Taller de Programación I (75.42 - 95.08)

Ejercicio Final - Portal

Manual de Proyecto

1er Cuatrimestre 2019

Integrantes	
Alumno	**Padrón**
Sebastián Ignacio Penna	98752
Jonathan Claudio Medina	100052

División de Tareas

El ejercicio final propuesto por la cátedra fue diseñado y diagramado (como se mostrará en las siguientes secciones) para grupos de 3 personas. Sin embargo, previo a la segunda semana (donde se comenzaría a desarrollar el código) quien sería la tercer integrante del grupo abandonó la cursada y como consecuencia el grupo quedaría con tan solo dos integrantes.

Por dicha razón, se nos indicó que aquellas tareas relacionadas al editor de escenarios no debían ser realizadas, ya que estas hubieran sido tarea de un tercer integrante. De ésta manera la división de tareas que se realizó fue:

- Servidor: Sebastian Penna;
- Cliente: Jonathan Medina.

Dentro de cada uno de estas divisiones se incluyen diversas tareas que serán desarrolladas a lo largo de éste informe.

Evolución del Proyecto

Cronograma Propuesto

	Alumno 1 **Servidor - Modelo**	**Alumno 2** **Cliente - Modelo**	**Alumno 3** **Cliente - Editor**
Semana 1 (30/04/2019)	- 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*). - Prueba de concepto con ffmpeg.
Semana 2 (07/05/2019)	- Escenario (bloques, rocas y otras cosas no dinámicas)	- Renderizado del escenario incluyendo la cámara.	- Edición de un escenario básico con solo objetos estáticos.
Semana 3 (14/05/2019)	- Chell (jugador), carga/descarga de rocas y bolas de energía y otros elementos dinámicos.	- Animación de los elementos dinámicos.	- Edición de un escenario incluyendo la locación de las Chells y del pastel.
Semana 4 (21/05/2019)	- Portales y física completa. Lógica de compuertas	- Finalización de la parte gráfica incluyendo la pin tool.	- Edición de las condiciones booleanas para cada compuerta.
Semana 5 (28/05/2019)	- Servidor multipartidas con partidas multijugador. Condiciones de victoria y derrota.	- Música y sonido. Pantallas de login, creación de partidas y de unirse a partidas.	- Captura de video de una partida: inicio y frenado a voluntad.
Semana 6 (04/06/2019)	- Testing - Correcciones y *tuning* del Servidor - Documentación	- Testing - Correcciones y *tuning* del Cliente - Documentación	- Testing - Correcciones y *tuning* del Editor - Documentación
Entrega el 11/06/2019			
Semana 7 (11/06/2019)	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación	- Testing y corrección de bugs - Documentación
Semana 8 (18/06/2019)	- Testing - Correcciones sobre la primer entrega - Armado del entregable	- Testing - Correcciones sobre primer entrega - Armado del entregable	- Testing - Correcciones sobre primer entrega - Armado del entregable
Reentrega el 25/06/2019			

Como se mencionó anteriormente, la columna respectiva al Alumno 3 (editor) no fue realizada dentro de este trabajo, a excepción de las tareas propuestas en la semana 5, es decir, la captura de video de una partida.

Cronograma Real

	Sebastian Penna **Servidor - Modelo**	**Jonathan Medina** **Client - Modelo**
Semana 1 (30/04/2019)	-Lectura documentación Box2D	- Lectura de documentación de SDL
Semana 2 (07/05/2019)	-Instalación Box2D y CppUnit -Creación primeras clases en base al enunciado	-Lectura de documentación de SDL. -Instalación de SDL
Semana 3 (14/05/2019)	-Primeras pruebas Box2D -Creación bloques (no diagonales) -Movimientos Chell	- Mostrar animaciones desplazándose por la pantalla
Semana 4 (21/05/2019)	-Transmisores y receptores energía -Bolas de Energía -Rocas -Desarrollo Protocolo	- Creación de la cámara - Mostrar bloques, ácido, y animaciones que se repiten - Creacion del WorldView
Semana 5 (28/05/2019)	-Parser Yaml -Botones y lógica Compuertas -Bloques diagonales -Desarrollo protocolo y comienzo traducción	- Arreglo de la cámara - Creación de la gate - Rocas - Creación de un fake server
Semana 6 (04/06/2019)	-Desarrollo del servidor multipartidas y protocolo -Traductor del protocolo	- Armado de recepción de objetos del servidor - Animaciones que cambian de estado - Correcciones de la vista
Semana 7 (11/06/2019)	-Puesta en funcionamiento servidor -GameLoop y creación de threads -Vinculación servidor-cliente -Suicidio y asesinato de chells -Testing -Corrección bugs juego y servidor	-Correcciones de la vista. -Renderizar en tiempo constante -Envío de señales al servidor -Recepción de objetos del servidor
Semana 8 (18/06/2019)	-Handshake servidor-cliente -Creación de portales y pin tool -Carga de roca -Condición victoria -Captura video -Testing -Corrección bugs juego y servidor -Documentación	- MusicPlayer - Creación de portales y pintool - Unirse a partidas - GUI para unirse a partidas mediante QT - Captura del video - Corrección bugs del juego - Documentación

Inconvenientes Encontrados

Nuevas Herramientas

En primer lugar una dificultad a lo largo del trabajo fue aprender nuevas bibliotecas a medida que se desarrollaba el mismo, como fue el caso de Box2D, SDL o yaml-cpp. Tener que resolver distintas dificultades en el código a medida que se estudiaban dichas bibliotecas fue un gran freno en un principio, dado que ninguno de los integrantes tenía experiencia previa en el desarrollo de un juego a este nivel, con múltiples partidas, multijugador y con simulación de física.

En el transcurso del trabajo y aprendiendo sobre la marcha se fueron encontrando mejores soluciones a código previamente creado (y en algunas ocasiones con varias fallas debido a la inexperiencia), lo que implicaba dedicar tiempo a realizar correcciones y evitar futuros problemas cuando se pusiera en funcionamiento el juego.

Al finalizar el trabajo, se decidió armar una pantalla de login, mediante QT, por lo que implicó aprender una librería en muy poco tiempo.

Desarrollo de Pruebas

Dado que como se mostró en el cronograma real no se vincularon servidor y cliente hasta un punto avanzado verificar que el modelo en desarrollo era correcto implicaba numerosas pruebas a través de cpp unit. Con la gran cantidad de features y objetos que podían interactuar en el juego la creación de pruebas para asegurar que al momento de unir ambas partes habría la menor cantidad de errores implicó gran parte del tiempo.

Del lado del cliente se decidió utilizar un “FakeServer” simulando el envío de mensajes del servidor al cliente, utilizando valores que el servidor podría llegar a enviar al cliente.

Gran cantidad de Threads

Dado que en ésta cursada se tuvieron las primeras experiencias con threads el desarrollo de un juego como el propuesto resultó de gran magnitud en el uso de los mismos. Se debió controlar rigurosamente la creación de los mismos y las distintas race conditions, dado que al tratarse de un servidor multi partidas con múltiples jugadores la cantidad de hilos que podrían generarse no tiene límite en principio.

Coordinar todos los hilos para interactuar correctamente tanto para el envío como recepción de datos, así como el desarrollo del gameloop fueron una de las tareas que más tiempo requirieron en el transcurso del trabajo.

Traducción de Posiciones

Un problema a resolver fue en cuestión a las posiciones. En primer lugar el modelo de Box2D trabaja en unidades métricas, mientras que en SDL al trabajar con la view del juego se utilizaban, como es de esperarse, pixeles. En primer lugar se debió buscar la correcta conversión de pixeles a metros, tanto para los movimientos como para el dibujo de los distintos objetos.

Pero además se debieron traducir las posiciones en cuanto a como estaban diseñadas, ya que Box2D nos entrega posiciones centrales y se debió decidir un punto en común dentro del protocolo para enviar dichos datos. También sobre SDL los ejes de posiciones se encuentran invertidos, implicando aún más trabajo en la conversión.

Jugabilidad

Uno de los puntos donde se intentó poner más énfasis en el testeo y correcciones fue en la jugabilidad del juego, valga la redundancia.

En nuestro caso se trató de la primera experiencia en el desarrollo de un juego con simulación física, animaciones y una orientación a eventos, por lo tanto conseguir que el desarrollo del juego resultara realista fue una de las dificultades más grandes encontradas.

Hacer movimientos que animarán correctamente, las colisiones con distintos cuerpos y la teletransportación fueron los principales puntos donde se encontraron conflictos para hacer del juego una experiencia agradable y jugable para el usuario, sabiendo que quedan aún más opciones de tuning para mejorarlo.

Leaks SDL en Valgrind

Al ejecutar el cliente sdl con valgrind resultó muy complejo evaluar su salida, dado la gran cantidad de errores que mostraba la biblioteca SDL, ajenos a nosotros. La ejecución del juego no llegaba a concluir como consecuencia de ésta problemática que nos encontramos al utilizar esta biblioteca.

Se intentó utilizar un archivo de supresiones, para poder evitar estos leaks, pero visto que no se ha logrado hacerlo funcionar, se decidió suspender esta tarea.

Sprites del juego

Los sprites del juego tuvieron que ser modificados un poco dado que no se ajustaban a las necesidades que se tenían, esto conllevo una inversión de tiempo no planeada. (Aprender a utilizar herramientas como GIMP)

Análisis de Puntos Pendientes

Editor

El punto más obvio en cuanto a quedar pendiente es el desarrollo del editor. La ausencia de un tercer integrante imposibilito el desarrollo del mismo, dado que encargarnos entre dos de la creación del mismo hubiese resultado una gran dificultad.

Movimientos de Chell

Uno de los puntos a mejorar dentro del juego es la forma en que la chell verifica y decide sus movimientos. En el juego desarrollado la Chell trabaja con variables booleanas para evaluar su estado y posición y en base a ellas definir su comportamiento. Esto se tornó aún más complicado a medida que se desarrollaron las colisiones con los distintos cuerpos que también pertenecen al mundo de Portal.

La forma de mejorar ésto sería a través de sensores, herramienta de Box2D que a través de fixtures utilizados como sensores permitirían detectar entre otras cosas que cara de la chell colisionó, si se encuentra en el aire, etc. Su ausencia se debe a que el movimiento de las chells fue uno de los primeros features en desarrollar, entonces el uso de sensores al recién comenzar a utilizar la biblioteca Box2D no resultaba tarea sencilla, quedando posteriormente el código en base a booleanos y siendo su cambio una tarea que conlleva tiempo, algo escaso en el desarrollo de éste trabajo.

Clase de la hereden objetos que se teletransportan

Una clase que se podría haber creado y por tiempos no se hizo podría haber sido una clase madre de la cual heredan aquellos objetos que tiene la capacidad de pasar a través de los portales. Tanto en la documentación técnica como en el código se podrá observar que hay ciertos métodos que tienen la misma firma e implementación cuando se debe producir una teletransportación.

Recortar imágenes enormes

Algo que hubiera sido muy fácil de hacer, pero por falta de tiempo no se realizó es recortar los sprites enormes, en pequeños Sprites, para así poder reducir considerablemente el consumo de memoria, y a demás poder soportar computadoras de menor recursos, que quizás no puedan contener en memoria.

Resize de ventana mostrar siempre lo mismo

Algo que también es sencillo de realizar, pero que demanda una cantidad considerable de tiempo es esto. La idea es mover el método de View que transforma metros a pixels a la cámara, y que la cámara conozca tanto los pixeles y los metros.

Entonces cada vez que tenía que dibujar un objeto en la vista, se transformaban todos datos a pixeles, y se mostraba al cliente.

La camara deberia recibir el ancho y alto de la ventana, que actualmente lo hace pero con otro propósito, poder mantener la chell en el medio de la pantalla en todo instante.

Colores Chell

Si bien el método ya estaba hecho en el TextureFactory (setColor) utilizando SDL_SetTextureColorMod, la idea era dependiendo de un id de la chell, devolverle un color diferente. Dado que se decidió cortar las imágenes de Chell a último momento, imposibilito poder llegar con este feature.

Una mejor forma de hacerlo mejor estéticamente es modificar mediante una herramienta de edición de imágenes como GIMP, a los Sprites, dado que el SDL_SetTextureColorMod afecta a toda la textura.

Poder comenzar y pausar grabación dentro de la partida

Uno de los puntos pendientes del trabajo es el desarrollo de la posibilidad de generar un video de la partida. Por el escaso tiempo y sabiendo que la tarea de trabajar con ffmpeg correspondía al tercer integrante del grupo, nos resultó complejo alcanzar a desarrollar éste aspecto.

Para que la función quede totalmente accesible para el usuario la idea hubiese sido colocar una botonera dentro de la partida donde el usuario pudiera decidir cuando comenzar y finalizar la grabación de la misma. Dadas las circunstancias se decidió brindar las herramientas para que el usuario pudiera decidir grabar por terminal, por fuera de la partida.

Pisar la bola de energía

Dado que se esperaba llegar a completar el refactor de la cámara, se decidió posponer este feature. Actualmente cada vez que llega una bola de energía, se pisa. Esto se podría solucionar muy fácilmente luego de realizar ese refactor, con la implementación actual, implicaría ensuciar un poco el código de la bola de energía.

Compuertas Nor/Or

Otro de los puntos pendientes en el juego es brindar la posibilidad de que existan compuertas cuya lógica no dependiera tán solo de la activación de varios botones, sino que

se pudiera también abrir la misma si uno u otro se encuentra activado o mismo si alguno **no** esta activado.

Dado que la lógica de compuertas pudo resolverse en el caso base *and* se decidió posponer ésta tarea para concluir con otros aspectos del trabajo que se consideraron más importantes, como el desarrolló del servidor multipartidas o alcanzar a completar todos los features que se requerían en la consigna.

Sin embargo, la implementación de ésta acción fue pensada y no se dejó el concepto sin considerar. Para desarrollar compuertas or y nor se debería agregar otro atributo a las gates dentro de los archivos YAML correspondientes a ambos casos, y modificar el existente por el caso específico de botones *and*. De ésta manera se tendrían tres listados: botones and, botones or y botones not. Los receivers no se consideran ya que una vez activados así permanecen, por lo tanto su uso siempre será un caso *and*.

Dentro del código de Gate se debiera agregar dos nuevos vectores que almacenan los datos obtenidos y en cada step se tendría que iterar sobre cada uno y verificar el estado de cada botón, exactamente como se realiza en éstos momentos. Además, durante la iteración se tendría que verificar si dicho botón corresponde a aquellos que se busca no estén activados, consultando si su id se encuentra dentro del vector de botones not.

Herramientas

Clion

Ambos alumnos trabajamos en el desarrollo del trabajo utilizando la IDE Clion, la cual consideramos proporciona grandes ventajas a la hora de programar en lenguaje C o C++. Además cuenta con una herramienta de debug interna visual que facilitó en varias ocasiones la detección de errores en el código, lo cual con GDB, por ejemplo, hubiese sido muy complejo.

Git

Para trabajar en conjunto se creó un repositorio en Git, donde ambos alumnos actualizamos continuamente el proyecto a medida que realizamos nuevos aportes o correcciones.

StarUml

Para la generación de diagramas UML se utilizó éste software, muy simple y veloz.

Valgrind

A través de Valgrind se comprobó que el servidor no generará leks ni tuviera accesos indebidos a memoria. Como se menciona previamente ejecutar valgrind con el cliente mostró problemas provenientes de SDL.

CppUnit

Para llevar a cabo unit testing en el modelo se decidió utilizar CppUnit, la cuál provee herramientas para un sencillo desarrollo y mantenimiento de numerosas pruebas que incrementarían con el tiempo y podrían sufrir modificaciones.

Conclusiones

El gran aporte de éste trabajo fue tener una primera oportunidad de desarrollar un trabajo de mayor magnitud a los que usualmente se tienen en las distintas asignaturas de la facultad. La división de tareas, organización en el tiempo y entregas semanales nos mostraron una idea de cómo se desarrolla un trabajo de estas características.

Tener que presentar semanalmente los avances necesarios para llegar a la última entrega de forma adecuada implican una presión y gran cantidad de horas dedicadas al trabajo para poder conseguir el objetivo de tener todo el programa funcional. El hecho de haber perdido un integrante al comienzo consideramos dificulto e hizo más difícil el trabajo, ya que a pesar de que se nos quitó ciertas tareas respectivas a un tercer alumno, el encargado del editor podría haber estado en un punto medio entre servidor y cliente, aportando ideas a ambos lados a medida que continuaba sus tareas. Siendo dos integrantes solamente cada uno estuvo enfocado en sus tareas, las cuales estaban totalmente aisladas y el único punto en común que compartían era la creación de un protocolo a respetar para el envío y recepción de datos.

Creemos que el aprendizaje de nuevas herramientas (como es el caso de Box2D, SDL o QT) a medida que se debía avanzar en el trabajo fue una tarea dificultosa, que requirió tiempo y dedicación para obtener las mejores soluciones a los distintos problemas. Quizás con más tiempo se podría haber generado un producto de mejor calidad, con mejor código y aún más jugabilidad.

Como conclusión, este trabajo fue una gran fuente de aprendizaje en trabajo conjunto teniendo que realizar constantes avances y debiendo presentar los mismos, no solo pensando en una corrección por parte de los profesores, sino generar un producto que sea descargable y utilizable para cualquier usuario.