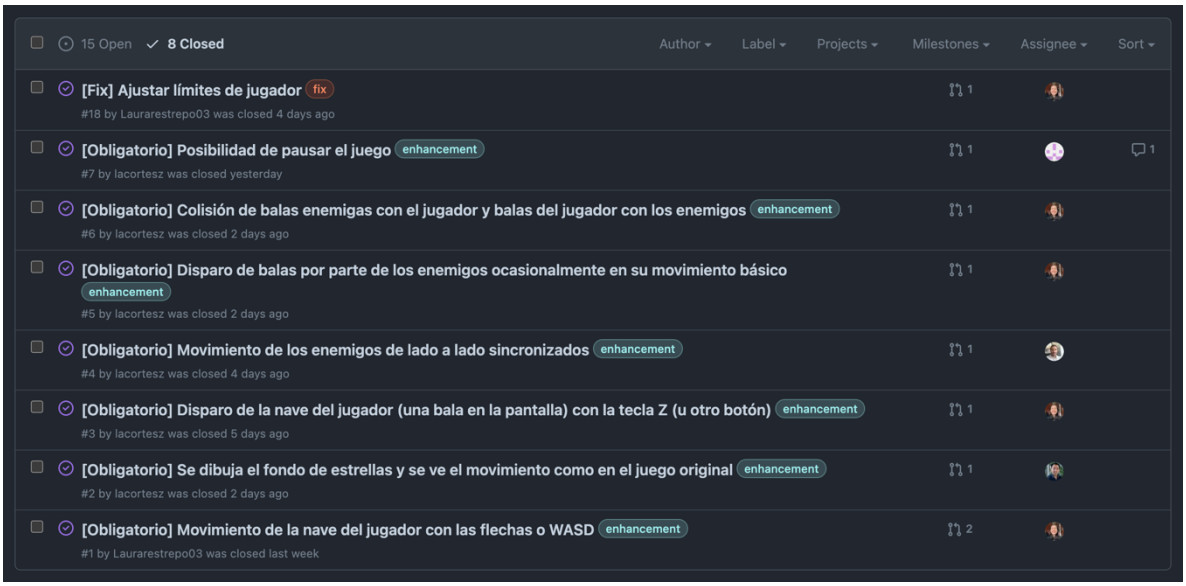
Documento avance semana 5

El enlace a nuestro repositorio es el siguiente:
https://github.com/Laurarestrepo03/Ocarina-Galaxian

Avance

Issues terminados

Repartimos las tareas a trabajar en issues según la rúbrica (ver /issues). De momento, estos son los que llevamos terminados:



Notas:

- Por ahora, no eliminamos la entidad del jugador cuando colisiona con una bala porque no hemos implementado niveles o vidas.
- El disparo de balas por parte de los enemigos es inmediato. Queremos generar un delay para que no sea abrumadora la cantidad de balas, pero estamos viendo cómo. Más detalles al final.

Componentes creados

Los componentes que hemos creado por ahora:

- CTagBullet: Para diferenciar las balas, más que todo usado para destruir las entidades cuando chocan con los límites de la pantalla.
- CTagEnemyBullet: Para diferenciar las balas de los enemigos.
- CTagEnemy: Para diferenciar a los enemigos. Tiene atributos que indican el tipo de enemigo (útil para conocer información de su configuración), número de balas disparadas (si un

enemigo tira varias, sirve para tener una cuenta de cuántas lleva), en qué estado de disparo se encuentra, y un temporizador (para causar cierto delay entre las balas disparadas).[1]
- CTagPause: Para diferenciar la entidad de pausa. Tiene un atributo de temporizador que se usa para manejar el parpadeo.
- CTagPlayerBullet: Para diferenciar la bala del jugador. Tiene un atributo que indica si ya se disparó, para controlar que no se dispare otra si la última lanzada sigue en el aire.
- CTagPlayer: Para diferenciar al jugador. Tiene atributos de teclas presionadas, útiles para que no se aumente la velocidad si se presionan las mismas usadas para una dirección (ejemplo: si se presiona la derecha y la tecla D, que solo se cuente como una vez).
- CAnimation: Para gestionar la información relacionada a la animación de una entidad.
- CEnemySpawner: Para gestionar la información correspondiente a la aparición de enemigos.
- CExplosionState: Para gestionar el estado de una explosión.
- CInputCommand: Para gestionar la información de un input, como las llaves que usa (es decir, un comando puede tener más de una llave), su nombre, y el estado en el que se encuentra.
- CStarField: Para gestionar la información relacionada a las estrellas del fondo.
- CSurface: Para gestionar la información de superficie de una entidad.
- CTransform: Para gestionar la información de posición de una entidad.
- CVelocity: Para gestionar la información de velocidad de una entidad.

Sistemas creados

Los sistemas que hemos creado por ahora son:
- system_animation: Para regular la animación.
- system_bullet_limit: Para destruir las entidades de bala (tanto del jugador como enemigos) cuando chocan con los límites de la pantalla.
- system_collision_bullet_enemy: Para destruir al enemigo y la bala cuando una bala del jugador choca con un enemigo. También crea una explosión.
- system_collision_bullet_player: Para destruir al jugador y la bala cuando una bala de algún enemigo choca con el jugador. También crea una explosión. Como se mencionó, por ahora no se destruye al jugador mientras se terminan otras funcionalidades.
- system_draw_stars: Para dibujar las estrellas del fondo.
- system_enemy_bullet_spawn: Para generar balas por parte de los enemigos. Funciona buscando si hay un enemigo disparando, y si sí, cada 0.2 dispara balas (teniendo en cuenta el límite de balas definido para él). Si no hay enemigos disparando, busca uno aleatorio de los que no hayan disparado y lo selecciona como el que va a disparar. Por último, si ya todos los enemigos dispararon, se vuelve a comenzar.
- system_enemy_movement: Para coordinar el movimiento sincronizado de lado a lado de los enemigos.
- system_explosion_state: Para regular el estado de animación de una explosión. Principalmente usado para eliminar la entidad cuando se termina la explosión.
- system_input_player: Para controlar el input de un jugador e invocar la acción correspondiente.
- system_movement: Para controlar la posición de las entidades dada su velocidad.

[1] Con respecto a estos atributos, tenemos una duda que se encuentra más adelante.

- system_pause: Para destruir la entidad de pausa, es decir, se llama cuando se reanuda el juego.
- system_player_bullet_rest_pos: Para controlar la posición de la bala del jugador cuando está en reposo, es decir, cuando no se ha disparado. Básicamente, calcula su posición con respecto a la posición actual del jugador.
- system_player_bullet_spawn: Para crear una nueva bala para el jugador cuando la última se haya destruido.
- system_player_limit: Para controlar que un jugador no puede ir más allá del límite definido, que es un poco menos que la pantalla.
- system_rendering: Para mostrar las entidades en la pantalla. Gracias a él también se puede manejar el parpadeo del texto de pausa.
- system_star_field: Para controlar el movimiento de las estrellas del fondo, así como su parpadeo (que se maneja cambiando el color de la estrella al del fondo).

Funciones prefab

- create_square: Crea cuadrados.
- create_sprite: Crea sprites.
- create_player: Crea al jugador haciendo uso de la función para crear sprites, y le añade la etiqueta de jugador.
- create_bullet: Crea una bala haciendo uso de la función para crear cuadrados y, dependiendo de si la bala le pertenece al jugador o a un enemigo, le añade la etiqueta correspondiente entre bala de jugador o bala de enemigo. En ambos casos de añade la etiquete general de bala.
- create_input_player: Crea entidades para los inputs del jugador a partir de teclas dadas haciendo uso del componente CInputCommand.
- create_star: Crea las entidades de las estrellas del fondo haciendo uso de la función para crear sprites.[2]
- create_enemy: Crea las entidades de los enemigos haciendo uso de la función para crear sprites, y añade la etiqueta de enemigo.
- create_level: Crea los enemigos en el mundo haciendo uso de la función create_enemy.
- create_explosion: Crea una explosión haciendo uso de la función para crear sprites. También reproduce el sonido correspondiente.
- create_pause_text: Crea el texto de pausa y le añade la etiqueta correspondiente a la entidad.

Servicios

Los servicios que hemos creado por ahora son:

- fonts_service: Para las fuentes. Su llave está compuesta de una combinación entre el path y el tamaño.
- images_service: Para las imágenes:
- sounds_service: Para los sonidos.

[2] Con respecto a esto, vamos a hacer un cambio para que use la función de crear cuadrados, porque las estrellas no tienen imágenes.

Archivos de configuración

- enemies: Define la configuración de cada enemigo, incluyendo su imagen, tipo, velocidad mínima, velocidad máxima, animaciones y máxima cantidad de balas seguidas que puede disparar.
- enemy_bullet: Define la configuración para las balas de los enemigos, incluyendo su color, tamaño y velocidad.
- enemy_explosion: Define la configuración para la explosión de un enemigo, incluyendo su sonido, imagen y animaciones.
- interface: Define la configuración de la interfaz, principalmente los textos, incluyendo su fuente, sonido (si lo tiene), color y tamaño.
- level_01: Define la configuración de las líneas de enemigos que aparecen en la pantalla.
- player_bullet: Define la configuración para la bala del jugador, incluyendo su sonido, color, tamaño y velocidad.
- player_explosion: Define la configuración para la explosión del jugador, incluyendo su sonido, imagen y animaciones.
- player: Define la configuración para el jugador, incluyendo su imagen, punto de generación y velocidad.
- starfield: Define la configuración de las estrellas del fondo, incluyendo los colores posibles, la cantidad, velocidad, cada cuanto parpadea, y tamaño.
- window: Define la configuración para la ventana, incluyendo el título, tamaño, color de fondo, y framerate.

Dudas

- La etiqueta de enemigo tiene atributos que indican en qué estado de disparo se encuentra (*NOT_FIRED*, *FIRING*, *FIRED*), cuántas balas ha disparado y un temporizador para controlar cuándo hará el siguiente disparo. ¿Está bien dejar esos atributos en esa etiqueta, o deberíamos crear un componente aparte?
- Queremos hacer un delay entre los disparos de los enemigos, es decir, cuando un enemigo termine de disparar, que no necesariamente otro dispare de inmediato. ¿Podemos crear un componente parecido a CEnemySpawner para regular los tiempos entre los disparos, o qué nos sugieren?
- En general, ¿hay algo que estemos haciendo mal o que recomiendan cambiar?

¡Gracias!