Taller de Programación I (75.42)
Facultad de Ingeniería
Universidad de Buenos Aires

Jazz Jackrabbit 2

Índice

Introducción

En esta oportunidad se hará una remake multiplayer del juego Jazz Jackrabbit 2 de Epic MegaGames, un shooter de plataformas 2D de fines de los 90s.

Descripción

El jugador puede seleccionar uno de los 3 personajes:

 Jazz

 Spaz

 Lori

Todos los personajes comparten las mismas habilidades (saltar, disparar, correr) pero cada personaje

tiene un ataque especial.

Acciones comunes

 Disparar: tanto estando quieto como cuando corre, el jugador puede disparar con el tipo de munición seleccionada.

 Correr: se desplaza por el escenario.

 Correr muy rapido: igual que la accion de correr, pero mas rapido.

 Saltar: un salto tanto hacia arriba como hacia adelante.

Estados

 Intoxicado: al comer una fruta envenenada el personaje queda por cierto tiempo intoxicado. Puede moverse pero no puede disparar.

 Recibir daño: cuando recibe un impacto.

  Muerte: cuando se acaba la vida. El personaje revive tras unos segundos en algún lado del escenario.

Todos los personajes tienen una cantidad de vida: cuando reciben daño esta se reduce, cuando comen una zanahoria (que obtienen del escenario), esta aumenta.

Acciones especiales

 Puñetazo hacia arriba: Jazz puede hacer un salto vertical, sin posibilidad de moverse lateralmente, pero realizando un daño con todo lo que toque.

 Patada de corto alcance: Lori puede hacer un patada voladora de corto alcance mientras da un salto (tal como si fuese una accion de salto), realizando un daño con todo lo que toque.

 Patada hacia un costado: Spazz puede hacer hacer una patada que la desplaza de forma lateral, sin poder saltar hacia arriba, realizando un daño con todo lo que toque.

Armas/Municiones

En el juego, todas las armas son funcionalmente similares. La diferencia entre una y otra está en:

- Munición: todas las armas tienen munición finita salvo el arma inicial que tiene munición infinita.
- Velocidad de disparo: el tiempo entre que se realiza uno y se puede realizar el siguiente
- Velocidad del proyectil: las balas/misiles se mueven a cierta velocidad.
- Daño: cada proyectil causa un daño distinto.

Todos los jugadores deberán tener siempre a disposición su “arma inicial” e irán obteniendo nuevas al ir juntando municiones de ellas.

Algunos ejemplos de las municiones y proyectiles (hay más en los sprite sheets):

Gemas y monedas

 Dan puntos al jugador que las obtenga.

Enemigos

En el escenario habrá ciertos enemigos que los jugadores podrán eliminar para obtener municiones, vidas o puntos. Algunos enemigos solo caminan, otros pueden volar. Todos hacen daño al tocar a un jugador.
Los enemigos muertos luego de un tiempo vuelven a aparecer.

Se deben implementar al menos 3 enemigos. Cuanta vida tiene cada uno, qué daño producen, cuantos puntos dan, cada cuanto tiempo vuelven a aparecer y con qué probabilidad dejan caer una munición o una vida queda a elección de los desarrolladores (que deberá ser configurable vía un archivo)

Algunos ejemplos:

Escenarios

Hay múltiples estilos de escenario que se pueden elegir. Todos ellos son funcionalmente iguales con pisos y paredes sólidas que los personajes no pueden traspasar. El terreno tiene tanto plataformas horizontales como diagonales.
Se deben implementar al menos 2 estilos de escenarios (por ejemplo el castillo y la playa) y algunos escenarios/mapas para cada uno para que sirvan para la demo.

Multijugador

Cada jugador deberá poder o crear una partida para N jugadores o deberá poder unirse a una partida ya creada.

Dentro de una partida los jugadores compiten entre sí para ver quien puede recolectar más puntos sea matando enemigos, recolectando gemas y monedas o matando a otros jugadores.
Las partidas terminan por tiempo y al finalizar se debe mostrar la tabla de resultados con los jugadores y su puntaje ordenado por puntaje.

Cámara

La cámara muestra una porción del escenario (los escenarios pueden ser muy largos y no entrar en la vista de la cámara) y debe enfocarse en el personaje activo y seguirlo a medida que se desplaza.

Animaciones

El juego no debe mostrar imágenes estáticas sino pequeñas animaciones para darle mayor realismo:

- Los personajes tienen una animacion para correr, saltar, disparar…
- El movimiento de los proyectiles
- Las explosiones.

Sonidos

Como todo juego se debe reproducir sonidos para darle realismo a los eventos y acciones que suceden:

- Cuando hay disparos.
- Cuando hay una explosión.

Si la cantidad de eventos que suceden es muy grande, algunos sonidos pueden ser evitados para no saturar al jugador con tanta información.

Musica ambiente

El juego debe reproducir una música ambiente, con un volumen relativamente bajo.

Interfaz del jugador

Debe mostrarse el puntaje, vida, municiones, arma seleccionada del jugador, así como el tiempo restante de la partida y un top 3 con los puntajes más altos hasta el momento.

El jugador deberá poder seleccionar qué arma usar.

Al finalizar el escenario se deberá mostrar la tabla de scoring.

Configuración

Todos los atributos en este enunciado son a modo de ejemplo y deben poderse cambiar vía un archivo de configuración YAML.
Por ejemplo se debe poder cambiar el daño de un arma; el radio de una explosion; etc.

La idea es que no haya valores hardcodeados en el código: el archivo de configuración te permitirá ir

tuneando el juego para hacerlo más fácil o difícil y le permitirá al docente probar rápidamente una funcionalidad sin tener que jugar mucho tiempo (por ejemplo al setear la vida de los enemigos en 1 es trivial probar su muerte y los drops).

Cheat

El juego debe tener cheats (trucos) que permitan **probar más fácilmente el trabajo**. Cuales y como ejecutarlos están definido por ustedes (se recomienda ser pragmáticos y buscar la facilidad)

Testing

Testear de forma automática todos los componentes de un juego en general es complicadísimo y en muchos casos imposible de hacer en su totalidad.

Sin embargo hay partes que son fácilmente testeables y que deberán presentar tests automáticos de **todo** el protocolo de comunicación entre el cliente y servidor.

Nota: para el testeo del protocolo podes usar sockets reales y testear con `tiburoncín` o puedes usar sockets fake. Ambas opciones tienen sus pros y contras. No te cases con ninguna y hace una PoC para ver cual te conviene.

Entregable

Junto con la documentación (manual de usuario, documentación técnica y del proyecto) se deberá entregar un Vagrantfile que instale las dependencias del proyecto, que clone el proyecto git, lo compile y lo instale en la vm. El Vagrantfile (y archivos adicionales que lo consideren si los hay) deben estar en el proyecto de git. Se recomienda usar como base un Ubuntu Focal: https://app.vagrantup.com/ubuntu/boxes/focal64

Aplicaciones Requeridas

Cliente

Se deberá implementar un cliente gráfico para que el usuario pueda conectarse al servidor, crear o unirse a una partida eligiendo el escenario a jugar.

Servidor

Se deberá implementar un servidor con soporte de múltiples partidas en simultáneo. Deberá poder indicarle a los clientes que se conecta qué escenarios hay disponibles así como también que partidas ya están creadas y están disponibles para que el usuario pueda unirse a alguna de ellas.

Editor (para grupos de 4 alumnos)

Se deberá implementar un editor de niveles. El editor deberá poder crear niveles nuevos así como también editar preexistentes. El servidor deberá poder cargarlos (al momento de iniciar el servidor) y enviarle a los

clientes el nivel.

El editor deberá poder posicionar los elementos de un escenario (terreno, paredes, monedas, gemas) y los spawn points (lugares en donde un jugador o enemigo podrá aparecer) así como también el fondo del escenario y otros detalles gráficos.

Restricciones

La siguiente es una lista de restricciones técnicas exigidas:

1. El sistema se debe realizar en ISO C++17 utilizando librerías QT y/o SDL.
2. La información de configuración debe ser almacenada en formato YAML. No está permitido utilizar una implementación propia de lectura y escritura de YAML: deben usar una librería.
3. Es condición necesaria para la aprobación del trabajo práctico la entrega de la documentación mínima exigida (consultar sitio de la cátedra). Es importante recordar que cualquier elemento faltante o de dudosa calidad pone en riesgo la aprobación del ejercicio.
4. Debe poder compilar y jugar en un Ubuntu 22.04 LTS. Las instrucciones de compilación, instalación y juego deben ser para esa plataforma y el trabajo será evaluado en ella.

Referencias

[1] Jazz Jackrabbit 2: https://en.wikipedia.org/wiki/Jazz_Jackrabbit_2
[2] Sprites: https://www.spriters-resource.com/pc_computer/jazzjackrabbit2thesecretfiles/
[3] Sonidos: https://www.youtube.com/watch?v=aGM4J060VCI
[4] Musica: https://www.youtube.com/watch?v=WjM9mFy22ks
[5] YAML: https://es.wikipedia.org/wiki/YAML