Tower Defense

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 (14/11/2017). **Entrega:** clase 16 (28/11/2017).
Temas de Repaso	● Aplicaciones Cliente-Servidor multi-threading. ● Interfaces gráficas con *gtkmm/cairo/SDL* ● 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 juego a implementar es una variante del clásico género Tower Defense en donde una serie de enemigos emergen desde un portal o varios y se mueven por caminos predefinidos hacia su destino y es el objetivo de los jugadores evitar que lleguen a destino.

Para ello, los jugadores deben cooperar entre sí y colocar torres de defensa que atacarán automáticamente a los enemigos así como también lanzar hechizos contra los enemigos directamente.

Los enemigos aparecen de a hordas cada cierto tiempo, dependiendo de cada nivel. Luego de haber acabado con todas las hordas, los jugadores logran la victoria; si un solo enemigo logra llegar a destino, los jugadores pierden.

Descripción

Jugadores

Como se mencionó anteriormente el juego es multijugador con un límite de hasta 4 jugadores por partida. Cada jugador tendrá un elemento asociado que lo habilitará a poder construir torres y hechizos de ese elemento.

Los elementos son tierra, fuego, agua y aire, y se distribuyen entre los jugadores participantes sin repetición (no puede haber dos jugadores con el mismo elemento, por ejemplo).
En el caso de haber menos jugadores que 4, un jugador podrá tener más de un elemento a su disposición.

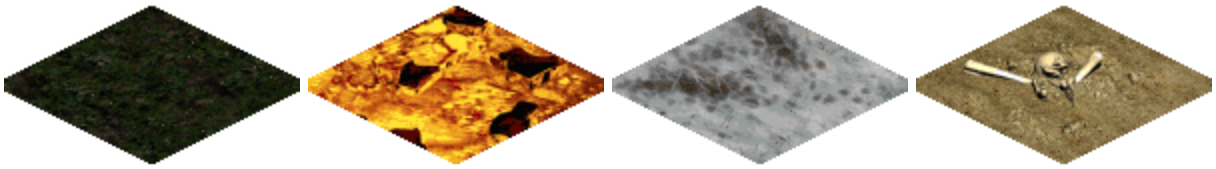
Escenario

El escenario donde transcurre la partida consta de uno o más portales de entrada por donde los enemigos emergerán y uno o más portales de salida por donde los enemigos intentarán salir.


Los enemigos no pueden caminar por cualquier parte sino que lo harán por rutas predefinidas sobre los caminos. No se puede construir torres sobre un camino.
En cambio, las torres solo pueden construirse sobre *terreno firme*. Un escenario cuenta con unos pocos terrenos firmes lo que limita a los jugadores en la cantidad de torres a construir y su ubicación.

Un escenario tendrá un *ambiente* particular que completará el resto de los terrenos que no son ni los *caminos* ni los *terrenos firme*.
Un escenario ambientado en la pradera tendrá terrenos como tierra y pasto, mientras que un escenario ambientado en una zona gélida tendrá nieve e hielo.

La temática del juego no cambia la mecánica de este, tan solo sirve para mostrar imágenes o sprites distintos.
Se deben implementar 4 temáticas distintas: pradera, volcán, gélido y desierto.



Torres

Cada jugador dispone de una serie de torres para colocar. Puede colocar tantas como desee siempre que tenga un lugar donde colocarlas.

Una vez construida una torre el jugador debe esperar 20 segundos para poder construir la siguiente.

Torre de Tierra

Ataque: lanza piedras contra un objetivo. No puede atacar a unidades aereas.
Rango: 2
Frecuencia: 1 lanzamiento cada 5 segundos.
Daño: 20

Upgrades

Rango: +1 por cada (500 * 2^n) puntos de experiencia.
Daño: +10 por cada (100 * 1.5^n) puntos de experiencia.

Torre de Fuego

Ataque: lanza bolas de fuego contra un objetivo. El impacto genera una explosión que daña a todos los enemigos que estén cerca.
Rango: 3
Frecuencia: 1 lanzamiento cada 3 segundos.
Daño: 6 (contra el objetivo); 3 (contra los enemigos cercanos)
Alcance del impacto: 1

Upgrades

Rango: +1 por cada (100 * 2^n) puntos de experiencia.
Daño: +6 (contra el objetivo) y +3 (contra los enemigos cercanos) por cada (100 * 1.5^n) puntos de experiencia.
Alcance del impacto: +1 por cada (500 * 2^n) puntos de experiencia.

Torre de Agua

Ataque: lanza bloques de agua/hielo contra un objetivo que además de dañarlo lo ralentiza por cierta cantidad de segundos.
Rango: 3
Frecuencia: 1 lanzamiento cada 3 segundos.
Daño: 4 (contra el objetivo)
Ralentizado: 25% de la velocidad del enemigo por 2 segundos.

Upgrades:

Rango: +1 por cada (100 * 2^n) puntos de experiencia.
Daño: +4 (contra el objetivo) por cada (100 * 1.5^n) puntos de experiencia.
Ralentizado: +25% de la velocidad del enemigo (hasta llegar a 100%), +1 segundo de duración por cada (100 * 2^n) puntos de experiencia.

Torre de Aire

Ataque: lanza rafagas contra un objetivo. Causa un especial daño contra objetivos aéreos. En ambos casos, cada impacto hace retroceder al enemigo.
Rango: 5
Frecuencia: 1 lanzamiento cada 5 segundos.
Daño: 2 (contra un objetivo no aéreo); 10 (contra un objetivo aéreo).

Upgrades:

Rango: +1 por cada (100 * 2^n) puntos de

experiencia. puntos de experiencia.
Daño: +2 (contra un objetivo no aéreo) y +10
(contra un objetivo aéreo) por cada (100 * 1.5^n)

Experiencia y upgrades

Cada vez que una torre daña a una unidad enemiga, la misma cantidad de puntos son sumados como puntos de experiencia para la torre.
Si una torre mata a un enemigo ademas recibe un bonus de puntos de experiencia igual al 50% de los puntos de vida que tenía el enemigo.

Las torres pueden ser upgradeadas consumiendo los puntos de experiencia. A mayor nivel de upgrade mayor será la cantidad de puntos necesaria.

Por ejemplo, supongamos una Torre de Agua recién construida. Ella dispone de 3 posibles upgrades:

- Rango: +1
- Daño: +4
- Ralentizado: +25% de la velocidad del enemigo ,+1 segundo de duración.

Como esta torre no tiene ningun upgrade aplicado, el nivel de upgrade **n** actual es 1. Por lo tanto, las cantidades de experiencia necesaria para realizar un upgrade son:

- Rango: (100 * 2^n) == (100 * 2^1) == 200 puntos de experiencia.
- Daño: (100 * 1.5^n) == (100 * 1.5^1) == 150 puntos de experiencia.
- Ralentizado: (100 * 2^n) == (100 * 2^1) == 200 puntos de experiencia.

Supongamos que dicha torre ataca a un enemigo que tiene 10 puntos de vida. Dispara 3 veces, donde los primeros 2 impactos le sacan 4 puntos de vida cada uno y el último le saca solo 2 (4+4+2 = 10).
Por cada daño realizado recibe la misma cantidad de puntos: 4+4+2 = 10 puntos de experiencia.
Ademas, por haber destruido al enemigo recibe un bonus de 50% * 10 = 5 puntos de experiencia.
La torre adquiere entonces un total de 15 puntos.

Supongamos ahora que la torre elimina a 14 enemigos en total, dando le un total de 14 * 15 = 210 puntos de experiencia.
El jugador ahora puede optar por realizar un upgrade.
Supongamos que desea mejorar el rango por lo que hace un upgrade del Rango gastando 200 puntos de experiencia.

Ahora la torre tiene solo 10 puntos de experiencia y cuenta con un nivel de upgrade **n** para el Rango de 2 y 1 para el resto de los posibles upgrades. Por lo tanto, las cantidades de experiencia necesaria para realizar un upgrade son:

- Rango: (100 * 2^n) == (100 * 2^2) == 400 puntos de experiencia. (n == 2)
- Daño: (100 * 1.5^n) == (100 * 1.5^1) == 150 puntos de experiencia. (n == 1)
- Ralentizado: (100 * 2^n) == (100 * 2^1) == 200 puntos de experiencia. (n == 1)

Hechizos

El jugador puede lanzar hechizos directamente con sólo seleccionar qué habilidad especial desea usar y luego seleccionar sobre qué zona aplicar.

Terraforming

Habilidad de Tierra que transforma cualquier terreno (excepto un camino) en un terreno firme donde se pueda construir una torre.
Tiempo de recarga: 20 segundos.

Grieta

Habilidad de Tierra que abre una grieta en un camino. Todo aquel que pase por la grieta cae en ella y es destruido inmediatamente. Los enemigos aéreos no son afectados.
Duración de la grieta: 1 segundo.
Tiempo de recarga: 40 segundos.

Meteorito

Habilidad de Fuego que lanza un meteorito sobre un lugar particular y daña a todos los enemigos cercanos
Daño: 30 (contra el objetivo); 10 (contra los enemigos cercanos)
Alcance del impacto: 2
Tiempo de recarga: 20 segundos.

Muro de fuego

Habilidad de Fuego que prende fuego un lugar particular dañando a todo aquel que pase por él.
Daño: 10
Duración del muro: 5 segundos.
Tiempo de recarga: 10 segundos.

Congelacion

Habilidad de Agua que no realiza daño alguno pero que ralentiza al 100% un enemigo en particular.
Ralentizado: 100% de la velocidad del enemigo por 5 segundos.
Tiempo de recarga: 15 segundos.

Ventisca

Habilidad de Agua que crea una tormenta congeladora sobre una zona que daña a todo aquel que pase por ella y los ralentiza.
Daño: 5
Ralentizado: 25% de la velocidad del enemigo por 2 segundos.
Duración de la ventisca: 5 segundos.
Tiempo de recarga: 20 segundos.

Tornado

Habilidad de Aire que crea un tornado en un lugar particular. El tornado no se desplaza pero daña a todo aquel que pase por él.
Daño: 1-50 (aleatorio)
Duración del tornado: 10 segundos.
Tiempo de recarga: 20 segundos.

Rayos

Habilidad de Aire que lanza un rayo sobre un enemigo.
Daño: 1-50 (aleatorio)
Tiempo de recarga: 10 segundos

Enemigos

Abmonible

Vida: 200 puntos.
Velocidad: 1
Aéreo: no.

Demonio verde

Vida: 300 puntos.
Velocidad: 1
Aéreo: no.

Halcon sangriento

Vida: 100 puntos.
Velocidad: 4
Aéreo: si.

Espectro

Vida: 100 puntos.
Velocidad: 6
Aéreo: si.

Hombre cabra

Vida: 100 puntos.
Velocidad: 2
Aéreo: no.

No muerto

Vida: 20 puntos.
Velocidad: 10
Aéreo: no.

Jugabilidad

Cada jugador debe poder ver una porción del escenario con una vista de tipo agila.

Las torres deben atacar automáticamente a las unidades enemigas sin que el jugador lo indique siempre que estas estén dentro de su alcance.

Para poder coordinar las acciones entre los jugadores, además de mostrar el escenario se mostrará una pequeña sala de chat en el cual se podrá enviar y recibir mensajes entre los participantes.

Cada jugador podrá además marcar una zona del escenario temporalmente. Esta marca será visible por todos los jugadores lo que le permitirá a un jugador decir por chat “construye una torre aquí” y marcar en el escenario en dónde.

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.

Animaciones

Tanto las unidades como las torres y portales no son mostrados con imágenes estáticas sino con pequeñas animaciones.
Los enemigos deben tener una animación mientras se desplazan por el camino y cuando mueren así como también las torres deben tener una animación mientras disparan.

Interfaz del jugador

Se debe mostrar la parte del mapa que el jugador está viendo permitiéndole moverse al desplazar el mouse o al usar las teclas de dirección (arriba, abajo, izquierda, derecha). La vista que el jugador posee es conocida como *vista de águila*, en donde el jugador puede ver el mapa y sus elementos desde arriba.

El mapa debe mostrarse con una proyección isométrica.

Al seleccionar una torre del jugador se debe poder ver una descripción de ella: cuánto daño puede causar, su rango, su experiencia acumulada y demás.
Se debe poder ver qué upgrades están disponibles y dar la posibilidad de realizarlo.

Al finalizar el escenario se deberá mostrar una pantalla de victoria o derrota dependiendo de cada caso.

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 nivel o escenario a jugar así como también el elemento (tierra, fuego, agua o aire) con el cual jugar.

Servidor

Se deberá implementar un servidor con soporte de múltiples partidas en simultáneo. Deberá poder indicarle a los clientes que se conecta qué niveles o 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 de los enemigos (vida, velocidad, …), de las torres (daño, rango, cantidad de puntos de experiencia necesarios para un upgrade, …) y de los hechizos (daño, tiempo de recarga, …) deben ser configurables por archivo.

Editor

Se deberá implementar un editor de niveles o escenarios que permita:

- Elegir el *ambiente* del escenario.
- Colocar los terrenos firmes, caminos, portales y demás.
- Definir la cantidad de hordas que tendrá el nivel:
    - La cantidad de enemigos y los tipos de cada horda.
    - Los tiempos entre cada horda.

Además deberá realizar chequeos de consistencia para evitar errores por parte del diseñador como:

- Un escenario sin portales de entrada o de salida.
- Portales que no están conectados por un camino o caminos que llevan a ningún lado.
- Un escenario sin hordas u hordas sin enemigos.

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 (03/10/2017)	Draft del modelo (incluyendo lógica del juego, chat y partidas multijugador)	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*).
Semana 2 (10/10/2017)	Modelado del escenario, las hordas, las torres y los disparos.	Dibujado del escenario con proyección isométrica. Dibujado de animaciones (portales, torres, enemigos, disparos, explosiones)	Editor, creación de escenarios definiendo qué ambiente usar, y cómo son las hordas del nivel.
Semana 3 (17/10/2017)	Finalización del modelado del juego.	Interfaz gráfica al seleccionar una torre (mostrar los detalles). Interfaz para el upgrade. Scrolling.	Permitirle al usuario dibujar sus propios escenarios: los terrenos, los caminos, los portales.
Semana 4 (24/10/2017)	Lógica para la creación de partidas multijugador y múltiples partidas.	Interfaz para el lanzamiento de hechizos. Interfaz para el chat y la señal temporal.	Finalización del editor.
Semana 5 (31/10/2017)	Soporte para la sala de chat. Finalización de la implementación multijugador.	Interfaz para la conexión con el servidor, elegir un nivel, crear/unirse a una partida. Pantallas de victoria y derrota.	Implementación del sistema de sonidos. Carga y descarga de niveles.
Semana 6 (07/11/2017) **Preentrega**	- Testing - Correcciones y *tuning* del Servidor - Documentación	- Testing - Correcciones y *tuning* del Cliente - Documentación	- Testing - Correcciones y *tuning* del Editor - Documentación
Preentrega el 14/11/2017			
Semana 7 (14/11/2017)	- Correcciones sobre Preentrega - Testing y corrección de bugs - Documentación	- Correcciones sobre Preentrega - Testing y corrección de bugs - Documentación	- Correcciones sobre Preentrega - Testing y corrección de bugs - Documentación

Semana 8 (21/11/2017) **Entrega**	- Testing - Documentación - Armado del entregable	- Testing - Documentación - Armado del entregable	- Testing - Documentación - Armado del entregable
Entrega el 28/11/2017			

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 y/o SDL*.
2. Con el objetivo de facilitar el desarrollo de las interfaces de usuario, se permite el uso de Glade.
3. Los archivos de configuración deben ser almacenados en formato YAML. 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.
4. 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.
5. De forma opcional, se sugiere la utilización de alguna librería del estilo xUnit [2]. Si bien existen varias librerías disponibles en lenguaje C++ [3], se recomienda optar por CxxTest [4] o CppUnit [5].

Referencias

[1] Sprites: https://www.spriters-resource.com/pc_computer/diablo2diablo2lordofdestruction/
[2] Frameworks XUnit: http://en.wikipedia.org/wiki/XUnit
[3] Variantes XUnit para C/C++: http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#C.2B.2B
[4] CxxTest: http://cxxtest.com/
[5] CppUnit: http://sourceforge.net/apps/mediawiki/cppunit/index.php?title=Main_Page