Eternal rift

1. Descripción general

El juego es un plataformas 2D con mecánicas roguelike, donde el jugador controla a Crow Orion, un superviviente atrapado en un bucle temporal tras una catástrofe global causada por un meteorito alienígena. Cada vez que el jugador muere, su progreso se reinicia, pero obtiene mejoras genéticas (velocidad de movimiento, más vida, entre otros) y objetos más poderosos en cada *run*, lo que permite una sensación de mejora y avance. La vista del juego es lateral, tipo side-scroller, y el enfoque está puesto en el movimiento ágil y preciso.

2. Pilares de diseño y género

El género del videojuego será un plataformas 2D con elementos roguelike. Con eso en mente, se han definido varios pilares de diseño clave para garantizar una experiencia de juego entretenida y con una progresión desafiante.

- **Progresión roguelike:** Cada run es diferente, con niveles generados de forma aleatoria y mejoras genéticas distintas que alteran las habilidades del protagonista.
- **Movimiento ágil y combate dinámico:** El jugador deberá el disparo a enemigos, dominar el salto,el dash, y otras mecánicas de movimiento tanto para superar obstáculos como para enfrentar enemigos.
- **Exploración y descubrimiento narrativo:** El jugador irá revelando la historia a medida que progresa y muere, descubriendo nuevos elementos tras cada reinicio.
- **Desafío progresivo:** A medida que avanza el juego, la dificultad será alta, pero el jugador se vuelve más fuerte con cada muerte, proporcionando una curva lineal en cuanto dificultad del juego en función del tiempo jugado.

Con respecto a los jugadores objetivo, apuntamos a un público mayor de edad, que esté en búsqueda de un desafío y que disfrute de juegos de plataformas, al cual se le agrega una progresión roguelike. Será un juego ideal para aquellos que valoran agilidad en el movimiento, la necesidad de aprender de sus errores y superar obstáculos a través de la mejora continua y el dominio de las mecánicas del juego. El enfoque en una narrativa complega, que se irá

desarrollando con el progreso de las *runs*, y un sistema de mutaciones aleatorias también atraerá a jugadores interesados en la exploración y la rejugabilidad.

3. Narrativa

La historia se desarrolla en un mundo devastado tras la caída de un gigantesco meteorito, que no solo trajo destrucción, sino fragmentos alienígenas con energía ilimitada. Dos facciones rivales, una del norte y otra del sur, se sumergen en una feroz competencia por el control de estos fragmentos. Luego de una época de violentos enfrentamientos y una destrucción incesante, el norte salió victorioso, aprovechando el poder de los fragmentos alienígenas para desarrollar tecnología avanzada capaz de generar energía ilimitada.

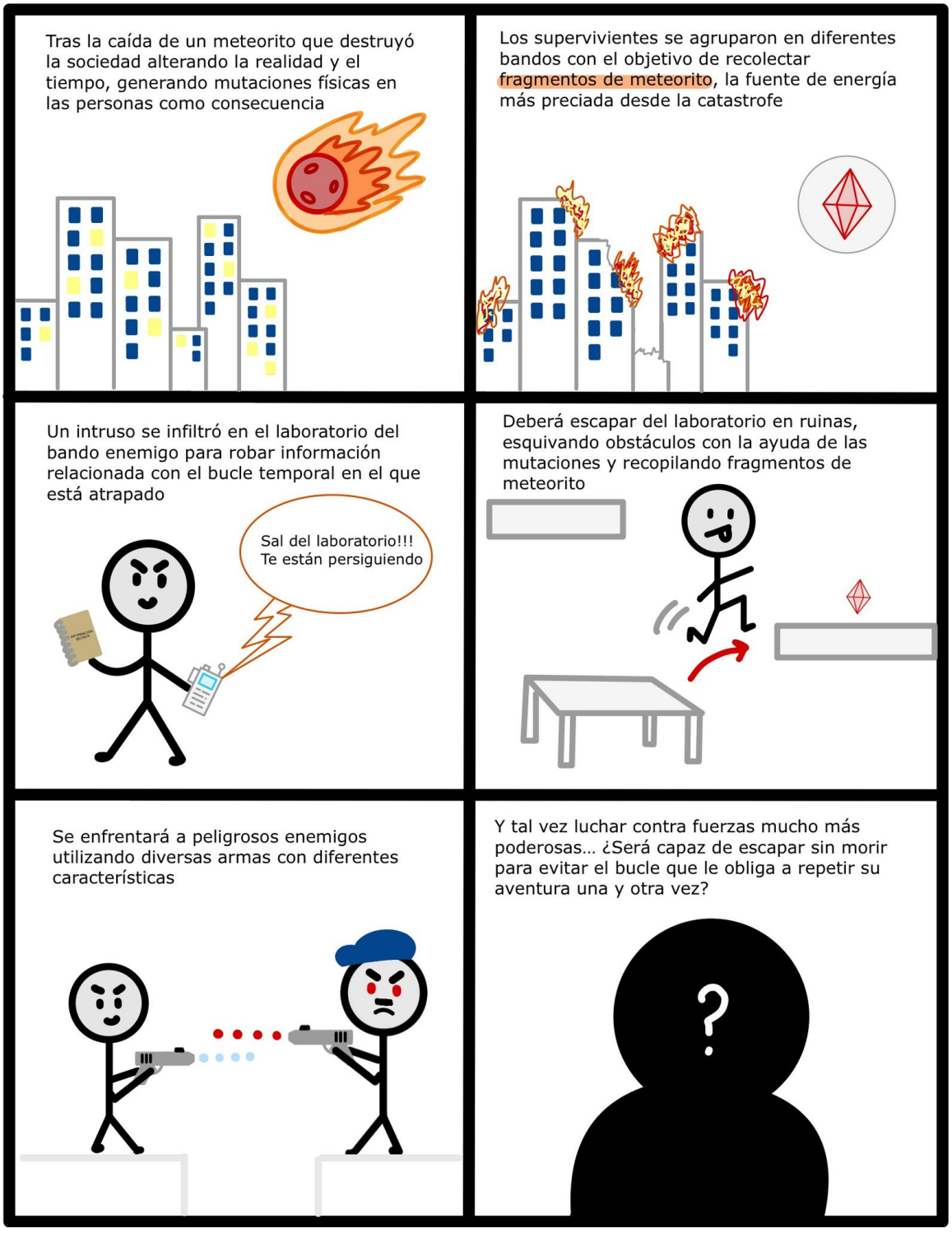
Durante un sabotaje, la energía del meteorito es desatada, atrapando a los supervivientes en un bucle temporal que los obliga a revivir sus muertes una y otra vez.

Crow Orion, el protagonista, es uno de los afectados por este fenómeno. Con la ayuda de la Dra. Evee Kross, que actúa como su guía en las primeras fases del juego, deberá navegar un mundo roto y mutante, enfrentándose a las fuerzas de las facciones rivales y a monstruos mutados por la energía alienígena.

Cada vez que muere, Crow regresa al inicio de su búsqueda, pero con nuevas mutaciones genéticas que alteran su habilidad para moverse y combatir.

3.1 Storyboard

A continuación, se muestra el storyboard para el videojuego:



4. Mecánicas

4.1. Movimiento:

- **Salto:** El jugador puede moverse verticalmente y evitar obstáculos o atacar desde el aire.

- **Dash:** Habilidad clave para moverse rápidamente a través de los niveles o esquivar ataques.
- **Mutaciones genéticas:** Cada vez que el jugador muere, adquiere una mutación aleatoria que puede alterar la jugabilidad. Ejemplos:
    - **Gravedad reducida:** Permite saltos más altos y caídas más lentas.
    - **Paso ligero:** Aumenta la velocidad de movimiento.

4.2. Combate:

- **Armas y objetos:** El jugador podrá obtener y mejorar armas de larga sitancia y habilidades a lo largo de cada run.
- **Ataques combinados con movimiento:** Las mecánicas de combate se integran con el movimiento ágil, permitiendo al jugador atacar mientras se mueve rápidamente por el nivel.

4.3. Progresión roguelike:

- **Niveles aleatorios:** Las salas se generan de forma procedural, cambiando el diseño y la disposición de enemigos y objetos en cada run.
- **Mejoras permanentes:** A medida que el jugador avanza, obtendrá mejoras que se mantendrán tras cada run, proporcionando una sensación de progreso continuo.

5. Aspectos técnicos

En cuanto al código, nos hemos enfocado en tres aspectos clave del videojuego para esta primera iteración: movimiento, disparo y object pooling.

5.1 Game Object Player

El game object *Player* utiliza componentes clave que permiten el control de sus movimientos, fundamentalmente a través de un Rigidbody2D y un Collider2D (para esta iteración lo hemos hecho cápsula, aunque en las siguientes representará el sprite del personaje). El *Player* es afectado por las físicas del mundo 2D, utilizando la gravedad para los saltos y movimientos verticales. Por otro lado, el collider permite la detección de colisiones para el personaje, por ejemplo con paredes, optimizando la interacción con el entorno.

El comportamiento de su movimiento está dividido en dos scripts:

1. Player_Move: gestiona el movimiento horizontal del jugador. El jugador controla la dirección mediante las teclas *A* y *S*.
2. Jump: se encarga del movimiento vertical del jugador. Esta mecánica le otorga al jugador maoyr flexibilidad en su movimiento ya que, si se suelta la tecla de salto (en este caso, el espacio) antes de alcanzar el pico de este, el *Player* detiene su ascenso, permitiendo un mayor control sobre la altura del salto.

Este sistema asegura que el movimiento del *Player* sea ágil, preciso y esté basado en la física, proporcionando una jugabilidad fluida y responsive para el jugador.

5.2. Object pooling

En el juego utilizamos un sistema de **Object Pooling** para mejorar el rendimiento, especialmente en la gestión de balas y enemigos. Este método permite reutilizar instancias de objetos en lugar de crear y destruir nuevos constantemente, lo que reduce significativamente la carga de procesamiento y mejora la fluidez del juego.
El sistema está diseñado utilizando el patrón Singleton, lo que garantiza que haya una única instancia centralizada de las pools. Estas pools, estructuradas como colas FIFO, aseguran que los objetos se reutilicen de manera ordenada, evitando conflictos al solicitar múltiples instancias simultáneamente. El uso del Singleton facilita el acceso global a estas colas desde cualquier otro script del juego, asegurando un manejo eficiente y controlado de los recursos compartidos.

6. Estilo de arte

El juego contará con un estilo pixel art. El mundo devastado y fragmentado tendrá un tono sombrío pero con colores vibrantes para resaltar la energía alienígena. Los escenarios mostrarán ruinas de civilización mezcladas con elementos tecnológicos avanzados y vegetación mutante, proporcionando un contraste visual entre lo natural y lo sobrenatural.

Algunos de los enemigos tendrán un diseño estilizado, con mutaciones genéticas visibles en sus cuerpos, lo que subrayará la influencia de los fragmentos alienígenas sobre el mundo y sus habitantes.
A continuación, presentamos a nuestro protagonista, Crow Orion. Queremos que su diseño refleje su fortaleza, la cual ha desarrollado para resistir el caos que lo rodea.

7. Referencias a juegos previos

Para guiarnos en el desarrollo, vamos a fijarnos en juegos como *Rayman Legends*, *Celeste* o *Dead Cells*.

De *Rayman Legends,* podemos extraer ideas para el movimiento y cómo el diseño de niveles se adapta al mismo. En este juego, la lucha que hay dentro de los niveles está completamente centrada en el plataformeo, de manera que los movimientos de combate también son útiles para su movilidad.

De *Celeste,* utilizaremos ideas para el diseño de niveles y el arte del juego, debido a que nuestro juego utilizará un estilo pixel art.

Por último, *Dead Cells* es el estilo que estamos buscando. Implementa tanto las plataformas como la progresión *roguelike* que queremos para nuestro juego. La gran diferencia con los juegos anteriores, es que *Dead Cells* está mucho más centrado en la lucha con movimientos como rodar para esquivar y caer fuerte para dañar. Del aspecto *rogue,* nos fijamos en la manera en la que se justifica el bucle jugable (reinicio de la partida al morir) y en entregar diferentes mejoras y armas al jugador en cada *run*.

8. Organización del equipo

Para esta iteración, los roles de equipo se han distribuido de la siguiente manera:

- **Team Leader**: Victor de los Santos Llana
- **Programmers**: Ignacio Boigues Mexia, Gabriel Peña Sanchez, Daniel Francisco Lanzas Larrañeta
- **Quality Assestment**: María Ruiz Castro, Ian Bernasconi

Por último, el link al repositorio github es el siguiente:

https://github.com/dflanzs/proyectoFdV