Taller de Programación I (75.42 / 95.08)

Jazz Jackrabbit 2

Trabajo Práctico Grupal - Manual del proyecto

Curso Veiga - Grupo 7

Integrantes

Avalos, Victoria	108434
Castro Martinez, José Ignacio	106957
Diem, Walter Gabriel	105618

Índice

División de tareas

Las tareas fueron divididas de la siguiente manera:

	AVALOS	CASTRO	DIEM
Semana 1 07/05 - 14/05	• Se plantea la estructura del cliente • Primer implementación del Renderer	• Se plantea la estructura del servidor en general	• Pruebas de concepto de Qt • Comienzo de armado de scripts de building y el CMake • Armado de manual de usuario
Semana 2 14/05 - 28/05	• Avance de la estructura del cliente • Planteo de las comunicaciones del Lobby, separándolas del resto del cliente.	• Se plantea la estructura del engine (solamente jugadores) y definiciones naive de las clases • Se arranca con la estructura base del servidor: Accepter, Protocolo, Game, Sender y Receiver	• Construcción de la ventana del lobby con Qt • Pruebas de concepto de SDL2 • Armado de scripts de building • Comienzo de armado del Vagrantfile • Armado de manual de usuario
Semana 3 28/05 - 04/06	• Fixes del cliente y del protocolo del cliente, agregando manejo de errores y ajustando a las nuevas necesidades • Lobby finalizado, con su propio protocolo y estructura secuencial. • Integración del el cliente con el Renderer	• Estructura global del servidor, se realizan correcciones sobre el tratamiento de la comunicación en el lobby • Se avanza sobre el proyecto en generar: monitor de juegos, monitor de partida y wrapper del juego	• Retoques al lobby • Integración del protocolo con el lobby • Armado de assets y spritesheets • Comienzo de construcción del motor de renderizado del juego • Integración del yaml parser • Armado del Vagrantfile y scripts de provisioning • Comienzo de integración del protocolo para el juego • Armado de manual de usuario
Semana 4 04/06 - 11/06	• Establecimiento de la división	• Se detecta race conditions en el	• Construcción del renderizado del

	de responsabilidades en el engine. ● Avance con la clase BasePlayer. ● Comienzo a implementar el sistema de colisiones con Rectángulos ● Avance con las armas ● Avance con los estados	abordaje del servidor (Hilo sender), se realiza la corrección y se procede a la integración ● Se realizan las definiciones correspondientes sobre los Protocolos cliente - servidor ● Se agrega el procesamiento de comandos para poder comunicar jugadores que se registran e inicializar la partida	mapa, el movimiento de la cámara y personajes jugables ● Armado de spritesheets ● Integración completa con el protocolo del juego
Semana 5 11/06 - 18/06	● Integro el engine con el servidor ● Finalizo con las colisiones y la gravedad ● Implemento disparos y balas ● Implemento ítems ● Implemento el sistema de respawn del jugador	● Refactor sobre el arranque del juego una vez se conecta la cantidad necesaria de jugadores ● Se continua con la integración ● Trabajo sobre el graceful shutdown ● Manejo de deadlocks en el cierre ● Migración raw pointers to smart pointers ● Debug de los casos de cierre incorrecto	● Construcción del código de animaciones para todos las entidades ● Armado de spritesheets, sonidos y música ● Construcción del motor de sonido y música ● Optimización del motor gráfico ● Corrección de bugs de renderizado ● Agregado de segundo escenario
Semana 6 18/06 - 25/06	● Armado de documentación ● Implementación de las últimas features restantes del engine: enemigos y ataques especiales ● Posicionamiento de ítems y enemigos ● Finalizo que se efectuen los daños a jugadores y enemigos	● Armado de documentación ● Construcción de test ● Más debug de los casos del jugador ● Documentación técnica del server	● Corrección de bugs de sonido y animaciones ● Optimización del motor de sonido ● Construcción de movimiento en tiles diagonales ● Armado de documentación

El plan inicial contemplaba tener la integración completa del juego con el protocolo del juego una semana antes de lo realizado, es decir, en la semana 3 en lugar de la semana 4. Eso permitiría para ese momento visualizar las animaciones ya construidas hasta ese momento, y el resto de las features. Sin embargo, se logró recuperar el tiempo más adelante.

Herramientas utilizadas

De desarrollador

Los IDEs utilizados fueron Visual Studio Code y CLion. Para el proceso de debugging se utilizó el debugger incorporado en VSCode y un archivo de configuración para el mismo.
Para el trabajo colaborativo se apalancar el uso de pre-commit para estandarizar el código antes de realizar un commit al repositorio. Se tiene configurados hooks de commitzen, cpplint y clang. Estos hooks también corren en el repositorio remoto en Github mediante el uso de Github Actions.
Para el armado de spritesheets de tamaño custom partiendo de las spritesheets originales se utilizó Photoshop y TexturePacker para cortar solamente las animaciones relevantes y generar un yaml con coordenadas de los nuevos sprites.
Para el armado de assets de audio (efectos de sonido y música) se utilizó Audacity.

Para la construcción del TP

Se utilizaron las siguientes bibliotecas para construir el juego en el lenguaje de programación C++ respetando el estándar de C++17:

- Qt (en su versión 5): para el desarrollo de la ventana del lobby del juego.
  Documentación: https://doc.qt.io/qt-5/
- SDL2 con el wrapper en c++ libSDL2pp: para el desarrollo de la ventana del juego, el motor gráfico y el motor de sonidos.
  Documentación: https://github.com/libSDL2pp/libSDL2pp, https://lazyfoo.net/tutorials/SDL
- yaml-cpp (en su versión 0.8): para permitir hacer archivos de configuración en un formato estándar, poder parsearlos y cargarlos.
  Documentación: https://github.com/jbeder/yaml-cpp
- Shell scripting: para armar scripts ejecutables que permitan instalar librerías, buildear y correr el proyecto.
- Vagrant: para crear la definición de una máquina virtual de VirtualBox que pueda instalar todo lo necesario para buildear y ejecutar el juego en un entorno portátil, sin depender de las dependencias de los sistemas operativos usados durante el desarrollo.
  https://developer.hashicorp.com/vagrant/docs/vagrantfile

Experiencia de desarrollo

Puntos retadores

Algunos puntos del juego que presentaron dificultades pero pudimos superar fueron:

- Sistema de carga de spritesheets y armado de los sprites.

- Manejo de múltiples partidas en simultáneo con un manejo de sockets y threads adecuado.

Alcance

Se logró llegar al alcance completo del TP deifnido en la consigna. No se desarrolló el editor de niveles ya que el grupo consta de 3 personas y esa feature era para grupos de 4 integrantes.

Errores conocidos

- Al momento de desconexión de clientes se imprime un mensaje de error en la consola del servidor. Esto se puede solucionar aplicando una mejora en el manejo de errores.

Mejoras posibles

- Mejora del protocolo del juego (envío y recepción de snapshots): en lugar de enviar un struct packed como se está haciendo en este momento, se puede enviar y recibir cada componente de la snapshot por separado. De esta manera se hace un mejor uso de la red.
- Responsabilidad de cargar las coordenadas del mapa: se tiene un archivo YAML con las coordenadas de las tiles del mapa que tanto el cliente como el servidor cargan. Una manera más óptima de manejar esto sería que el servidor cargue las coordenadas y se las envíe a los clientes.