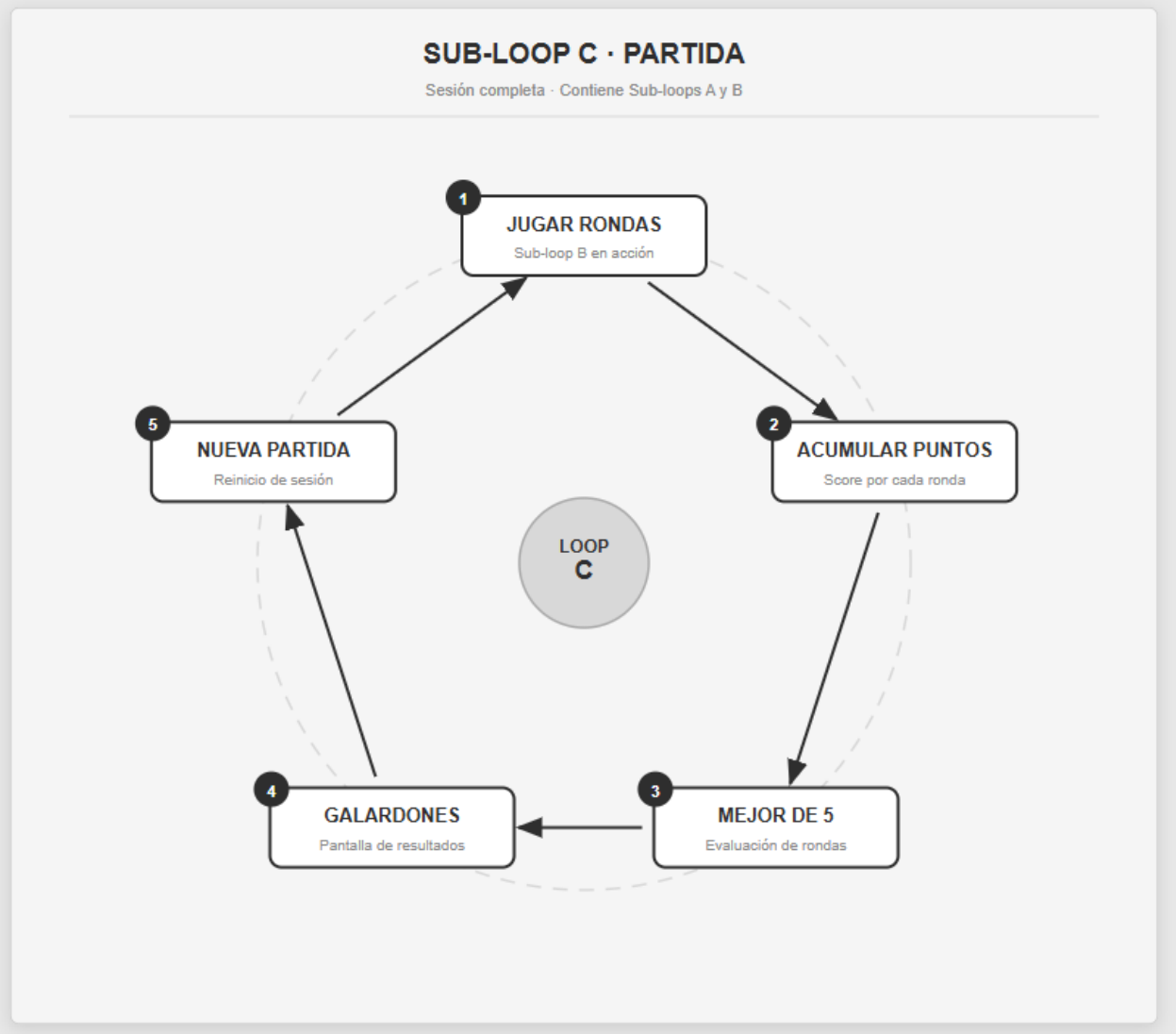
- ***Sub-loop A : Combate***




- ***Sub-loop B · Ronda***



- ***Sub-loop B : Partida***



## High Level Design vs Low Level Design

- ***High Level***

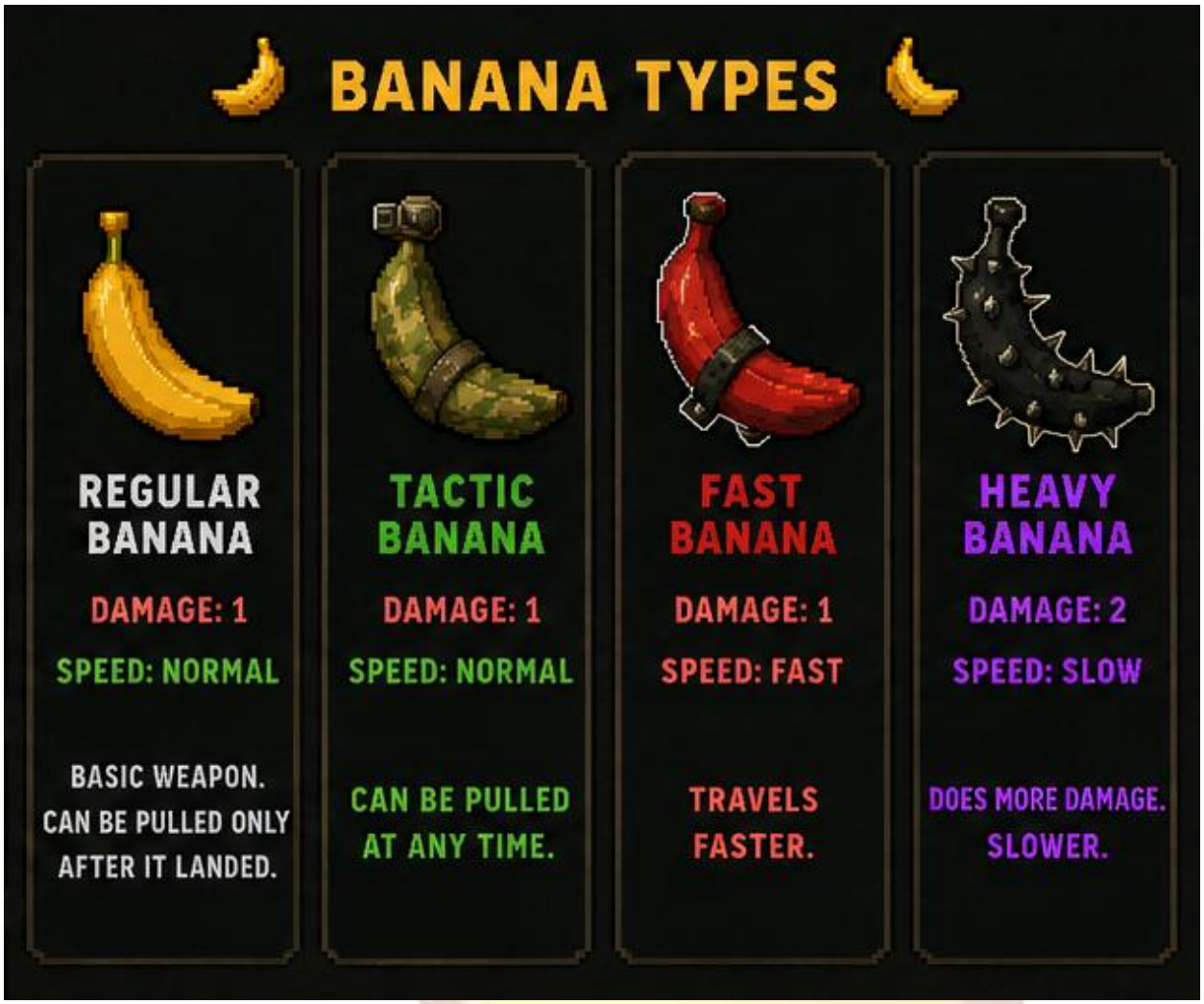
*Core loop, géneros, modos de juego, número de rondas, sistema de puntuación general, variantes de banana como categorías.*

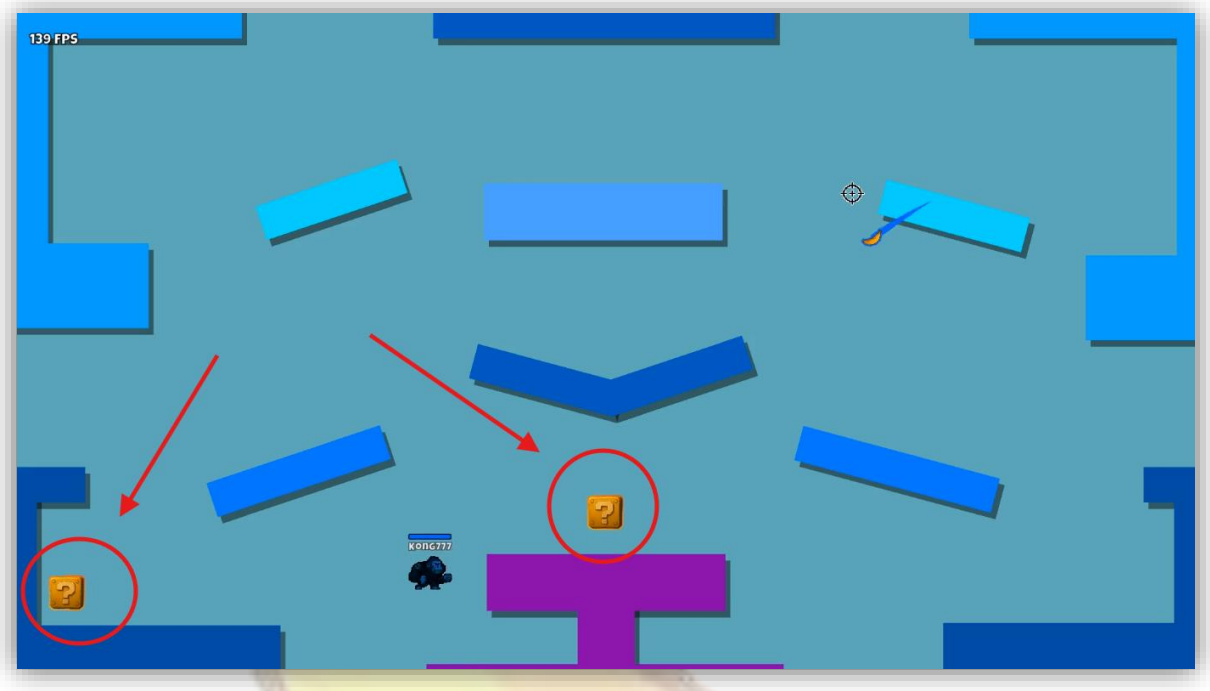
- ***Low Level***

*Velocidad exacta de cada variante, coeficientes de rebote, valores de cooldown del Dash (0.15s), multiplicadores de puntuación (+100/+250/+300) bonus de velocidad (+35%), etc. Entre otras variables que pueden haber.*

## Mecánica única: el proyectil persistente y retornable

La banana estándar mantiene su energía cinética al rebotar usando el vector reflejo sobre normales de superficie. Nunca desaparece hasta ser invocada por su dueño (Recall) o ser capturada por un rival.

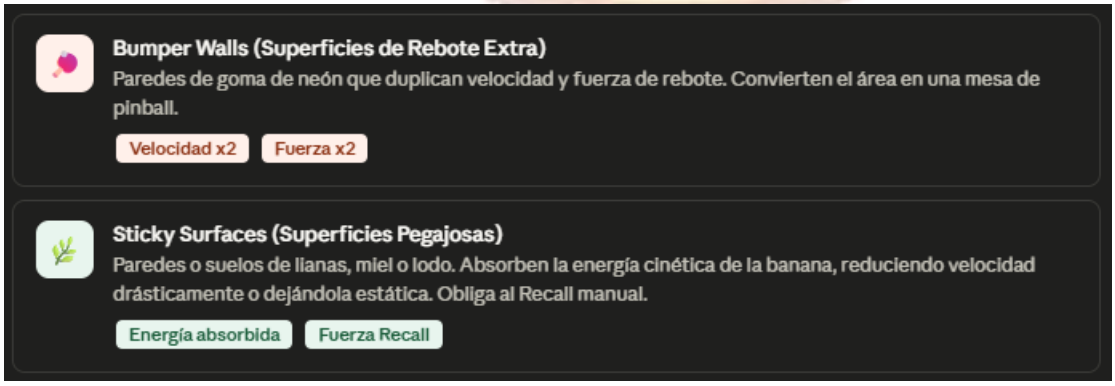

BANANA TYPES
REGULAR BANANA
DAMAGE: 1
SPEED: NORMAL
BASIC WEAPON. CAN BE PULLED ONLY AFTER IT LANDED.
TACTIC BANANA
DAMAGE: 1
SPEED: NORMAL
CAN BE PULLED AT ANY TIME.
FAST BANANA
DAMAGE: 1
SPEED: FAST
TRAVELS FASTER.
HEAVY BANANA
DAMAGE: 2
SPEED: SLOW
DOES MORE DAMAGE. SLOWER.



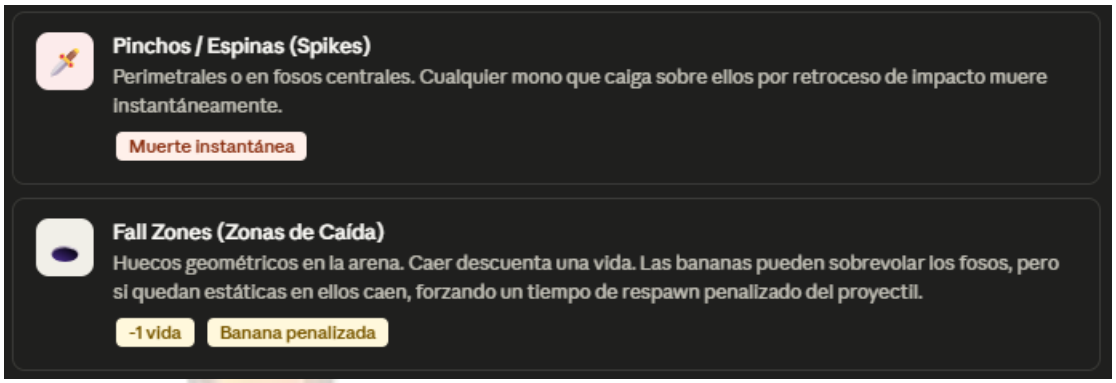

## Diseño de entornos – (NO IMPLEMENTADO – FUTURAS FEATURES)

El entorno actúa como extensión orgánica del combate. La geometría interactiva reemplaza mecánicas complejas: el mapa es un participante activo del juego, no un decorado.

- **Superficies especiales**



### Peligros del mapa (hazards)



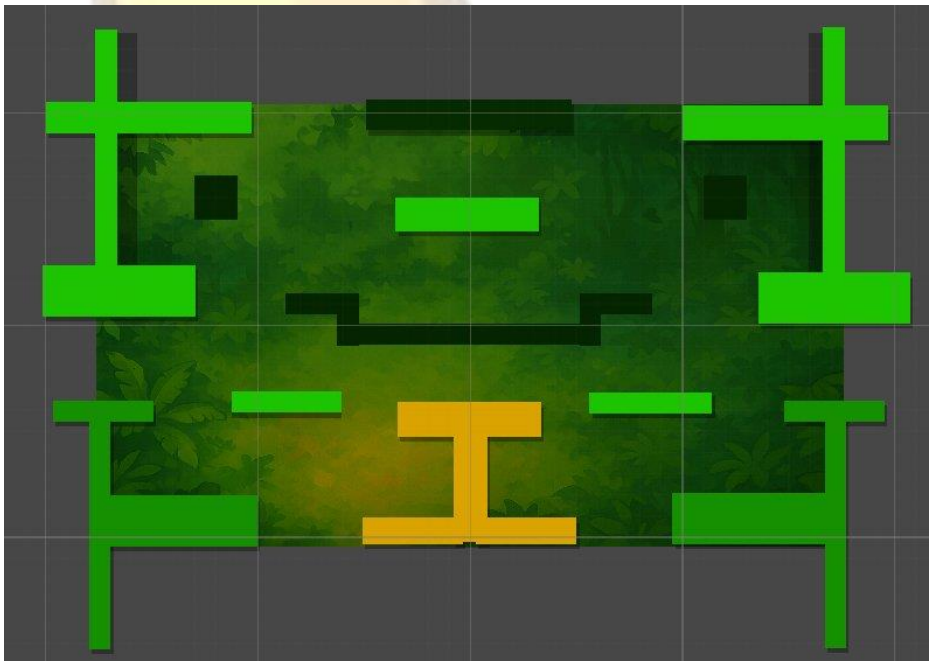

Al hacer que el entorno sea un participante activo en lugar de un simple decorado, el juego logra muchísima profundidad estratégica sin abrumar al jugador con controles complejos o combinaciones de botones interminables. La interacción entre una mecánica base sencilla (lanzar/recuperar la banana) y un entorno complejo genera situaciones únicas en cada partida.

### *Niveles: Referencias y Layout*

*Cada nivel grayboxeado en Unity parte de una referencia visual de TowerFall (la influencia directa de Bananarang en términos de diseño de plataformas y espaciado). Las imágenes de la izquierda muestran la referencia tomada; las de la derecha, el layout de colisión resultante implementado en el editor.*

***Nivel 1***

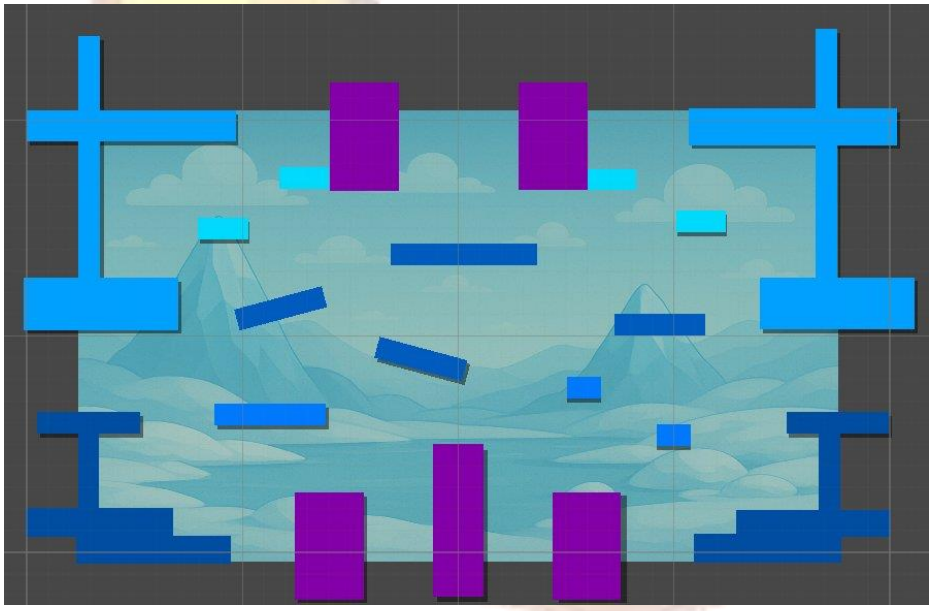
*Referencia: TowerFall — nivel de cueva con plataformas colgantes y cofres.*



*Layout final (Unity): Nivel 1 — plataformas en cruz simétricas, viga central de apoyo.*

***Nivel 2***

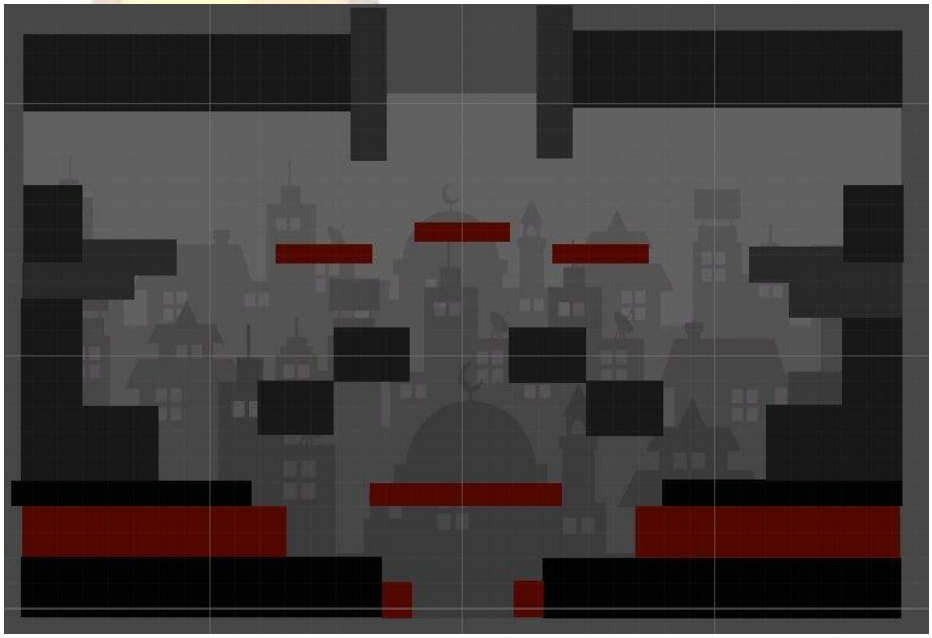
*Referencia: TowerFall Ascension — templo sumergido con estructura central.*



*Layout final (Unity): Nivel 2 — ambientación helada, plataformas diagonales y verticales.*

***Nivel 3***

*Referencia: TowerFall Ascension — nivel de cielo con plataformas colgantes de cadenas.*



*Layout final (Unity): Nivel 3 — ambientación urbana nocturna, plataformas anchas inferiores.*

# PARTE 3 — ARTE, UX Y GAME FEEL

## Dirección de arte & Game Feel

### *Estilo visual*

Pixel Art de 16/32 bits potenciado con iluminación dinámica moderna, efectos de partículas fluidos y fondos animados (junglas tropicales, templos con cascadas de píxeles). Estética retro con juiciness contemporáneo.

### *Controles*

Input a 0 frames de delay. Aceleración/desaceleración con curvas suaves de fricción.

🔊 *Pasos pixelados sobre madera / roca*

### *Física banana*

Trail de color por propietario. Trayectorias nítidas con líneas de partículas.

🔊 *"¡Clack!" o "¡Bink!" en cada rebote*

### *Impacto / Kill*

Freeze Frame 2–4ms + Screen Shake proporcional a la velocidad del plátano.

🔊 *Explosión sorda + grito caricaturesco*

### *Recall*

Animación magnética: banana gira en su eje en línea recta con destellos eléctricos.

🔊 *Whoosh inverso + silbido ascendente + golpe seco*

### *Power-up*

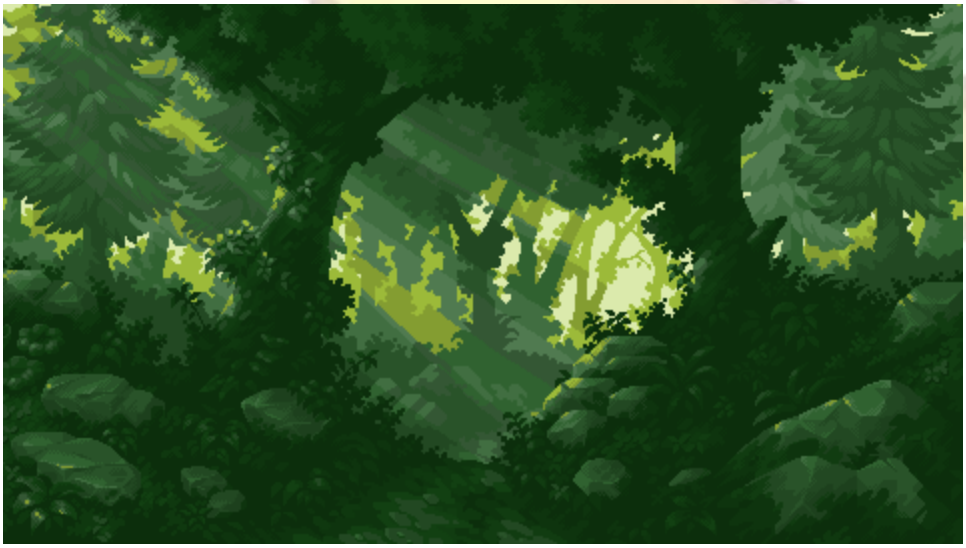
Flash de color del tipo de power-up. Aura visible sobre el mono activo.

🔊 *Jingle corto tipo 8-bit*

### *Eliminación*

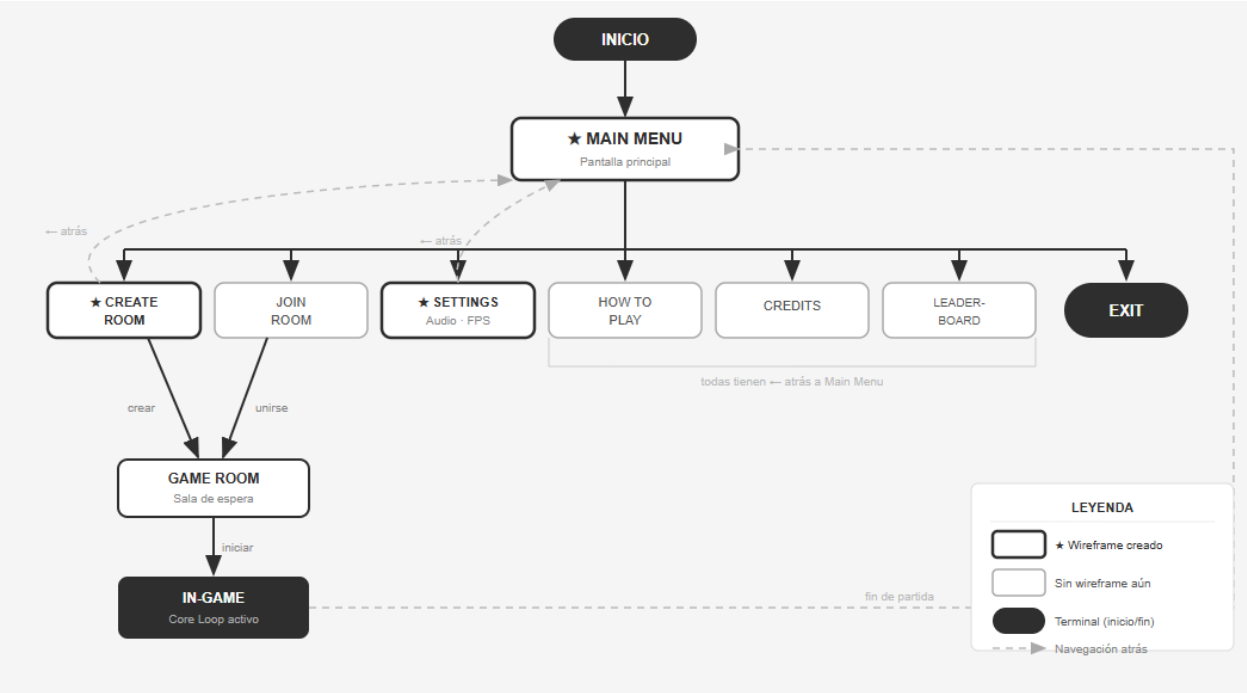
Explosión de píxeles del color del mono eliminado. Reaparición con efecto de destello.

🔊 *Chiptune dramático de 1 segundo*

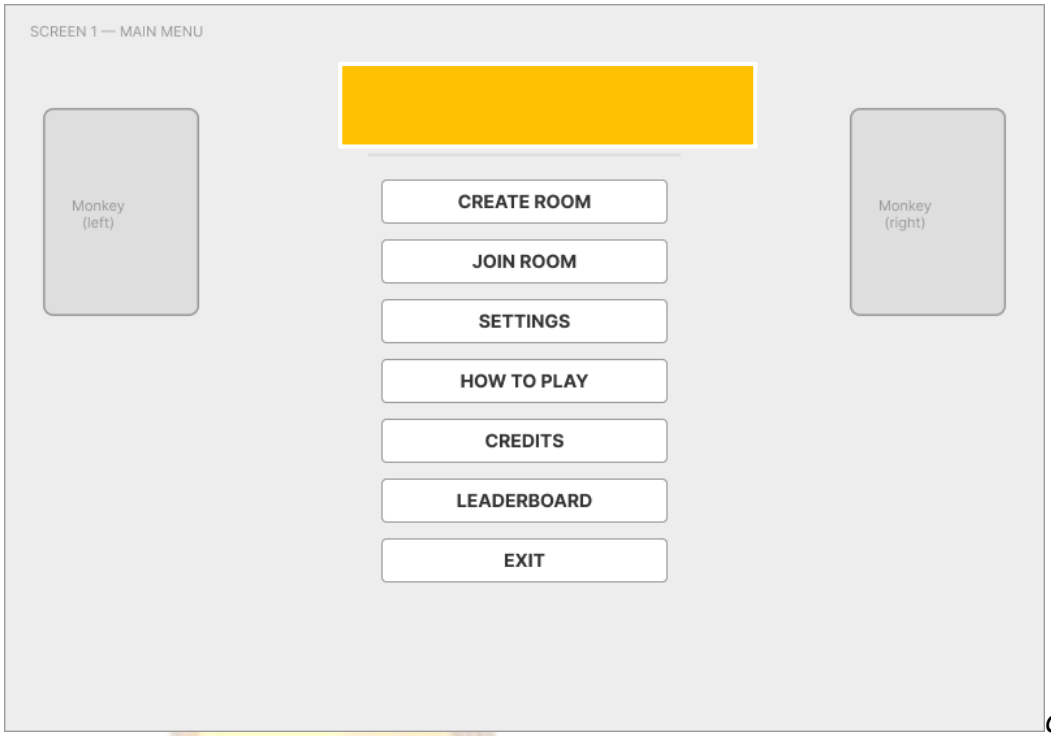



BANANARANG

## Diagrama de Flujo de Pantallas y Menús

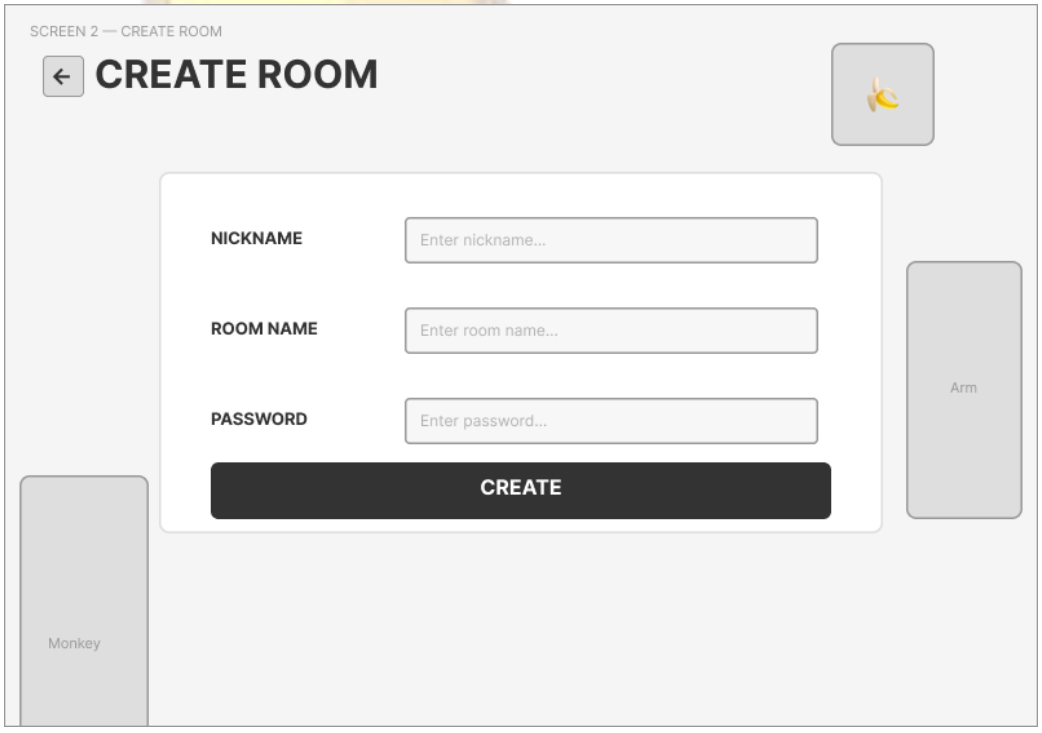



*Idea principal de creación de flujo de pantallas del juego*

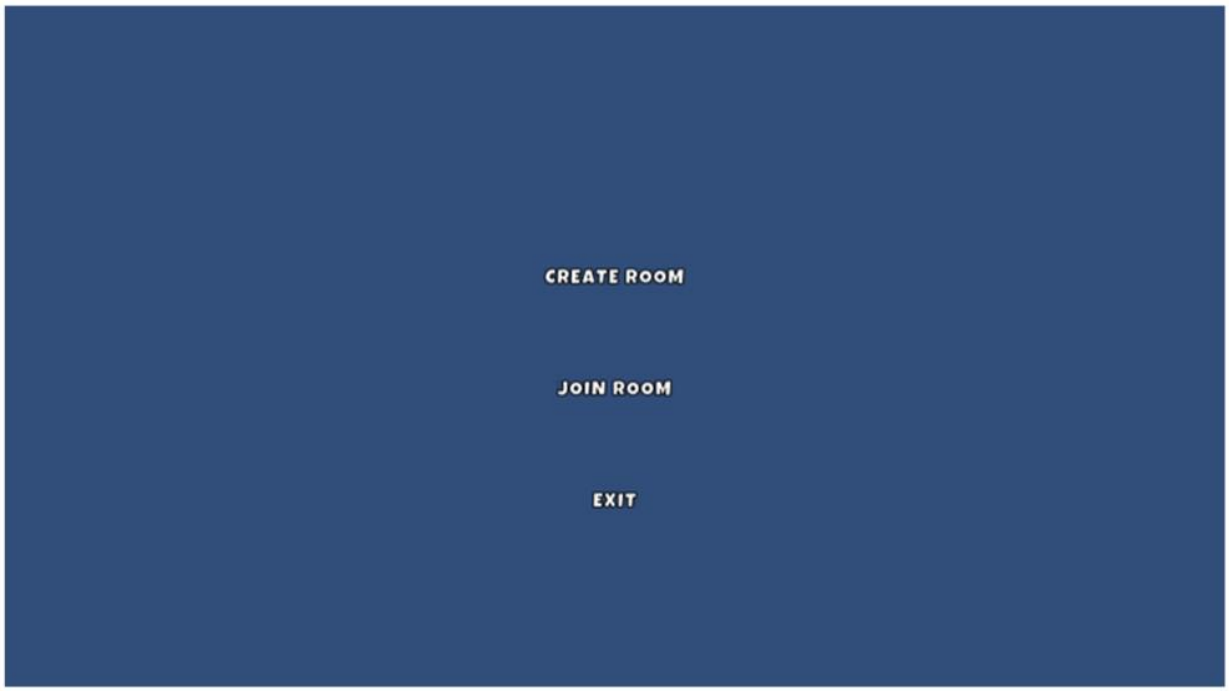



*dad*

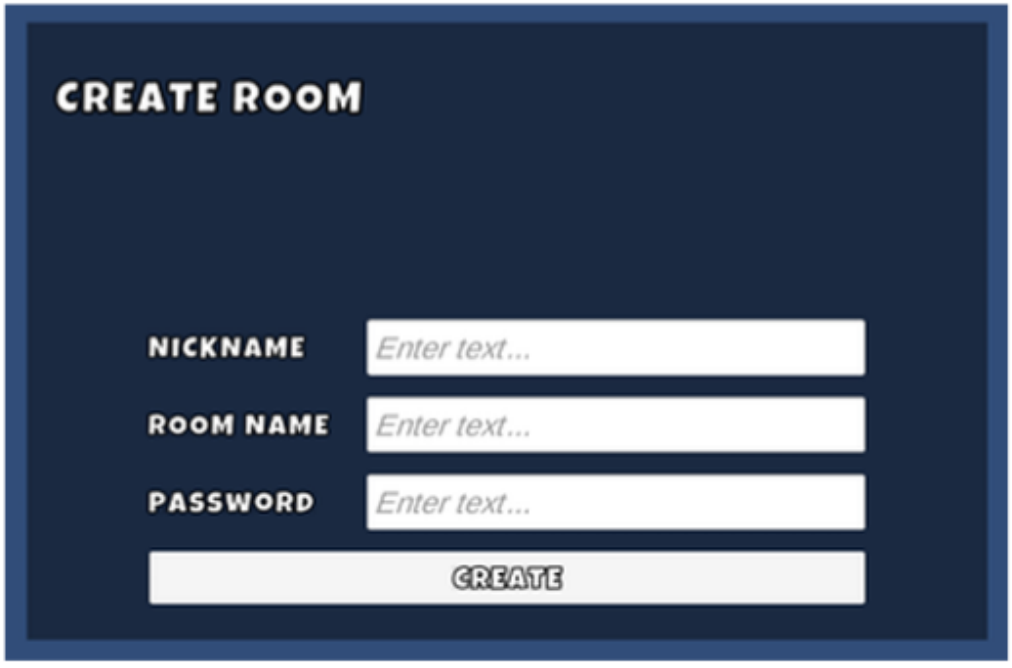
*Proceso de creación de flujo de pantallas: **Main Menu***



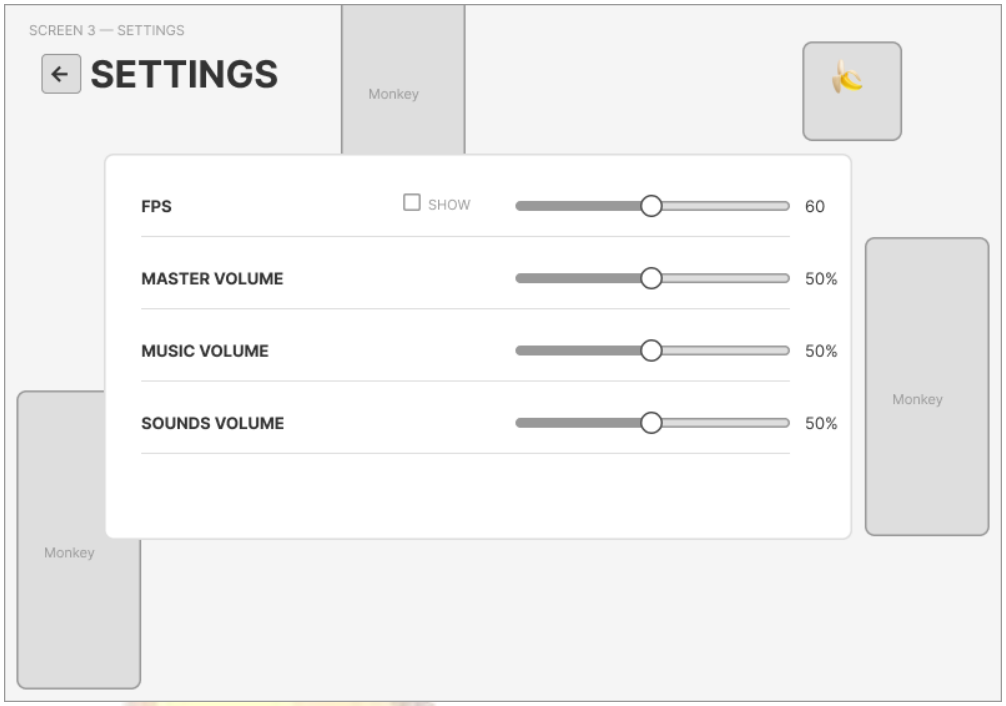

*Proceso de creación de flujo de pantallas: **Create Room***




*Posible **Main Menu***




*Posible **Create Room***




*Proceso de creación de flujo de pantallas: **Settings***

*Posible menú **Settings***

### *Pantallas finales (arte definitivo)*

*Versión final de arte para las pantallas clave del juego, ya integradas al estilo visual definido en la Parte 3.*

***Main Menu — versión final***


*Pantalla principal: logo Bananarang, ambientación de jungla con monos animados.*

***Loading Screen***

*Patrón decorativo de gorilas usado como fondo de carga entre escenas.*

***How to Play***


*Pantalla de instrucciones: controles (teclado/joystick), tipos de banana y sistema de puntos.*

### *Pantallas in-game*

*Capturas del juego en ejecución mostrando las pantallas principales del flujo de partida.*

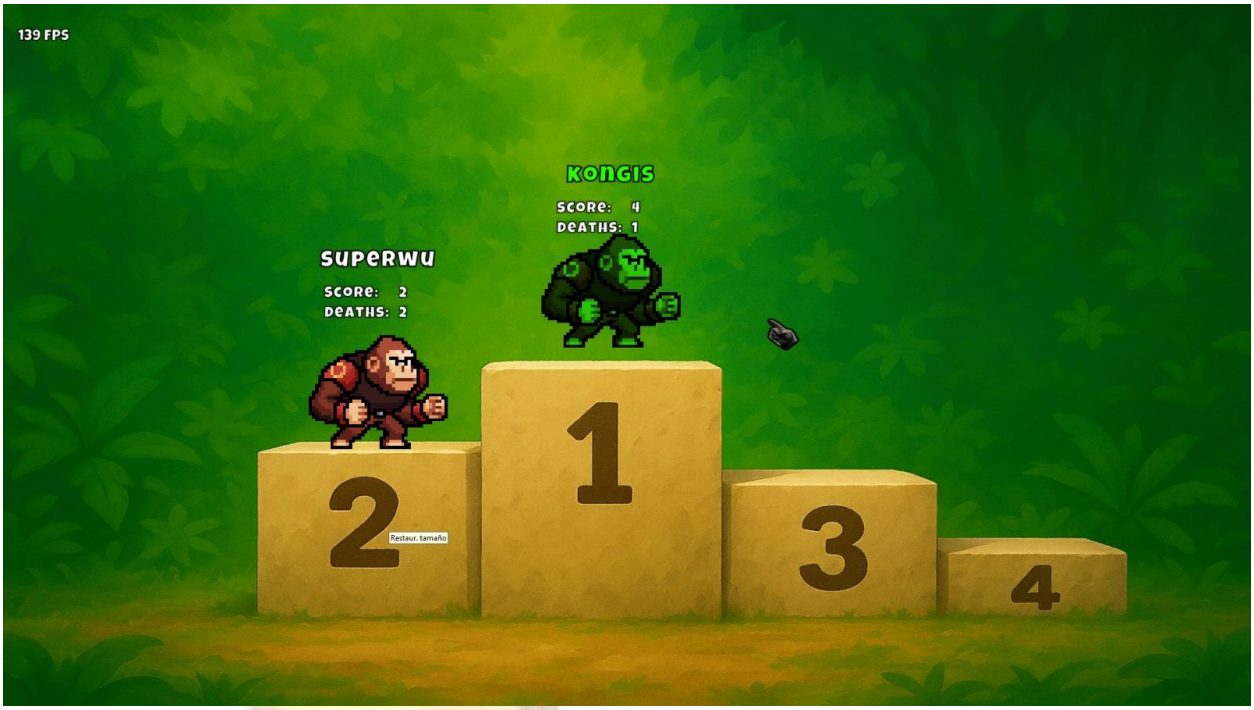
***Points — Scoreboard***


*Scoreboard in-game: muestra ronda actual, nombre del mapa, y estadísticas (Score / Deaths) de cada jugador.*

***Waiting Room***


*Sala de espera (lobby): los jugadores eligen su gorila y marcan Ready antes de iniciar la partida.*

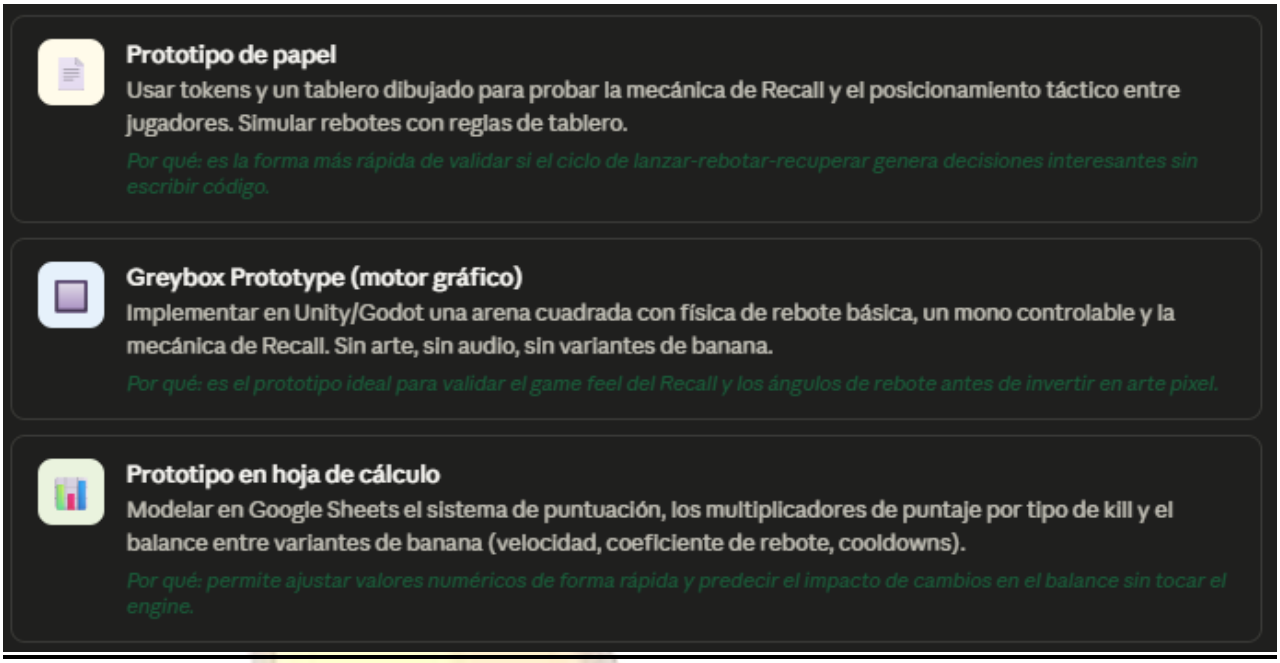
***Podiums***




*Pantalla de podio: ranking final ordenado por Score y Deaths al completar todas las rondas.*

# PARTE 4 — PRODUCCIÓN E INVESTIGACIÓN

## Estrategia de prototipado




### *Metodología de prototipado*

- ***Fase 1***

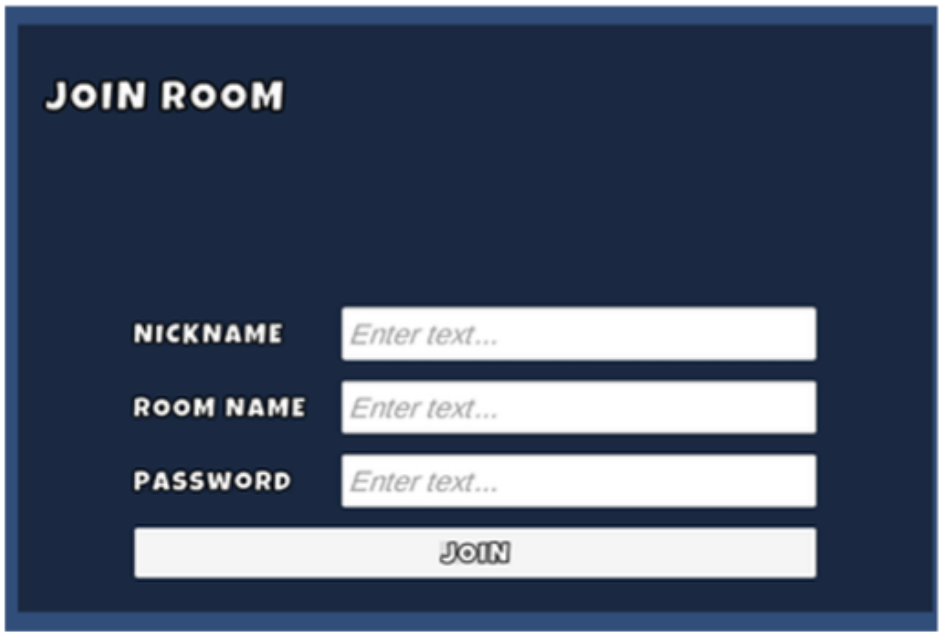
Rapid Prototyping en papel: múltiples variaciones del core loop (con y sin Recall automático, con y sin persistencia de banana) probadas en sesiones cortas de 20 minutos para descartar rápidamente las que no generan decisiones interesantes.

- ***Fase 2***

Iterative Prototyping en Greybox: construir la mecánica base, probarla, aprender y mejorar. Ciclo de 1 semana por iteración enfocada en una variable a la vez (física de rebote, balance de velocidades, feel del Recall).

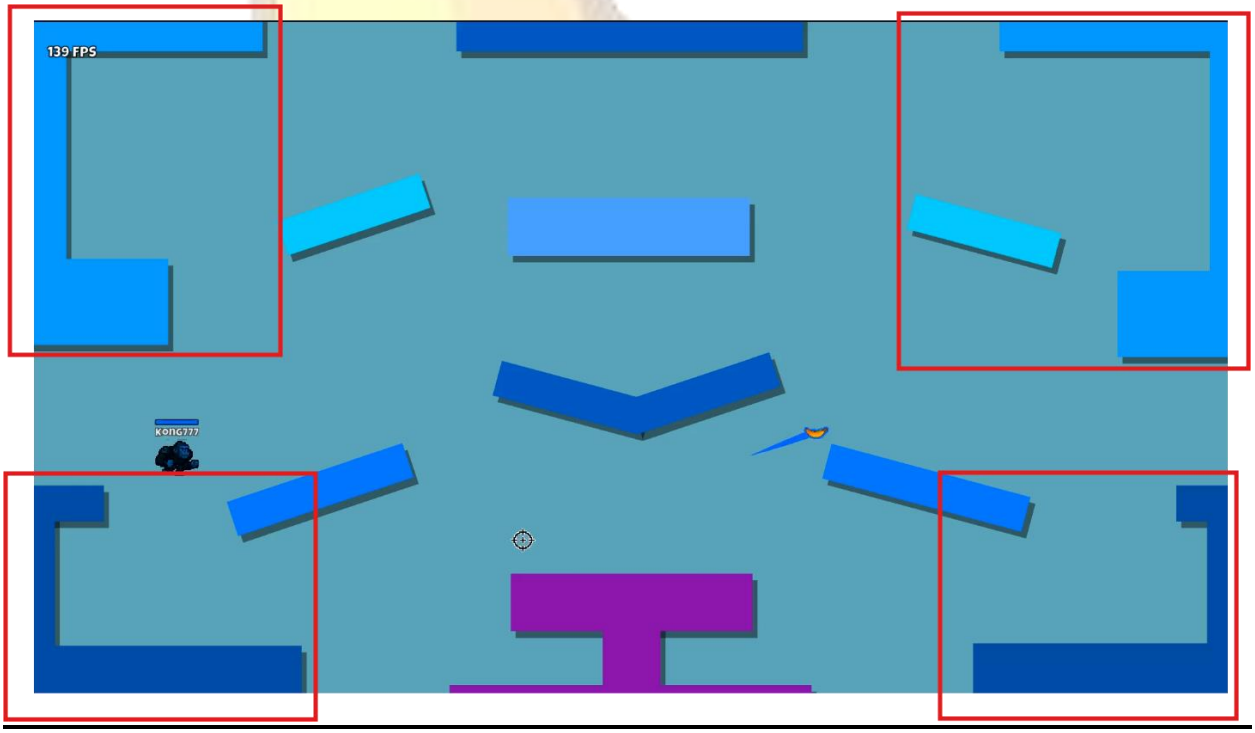
- ***Fase 3***

Parallel Prototyping de variantes de banana: desarrollar en paralelo prototipos de la Banana Explosiva y la Banana Dividida para comparar cuál agrega más valor táctico al core loop antes de implementar ambas.




*Posible menú **Join Room***

## ***PROTOTIPADO***




*Módulos base para prototipado.*

# PARTE 5 — ARQUITECTURA DE SISTEMAS Y FLUJO DE JUEGO

## Arquitectura de sistemas y flujo de juego

### *11.1 — Flujo de escenas*

***Secuencia general***

*Bootstrap → Main Menu → Room (lobby) → Level 1–3 (rotación aleatoria) → Podium.*

*El sistema cumple con la regla de ser extensible: agregar un Level 4 solo requiere sumarlo a la lista de niveles disponibles, sin tocar la lógica de rotación.*

***Detalle por escena***

- ***Bootstrap**: Se carga de forma aditiva al inicio para instanciar todos los singletons (managers de red, escena, input, audio) y luego se descarga.*
- ***Main Menu**: El jugador puede crear una room, unirse a una existente, ver cómo jugar, ajustes y créditos.*
- ***Room (lobby)**: Hasta 4 slots. Cada jugador elige skin sin repetir color y marca "Ready". Al estar todos listos, el host inicia un countdown y se carga un nivel aleatorio. La room se cierra al iniciar (no entra nadie nuevo).*
- ***Level**: Ronda real de combate. Al quedar 1 jugador vivo, zoom-in de cámara durante unos segundos y carga del siguiente nivel aleatorio.*
- ***Podium**: Se alcanza al llegar a totalRounds. Ranking final ordenado por Score desc, Deaths asc.*

***Sistema responsable: PlayersManager***

*Mantiene la lista currentLevels y va sacando un nivel al azar; cuando se vacía, la reinicia. El nivel actual queda excluido del próximo sorteo, garantizando que nunca se repita el mismo nivel dos veces seguidas.*

### *11.2 — Capa de red (Photon)*

***PhotonNetworkManager***

*Singleton que envuelve la conexión y los callbacks de Photon (OnJoinedRoom, OnPlayerEnteredRoom, etc.), reexponiéndolos como eventos de C# para que el resto del juego no dependa directamente de PUN.*

***Parámetros de red***

- ***Jugadores máximos**: 4*
- ***Validación de nick / room / pass**: 3–12 caracteres*
- ***Send rate**: 60*
- ***Serialization rate**: 30*

***Identificador de room***

*roomName + "_" + password — no hay matchmaking público, solo rooms privadas por nombre y clave.*

***Sincronización de escena***

*Con AutomaticallySyncScene = true, cuando el host ejecuta LoadLevel todos los clientes lo siguen automáticamente. El cambio de escena siempre se hace vía PhotonNetwork.LoadLevel, nunca SceneManager.LoadScene, para mantener todo sincronizado en red.*

***Custom properties del Player***

- ***SkinIndex**: color elegido en la sala*
- ***IsReady**: estado del checkbox del lobby*
- ***Score**: estadística de partida*
- ***Deaths**: estadística de partida*

### *11.3 — Flujo de jugador*

***Spawn***

*PlayerSpawnerManager toma el orden de PhotonNetwork.PlayerList y asigna a cada jugador uno de los spawnPositions hijos (módulo cantidad). Cada cliente instancia su propio Player con PhotonNetwork.Instantiate. El PlayerView recibe un SpawnIndex para orientarse.*

***Skins***

*PlayerSkinManager asigna un color aleatorio no ocupado al entrar a la sala. En el lobby puede cambiarse manualmente con flechas, solo entre los disponibles. Se persiste en SkinIndex y se aplica al sprite, la barra de vida y el outline + trail del bumerán.*

***Inputs***

*PlayerInputsManager centraliza el Input System con las acciones: Move, Jump, Attack, Interact, BackUI, TabUI, Settings.*

*HybridCursorManager gestiona un cursor virtual sobre canvas compatible con mouse y joystick, con dos sprites: UI pointer (menús) y battle pointer (in-game). Es la fuente de posición que usa el ataque para apuntar.*

***Control de update centralizado***

*UpdateManager expone OnUpdate y OnFixedUpdate estáticos. Casi todos los scripts de gameplay (Player, Boomerang, PowerUpSpawner, PlayersManager) se suscriben ahí en lugar de tener su propio Update.*

*Estos eventos no se invocan durante panel de loading, panel de salir, o countdown de ronda, pausando toda la lógica de gameplay sin tocar Time.timeScale (la red sigue activa).*

***Movimiento, salto y vida***

- *Movimiento horizontal sobre Rigidbody2D (velocity.x).*
- *Salto: impulso vertical si está grounded (BoxCast contra layers Floor y Player), lo que permite pisar cabezas como piso.*
- *Sprite invertido por localScale.x; nickname y barra de vida se contra-invierten para verse siempre derechos.*
- *Daño vía RPC GetDamage(damage, attackerActorNr, boomerangType), con efecto blink al recibirlo.*

***Secuencia de muerte***

1. ***Instancia Skull + blood** — calavera con física que rebota y emite partículas; sistema de partículas de sangre.*
2. ***+1 Deaths** — al propio jugador, en CustomProperties.*
3. ***+1 Score** — al atacante, en CustomProperties.*
4. ***Notificación UI** — "X killed Y".*
5. ***Destrucción en red** — vía PhotonNetwork.Destroy.*

***Condición de victoria de ronda***

*En cada Update del PlayerModel: si PlayersManager.CurrentPlayers.Count < 2 y el jugador sigue vivo (hasWonTheRound == false), gana la ronda: suma 1 a su Score, congela su rigidbody y dispara OnPlayerWinCurrentRound (panel "WIN"). El jugador muerto recibe el panel "LOSE".*

### *11.4 — Sistema de combate: bumerán*

***Sistema central del juego***

*Existe siempre un bumerán por jugador, que se cambia al recoger un power-up. Es el sistema más intrínseco al diseño: sin la mecánica de lanzar/recuperar, Bananarang perdería su identidad.*

***Variantes de bumerán***

- ***Default**: el inicial. Una vez lanzado, queda clavado donde colisiona o pegado a un jugador al impactarlo. Hay que esperar a que se detenga para poder llamarlo de vuelta.*
- ***Fast**: igual al Default pero con mayor velocidad.*
- ***Damageable**: daña un poco más mientras esté tocando al jugador (no solo en el frame de impacto), con cooldown por jugador (damageCooldown). Mantiene la regla de solo poder traerlo si está quieto.*
- ***Returnable**: único que se puede llamar de vuelta en cualquier momento, incluso volando, marcando el collider como trigger durante el retorno.*

***Ciclo de estados***

1. ***En mano** — parent del jugador, rigidbody desactivado, collider apagado.*
2. ***Lanzado (ThrowBoomerang)** — collider y rigidbody activos, dirección normalizada hacia el cursor del lanzador. Rotación visual aleatoria CW/CCW. Ignora colisión con el propio jugador.*
3. ***Pegado a escenario** — al chocar contra algo que no sea Player, queda estático.*
4. ***Pegado a víctima** — al impactar a un Player enemigo, queda parented al enemigo y aplica daño. Si pasa timeToGetBoomerangBackIfIsCollidingWithSomePlayer o la víctima muere, vuelve automáticamente.*
5. ***Retornando (ReturnBoomerang)** — se mueve hacia BoomerangHandPosition del dueño cada FixedUpdate. Vuelve a habilitar colisión con el dueño.*
6. ***Recolección** — al entrar al collider del dueño, vuelve a ser hijo del jugador y queda listo para lanzarse otra vez.*

***Reglas de combate***

- ***No friendly fire**: el bumerán propio no daña al dueño (colisión con sí mismo descartada).*
- ***Bumerangs entre sí**: si dos colisionan y el atacante no está retornando, fuerza el retorno del propio para evitar locks.*
- ***Detección de daño**: siempre desde el dueño (photonView.IsMine), propagada con RPCs (GetDamage al objetivo, OnBoomerangCollisionEnter a todos).*
- ***Apuntado**: dirección calculada con ScreenToWorldPoint(cursorPos) desde la posición de la mano: apuntado libre, no atado al eje del jugador.*

***Si el atacante muere antes de que vuelva su bumerán***

*El bumerán "stuck" en una víctima retorna automáticamente.*

### *11.5 — Power-ups*

***PowerUpBoomerangSpawner***

*Corre solo en el host (PhotonNetwork.IsMasterClient). Tiene una lista de spawnPositions hijos y carga todos los prefabs de Resources/Prefabs/PowerUpBoomerangs/.*

***Lógica de spawn***

- *Cada timeToSpawnNewBoomerang segundos elige posición y prefab al azar, instanciándolos como RoomObject (no muere si el host cambia).*
- ***Un power-up por vez**: la variable hasSpawnPowerUp bloquea el spawn hasta que se recoja.*
- *No spawnea cuando queda un solo jugador (la ronda está por terminar).*
- *Visualmente flotan con efecto seno (idle).*

***Recolección***

*El primer jugador que toca el trigger lo agarra: se le instancia el bumerán correspondiente y se destruye el actual (vía PhotonNetwork).*

***Prefabs existentes***

- *PowerUpBoomerangFast*
- *PowerUpBoomerangReturnable*
- *PowerUpBoomerangDamagable*

### *11.6 — Rondas, niveles y puntaje*

***PlayersManager***

*Mantiene currentRound, totalRounds, la lista de jugadores vivos (CurrentPlayers) y la lista de niveles disponibles.*

***Fin de ronda***

*Cuando currentPlayers.Count <= 1 tras una muerte: zoom-in de cámara al sobreviviente durante cameraEffectDuringTime, espera 1.5s, y:*

- ***Si currentRound >= totalRounds**: carga Podium y dispara RoundEndedEvent.*
- ***Si no**: carga otro nivel aleatorio sin repetir.*

*Solo el host decide el cambio de escena; el resto sigue gracias a AutomaticallySyncScene.*

***StatsManager***

*Actualiza Score (kills + 1 por ganar la ronda) y Deaths (al morir) en CustomProperties. Se persisten entre escenas porque viven en el Player de Photon, y se resetean al abandonar la room.*

***TimeManager***

*Al cargar una escena Level, dispara un countdown ("3, 2, 1, GO!"). Hasta que termina, UpdateManager no propaga sus eventos: los jugadores quedan congelados y no pueden lanzar bumerangs.*

***Otros sistemas relevantes***

- ***Teleport2D**: trigger que teletransporta Player y Boomerang vía RPC Teleport(pos). Único BoomerangHandPosition por jugador, también soportado. Útil como portales o wrap-around de pantalla.*
- ***Skull**: calavera al morir, con Rigidbody2D que rebota y emite partículas. Cosmético, pero instanciado en red.*

***Resumen del loop principal***

1. ***Conectarse al servidor** — Main Menu.*
2. ***Crear o unirse a una room privada** — nombre + contraseña.*
3. ***En el lobby** — todos eligen skin y marcan Ready.*
4. ***Host hace countdown** — carga nivel aleatorio, room cerrada.*
5. ***Cada cliente spawnea** — su Player con bumerán Default.*
6. ***Combate** — bumerangs apuntados al cursor, con power-ups que cambian el tipo. El último vivo gana la ronda y suma 1 al Score.*
7. ***Zoom al ganador** — fade, y carga de otro nivel para la siguiente ronda.*
8. ***Al llegar a totalRounds** — Podium ordenado por Score / Deaths.*

## Análisis de mercado y referentes

### *Boomerang Fu*

***Inspiración***: combate caótico con proyectiles de trayectoria curva y eliminación de un solo golpe.

***Diferenciación***: En *Boomerang Fu* los proyectiles desaparecen al fallar. En Bananarang, la persistencia física del plátano transforma el mapa en una red táctica de control de zonas activa durante toda la ronda.

### *TowerFall Ascension*

***Inspiración***: precisión matemática extrema, gestión del espacio vertical/horizontal, tensión por quedarse sin munición.

***Diferenciación***: *TowerFall* obliga al jugador a exponerse para recoger flechas del suelo. En Bananarang, el Recall permite recuperar la munición de forma remota e interactiva, pudiendo matar enemigos en el trayecto de retorno. Esto incentiva un juego mucho más agresivo.

### *Oportunidad de mercado*

El segmento de party games competitivos de arena para PC y consola ha crecido significativamente con el auge de los juegos de sesión corta y el gaming local. Bananarang ocupa un nicho no cubierto: la gestión táctica del proyectil persistente en un formato accesible y caótico.


*Tope de gama en el género.*

*Referencia visual de TowerFall Ascension*

*Referencia visual de TowerFall Ascension*

*Referencia visual Boomerang Fu*

*Referencia visual Boomerang Fu*

## *OTRAS REFERENCIAS Y COMPETENCIAS DE JUEGOS DEL ESTILO*



## BIBLIOGRAFÍA:

***https://towerfall.wiki.gg/***

***https://www.youtube.com/watch?v=4mnJlzJTTsc***

***https://boomerangfu.fandom.com/wiki/Boomerang_Fu***

***https://glossary.infil.net/?t=Arena%20Fighter***

***Continuará…***


***GRACIAS POR LEER!***