Facultad de Ingeniería - U.B.A.

75.42 Taller de Programación I

1er. Cuatrimestre de 2018

Trabajo Práctico Final: Worms

Manual de proyecto

Grupo: 4

Corrector: Pablo Daniel Roca

Alejandro Daneri, Padrón: 97839
alejandrodaneri07@gmail.com

Matias Leandro Feld, Padrón: 99170
feldmatias@gmail.com

Agustín Zorzano, Padrón: 99224
aguszorza@gmail.com

Repositorio a Github:
https://github.com/AlejandroDaneri/tp-final-taller

1 Integrantes

Grupo 4:

Alumno	Padrón	Mail
Alejandro Daneri	97839	alejandrodaneri07@gmail.com
Matias Feld	99170	feldmatias@gmail.com
Agustin Zorzano	99224	aguszorza@gmail.com

Corrector:

Pablo Daniel Roca

2 Enunciado

El enunciado se encuentra adjunto al final de este informe.

3 División de tareas

La división de las tareas se basó en lo propuesto por el enunciado, teniendo en cuenta que el proyecto consiste de tres ejecutables distintos, cada uno con su propia funcionalidad.

Alejandro se ocupó de la parte correspondiente al editor. Se encargó de crear la interfaz gráfica y agregarle toda la funcionalidad necesaria. Además se encargó de buscar las imágenes necesarias para mejorar la interfaz. Por otro lado se encargó de generar la persistencia de los mapas, tanto como para futuras ediciones como para la carga de los mapas dentro del juego.

Agustín se encargó de la parte correspondiente al cliente. En este caso debió crear toda la interfaz de usuario que permite visualizar el juego, incluyendo animaciones y sonidos. También debió crear la funcionalidad que le permite recibir información del servidor y actualizar la interfaz en consecuencia. También se ocupó de generar todas las formas posibles que tiene el usuario de interactuar con el juego, como botones del teclado y mouse.

Matías se encargó de la parte correspondiente al servidor. Por un lado tuvo que crear toda la lógica interna del juego, simulando un mundo físico real donde hay objetos que se mueven e interactúan entre sí. Por otro lado se encargó de crear el protocolo de comunicación entre el servidor y el cliente, incluyendo todos los posibles mensaje que se pueden enviar a través de la red. Por último, se encargó de crear toda la parte del servidor que permite manejar múltiples clientes y múltiples partidas al mismo tiempo.

4 Evolución del proyecto

El cronograma en el que nos basamos es el propuesto por el enunciado. El cronograma real fue bastante similar, aunque nos algunos cambios.

Semana	Cliente	Servidor	Editor
1	Se realizó un modelo básico de cómo funciona la interfaz. Se agregaron imágenes que se movían recto y en parábola.	Se realizó un modelo básico de cómo funciona la lógica del juego. Se comenzó con la parte básica del juego, agregando objetos al mundo y que caigan con la gravedad. Se realizó el protocolo básico de comunicación entre el cliente y el servidor.	Se realizó un modelo básico de cómo funciona la interfaz, y la funcionalidad que debe ofrecer.
2	Se agregó el movimiento de los gusanos en función de los datos recibidos del servidor. Se realizaron mejoras en la interfaz. Se agregaron las imágenes de las vigas y las armas que se disparan.	Se agregó la lógica principal del juego, como el movimiento de los gusanos y el disparo de armas. Se agregaron armas que lanzan fragmentos al explotar y armas que explotan en un tiempo determinado. Se agregó la lectura del archivo de configuración.	Se agregó la funcionalidad básica necesaria, como la ubicación de una imagen en una posición específica, seleccionar distintos tipos de imágenes para ubicar, y mover o borrar una imagen.
3	Se agregaron los botones de selección de arma, así como los controles por teclado y mouse. Se agregó comportamiento al gusano como saltar o moverse. Se agregaron vistas en el juego como el contador del turno o la lista de jugadores.	Se terminó la lógica del juego. Se agregaron detalles como las vigas inclinadas donde el gusano no debe deslizar y el viento. Se agregó comportamiento para cuando un jugador se desconecta inesperadamente.	Se agregaron las opciones para permitir al usuario elegir las municiones de las armas. Se mejoró la interfaz de usuario. Se permite la rotación de una imagen.
4	Se agregaron sonidos para distintas acciones, así como una música de fondo. Se agregaron distintas animaciones, como el movimiento del gusano y la explosión de un arma. Se comenzó con los menúes que conectan con el servidor para elegir una partida.	Se realizó la lógica para permitir partidas con muchos jugadores y múltiples partidas al mismo tiempo.	Se agregó funcionalidad para permitir seleccionar la imagen a mover, rotar o borrar. También se agregó el guardado y cargado de mapas en archivos.

5	Se terminaron todas las pantallas necesarias. Se agrego la posibilidad de hacer scrolling de la pantalla del juego.	Corrección de bugs y mejoras.	Se terminaron todas las pantallas necesarias.
6	Corrección de bugs y mejoras	Corrección de bugs y mejoras.	Corrección de bugs y mejoras.

5 Inconvenientes encontrados

- El principal inconveniente encontrado fue en el cliente, ya que la pantalla se congelaba o aparecían errores que parecían ser al azar. El problema se debió a que se estaba utilizando mal la librería gráfica. Se estaba modificando la interfaz gráfica desde el hilo de recepción de datos, que era un hilo de ejecución distinto al hilo principal de dibujado. Se solucionó moviendo todos los cambios de interfaz al hilo principal.

- Otro problema que tuvimos fue el de rotar imágenes, ya que la librería solo permitía rotar una imagen 90 grados, y esto no nos servía para implementar las vigas con distintos ángulos. La solución fue crear una imagen distinta para cada rotación posible, en lugar de rotar una única imagen.

- El aprendizaje de la librería gráfica GTK requirió la dedicación de una cierta cantidad de tiempo al inicio del trabajo práctico. Una vez iniciado el proyecto se fueron encontrando problemas que necesitaron de una nueva etapa de investigación sobre esta librería para poder encontrar la solución al problema. Este mismo problema ocurrió con la librería SDL_MIXER pero en menor medida.

6 Herramientas

6.1 Librerías y compilación

- Para la interfaz gráfica, tanto del cliente como del editor, se utilizó la librería Gtkmm para el lenguaje C++, en su versión 3. En conjunto con esto, se utilizó la herramienta Glade para generar interfaces de una forma visual.

- Para los sonidos del juego se usó SDL mixer, en su versión 2.

- Para simular la física del juego se utilizó la librería Box2d, que permite simular la gravedad, colisiones entre objetos, movimiento de objetos, etc.

- Para la lectura y escritura de archivos de configuración en formato Yaml, se utilizó la herramienta yaml-cpp.

- Para la compilación y prueba del programa se utilizó la herramienta Cmake, que permite generar un makefile de compilación según los comandos indicados.

6.2 Debug

- Para debuguear y buscar errores se utilizaron tanto valgrind como gdb.

6.3 Control de versiones

- Para el control de versiones se utilizó Git, con un repositorio en la plataforma Github.

6.4 Documentación

- Para la lista de tareas, se utilizó la plataforma Trello.

- Para generar los diagramas de documentación se utilizó la herramienta Astah o la herramienta online lucidchart.

- Para la escritura de la documentación se utilizó la plataforma Google Docs.

7 Conclusiones

La programación de un juego en equipo requiere de la coordinación entre los miembros para poder crear un modelo que se comunique internamente de forma coherente y sencilla. En la división de tareas, hay puntos que son más críticos o indispensables que otros. Esto se pudo apreciar principalmente en las secciones del servidor y el cliente, en donde si uno se atrasaba, dificultaba el trabajo del otro o bien dificultaba la verificación del correcto funcionamiento de los cambios realizados.

Por otro lado, se requiere realizar un análisis profundo del proyecto a realizar, previo al comienzo del desarrollo, para permitir la organización del equipo, y principalmente la correcta división de tareas para cumplir con el objetivo en tiempo y forma.

Worms

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 (12/06/2018). **Entrega:** clase 16 (26/06/2018).
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 trabajo final consiste en la remake del icónico Worms [1]: un juego multijugador en el que los jugadores controlan un pequeño ejército de gusanos altamente armados en la que se enfrentarán a muerte.

El juego tendrá una simulación de física para la trayectoria de los misiles, las explosiones y otras dinámicas usando el framework Box2D [2].

Descripción

Escenario

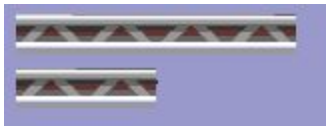
El escenario estará compuesto por vigas indestructibles.

Los gusanos pueden caminar y desplazarse por las vigas horizontales o que tienen una ligera pendiente (menores o iguales a 45 grados).

Las vigas con mayor pendiente y verticales no pueden ser recorridas por los gusanos. Si algún gusano se encontrase sobre una viga tal, el gusano se debe resbalar y caer por acción de la gravedad.

Hay 2 longitudes posibles de vigas: largas (6 mts) y cortas (3 mts) y ambas tienen una altura de 0.80 mts.

Salvo su longitud no tienen ninguna otra diferencia.



La parte inferior del escenario está cubierta de agua. Cualquier gusano que caiga en ella muere ahogado.

El fondo de pantalla debe ser una imagen sin otro objetivo más que la estética.

Armas con mira

Algunas armas le permiten al jugador apuntar hacia donde quiere disparar.

El jugador puede usar las flechas de arriba (aumenta el ángulo) y abajo (reduce el ángulo) para direccionar la mira.

Los ángulos posibles están en el rango de -90 grados (el gusano apunta verticalmente hacia abajo) y 90 grados (el gusano apunta verticalmente hacia arriba).

Un ángulo de 0 grados apunta horizontalmente.

El jugador puede usar las flechas izquierda y derecha para apuntar hacia la izquierda y derecha respectivamente.

Arma de combate cuerpo a cuerpo

Simplemente algunas armas producen el daño en el lugar donde está el gusano.

La mayoría de las armas suelen ser a distancia.

Potencia de disparo variable

Algunas armas permiten un ajuste de la potencia del disparo.

Presionando y manteniendo presionada la tecla Espacio, la potencia se acumula.

Cuando el jugador suelta la tecla o bien se llega a un máximo de potencia, el disparo se produce con una velocidad proporcional a la potencia alcanzada.

Cuenta regresiva

Los proyectiles de algunas armas no explotan en el instante del impacto sino que lo hacen luego de cierta cantidad de segundos.

El jugador puede prefijar el tiempo de espera presionando algunas de las teclas '1', '2', ‘3’, ‘4’, '5' seleccionando así una cuenta regresiva de 1, 2, 3, 4 o 5 segundos respectivamente.

Por defecto, la cuenta regresiva es de 5 segundos.

Teledirigido

Algunos proyectiles, e incluso algunas herramientas, no necesitan apuntar; el jugador hace click con el mouse en alguna parte del escenario para determinar el punto exacto donde el proyectil debería caer o la herramienta debería usarse.

Trayectoria afectada por el viento

Algunos proyectiles son afectados por la dirección y velocidad del viento. El viento puede ir de izquierda a derecha o al revés con una velocidad que varía desde los 0.2 mts/seg a hasta los 10 mts/seg.

Municiones

Es la cantidad de veces que un gusano puede usar el arma (o herramienta).

Daño

Cada arma genera un daño **a todos** los gusanos que están en la cercanía del ataque (típicamente una explosión).

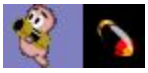
El daño es inversamente proporcional al rango de la explosión: a menor distancia del epicentro de la explosión, mayor el daño.

Cada explosión es característica del arma usada y tiene un daño máximo (en el epicentro) y rango particular.

Un gusano que se encuentra en el rango de la explosión adquiere cierta velocidad inversamente

proporcional al rango de la explosión (los gusanos salen volando debido a la explosión) y adquiere cierta dirección.

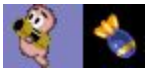
Armas y herramientas



Bazooka

Arma predilecta de los gusanos que dispara un misil que estalla al impactar con un tiro parabólico.

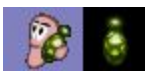
Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: no
Teledirigido: no
Trayectoria afectada por el viento: si
Municiones: infinitas
Daño: 50 pts (epicentro); 2 mts (radio)



Mortero

Es igual a la bazooka pero al estallar lanza fragmentos al aire con trayectoria parabólica que estallan al impactar.

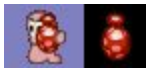
Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: no
Teledirigido: no
Trayectoria afectada por el viento: sí
Municiones: 10
Daño (del estallido principal): 50 pts (epicentro); 2 mts (radio)
Daño (de cada fragmento): 10 pts (epicentro); 2 mts (radio)
Cantidad de fragmentos: 6



Granada Verde

La segunda arma predilecta de los gusanos. Como otros tipos de granada no se ve afectada por el viento.

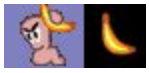
Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: sí
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: infinitas
Daño: 30 pts (epicentro); 2 mts (radio)



Granada Roja

Al igual que el Mortero, al explotar lanza fragmentos al aire con tiro parabólico que estallan al impacto.

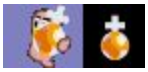
Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: sí
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: 10
Daño (del estallido principal): 30 pts (epicentro); 2 mts (radio)
Daño (de cada fragmento): 10 pts (epicentro); 2 mts (radio)
Cantidad de fragmentos: 6



Banana

Es un tipo de granada que tiene la particularidad de rebotar varias veces de forma muy elástica hasta explotar.

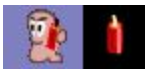
Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: sí
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: 5
Daño: 70 pts (epicentro); 4 mts (radio)



Granada Santa

Una de las armas más poderosas en el arsenal. Justo antes de estallar, produce un sonido característico.

Arma con mira: sí
Combate cuerpo a cuerpo: no
Disparo con potencia variable: sí
Cuenta regresiva: sí
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: 2
Daño: 110 pts (epicentro); 8 mts (radio)



Dinamita

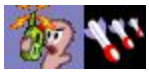
Al activarse, el gusano deja en el lugar una dinámica activa que estalla luego de cierta cantidad de segundos.

Arma con mira: no
Combate cuerpo a cuerpo: sí
Disparo con potencia variable: no
Cuenta regresiva: sí
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: 5
Daño: 50 pts (epicentro); 4 mts (radio)

Bate de Baseball

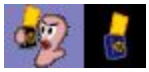
Al activarse el arma, el gusano batea lanzando a todo aquel cercano. La dirección de los gusanos lanzados dependerá de la dirección del bateo.

Arma con mira: sí
Combate cuerpo a cuerpo: sí
Disparo con potencia variable: no
Cuenta regresiva: no
Teledirigido: no
Trayectoria afectada por el viento: no
Municiones: infinitas
Daño: 10 pts



Ataque Aereo

Caen del cielo 6 misiles hacia el objetivo marcado por el jugador cada uno de ellos explotando independientemente. Los misiles pueden impactar antes si se encuentran en su trayectoria con algún otro objeto.
Arma con mira: no
Combate cuerpo a cuerpo: no
Disparo con potencia variable: no
Cuenta regresiva: no
Teledirigido: sí
Trayectoria afectada por el viento: sí
Municiones: 2
Daño (por cada misil): 40 pts (epicentro); 2 mts (radio)



Teletransportación

Al activarse la herramienta, el gusano puede teletransportarse a cualquier parte del escenario (salvo el interior de una viga).
Arma con mira: no
Combate cuerpo a cuerpo: no
Disparo con potencia variable: no
Cuenta regresiva: no
Teledirigido: sí
Trayectoria afectada por el viento: no
Municiones: infinitas
Daño: ninguno

Movimientos de los gusanos

Los gusanos pueden desplazarse con las flechas izquierda y derecha a una velocidad pequeña de 0.2 mts/seg. Presionando Enter, el gusano puede dar un salto hacia adelante de 1 mts y de 0.5 mts de alto. Presionando Retroceso, el gusano de una vuelta hacia atrás de 0.2 mts y de 1.2 mts de alto.

Un gusano puede caer de una altura de 2 mts sin sufrir daño alguno y podrá continuar moviéndose. Distancias mayores producen un daño proporcional a la altura (1 punto por metro) con un máximo de 25

puntos.

Un gusano no puede empujar a otro al desplazarse o saltar. De hecho el otro gusano permanece quieto en el lugar.

Gusanos y explosiones

Un gusano saldrá disparado por efecto de una explosión si este se encuentra en su rango.

Al salir disparado, el gusano sufre una fuerza y adquiere velocidad: la dirección y magnitud dependen de la distancia y dirección del epicentro de la explosión. Vease la seccion "Daño".

En su vuelo, el gusano puede rebotar contra las paredes.

Turnos y rondas

El juego es por turnos: un jugador puede tener múltiples gusanos pero el turno del jugador sólo le permitirá interactuar y controlar a uno de ellos, siendo el gusano elegido automáticamente de manera cíclica.

El turno del jugador termina si alguna de las siguientes condiciones se cumple:

- si el gusano activo sufre algún daño.
- si el gusano dispara o hace uso de una herramienta.
- si pasan más de 60 segundos.

Si el gusano activo dispara o hace uso de una herramienta, el jugador dispone de 3 segundos adicionales para continuar moviendo al gusano.

Transcurridos estos segundos el turno del jugador termina, pero el turno del siguiente jugador no arranca inmediatamente sino que el juego debe esperar a que todos los proyectiles impacten, las explosiones sucedan y los gusanos queden quietos (posiblemente se moverán debido a alguna explosión).

Recién en ese instante, el juego le da control al siguiente jugador y se da por comenzado su turno.

Una vez que todos los jugadores jugaron un turno cada uno, se termina una ronda.

Inmediatamente comienza la siguiente ronda volviendo cada jugador a jugar un nuevo turno en el mismo orden de la ronda anterior.

Si un jugador perdió, este es salteado en la ronda. El jugador que perdió puede seguir viendo la partida pero no puede hacer nada más en ella.

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 gusano activo y seguirlo a medida que se desplaza.

El jugador puede mover la cámara con el mouse para poder ver otras partes del escenario pero la cámara volverá a enfocarse en el gusano en cuanto este se mueva o intente disparar.

Cuando el gusano activo dispara un proyectil, la cámara debe seguir al proyectil hasta que este impacte.

Tras el impacto, la cámara debe seguir la posición de algún gusano que se esté moviendo (posiblemente debido al impacto).

Cuando este gusano termine de moverse (o se muera), la cámara debe elegir a otro gusano en movimiento y continuar así hasta que ninguno se mueva.

Animaciones

El juego no debe mostrar imágenes estáticas sino pequeñas animaciones para darle mayor realismo [3]:

- El movimiento de los gusanos: cuando se desplazan, saltan, vuelan entre otros.
- 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[4]:

- Cuando hay disparos.
- Cuando hay una explosión.
- Cuando un gusano da un salto
- Cuando un gusano 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

Se debe mostrar la parte del mapa que el jugador está viendo permitiéndole moverse al desplazar el mouse como lo explicado en la sección Cámara.

Cada jugador tendrá un color asociado.

La vida de cada gusano debe mostrarse encima de cada uno de ellos y debe mostrarse con el mismo color del jugador para poder identificar qué gusano es de quien.

También debe poderse ver la suma total de vidas de los gusanos por cada jugador.

El jugador deberá poder seleccionar qué arma o herramienta usar en su turno.

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 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.

Al momento de iniciar la partida el servidor deberá asignar de forma aleatoria a cada jugador los gusanos disponibles en el escenario.

En caso de que la cantidad de gusanos no sea divisible por la cantidad de jugadores, los gusanos de los jugadores con menos cantidad de gusanos tendrán un +25 puntos de vida para compensar.

Todos los atributos de los gusanos (velocidad, altura de salto, etc), de las armas (daño, rango, municiones, etc) y cualquier otro parámetro 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:

- Elegir el fondo de pantalla.
- Colocar las vigas y gusanos.
- Definir qué armas y herramientas pueden usarse y la cantidad de municiones de ellas (sobre escribiendo los valores por defecto.
- Vida inicial de los gusanos.

Además deberá realizar chequeos de consistencia para evitar errores por parte del diseñador como un escenario sin gusanos.

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 (01/05/2018)	- 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*). - Determinar a partir del draft qué mensajes se necesitarán en el protocolo de comunicación.
Semana 2 (08/05/2018)	- Modelado del escenario, los gusanos, los disparos y explosiones.	- Dibujado del escenario. - Dibujado de animaciones (gusanos, disparos, explosiones) - Controles por teclado.	- Editor, creación de escenarios.
Semana 3 (15/05/2018)	- Finalización del modelado del juego.	- Finalización de la gráfica del escenario. - Controles por mouse. - Scrolling.	- Finalización del editor. - Carga y guardado de los escenarios (archivos).
Semana 4 (22/05/2018)	- Lógica para la creación de partidas multijugador y múltiples partidas.	- Interfaz gráfica al seleccionar un arma o herramienta.	- Implementación del sistema de sonidos y música.
Semana 5 (29/05/2018)	- Finalización de la implementación multijugador.	- Interfaz para la conexión con el servidor, elegir un nivel, crear/unirse a una partida.	- Creación de escenarios de prueba y preparación de la demo. - Pantallas de victoria y derrota.
Semana 6 (05/06/2018)	- Testing - Correcciones y *tuning* del Servidor - Documentación	- Testing - Correcciones y *tuning* del Cliente - Documentación	- Testing - Correcciones y *tuning* del Editor - Documentación
Preentrega el 12/06/2018			
Semana 7 (12/06/2018)	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación
Semana 8 (19/06/2018)	- Testing - Correcciones sobre Preentrega - Armado del entregable	- Testing - Correcciones sobre Preentrega - Armado del entregable	- Testing - Correcciones sobre Preentrega - Armado del entregable
Entrega el 26/06/2018			

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. 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. 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. Entrega de uno o varios escenarios con la suficiente diversidad de elementos a tal fin que sea fácil mostrar las funcionalidades implementadas.
6. 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] Worms: https://es.wikipedia.org/wiki/Worms_(serie)
[2] Box2D: http://box2d.org/manual.pdf
[3] Sprites: https://www.spriters-resource.com/pc_computer/wormsgeddon/sheet/13597/?source=genre
[4] Sonidos: https://www.youtube.com/watch?v=F9YMb0M89DI
[5] Musica ambiente:
https://www.youtube.com/watch?v=Yc-M2OVIkUs&index=2&list=PL2CAB64D6B77AD3ED
[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