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 del módulo 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 para la carga de los mapas dentro del juego como para futuras ediciones.

Agustín se encargó del módulo cliente. En este caso debió crear toda la interfaz de usuario que permite visualizar el juego, incluyendo animaciones y sonidos. Creó 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 del módulo 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 mensajes que se pueden enviar a través de la red. Por último, agregó toda la funcionalidad 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 los elementos básicos del juego, agregando objetos al mundo y caídas 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.
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 agregó la lógica para permitir partidas con muchos jugadores y se implementó el soporte para 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.