Portal

Ejercicio Final

Objetivos	● Afianzar los conocimientos adquiridos durante la cursada. ● Poner en práctica la coordinación de tareas dentro de un grupo de trabajo. ● Realizar un aplicativo de complejidad media con niveles aceptables de calidad y usabilidad.
Instancias de Entrega	**Pre-Entrega:** clase 14 (11/06/2019). **Entrega:** clase 16 (25/06/2019).
Temas de Repaso	● Aplicaciones Cliente-Servidor multi-threading. ● Interfaces gráficas con *gtkmm/cairo/SDL/qt* ● Manejo de errores en C++
Criterios de Evaluación	● Criterios de ejercicios anteriores. ● Construcción de un sistema Cliente-Servidor de complejidad media. ● Empleo de buenas prácticas de programación en C++. ● Coordinación de trabajo grupal. ● Planificación y distribución de tareas para cumplir con los plazos de entrega pautados. ● Cumplimiento de todos los requerimientos técnicos y funcionales. ● Facilidad de instalación y ejecución del sistema final. ● Calidad de la documentación técnica y manuales entregados. ● Buena presentación del trabajo práctico y cumplimiento de las normas de entrega establecidas por la cátedra (revisar criterios en sitio de la materia).

Índice

Introducción

El presente trabajo consiste en implementar una variante 2D del juego *Portal* [1] en el que los jugadores resolverán problemas de ingenio usando un dispositivo de creación de portales inter-espaciales formalmente conocido como *the Aperture Science Handheld Portal Device*.

Esta variante del juego será implementada en 2D en un modo multijugador cooperativo.

El juego tendrá una simulación de física para la trayectoria de los objetos y otras dinámicas usando el framework *Box2D* [2] y deberá poder grabar un video de las partidas usando *ffmpeg* [3].

Descripción

El juego consiste en un escenario constituido por bloques, rocas y compuertas en el que los jugadores (representados por Chell) debe moverse y sortear obstáculos usando su ingenio y su *portal gun* o *the Aperture Science Handheld Portal Device*.

El objetivo de cada nivel o escenario es poder que **todos** los jugadores lleguen al pastel (un lugar específico del escenario).

La muerte de uno de los jugadores **no** implica que el resto no pueda seguir jugando: es válido que, si todos los jugadores menos uno pudieron llegar al pastel, el jugador restante se *eutanasie* (suicide) o que sus compañeros lo *eutanasien* (lo maten), para así ganar, todo por el bien común y de la ciencia.

En este sentido, aunque el espíritu del juego es *cooperativo*, puede tornarse en *competitivo*.

Bloques

El escenario estará compuesto por bloques.

Bloques de roca: no ofrecen una superficie lo suficientemente plana para la formación de un portal ni para la reflexión de las bolas de energía.

Bloques de metal: sus superficies planas permiten la creación de portales y la reflexión de las bolas de energía.

Bloques de metal en diagonal: al igual que los bloques de metal estos permiten la creación de portales y la reflexión de bolas de energía. La particularidad de este bloque es el ángulo 45 grados de una de sus superficies. Hay 4 variantes de este tipo de bloques, una por cada ángulo.

Los jugadores y otros objetos pueden caminar por sobre los bloques; en el caso de los bloques en diagonal, tanto los jugadores como los objetos se resbalan y caen debido a la pendiente de la superficie en diagonal.

Ácido

El ácido se encuentra depositado sobre uno o varios bloques y es letal para los jugadores.

Botones

Los botones se encuentran sobre los bloques y son activados por peso: mientras un jugador o alguna roca estén sobre él el botón quedará en estado activado; sin peso, el botón se desactiva.

Rocas

Las rocas pueden ser cargadas y depositadas por los jugadores. Hay diferentes tipos de rocas pero todas ellas tienen la misma funcionalidad y la única diferencia es la imagen de cada roca.

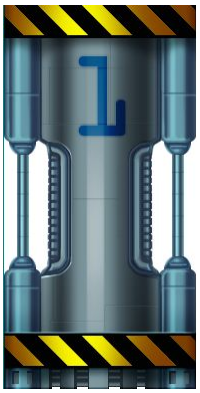
Emisores y receptores de energía

En el escenario pueden haber uno o más emisores de energía. Estos lanzan con cierta frecuencia *bolas de energía* que se mueven horizontal o verticalmente. Los receptores en cambio son elementos pasivos que esperan recibir una bola de energía y cuando lo hacen quedan activados.

Bolas de energía

Son emitidas por los *emisores*, sirven para activar a los *receptores* y se mueven a una velocidad determinada en dirección horizontal o vertical y rebotan (reflejan) sobre superficies de *metal* (sobre superficies en diagonal estas cambian de dirección). Luego de un cierto tiempo estas desaparecen.

Compuertas



Las compuertas no permiten el paso de ningún objeto y para abrirlas los jugadores deberán activar uno o varios botones o receptores de energía.

La combinación dependerá de cada compuerta: una puede abrirse con sólo mantener presionado un botón, otra puede activarse presionando dos botones **y (and)** un receptor, otra un botón **o (or)** otro botón.

Otras combinaciones son posibles incluyendo el operador **negación (not)**.

Barreras de energía

Las barreras de energía permiten el paso de los jugadores y las bolas de energía pero no las rocas ni de los rayos para crear los portales. Si un jugador traspasa la barrera cargando una roca esta se desintegra. Hay barreras horizontales como verticales.

Jugador

Los jugadores encarnaran a Chell, la protagonista del juego original Portal [1]. Chell es capaz de moverse lateralmente, de saltar y de disparar un rayo para la creación de portales con su *portal gun* o *the Aperture Science Handheld Portal Device*.
Chell cuenta con un equipamiento especial que evita que sufra daños al caer, no importa desde qué tan alto. Sin embargo ella es vulnerable y morirá si le cae una roca encima, se le cierra una compuerta encima, cae en ácido o es impactada por una bola de energía.

Creación de portales

Hay 2 portales para cada jugador, el azul y el naranja. Una vez creados estos forman un túnel bidireccional que puede teletransportar a los jugadores, rocas y bolas de energía instantáneamente.

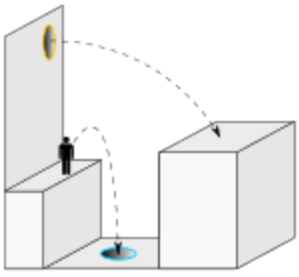
La orientación del portal dependerá de la orientación de la superficie del bloque en donde se creó: horizontal, vertical o diagonal.

Cada jugador podrá crear tantos portales como quiera pero la creación de un portal azul implica que el portal azul anterior del mismo jugador se desactiva (desaparece). Lo mismo aplica al portal naranja.

El jugador puede usar el mouse para indicar la dirección hacia a donde disparar un rayo en línea recta que al impactar crea un portal. Puede crear un portal azul o uno naranja haciendo click izquierdo o derecho.

El o los portales de un jugador pueden desactivarse a modo de *reset* usando el teclado.

Conservación del movimiento relativa



Cuando un jugador u otro elemento ingresa a un portal a cierta velocidad este sale por el portal opuesto con la misma velocidad en magnitud y misma dirección **respecto** a la normal del portal.

Desde la óptica del jugador o elemento que se mueve entre portales su *cantidad de movimiento* es *conservada* tanto en magnitud como en dirección.

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 jugador y seguirlo a medida que se desplaza.

Animaciones

El juego no debe mostrar imágenes estáticas sino pequeñas animaciones para darle mayor realismo [4]:

- El movimiento de los jugadores: cuando se desplazan, saltan, disparan, mueren.
- El movimiento de las bolas de energía, las compuertas y los portales.
- Activación y desactivación de botones y otros elementos..

Sonidos

Como todo juego se debe reproducir sonidos para darle realismo a los eventos y acciones que suceden[5]:

- Cuando hay disparos.
- Cuando un jugador salta.
- Cuando un jugador muere.

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[5].

Interfaz del jugador

El juego debe poder renderizarse en full screen y en modo ventana con el tamaño de esta configurable.

Cada jugador tendrá un color asociado de tal forma que se puedan distinguir las distintas Chells.

Pin tool

El jugador puede crear un *pin*, una marca de donde uno de sus compañeros debe crear un portal. La mecánica de la creación de pins es igual a la de creación de portales (usando el mouse) pero presionando otro botón/tecla para diferenciar la creación de uno de otro.
Los pins desaparecen luego de cierto tiempo o cuando otro pin del mismo jugador es creado.

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.
Con la aplicación cliente el jugador podrá además de jugar iniciar o frenar la grabación en video de la partida actual.

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.

Todos los atributos del juego (velocidad, altura de salto, etc) deben ser configurables por archivo.

> *Es importante que todos los parámetros sean configurables: permite que que se ajusten para tener un juego más balanceado y divertido a la vez que le permite a los docentes realizar pruebas.*

Editor

Se deberá implementar un editor de escenarios que permita:

- Crear nuevos escenarios (o niveles) o editar previos.
- Definir la locación de los bloques, rocas, compuertas y otros elementos que forman el escenario.
- Definir la locación inicial de los jugadores y cuantos jugadores son requeridos para que se unan y puedan jugar a dicho escenario y la locación del pastel.
- Cada nivel debe tener además una imagen estática que sirva de fondo.
- Definir las reglas (lógica booleana) para abrir o cerrar cada compuerta en función de la activación o desactivación de uno o varios botones y/o receptores de energía. *Nota: hacer que cada botón y receptor de energía tenga un nombre.*
- Verificar ciertas condiciones mínimas como al menos una Chell y un solo pastel.

Distribución de Tareas Propuesta

Con el objetivo de organizar el desarrollo de las tareas y distribuir la carga de trabajo, es necesario planificar las actividades y sus responsables durante la ejecución del proyecto. La siguiente tabla plantea una posible división de tareas de alto nivel que puede ser tomada como punto de partida para la planificación final del trabajo:

	Alumno 1 Servidor - Modelo	Alumno 2 Cliente - Modelo	Alumno 3 Cliente - Editor
Semana 1 (30/04/2019)	- Draft del modelo (incluyendo lógica del juego y partidas multijugador) - Prueba de concepto con Box2D.	- Mostrar una imagen. - Mostrar una animación. - Mostrar ambas en un lugar fijo o desplazándose por la pantalla (movimiento).	- Draft del cliente y del editor (*wireframe*). - Prueba de concepto con ffmpeg.
Semana 2 (07/05/2019)	- Escenario (bloques, rocas y otras cosas no dinámicas)	- Renderizado del escenario incluyendo la cámara.	- Edición de un escenario básico con solo objetos estáticos.
Semana 3 (14/05/2019)	- Chell (jugador), carga/descarga de rocas y bolas de energía y otros elementos dinámicos.	- Animación de los elementos dinámicos.	- Edición de un escenario incluyendo la locación de las Chells y del pastel.
Semana 4 (21/05/2019)	- Portales y física completa. Lógica de compuertas	- Finalización de la parte gráfica incluyendo la pin tool.	- Edición de las condiciones booleanas para cada compuerta.
Semana 5 (28/05/2019)	- Servidor multipartidas con partidas multijugador. Condiciones de victoria y derrota.	- Música y sonido. Pantallas de login, creación de partidas y de unirse a partidas.	- Captura de video de una partida: inicio y frenado a voluntad.
Semana 6 (04/06/2019)	- Testing - Correcciones y *tuning* del Servidor - Documentación	- Testing - Correcciones y *tuning* del Cliente - Documentación	- Testing - Correcciones y *tuning* del Editor - Documentación
Entrega el 11/06/2019			
Semana 7 (11/06/2019)	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación
Semana 8 (18/06/2019)	- Testing - Correcciones sobre la primer entrega - Armado del entregable	- Testing - Correcciones sobre primer entrega - Armado del entregable	- Testing - Correcciones sobre primer entrega - Armado del entregable
Reentrega el 25/06/2019			

Restricciones

La siguiente es una lista de restricciones técnicas exigidas por el cliente:

1. El sistema se debe realizar en C++11 utilizando librerías *gtkmm, SDL y/o qt*.

2. Los archivos de configuración deben ser almacenados en formato *YAML* [6]. A tal fin, y con el objetivo de minimizar tiempos y posibles errores, se permiten distintas librerías externas (consultar sitio de la cátedra). No está permitido utilizar una implementación propia de lectura y escritura de *YAML*.
3. Para la simulación de la física del juego se debe usar el framework *Box2D* [2].
4. Para la grabación del video de las partidas se debe usar *ffmpeg* [3].
5. 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.
6. Entrega de uno o varios escenarios con la suficiente diversidad de elementos a tal fin que sea fácil mostrar las funcionalidades implementadas.
7. De forma opcional, se sugiere la utilización de alguna librería del estilo xUnit [7]. Si bien existen varias librerías disponibles en lenguaje C++ [8], se recomienda optar por CxxTest [9] o CppUnit [10].

Referencias

[1] Portal: https://en.wikipedia.org/wiki/Portal_(video_game)
[2] Box2D: http://box2d.org/manual.pdf
[3] ffmpeg: https://ffmpeg.org/
[4] Sprites: https://www.spriters-resource.com/pc_computer/mightyswitchforcehyperdriveedition/
[5] Efectos y música ambiente: https://www.youtube.com/watch?v=2pByCegIjpU
https://www.youtube.com/watch?v=Kgny67NTtw0 https://theportalwiki.com/wiki/GLaDOS_voice_lines
[6] YAML: https://es.wikipedia.org/wiki/YAML
[7] Frameworks XUnit: http://en.wikipedia.org/wiki/XUnit
[8] Variantes XUnit para C/C++: http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#C.2B.2B
[9] CxxTest: http://cxxtest.com/
[10] CppUnit: http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page