FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Taller de Programación I (TA 045)
Cátedra Veiga

Duck Game

Trabajo Práctico Grupal - Manual de Proyecto
Grupo 14

Integrantes:
- Nicolás Ezequiel Grüner - 110835
- Victoria Fernandez Delgado - 110545
- Bautista Boeri - 110898
- Franco Daniel Capra - 99642

División de tareas

Al comienzo decidimos que cada uno se encargaría de un aspecto del trabajo práctico.
Nicolas: Interfaz gráfica, todo lo relacionado a SDL.
Bautista: Editor y parte gráfica del lobby.
Victoria: Lógica del juego
Franco: Protocolo, threads y comunicación entre cliente y servidor

Finalmente, todos terminamos involucrándonos en distintos aspectos del trabajo práctico y no solo en el asignado al comienzo. La lógica del juego fue la parte con mayor carga del trabajo y , por ende, la última en ser terminada.

Semana 1:

- Primer acercamiento a SDL. Tutoriales de Lazy Foo. Modelo de prueba de un pato moviéndose por la pantalla.
- Implementación del movimiento del pato
- Primer acercamiento a QT y QT Creator. Estructuración básica del lobby
- Comunicación entre cliente y servidor para 2 jugadores.

Semana 2:

- Uso de SDL2pp en el proyecto, primeras ideas. Búsqueda de sprites para usar en el juego.
- Creación de gamemap con los patos. Integración con el gameloop
- Incorporación de música al lobby
- Finalización de la estructura del lobby y comienzo de la estructura del editor.
- Protocolo para el lobby incluyendo la creación de partidas y la unión a existentes.
- Protocolo preliminar para un broadcast.

Semana 3:

- Se puede ver el pato en la pantalla y se agregan más sprites de las armas.
- Elaboración del DTO player, que contiene la información necesaria para graficar el pato.

- Configuraciones en el YAML para el pato y las armas.
- Base para las armas con clase Weapon, algunas armas implementadas.
- Incorporación del protocolo y servidor en el lobby.
- Iteración sobre el protocolo del broadcast para el movimiento de los patos.

Semana 4:

- Incorporación de clases para dibujar la armadura, el casco, las balas, el arma y las cajas.
- Adición de algunos sonidos.
- Elaboración de DTOs para el arma, balas, casco, caja y armadura
- Nuevos DTOs para el envío y lectura del mapa.
- Incorporación de una clase que permite dibujar el mapa.
- Desde la parte gráfica, está todo preparado para que se dibuje el pato.
- Nuevos eventos GameOver y Score para el sistema de rondas.
- La interfaz gráfica ya está lista para mostrar los puntajes y la pantalla de victoria.
- Implementación de las armas desde la lógica.
- Incorporación de clase para las balas.
- Se pueden colocar tiles en el editor
- Se puede guardar y cargar mapas en el editor
- Se puede cambiar el fondo en el editor
- Se puede borrar todo el contenido de la grilla como de una posición particular en el editor
- Primera versión de interpretación de los mapas en el yaml y en la lógica
- Creación de tests para el protocolo.

Semana 5:

- Se conecta la lógica con la interfaz gŕafica. Ya se puede ver al pato moviéndose, recibiendo las actualizaciones del servidor. Corrección de detalles gráficos y lógicos.
- Soporte del protocolo para todas las acciones del pato y para las cajas.
- Incorporación de todas las armas del juego en la parte gráfica.

- Fix a algunos sprites y al contorno de lo dibujado.
- Se puede ver el pato con armadura
- Comienzo en la elaboración de throwables como granadas y bananas
- Implementación de explosiones.
- Fixes sobre algunas restricciones a la hora de unirse a las partidas.
- Agregado de colores en el lobby a los jugadores.

Semana 6:

- Se puede dibujar el pato con casco. El pato puede morir. Se pueden dibujar las armas/cascos/armaduras que aparezcan en el piso. Se dibujan las granadas lanzadas y sus explosiones las cuales reproducen un sonido. Ajustes visuales sobre el pato.
- Adición de la cámara en el juego.
- Se agregan más sprites y se arreglan algunos sprites del pato.
- El protocolo ahora soporta: explosiones, balas, jugadores, cajas, ítems que aparezcan y arrojables.
- Implementación de la lógica para el transcurso de rondas y el sistema de victoria.
- Se implementan las colisiones verticales con las plataformas y las del pato con las balas.
- Implementación de las cajas, su destrucción y pueden aparecer items en el piso.
- Se termina de implementar las granadas y bananas
- Separación del uso del yaml-cpp en otra clase en el editor
- Fixes sobre el comportamiento del lobby con ciertos eventos de QT.

Semana 7:

- Ajustes de la lógica para la sincronización con la parte gráfica: las balas salen correctamente, las cajas reproducen el ruido cuando se rompen y dejan de tener hitbox.
- Ahora por cada ronda se puede ver un nuevo mapa.
- Incorporación de los cheats.

- Los puntajes y la pantalla de victoria ya no están hardcodeados desde lo gráfico y se reciben del server.
- Cambios menores a armas: la escopeta recarga, la Ak-47 y la magnum tienen dispersión.
- Corremos los pre commits.
- Cambios en la manera de recibir las estructuras de las plataformas del mapa en la lógica. Se implementan las colisiones horizontales del pato con las plataformas.
- El pato puede agarrar y soltar armas y/o armaduras.

Herramientas utilizadas

- IDEs: VSCode y CLion
- SDL2pp.
- Gimp.
- Tutorial Lazy Foo para SDL.
- QT6
- QT6 Multimedia
- QTCreator
- Tiburoncin para la comunicación.
- Bpytop para observar el comportamiento de los threads una vez cerrado el juego.

Puntos más problemáticos

Durante el proyecto, la mayor dificultad en general fue coordinar para juntar las distintas partes para componer el juego final. La primera conexión entre el cliente y el servidor fue el punto que más costó.

De parte de la interfaz gráfica, lo más complicado fueron las cuestiones de diseño, sobre cómo plantear cada clase y la estructura general para que sea compatible con SDL.

Por parte de la lógica, las colisiones entre los patos y las estructuras presentó muchas dificultades. También, el comportamiento de la Laser

Rifle al rebotar contra las superficies fue de lo más difícil de solucionar e implementar correctamente.

En el lado del editor, se dificultó el guardado de los mapas de forma correcta para que la lógica pueda interpretarlos luego. Agrupar los tiles que se encuentran consecutivamente en horizontal y en vertical presentó algunos inconvenientes.

Por último, del lado del protocolo y sockets, la mayoría de las complicaciones se presentaron en el envío o lectura incorrecta de los mensajes enviados entre clientes y servidor. El debuggeo de estos errores fue muy costoso a nivel tiempo ya que los errores no eran evidentes y requirieron debuggear múltiples threads y acciones desde el servidor hacia el cliente o viceversa.

Errores conocidos

- Si el pato colisiona con una estructura cuando está siendo desplazado hacia atrás por el recoil, este la traspasa.
- La laser rifle tenía comportamientos muy extraños al disparar hacia arriba, por lo que decidimos no dejar que se pueda disparar arriba con este arma.
- Si el pato se arroja de la plataforma y choca contra ella durante la caída, puede suceder que se teletransporte encima de ella nuevamente.
- El procesamiento del mapa en la lógica genera que haya estructuras que se interpretan más de una vez, aunque no afecta en el funcionamiento, pues puede haber repetidos pero no faltantes.
- La colisión de las bananas como de las granadas en el eje horizontal es ineficiente en términos prácticos, si se choca con una pared, esta sube hasta arriba de todo de la pared y sigue su recorrido desde esa posición.

¿Qué cambiaríamos?

Intentaríamos tener una mejor comunicación entre el grupo, de forma que hubiéramos podido coordinar cuanto antes juntar las distintas partes del proyecto para así avanzar y poder probar las implementaciones a medida que programabamos.
Por otra parte, dividirnos las tareas podría haber sido de mucha ayuda para no pisarnos entre los integrantes del grupo y lograr ser más eficientes.