FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Wolfenstein 3D[1]

Manual de Proyecto

[75.42] Taller de Programación I
Segundo cuatrimestre de 2020

Chiara Bauni - 102981

Damian Ganopolsky - 101168

[1]Link a repositorio https://github.com/DamianGanopolsky/TDPWolfenstein

Índice

1. Enunciado **2**

2. División de tareas **2**

3. Evolución del Proyecto **2**
3.1. Cronograma del servidor 2
3.2. Cronograma del cliente 2
3.3. Cronograma del editor 3

4. Inconvenientes encontrados **3**
4.1. Inconvenientes generales 3
4.2. Inconvenientes técnicos 4

5. Análisis de puntos pendientes **4**

6. Herramientas **4**

7. Conclusiones **5**
7.1. Conclusiones personales 5

1. Enunciado

El enunciado inicial del proyecto para 3 integrantes se encuentra en la carpeta Docs del repositorio.

2. División de tareas

Inicialmente la tarea fue dividida según la organización propuesta por el enunciado, pero por ciertos inconvenientes esto cambió en el desarrollo del TP.

Chiara Bauni se dedicó al diseño y armado del servidor, con la lógica del modelo y la comunicación con múltiples clientes en simultáneo.

Damian Ganopolsky se dedicó al diseño e implementación del editor, y al cargado de mapas y configuración en formato YAML en las distintas aplicaciones. Además, se dedicó al diseño de la interfaz gráfica(sonido, música, animaciones, pantallas de interacción con el usuario), junto con el modelo completo del cliente y su conexión con el servidor.

A su vez, hubo un tercer integrante en el grupo que se había dedicado al modelado de rayos para el ray casting de las paredes, y al cargado de la UI inicial.

3. Evolución del Proyecto

3.1. Cronograma del servidor

Para el desarrollo del servidor, se decidió seguir el siguiente cronograma

- Semana 1: Se desarrolla el diseño general de las clases que intervienen en el juego, identificación de clases, colas e hilos necesarios para la conexión el server. Comienzo del desarrollo del modelo.
- Semana 2: Desarrollo de la lógica del juego y se crea el esqueleto de las entidades que intervienen en la comunicación entre el cliente y el servidor.
- Semana 3: Definición del protocolo e implementación de la lógica de las partidas y la lógica de ataque.
- Semana 4: Implementación de la llegada de distintos comandos. Conexión de múltiples jugadores, testeo de lógica y conexión con los clientes.
- Semana 5: Primer entrega: Agregado de pre_game, post_game y login. Lógica de daño, testeo y corrección de errores.
- Semana 6: Documentación, se agregan las correcciones marcadas. Se corrigen los leaks.
- Semana 7: Entrega final. Documentación final, correcciones, arreglo de errores.

Este fue el cronograma que nos hubiese gustado seguir sin embargo la mayoría de las tareas de las semanas 3 y 4 fueron realizadas durante el receso de Enero, lo cual nos dio tiempo para finalizar el trabajo. A pesar de que también, fue durante ese tiempo que el tercer integrante decidió abandonar la materia por lo cual la carga se volvió más pesada para ambos integrantes restantes.

3.2. Cronograma del cliente

- Semana 1, Semana 2, Semana 3,Semana 4: Ray casting de las paredes, lógica del renderizado usando el ray casting. Visualización de un enemigo. Visualización de la interfaz básica, movimiento y rotación del personaje a través de un mapa construido en base a operaciones de módulo. Animación del disparo con pistola.

- Semana anterior a la 5 y durante la misma: Cargado de los distintos personajes y visualización en base a sus ángulos, cargado de todos los objetos, cambio en el uso del handler(deja de estar manejado por otro hilo). Reproducción de música y sonidos. Animaciones de disparos con todas las armas. Carga de mapas dinámicos no hardcodeados, en formato yaml. Configuracion basica del cliente en formato yaml. Agregado de números restantes en la interfaz gráfica, cambios en su renderización para que sean en base a texto para la modificación dinámica de dichos números. Agregado de hilos y colas para la comunicación con el servidor. Manejo de eventos básicos recibidos por el servidor. Visualización de animaciones de disparos producidos por enemigos.
- Semana 6: Animaciones de muertes, agregado de más sonidos, corrección de errores en la comunicación con el servidor. Agregado de todos los eventos restantes del protocolo de comunicación con el servidor, para el actualizado del modelo del cliente. Pantalla en caso de perder y que continue el juego para los otros jugadores.
- Semana 7: Pantalla de espera y pantalla final de estadísticas. Corrección y mejoras en el uso del constant rate loop, optimización de función renderizadora para que deje de recibir como parámetro el mapa completo en cada iteración de un for. Animaciones básicas de movimiento de los distintos personajes. Recepción de más parámetros por configuración para la optimización del uso de la CPU.

3.3. Cronograma del editor

- Semana 1: Draft del editor, planteo de clases básicas del mismo. Reproducción de sonidos y música, visualización de texto en pantalla.
- Semana 2: Diseño y carga de imágenes del editor, actualización de la aplicación en base a eventos, drag & drop de items básicos.
- Semana 3: Diseño completo de la interfaz, scrolling del mapa con WASD, optimizaciones en el recorrido del mapa.
- Semana 4: Agregado de Click & drag para el editado de objetos(útil para las paredes), diseño de pantallas final e inicial, lógica para cargar mapas dinámicos y modificarlos, parametrización del mapa a cargar. Configuración de resolución por archivos yaml.
- Previo a semana 5: Agregado de archivos yaml para la apertura y guardado de mapas.

4. Inconvenientes encontrados

4.1. Inconvenientes generales

El principal inconveniente con el que nos encontramos fue un par de semanas antes de la primera entrega del proyecto. El integrante del grupo encargado del desarrollo del cliente, al tener ciertos inconvenientes en el mes de diciembre, se había comprometido a seguir y terminar sus tareas en Enero, pero, sufrió ciertos problemas personales en dicho mes y no pudo continuar con el trabajo práctico, abandonando la materia.

Luego de que abandonara la materia, se vio que al cliente le faltaba mucho desarrollo para ser terminado, ya que, únicamente se contaba con la visualización de un personaje y paredes en ubicaciones “hardcodeadas”. Además, se contaba con una interfaz gráfica inicial para el mostrado de vidas, y otras estadísticas, pero los números mostrados también estaban puestos de manera fija y debido a que se generaban mediante una concatenación de imágenes de números, provocaban varios problemas al ser cambiados dinámicamente.

4.2. Inconvenientes técnicos

Al momento de conectar las aplicaciones cliente y servidor se tuvieron ciertos problemas debido a que no se había hablado exhaustivamente de ciertos requerimientos en los envíos de mensajes del protocolo. Para solucionar dichos problemas hicimos meets diarios para asegurarnos que ambos entendiéramos parte del funcionamiento de la otra aplicación y que los mensajes enviados entre las aplicaciones sean completamente factibles tanto para el cliente como para el servidor. Además, al unir ambas aplicaciones se pudieron notar ciertos errores en el modelo que anteriormente no se pudieron testear y mediante la visualización de la aplicación conjunta se pudieron ver y solucionar.

5. Análisis de puntos pendientes

- Variedad de paredes y elementos de decoración que no se priorizaron en el armado de la interfaz gráfica del cliente
- Puertas, si bien la lógica está presente en el servidor, no se pueden mostrar correctamente con sus animaciones debido a cierta falta de flexibilidad en la lógica del renderizado del ray casting.
- Optimización de los tiempos de renderizado, si bien con respecto a la primera entrega se optimizaron dichos tiempos(que causan el "cuello de botella" en el cliente) en alrededor de un 30 %, una optimización más profunda implica un cambio de raíz en el ray casting que conllevaria cambiar gran parte de dicha lógica. En el listado de correcciones de la primera entrega hechas se encuentra una explicación detallada de cómo se optimizó el uso de CPU.
- Vista del juego en fullscreen
- En el caso de el instalador, no logramos que se instale en el Binarie Path que establecimos. Por falta de tiempo para finalizar el mismo, optamos por que el usuario use el instalador pero que lo corra dentro de la carpeta build para no tener inconvenientes con el lugar en el que se instalan los paquetes.

6. Herramientas

- Control de versiones: git, además de la plataforma GitHub.
- Para edición del código fuente, se utilizo Visual Studio Code
- Para verificar y solucionar problemas de memoria, se usó Valgrind
- Para el diseño de las interfaces gráficas se utilizó SDL2, junto con sus librerias sdl2_image, sdl2_ttf, sdl2_mixer
- Para la edición de imágenes usadas en el editor y el cliente, se utilizaron Pix, Gimp, Paint y algunas herramientas en línea
- Para el armado de archivos de configuración y para la serialización de mapas se usó el formato YAML, particularmente usamos la librería para C++ "yaml-cpp"
- Para el proceso de compilación se usó CMake
- Para la confección de los diagramas UML se usaron PlantUml y StarUml

7. Conclusiones

En el transcurso del proyecto aprendimos a usar muchas herramientas, como también se aprendieron varios conceptos para el armado de una aplicación completa distribuida. Previamente, en los trabajos prácticos de la materia aprendimos varias herramientas que nos ayudaron a lo largo del desarrollo del proyecto y de no haberlas aprendido se nos hubiera hecho muy difícil entender cómo resolver ciertos problemas que se nos fueron presentando. A su vez, el protocolo de comunicación entre el servidor y cliente nos permitió entender cómo se manejan las aplicaciones distribuidas, ya que, en los trabajos prácticos realizados con anterioridad los mensajes enviados entre las aplicaciones eran bastante simples y no modelaban una aplicación de mayor dificultad como lo fue esta.

Como conclusión final del proyecto, se puede decir que terminar teniendo las aplicaciones funcionando correctamente fue muy gratificante luego del gran esfuerzo hecho en el desarrollo del trabajo práctico y de los inconvenientes que se fueron presentando en el transcurso del mismo.

7.1. Conclusiones personales

Chiara Bauni

La gran parte de mi trabajo estuvo puesto en el server y su comunicación con el cliente. Por esto siento que aprendí, frente a un trabajo de gran tamaño, a desarrollar la lógica de un juego y lograr desarrollar una comunicación con múltiples cliente. Hacer un servidor de tal complejidad fue desafiante pero muy enriquecedor.

Damian Ganopolsky

Como conclusión, puedo decir que aprendí muchas herramientas y conceptos que antes del desarrollo del proyecto no conocía o había aplicado. En particular, nunca había usado archivos de serialización como lo son los yaml, y me permitieron entender conceptos que tenía únicamente de manera teórica, como lo son la persistencia de datos.

Además, tampoco había usado librerías para el renderizado de imágenes, por lo que luego de resolver ciertas dudas con el manejo de las librerías SDL, me fue muy provechoso poder ver gráficamente los cambios que se hacían. Por último, lo que más me gustó fue emplear trigonometría para poder determinar el daño con los disparos, entre otras cosas.