

Argentum - Trabajo Final

Manual Técnico

Taller de Programación 1
Primer cuatrimestre de 2020

Alumno	Número de Padrón	Email
Ruiz, Francisco	99429	fran7ruiz9@gmail.com
Belosevich, Victor	97757	victor.belosevich94@gmail.com

Índice

1. Requerimientos de software	2
2. Descripción general	2
3. Cliente	2
3.1. Descripción general	2
3.1.1. Estructura	2
3.1.2. Estados del personaje	3
3.1.3. Wrappers de SDL	3
3.1.4. Renderización	3
3.2. Clases	3
3.3. Diagramas UML	8
4. Servidor	9
4.1. Descripción general	9
4.1.1. Estructura	9
4.1.2. Proceso para aceptar un nuevo cliente	10
4.1.3. Inicio del World y carga del Board	10
4.2. Clases	10
4.3. Diagramas UML	12
5. Protocolo de Comunicación	13
5.1. Mapa	14
5.2. PlayerInfo	14
5.3. GameObjects	15
5.4. InputInfo	15
5.5. Mensaje de Inicio	15
5.6. Mensaje de NPC	16
6. Código	16

1. Requerimientos de software

El juego fue desarrollado en Ubuntu 19.10 o mayor y puede correrse en distribuciones Debian. Para desarrollar y ejecutar el juego es necesario tener instalado las siguientes librerías:

- SDL2 versión 2.0.10
- SDL2-image versión 2.0.5
- SDL2-mixer versión 2.0.4
- SDL2-ttf versión 2.0.15

Para compilar e instalar el mismo, se utilizó la herramienta *Cmake* versión 3.14 o mayor. La otra librería externa utilizada es *RapidJson*, para la carga del mapa y el archivo de configuración con la versión 1.1.0, que ya se encuentra incluida en el proyecto. Por último, el compilador `g++` utilizado es versión 9.2.1 y `gdb`, utilizado para debugging, en su versión 8.3.

2. Descripción general

Argentum se dividió en 2 módulos, el servidor y el cliente.

El servidor es el encargado de levantar el mapa del mundo de Argentum, la configuración del juego, ambos desde archivos json, y mantener el estado del mismo todo el tiempo que este se esté ejecutando. Es capaz de manejar múltiples clientes y a medida que cada uno de ellos se va del juego, elimina a su personaje del mapa.

El cliente es una aplicación gráfica que le permite al usuario desenvolverse en las tierras de Argentum. Al iniciar, por líneas de comando indicará la raza, clase para su jugador, el host y el puerto al cual debe conectarse. Una vez establecida la conexión, el servidor enviará la información del mapa estático del juego y el primer "paquete" con la información propia del personaje del jugador.

Una vez que ya el jugador ha sido inicializado en el mundo, el servidor le irá notificando al cliente cada cierto intervalo de tiempo, el estado actual del mundo e incluso el estado actual del personaje del jugador. El cliente procesará estos mensajes recibidos y renderizará por pantalla todas las novedades que le fueron informadas.

El juego seguirá funcionando y los personajes de cada uno de los clientes estarán presentes en el mapa, siempre y cuando estos sigan jugando. Una vez que el cliente se desconecta y cierra su ventana, el servidor sacará del mapa a su personaje.

3. Cliente

3.1. Descripción general

Como mencionamos previamente, el cliente es una aplicación de interfaz gráfica. Esta interfaz fue desarrollada en SDL2, con el uso de sus librerías adicionales como SDL2-image, SDL2-mixer y SDL2-ttf.

El cliente recibe 4 parámetros de entrada desde la línea de comandos. Estos parámetros, nos permitirán conocer la raza, clase, host y puerto respectivamente.

Una vez, que se haya establecido la comunicación, el juego nos dará la bienvenida con una pantalla de "carga" que al presionar la tecla *ENTER* nos permitirá comenzar a jugar.

3.1.1. Estructura

El cliente se divide en 3 threads:

- Gameloop (render)
- Receiver
- Dispatcher

El hilo principal del cliente es el Gameloop que será el encargado de establecer la conexión inicial con el servidor. Esta conexión está basada en 3 pasos que se explicarán en la sección *Protocolo de Comunicación*.

Esta conexión se hace a través de un socket que estará encapsulado dentro de una clase llamada *Communication-Protocol*. Esta clase nos permite que varios hilos puedan trabajar con el mismo socket y que no quede en un estado inválido, ya que la implementación de nuestro *Socket* al pasarlo como referencia, se pierde el *ownership* del mismo.

Una vez que haya obtenido la información de su personaje y el mapa estático del mundo de Argentum, el gameloop lanzará a correr a los otros 2 threads, pasándole la instancia de *CommunicationProtocol* como referencia dado que estos serán los encargados de recibir el estado del mundo y de enviar los inputs del jugador al servidor, respectivamente.

A medida que vaya pasando el juego, el gameloop irá retirando de una queue de mensajes, *DataQueue*, la información del servidor para actualizar su modelo en consecuencia, y a partir de ahí renderizarlo según corresponda. Esta *DataQueue* no es bloqueante al realizar el pop y de esta manera se evita que el gameloop quede bloqueado esperando información del servidor y no pueda seguir renderizando o atendiendo inputs del jugador. Toda esta información es recibida por el hilo Receiver que los pushearán en la queue recién mencionada.

Por otro lado, el gameloop irá atendiendo a los inputs que realice el usuario, estos pueden ser por teclado o por eventos de mouse, dependiendo de la actividad que realice. Estos inputs serán enviados, o no, al servidor, que es el encargado de resolver la parte lógica de los inputs, dependiendo del estado del personaje del cliente. Todos los inputs son pusheados en una queue de comandos, *InputQueue*, que el hilo Dispatcher irá vaciando bloqueándose en el pop en caso de que esté sin ningún input. Si el input que obtuvo al realizar el pop es distinto a nada, procederá a encodear el comando para enviarlo al servidor. En caso contrario, ignorará el input del jugador.

Por último, una vez actualizado el estado del mundo, renderizarlo y haber manejado todos los eventos del jugador, el hilo principal "dormirá" el tiempo necesario para completar el loop en un tiempo estipulado como **gameloop time**.

3.1.2. Estados del personaje

Para manejar esta parte del diseño se utilizó el patrón **State**. Esto nos permitió enfrentarnos a los diferentes comandos disponibles en el juego y que cada uno actué de manera polifórmica dependiendo del estado del personaje. De esta manera, evitamos saturar al servidor con información inútil y que haría más lento la ejecución de las órdenes. Un ejemplo de esto, es el estado *ResurrectState* en los cuales ningún tipo de input es válido ya que el personaje, hecho un fantasma se desplaza automáticamente por el mapa hasta el sacerdote más cercano para que lo reviva. Pero a diferencia del estado anterior, el estado *StillState* permitirá ingresar casi cualquier tipo de acción, salvo aquellas que requieren de estar interactuando con algún profesional de las ciudades.

Hay ciertos estados que se verán en el servidor, como por ejemplo *EquipStateCharacter*, que son absorbidos por el estado *StillState* en el cliente.

El estado del jugador es enviado desde el servidor cada cierto intervalo de tiempo, dependiendo en cuál esté puede tener alguna animación o sonido característico, por ejemplo si el jugador se encuentra en el estado *MeditateState* se verá el efecto de esta acción.

3.1.3. Wrappers de SDL

Para mejorar las prestaciones de SDL, se realizaron wrappers RAII para las ventanas, texturas, fuentes y sonidos. Esto nos permitió trabajar con mayor comodidad y no depender de hacer llamadas a funciones en cualquier lado del código para liberar memoria utilizada por las clases y estructuras de SDL.

Además, dado que la carga de una imagen o sonido es costosa, se decidió generar 2 clases encargadas de realizar la carga de las mismas al inicio del cliente, el *MusicManager* y el *TextureManager*. Éstas clases sirven como elementos de consulta para las demás que requieran de una determinada textura para renderizarse o un sonido para acompañar la animación.

3.1.4. Renderización

La renderización del juego está principalmente distribuida en 2 clases, la *UI* y la *Camera*. La primera, es la encargada de mostrar por pantalla, el estado general del jugador, ya sea su vida, su inventario, su experiencia o su equipamiento. A su vez, en caso de que el jugador esté interactuando con algún NPC, también renderizará las acciones e items que estos nos dispongan.

Por otro lado, la clase *Camera* es la vista de águila centrada en el jugador objetivo para este cliente. Todas las texturas se renderizarán según su posición relativa a la posición de la cámara. Ambas tienen al jugador como atributo, dado que es necesario para poder mostrar la información actual al usuario.

3.2. Clases

A continuación se listarán las principales clases utilizadas dentro del cliente:

- **Game:** Es el hilo principal del cliente. Al crearse, inicializa la ventana donde se va a renderizar el juego, el hilo Dispatcher y el hilo Receiver, las queues que utilizará para comunicarse con estos 2, y el *CommunicationProtocol* que como ya vimos tiene al socket dentro de su clase. Tiene un método *init* que recibe los parámetros del juego y los utiliza para realizar la conexión inicial con el servidor, cargar los assets, generar la UI y la Camera. Por último, el método *run* es propiamente el gameloop del juego, del lado del cliente.

- **Dispatcher:** Hereda de la clase *Thread*. Al crearse recibe el *CommunicationProtocol*, que contendrá el socket ya conectado al servidor y la *InputQueue* que utilizará para comunicarse con el Game. Además, encodeará a los *InputInfo*, que obtiene de la queue, para enviarlos al servidor, siempre y cuando el input sea un comando válido.
- **Receiver:** Hereda de la clase *Thread*. Al crearse recibe el *CommunicationProtocol*, que contendrá el socket ya conectado al servidor y la *DataQueue* que utilizará para comunicarse con el Game, pusheando los *Message* recibidos desde el servidor.
- **CharacterState:** Esta clase es utilizada por los personajes del juego para retratar su estado. Las implementaciones de esta clase abstracta permitirán o no el efecto de ciertos comando. Cada uno de los estados se identifica con un *CharacterStateID*. Las clases que heredan de *CharacterState* son:
 - **AttackState**
 - **InteractState**
 - **MeditateState**
 - **MoveState**
 - **ResurrectState**
 - **StillState**
- **Character:** Es una clase abstracta que representa a los personajes presentes en el mundo de Argentum. Posee una referencia al *TextureManager* y al *MusicManager*, un id, y su posición. Además tiene como atributo una *Direction*, un *Body*, *Head*, *Helmet*, *Shield*, *Weapon* y *Animation*. La renderización de los mismos se hacen en una posición relativa a la cámara del juego centrada en el *Player*.
- **Player:** Es el jugador del cliente. Hereda de *Character*. Tiene como atributo a un *PlayerInfo* que es donde viene la información desde el servidor. Tiene un método *update* que actualiza las animaciones de los frames equipados y otro método llamado *updatePlayerInfo* que recibe un *PlayerInfo* y que actualiza el estado general del *Player* para sincronizarlo con el servidor. Por último, es capaz de manejar los inputs del usuario según el *CharacterState* en el cuál se encuentre.
- **NPC:** Hereda de *Character*. Representa a todos los demás personajes o items dropeados en el mapa que haya en Argentum. Tiene una funcionalidad similar a *Player*, sin poder manejar los inputs de usuario. Si se trata de una criatura, dependiendo de su *Body* también se asignará algún sonido característico.
- **Item:** Es una clase muy simple que solo conoce su textura, el ancho y el alto de la misma. Sabe renderizarse en la posición indicada. Es utilizada para renderizar en el mapa a aquellos objetos resultantes de algún drop.
- **Body:** Hereda de *Item*. Es una clase abstracta de la cual heredan todos los cuerpos o armaduras disponibles en Argentum. Contiene una *Direction* y conoce la cantidad de frames de la animación que se le serán seteados por las respectivas implementaciones. Además, es la encargada de realizar el *update* de las animaciones de este estilo. Todos los *Body* son identificados con id del tipo *BodyID*:
 - **BlueCommonBody**
 - **RedCommonBody**
 - **GreenCommonBody**
 - **BlueTunic**
 - **LeatherArmor**
 - **PlateArmor**
 - **BankerBody**
 - **PriestBody**
 - **MerchantBody**
 - **GhostBody**
 - **GoblinBody**
 - **SpiderBody**
 - **SkeletonBody**
 - **ZombieBody**

- **Head:** Hereda de *Item*. Es el equivalente de Body para las cabezas que representan a las distintas razas disponibles en el juego. Se identifican con un id del tipo *HeadID*:
 - **ElfHead**
 - **GnomeHead**
 - **DwarfHead**
 - **HumanHead**
 - **PriestHead**
 - **ZombieHead**
- **Helmet:** Hereda de *Item*. Es el equivalente de Head para los cascos que están disponibles en el juego. Se identifican con un id del tipo *HelmetID*:
 - **MagicHat**
 - **Hood**
 - **IronHelmet**
- **Shield:** Hereda de *Item*. Es el equivalente de Body para los escudos que están disponibles en el juego. Se identifican con un id del tipo *ShieldID*:
 - **TurtleShield**
 - **IronShield**
- **Weapon:** Hereda de *Item*. Es el equivalente de Body para las armas que están disponibles en el juego. Se identifican con un id del tipo *WeaponID*:
 - **AshStick**
 - **Ax**
 - **CompoundArc**
 - **Crosier**
 - **SimpleArc**
 - **Hammer**
 - **LongSword**
 - **GnarledStick**
 - **ElficFlaute**
- **Animation:** Muy similar a las anteriores pero representa a los efectos visuales y auditivos que se producen en el juego, estos pueden ser producidos por ataques de armas o por acciones que genere el jugador. Tienen una textura y un efecto que serán ejecutados en loop o no, dependiendo de su implementación en las clases que hereden de esta. Algunas de las implementaciones son:
 - **CureAnimation**
 - **ExplotionAnimation**
 - **HitAnimation**
 - **MagicArrowAnimation**
 - **MeditateAnimation**
 - **MissileAnimation**
- **UI:** Clase encargada de renderizar el estado general de todo el jugador. Tiene un puntero al jugador del cual debe mostrar las estadísticas, ya sea por su inventario o cantidad de vida, oro, etc o el equipamiento actual del jugador. Además, en caso de que el jugador esté en estado *Interact* se mostrará por pantalla la interfaz correspondiente al NPC con el cuál se esté interactuando. Esto se realiza delegándolo en clases que heredan de *NPCInterface* como lo pueden ser *PriestInterface*, *BankerInterface* o *MerchantInterface* donde cada una de ellas mostrará con el layout correspondiente la información que este profesional tenga para nosotros. Esta información es obtenida a través de un objeto del tipo *NPCInfo* que es enviado por el servidor en caso de estar interactuando.

- **Button:** Son los distintos tipos de botones que se pueden apreciar en la interfaz. Esta es una clase abstracta que tiene diferentes implementaciones como un *SelectButton* utilizado para seleccionar items del inventario o *RaisedButton* utilizado para generar acciones dependiendo de qué botón se trate, que llamarán al handle del *Player* correspondiente a ese evento.
- **Camera:** Es la clase destinada a mantener la cámara del juego centrada sobre el jugador que es seteado como objetivo. Al momento de crearla, tiene conocimiento de las dimensiones del mapa para no traspasar los límites del mismo. Si se le da un punto dado puede informar la distancia que tiene al objetivo de la cámara. Esto fue hecho para poder manejar los sonidos de ciertos NPC y que estos estén relativamente cerca para ejecutar su efecto sonoro.
- **Presentation:** Esta clase es la pantalla de carga del juego. Esperará que se ingrese un *ENTER* para continuar.
- **GameMap:** Es el mapa estático del juego. Está compuesto por un vector de *Tile* y un map que tiene todos las texturas utilizadas para renderizar el mapa. Es capaz de dibujar las distintas capas que lo componen, diferenciando el suelo de aquellos objetos que están en capas superiores, como casas y árboles.
- **Window:** Es el wrapper correspondiente a una ventana de SDL. Encapsula la creación, manipulación y liberación de los recursos asociados. Puede realizar el manejo de eventos correspondientes al tamaño de la ventana.
- **Tile:** Contiene una *Texture* y las dimensiones de la misma. Además conoce la posición en la cual debe ser dibujada. Es utilizada para renderizar el mapa estático de Argentum.
- **Texture:** Es el wrapper correspondiente a una textura de SDL. Encapsula la creación, manipulación y liberación de los recursos asociados. Tiene una referencia al renderer que la puede utilizar. Además posee un método *render* que recibe 2 *SDLRect* que permitirán renderizar una porción de textura en una porción del *Renderer*.
- **TextureManager:** Es la clase que contiene a todas las *Texture* que se cargan al iniciar el juego. Las identifica con un *TextureID* y puede ser consultada desde varios métodos *getTexture* que devuelve como referencia la textura correspondiente al parámetro indicado. En caso de que este parámetro no sea un *TextureID*, y sea, por ejemplo un *BodyID*, realizará una conversión para encontrar la textura correspondiente.
- **Music:** Es el wrapper correspondiente a un *MixMusic* de SDL. Encapsula la creación, manipulación y liberación de los recursos asociados. Es utilizado para reproducir la música de fondo del juego.
- **Effect:** Es el wrapper correspondiente a un *MixChunk* de SDL. Encapsula la creación, manipulación y liberación de los recursos asociados. Es utilizado para la carga y reproducción de los efectos de sonido de las animaciones o sonido ambiente que contiene el juego.
- **MusicManager:** Análogo al *TextureManager* pero en vez de *Texture* contiene objetos de tipo *Music* y *Effect*. A todos los identifica por un *MusicID* que si es pasado como parámetro a los métodos *getMusic* y *getEffect* se obtendrá una referencia constante a dicho elemento.
- **Font:** Es el wrapper correspondiente a una *TTFFont*. Encapsula la creación, manipulación y liberación de los recursos asociados. Al construirla, se genera a partir de un path, con un tamaño y color determinado que luego podrá modificarse. Permite crear texturas con un determinado texto. Estas texturas quedarán en responsabilidad del usuario en ser liberadas.

A continuación se listarán las clases comunes para ambas aplicaciones:

- **GameObjectInfo:** Es una clase destinada a informar el estado actual de cualquier personaje u objeto no estático del mundo de Argentum. Es la información utilizada por el NPC del cliente para renderizarse como corresponda. Contiene un id, la posición, la dirección a la cual está orientado, un atributo que representa las texturas que componen al objeto, como por ejemplo que *BodyID* y *HeadID* tiene. También indica el *CharacterStateID*, y si está siendo atacado, se indica el *WeaponID*, esto es para renderizar la animación correspondiente.
- **PlayerInfo:** Hereda de *GameObjectInfo*. Muy similar en su propósito pero es exclusivo para el personaje del cliente al cual el servidor se lo envía. Se informa la vida, el mana, el inventario, entre otras cosas, información solamente útil para el personaje del cliente.
- **InputInfo:** Es un struct utilizado para encapsular los inputs del usuario. Tiene como atributos el *InputID* que sirve para identificar la orden, un adicional que puede contener cualquier información extra dependiendo el comando y un *Point*, por si se trata de un input que requiera una posición en el mapa.

- **NPCInfo:** Es un struct utilizado para encapsular la información que tiene para brindarnos el NPC con el cual estamos interactuando. Contiene el tipo de NPC con el que interactuamos, las acciones que puede realizar, los items que tiene, etc.
- **Message:** Es un wrapper del mensaje recibido por el socket y es generado por el *CommunicationProtocol*. Conformado por un vector de uint8 que almacena la información recibida, la longitud del mismo y el tipo de mensaje que es. Cada tipo de mensaje se explicará más en detalle en la sección **Protocolo de Comunicación**. Además brinda un mecanismo de lectura para ir avanzando por los bytes.
- **BlockingQueue:** Es un template de una cola bloqueante que permite un acceso controlado a los recursos encolados. En su constructor tiene un booleano para indicar si es necesario que se bloquee al realizar el pop.
- **DataQueue:** Especificación de la BlockingQueue para datos de tipo *Message*.
- **InputQueue:** Especificación de la BlockingQueue para datos de tipo *InputInfo*.
- **Socket:** Clase que encapsula el comportamiento básico de un socket. Realiza el bind, accept, connect, send, receive, etc. Al pasarlo por referencia se pierde el ownership del mismo. No tiene habilitados los constructores por copia.
- **CommunicationProtocol:** Es un wrapper del Socket mencionado anteriormente. Surgió por la necesidad de tener una clase que permita utilizar el mismo socket por diferentes hilos de ejecución, para poder mantener una mejor comunicación sin que se que el hilo principal quede bloqueado. Tiene como métodos *connect*, *send* que wrapper los métodos del *Socket* y un método *receive* que retorna un *Message* con la longitud, el tipo de mensaje y el mensaje en sí.
- **Decoder:** Clase estática capaz de encodear y decodear las diferentes cosas que se envían en el protocolo de comunicación diseñado para Argentum. En la sección **Protocolo de Comunicación** ahondaremos más en esta clase.
- **Chrono:** Clase diseñada para devolver la cantidad de milisegundos que hubo entre dos llamadas a la función *lap*.
- **Point:** Clase diseñada para manejar coordenadas (x,y). Sobrecarga diversos operadores que permiten manejar de mejor manera las posiciones de los jugadores dentro del mundo de Argentum.
- **Random:** Clase estática diseñada para generar números aleatorios entre intervalos, estos pueden ser float o int.
- **Thread:** Clase abstracta y que encapsula el comportamiento de un thread. Tiene implementado los métodos *start* y *join*, dejando a sus clases hijas la implementación del método *run*.
- **ObjectLayer:** Clase que se utiliza para almacenar las posiciones estáticas de los objetos dentro del mapa.
- **JsonReader:** Clase estática que recibe un archivo .json y lo serializa a un objeto json que luego será parseado.
- **StaticObject:** Tiene las dimensiones de los objetos estáticos como las casa, los NPC, los árboles y los puntos de nido desde donde aparecen las criaturas. Esta clase es utilizada en la creación del Board para localizar las celdas que están ocupadas con objetos estáticos.
- **TiledMap:** Clase utilizada como DTO que tiene la información estática del mapa creado con la herramienta Tiled. Esta clase se encodea en el servidor y es decodeada en el cliente para su posterior transformación a la clase GameMap. El proceso que se realiza para encodear y decodear el mapa se explica en detalle en la sección Protocolo de Comunicación.
- **TileLayer:** Clase que contiene información sobre la distribución de los tiles en el mapa. Posee los id de cada tile que luego van a ser utilizados en conjunto con la clase TileSet para el renderizado del mapa.
- **TileSet:** Clase que contiene el path a la imagen que en el cliente se utilizará para generar la textura correspondiente. También tiene el firstgid que cumple el propósito de vincular cada tileset con el tile correspondiente.

3.3. Diagramas UML

El siguiente diagrama nos permite observar como está compuesta la clase *Game* y como se relacionan los threads *Dispatcher* y *Receiver* con ella. A través de las respectivas colas bloqueantes y utilizando el mismo *Communication-Protocol* compuesto por un *Socket*.

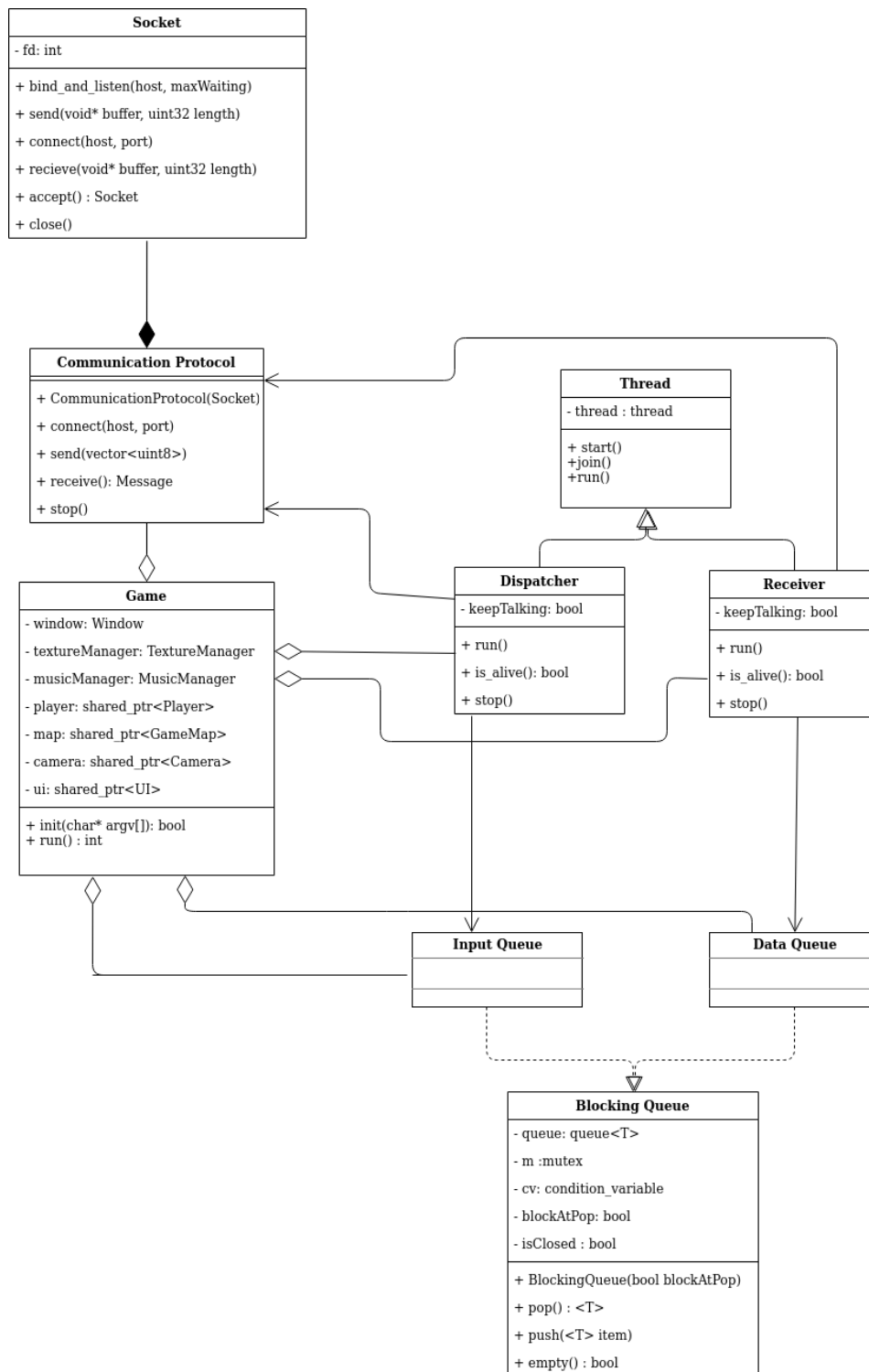


Figura 1: Diagrama Clase Game

A continuación, veremos un diagrama de secuencia que nos permite entender la conexión inicial desde el punto de vista del cliente.

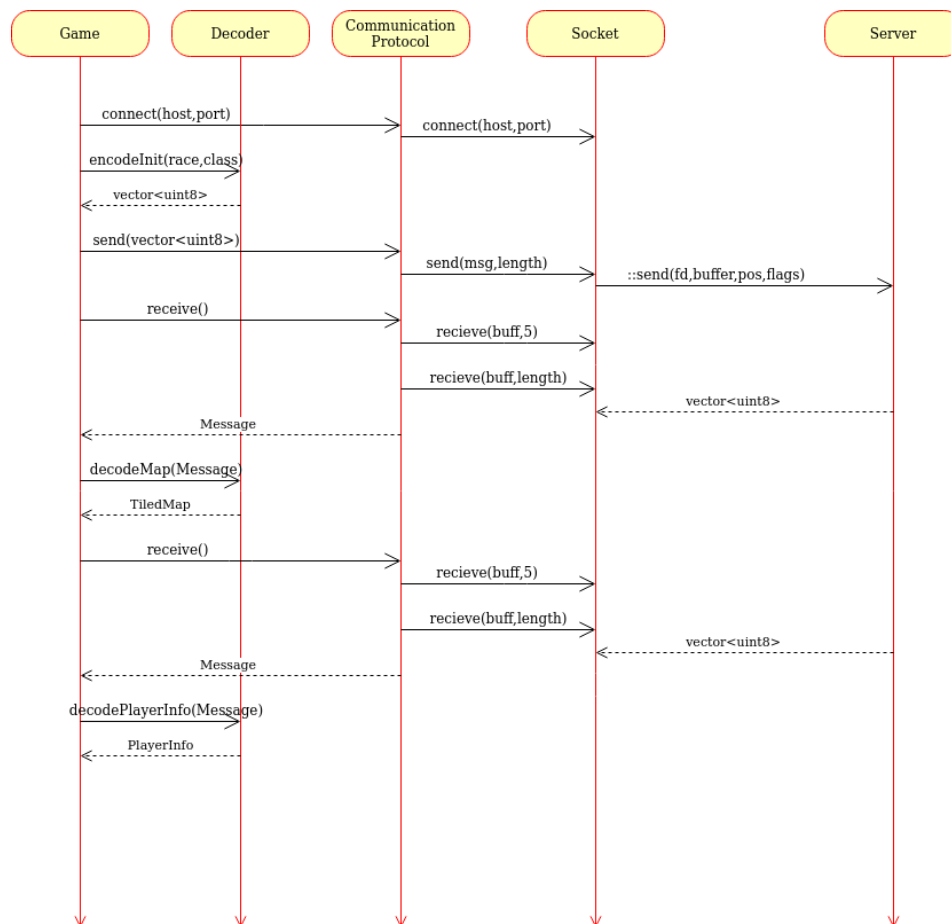


Figura 2: Diagrama Conexión Inicial

Una vez sucedidos los pasos explicados en la Figura 2, es cuando el hilo principal, lanza a correr los hilos Dispatcher y Receiver pasándoles como referencia al *CommunicationProtocol* que contiene el *Socket* conectado al servidor.

4. Servidor

4.1. Descripción general

El servidor recibe un único parámetro, el archivo de configuración, que es donde se encuentra entre otras cosas, las descripciones de las distintas razas, criaturas y clases. También nos brinda información sobre el puerto en donde el servidor estará aceptando nuevos clientes.

Una vez que el servidor haya cargado el archivo de configuración y el mapa, ambos archivos *.json*, procede a lanzar los thread World y PlayerAceptor y se queda esperando un carácter por entrada estándar. En caso que dicho carácter sea la letra 'q', el hilo procede a cerrar de forma ordenada los otros 2 hilos y se cierra.

4.1.1. Estructura

El servidor se divide en 3 threads:

- Main
- World
- PlayerAceptor

El hilo World, se encarga de inicializar el mapa creado previamente con Tiled y de tenerlo en memoria, para poder pasarlo a los clientes que se conectan. El hilo PlayerAceptor, como bien indica su nombre, es el encargado de aceptar nuevos clientes que son insertados en el World, el cual notifica los eventos que ocurren y recibe las acciones de todos los clientes.

4.1.2. Proceso para aceptar un nuevo cliente

Cada vez que se detecta una nueva conexión, el acceptor lanza un `ThLobbyPlayer`, al cual se le pasa una referencia al `World` y por lo tanto cuando recibe un mensaje de inicio del cliente es el encargado de que se cree un nuevo personaje con la raza y la clase especificados en el mensaje. También se encarga de enviar tanto el mapa como la información del personaje creado. El `World` añade este nuevo cliente a su lista de jugadores activos y mantendrá comunicación con el mismo, hasta que se cierre el servidor o el cliente abandone la partida. Luego de realizados estos pasos, se realiza un join al `ThLobbyPlayer` y queda a la espera de ser eliminado por el `PlayerAcceptor`. La comunicación con el `World` se da a través del `ThPlayer`.

4.1.3. Inicio del World y carga del Board

El `World` al iniciarse, carga el archivo *finishedMap.json* y con ayuda del `JsonReader` genera un objeto json que es utilizado por la clase `TiledMap` para crear el mapa. Luego se instancian las distintas profesiones y con el `TiledMap` anteriormente cargado y el archivo de configuración, se procede a inicializar el `Board`. El `Board` hace uso de las dimensiones del mapa para determinar la cantidad de celdas a crear y con la información obtenida en los `objectLayers`, determina que celdas serán ocupadas por los objetos estáticos del mapa (Construcciones, árboles y murallas) con los que los personajes puedan colisionar. También configura las celdas que pertenecen a una de las ciudades o a nidos de criaturas. Por último el `World` se encarga de colocar a los npc en sus respectivas posiciones iniciales y de agregar a las criaturas en los distintos nidos hasta que se llegue al límite establecido en el archivo de configuración.

4.2. Clases

A continuación se listarán las principales clases utilizadas dentro del servidor:

- **World:** Una vez inicializada la clase, por el proceso explicado anteriormente, se encarga de realizar el update periódico de todos los `gameObjects` e informar a los distintos clientes los eventos ocurridos durante ese periodo de tiempo. También gestiona la creación y destrucción de los de las criaturas y de los ítems generados a partir de la destrucción de las mismas.
- **PlayerAcceptor:** Hereda de `Thread` y es el hilo responsable de ir aceptando a los nuevos jugadores que se conectan al servidor. A medida que acepta a un nuevo cliente, lanzará a correr a otro thread llamado *ThLobbyPlayer*.
- **ThLobbyPlayer:** Hereda de `Thread`. Es el encargado de realizar la comunicación inicial con el cliente e indicarle al mundo la raza y clase que este eligió. A su vez enviará el mapa estático y su primer `PlayerInfo`.
- **ThPlayer:** Clase encargada de enviar la información actualizada del mundo al cliente. Esto no se realiza todo el tiempo, si no que se hace periódicamente cuando el servidor realiza el update de los `gameObjects`. La información que se envía al cliente está compuesta por un `PlayerInfo`, una lista de todos los `gameObjects` del juego, sin contar al jugador correspondiente al cliente en cuestión, y se envía un `NPCInfo` en caso de que el jugador esté interactuando con un NPC. La comunicación se efectúa utilizando la clase `Communication Protocol` y el `Decoder`. Los distintos mensajes enviados, serán detallados en la sección `Protocolo de Comunicación`.
- **ThPlayerReceiver:** Es la clase encargada de recibir los inputs del cliente y los encola en la `InputsQueue` del `gameCharacter` asociado al `ThClient`. De esta forma el personaje cuando termina un estado, utiliza estos inputs para modificar su estado.
- **Board:** Clase encargada de contener a las celdas del juego y realizar operaciones entre ellas. Algunas de las operaciones que nos permite realizar son:
 - Obtener la distancia entre 2 celdas, para eso se utilizó la distancia Manhattan.
 - Obtener una celda a partir de un punto y viceversa.
 - Obtener las celdas adyacentes a una celda.
 - Obtener la siguiente celda, dada una dirección de movimiento.
 - Obtener el camino mínimo entre 2 celdas, utilizando el algoritmo A*.
- **NestContainer:** Clase que contiene información de los distintos nidos que se encuentran por el mapa. Permite obtener el nido con menos cantidad de criaturas de forma eficiente.
- **Nest:** Es la clase encargada de configurar los límites por donde se pueden mover las criaturas que fueron creadas en ese nido. Almacena los ids de las criaturas que posee y resulta útil a la hora de notificar la presencia de un personaje en el nido.

- **GameObjectContainer:** Es la clase encargada de almacenar a todos los gameObjects del juego, permite realizar una serie de operaciones con ellos.
- **GameObject:** Clase abstracta que se utiliza para representar un objeto de juego en el servidor. Dicho objeto puede tomar la forma de un jugador, un npc, una criatura o un ítem. Además se encarga de generar el GameObjectInfo que luego será enviado al cliente
- **GameCharacter:** Es la clase que modela a un personaje del lado del servidor. Recibe los inputs enviados por el cliente y los procesa durante el update. Para eso, hace uso de su estado actual. También se encarga de generar el PlayerInfo que luego va a ser enviado al cliente.
- **Creature:** Es un clase que sirve para modelar una criatura del lado del servidor. Al igual que GameCharacter utiliza estados para realizar su update con la diferencia de que no recibe inputs del cliente, sino que los genera por su cuenta permitiéndole alternar su estado actual. La mayor parte del tiempo se encuentra moviéndose sin rumbo dentro de su nido, esperando que un jugador entre en el mismo. Es en ese momento que actualiza su estado para empezar a perseguirlo y eventualmente atacar.
- **NPCServer:** Esta clase modela un NPC del lado del servidor de forma tal que pueda interactuar con el personaje. Todo NPC tiene una profesión que le permite manejar de forma polimórfica la interacción con el personaje. Las profesiones están divididas en 3 variantes, las cuales son:
- **Profession:** Clase abstracta que modela el conocimiento que tendrá un NPCServer. Cada una de sus implementaciones como lo son *Banker*, *Merchant* y *Priest* serán capaz de procesar el input del usuario y brindará un *NPCInfo* con la información correspondiente a esa profesión.
- **ObjectItem:** Hereda de GameObject, su propósito es el de modelar un ítem del lado del servidor. Después de un determinado tiempo y si no fue tomado por algún jugador, desaparece del mapa.
- **State:** Es la clase abstracta de la que dependen tanto los personajes como las criaturas para realizar su update. A pesar que en cliente los estados tienen un comportamiento homólogo, en el servidor no pasa lo mismo. Debido a esto, se procedió a tener ambos tipos de estado por separado, resultando en la creación de 2 clases que heredan de State. Las mismas son:
 - **StatePoolCharacter**
 - **StatePoolCreature**
- **StatePool:** Es la clase abstracta que almacena los posibles estados que puede llegar a tener un GameObject. Al inicializar solo tiene el estado Still y los demás estados se van creando, conforme se necesiten. Tiene un solo estado como el actual y lo va alternando, según los inputs que recibe del jugador o los creados automáticamente por las criaturas. Al tener 2 máquinas de estado distintas, una para el personaje y otra para las criaturas, fue necesario hacer 2 clases que hereden de esta clase.
- **StatePoolCharacter:** Hereda de StatePool, es el encargado de manejar los StatePoolCharacter, que pueden tomar los siguientes valores:
 - **StillStateCharacter**
 - **MoveStateCharacter**
 - **AttackStateCharacter**
 - **EquipStateCharacter**
 - **InteractStateCharacter**
 - **MeditateStateCharacter**
 - **TakeAndDropStateCharacter**
 - **ResurrectStateCharacter**
- **StatePoolCreature:** Hereda de StatePool, es el encargado de manejar los StatePoolCreature, que pueden tomar los siguientes valores:
 - **StillStateCreature**
 - **MoveStateCreature**
 - **AttackStateCreature**
 - **PursuitStateCreature**

4.3. Diagramas UML

El siguiente diagrama nos permite observar como está compuesta la clase *World* y como se relaciona con las clases que la componen. En

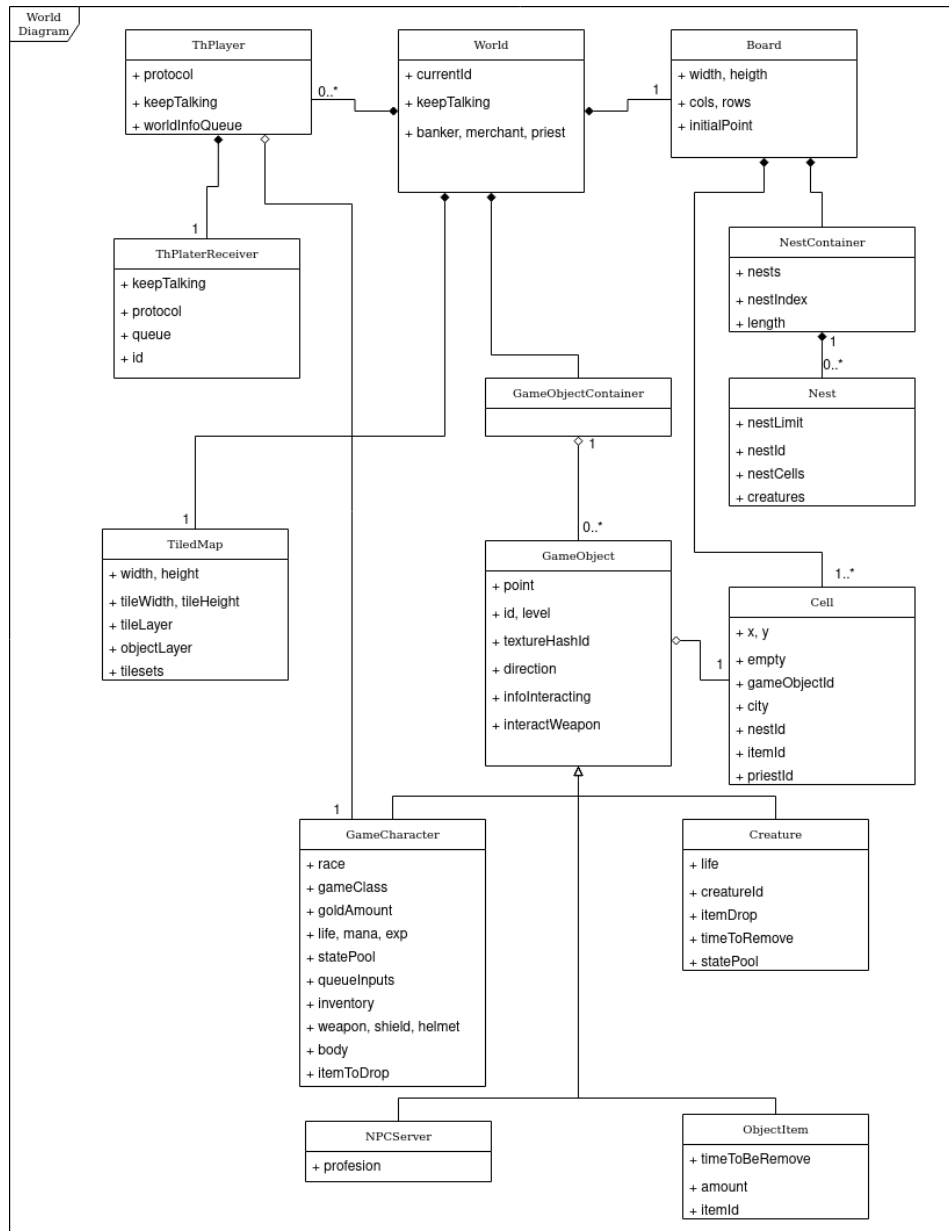


Figura 3: Diagrama Clase World

A continuación, se podrán observar los diagramas de estado para los *Character* y las *Creatures* respectivamente. En ellos se puede observar algunos de los mensajes que realizan que un personaje de Argentum, cambie de estado:

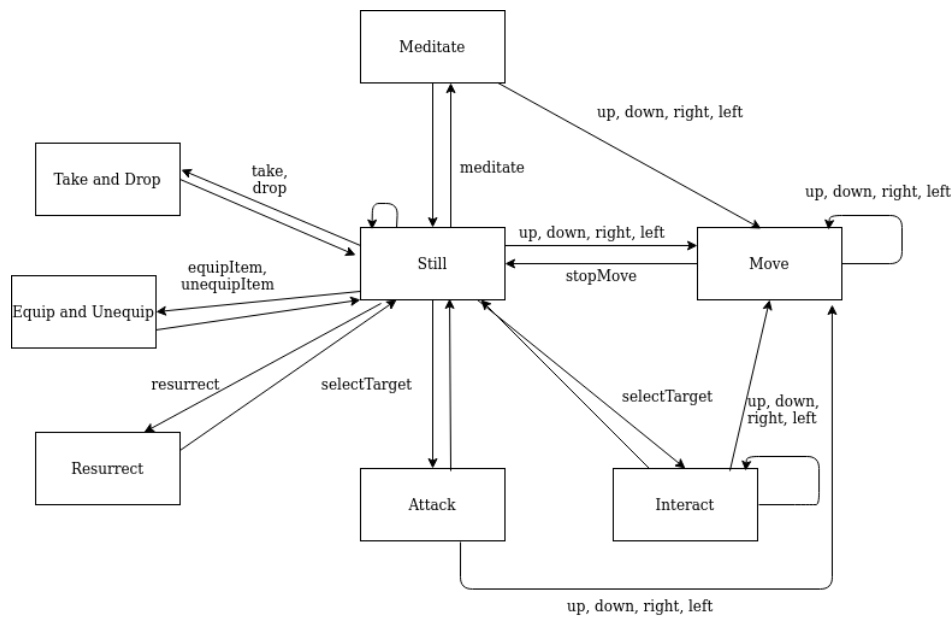


Figura 4: Diagrama Estado Character

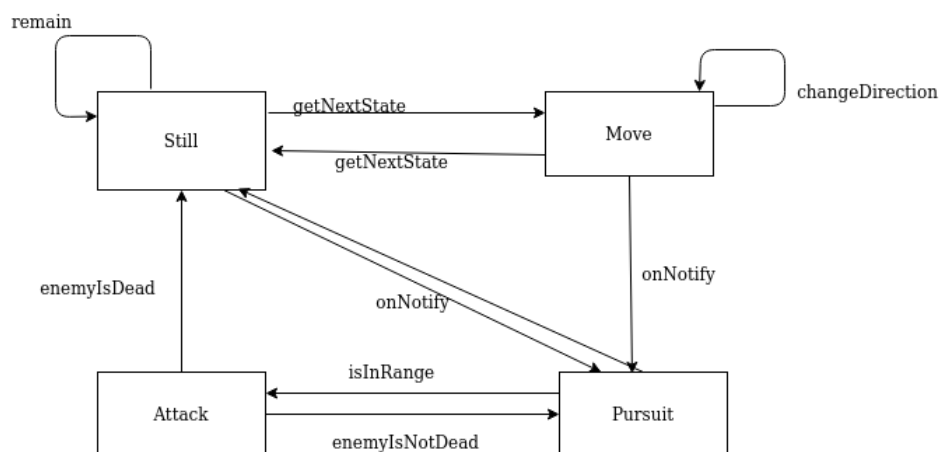


Figura 5: Diagrama Estado Creatures

5. Protocolo de Comunicación

El protocolo de comunicación utilizado fue diseñado a medida de las necesidades del juego. Se trata de un protocolo binario que consta de diferentes tipos de mensajes.

Al entablarse una comunicación entre el cliente y el servidor, el cliente enviará un mensaje de inicio que está compuesto por la raza y la clase seleccionada para su personaje. Una vez recibido esto, el servidor le enviará al cliente el mapa estático y la información inicial para un nuevo jugador creado con las características indicadas en el mensaje de inicio.

Una vez realizado estos 3 pasos, ya el servidor irá enviando cada cierto intervalo de tiempo a todos sus clientes el estado del mundo. Esto incluye el correspondiente PlayerInfo para ese cliente y todos los GameObjects que estén dentro del mundo de Argentum en ese momento.

En caso de que el jugador esté interactuando con alguno de los distintos profesionales dispersos por el mapa, también se enviará un mensaje con la información que este tenga para brindarnos.

Por último, el cliente irá enviando al servidor todos los comandos ingresados por el jugador. Para ello, también hay una codificación especial.

El encargado de generar todo el encoding y decoding de los mensajes es una clase estática llamada *Decoder*. Cada método de encoding recibirá el tipo de dato a convertir y devolverá un vector de bytes. Por otro lado, cada método de decoding recibirá un *Message* y devolverá el tipo de dato ya creado. A continuación se explicará los diversos mensajes y la forma de entender cada uno.

Antes de cada mensaje, se envía la longitud que tendrá el mismo en un entero sin signo de 4 bytes en big endian. Seguido de estos 4 bytes, viene un byte que indica el tipo de mensaje que es. Por último viene el mensaje. Todos los números están en formato big endian.

5.1. Mapa

Si el primer byte del mensaje es un 0, estamos frente al mapa estático del juego. En el cuál se le notifica al cliente la estructura del mapa estático. Para ello, el mensaje se encodea de la siguiente manera:

- **Ancho Mapa:** entero de 2 bytes sin signo en big endian.
- **Alto Mapa:** entero de 2 bytes sin signo en big endian.
- **Ancho Tile:** 1 byte indicando el ancho del tile.
- **Alto Tile:** 1 byte indicando el alto del tile.
- **Cantidad de Layers:** 1 byte indicando la cantidad de layers a enviar.
- **Cantidad Tiles en Layer:** entero de 2 bytes sin signo en big endian, que indica la cantidad de tiles en cada layer.
- A continuacion se deberá enviar Cantidad de Layers * Cantidad Tiles en Layer, la información de cada tile codificada de la siguiente manera :
- **Información de Tile:** entero de 2 bytes sin signo en big endian.
- Comienzo de codificación de TileSets:
- **Cantidad de TileSets:** entero de 4 bytes sin signo en big endian.
- Por cada tileSet especificado en Cantidad de TileSets, se deberá codificar lo siguiente:
- **Primer GID TileSet:** entero de 2 bytes sin signo en big endian.
- **TileSet ID:** entero de 2 bytes sin signo en big endian.

5.2. PlayerInfo

Si el primer byte del mensaje es un 1, estamos frente a un tipo de mensaje de PlayerInfo. En el cuál, se lo notifica al cliente del estado actual de su jugador. Para ello, el mensaje se encodea de la siguiente manera:

- **ID:** entero de 2 bytes sin signo en big endian.
- **Vida:** entero de 2 bytes sin signo en big endian.
- **Vida Máxima:** entero de 2 bytes sin signo en big endian.
- **Mana:** entero de 2 bytes sin signo en big endian.
- **Mana Máximo:** entero de 2 bytes sin signo en big endian.
- **Experiencia:** entero de 4 bytes sin signo en big endian.
- **Experiencia Máxima:** entero de 4 bytes sin signo en big endian.
- **Nivel:** entero de 2 bytes sin signo en big endian.
- **Oro:** entero de 2 bytes sin signo en big endian.
- **Oro seguro:** entero de 2 bytes sin signo en big endian.
- **Equipamiento:** 5 bytes que representan al HelmetID, HeadID, BodyID, ShieldID y WeaponID. Siempre deben estar en el mismo orden, en caso de que no tenga equipado nada, irá un byte en 0.
- **Inventario:** 9 bytes que representan a los ItemsInventoryID que están dentro del inventario del jugador. En caso de que la posición esté vacía ira un byte en 0.
- **Estado:** entero de 2 bytes sin signo en big endian. Representa el CharacterStateID del jugador.

- **Dirección:** entero de 2 bytes sin signo en big endian. Representa la Direction del jugador.
- **Posición:** 2 enteros de 2 bytes sin signo en big endian. Representan la posición en x y la posición en y respectivamente.
- **Atacado por:** un byte que representa el WeaponID en caso de que el jugador esté siendo atacado por algún arma en particular. Si no hay ataque, el byte va en 0.

5.3. GameObjects

Si el primer byte del mensaje es un 2, estamos frente a un tipo de mensaje de GameObjects. En el cuál, se lo notifica al cliente el estado de todos los objetos renderizables del mundo. Para ello, el mensaje se encodea de la siguiente manera:

- **Cantidad de Objetos:** un entero sin signo de 4 bytes en big endian. Es la cantidad de objetos que contiene el vector de GameObjectsInfo a decodificar.
- Por cada objeto del mundo se tendrá que codificar lo siguiente y de manera contigua:
- **ID:** entero de 2 bytes sin signo en big endian.
- **Tipo:** un byte que si es 0, se trata de un personaje y si es 1 de un item resultante de algún drop.
- **Equipamiento:** 6 bytes que representan al HelmetID, HeadID, BodyID, ShieldID, WeaponID y ItemsInventoryID Siempre deben estar en el mismo orden, en caso de que no tenga equipado nada, irá un byte en 0.
- **Estado:** entero de 2 bytes sin signo en big endian. Representa el CharacterStateID del jugador. En caso de ser un item, ira en 0.
- **Dirección:** entero de 2 bytes sin signo en big endian. Representa la Direction del jugador. En caso de ser un item, ira en 0.
- **Posición:** 2 enteros de 2 bytes sin signo en big endian. Representan la posición en x y la posición en y respectivamente.
- **Atacado por:** un byte que representa el WeaponID en caso de que el jugador esté siendo atacado por algún arma en particular. Si no hay ataque, el byte va en 0.

5.4. InputInfo

Si el primer byte del mensaje es un 3, estamos en frente a un comando del usuario. Este mensaje va desde el cliente al servidor. Para ello, el mensaje se encodea de la siguiente manera:

- **InputID:** 1 byte indicando el id del input ingresado.
- **Posición:** 2 enteros de 2 bytes sin signo en big endian. Representan la posición en x y la posición en y respectivamente del click realizado por el usuario.
- **Additional:** entero de 2 bytes sin signo en big endian. Representa una información adicional que puede ser de utilidad para algunos inputs.

5.5. Mensaje de Inicio

Si el primer byte del mensaje es un 4, estamos en frente a un comando del usuario. Este mensaje va desde el cliente al servidor. Para ello, el mensaje se encodea de la siguiente manera:

- **RaceID:** un byte que representa el id de la raza elegida por el jugador.
- **ClassID:** un byte que representa el id de la clase elegida por el jugador.

5.6. Mensaje de NPC

Si el primer byte del mensaje es un 5, estamos frente a un mensaje que contiene un *NPCInfo*, dado que el jugador está interactuando con un profesional. Para ello, el mensaje se encodea de la siguiente manera:

- **Tipo NPC:** un byte que nos indica con que NPC estamos interactuando. Si es un 1 estamos hablando con un Comerciante, si es un 2 con el sacerdote y si es un 3 con el banquero.
- **Cantidad Acciones:** un byte que nos indica la cantidad de acciones que realiza el NPC.
- **Acciones:** un byte que representa el id de cada una de las acciones que realiza el NPC.
- **Cantidad Items:** un byte que nos indica la cantidad de items que tiene disponible ese NPC.
- **Items:** son duplas de un byte y un entero de 2 bytes sin signo en big endian. Representan el id del item junto a su costo en oro.
- **Oro:** un entero de 2 bytes sin signo en big endian que representa la cantidad de oro depositada en la cuenta bancaria. Aplica solamente para el banquero.
- **Cantidad Items Depositados:** un byte que nos indica la cantidad de items que tiene disponible el banquero en la cuenta bancaria.
- **Items en Banco:** un byte por cada item en el banco que representa el `ItemsInventoryID` del mismo.

6. Código

jul 21, 20 15:20	WorldInfoQueue.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_WORLDINFOQUEUE_H	
2	#define ARGENTUM_TALLER_WORLDINFOQUEUE_H	
3		
4	#include "WorldInfo.h"	
5	#include "../common/BlockingQueue.h"	
6		
7	using WorldInfoQueue = BlockingQueue<WorldInfo>;	
8		
9	#endif //ARGENTUM_TALLER_WORLDINFOQUEUE_H	

jul 21, 20 15:20	WorldInfo.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_WORLDINFO_H	
2	#define ARGENTUM_TALLER_WORLDINFO_H	
3		
4		
5	#include "../common/PlayerInfo.h"	
6	#include "GameObject.h"	
7		
8	class WorldInfo {	
9	private:	
10	PlayerInfo playerInfo;	
11	std::vector<GameObjectInfo> gameObjectsInfo;	
12	NPCInfo npcInfo;	
13	public:	
14	WorldInfo(std::vector<std::shared_ptr<GameObject>> gameObjects, uint id);	
15		
16	const PlayerInfo &getPlayerInfo() const ;	
17		
18	const std::vector<GameObjectInfo> &getGameObjectsInfo() const ;	
19		
20	const NPCInfo &getNpcInfo() const ;	
21		
22	virtual ~WorldInfo();	
23		
24	};	
25		
26		
27	#endif //ARGENTUM_TALLER_WORLDINFO_H	

jul 21, 20 15:20	WorldInfo.cpp	Page 1/1
1	<code>#include "WorldInfo.h"</code>	
2	<code>#include "GameCharacter.h"</code>	
3		
4	<code>WorldInfo::~WorldInfo() = default;</code>	
5	<code>WorldInfo::WorldInfo(std::vector<std::shared_ptr<GameObject>> gameObjects, uint</code>	
6	<code>id) :</code>	
7	<code> playerInfo(), gameObjectsInfo(), npcInfo(){</code>	
8	<code> std::vector<std::shared_ptr<GameObject>>::iterator iter;</code>	
9	<code> iter = gameObjects.begin();</code>	
10	<code> while (iter != gameObjects.end()){</code>	
11	<code> if ((*iter)→getId() == id){</code>	
12	<code> this→npcInfo = (*iter)→getInteractInfo();</code>	
13	<code> this→playerInfo = (*iter)→getPlayerInfo();</code>	
14	<code> iter = gameObjects.erase(iter);</code>	
15	<code> } else {</code>	
16	<code> this→gameObjectsInfo.push_back((*iter)→getGameObjectInfo());</code>	
17	<code> iter++;</code>	
18	<code> }</code>	
19	<code> }</code>	
20	<code> }</code>	
21		
22	<code> const PlayerInfo &WorldInfo::getPlayerInfo() const {</code>	
23	<code> return playerInfo;</code>	
24	<code> }</code>	
25	<code> const std::vector<GameObjectInfo> &WorldInfo::getGameObjectsInfo() const {</code>	
26	<code> return gameObjectsInfo;</code>	
27	<code> }</code>	
28		
29	<code> const NPCInfo &WorldInfo::getNpcInfo() const {</code>	
30	<code> return npcInfo;</code>	
31	<code> }</code>	
32		

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	World.h	Page 1/2
1	<code>#ifndef WORLD_H</code>	
2	<code>#define WORLD_H</code>	
3		
4	<code>#include <string></code>	
5	<code>#include <unordered_map></code>	
6	<code>#include <mutex></code>	
7	<code>#include <atomic></code>	
8	<code>#include "../common/TiledMap.h"</code>	
9	<code>#include "Board.h"</code>	
10	<code>#include "GameObject.h"</code>	
11	<code>#include "GameStatsConfig.h"</code>	
12	<code>#include "../common/ObjectLayer.h"</code>	
13	<code>#include "../common/Thread.h"</code>	
14	<code>#include "../common/InputQueue.h"</code>	
15	<code>#include "GameCharacter.h"</code>	
16	<code>#include "ThPlayer.h"</code>	
17	<code>#include "Banker.h"</code>	
18	<code>#include "Merchant.h"</code>	
19	<code>#include "Priest.h"</code>	
20	<code>#include "GameObjectsContainer.h"</code>	
21		
22		
23	<code>class World: public Thread {</code>	
24	<code>private:</code>	
25	<code> TiledMap map;</code>	
26	<code> Board board;</code>	
27	<code> GameStatsConfig& gameStatsConfig;</code>	
28	<code> GameObjectsContainer gameObjectsContainer;</code>	
29	<code> std::atomic<uint> current_id;</code>	
30	<code> std::atomic<bool> keepTalking;</code>	
31	<code> mutable std::mutex m;</code>	
32	<code> std::unordered_map<uint, ThPlayer*> players;</code>	
33	<code> Banker* banker;</code>	
34	<code> Merchant* merchant;</code>	
35	<code> Priest* priest;</code>	
36		
37	<code> void addNPCs(std::vector<ObjectLayer> objectLayers);</code>	
38		
39	<code> uint getNextId();</code>	
40		
41	<code> std::vector<std::shared_ptr<GameObject>> getUpdatedGameObjects();</code>	
42		
43	<code> void addCreatures();</code>	
44		
45	<code> void generateCreature();</code>	
46		
47	<code> void generateItem(const DropItem &dropItem, const std::shared_ptr<Cell>& emp</code>	
48	<code>tyCell);</code>	
49	<code> void clearFinishedPlayers();</code>	
50		
51	<code> void removeCreaturesAndItems();</code>	
52		
53	<code> void update();</code>	
54		
55	<code> void checkDrops();</code>	
56	<code>public:</code>	
57	<code> explicit World(GameStatsConfig& configuration);</code>	
58		
59	<code> TiledMap& getStaticMap();</code>	
60		
61	<code> std::shared_ptr<GameCharacter> createCharacter(RaceID race, GameClassID game</code>	
62	<code>Class);</code>	
63		
64	<code> virtual void run();</code>	

2/217

jul 21, 20 15:20	World.h	Page 2/2
65	void stop();	
66		
67	void addPlayer(ThPlayer* aPlayer, uint id);	
68		
69	virtual ~World();	
70	};	
71		
72	#endif	

jul 21, 20 15:20	World.cpp	Page 1/3
1	#include "World.h"	
2	#include "../common/JsonReader.h"	
3	#include "NPCServer.h"	
4	#include "../common/Random.h"	
5	#include "Creature.h"	
6	#include "ObjectItem.h"	
7	#include <iostream>	
8		
9	#define GAMELOOPTIME 1000000/45.0	
10		
11	World::World(GameStatsConfig& configuration) : gameStatsConfig(configuration),	
12	current_id(1), keepTalking(true) {	
13	std::string path(CONFIG_DIR+std::string("/finishedMap.json"));	
14	rapidjson::Document jsonMap = JsonReader::read(path);	
15	this->banker = Banker::getInstance();	
16	this->merchant = Merchant::getInstance();	
17	this->merchant->init(configuration.getItems());	
18	this->priest = Priest::getInstance();	
19	this->priest->init(configuration.getItems());	
20	this->merchant->init(configuration.getItems());	
21	this->priest->init(configuration.getItems());	
22	this->map = TiledMap(jsonMap);	
23	this->board = Board(map, GameStatsConfig::getNestCreatureLimit());	
24	addNPCs(map.getObjectLayers());	
25	addCreatures();	
26	}	
27		
28	uint World::getNextId() {	
29	return current_id++;	
30	}	
31		
32	std::shared_ptr<GameCharacter> World::createCharacter(RaceID race, GameClassID g	
33	ameClass) {	
34	std::unique_lock<std::mutex> lock(m);	
35	uint id = getNextId();	
36	std::shared_ptr<Cell> initialCell = board.getInitialCell();	
37	initialCell->occupied(id);	
38	std::shared_ptr<GameCharacter> aCharacter(new GameCharacter(id, race, gameCl	
39	ass, initialCell, board.getPointFromCell(initialCell));	
40	gameObjectsContainer.addGameObject(aCharacter, id);	
41	return aCharacter;	
42	}	
43	void World::addNPCs(std::vector<ObjectLayer> objectLayers) {	
44	for (auto &anObjectLayer : objectLayers) {	
45	if (anObjectLayer.getName() == "NPC") {	
46	std::shared_ptr<Cell> aCell;	
47	for (StaticObject &anPCObject : anObjectLayer.getObjects()) {	
48	uint id = getNextId();	
49	aCell = board.getCellFromPoint(anPCObject.getTopLeft());	
50	aCell->occupied(id);	
51	std::shared_ptr<NPCServer> aNPC(new NPCServer(id, anPCObject.get	
52	Name(), board.getPointFromCell(aCell), aCell));	
53	gameObjectsContainer.addGameObject(aNPC, id);	
54	}	
55	}	
56		
57	void World::addCreatures() {	
58	for (int i = 0; i < GameStatsConfig::getCreaturesLimit(); ++i) {	
59	generateCreature();	
60	}	
61		
62		
63	void World::generateCreature() {	

jul 21, 20 15:20	World.cpp	Page 2/3
64	uint id = getNextId();	
65	uint8_t randomId = Random::get(1, 4);	
66	try {	
67	Nest& aNest = board.getAvailableNest();	
68	std::shared_ptr<Cell> initialCell = board.getInitialCellInNest(aNest);	
69	initialCell->occupied(id);	
70	aNest.addCreature(id);	
71	std::shared_ptr<Creature> aCreature(new Creature(id, CreatureID(randomId	
72), initialCell, board.getPointFromCell(initialCell));	
73	gameObjectsContainer.addGameObject(aCreature, id);	
74	} catch (Exception& e) {	
75	std::cout << "Cannot create creature" << std::endl;	
76	}	
77	}	
78		
79	void World::generateItem(const DropItem& dropItem, const std::shared_ptr<Cell>&	
80	initialCell) {	
81	uint id = getNextId();	
82	std::shared_ptr<ObjectItem> aObjectItem(new ObjectItem(id, board.getPointFro	
83	mCell(initialCell), initialCell, dropItem);	
84	initialCell->setItemId(id);	
85	gameObjectsContainer.addGameObject(aObjectItem, id);	
86	std::cout << "Item created -- ID: " << id << std::endl;	
87	}	
88	void World::run() {	
89	Chrono chrono;	
90	double initLoop, endLoop, sleep;	
91	int amountCreaturesDiff;	
92	std::vector<std::shared_ptr<GameObject>> gameObjects;	
93	while (keepTalking) {	
94	initLoop = chrono.lap();	
95	amountCreaturesDiff = GameStatsConfig::getCreaturesLimit() - board.getAm	
96	ountCreatures();	
97	if (amountCreaturesDiff > 0) {	
98	generateCreature();	
99	}	
100	update();	
101	clearFinishedPlayers();	
102	for (auto &aPlayer : players) {	
103	gameObjects = gameObjectsContainer.getUpdatedGameObjects();	
104	aPlayer.second->update(WorldInfo(gameObjects, aPlayer.first));	
105	}	
106	checkDrops();	
107	removeCreaturesAndItems();	
108	endLoop = chrono.lap();	
109	sleep = GAMELOOPTIME - (endLoop - initLoop);	
110	if (sleep > 0)	
111	usleep(sleep);	
112	}	
113		
114	TiledMap& World::getStaticMap() {	
115	return this->map;	
116	}	
117		
118	void World::stop() {	
119	this->keepTalking = false;	
120	for (auto & player : this->players) {	
121	player.second->stop();	
122	player.second->join();	
123	delete player.second;	
124	}	
125	}	

jul 21, 20 15:20	World.cpp	Page 3/3
126	void World::update() {	
127	gameObjectsContainer.update(board);	
128	}	
129		
130		
131	std::vector<std::shared_ptr<GameObject>> World::getUpdatedGameObjects() {	
132	return gameObjectsContainer.getUpdatedGameObjects();	
133	}	
134		
135	void World::addPlayer(ThPlayer *aPlayer, uint id) {	
136	aPlayer->start();	
137	players.insert({id, aPlayer});	
138	}	
139		
140	void World::clearFinishedPlayers() {	
141	auto iter = this->players.begin();	
142	while (iter != this->players.end()) {	
143	if (!(*iter).second->is_alive()) {	
144	(*iter).second->join();	
145	gameObjectsContainer.deleteGameObject((*iter).first, board);	
146	delete (*iter).second;	
147	iter = this->players.erase(iter);	
148	} else {	
149	iter++;	
150	}	
151	}	
152	}	
153		
154	void World::removeCreaturesAndItems() {	
155	gameObjectsContainer.removeCreaturesAndItems(board);	
156	}	
157		
158	void World::checkDrops() {	
159	for (auto &aGameObject : gameObjectsContainer.getUpdatedGameObjects()) {	
160	if (aGameObject->canDropsItems()) {	
161	for (auto &dropItem : aGameObject->getDrop()) {	
162	generateItem(dropItem, board.getNextEmptyCell(aGameObject->getAc	
163	tualCell()));	
164	}	
165	}	
166	}	
167		
168	World::~World() {	
169	delete priest;	
170	delete merchant;	
171	delete banker;	
172	}	
173		

jul 21, 20 15:20	ThPlayerReceiver.h	Page 1/1
1	<code>#ifndef THPLAYERRECEIVER_H</code>	
2	<code>#define THPLAYERRECEIVER_H</code>	
3		
4	<code>#include <atomic></code>	
5	<code>#include "../common/Thread.h"</code>	
6	<code>#include "../common/CommunicationProtocol.h"</code>	
7	<code>#include "../common/Decoder.h"</code>	
8	<code>#include "../common/InputQueue.h"</code>	
9		
10	<code>class ThPlayerReceiver: public Thread {</code>	
11	<code>private:</code>	
12	<code>std::atomic<bool> keepTalking;</code>	
13	<code>std::shared_ptr<CommunicationProtocol> protocol;</code>	
14	<code>InputQueue& queue;</code>	
15	<code>uint id{0};</code>	
16		
17	<code>public:</code>	
18	<code>explicit ThPlayerReceiver(std::shared_ptr<CommunicationProtocol> protocol, I</code>	
19	<code>nputQueue& queue);</code>	
20	<code>virtual void run();</code>	
21		
22	<code>void stop();</code>	
23		
24	<code>void setId(uint id);</code>	
25		
26	<code>virtual ~ThPlayerReceiver();</code>	
27	<code>};</code>	
28		
29	<code>#endif</code>	

jul 21, 20 15:20	ThPlayerReceiver.cpp	Page 1/1
1	<code>#include "ThPlayerReceiver.h"</code>	
2	<code>#include <iostream></code>	
3	<code>#include <utility></code>	
4	<code>#include "../common/SocketException.h"</code>	
5		
6	<code>#define UNKNOWN_ERROR "Unknow Error"</code>	
7	<code>#define ERRORSOCKET "Error en la comunicaci3n en ThPlayerReceiver::run() "</code>	
8	<code>#define ERRORRECEIVER "Error en ThPlayerReceiver::run() "</code>	
9		
10		
11	<code>ThPlayerReceiver::ThPlayerReceiver(std::shared_ptr<CommunicationProtocol> protoc</code>	
12	<code>ol, InputQueue& queue) :</code>	
13	<code>keepTalking(true), protocol(std::move(protocol)), queue(queue) {}</code>	
14	<code>void ThPlayerReceiver::setId(uint id) {</code>	
15	<code>this->id = id;</code>	
16	<code>}</code>	
17		
18	<code>void ThPlayerReceiver::run() {</code>	
19	<code>Message msg;</code>	
20	<code>InputInfo input;</code>	
21	<code>while(this->keepTalking) {</code>	
22	<code>try{</code>	
23	<code>msg = this->protocol->receive();</code>	
24	<code>input = Decoder::decodeCommand(msg);</code>	
25	<code>this->queue.push(std::move(input));</code>	
26	<code>} catch (const SocketException& e) {</code>	
27	<code>std::cerr << ERRORSOCKET << e.what() << std::endl;</code>	
28	<code>this->keepTalking = false;</code>	
29	<code>} catch(const std::exception& e){</code>	
30	<code>std::cerr << ERRORRECEIVER << e.what() << std::endl;</code>	
31	<code>this->keepTalking = false;</code>	
32	<code>}catch (...) {</code>	
33	<code>this->keepTalking = false;</code>	
34	<code>std::cerr << UNKNOWN_ERROR << std::endl;</code>	
35	<code>}</code>	
36	<code>}</code>	
37	<code>}</code>	
38		
39	<code>void ThPlayerReceiver::stop() {</code>	
40	<code>this->keepTalking = false;</code>	
41	<code>}</code>	
42		
43	<code>ThPlayerReceiver::~ThPlayerReceiver() = default;</code>	

jul 21, 20 15:20	ThPlayer.h	Page 1/1
1	<code>#ifndef SERVERPLAYER_H</code>	
2	<code>#define SERVERPLAYER_H</code>	
3		
4	<code>#include <atomic></code>	
5	<code>#include <mutex></code>	
6	<code>#include <condition_variable></code>	
7	<code>#include "../common/Thread.h"</code>	
8	<code>#include "../common/CommunicationProtocol.h"</code>	
9	<code>#include "../common/Socket.h"</code>	
10	<code>#include "ThPlayerReceiver.h"</code>	
11	<code>#include "../common/PlayerInfo.h"</code>	
12	<code>#include "GameCharacter.h"</code>	
13	<code>#include "WorldInfoQueue.h"</code>	
14		
15		
16	<code>class ThPlayer : public Thread {</code>	
17	<code>private:</code>	
18	<code> std::shared_ptr<CommunicationProtocol> protocol;</code>	
19	<code> std::atomic<bool> keepTalking;</code>	
20	<code> std::shared_ptr<GameCharacter> character;</code>	
21	<code> ThPlayerReceiver receiver;</code>	
22	<code> WorldInfoQueue worldInfoQueue;</code>	
23		
24	<code>public:</code>	
25	<code> ThPlayer(const std::shared_ptr<CommunicationProtocol>& protocol, std::shared_ptr<GameCharacter> aCharacter);</code>	
26		
27	<code> virtual void run();</code>	
28		
29	<code> void stop();</code>	
30		
31	<code> bool is_alive() const;</code>	
32		
33	<code> void update(const WorldInfo& worldInfo);</code>	
34		
35	<code> virtual ~ThPlayer();</code>	
36	<code>};</code>	
37		
38		
39	<code>#endif</code>	

jul 21, 20 15:20	ThPlayer.cpp	Page 1/1
1	<code>#include "ThPlayer.h"</code>	
2	<code>#include <iostream></code>	
3	<code>#include <utility></code>	
4	<code>#include "../common/SocketException.h"</code>	
5		
6	<code>#define UNKNOWN_ERROR "Unknow Error"</code>	
7	<code>#define ERRORSOCKET "Error en la comunicaci3n en ThPlayer::run() "</code>	
8	<code>#define ERRORDISPATCHER "Error en ThPlayer::run() "</code>	
9		
10	<code>ThPlayer::ThPlayer(const std::shared_ptr<CommunicationProtocol>& protocol, std::shared_ptr<GameCharacter> aCharacter) :</code>	
11	<code> protocol(protocol), keepTalking(true), character(std::move(aCharacter)),</code>	
12	<code> receiver(protocol, character->getQueueInputs()), worldInfoQueue(true) {}</code>	
13		
14	<code>void ThPlayer::run() {</code>	
15	<code> this->receiver.setId(character->getId());</code>	
16	<code> this->receiver.start();</code>	
17	<code> while (this->keepTalking) {</code>	
18	<code> try{</code>	
19	<code> WorldInfo worldInfo = worldInfoQueue.pop();</code>	
20	<code> const PlayerInfo& aPlayerInfo = worldInfo.getPlayerInfo();</code>	
21	<code> this->protocol->send(Decoder::encodePlayerInfo(aPlayerInfo));</code>	
22	<code> this->protocol->send(Decoder::encodeGameObjects(worldInfo.getGameObjsInfo()));</code>	
23	<code> if (aPlayerInfo.getState() == CharacterStateID::Interact ^ worldInfo.getNPCInfo().type != 0) {</code>	
24	<code> this->protocol->send(Decoder::encodeNPCInfo(worldInfo.getNPCInfo()));</code>	
25	<code> }</code>	
26	<code> } catch(const SocketException& e) {</code>	
27	<code> std::cout << ERRORSOCKET << e.what() << std::endl;</code>	
28	<code> this->stop();</code>	
29	<code> } catch(const std::exception& e) {</code>	
30	<code> std::cerr << ERRORDISPATCHER << e.what() << std::endl;</code>	
31	<code> this->stop();</code>	
32	<code> } catch (...) {</code>	
33	<code> std::cerr << UNKNOWN_ERROR << std::endl;</code>	
34	<code> this->stop();</code>	
35	<code> }</code>	
36	<code> }</code>	
37	<code>}</code>	
38		
39		
40	<code>void ThPlayer::stop() {</code>	
41	<code> this->keepTalking = false;</code>	
42	<code> this->receiver.stop();</code>	
43	<code> this->receiver.join();</code>	
44	<code> this->protocol->stop();</code>	
45	<code>}</code>	
46		
47	<code>bool ThPlayer::is_alive() const {</code>	
48	<code> return this->keepTalking;</code>	
49	<code>}</code>	
50		
51	<code>void ThPlayer::update(const WorldInfo& worldInfo) {</code>	
52	<code> worldInfoQueue.push(worldInfo);</code>	
53	<code>}</code>	
54		
55	<code>ThPlayer::~ThPlayer()= default;</code>	

jul 21, 20 15:20	ThLobbyPlayer.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_THLOBBYPLAYER_H</code>	
2	<code>#define ARGENTUM_TALLER_THLOBBYPLAYER_H</code>	
3		
4		
5	<code>#include "../common/Thread.h"</code>	
6	<code>#include "World.h"</code>	
7	<code>#include "../common/Socket.h"</code>	
8	<code>#include "../common/CommunicationProtocol.h"</code>	
9		
10	<code>class ThLobbyPlayer : public Thread {</code>	
11	<code>private:</code>	
12	<code>std::atomic<bool> keepTalking;</code>	
13	<code>World& world;</code>	
14	<code>std::shared_ptr<CommunicationProtocol> protocol;</code>	
15	<code>public:</code>	
16	<code>ThLobbyPlayer(Socket socket, World& world);</code>	
17	<code>void run() override;</code>	
18	<code>void stop();</code>	
19		
20	<code>void stop();</code>	
21		
22	<code>bool is_alive() const;</code>	
23		
24	<code>~ThLobbyPlayer() override;</code>	
25	<code>};</code>	
26		
27		
28	<code>#endif // ARGENTUM_TALLER_THLOBBYPLAYER_H</code>	

jul 21, 20 15:20	ThLobbyPlayer.cpp	Page 1/1
1	<code>#include <iostream></code>	
2	<code>#include <memory></code>	
3	<code>#include "ThLobbyPlayer.h"</code>	
4		
5	<code>#define INITMSG 0x04</code>	
6		
7	<code>void ThLobbyPlayer::run() {</code>	
8	<code>Message welcomeMsg = this->protocol->receive();</code>	
9	<code>std::shared_ptr<GameCharacter> aCharacter;</code>	
10	<code>if (welcomeMsg.getType() == INITMSG) {</code>	
11	<code>RaceID race = (RaceID) welcomeMsg.read8();</code>	
12	<code>GameClassID gameClass = (GameClassID) welcomeMsg.read8();</code>	
13	<code>aCharacter = world.createCharacter(race, gameClass);</code>	
14	<code>}</code>	
15	<code>std::vector<uint8_t> map = Decoder::encodeMap(this->world.getStaticMap());</code>	
16	<code>this->protocol->send(map);</code>	
17	<code>this->protocol->send(Decoder::encodePlayerInfo(aCharacter->getPlayerInfo()));</code>	
18	<code>world.addPlayer(new ThPlayer(protocol, aCharacter), aCharacter->getId());</code>	
19	<code>keepTalking = false;</code>	
20	<code>}</code>	
21		
22	<code>ThLobbyPlayer::~ThLobbyPlayer() = default;</code>	
23		
24	<code>ThLobbyPlayer::ThLobbyPlayer(Socket socket, World &world) :</code>	
25	<code>keepTalking(true), world(world) {</code>	
26	<code>protocol = std::make_shared<CommunicationProtocol>(std::move(socket));</code>	
27	<code>}</code>	
28		
29	<code>void ThLobbyPlayer::stop() {</code>	
30	<code>this->keepTalking = false;</code>	
31	<code>this->protocol->stop();</code>	
32	<code>}</code>	
33		
34	<code>bool ThLobbyPlayer::is_alive() const {</code>	
35	<code>return keepTalking;</code>	
36	<code>}</code>	

jul 21, 20 15:20	TakeAndDropStateCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER TAKEANDDROPSTATECHARACTER_H	
2	#define ARGENTUM_TALLER TAKEANDDROPSTATECHARACTER_H	
3		
4		
5	#include "StateCharacter.h"	
6		
7	class TakeAndDropStateCharacter : public StateCharacter {	
8	public:	
9	explicit TakeAndDropStateCharacter();	
10		
11	~TakeAndDropStateCharacter() override;	
12		
13	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec	
14	t>> &gameObjects, Board &board) override;	
15	StateID getNextStateID(InputInfo info) override;	
16	StateID getResetStateID() override;	
17		
18	bool isAttacking() override;	
19		
20		
21	void init(InputInfo aInputInfo) override;	
22		
23	bool isMeditating() override;	
24	};	
25		
26		
27	#endif //ARGENTUM_TALLER TAKEANDDROPSTATECHARACTER_H	

jul 21, 20 15:20	TakeAndDropStateCharacter.cpp	Page 1/1
1	#include "TakeAndDropStateCharacter.h"	
2	#include "../GameCharacter.h"	
3	#include "../ObjectItem.h"	
4		
5	TakeAndDropStateCharacter::~TakeAndDropStateCharacter() {	
6	stateId = StateID::TakeDrop;	
7	}	
8		
9	TakeAndDropStateCharacter::TakeAndDropStateCharacter() : StateCharacter() {}	
10		
11	void TakeAndDropStateCharacter::performTask(uint id, std::unordered_map<uint, st	
12	d::shared_ptr<GameObject>> &gameObjects, Board &board) {	
13		
14	std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<GameCh	
15	aracter>(gameObjects.at(id));	
16	std::shared_ptr<Cell> characterCell = aCharacter->getActualCell();	
17	switch (inputInfo.input) {	
18	case InputID::dropItem:	
19	if (!characterCell->hasItem()) {	
20	aCharacter->dropItem(inputInfo.additional - 1);	
21	break ;	
22	case InputID::takeItem:	
23	if (characterCell->hasItem()) {	
24	std::shared_ptr<ObjectItem> anItem = std::dynamic_pointer_cast<O	
25	bjectItem>(gameObjects.at(characterCell->getItemId()));	
26	if (aCharacter->takeItem(anItem->getItemId(), anItem->getAmount()))	
27	{	
28	anItem->take();	
29	}	
30	break ;	
31	default :	
32	break ;	
33	finalized = true;	
34	}	
35		
36	StateID TakeAndDropStateCharacter::getNextStateID(InputInfo info) {	
37	StateID nextState = StateID::Still;	
38	if (info.input == InputID::up ∨ info.input == InputID::down ∨	
39	info.input == InputID::left ∨ info.input == InputID::right) {	
40	nextState = StateID::Move;	
41	}	
42	return nextState;	
43	}	
44		
45	StateID TakeAndDropStateCharacter::getResetStateID() {	
46	return StateID::Still;	
47	}	
48		
49	bool TakeAndDropStateCharacter::isAttacking() {	
50	return false;	
51	}	
52		
53	bool TakeAndDropStateCharacter::isMeditating() {	
54	return false;	
55	}	
56		
57	void TakeAndDropStateCharacter::init(InputInfo aInputInfo) {	
58	inputInfo = aInputInfo;	
59	}	

jul 21, 20 15:20	StillStateCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STILLSTATECREATURE_H	
2	#define ARGENTUM_TALLER_STILLSTATECREATURE_H	
3		
4	#include "State.h"	
5	#include "StateCreature.h"	
6		
7	class StillStateCreature : public StateCreature {	
8	private:	
9	public:	
10	StillStateCreature();	
11	~StillStateCreature() override;	
12		
13	bool isOnPursuit(uint pursuitId) override;	
14		
15	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec	
16	t>> &gameObjects, Board &board) override;	
17		
18	StateID getNextStateID(InputInfo info) override;	
19	bool isAttacking() override;	
20		
21	StateID getResetStateID() override;	
22		
23	void init(InputInfo aInputInfo) override;	
24	};	
25		
26		
27	#endif //ARGENTUM_TALLER_STILLSTATECREATURE_H	

jul 21, 20 15:20	StillStateCreature.cpp	Page 1/1
1	#include "StillStateCreature.h"	
2		
3	StillStateCreature::StillStateCreature() : StateCreature() {	
4	finalized = true;	
5	stateId = StateID::Still;	
6	}	
7		
8	void StillStateCreature::performTask(uint id, std::unordered_map<uint, std::sha	
9	red_ptr<GameObject>> &gameObjects,	
10	Board &board) {}	
11		
12	StateID StillStateCreature::getNextStateID(InputInfo info) {	
13	return StateID::Move;	
14	}	
15	bool StillStateCreature::isOnPursuit(uint pursuitId) {	
16	return false;	
17	}	
18	bool StillStateCreature::isAttacking() {	
19	return false;	
20	}	
21		
22	void StillStateCreature::init(InputInfo aInputInfo) {	
23	inputInfo = aInputInfo;	
24	}	
25		
26	StateID StillStateCreature::getResetStateID() {	
27	return StateCreature::getResetStateID();	
28	}	
29		
30		
31	StillStateCreature::~StillStateCreature() = default ;	

jul 21, 20 15:20	StillStateCharacter.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_STILLSTATECHARACTER_H</code>	
2	<code>#define ARGENTUM_TALLER_STILLSTATECHARACTER_H</code>	
3		
4	<code>#include "StateCharacter.h"</code>	
5		
6	<code>class StillStateCharacter: public StateCharacter {</code>	
7	<code>public:</code>	
8	<code>StillStateCharacter();</code>	
9		
10	<code>~StillStateCharacter() override;</code>	
11		
12	<code>StateID getNextStateID(InputInfo info) override;</code>	
13	<code>StateID getResetStateID() override;</code>	
14		
15	<code>void init(InputInfo aInputInfo) override;</code>	
16		
17	<code>bool isMeditating() override;</code>	
18		
19	<code>bool isAttacking() override;</code>	
20		
21	<code>void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec</code>	
22	<code>t>> &gameObjects, Board &board) override;</code>	
23	<code>};</code>	
24		
25		
26		
27	<code>#endif //ARGENTUM_TALLER_STILLSTATECHARACTER_H</code>	

jul 21, 20 15:20	StillStateCharacter.cpp	Page 1/1
1	<code>#include "StillStateCharacter.h"</code>	
2	<code>#include <iostream></code>	
3		
4	<code>StillStateCharacter::~StillStateCharacter() = default;</code>	
5		
6	<code>StillStateCharacter::StillStateCharacter() : StateCharacter() {</code>	
7	<code>finalized = true;</code>	
8	<code>stateId = StateID::Still;</code>	
9	<code>}</code>	
10		
11	<code>void StillStateCharacter::performTask(uint id,</code>	
12	<code>std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Boar</code>	
13	<code>d &board) {}</code>	
14	<code>StateID StillStateCharacter::getNextStateID(InputInfo info) {</code>	
15	<code>StateID nextStateId = StateID::Still;</code>	
16	<code>if (info.input == InputID::up ∨ info.input == InputID::down ∨</code>	
17	<code>info.input == InputID::left ∨ info.input == InputID::right) {</code>	
18	<code>nextStateId = StateID::Move;</code>	
19	<code>} else if (info.input == InputID::selectTarget) {</code>	
20	<code>nextStateId = StateID::Transition;</code>	
21	<code>} else if (info.input == InputID::equipItem ∨ info.input == InputID::unequipIt</code>	
22	<code>em) {</code>	
23	<code>nextStateId = StateID::Equip;</code>	
24	<code>} else if (info.input == InputID::takeItem ∨ info.input == InputID::dropItem)</code>	
25	<code>{</code>	
26	<code>nextStateId = StateID::TakeDrop;</code>	
27	<code>} else if (info.input == InputID::meditate) {</code>	
28	<code>nextStateId = StateID::Meditate;</code>	
29	<code>} else if (info.input == InputID::resurrect) {</code>	
30	<code>nextStateId = StateID::Resurrect;</code>	
31	<code>}</code>	
32	<code>return nextStateId;</code>	
33	<code>StateID StillStateCharacter::getResetStateID() {</code>	
34	<code>return StateID::Still;</code>	
35	<code>}</code>	
36		
37	<code>bool StillStateCharacter::isAttacking() {</code>	
38	<code>return false;</code>	
39	<code>}</code>	
40		
41	<code>bool StillStateCharacter::isMeditating() {</code>	
42	<code>return false;</code>	
43	<code>}</code>	
44		
45	<code>void StillStateCharacter::init(InputInfo aInputInfo) {</code>	
46	<code>inputInfo = aInputInfo;</code>	
47	<code>}</code>	
48		

jul 21, 20 15:20	StateTranslator.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_STATETRANSLATOR_H</code>	
2	<code>#define ARGENTUM_TALLER_STATETRANSLATOR_H</code>	
3		
4		
5	<code>#include "../common/Identificators.h"</code>	
6		
7	<code>class StateTranslator {</code>	
8	<code>public:</code>	
9	<code>StateTranslator();</code>	
10		
11	<code>static CharacterStateID stateToCharacterState(StateID stateId);</code>	
12		
13	<code>virtual ~StateTranslator();</code>	
14	<code>};</code>	
15		
16		
17	<code>#endif //ARGENTUM_TALLER_STATETRANSLATOR_H</code>	

jul 21, 20 15:20	StateTranslator.cpp	Page 1/1
1	<code>#include "StateTranslator.h"</code>	
2		
3	<code>StateTranslator::StateTranslator() = default;</code>	
4		
5	<code>StateTranslator::~StateTranslator() = default;</code>	
6		
7	<code>CharacterStateID StateTranslator::stateToCharacterState(StateID stateId) {</code>	
8	<code>CharacterStateID characterStateId = CharacterStateID::Still;</code>	
9	<code>switch (stateId) {</code>	
10	<code>case StateID::Still:</code>	
11	<code>characterStateId = CharacterStateID::Still;</code>	
12	<code>break;</code>	
13	<code>case StateID::Move:</code>	
14	<code>characterStateId = CharacterStateID::Move;</code>	
15	<code>break;</code>	
16	<code>case StateID::Equip:</code>	
17	<code>characterStateId = CharacterStateID::Still;</code>	
18	<code>break;</code>	
19	<code>case StateID::Transition:</code>	
20	<code>characterStateId = CharacterStateID::Still;</code>	
21	<code>break;</code>	
22	<code>case StateID::Interact:</code>	
23	<code>characterStateId = CharacterStateID::Interact;</code>	
24	<code>break;</code>	
25	<code>case StateID::Attack:</code>	
26	<code>characterStateId = CharacterStateID::Attack;</code>	
27	<code>break;</code>	
28	<code>case StateID::Pursuit:</code>	
29	<code>characterStateId = CharacterStateID::Move;</code>	
30	<code>break;</code>	
31	<code>case StateID::Meditate:</code>	
32	<code>characterStateId = CharacterStateID::Meditate;</code>	
33	<code>break;</code>	
34	<code>case StateID::TakeDrop:</code>	
35	<code>characterStateId = CharacterStateID::Still;</code>	
36	<code>break;</code>	
37	<code>case StateID::Resurrect:</code>	
38	<code>characterStateId = CharacterStateID::Resurrect;</code>	
39	<code>break;</code>	
40	<code>}</code>	
41	<code>return characterStateId;</code>	
42	<code>}</code>	

jul 21, 20 15:20	StatePool.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STATEPOOL_H	
2	#define ARGENTUM_TALLER_STATEPOOL_H	
3		
4	#include <GameObject.h>	
5	#include "State.h"	
6		
7	class StatePool {	
8	public:	
9	StatePool();	
10		
11	virtual void changeState(StateID id, InputInfo aInputInfo) = 0;	
12		
13	virtual void setNextState(StateID id, InputInfo aInputInfo) = 0;	
14		
15	virtual void updateState() = 0;	
16		
17	virtual void performTask(std::unordered_map<uint, std::shared_ptr<GameObject>	
18	>> &gameObjects,	
19	Board &board) = 0;	
20		
21	virtual StateID getStateId() = 0;	
22	virtual ~StatePool();	
23	};	
24		
25	#endif //ARGENTUM_TALLER_STATEPOOL_H	

jul 21, 20 15:20	StatePoolCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STATEPOOLCREATURE_H	
2	#define ARGENTUM_TALLER_STATEPOOLCREATURE_H	
3		
4		
5	#include "StatePool.h"	
6	#include "StateCreature.h"	
7		
8	class StatePoolCreature : public StatePool {	
9	private:	
10	GameObject& creature;	
11	std::shared_ptr<StateCreature> actualState;	
12	std::unordered_map<StateID, std::shared_ptr<StateCreature>, std::hash<StateID>	
13	>> states;	
14	public:	
15	explicit StatePoolCreature(GameObject &creature);	
16		
17	void updateState() override;	
18		
19	bool startChasing(uint pursuitId);	
20		
21	std::shared_ptr<StateCreature> generateState(StateID stateId);	
22	~StatePoolCreature() override;	
23		
24	void changeState(StateID id, InputInfo aInputInfo) override;	
25		
26	void setNextState(StateID id, InputInfo aInputInfo) override;	
27		
28	StateID getStateId() override;	
29		
30	void performTask(std::unordered_map<uint, std::shared_ptr<GameObject>> &game	
31	Objects, Board &board) override;	
32	};	
33		
34		
35	#endif //ARGENTUM_TALLER_STATEPOOLCREATURE_H	

jul 21, 20 15:20	StatePoolCreature.cpp	Page 1/2
1	<code>#include "StatePoolCreature.h"</code>	
2	<code>#include "StillStateCreature.h"</code>	
3	<code>#include "AttackStateCreature.h"</code>	
4	<code>#include "MoveStateCreature.h"</code>	
5	<code>#include "PursuitStateCreature.h"</code>	
6		
7	<code>StatePoolCreature::StatePoolCreature(GameObject &aCreature) : creature(aCreature</code>	
8	<code>),</code>	
9	<code>actualState(std::shared_ptr<StateCreature>(new StillStateCreature())) {</code>	
10	<code>states.insert(std::pair<StateID,</code>	
11	<code>std::shared_ptr<StateCreature>>(actualState->getStateId(), actualSta</code>	
12	<code>te));</code>	
13	<code>}</code>	
14	<code>void StatePoolCreature::updateState() {</code>	
15	<code>if (actualState->isOver()) {</code>	
16	<code>InputInfo aInputInfo;</code>	
17	<code>if (actualState->getStateId() == StateID::Pursuit actualState->getState</code>	
18	<code>Id() == StateID::Attack) {</code>	
19	<code>aInputInfo = actualState->getInputInfo();</code>	
20	<code>} else {</code>	
21	<code>aInputInfo = creature.getNextInputInfo();</code>	
22	<code>}</code>	
23	<code>StateID nextStateId = actualState->getNextStateID(aInputInfo);</code>	
24	<code>setNextState(nextStateId, aInputInfo);</code>	
25	<code>}</code>	
26	<code>void StatePoolCreature::setNextState(StateID id, InputInfo aInputInfo) {</code>	
27	<code>changeState(id, aInputInfo);</code>	
28	<code>}</code>	
29		
30	<code>void StatePoolCreature::changeState(StateID id, InputInfo aInputInfo) {</code>	
31	<code>std::shared_ptr<StateCreature> nextState;</code>	
32	<code>try {</code>	
33	<code>nextState = states.at(id);</code>	
34	<code>} catch (std::exception &e) {</code>	
35	<code>nextState = generateState(id);</code>	
36	<code>}</code>	
37	<code>nextState->init(aInputInfo);</code>	
38	<code>actualState = nextState;</code>	
39	<code>}</code>	
40		
41	<code>std::shared_ptr<StateCreature> StatePoolCreature::generateState(StateID stateId)</code>	
42	<code>{</code>	
43	<code>std::shared_ptr<StateCreature> newState;</code>	
44	<code>switch (stateId) {</code>	
45	<code>case StateID::Move:</code>	
46	<code>newState = std::shared_ptr<StateCreature>(new MoveStateCreature());</code>	
47	<code>break;</code>	
48	<code>case StateID::Attack:</code>	
49	<code>newState = std::shared_ptr<StateCreature>(new AttackStateCreature());</code>	
50	<code>break;</code>	
51	<code>case StateID::Pursuit:</code>	
52	<code>newState = std::shared_ptr<StateCreature>(new PursuitStateCreature());</code>	
53	<code>break;</code>	
54	<code>default:</code>	
55	<code>break;</code>	
56	<code>}</code>	
57	<code>states.insert(std::pair<StateID,</code>	
58	<code>std::shared_ptr<StateCreature>>(newState->getStateId(), newState));</code>	
59	<code>return newState;</code>	
60	<code>}</code>	

jul 21, 20 15:20	StatePoolCreature.cpp	Page 2/2
61	<code>bool StatePoolCreature::startChasing(uint pursuitId) {</code>	
62	<code>return ~actualState->isAttacking() ^ ~actualState->isOnPursuit(pursuitId);</code>	
63	<code>}</code>	
64		
65	<code>StateID StatePoolCreature::getStateId() {</code>	
66	<code>return actualState->getStateId();</code>	
67	<code>}</code>	
68		
69	<code>void StatePoolCreature::performTask(std::unordered_map<uint, std::shared_ptr<Gam</code>	
70	<code>eObject>> &gameObjects, Board &board) {</code>	
71	<code>actualState->performTask(creature.getId(), gameObjects, board);</code>	
72	<code>}</code>	
73	<code>StatePoolCreature::~StatePoolCreature() = default;</code>	

jul 21, 20 15:20	StatePool.cpp	Page 1/1
1	<code>#include "StatePool.h"</code>	
2		
3	<code>StatePool::~StatePool() = default;</code>	
4		
5	<code>StatePool::StatePool() = default;</code>	

jul 21, 20 15:20	StatePoolCharacter.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_STATEPOOLCHARACTER_H</code>	
2	<code>#define ARGENTUM_TALLER_STATEPOOLCHARACTER_H</code>	
3		
4	<code>#include "StatePool.h"</code>	
5	<code>#include "StateCharacter.h"</code>	
6		
7	<code>class StatePoolCharacter : public StatePool {</code>	
8	<code>private:</code>	
9	<code> GameObject& character;</code>	
10	<code> std::shared_ptr<StateCharacter> actualState;</code>	
11	<code> std::unordered_map<StateID, std::shared_ptr<StateCharacter>, std::hash<StateID>> states;</code>	
12	<code>public:</code>	
13	<code> explicit StatePoolCharacter(GameObject &aCharacter);</code>	
14		
15	<code> void updateState() override;</code>	
16		
17	<code> ~StatePoolCharacter() override;</code>	
18		
19	<code> void performTask(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) override;</code>	
20		
21	<code> bool isPossibleDeadState(StateID id);</code>	
22		
23	<code> bool isMeditating();</code>	
24		
25	<code> void changeState(StateID id, InputInfo aInputInfo) override;</code>	
26		
27	<code> void setNextState(StateID id, InputInfo aInputInfo) override;</code>	
28		
29	<code> StateID getStateId() override;</code>	
30		
31	<code> std::shared_ptr<StateCharacter> generateState(StateID id);</code>	
32	<code>};</code>	
33		
34		
35	<code>#endif // ARGENTUM_TALLER_STATEPOOLCHARACTER_H</code>	

jul 21, 20 15:20	StatePoolCharacter.cpp	Page 1/2
1	<code>#include "StatePoolCharacter.h"</code>	
2	<code>#include "StillStateCharacter.h"</code>	
3	<code>#include "MoveStateCharacter.h"</code>	
4	<code>#include "EquipStateCharacter.h"</code>	
5	<code>#include "InteractStateCharacter.h"</code>	
6	<code>#include "AttackStateCharacter.h"</code>	
7	<code>#include "MeditateStateCharacter.h"</code>	
8	<code>#include "TakeAndDropStateCharacter.h"</code>	
9	<code>#include "ResurrectStateCharacter.h"</code>	
10		
11	<code>StatePoolCharacter::StatePoolCharacter(GameObject &aCharacter) : character(aCharacter),</code>	
12	<code>actualState(s</code>	
13	<code>td::shared_ptr<StateCharacter>(new StillStateCharacter())) {</code>	
14	<code>states.insert(std::pair<StateID,</code>	
15	<code>std::shared_ptr<StateCharacter>>(actualState->getStateId(), actualSt</code>	
16	<code>ate));</code>	
17	<code>}</code>	
18	<code>void StatePoolCharacter::updateState() {</code>	
19	<code>if (actualState->isOver()) {</code>	
20	<code>StateID nextStateId;</code>	
21	<code>InputInfo aInputInfo = actualState->getInputInfo();</code>	
22	<code>if (character.hasAnInputInfo()) {</code>	
23	<code>aInputInfo = character.getNextInputInfo();</code>	
24	<code>nextStateId = actualState->getNextStateID(aInputInfo);</code>	
25	<code>} else {</code>	
26	<code>nextStateId = actualState->getResetStateID();</code>	
27	<code>}</code>	
28	<code>setNextState(nextStateId, aInputInfo);</code>	
29	<code>}</code>	
30		
31	<code>void StatePoolCharacter::setNextState(StateID id, InputInfo aInputInfo) {</code>	
32	<code>if (id == StateID::Transition) {</code>	
33	<code>id = character.getActualCell()->isCity() ? StateID::Interact : StateID::</code>	
34	<code>Attack;</code>	
35	<code>}</code>	
36	<code>if (character.isDead() ^ ~isPossibleDeadState(id)) {</code>	
37	<code>return;</code>	
38	<code>changeState(id, aInputInfo);</code>	
39	<code>}</code>	
40		
41	<code>bool StatePoolCharacter::isPossibleDeadState(StateID id) {</code>	
42	<code>return id == StateID::Still v id == StateID::Move v id == StateID::Interact v i</code>	
43	<code>d == StateID::Resurrect;</code>	
44	<code>}</code>	
45	<code>void StatePoolCharacter::changeState(StateID id, InputInfo aInputInfo) {</code>	
46	<code>std::shared_ptr<StateCharacter> nextState;</code>	
47	<code>try {</code>	
48	<code>nextState = states.at(id);</code>	
49	<code>} catch (std::exception &e) {</code>	
50	<code>nextState = generateState(id);</code>	
51	<code>}</code>	
52	<code>nextState->init(aInputInfo);</code>	
53	<code>actualState = nextState;</code>	
54	<code>}</code>	
55		
56	<code>std::shared_ptr<StateCharacter> StatePoolCharacter::generateState(StateID stateI</code>	
57	<code>d) {</code>	
58	<code>std::shared_ptr<StateCharacter> newState;</code>	
59	<code>switch (stateId) {</code>	
60	<code>case StateID::Move:</code>	
61	<code>newState = std::shared_ptr<StateCharacter>(new MoveStateCharacter())</code>	

jul 21, 20 15:20	StatePoolCharacter.cpp	Page 2/2
61	<code>;</code>	
62	<code>break;</code>	
63	<code>case StateID::Equip:</code>	
64	<code>newState = std::shared_ptr<StateCharacter>(new EquipStateCharacter())</code>	
65	<code>);</code>	
66	<code>break;</code>	
67	<code>case StateID::Interact:</code>	
68	<code>newState = std::shared_ptr<StateCharacter>(new InteractStateCharacter</code>	
69	<code>r());</code>	
70	<code>break;</code>	
71	<code>case StateID::Attack:</code>	
72	<code>newState = std::shared_ptr<StateCharacter>(new AttackStateCharacter(</code>	
73	<code>r());</code>	
74	<code>break;</code>	
75	<code>case StateID::Meditate:</code>	
76	<code>newState = std::shared_ptr<StateCharacter>(new MeditateStateCharacter</code>	
77	<code>r());</code>	
78	<code>break;</code>	
79	<code>case StateID::TakeDrop:</code>	
80	<code>newState = std::shared_ptr<StateCharacter>(new TakeAndDropStateChara</code>	
81	<code>cter());</code>	
82	<code>break;</code>	
83	<code>case StateID::Resurrect:</code>	
84	<code>newState = std::shared_ptr<StateCharacter>(new ResurrectStateCharact</code>	
85	<code>er());</code>	
86	<code>break;</code>	
87	<code>default:</code>	
88	<code>break;</code>	
89	<code>states.insert(std::pair<StateID,</code>	
90	<code>std::shared_ptr<StateCharacter>>(newState->getStateId(), newState));</code>	
91	<code>return newState;</code>	
92	<code>}</code>	
93	<code>StateID StatePoolCharacter::getStateId() {</code>	
94	<code>return actualState->getStateId();</code>	
95	<code>}</code>	
96	<code>void StatePoolCharacter::performTask(std::unordered_map<uint, std::shared_ptr<Ga</code>	
97	<code>meObject>> &gameObjects, Board &board) {</code>	
98	<code>actualState->performTask(character.getId(), gameObjects, board);</code>	
99	<code>}</code>	
100	<code>bool StatePoolCharacter::isMeditating() {</code>	
101	<code>return actualState->isMeditating();</code>	
102	<code>}</code>	
103	<code>StatePoolCharacter::~StatePoolCharacter() = default;</code>	

jul 21, 20 15:20	State.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STATE_H	
2	#define ARGENTUM_TALLER_STATE_H	
3		
4	#include <GameObject.h>	
5	#include "../common/Identificators.h"	
6		
7	class State {	
8	protected:	
9	bool finalized = false;	
10	InputInfo inputInfo;	
11	StateID stateId;	
12	public:	
13	State();	
14		
15	virtual ~State();	
16		
17	StateID getStateId() const ;	
18		
19	const InputInfo &getInputInfo() const ;	
20		
21	bool isOver() const ;	
22		
23	virtual void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) = 0;	
24		
25	virtual StateID getNextStateID(InputInfo aInputInfo) = 0;	
26		
27	virtual StateID getResetStateID() = 0;	
28		
29	virtual bool isAttacking() = 0;	
30		
31	virtual void init(InputInfo aInputInfo) = 0;	
32	};	
33		
34		
35	#endif //ARGENTUM_TALLER_STATE_H	

jul 21, 20 15:20	StateCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STATECREATURE_H	
2	#define ARGENTUM_TALLER_STATECREATURE_H	
3		
4	#include "State.h"	
5		
6	class StateCreature : public State {	
7	public:	
8	~StateCreature() override;	
9		
10	StateID getResetStateID() override;	
11		
12	virtual bool isOnPursuit(uint pursuitId) = 0;	
13	};	
14		
15		
16	#endif //ARGENTUM_TALLER_STATECREATURE_H	

jul 21, 20 15:20	StateCreature.cpp	Page 1/1
1	<code>#include "StateCreature.h"</code>	
2		
3	<code>StateCreature::~StateCreature() = default;</code>	
4		
5	<code>StateID StateCreature::getResetStateID() {</code>	
6	<code> return StateID::Still;</code>	
7	<code>}</code>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	State.cpp	Page 1/1
1	<code>#include "State.h"</code>	
2		
3	<code>State::State() {</code>	
4	<code> InputInfo info;</code>	
5	<code> info.input = InputID::nothing;</code>	
6	<code> inputInfo = info;</code>	
7	<code> stateId = StateID::Still;</code>	
8	<code>}</code>	
9		
10	<code>State::~State() = default;</code>	
11		
12	<code>bool State::isOver() const {</code>	
13	<code> return finalized;</code>	
14	<code>}</code>	
15		
16	<code>StateID State::getStateId() const {</code>	
17	<code> return stateId;</code>	
18	<code>}</code>	
19		
20	<code>const InputInfo &State::getInputInfo() const {</code>	
21	<code> return inputInfo;</code>	
22	<code>}</code>	
23		
24		
25		
26		

17/217

jul 21, 20 15:20	StateCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_STATECHARACTER_H	
2	#define ARGENTUM_TALLER_STATECHARACTER_H	
3		
4	#include "State.h"	
5		
6	class StateCharacter : public State {	
7	public:	
8	StateCharacter();	
9		
10	~StateCharacter() override;	
11		
12	virtual bool isMeditating() = 0;	
13	};	
14		
15		
16	#endif //ARGENTUM_TALLER_STATECHARACTER_H	

jul 21, 20 15:20	StateCharacter.cpp	Page 1/1
1	#include "StateCharacter.h"	
2		
3	StateCharacter::~StateCharacter() = default ;	
4		
5	StateCharacter::StateCharacter() = default ;	
6		

jul 21, 20 15:20	ResurrectStateCharacter.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_RESURRECTSTATECHARACTER_H</code>	
2	<code>#define ARGENTUM_TALLER_RESURRECTSTATECHARACTER_H</code>	
3		
4	<code>#include <MovementCharacter.h></code>	
5	<code>#include "StateCharacter.h"</code>	
6		
7	<code>class ResurrectStateCharacter : public StateCharacter {</code>	
8	<code>private:</code>	
9	<code>std::shared_ptr<Cell> aPriestCell = nullptr;</code>	
10	<code>MovementCharacter movement;</code>	
11	<code>public:</code>	
12	<code>ResurrectStateCharacter();</code>	
13	<code>~ResurrectStateCharacter() override;</code>	
14		
15	<code>void</code>	
16	<code>performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject>> &</code>	
17	<code>gameObjects, Board &board) override;</code>	
18	<code>StateID getNextStateID(InputInfo info) override;</code>	
19		
20	<code>StateID getResetStateID() override;</code>	
21		
22	<code>void init(InputInfo aInputInfo) override;</code>	
23		
24	<code>bool isAttacking() override;</code>	
25		
26	<code>bool isMeditating() override;</code>	
27	<code>};</code>	
28		
29		
30		
31	<code>#endif //ARGENTUM_TALLER_RESURRECTSTATECHARACTER_H</code>	

jul 21, 20 15:20	ResurrectStateCharacter.cpp	Page 1/2
1	<code>#include <GameCharacter.h></code>	
2	<code>#include "ResurrectStateCharacter.h"</code>	
3		
4	<code>ResurrectStateCharacter::~ResurrectStateCharacter() = default;</code>	
5		
6	<code>ResurrectStateCharacter::ResurrectStateCharacter() : StateCharacter() {</code>	
7	<code>finalized = false;</code>	
8	<code>stateId = StateID::Resurrect;</code>	
9	<code>}</code>	
10		
11	<code>void ResurrectStateCharacter::performTask(uint id, std::unordered_map<uint, std::</code>	
12	<code>shared_ptr<GameObject>> &gameObjects, Board &board) {</code>	
13	<code>std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<GameCh</code>	
14	<code>aracter>(gameObjects.at(id));</code>	
15	<code>std::shared_ptr<Cell> characterCell = aCharacter->getActualCell();</code>	
16	<code>if (!aCharacter->isDead()) {</code>	
17	<code>finalized = true;</code>	
18	<code>return;</code>	
19	<code>}</code>	
20	<code>if (aPriestCell == nullptr) {</code>	
21	<code>aPriestCell = board.getCloserPriest(characterCell);</code>	
22	<code>}</code>	
23	<code>if (!movement.hasStart()) {</code>	
24	<code>if (board.getDistance(characterCell, aPriestCell) == 2) {</code>	
25	<code>finalized = true;</code>	
26	<code>aCharacter->setDirection(board.getDirection(characterCell, aPrie</code>	
27	<code>stCell));</code>	
28	<code>aCharacter->cure();</code>	
29	<code>} else {</code>	
30	<code>std::shared_ptr<Cell> newCell;</code>	
31	<code>newCell = board.getBestCell(characterCell, aPriestCell, true);</code>	
32	<code>if (newCell != characterCell) {</code>	
33	<code>Direction aDirection = board.getDirection(characterCell, new</code>	
34	<code>Cell);</code>	
35	<code>aCharacter->setDirection(aDirection);</code>	
36	<code>movement.start(board.getPointFromCell(aCharacter->getActualC</code>	
37	<code>ell()), aDirection,</code>	
38	<code>aCharacter->getActualCell()->free();</code>	
39	<code>newCell->occupied(id);</code>	
40	<code>aCharacter->setCell(newCell);</code>	
41	<code>}</code>	
42	<code>} else {</code>	
43	<code>aCharacter->setPoint(movement.doStep());</code>	
44	<code>if (movement.isOver()) {</code>	
45	<code>movement.reset();</code>	
46	<code>}</code>	
47	<code>}</code>	
48	<code>StateID ResurrectStateCharacter::getNextStateID(InputInfo info) {</code>	
49	<code>return StateID::Still;</code>	
50	<code>}</code>	
51	<code>StateID ResurrectStateCharacter::getResetStateID() {</code>	
52	<code>return StateID::Still;</code>	
53	<code>}</code>	
54		
55	<code>bool ResurrectStateCharacter::isAttacking() {</code>	
56	<code>return false;</code>	
57	<code>}</code>	
58		
59	<code>bool ResurrectStateCharacter::isMeditating() {</code>	
60	<code>return false;</code>	
61	<code>}</code>	

jul 21, 20 15:20	ResurrectStateCharacter.cpp	Page 2/2
62	}	
63		
64	void ResurrectStateCharacter::init(InputInfo aInputInfo) {	
65	aPriestCell = nullptr;	
66	finalized = false;	
67	movement.reset();	
68	}	

jul 21, 20 15:20	PursuitStateCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_PURSUITSTATECREATURE_H	
2	#define ARGENTUM_TALLER_PURSUITSTATECREATURE_H	
3		
4		
5	#include <MovementCreature.h>	
6	#include "State.h"	
7	#include "../Creature.h"	
8	#include "StateCreature.h"	
9		
10	class PursuitStateCreature : public StateCreature {	
11	private:	
12	uint pursuitId;	
13	MovementCreature movement;	
14	bool canAttack = false;	
15		
16	public:	
17	explicit PursuitStateCreature();	
18		
19	~PursuitStateCreature() override;	
20		
21	bool isOnPursuit(uint pursuitId) override;	
22		
23	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec	
24	t>> &gameObjects, Board &board) override;	
25	StateID getNextStateID(InputInfo info) override;	
26		
27	StateID getResetStateID() override;	
28		
29	void init(InputInfo aInputInfo) override;	
30		
31	bool isAttacking() override;	
32	};	
33		
34		
35	#endif //ARGENTUM_TALLER_PURSUITSTATECREATURE_H	

jul 21, 20 15:20	PursuitStateCreature.cpp	Page 1/2
1	<code>#include "PursuitStateCreature.h"</code>	
2	<code>#include "StillStateCreature.h"</code>	
3	<code>#include "../GameCharacter.h"</code>	
4		
5	<code>PursuitStateCreature::~PursuitStateCreature() = default;</code>	
6		
7	<code>PursuitStateCreature::PursuitStateCreature() : StateCreature(), pursuitId(0) {</code>	
8	<code> stateId = StateID::Pursuit;</code>	
9	<code> finalized = false;</code>	
10	<code>}</code>	
11		
12	<code>void PursuitStateCreature::performTask(uint id, std::unordered_map<uint, std::sh</code>	
13	<code>ared_ptr<GameObject>> &gameObjects, Board &board) {</code>	
14	<code> std::shared_ptr<Creature> aCreature = std::dynamic_pointer_cast<Creature>(ga</code>	
15	<code>meObjects.at(id));</code>	
16	<code> try {</code>	
17	<code> std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<Ga</code>	
18	<code>meCharacter>(gameObjects.at(pursuitId));</code>	
19	<code> std::shared_ptr<Cell> creatureCell = aCreature->getActualCell();</code>	
20	<code> std::shared_ptr<Cell> enemyCell = aCharacter->getActualCell();</code>	
21	<code> if (!movement.hasStart()) {</code>	
22	<code> if (enemyCell->getNestId() == creatureCell->getNestId()) {</code>	
23	<code> if (board.getDistance(creatureCell, enemyCell) == 1) {</code>	
24	<code> canAttack = true;</code>	
25	<code> finalized = true;</code>	
26	<code> aCreature->setDirection(board.getDirection(creatureCell, ene</code>	
27	<code>myCell));</code>	
28	<code> } else {</code>	
29	<code> std::shared_ptr<Cell> newCell;</code>	
30	<code> newCell = board.getBestCell(creatureCell, enemyCell, false);</code>	
31	<code> if (newCell != aCreature->getActualCell()) {</code>	
32	<code> Direction aDirection = board.getDirection(creatureCell,</code>	
33	<code>newCell);</code>	
34	<code> aCreature->setDirection(aDirection);</code>	
35	<code> movement.start(board.getPointFromCell(aCreature->getActu</code>	
36	<code>alCell()), aDirection, aCreature->getCreatureId());</code>	
37	<code> aCreature->getActualCell()->free();</code>	
38	<code> newCell->occupied(id);</code>	
39	<code> aCreature->setCell(newCell);</code>	
40	<code> }</code>	
41	<code> } else {</code>	
42	<code> finalized = true;</code>	
43	<code> }</code>	
44	<code> } else {</code>	
45	<code> aCreature->setPoint(movement.doStep());</code>	
46	<code> if (movement.isOver()) {</code>	
47	<code> movement.reset();</code>	
48	<code> }</code>	
49	<code> }</code>	
50	<code> } catch (const std::out_of_range& e) {</code>	
51	<code> canAttack = false;</code>	
52	<code> finalized = true;</code>	
53	<code> }</code>	
54	<code>bool PursuitStateCreature::isOnPursuit(uint aPursuitId) {</code>	
55	<code> return pursuitId == aPursuitId;</code>	
56	<code>}</code>	
57	<code>bool PursuitStateCreature::isAttacking() {</code>	
58	<code> return false;</code>	
59	<code>}</code>	
60		

jul 21, 20 15:20	PursuitStateCreature.cpp	Page 2/2
61	<code>StateID PursuitStateCreature::getNextStateID(InputInfo info) {</code>	
62	<code> StateID nextStateId = StateID::Still;</code>	
63	<code> if (canAttack) {</code>	
64	<code> nextStateId = StateID::Attack;</code>	
65	<code> }</code>	
66	<code> return nextStateId;</code>	
67	<code>}</code>	
68		
69	<code>StateID PursuitStateCreature::getResetStateID() {</code>	
70	<code> return StateID::Pursuit;</code>	
71	<code>}</code>	
72		
73	<code>void PursuitStateCreature::init(InputInfo aInputInfo) {</code>	
74	<code> inputInfo = aInputInfo;</code>	
75	<code> pursuitId = aInputInfo.additional;</code>	
76	<code> movement.reset();</code>	
77	<code> canAttack = false;</code>	
78	<code> finalized = false;</code>	
79	<code>}</code>	

jul 21, 20 15:20	MoveStateCreature.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_MOVESTATECREATURE_H</code>	
2	<code>#define ARGENTUM_TALLER_MOVESTATECREATURE_H</code>	
3		
4		
5	<code>#include <MovementCreature.h></code>	
6	<code>#include "State.h"</code>	
7	<code>#include "StateCreature.h"</code>	
8		
9	<code>class MoveStateCreature : public StateCreature {</code>	
10	<code>private:</code>	
11	<code>Direction direction;</code>	
12	<code>MovementCreature movement;</code>	
13	<code>public:</code>	
14	<code>MoveStateCreature();</code>	
15	<code>~MoveStateCreature() override;</code>	
16	<code>void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec</code>	
17	<code>t>> &gameObjects, Board &board) override;</code>	
18		
19	<code>bool isOnPursuit(uint pursuitId) override;</code>	
20		
21	<code>void init(InputInfo aInputInfo) override;</code>	
22		
23	<code>StateID getNextStateID(InputInfo info) override;</code>	
24	<code>StateID getResetStateID() override;</code>	
25		
26	<code>bool isAttacking() override;</code>	
27	<code>};</code>	
28		
29		
30		
31	<code>#endif //ARGENTUM_TALLER_MOVESTATECREATURE_H</code>	

jul 21, 20 15:20	MoveStateCreature.cpp	Page 1/2
1	<code>#include <iostream></code>	
2	<code>#include "MoveStateCreature.h"</code>	
3	<code>#include "StillStateCreature.h"</code>	
4	<code>#include "../Creature.h"</code>	
5		
6	<code>MoveStateCreature::~MoveStateCreature() = default;</code>	
7		
8	<code>MoveStateCreature::MoveStateCreature() : StateCreature(){</code>	
9	<code>stateId = StateID::Move;</code>	
10	<code>direction = Direction::down;</code>	
11	<code>finalized = false;</code>	
12	<code>}</code>	
13		
14	<code>void MoveStateCreature::performTask(uint id, std::unordered_map<uint, std::share</code>	
15	<code>d_ptr<GameObject>> &gameObjects, Board &board) {</code>	
16	<code>std::shared_ptr<Creature> aCreature = std::dynamic_pointer_cast<Creature>(ga</code>	
17	<code>meObjects.at(id));</code>	
18	<code>if (!movement.hasStart()) {</code>	
19	<code>std::shared_ptr<Cell> newCell;</code>	
20	<code>aCreature->setDirection(direction);</code>	
21	<code>if (board.creatureCanMove(aCreature->getActualCell(), direction)) {</code>	
22	<code>newCell = board.getNextCell(aCreature->getActualCell(), direction);</code>	
23	<code>movement.start(board.getPointFromCell(aCreature->getActualCell(), d</code>	
24	<code>irection, aCreature->getCreatureId());</code>	
25	<code>aCreature->getActualCell()->free();</code>	
26	<code>newCell->occupied(id);</code>	
27	<code>aCreature->setCell(newCell);</code>	
28	<code>} else {</code>	
29	<code>finalized = true;</code>	
30	<code>}</code>	
31	<code>} else {</code>	
32	<code>aCreature->setPoint(movement.doStep());</code>	
33	<code>if (movement.isOver()) {</code>	
34	<code>finalized = true;</code>	
35	<code>}</code>	
36	<code>}</code>	
37	<code>bool MoveStateCreature::isOnPursuit(uint pursuitId) {</code>	
38	<code>return false;</code>	
39	<code>}</code>	
40	<code>bool MoveStateCreature::isAttacking() {</code>	
41	<code>return false;</code>	
42	<code>}</code>	
43		
44	<code>StateID MoveStateCreature::getNextStateID(InputInfo info) {</code>	
45	<code>return StateID::Move;</code>	
46	<code>}</code>	
47		
48	<code>StateID MoveStateCreature::getResetStateID() {</code>	
49	<code>return StateID::Move;</code>	
50	<code>}</code>	
51		
52	<code>void MoveStateCreature::init(InputInfo aInputInfo) {</code>	
53	<code>movement.reset();</code>	
54	<code>inputInfo = aInputInfo;</code>	
55	<code>switch(aInputInfo.input) {</code>	
56	<code>case InputID::up:</code>	
57	<code>direction = Direction::up;</code>	
58	<code>break;</code>	
59	<code>case InputID::down:</code>	
60	<code>direction = Direction::down;</code>	
61	<code>break;</code>	
62	<code>case InputID::left:</code>	
63	<code>direction = Direction::left;</code>	

jul 21, 20 15:20	MoveStateCreature.cpp	Page 2/2
64	break;	
65	case InputID::right:	
66	direction = Direction::right;	
67	break;	
68	default:	
69	direction = Direction::up;	
70	break;	
71	}	
72	finalized = false;	
73	}	

jul 21, 20 15:20	MoveStateCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_MOVESTATECHARACTER_H	
2	#define ARGENTUM_TALLER_MOVESTATECHARACTER_H	
3		
4		
5	#include <MovementCharacter.h>	
6	#include "State.h"	
7	#include "StateCharacter.h"	
8		
9	class MoveStateCharacter : public StateCharacter {	
10	private:	
11	Direction direction;	
12	MovementCharacter movement;	
13		
14	public:	
15	MoveStateCharacter();	
16	~MoveStateCharacter() override;	
17		
18		
19	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec	
20	t>> &gameObjects, Board &board) override;	
21	bool isAttacking() override;	
22		
23	bool isMeditating() override;	
24		
25	void init(InputInfo aInputInfo) override;	
26		
27	StateID getNextStateID(InputInfo info) override;	
28		
29	StateID getResetStateID() override;	
30	};	
31		
32		
33	#endif //ARGENTUM_TALLER_MOVESTATECHARACTER_H	

jul 21, 20 15:20	MoveStateCharacter.cpp	Page 1/2
	<pre> 1 #include <iostream> 2 #include "MoveStateCharacter.h" 3 #include "../GameCharacter.h" 4 #include "StillStateCharacter.h" 5 #include "../Creature.h" 6 7 MoveStateCharacter::~MoveStateCharacter() = default; 8 9 MoveStateCharacter::MoveStateCharacter() : StateCharacter() { 10 stateId = StateID::Move; 11 direction = Direction::down; 12 } 13 14 void MoveStateCharacter::performTask(uint id, 15 std::unordered_map<uint, std::shared_ptr<Ga 16 meObject>> &gameObjects, 17 Board &board) { 18 std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<GameCh 19 aracter>(gameObjects.at(id)); 20 if (!movement.hasStart()) { 21 aCharacter->setDirection(direction); 22 std::shared_ptr<Cell> newCell; 23 if (board.characterCanMove(aCharacter->getActualCell(), direction)) { 24 newCell = board.getNextCell(aCharacter->getActualCell(), direction); 25 movement.start(board.getPointFromCell(aCharacter->getActualCell()), 26 direction, aCharacter->getRace()); 27 aCharacter->getActualCell()->free(); 28 newCell->occupied(id); 29 aCharacter->setCell(newCell); 30 uint nestId = newCell->getNestId(); 31 if (nestId != 0 & !aCharacter->isDead()) { 32 std::shared_ptr<Creature> aCreature; 33 for (auto &creatureId : board.getCreaturesInNest(nestId)) { 34 aCreature = std::dynamic_pointer_cast<Creature>(gameObjects. 35 at(creatureId)); 36 aCreature->notify(id); 37 } 38 } else { 39 finalized = true; 40 } 41 } else { 42 aCharacter->setPoint(movement.doStep()); 43 if (movement.isOver()) { 44 finalized = true; 45 } 46 } 47 } 48 StateID MoveStateCharacter::getNextStateID(InputInfo info) { 49 StateID nextStateId = StateID::Still; 50 if (info.input == InputID::up info.input == InputID::down 51 info.input == InputID::left info.input == InputID::right) { 52 nextStateId = StateID::Move; 53 } else if (info.input == InputID::stopMove) { 54 nextStateId = StateID::Still; 55 } 56 return nextStateId; 57 } 58 StateID MoveStateCharacter::getResetStateID() { 59 return StateID::Move; 60 } 61 } 62 </pre>	

jul 21, 20 15:20	MoveStateCharacter.cpp	Page 2/2
	<pre> 63 64 bool MoveStateCharacter::isAttacking() { 65 return false; 66 } 67 68 bool MoveStateCharacter::isMeditating() { 69 return false; 70 } 71 72 void MoveStateCharacter::init(InputInfo aInputInfo) { 73 movement.reset(); 74 switch(aInputInfo.input) { 75 case InputID::up: 76 direction = Direction::up; 77 break; 78 case InputID::down: 79 direction = Direction::down; 80 break; 81 case InputID::left: 82 direction = Direction::left; 83 break; 84 case InputID::right: 85 direction = Direction::right; 86 break; 87 default: 88 direction = Direction::down; 89 break; 90 } 91 inputInfo = aInputInfo; 92 finalized = false; 93 } 94 </pre>	

jul 21, 20 15:20	MeditateStateCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_MEDITATESTATECHARACTER_H	
2	#define ARGENTUM_TALLER_MEDITATESTATECHARACTER_H	
3		
4		
5	#include "StateCharacter.h"	
6		
7	class MeditateStateCharacter : public StateCharacter {	
8	public:	
9	MeditateStateCharacter();	
10		
11	~MeditateStateCharacter() override;	
12		
13	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec	
14	t>> &gameObjects, Board &board) override;	
15	StateID getNextStateID(InputInfo info) override;	
16	StateID getResetStateID() override;	
17		
18	bool isAttacking() override;	
19		
20	bool isMeditating() override;	
21		
22	void init(InputInfo aInputInfo) override;	
23		
24	};	
25		
26	#endif //ARGENTUM_TALLER_MEDITATESTATECHARACTER_H	
27		
28		

jul 21, 20 15:20	MeditateStateCharacter.cpp	Page 1/1
1	#include "MeditateStateCharacter.h"	
2		
3	MeditateStateCharacter::~MeditateStateCharacter() = default;	
4		
5	MeditateStateCharacter::MeditateStateCharacter() {	
6	finalized = true;	
7	stateId = StateID::Meditate;	
8	}	
9		
10	void MeditateStateCharacter::performTask(uint id, std::unordered_map<uint, std::	
11	shared_ptr<GameObject>> &gameObjects, Board &board) {	
12		
13		
14	bool MeditateStateCharacter::isAttacking() {	
15	return false;	
16	}	
17		
18	bool MeditateStateCharacter::isMeditating() {	
19	return true;	
20	}	
21		
22	StateID MeditateStateCharacter::getNextStateID(InputInfo info) {	
23	StateID nextStateId = StateID::Still;	
24	if (info.input == InputID::up ∨ info.input == InputID::down ∨	
25	info.input == InputID::left ∨ info.input == InputID::right) {	
26	nextStateId = StateID::Move;	
27	}	
28	return nextStateId;	
29	}	
30		
31	StateID MeditateStateCharacter::getResetStateID() {	
32	return StateID::Meditate;	
33	}	
34		
35	void MeditateStateCharacter::init(InputInfo aInputInfo) {	
36	inputInfo = aInputInfo;	
37	}	

jul 21, 20 15:20	InteractStateCharacter.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_INTERACTSTATECHARACTER_H</code>	
2	<code>#define ARGENTUM_TALLER_INTERACTSTATECHARACTER_H</code>	
3		
4	<code>#include "StateCharacter.h"</code>	
5		
6	<code>class InteractStateCharacter: public StateCharacter {</code>	
7	<code>private:</code>	
8	<code>bool interacting = false;</code>	
9	<code>std::shared_ptr<GameObject> aNpc = nullptr;</code>	
10	<code>public:</code>	
11	<code>InteractStateCharacter();</code>	
12		
13	<code>~InteractStateCharacter() override;</code>	
14		
15	<code>void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObjec</code>	
16	<code>t>> &gameObjects, Board &board) override;</code>	
17	<code>void init(InputInfo aInputInfo) override;</code>	
18		
19	<code>StateID getNextStateID(InputInfo info) override;</code>	
20		
21	<code>StateID getResetStateID() override;</code>	
22		
23	<code>bool isMeditating() override;</code>	
24		
25	<code>bool isAttacking() override;</code>	
26		
27	<code>};</code>	
28	<code>#endif</code>	
29		

jul 21, 20 15:20	InteractStateCharacter.cpp	Page 1/2
1	<code>#include "InteractStateCharacter.h"</code>	
2	<code>#include "../GameCharacter.h"</code>	
3		
4	<code>InteractStateCharacter::InteractStateCharacter() : StateCharacter() {</code>	
5	<code>stateId = StateID::Interact;</code>	
6	<code>}</code>	
7		
8	<code>InteractStateCharacter::~InteractStateCharacter() = default;</code>	
9		
10	<code>void InteractStateCharacter::performTask(uint id, std::unordered_map<uint, std::</code>	
11	<code>shared_ptr<GameObject>> &gameObjects, Board &board) {</code>	
12	<code>std::shared_ptr<GameObject> aCharacter = std::dynamic_pointer_cast<GameCh</code>	
13	<code>aracter>(gameObjects.at(id));</code>	
14	<code>std::shared_ptr<Cell> npcCell = board.getCellFromPoint(inputInfo.position);</code>	
15	<code>if (!interacting) {</code>	
16	<code>if (npcCell->getGameObjectId() != 0 ^ npcCell != aCharacter->getActualCell</code>	
17	<code>(id) {</code>	
18	<code>aNpc = gameObjects.at(npcCell->getGameObjectId());</code>	
19	<code>if (board.getDistance(npcCell, aCharacter->getActualCell()) == 1) {</code>	
20	<code>NPCInfo info = aNpc->interact(*aCharacter, inputInfo);</code>	
21	<code>if (info.type != 0) {</code>	
22	<code>aCharacter->setInteractInfo(info);</code>	
23	<code>interacting = true;</code>	
24	<code>finalized = true;</code>	
25	<code>}</code>	
26	<code>} else {</code>	
27	<code>finalized = true;</code>	
28	<code>}</code>	
29	<code>} else {</code>	
30	<code>if (aCharacter->hasAnInputInfo()) {</code>	
31	<code>InputInfo info = aCharacter->getNextInputInfo();</code>	
32	<code>if (aCharacter->isDead() ^ !(info.input == InputID::resurrect)) {</code>	
33	<code>finalized = true;</code>	
34	<code>return;</code>	
35	<code>if (info.input == InputID::buy ^ info.input == InputID::cure ^</code>	
36	<code>info.input == InputID::sell ^ info.input == InputID::resurrect ^</code>	
37	<code>info.input == InputID::retireItem ^ info.input == InputID::retireG</code>	
38	<code>old ^</code>	
39	<code>info.input == InputID::depositItem ^ info.input == InputID::deposi</code>	
40	<code>tGold) {</code>	
41	<code>try {</code>	
42	<code>aCharacter->setInteractInfo(aNpc->interact(*aCharacter, info</code>	
43	<code>));</code>	
44	<code>catch (Exception &e) {</code>	
45	<code>finalized = true;</code>	
46	<code>} else {</code>	
47	<code>finalized = true;</code>	
48	<code>}</code>	
49	<code>}</code>	
50	<code>}</code>	
51		
52	<code>StateID InteractStateCharacter::getNextStateID(InputInfo info) {</code>	
53	<code>StateID nextStateId = StateID::Still;</code>	
54	<code>if (info.input == InputID::up ^ info.input == InputID::down ^</code>	
55	<code>info.input == InputID::left ^ info.input == InputID::right) {</code>	
56	<code>nextStateId = StateID::Move;</code>	
57	<code>return nextStateId;</code>	
58	<code>}</code>	
59		
60		

jul 21, 20 15:20	InteractStateCharacter.cpp	Page 2/2
61	StateID InteractStateCharacter::getResetStateID() {	
62	return StateID::Still;	
63	}	
64		
65	bool InteractStateCharacter::isAttacking() {	
66	return false;	
67	}	
68		
69	bool InteractStateCharacter::isMeditating() {	
70	return false;	
71	}	
72		
73	void InteractStateCharacter::init(InputInfo aInputInfo) {	
74	interacting = false;	
75	aNpc = nullptr;	
76	inputInfo = aInputInfo;	
77	finalized = false;	
78	}	
79		
80		

jul 21, 20 15:20	EquipStateCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_EQUIPSTATECHARACTER_H	
2	#define ARGENTUM_TALLER_EQUIPSTATECHARACTER_H	
3		
4		
5	#include "StateCharacter.h"	
6		
7	class EquipStateCharacter : public StateCharacter {	
8	private:	
9	int itemToChange{0};	
10	public:	
11	explicit EquipStateCharacter();	
12		
13	~EquipStateCharacter() override;	
14		
15	void	
16	performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject>> &	
17	gameObjects, Board &board) override;	
18	StateID getNextStateID(InputInfo info) override;	
19	StateID getResetStateID() override;	
20		
21	bool isMeditating() override;	
22		
23	void init(InputInfo aInputInfo) override;	
24		
25	bool isAttacking() override;	
26		
27	};	
28		
29		
30	#endif //ARGENTUM_TALLER_EQUIPSTATECHARACTER_H	

jul 21, 20 15:20	EquipStateCharacter.cpp	Page 1/1
1	#include "EquipStateCharacter.h"	
2	#include "../GameCharacter.h"	
3		
4	EquipStateCharacter::~EquipStateCharacter() = default ;	
5		
6	EquipStateCharacter::EquipStateCharacter() : StateCharacter() {	
7	finalized = true ;	
8	stateId = StateID::Equip;	
9	}	
10		
11	void EquipStateCharacter::performTask(uint id,	
12	std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) {	
13		
14	std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<GameCharacter>(gameObjects.at(id));	
15	if (inputInfo.input == InputID::unequipItem) {	
16	aCharacter->unequipItem(inputInfo.additional);	
17	} else {	
18	if (inputInfo.additional != 0) {	
19	aCharacter->equipItem(inputInfo.additional);	
20	}	
21	}	
22	}	
23		
24	bool EquipStateCharacter::isAttacking() {	
25	return false ;	
26	}	
27		
28	bool EquipStateCharacter::isMeditating() {	
29	return false ;	
30	}	
31		
32	StateID EquipStateCharacter::getNextStateID(InputInfo info) {	
33	StateID nextStateId = StateID::Still;	
34	if (info.input == InputID::up ∨ info.input == InputID::down ∨	
35	info.input == InputID::left ∨ info.input == InputID::right) {	
36	nextStateId = StateID::Move;	
37	}	
38	return nextStateId;	
39	}	
40		
41	StateID EquipStateCharacter::getResetStateID() {	
42	return StateID::Still;	
43	}	
44		
45	void EquipStateCharacter::init(InputInfo aInputInfo) {	
46	inputInfo = aInputInfo;	
47	}	

jul 21, 20 15:20	AttackStateCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_ATTACKSTATECREATURE_H	
2	#define ARGENTUM_TALLER_ATTACKSTATECREATURE_H	
3		
4		
5	#include "StateCreature.h"	
6		
7	class AttackStateCreature : public StateCreature {	
8	private:	
9	uint enemyId = 0;	
10	bool enemyIsDead;	
11	uint8_t timeBetweenAttacks = 0;	
12	std::shared_ptr<GameObject> aEnemy = nullptr;	
13	public:	
14	AttackStateCreature();	
15		
16	~AttackStateCreature() override;	
17		
18	void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) override;	
19		
20	StateID getNextStateID(InputInfo info) override;	
21		
22	StateID getResetStateID() override;	
23		
24	bool isOnPursuit(uint pursuitId) override;	
25		
26	void init(InputInfo aInputInfo) override;	
27		
28	bool isAttacking() override;	
29	};	
30		
31		
32	#endif //ARGENTUM_TALLER_ATTACKSTATECREATURE_H	

jul 21, 20 15:20	AttackStateCreature.cpp	Page 1/2
1	#include "AttackStateCreature.h"	
2	#include "../Creature.h"	
3		
4	AttackStateCreature::~AttackStateCreature() = default;	
5		
6	AttackStateCreature::AttackStateCreature() : StateCreature(), enemyIsDead(false)	
7	{	
8	stateId = StateID::Attack;	
9	finalized = false;	
10	}	
11	void AttackStateCreature::performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) {	
12		
13	if (timeBetweenAttacks == 0) {	
14	timeBetweenAttacks = 30;	
15	std::shared_ptr<Creature> aCreature = std::dynamic_pointer_cast<Creature	
16	>(gameObjects.at(id));	
17	try {	
18	aEnemy = gameObjects.at(enemyId);	
19	if (aEnemy->isDead()) {	
20	enemyIsDead = true;	
21	finalized = true;	
22	return;	
23	} catch (const std::exception& e) {	
24	enemyIsDead = true;	
25	finalized = true;	
26	return;	
27	}	
28	std::shared_ptr<Cell> creatureCell = aCreature->getActualCell();	
29	std::shared_ptr<Cell> enemyCell = aEnemy->getActualCell();	
30		
31	if (board.getDistance(creatureCell, enemyCell) == GameStatsConfig::getWeaponDistance(WeaponID::Nothing)) {	
32	aEnemy->receiveDamage(GameStatsConfig::getDamage(aCreature->getCreatureId()),	
33	WeaponID::Nothing);	
34	} else {	
35	if (aEnemy != nullptr) {	
36	aEnemy->setInteractWeapon(WeaponID::Nothing);	
37		
38	finalized = true;	
39	}	
40	} else {	
41	timeBetweenAttacks--;	
42	}	
43	}	
44		
45	bool AttackStateCreature::isOnPursuit(uint pursuitId) {	
46	return false;	
47	}	
48		
49	bool AttackStateCreature::isAttacking() {	
50	return true;	
51	}	
52		
53	StateID AttackStateCreature::getNextStateID(InputInfo info) {	
54	StateID nextStateId = StateID::Still;	
55	if (!enemyIsDead) {	
56	nextStateId = StateID::Pursuit;	
57	}	
58	return nextStateId;	
59	}	
60		
61	StateID AttackStateCreature::getResetStateID() {	

jul 21, 20 15:20	AttackStateCreature.cpp	Page 2/2
62	return StateID::Still;	
63	}	
64		
65	void AttackStateCreature::init(InputInfo aInputInfo) {	
66	inputInfo = aInputInfo;	
67	enemyId = aInputInfo.additional;	
68	enemyIsDead = false;	
69	timeBetweenAttacks = 0;	
70	aEnemy = nullptr;	
71	finalized = false;	
72	}	

jul 21, 20 15:20	AttackStateCharacter.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_ATTACKSTATECHARACTER_H</code>	
2	<code>#define ARGENTUM_TALLER_ATTACKSTATECHARACTER_H</code>	
3		
4		
5	<code>#include "State.h"</code>	
6	<code>#include "StateCharacter.h"</code>	
7	<code>class AttackStateCharacter : public StateCharacter {</code>	
8	<code>private:</code>	
9	<code>uint8_t timeBetweenAttacks = 0;</code>	
10	<code>bool enemyReceiveDamage = false;</code>	
11	<code>std::shared_ptr<GameObject> aEnemy = nullptr;</code>	
12	<code>public:</code>	
13	<code>AttackStateCharacter();</code>	
14	<code>~AttackStateCharacter() override;</code>	
15		
16	<code>void performTask(uint id, std::unordered_map<uint, std::shared_ptr<GameObject></code>	
17	<code>t>> &gameObjects, Board &board) override;</code>	
18	<code>StateID getNextStateID(InputInfo info) override;</code>	
19	<code>StateID getResetStateID() override;</code>	
20		
21	<code>void init(InputInfo aInputInfo) override;</code>	
22		
23	<code>bool isAttacking() override;</code>	
24	<code>bool isMeditating() override;</code>	
25		
26	<code>};</code>	
27		
28		
29		
30		
31		
32	<code>#endif //ARGENTUM_TALLER_ATTACKSTATECHARACTER_H</code>	

jul 21, 20 15:20	AttackStateCharacter.cpp	Page 1/2
1	<code>#include "AttackStateCharacter.h"</code>	
2	<code>#include "../GameCharacter.h"</code>	
3		
4	<code>AttackStateCharacter::~AttackStateCharacter() = default;</code>	
5		
6	<code>AttackStateCharacter::AttackStateCharacter() : StateCharacter() {</code>	
7	<code>stateId = StateID::Attack;</code>	
8	<code>}</code>	
9		
10	<code>void AttackStateCharacter::performTask(uint id, std::unordered_map<uint, std::sh</code>	
11	<code>ared_ptr<GameObject>> &gameObjects,</code>	
12	<code>Board &board) {</code>	
13	<code>std::shared_ptr<GameCharacter> aCharacter = std::dynamic_pointer_cast<GameCh</code>	
14	<code>aracter>(gameObjects.at(id));</code>	
15	<code>if (timeBetweenAttacks == 0) {</code>	
16	<code>timeBetweenAttacks = 20;</code>	
17	<code>std::shared_ptr<Cell> enemyCell = board.getCellFromPoint(inputInfo.posit</code>	
18	<code>ion);</code>	
19	<code>if (enemyCell == aCharacter->getActualCell() ^ aCharacter->getWeapon() ==</code>	
20	<code>WeaponID::ElficFlaute) {</code>	
21	<code>aCharacter->restoreHealth();</code>	
22	<code>aEnemy = aCharacter;</code>	
23	<code>return;</code>	
24	<code>if (enemyCell->getGameObjectId() != 0 ^ enemyCell != aCharacter->getActual</code>	
25	<code>Cell()) {</code>	
26	<code>try {</code>	
27	<code>aEnemy = gameObjects.at(enemyCell->getGameObjectId());</code>	
28	<code>if (!aEnemy->isDead() ^</code>	
29	<code>board.getDistance(aCharacter->getActualCell(), enemyCell) <=</code>	
30	<code>GameStatsConfig::getWeaponDistance(aCharacter->getWeapon()) ^</code>	
31	<code>aCharacter->canUseWeapon() ^ aEnemy->canBeAttacked(aChar</code>	
32	<code>acter->getLevel())) {</code>	
33	<code>float damage = GameStatsConfig::getDamage(aCharacter->getRac</code>	
34	<code>e(), aCharacter->getWeapon());</code>	
35	<code>aCharacter->consumeMana();</code>	
36	<code>aEnemy->receiveDamage(damage, aCharacter->getWeapon());</code>	
37	<code>aCharacter->gainExp(aEnemy->isDead() ?</code>	
38	<code>GameStatsConfig::getAdditionalExp(damage, aEnemy->getMax</code>	
39	<code>Life(), aCharacter->getLevel(), aEnemy->getLevel()) :</code>	
40	<code>GameStatsConfig::getExp(damage, aCharacter->getLevel(),</code>	
41	<code>aEnemy->getLevel());</code>	
42	<code>if (aCharacter->getExp() >= GameStatsConfig::getNextLevelLimi</code>	
43	<code>t(aCharacter->getLevel())) {</code>	
44	<code>aCharacter->upLevel();</code>	
45	<code>enemyReceiveDamage = true;</code>	
46	<code>} catch (const std::exception& e) {</code>	
47	<code>std::cout << "Cannot get Enemy" << std::endl;</code>	
48	<code>}</code>	
49	<code>if (!enemyReceiveDamage) {</code>	
50	<code>finalized = true;</code>	
51	<code>} else {</code>	
52	<code>timeBetweenAttacks--;</code>	
53	<code>if (timeBetweenAttacks == 0) {</code>	
54	<code>finalized = true;</code>	
55	<code>if (aEnemy != nullptr) {</code>	
56	<code>aEnemy->setInteractWeapon(WeaponID::Nothing);</code>	
57	<code>}</code>	
58	<code>}</code>	
59	<code>}</code>	

jul 21, 20 15:20	AttackStateCharacter.cpp	Page 2/2
56	}	
57		
58	bool AttackStateCharacter::isAttacking() {	
59	return true;	
60	}	
61		
62	bool AttackStateCharacter::isMeditating() {	
63	return false;	
64	}	
65		
66	StateID AttackStateCharacter::getNextStateID(InputInfo info) {	
67	StateID nextStateId = StateID::Still;	
68	if (info.input == InputID::up ∨ info.input == InputID::down ∨	
69	info.input == InputID::left ∨ info.input == InputID::right) {	
70	nextStateId = StateID::Move;	
71	}	
72	return nextStateId;	
73	}	
74		
75	StateID AttackStateCharacter::getResetStateID() {	
76	return StateID::Still;	
77	}	
78		
79	void AttackStateCharacter::init(InputInfo aInputInfo) {	
80	inputInfo = aInputInfo;	
81	finalized = false;	
82	aEnemy = nullptr;	
83	timeBetweenAttacks = 0;	
84	enemyReceiveDamage = false;	
85	}	

jul 21, 20 15:20	server_main.cpp	Page 1/1
1	#include <iostream>	
2	#include "World.h"	
3	#include "../common/JsonReader.h"	
4	#include "PlayerAcceptor.h"	
5	#include <string>	
6		
7	#define EXIT "q"	
8	#define INVALIDARGUMENTS "Error argumentos inválidos: arguentum_server <path configuracion>"	
9		
10	int main(int argc, char* argv[]) {	
11	if (argc != 2) {	
12	std::cout << INVALIDARGUMENTS << std::endl;	
13	return 1;	
14	}	
15	rapidjson::Document jsonConfigStats = JsonReader::read(argv[1]);	
16	GameStatsConfig gameStatsConfig(jsonConfigStats);	
17		
18	World world(gameStatsConfig);	
19	PlayerAcceptor acceptor(gameStatsConfig.getPort(), world);	
20		
21	acceptor.start();	
22	world.start();	
23		
24	std::string input;	
25	while (input != EXIT) {	
26	std::cin >> input;	
27	}	
28		
29	world.stop();	
30	world.join();	
31	std::cout << "Server Cerrado" << std::endl;	
32	acceptor.stop();	
33	acceptor.join();	
34	return 0;	
35	}	
36		

jul 21, 20 15:20	ServerItem.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_SERVERITEM_H	
2	#define ARGENTUM_TALLER_SERVERITEM_H	
3		
4		
5	#include "../common/Identificators.h"	
6		
7	class ServerItem {	
8	private:	
9	ItemsInventoryID id;	
10	int16_t amount;	
11	public:	
12	ServerItem(ItemsInventoryID id, int16_t amount);	
13	ItemsInventoryID getId() const ;	
14		
15		
16	friend bool operator==(ServerItem &thisItem, const ServerItem& otherItem);	
17	friend bool operator!=(ServerItem &thisItem, const ServerItem& otherItem);	
18		
19	void increaseAmount();	
20		
21	void decreaseAmount();	
22		
23		
24	virtual ~ServerItem();	
25	};	
26		
27		
28	#endif //ARGENTUM_TALLER_SERVERITEM_H	
29		

jul 21, 20 15:20	ServerItem.cpp	Page 1/1
1	#include "ServerItem.h"	
2		
3	ServerItem::~ServerItem() = default ;	
4		
5	ServerItem::ServerItem(ItemsInventoryID id, int16_t amount) : id(id), amount(amo unt) {}	
6		
7	ItemsInventoryID ServerItem::getId() const {	
8	return id;	
9	}	
10		
11	void ServerItem::increaseAmount() {	
12	amount++;	
13	}	
14		
15	void ServerItem::decreaseAmount() {	
16	amount--;	
17	if (amount == 0) {	
18	id = ItemsInventoryID::Nothing;	
19	amount++;	
20	}	
21	}	
22		
23	bool operator==(ServerItem &thisItem, const ServerItem &otherItem) {	
24	return thisItem.getId() == otherItem.getId();	
25	}	
26		
27	bool operator!=(ServerItem &thisItem, const ServerItem &otherItem) {	
28	return !(thisItem == otherItem);	
29	}	

jul 21, 20 15:20	Profession.h	Page 1/1
1	<code>#ifndef PROFESSION_H</code>	
2	<code>#define PROFESSION_H</code>	
3		
4	<code>#include <vector></code>	
5	<code>#include "../common/Identificators.h"</code>	
6	<code>#include "GameObject.h"</code>	
7	<code>#include "GameCharacter.h"</code>	
8		
9	<code>class Profession {</code>	
10	<code>protected:</code>	
11	<code>std::vector<ActionsProfessionID> actions;</code>	
12	<code>public:</code>	
13	<code>Profession() : actions() {}</code>	
14		
15	<code>virtual ~Profession(){};</code>	
16		
17	<code>std::vector<ActionsProfessionID> getActions() const {</code>	
18	<code>return actions;</code>	
19	<code>}</code>	
20		
21	<code>virtual void processInput(GameCharacter& character, InputInfo inputInfo) = 0</code>	
22	<code>;</code>	
23	<code>virtual NPCInfo getInfo(uint id) = 0;</code>	
24	<code>};</code>	
25		
26	<code>#endif</code>	

jul 21, 20 15:20	Priest.h	Page 1/1
1	<code>#ifndef PRIEST_H</code>	
2	<code>#define PRIEST_H</code>	
3		
4	<code>#include "Profession.h"</code>	
5	<code>#include <unordered_map></code>	
6	<code>#include <vector></code>	
7	<code>#include "../common/Identificators.h"</code>	
8	<code>#include "states/State.h"</code>	
9		
10	<code>class Priest : public Profession {</code>	
11	<code>private:</code>	
12	<code>Priest();</code>	
13	<code>std::unordered_map<ItemsInventoryID, uint, std::hash<ItemsInventoryID>> items</code>	
14	<code>;</code>	
15		
16	<code>static Priest* priest;</code>	
17	<code>ItemsInventoryID buyItem(ItemsInventoryID idItem, uint* balance) const;</code>	
18		
19	<code>std::unordered_map<ItemsInventoryID, uint> getItems() const;</code>	
20		
21	<code>public:</code>	
22	<code>static Priest* getInstance();</code>	
23		
24	<code>void init(const std::map<ItemsInventoryID, ItemInfo>& itemsToInit);</code>	
25		
26	<code>virtual NPCInfo getInfo(uint id);</code>	
27		
28	<code>void processInput(GameCharacter &character, InputInfo inputInfo) override;</code>	
29		
30	<code>virtual ~Priest();</code>	
31	<code>};</code>	
32		
33	<code>#endif</code>	

jul 21, 20 15:20	Priest.cpp	Page 1/2
1	#include "Priest.h"	
2		
3	Priest* Priest::priest = nullptr;	
4		
5	Priest::Priest() :items() {	
6	this->actions.push_back(ActionsProfessionID::Cure);	
7	this->actions.push_back(ActionsProfessionID::Resurrect);	
8	this->actions.push_back(ActionsProfessionID::Buy);	
9	}	
10		
11		
12	Priest* Priest::getInstance() {	
13	if (priest == nullptr)	
14	priest = new Priest();	
15	return priest;	
16	}	
17		
18	void Priest::init(const std::map<ItemsInventoryID, ItemInfo>&itemsToInit) {	
19	for (auto &iter: itemsToInit){	
20	if(iter.second.type == "Poison")	
21	this->items.insert({iter.first, iter.second.goldCost});	
22	}	
23	}	
24		
25		
26	ItemsInventoryID Priest::buyItem(ItemsInventoryID idItem, uint* balance) const {	
27	auto iter = items.find(idItem);	
28	if (*balance < (*iter).second)	
29	return ItemsInventoryID::Nothing;	
30	*balance -= (*iter).second;	
31	return idItem;	
32	}	
33		
34	std::unordered_map<ItemsInventoryID, uint> Priest::getItems() const {	
35	return items;	
36	}	
37		
38	NPCInfo Priest::getInfo(uint id) {	
39	NPCInfo info;	
40	info.type = 2;	
41	info.actions = actions;	
42	info.gold = 0;	
43	info.items = getItems();	
44	return info;	
45	}	
46		
47	void Priest::processInput(GameCharacter &character, InputInfo inputInfo) {	
48	uint goldAmount = 0;	
49	switch (inputInfo.input) {	
50	case InputID::buy:	
51	goldAmount = character.getGoldAmount();	
52	if (!character.inventoryIsFull()) {	
53	ItemsInventoryID aItem = buyItem(ItemsInventoryID(inputInfo.adit	
54	ional), &goldAmount);	
55	if (aItem != ItemsInventoryID::Nothing) {	
56	character.addItemToInventory(aItem);	
57	character.setGoldAmount(goldAmount);	
58	}	
59	break;	
60	case InputID::resurrect:	
61	if (character.isDead())	
62	character.cure();	
63	break;	
64	case InputID::cure:	
65	character.cure();	

jul 21, 20 15:20	Priest.cpp	Page 2/2
66	break;	
67	default:	
68	break;	
69	}	
70	}	
71		
72	Priest::~~Priest() {}	

jul 21, 20 15:20	PlayerAcceptor.h	Page 1/1
1	<code>#ifndef PLAYERACCEPTOR_H</code>	
2	<code>#define PLAYERACCEPTOR_H</code>	
3		
4	<code>#include "../common/Thread.h"</code>	
5	<code>#include "../common/Socket.h"</code>	
6	<code>#include "ThPlayer.h"</code>	
7	<code>#include "World.h"</code>	
8	<code>#include "ThLobbyPlayer.h"</code>	
9		
10	<code>class PlayerAcceptor: public Thread {</code>	
11	<code>private:</code>	
12	<code> Socket socket;</code>	
13	<code> World& world;</code>	
14	<code> std::vector<ThLobbyPlayer*> players;</code>	
15	<code> std::atomic<bool> keepTalking;</code>	
16		
17	<code> void clear_finished_games();</code>	
18	<code> void stop_players();</code>	
19	<code>public:</code>	
20	<code> explicit PlayerAcceptor(const std::string& port, World& world);</code>	
21		
22	<code> virtual void run();</code>	
23		
24	<code> void stop();</code>	
25		
26	<code> virtual ~PlayerAcceptor();</code>	
27	<code>};</code>	
28		
29	<code>#endif</code>	

jul 21, 20 15:20	PlayerAcceptor.cpp	Page 1/1
1	<code>#include "PlayerAcceptor.h"</code>	
2	<code>#include <sys/socket.h></code>	
3	<code>#include <iostream></code>	
4	<code>#define MAX_WAITING 20</code>	
5		
6	<code>PlayerAcceptor::PlayerAcceptor(const std::string& port, World& world) : socket()</code>	
7	<code>{</code>	
8	<code> world(world), players(), keepTalking(true){</code>	
9	<code> this->socket.bind_and_listen(port.data(), MAX_WAITING);</code>	
10	<code> std::cout << "Se hizo el bind correctamente " << port.data() << std::endl;</code>	
11	<code> }</code>	
12	<code>void PlayerAcceptor::clear_finished_games() {</code>	
13	<code> std::vector<ThLobbyPlayer*>::iterator iter;</code>	
14	<code> iter = this->players.begin();</code>	
15	<code> while (iter != this->players.end()){</code>	
16	<code> if (!(*iter)->is_alive()){</code>	
17	<code> std::cout << "Se cerro el lobby de un jugador" << std::endl;</code>	
18	<code> (*iter)->join();</code>	
19	<code> delete (*iter);</code>	
20	<code> iter = this->players.erase(iter);</code>	
21	<code> } else {</code>	
22	<code> iter++;</code>	
23	<code> }</code>	
24	<code> }</code>	
25	<code>}</code>	
26		
27	<code>void PlayerAcceptor::run() {</code>	
28	<code> while (this->keepTalking) {</code>	
29	<code> try{</code>	
30	<code> Socket socketPlayer = this->socket.accept();</code>	
31	<code> auto player = new ThLobbyPlayer(std::move(socketPlayer), world);</code>	
32	<code> player->start();</code>	
33	<code> this->players.push_back(player);</code>	
34	<code> clear_finished_games();</code>	
35	<code> } catch (...) {</code>	
36	<code> }</code>	
37	<code> }</code>	
38	<code>}</code>	
39		
40		
41	<code>void PlayerAcceptor::stop_players() {</code>	
42	<code> for (auto & player : this->players) {</code>	
43	<code> player->stop();</code>	
44	<code> player->join();</code>	
45	<code> delete player;</code>	
46	<code> }</code>	
47	<code>}</code>	
48		
49	<code>void PlayerAcceptor::stop() {</code>	
50	<code> this->keepTalking = false;</code>	
51	<code> this->socket.shutdown(SHUT_RDWR);</code>	
52	<code> this->socket.close();</code>	
53	<code> stop_players();</code>	
54	<code>}</code>	
55		
56	<code>PlayerAcceptor::~PlayerAcceptor() = default;</code>	

jul 21, 20 15:20	ObjectItem.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_OBJECTITEM_H</code>	
2	<code>#define ARGENTUM_TALLER_OBJECTITEM_H</code>	
3		
4		
5	<code>#include "GameObject.h"</code>	
6		
7	<code>class ObjectItem : public GameObject {</code>	
8	<code>private:</code>	
9	<code> int timeToBeRemove = 45 * 300;</code>	
10	<code> int amount;</code>	
11	<code> ItemsInventoryID itemId;</code>	
12	<code>public:</code>	
13	<code> ObjectItem(uint id, const Point &initialPoint, const std::shared_ptr<Cell> &</code>	
14	<code> initialCell, const DropItem& dropItem);</code>	
15	<code> float getMaxLife() override;</code>	
16	<code> void update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjec</code>	
17	<code>ts, Board &board) override;</code>	
18	<code> CharacterStateID getStateId() override;</code>	
19	<code> bool isDead() override;</code>	
20		
21	<code> PlayerInfo getPlayerInfo() override;</code>	
22		
23	<code> bool hasAnInputInfo() override;</code>	
24		
25	<code> InputInfo getNextInputInfo() override;</code>	
26		
27	<code> bool isItem() override;</code>	
28		
29	<code> void take();</code>	
30		
31	<code> int getAmount() const;</code>	
32		
33	<code> ItemsInventoryID getItemId() const;</code>	
34		
35	<code> bool canDropsItems() override;</code>	
36		
37	<code> std::vector<DropItem> getDrop() override;</code>	
38		
39	<code> void receiveDamage(float damage, WeaponID weaponId) override;</code>	
40		
41	<code> NPCInfo interact(GameObject &character, InputInfo input) override;</code>	
42		
43	<code> bool isReadyToRemove() override;</code>	
44		
45	<code> void remove(Board &board) override;</code>	
46		
47	<code> virtual ~ObjectItem();</code>	
48		
49	<code> bool canBeAttacked(int enemyLevel) const;</code>	
50		
51	<code>};</code>	
52		
53		
54		
55	<code>#endif //ARGENTUM_TALLER_OBJECTITEM_H</code>	

jul 21, 20 15:20	ObjectItem.cpp	Page 1/2
1	<code>#include "ObjectItem.h"</code>	
2		
3	<code>ObjectItem::ObjectItem(uint id, const Point &initialPoint, const std::shared_ptr</code>	
4	<code><Cell> &initialCell, const DropItem& dropItem)</code>	
5	<code>: GameObject(id, initialPoint, initialCell), amount(dropItem.getAmount()), itemI</code>	
6	<code>d(dropItem.getId()) {</code>	
7	<code> std::string stringItemID = std::to_string((int)itemId);</code>	
8	<code> textureHashId = "ht00h00b00s00w00i" + (stringItemID.size() == 2 ? stringItemID</code>	
9	<code>: "0" + stringItemID);</code>	
10	<code>}</code>	
11	<code>float ObjectItem::getMaxLife() {</code>	
12	<code> return 0;</code>	
13	<code>}</code>	
14	<code>void ObjectItem::update(std::unordered_map<uint, std::shared_ptr<GameObject>> &g</code>	
15	<code>ameObjects, Board &board) {</code>	
16	<code> timeToBeRemove--;</code>	
17	<code>}</code>	
18	<code>CharacterStateID ObjectItem::getStateId() {</code>	
19	<code> return CharacterStateID::Still;</code>	
20	<code>}</code>	
21	<code>bool ObjectItem::isDead() {</code>	
22	<code> return false;</code>	
23	<code>}</code>	
24	<code>void ObjectItem::receiveDamage(float damage, WeaponID weaponId) {}</code>	
25		
26	<code>NPCInfo ObjectItem::interact(GameObject &character, InputInfo input) {</code>	
27	<code> return NPCInfo();</code>	
28	<code>}</code>	
29		
30	<code>bool ObjectItem::isReadyToRemove() {</code>	
31	<code> return timeToBeRemove <= 0;</code>	
32	<code>}</code>	
33		
34	<code>void ObjectItem::remove(Board &board) {</code>	
35	<code> getActualCell()->removeItem();</code>	
36	<code>}</code>	
37		
38	<code>std::vector<DropItem> ObjectItem::getDrop() {</code>	
39	<code> return std::vector<DropItem>();</code>	
40	<code>}</code>	
41		
42	<code>bool ObjectItem::isItem() {</code>	
43	<code> return true;</code>	
44	<code>}</code>	
45		
46	<code>bool ObjectItem::canDropsItems() {</code>	
47	<code> return false;</code>	
48	<code>}</code>	
49		
50	<code>ObjectItem::~ObjectItem() = default;</code>	
51		
52	<code>int ObjectItem::getAmount() const {</code>	
53	<code> return amount;</code>	
54	<code>}</code>	
55		
56	<code>ItemsInventoryID ObjectItem::getItemId() const {</code>	
57	<code> return itemId;</code>	
58	<code>}</code>	
59		
60	<code>void ObjectItem::take() {</code>	
61	<code> timeToBeRemove = 0;</code>	
62	<code>}</code>	

jul 21, 20 15:20	ObjectItem.cpp	Page 2/2
63	}	
64		
65	bool ObjectItem::canBeAttacked(int enemyLevel) const {	
66	return false;	
67	}	
68		
69	PlayerInfo ObjectItem::getPlayerInfo() {	
70	return GameObject::getPlayerInfo();	
71	}	
72		
73	bool ObjectItem::hasAnInputInfo() {	
74	return false;	
75	}	
76		
77	InputInfo ObjectItem::getNextInputInfo() {	
78	InputInfo info;	
79	info.input = InputID::nothing;	
80	return info;	
81	}	

jul 21, 20 15:20	NPCServer.h	Page 1/1
1	#ifndef NPC_H	
2	#define NPC_H	
3		
4	#include "GameObject.h"	
5	#include "GameCharacter.h"	
6	#include "Profession.h"	
7	#include "states/State.h"	
8		
9	class NPCServer : public GameObject{	
10	private:	
11	Profession* profession;	
12	public:	
13	NPCServer(uint id, const std::string& type, Point initialPoint, std::shared_ptr<Cell> initialCell);	
14		
15	virtual ~NPCServer();	
16		
17	void update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) override ;	
18		
19	CharacterStateID getStateId() override;	
20		
21	NPCInfo interact(GameObject& character, InputInfo input) override;	
22		
23	void receiveDamage(float damage, WeaponID weaponId) override;	
24		
25	bool isItem() override;	
26		
27	bool canDropItems() override;	
28		
29	PlayerInfo getPlayerInfo() override;	
30		
31	bool hasAnInputInfo() override;	
32		
33	InputInfo getNextInputInfo() override;	
34		
35	float getMaxLife() override;	
36		
37	std::vector<DropItem> getDrop() override;	
38		
39	bool isDead() override;	
40		
41	void remove(Board &board) override;	
42		
43	bool isReadyToRemove() override;	
44		
45	bool canBeAttacked(int enemyLevel) const override ;	
46	};	
47		
48	#endif	

jul 21, 20 15:20	NPCServer.cpp	Page 1/2
1	#include "NPCServer.h"	
2		
3	#include <utility>	
4	#include "Banker.h"	
5	#include "Merchant.h"	
6	#include "Priest.h"	
7	#include "states/StillStateCharacter.h"	
8		
9	NPCServer::~NPCServer() = default;	
10		
11	NPCServer::NPCServer(uint id, const std::string& type, Point initialPoint, std::shared_ptr<Cell> initialCell) :	
12	GameObject(id, initialPoint, std::move(initialCell)) {	
13	if (type == "banker") {	
14	textureHashId = "ht00h00b08s00w00";	
15	this->profession = Banker::getInstance();	
16	} else if (type == "merchant") {	
17	textureHashId = "ht00h00b09s00w00";	
18	this->profession = Merchant::getInstance();	
19	} else {	
20	cell->addPriest();	
21	textureHashId = "ht00h05b10s00w00";	
22	this->profession = Priest::getInstance();	
23	}	
24	}	
25		
26	void NPCServer::update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) {}	
27		
28	NPCInfo NPCServer::interact(GameObject& character, InputInfo inputInfo) {	
29	if (inputInfo.input != InputID::selectTarget) {	
30	this->profession->processInput(dynamic_cast<GameCharacter>(&character),	
31	inputInfo);	
32	NPCInfo info = this->profession->getInfo(character.getId());	
33	return info;	
34	}	
35		
36	CharacterStateID NPCServer::getStateId() {	
37	return CharacterStateID::Still;	
38	}	
39		
40	bool NPCServer::isReadyToRemove() {	
41	return false;	
42	}	
43		
44	bool NPCServer::isDead() {	
45	return false;	
46	}	
47		
48	void NPCServer::remove(Board &board) {	
49	cell->free();	
50	}	
51		
52	float NPCServer::getMaxLife() {	
53	return 0;	
54	}	
55		
56	void NPCServer::receiveDamage(float damage, WeaponID weaponId) {	
57	}	
58		
59	std::vector<DropItem> NPCServer::getDrop() {	
60	return std::vector<DropItem>();	
61	}	
62		
63	bool NPCServer::isItem() {	

jul 21, 20 15:20	NPCServer.cpp	Page 2/2
64	return false;	
65	}	
66		
67	bool NPCServer::canDropItems() {	
68	return false;	
69	}	
70		
71	bool NPCServer::canBeAttacked(int enemyLevel) const {	
72	return false;	
73	}	
74		
75	PlayerInfo NPCServer::getPlayerInfo() {	
76	return GameObject::getPlayerInfo();	
77	}	
78		
79	bool NPCServer::hasAnInputInfo() {	
80	return false;	
81	}	
82		
83	InputInfo NPCServer::getNextInputInfo() {	
84	InputInfo info;	
85	info.input = InputID::nothing;	
86	return info;	
87	}	

jul 21, 20 15:20	Node.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_NODE_H</code>	
2	<code>#define ARGENTUM_TALLER_NODE_H</code>	
3		
4		
5	<code>#include <memory></code>	
6	<code>#include "Cell.h"</code>	
7		
8	<code>class Node {</code>	
9	<code>private:</code>	
10	<code>int parent;</code>	
11	<code>int g;</code>	
12	<code>int h;</code>	
13	<code>int id;</code>	
14	<code>std::shared_ptr<Cell> cell;</code>	
15	<code>public:</code>	
16	<code>Node(int parent, int g, int h, std::shared_ptr<Cell> cell);</code>	
17		
18	<code>const std::shared_ptr<Cell> &getCell() const;</code>	
19		
20	<code>friend bool operator==(const Node& node1, const Node& node2);</code>	
21		
22	<code>friend bool operator!=(const Node& node1, const Node& node2);</code>	
23		
24	<code>int getF();</code>	
25		
26	<code>void setParent(int parent);</code>	
27		
28	<code>int getParent() const;</code>	
29		
30	<code>int getId() const;</code>	
31		
32	<code>int getG() const;</code>	
33		
34	<code>void setG(int g);</code>	
35		
36	<code>int getH() const;</code>	
37		
38	<code>void setH(int h);</code>	
39		
40	<code>virtual ~Node();</code>	
41		
42	<code>};</code>	
43		
44		
45	<code>#endif //ARGENTUM_TALLER_NODE_H</code>	

jul 21, 20 15:20	Node.cpp	Page 1/1
1	<code>#include "Node.h"</code>	
2		
3	<code>#include <utility></code>	
4		
5	<code>Node::Node(int parent, int g, int h, std::shared_ptr<Cell> aCell) :</code>	
6	<code>parent(parent), g(g), h(h), cell(std::move(aCell)) {</code>	
7	<code>id = std::stoi(std::to_string(cell->getX() + 1) + std::to_string(cell->getY(</code>	
8	<code>) + 1));</code>	
9	<code>}</code>	
10	<code>int Node::getG() const {</code>	
11	<code>return g;</code>	
12	<code>}</code>	
13		
14	<code>void Node::setG(int g) {</code>	
15	<code>Node::g = g;</code>	
16	<code>}</code>	
17		
18	<code>int Node::getH() const {</code>	
19	<code>return h;</code>	
20	<code>}</code>	
21		
22	<code>void Node::setH(int h) {</code>	
23	<code>Node::h = h;</code>	
24	<code>}</code>	
25		
26	<code>const std::shared_ptr<Cell> &Node::getCell() const {</code>	
27	<code>return cell;</code>	
28	<code>}</code>	
29		
30	<code>int Node::getF() {</code>	
31	<code>return g + h;</code>	
32	<code>}</code>	
33		
34	<code>int Node::getId() const {</code>	
35	<code>return id;</code>	
36	<code>}</code>	
37		
38	<code>bool operator==(const Node& node1, const Node &node2) {</code>	
39	<code>return node1.getCell() == node2.getCell();</code>	
40	<code>}</code>	
41		
42	<code>bool operator!=(const Node& node1, const Node &node2) {</code>	
43	<code>return ~(node1 == node2);</code>	
44	<code>}</code>	
45		
46	<code>void Node::setParent(int aParent) {</code>	
47	<code>parent = aParent;</code>	
48	<code>}</code>	
49		
50	<code>int Node::getParent() const {</code>	
51	<code>return parent;</code>	
52	<code>}</code>	
53		
54	<code>Node::~Node() = default;</code>	

jul 21, 20 15:20	NodeContainer.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_NODECONTAINER_H</code>	
2	<code>#define ARGENTUM_TALLER_NODECONTAINER_H</code>	
3		
4		
5	<code>#include <map></code>	
6	<code>#include "Node.h"</code>	
7	<code>class NodeContainer {</code>	
8	<code>private:</code>	
9	<code>std::map<int, Node> nodes;</code>	
10	<code>public:</code>	
11	<code>NodeContainer();</code>	
12		
13	<code>void insert(const Node& aNode);</code>	
14		
15	<code>bool has(const Node& aNode);</code>	
16		
17	<code>Node getBestNode();</code>	
18		
19	<code>virtual ~NodeContainer();</code>	
20		
21	<code>void modifyNode(int i, Node &node);</code>	
22		
23	<code>Node &get(int i);</code>	
24	<code>};</code>	
25		
26		
27	<code>#endif // ARGENTUM_TALLER_NODECONTAINER_H</code>	
28		

jul 21, 20 15:20	NodeContainer.cpp	Page 1/1
1	<code>#include "NodeContainer.h"</code>	
2		
3	<code>NodeContainer::NodeContainer() = default;</code>	
4		
5	<code>NodeContainer::~NodeContainer() = default;</code>	
6		
7	<code>void NodeContainer::insert(const Node& aNode) {</code>	
8	<code>nodes.insert(std::pair<int, Node>(aNode.getId(), aNode));</code>	
9	<code>}</code>	
10		
11	<code>bool NodeContainer::has(const Node& aNode) {</code>	
12	<code>bool insideContainer = true;</code>	
13	<code>try {</code>	
14	<code>nodes.at(aNode.getId());</code>	
15	<code>} catch (std::exception &e) {</code>	
16	<code>insideContainer = false;</code>	
17	<code>}</code>	
18	<code>return insideContainer;</code>	
19	<code>}</code>	
20		
21	<code>Node NodeContainer::getBestNode() {</code>	
22	<code>auto iter = nodes.begin();</code>	
23	<code>int bestF = 0;</code>	
24	<code>Node aNode = (*iter).second;</code>	
25	<code>while (iter != nodes.end()) {</code>	
26	<code>if (bestF == 0 (*iter).second.getF() < bestF) {</code>	
27	<code>bestF = (*iter).second.getF();</code>	
28	<code>aNode = (*iter).second;</code>	
29	<code>iter++;</code>	
30	<code>}</code>	
31	<code>nodes.erase(aNode.getId());</code>	
32	<code>return aNode;</code>	
33	<code>}</code>	
34		
35	<code>void NodeContainer::modifyNode(int id, Node &aNode) {</code>	
36	<code>Node &winnerNode = nodes.at(id);</code>	
37	<code>if (aNode.getG() + 1 < winnerNode.getG()) {</code>	
38	<code>winnerNode.setG(aNode.getG() + 1);</code>	
39	<code>winnerNode.setParent(aNode.getId());</code>	
40	<code>}</code>	
41	<code>}</code>	
42		
43	<code>Node &NodeContainer::get(int id) {</code>	
44	<code>return nodes.at(id);</code>	
45	<code>}</code>	
46		

jul 21, 20 15:20	Nest.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_NEST_H</code>	
2	<code>#define ARGENTUM_TALLER_NEST_H</code>	
3		
4		
5	<code>#include <vector></code>	
6	<code>#include <memory></code>	
7	<code>#include "Cell.h"</code>	
8		
9	<code>class Nest {</code>	
10	<code>private:</code>	
11	<code> uint8_t nestLimit;</code>	
12	<code> uint nestId;</code>	
13	<code> std::vector<std::shared_ptr<Cell>> nestCells;</code>	
14	<code> std::vector<uint> creatures;</code>	
15		
16	<code>public:</code>	
17	<code> explicit Nest(uint8_t nestLimit, uint nestId, std::vector<std::shared_ptr<Cell>> cells);</code>	
18		
19	<code> ~Nest();</code>	
20		
21	<code> void addCreature(uint id);</code>	
22		
23	<code> std::shared_ptr<Cell> getFreeCell();</code>	
24		
25	<code> friend bool operator<(const Nest& firstNest, const Nest& secondNest);</code>	
26		
27	<code> uint getNestId() const;</code>	
28		
29	<code> const std::vector<uint> &getCreatures() const;</code>	
30		
31	<code> int getAmountCreatures() const ;</code>	
32		
33	<code> void removeCreature(uint id);</code>	
34		
35	<code> bool isFull() const;</code>	
36	<code>};</code>	
37		
38		
39	<code>#endif //ARGENTUM_TALLER_NEST_H</code>	

jul 21, 20 15:20	Nest.cpp	Page 1/1
1	<code>#include "Nest.h"</code>	
2	<code>#include <utility></code>	
3		
4	<code>Nest::Nest(uint8_t nestLimit, uint nestId, std::vector<std::shared_ptr<Cell>> cells):</code>	
5	<code> nestLimit(nestLimit), nestId(nestId), nestCells(std::move(cells)) {}</code>	
6		
7	<code>bool Nest::isFull() const {</code>	
8	<code> bool isFull = creatures.size() == nestLimit;</code>	
9	<code> if (!isFull) {</code>	
10	<code> isFull = true;</code>	
11	<code> for (auto &aCell : nestCells) {</code>	
12	<code> if (aCell->isEmpty()) {</code>	
13	<code> isFull = false;</code>	
14	<code> break;</code>	
15	<code> }</code>	
16	<code> }</code>	
17	<code> }</code>	
18	<code> return isFull;</code>	
19	<code>}</code>	
20		
21	<code>std::shared_ptr<Cell> Nest::getFreeCell() {</code>	
22	<code> std::shared_ptr<Cell> returnedCell;</code>	
23	<code> for (auto &cell : nestCells) {</code>	
24	<code> if (cell->isEmpty()) {</code>	
25	<code> returnedCell = cell;</code>	
26	<code> }</code>	
27	<code> }</code>	
28	<code> return returnedCell;</code>	
29	<code>}</code>	
30		
31	<code>void Nest::addCreature(uint id) {</code>	
32	<code> creatures.push_back(id);</code>	
33	<code>}</code>	
34		
35	<code>const std::vector<uint> &Nest::getCreatures() const {</code>	
36	<code> return creatures;</code>	
37	<code>}</code>	
38		
39	<code>uint Nest::getNestId() const {</code>	
40	<code> return nestId;</code>	
41	<code>}</code>	
42		
43	<code>void Nest::removeCreature(uint id) {</code>	
44	<code> auto iter = creatures.begin();</code>	
45	<code> while (iter != creatures.end()) {</code>	
46	<code> if (*iter == id) {</code>	
47	<code> iter = this->creatures.erase(iter);</code>	
48	<code> } else {</code>	
49	<code> iter++;</code>	
50	<code> }</code>	
51	<code> }</code>	
52	<code>}</code>	
53		
54	<code>int Nest::getAmountCreatures() const {</code>	
55	<code> return creatures.size();</code>	
56	<code>}</code>	
57		
58	<code>bool operator<(const Nest& firstNest, const Nest &secondNest) {</code>	
59	<code> return firstNest.getAmountCreatures() < secondNest.getAmountCreatures();</code>	
60	<code>}</code>	
61		
62		
63	<code>Nest::~Nest() = default;</code>	

jul 21, 20 15:20	NestContainer.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_NESTCONTAINER_H</code>	
2	<code>#define ARGENTUM_TALLER_NESTCONTAINER_H</code>	
3		
4		
5	<code>#include <vector></code>	
6	<code>#include "Nest.h"</code>	
7	<code>class NestContainer {</code>	
8	<code>private:</code>	
9	<code>std::vector<Nest> nests;</code>	
10	<code>uint8_t nestIndex = 0;</code>	
11	<code>uint8_t length{};</code>	
12		
13	<code>uint8_t getNextIndex();</code>	
14	<code>public:</code>	
15	<code>NestContainer();</code>	
16		
17	<code>explicit NestContainer(std::vector<Nest> nest);</code>	
18		
19	<code>virtual ~NestContainer();</code>	
20		
21	<code>Nest& getNextNestAvailable();</code>	
22		
23	<code>std::vector<Nest> &getNests();</code>	
24		
25	<code>int getAmountCreatures();</code>	
26		
27	<code>Nest &getNest(uint i);</code>	
28	<code>};</code>	
29		
30		
31		
32	<code>#endif //ARGENTUM_TALLER_NESTCONTAINER_H</code>	

jul 21, 20 15:20	NestContainer.cpp	Page 1/1
1	<code>#include "NestContainer.h"</code>	
2	<code>#include "../common/Exception.h"</code>	
3		
4	<code>#include <utility></code>	
5	<code>#include <algorithm></code>	
6		
7	<code>NestContainer::NestContainer(std::vector<Nest> nests) : nests(std::move(nests))</code>	
8	<code>{</code>	
9	<code>length = this->nests.size();</code>	
10	<code>}</code>	
11	<code>Nest& NestContainer::getNextNestAvailable() {</code>	
12	<code>std::sort_heap(nests.begin(), nests.end());</code>	
13	<code>for (auto &aNest : nests) {</code>	
14	<code>if (!aNest.isFull()) {</code>	
15	<code>return aNest;</code>	
16	<code>}</code>	
17	<code>throw Exception("There is not available nests to add creatures");</code>	
18	<code>}</code>	
19	<code>// uint8_t counter = 0;</code>	
20	<code>Nest& nest = nests.at(getNextIndex());</code>	
21	<code>while (nest.isFull() && counter < length) {</code>	
22	<code>nest = nests.at(getNextIndex());</code>	
23	<code>counter++;</code>	
24	<code>}</code>	
25	<code>if (nest.isFull()) {</code>	
26	<code>return nest;</code>	
27	<code>}</code>	
28	<code>return nest;</code>	
29	<code>}</code>	
30		
31	<code>std::vector<Nest> &NestContainer::getNests() {</code>	
32	<code>return nests;</code>	
33	<code>}</code>	
34		
35	<code>Nest &NestContainer::getNest(uint nestId) {</code>	
36	<code>size_t index = 0;</code>	
37	<code>for (auto &aNest : nests) {</code>	
38	<code>if (nestId == aNest.getNestId()) {</code>	
39	<code>break;</code>	
40	<code>}</code>	
41	<code>index++;</code>	
42	<code>}</code>	
43	<code>return nests.at(index);</code>	
44	<code>}</code>	
45		
46	<code>int NestContainer::getAmountCreatures() {</code>	
47	<code>int amountCreatures = 0;</code>	
48	<code>for (auto &aNest : nests) {</code>	
49	<code>amountCreatures += aNest.getAmountCreatures();</code>	
50	<code>}</code>	
51	<code>return amountCreatures;</code>	
52	<code>}</code>	
53		
54	<code>uint8_t NestContainer::getNextIndex() {</code>	
55	<code>nestIndex++;</code>	
56	<code>nestIndex = nestIndex > length ? 0 : nestIndex;</code>	
57	<code>return nestIndex;</code>	
58	<code>}</code>	
59		
60	<code>NestContainer::NestContainer() = default;</code>	
61		
62	<code>NestContainer::~NestContainer() = default;</code>	
63		

jul 21, 20 15:20	Movement.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_MOVEMENT_H	
2	#define ARGENTUM_TALLER_MOVEMENT_H	
3		
4		
5	#include <stdint>	
6	#include "../common/Identificators.h"	
7	#include "GameStatsConfig.h"	
8		
9	class Movement {	
10	protected:	
11	bool finalized;	
12	bool initialized;	
13	Point firstPoint;	
14	Direction direction;	
15	float distance;	
16	float partialDistance;	
17	public:	
18	Movement();	
19		
20	bool isOver() const ;	
21		
22	bool hasStart() const ;	
23		
24	void reset();	
25		
26	void stop();	
27		
28	Point doStep();	
29		
30	virtual float getAmountMovement() = 0;	
31		
32	virtual ~Movement();	
33		
34	};	
35		
36		
37	#endif //ARGENTUM_TALLER_MOVEMENT_H	

jul 21, 20 15:20	MovementCreature.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_MOVEMENTCREATURE_H	
2	#define ARGENTUM_TALLER_MOVEMENTCREATURE_H	
3		
4		
5	#include "Movement.h"	
6		
7	class MovementCreature : public Movement {	
8	private:	
9	CreatureID creatureId = CreatureID::Nothing;	
10	public:	
11	MovementCreature();	
12		
13	void start(Point aFirstPoint, Direction aDirection, CreatureID creatureId);	
14		
15	virtual ~MovementCreature();	
16		
17	float getAmountMovement() override;	
18	};	
19		
20		
21	#endif //ARGENTUM_TALLER_MOVEMENTCREATURE_H	

jul 21, 20 15:20	MovementCreature.cpp	Page 1/1
1	#include "MovementCreature.h"	
2		
3	MovementCreature::MovementCreature() : Movement() {}	
4		
5	MovementCreature::~MovementCreature() = default ;	
6		
7	void MovementCreature::start(Point aFirstPoint, Direction aDirection, CreatureID aCreatureId) {	
8	creatureId = aCreatureId;	
9	partialDistance = 0.0f;	
10	distance = GameStatsConfig::getDistance();	
11	direction = aDirection;	
12	firstPoint = aFirstPoint;	
13	initialized = true ;	
14	}	
15		
16	float MovementCreature::getAmountMovement() {	
17	return GameStatsConfig::getAmountMovement(creatureId);	
18	}	

jul 21, 20 15:20	Movement.cpp	Page 1/1
1	#include "Movement.h"	
2		
3	Movement::Movement() : finalized(false), initialized(false), firstPoint(0.0,0.0)	
4	, direction(Direction::down), distance(0), partialDistance(0) {}	
5		
6	Movement::~Movement() = default ;	
7		
8	bool Movement::isOver() const {	
9	return finalized;	
10	}	
11		
12	bool Movement::hasStart() const {	
13	return initialized;	
14	}	
15		
16	void Movement::reset() {	
17	partialDistance = 0.0f;	
18	initialized = false ;	
19	finalized = false ;	
20	}	
21		
22	Point Movement::doStep() {	
23	Point newPoint = firstPoint;	
24	partialDistance = partialDistance + getAmountMovement() > distance ? distanc	
25	e : partialDistance + getAmountMovement();	
26	switch (direction) {	
27	case Direction::up:	
28	newPoint.y -= partialDistance;	
29	break ;	
30	case Direction::down:	
31	newPoint.y += partialDistance;	
32	break ;	
33	case Direction::left:	
34	newPoint.x -= partialDistance;	
35	break ;	
36	case Direction::right:	
37	newPoint.x += partialDistance;	
38	break ;	
39	}	
40	if (partialDistance ≥ distance) {	
41	finalized = true ;	
42	}	
43	return newPoint;	
44	}	
45		
46	void Movement::stop() {	
47	finalized = true ;	
48	}	

jul 21, 20 15:20	MovementCharacter.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_MOVEMENTCHARACTER_H	
2	#define ARGENTUM_TALLER_MOVEMENTCHARACTER_H	
3		
4		
5	#include "Movement.h"	
6		
7	class MovementCharacter : public Movement {	
8	private:	
9	RaceID raceId = RaceID::Nothing;	
10	public:	
11	MovementCharacter();	
12		
13	void start(Point aFirstPoint, Direction aDirection, RaceID raceId);	
14		
15	float getAmountMovement() override;	
16		
17	~MovementCharacter() override;	
18		
19	};	
20		
21		
22	#endif //ARGENTUM_TALLER_MOVEMENTCHARACTER_H	

jul 21, 20 15:20	MovementCharacter.cpp	Page 1/1
1	#include "MovementCharacter.h"	
2		
3	MovementCharacter::MovementCharacter() : Movement() {}	
4		
5	MovementCharacter::~~MovementCharacter() = default;	
6		
7	void MovementCharacter::start(Point aFirstPoint, Direction aDirection, RaceID aRaceId) {	
8	raceId = aRaceId;	
9	partialDistance = 0.0f;	
10	distance = GameStatsConfig::getDistance();	
11	direction = aDirection;	
12	firstPoint = aFirstPoint;	
13	initialized = true;	
14	}	
15		
16	float MovementCharacter::getAmountMovement() {	
17	return GameStatsConfig::getAmountMovement(raceId);	
18	}	

jul 21, 20 15:20	Merchant.h	Page 1/1
1	<code>#ifndef MERCHANT_H</code>	
2	<code>#define MERCHANT_H</code>	
3		
4	<code>#include <unordered_map></code>	
5	<code>#include "../common/Identificators.h"</code>	
6	<code>#include "GameStats.h"</code>	
7	<code>#include "Profession.h"</code>	
8	<code>#include "GameCharacter.h"</code>	
9		
10	<code>//Se optÃ por realizar esta clase como un Singleton debido a que</code>	
11	<code>//los diversos jugadores(Threads) no podrÃn modificar esta clase</code>	
12	<code>//ya que los items que se venden/compran no se alteran dentro de</code>	
13	<code>//esta clase</code>	
14	<code>class Merchant:public Profession {</code>	
15	<code>private:</code>	
16	<code>Merchant();</code>	
17	<code>std::unordered_map<ItemsInventoryID, uint, std::hash<ItemsInventoryID>> item</code>	
18	<code>s;</code>	
19		
20	<code>static Merchant* merchant;</code>	
21	<code>ItemsInventoryID buyItem(ItemsInventoryID idItem, uint* balance) const;</code>	
22		
23	<code>uint sellItem(ItemsInventoryID idItem) const;</code>	
24		
25	<code>std::unordered_map<ItemsInventoryID,uint> getItems() const;</code>	
26	<code>public:</code>	
27	<code>static Merchant* getInstance();</code>	
28		
29	<code>void init(const std::map<ItemsInventoryID, ItemInfo>& itemsToInit);</code>	
30		
31	<code>NPCInfo getInfo(uint id) override;</code>	
32		
33	<code>void processInput(GameCharacter &character, InputInfo inputInfo) override;</code>	
34		
35	<code>virtual ~Merchant();</code>	
36	<code>};</code>	
37		
38	<code>#endif</code>	

jul 21, 20 15:20	Merchant.cpp	Page 1/2
1	<code>#include "Merchant.h"</code>	
2	<code>#include "GameCharacter.h"</code>	
3		
4	<code>Merchant* Merchant::merchant = nullptr;</code>	
5		
6	<code>Merchant::Merchant() :items() {</code>	
7	<code> this->actions.push_back(ActionsProfessionID::Buy);</code>	
8	<code> this->actions.push_back(ActionsProfessionID::Sell);</code>	
9	<code>}</code>	
10		
11	<code>Merchant* Merchant::getInstance() {</code>	
12	<code> if (merchant == nullptr)</code>	
13	<code> merchant = new Merchant();</code>	
14	<code> return merchant;</code>	
15	<code>}</code>	
16		
17	<code>void Merchant::init(const std::map<ItemsInventoryID, ItemInfo>&itemsToInit) {</code>	
18	<code> for (auto &iter: itemsToInit){</code>	
19	<code> this->items.insert({iter.first, iter.second.goldCost});</code>	
20	<code> }</code>	
21	<code>}</code>	
22		
23	<code>ItemsInventoryID Merchant::buyItem(ItemsInventoryID iditem, uint* balance) const</code>	
24	<code>{</code>	
25	<code> uint itemPrice = items.at(idItem);</code>	
26	<code> if (*balance < itemPrice)</code>	
27	<code> return ItemsInventoryID::Nothing;</code>	
28	<code> *balance -= itemPrice;</code>	
29	<code> return iditem;</code>	
30	<code>}</code>	
31	<code>uint Merchant::sellItem(ItemsInventoryID idItem) const {</code>	
32	<code> if (ItemsInventoryID::Nothing == idItem) {</code>	
33	<code> return 0;</code>	
34	<code> }</code>	
35	<code> auto iter = items.find(idItem);</code>	
36	<code> return (*iter).second;</code>	
37	<code>}</code>	
38		
39	<code>std::unordered_map<ItemsInventoryID,uint> Merchant::getItems() const {</code>	
40	<code> return items;</code>	
41	<code>}</code>	
42		
43	<code>NPCInfo Merchant::getInfo(uint id) {</code>	
44	<code> NPCInfo info;</code>	
45	<code> info.type = 1;</code>	
46	<code> info.actions = actions;</code>	
47	<code> info.gold = 0;</code>	
48	<code> info.items = getItems();</code>	
49	<code> return info;</code>	
50	<code>}</code>	
51		
52	<code>void Merchant::processInput(GameCharacter &character, InputInfo inputInfo) {</code>	
53	<code> uint sell, goldAmount;</code>	
54	<code> switch (inputInfo.input) {</code>	
55	<code> case InputID::sell:</code>	
56	<code> sell = sellItem(ItemsInventoryID(inputInfo.aditional));</code>	
57	<code> character.removeItemFromInventory(ItemsInventoryID(inputInfo.adition</code>	
58	<code>al));</code>	
59	<code> character.gainGold(sell);</code>	
60	<code> break;</code>	
61	<code> case InputID::buy:</code>	
62	<code> goldAmount = character.getGoldAmount();</code>	
63	<code> if (!character.inventoryIsFull()) {</code>	
64	<code> ItemsInventoryID anItem = buyItem(ItemsInventoryID(inputInfo.adi</code>	
65	<code>tional), &goldAmount);</code>	

jul 21, 20 15:20	Merchant.cpp	Page 2/2
64	if (anItem \neq ItemsInventoryID::Nothing) {	
65	character.addToInventory(anItem);	
66	character.setGoldAmount(goldAmount);	
67	}	
68	break ;	
69	default :	
70	break ;	
71	}	
72	}	
73	}	
74		
75		
76	Merchant::~Merchant() {}	

jul 21, 20 15:20	ItemTranslator.h	Page 1/1
1	#ifndef ITEMTRANSLATOR_H	
2	#define ITEMTRANSLATOR_H	
3		
4	#include "../common/Identificators.h"	
5		
6	class ItemTranslator {	
7	public :	
8	ItemTranslator();	
9		
10	static ItemsInventoryID weaponToItem(WeaponID weapon);	
11	static ItemsInventoryID shieldToItem(ShieldID shield);	
12	static ItemsInventoryID bodyToItem(BodyID body);	
13	static ItemsInventoryID helmetToItem(HelmetID helmet);	
14		
15	static WeaponID itemToWeapon(ItemsInventoryID item);	
16	static ShieldID itemToShield(ItemsInventoryID item);	
17	static BodyID itemToBody(ItemsInventoryID item);	
18	static HelmetID itemToHelmet(ItemsInventoryID item);	
19		
20	~ItemTranslator();	
21	};	
22		
23	#endif	

jul 21, 20 15:20	ItemTranslator.cpp	Page 1/4
1	<code>#include "ItemTranslator.h"</code>	
2		
3	<code>ItemTranslator::ItemTranslator() = default;</code>	
4		
5	<code>ItemsInventoryID ItemTranslator::weaponToItem(WeaponID weapon){</code>	
6	<code>ItemsInventoryID idItem = ItemsInventoryID::Nothing;</code>	
7	<code>switch(weapon) {</code>	
8	<code>case WeaponID::SimpleArc:</code>	
9	<code>idItem = ItemsInventoryID::SimpleArc;</code>	
10	<code>break;</code>	
11	<code>case WeaponID::CompoundArc:</code>	
12	<code>idItem = ItemsInventoryID::CompoundArc;</code>	
13	<code>break;</code>	
14	<code>case WeaponID::LongSword:</code>	
15	<code>idItem = ItemsInventoryID::LongSword;</code>	
16	<code>break;</code>	
17	<code>case WeaponID::Hammer:</code>	
18	<code>idItem = ItemsInventoryID::Hammer;</code>	
19	<code>break;</code>	
20	<code>case WeaponID::Ax:</code>	
21	<code>idItem = ItemsInventoryID::Ax;</code>	
22	<code>break;</code>	
23	<code>case WeaponID::ElficFlaute:</code>	
24	<code>idItem = ItemsInventoryID::ElficFlaute;</code>	
25	<code>break;</code>	
26	<code>case WeaponID::AshStick:</code>	
27	<code>idItem = ItemsInventoryID::AshStick;</code>	
28	<code>break;</code>	
29	<code>case WeaponID::Crosier:</code>	
30	<code>idItem = ItemsInventoryID::Crosier;</code>	
31	<code>break;</code>	
32	<code>case WeaponID::GnarledStick:</code>	
33	<code>idItem = ItemsInventoryID::GnarledStick;</code>	
34	<code>break;</code>	
35	<code>case WeaponID::Nothing:</code>	
36	<code>idItem = ItemsInventoryID::Nothing;</code>	
37	<code>break;</code>	
38	<code>} return idItem;</code>	
39	<code>}</code>	
40		
41		
42	<code>ItemsInventoryID ItemTranslator::bodyToItem(BodyID body){</code>	
43	<code>ItemsInventoryID idItem = ItemsInventoryID::Nothing;</code>	
44	<code>switch(body) {</code>	
45	<code>case BodyID::BlueCommon:</code>	
46	<code>idItem = ItemsInventoryID::BlueCommon;</code>	
47	<code>break;</code>	
48	<code>case BodyID::GreenCommon:</code>	
49	<code>idItem = ItemsInventoryID::GreenCommon;</code>	
50	<code>break;</code>	
51	<code>case BodyID::RedCommon:</code>	
52	<code>idItem = ItemsInventoryID::RedCommon;</code>	
53	<code>break;</code>	
54	<code>case BodyID::LeatherArmor:</code>	
55	<code>idItem = ItemsInventoryID::LeatherArmor;</code>	
56	<code>break;</code>	
57	<code>case BodyID::BlueTunic:</code>	
58	<code>idItem = ItemsInventoryID::BlueTunic;</code>	
59	<code>break;</code>	
60	<code>case BodyID::PlateArmor:</code>	
61	<code>idItem = ItemsInventoryID::PlateArmor;</code>	
62	<code>break;</code>	
63	<code>default:</code>	
64	<code>idItem = ItemsInventoryID::Nothing;</code>	
65	<code>break;</code>	
66	<code>}</code>	

jul 21, 20 15:20	ItemTranslator.cpp	Page 2/4
67	<code>return idItem;</code>	
68	<code>}</code>	
69		
70	<code>ItemsInventoryID ItemTranslator::helmetToItem(HelmetID helmet){</code>	
71	<code>ItemsInventoryID idItem = ItemsInventoryID::Nothing;</code>	
72	<code>switch(helmet) {</code>	
73	<code>case HelmetID::Hood:</code>	
74	<code>idItem = ItemsInventoryID::Hood;</code>	
75	<code>break;</code>	
76	<code>case HelmetID::MagicHat:</code>	
77	<code>idItem = ItemsInventoryID::MagicHat;</code>	
78	<code>break;</code>	
79	<code>case HelmetID::IronHelmet:</code>	
80	<code>idItem = ItemsInventoryID::IronHelmet;</code>	
81	<code>break;</code>	
82	<code>case HelmetID::Nothing:</code>	
83	<code>idItem = ItemsInventoryID::Nothing;</code>	
84	<code>break;</code>	
85	<code>default:</code>	
86	<code>idItem = ItemsInventoryID::Nothing;</code>	
87	<code>break;</code>	
88	<code>}</code>	
89	<code>return idItem;</code>	
90	<code>}</code>	
91		
92	<code>ItemsInventoryID ItemTranslator::shieldToItem(ShieldID shield){</code>	
93	<code>ItemsInventoryID idItem = ItemsInventoryID::Nothing;</code>	
94	<code>switch(shield) {</code>	
95	<code>case ShieldID::IronShield:</code>	
96	<code>idItem = ItemsInventoryID::IronShield;</code>	
97	<code>break;</code>	
98	<code>case ShieldID::TurtleShield:</code>	
99	<code>idItem = ItemsInventoryID::TurtleShield;</code>	
100	<code>break;</code>	
101	<code>case ShieldID::Nothing:</code>	
102	<code>idItem = ItemsInventoryID::Nothing;</code>	
103	<code>break;</code>	
104	<code>}</code>	
105	<code>return idItem;</code>	
106	<code>}</code>	
107		
108	<code>WeaponID ItemTranslator::itemToWeapon(ItemsInventoryID item){</code>	
109	<code>WeaponID weapon;</code>	
110	<code>switch(item) {</code>	
111	<code>case ItemsInventoryID::SimpleArc:</code>	
112	<code>weapon = WeaponID::SimpleArc;</code>	
113	<code>break;</code>	
114	<code>case ItemsInventoryID::CompoundArc:</code>	
115	<code>weapon = WeaponID::CompoundArc;</code>	
116	<code>break;</code>	
117	<code>case ItemsInventoryID::LongSword:</code>	
118	<code>weapon = WeaponID::LongSword;</code>	
119	<code>break;</code>	
120	<code>case ItemsInventoryID::Hammer:</code>	
121	<code>weapon = WeaponID::Hammer;</code>	
122	<code>break;</code>	
123	<code>case ItemsInventoryID::Ax:</code>	
124	<code>weapon = WeaponID::Ax;</code>	
125	<code>break;</code>	
126	<code>case ItemsInventoryID::ElficFlaute:</code>	
127	<code>weapon = WeaponID::ElficFlaute;</code>	
128	<code>break;</code>	
129	<code>case ItemsInventoryID::AshStick:</code>	
130	<code>weapon = WeaponID::AshStick;</code>	
131	<code>break;</code>	
132	<code>case ItemsInventoryID::Crosier:</code>	

jul 21, 20 15:20	ItemTranslator.cpp	Page 3/4
133	weapon = WeaponID::Crosier;	
134	break;	
135	case ItemsInventoryID::GnarledStick:	
136	weapon = WeaponID::GnarledStick;	
137	break;	
138	default:	
139	weapon = WeaponID::Nothing;	
140	}	
141	return weapon;	
142	}	
143		
144	ShieldID ItemTranslator::itemToShield(ItemsInventoryID item){	
145	ShieldID shield = ShieldID::Nothing;	
146	switch (item) {	
147	case ItemsInventoryID::IronShield:	
148	shield = ShieldID::IronShield;	
149	break;	
150	case ItemsInventoryID::TurtleShield:	
151	shield = ShieldID::TurtleShield;	
152	break;	
153	default:	
154	shield = ShieldID::Nothing;	
155	}	
156	return shield;	
157	}	
158		
159	BodyID ItemTranslator::itemToBody(ItemsInventoryID item){	
160	BodyID body = BodyID::Nothing;	
161	switch (item) {	
162	case ItemsInventoryID::BlueCommon:	
163	body = BodyID::BlueCommon;	
164	break;	
165	case ItemsInventoryID::GreenCommon:	
166	body = BodyID::GreenCommon;	
167	break;	
168	case ItemsInventoryID::RedCommon:	
169	body = BodyID::RedCommon;	
170	break;	
171	case ItemsInventoryID::LeatherArmor:	
172	body = BodyID::LeatherArmor;	
173	break;	
174	case ItemsInventoryID::BlueTunic:	
175	body = BodyID::BlueTunic;	
176	break;	
177	case ItemsInventoryID::PlateArmor:	
178	body = BodyID::PlateArmor;	
179	break;	
180	default:	
181	body = BodyID::Nothing;	
182	break;	
183	}	
184	return body;	
185	}	
186		
187	HelmetID ItemTranslator::itemToHelmet(ItemsInventoryID item){	
188	HelmetID helmet = HelmetID::Nothing;	
189	switch (item) {	
190	case ItemsInventoryID::Hood:	
191	helmet = HelmetID::Hood;	
192	break;	
193	case ItemsInventoryID::MagicHat:	
194	helmet = HelmetID::MagicHat;	
195	break;	
196	case ItemsInventoryID::IronHelmet:	
197	helmet = HelmetID::IronHelmet;	
198	break;	

jul 21, 20 15:20	ItemTranslator.cpp	Page 4/4
199	case ItemsInventoryID::Nothing:	
200	helmet = HelmetID::Nothing;	
201	break;	
202	default:	
203	helmet = HelmetID::Nothing;	
204	break;	
205	}	
206	return helmet;	
207	}	
208		
209	ItemTranslator::~ItemTranslator() = default;	

jul 21, 20 15:20	Inventory.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_INVENTORY_H</code>	
2	<code>#define ARGENTUM_TALLER_INVENTORY_H</code>	
3		
4		
5	<code>#include "../common/Identificators.h"</code>	
6	<code>#include <vector></code>	
7		
8	<code>class Inventory {</code>	
9	<code>private:</code>	
10	<code>std::vector<ItemsInventoryID> inventoryItems;</code>	
11	<code>uint8_t itemsAmount;</code>	
12	<code>uint8_t limit;</code>	
13	<code>public:</code>	
14	<code>Inventory();</code>	
15		
16	<code>ItemsInventoryID getItem(int index) const;</code>	
17		
18		
19	<code>bool addItem(ItemsInventoryID aItemInventoryId);</code>	
20	<code>bool isEmpty();</code>	
21		
22	<code>bool isFull() const;</code>	
23		
24	<code>const std::vector<ItemsInventoryID> &getInventoryItems() const;</code>	
25		
26	<code>ItemsInventoryID removeItem(ItemsInventoryID aItemToRemove);</code>	
27		
28	<code>std::string getStringInventory() const;</code>	
29		
30		
31	<code>void clear();</code>	
32		
33	<code>virtual ~Inventory();</code>	
34	<code>};</code>	
35		
36		
37	<code>#endif //ARGENTUM_TALLER_INVENTORY_H</code>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Inventory.cpp	Page 1/2
1	<code>#include "Inventory.h"</code>	
2		
3	<code>#include <utility></code>	
4	<code>#include <algorithm></code>	
5	<code>#include "GameStatsConfig.h"</code>	
6		
7	<code>Inventory::Inventory() :</code>	
8	<code>inventoryItems(), itemsAmount(0),</code>	
9	<code>limit(GameStatsConfig::getInventoryLimit()){</code>	
10	<code>for (int i = 0; i < limit; ++i){</code>	
11	<code>inventoryItems.push_back(ItemsInventoryID::Nothing);</code>	
12	<code>}</code>	
13	<code>}</code>	
14		
15	<code>ItemsInventoryID Inventory::getItem(int index) const {</code>	
16	<code>return inventoryItems.at(index);</code>	
17	<code>}</code>	
18		
19	<code>bool Inventory::addItem(ItemsInventoryID aItemInventoryId) {</code>	
20	<code>bool added = false;</code>	
21	<code>if (!isFull() ^ aItemInventoryId != ItemsInventoryID::Nothing) {</code>	
22	<code>auto iter = std::find(inventoryItems.begin(), inventoryItems.end(), Item</code>	
23	<code>sInventoryID::Nothing);</code>	
24	<code>(*iter) = aItemInventoryId;</code>	
25	<code>itemsAmount++;</code>	
26	<code>added = true;</code>	
27	<code>return added;</code>	
28	<code>}</code>	
29		
30	<code>bool Inventory::isFull() const {</code>	
31	<code>return itemsAmount == limit;</code>	
32	<code>}</code>	
33		
34	<code>void Inventory::clear() {</code>	
35	<code>inventoryItems.clear();</code>	
36	<code>itemsAmount = 0;</code>	
37	<code>for (int i = 0; i < limit; ++i){</code>	
38	<code>inventoryItems.push_back(ItemsInventoryID::Nothing);</code>	
39	<code>}</code>	
40	<code>}</code>	
41		
42	<code>std::string Inventory::getStringInventory() const {</code>	
43	<code>std::string inv;</code>	
44	<code>std::string temp;</code>	
45	<code>for (int i = 0; i < limit; ++i){</code>	
46	<code>temp = std::to_string((int)inventoryItems.at(i));</code>	
47	<code>if (temp.size() == 1)</code>	
48	<code>inv += "0";</code>	
49	<code>inv += temp;</code>	
50	<code>if (i != limit-1)</code>	
51	<code>inv += " ";</code>	
52	<code>}</code>	
53	<code>return inv;</code>	
54	<code>}</code>	
55		
56	<code>ItemsInventoryID Inventory::removeItem(ItemsInventoryID aItemToRemove) {</code>	
57	<code>auto iter = std::find(inventoryItems.begin(), inventoryItems.end(), aItemToR</code>	
58	<code>emove);</code>	
59	<code>if (iter == inventoryItems.end() ^ aItemToRemove == ItemsInventoryID::Nothing)</code>	
60	<code>{</code>	
61	<code>return ItemsInventoryID::Nothing;</code>	
62	<code>(*iter) = ItemsInventoryID::Nothing;</code>	
63	<code>itemsAmount--;</code>	
64	<code>return aItemToRemove;</code>	

50/217

jul 21, 20 15:20	Inventory.cpp	Page 2/2
64	}	
65		
66	const std::vector<ItemsInventoryID> &Inventory::getInventoryItems() const {	
67	return inventoryItems;	
68	}	
69		
70	bool Inventory::isEmpty() {	
71	return itemsAmount == 0;	
72	}	
73		
74	Inventory::~Inventory() = default;	

jul 21, 20 15:20	GameStats.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_GAMESTATS_H	
2	#define ARGENTUM_TALLER_GAMESTATS_H	
3		
4	#include <stdint>	
5		
6	struct RaceInfo {	
7	float intelligent;	
8	float recoveryTime;	
9	float health;	
10	float mana;	
11	float constitution;	
12	float strength;	
13	float agility;	
14	};	
15		
16	struct GameClassInfo {	
17	float health;	
18	float mana;	
19	float meditation;	
20	};	
21		
22	struct ItemInfo {	
23	uint8_t id;	
24	std::string name;	
25	bool damage;	
26	uint8_t range;	
27	uint8_t minDamage;	
28	uint8_t maxDamage;	
29	uint8_t manaUsed;	
30	uint8_t manaRestored;	
31	uint8_t healthRestored;	
32	uint8_t minDefense;	
33	uint8_t maxDefense;	
34	uint8_t goldCost;	
35	std::string type;	
36	};	
37		
38	struct CreatureInfo {	
39	float health;	
40	float strength;	
41	float agility;	
42	float constitution;	
43	};	
44		
45	#endif //ARGENTUM_TALLER_GAMESTATS_H	

jul 21, 20 15:20	GameStatsConfig.h	Page 1/2
1	#ifndef ARGENTUM_TALLER_GAMESTATSCONFIG_H	
2	#define ARGENTUM_TALLER_GAMESTATSCONFIG_H	
3		
4	#include <unordered_map>	
5	#include "GameStats.h"	
6	#include "../common/Identificators.h"	
7	#include <rapidjson/document.h>	
8	#include <map>	
9		
10	class GameStatsConfig {	
11	private:	
12	static std::unordered_map<RaceID, RaceInfo, std::hash<RaceID>> races;	
13	static std::unordered_map<GameClassID, GameClassInfo, std::hash<GameClassID>	
14	> gameClasses;	
15	static std::map<ItemsInventoryID, ItemInfo> items;	
16	static std::unordered_map<CreatureID, CreatureInfo, std::hash<CreatureID>> c	
17	reatures;	
18	static std::string port;	
19	static float goldRandMin;	
20	static float goldRandMax;	
21	static float goldMaxMult;	
22	static float goldMaxPot;	
23	static float expMaxMult;	
24	static float expMaxPot;	
25	static float evadeRandMin;	
26	static float evadeRandMax;	
27	static float evadeProbability;	
28	static float expRandMin;	
29	static float expRandMax;	
30	static uint8_t levelDifference;	
31	static float minAgility;	
32	static uint8_t creaturesLimit;	
33	static uint8_t nestCreaturesLimit;	
34	static float distance;	
35	static int inventoryLimit;	
36	static int newbieLevel;	
37	static float loseExp;	
38	RaceInfo createRaceInfo(rapidjson::Value &value);	
39	GameClassInfo createGameClass(rapidjson::Value &value);	
40	ItemInfo createItem(rapidjson::Value& value);	
41	CreatureInfo createCreatureInfo(rapidjson::Value &value);	
42	public:	
43	GameStatsConfig();	
44	explicit GameStatsConfig(rapidjson::Document &json);	
45	virtual ~GameStatsConfig();	
46	static std::string getPort();	
47	static float getMaxHealth(RaceID raceId, GameClassID gameClassId, uint level	
48);	
49	static float getMaxHealth(CreatureID creatureId, uint level);	
50	static float getRecoveryHealth(RaceID raceId);	
51	static float getMaxMana(RaceID raceId, GameClassID gameClassId, uint level);	
52	static float getRecoveryMana(RaceID raceId);	
53	static float getRecoveryManaMeditation(RaceID raceId, GameClassID gameClassI	
54	d);	
55	static float getGoldDrop(CreatureID creatureId, uint level);	
56	static float getMaxGold(uint level);	
57	static float getNextLevelLimit(uint level);	
58	static float getExp(float damage, uint level, uint enemyLevel);	
59	static float getAdditionalExp(float damage, float enemyMaxLife, uint level,	
60	uint enemyLevel);	
61	static float getDamage(RaceID race, WeaponID weaponId);	
62	static float getDamage(CreatureID creatureId);	
63	static bool canEvade(RaceID raceId);	
64	static bool canEvade(CreatureID creatureId);	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	GameStatsConfig.h	Page 2/2
62	static float getDefense(BodyID bodyId, ShieldID shieldId, HelmetID helmetId)	
63);	
64	static float getDefense(CreatureID creatureId);	
65	static std::map<ItemsInventoryID, ItemInfo> getItems();	
66	static uint8_t getAmountMovement(RaceID raceId);	
67	static uint8_t getAmountMovement(CreatureID creatureId);	
68	static uint8_t getCreaturesLimit();	
69	static const ItemInfo getItem(ItemsInventoryID id);	
70	static uint8_t getNestCreatureLimit();	
71	static float getDistance();	
72	static int getInventoryLimit();	
73	static int getWeaponDistance(WeaponID aWeaponId);	
74	static int getWeaponCost(WeaponID aWeaponId);	
75	static bool canAttack(int level, int enemyLevel);	
76	static bool isNewbie(int level);	
77	static float restoreHealth(WeaponID aWeaponId);	
78	static float getLoseExp();	
79	};	
80		
81	#endif //ARGENTUM_TALLER_GAMESTATSCONFIG_H	

52/217

jul 21, 20 15:20	GameStatsConfig.cpp	Page 1/5
1	#include "GameStatsConfig.h"	
2	#include "../common/Random.h"	
3	#include "ItemTranslator.h"	
4	#include <iostream>	
5		
6	GameStatsConfig::GameStatsConfig(rapidjson::Document &json) {	
7	GameStatsConfig gameStatsConfig;	
8	port = json["port"].GetString();	
9	goldRandMin = json["goldRandMin"].GetFloat();	
10	goldRandMax = json["goldRandMax"].GetFloat();	
11	goldMaxMult = json["goldMaxMult"].GetFloat();	
12	goldMaxPot = json["goldMaxPot"].GetFloat();	
13	expRandMin = json["expRandMin"].GetFloat();	
14	expRandMax = json["expRandMax"].GetFloat();	
15	expMaxMult = json["expMaxMult"].GetFloat();	
16	expMaxPot = json["expMaxPot"].GetFloat();	
17	evadeRandMin = json["evadeRandMin"].GetFloat();	
18	evadeRandMax = json["evadeRandMax"].GetFloat();	
19	evadeProbability = json["evadeProbability"].GetFloat();	
20	levelDifference = json["levelDifference"].GetFloat();	
21	minAgility = json["minAgility"].GetFloat();	
22	creaturesLimit = json["creaturesLimit"].GetInt();	
23	nestCreaturesLimit = json["nestCreaturesLimit"].GetInt();	
24	distance = json["distance"].GetFloat();	
25	inventoryLimit = json["inventoryLimit"].GetInt();	
26	newbieLevel = json["newbieLevel"].GetInt();	
27	loseExp = json["loseExp"].GetFloat();	
28		
29	rapidjson::Value::Array racesArray = json["races"].GetArray();	
30	for (auto &aRace : racesArray) {	
31	races.insert(std::pair<RaceID, RaceInfo>(RaceID(aRace["id"].GetInt()), c	
32	reateRaceInfo(aRace));	
33		
34	rapidjson::Value::Array gameClassesArray = json["classes"].GetArray();	
35	for (auto &aGameClass : gameClassesArray) {	
36	gameClasses.insert(std::pair<GameClassID, GameClassInfo>(GameClassID(aGa	
37	meClass["id"].GetInt()), createGameClass(aGameClass));	
38	}	
39	rapidjson::Value::Array itemsArray = json["items"].GetArray();	
40	for (auto &aItem : itemsArray) {	
41	items.insert(std::pair<ItemsInventoryID, ItemInfo>(ItemsInventoryID(aItem	
42	m["id"].GetInt()), createItem(aItem));	
43	}	
44	rapidjson::Value::Array creaturesArray = json["creatures"].GetArray();	
45	for (auto &aCreature : creaturesArray) {	
46	creatures.insert(std::pair<CreatureID, CreatureInfo>(CreatureID(aCreatur	
47	e["id"].GetInt()), createCreatureInfo(aCreature));	
48	}	
49	}	
50	RaceInfo GameStatsConfig::createRaceInfo(rapidjson::Value &value) {	
51	RaceInfo aRaceInfo{};	
52	aRaceInfo.intelligent = value["intelligent"].GetFloat();	
53	aRaceInfo.recoveryTime = value["recoveryTime"].GetFloat();	
54	aRaceInfo.constitution = value["constitution"].GetFloat();	
55	aRaceInfo.strength = value["strength"].GetFloat();	
56	aRaceInfo.agility = value["agility"].GetFloat();	
57	aRaceInfo.health = value["health"].GetFloat();	
58	aRaceInfo.mana = value["mana"].GetFloat();	
59	return aRaceInfo;	
60	}	
61	GameClassInfo GameStatsConfig::createGameClass(rapidjson::Value &value) {	
62	GameClassInfo aGameClassInfo{};	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	GameStatsConfig.cpp	Page 2/5
63	aGameClassInfo.health = value["health"].GetFloat();	
64	aGameClassInfo.mana = value["mana"].GetFloat();	
65	aGameClassInfo.meditation = value["meditation"].GetFloat();	
66	return aGameClassInfo;	
67	}	
68		
69	ItemInfo GameStatsConfig::createItem(rapidjson::Value& value) {	
70	ItemInfo aItemInfo{};	
71	aItemInfo.name = value["name"].GetString();	
72	aItemInfo.damage = value["damage"].GetBool();	
73	aItemInfo.minDamage = value["minDamage"].GetInt();	
74	aItemInfo.maxDamage = value["maxDamage"].GetInt();	
75	aItemInfo.manaUsed = value["manaUsed"].GetInt();	
76	aItemInfo.manaRestored = value["manaRestored"].GetInt();	
77	aItemInfo.healthRestored = value["healthRestored"].GetInt();	
78	aItemInfo.minDefense = value["minDefense"].GetInt();	
79	aItemInfo.maxDefense = value["maxDefense"].GetInt();	
80	aItemInfo.goldCost = value["goldCost"].GetInt();	
81	aItemInfo.type = value["type"].GetString();	
82	aItemInfo.range = value["range"].GetInt();	
83	return aItemInfo;	
84	}	
85		
86	CreatureInfo GameStatsConfig::createCreatureInfo(rapidjson::Value& value) {	
87	CreatureInfo aCreatureInfo{};	
88	aCreatureInfo.strength = value["strength"].GetFloat();	
89	aCreatureInfo.agility = value["agility"].GetFloat();	
90	aCreatureInfo.health = value["health"].GetFloat();	
91	aCreatureInfo.constitution = value["constitution"].GetFloat();	
92	return aCreatureInfo;	
93	}	
94		
95	const ItemInfo GameStatsConfig::getItem(ItemsInventoryID id) {	
96	return items.at(id);	
97	}	
98		
99	float GameStatsConfig::getMaxHealth(RaceID raceId, GameClassID gameClass, uint l	
100	evel) {	
101	float max = races.at(raceId).constitution * gameClasses.at(gameClass).health	
102	* races.at(raceId).health * level;	
103	return max;	
104	}	
105	float GameStatsConfig::getRecoveryHealth(RaceID raceId) {	
106	return races.at(raceId).recoveryTime / (45 * 10);	
107	}	
108	float GameStatsConfig::getMaxMana(RaceID raceId, GameClassID gameClass, uint lev	
109	el) {	
110	float max = races.at(raceId).intelligent * races.at(raceId).mana * gameClass	
111	es.at(gameClass).mana * level;	
112	return max;	
113	}	
114	float GameStatsConfig::getRecoveryMana(RaceID race) {	
115	return races.at(race).recoveryTime / (45 * 10);	
116	}	
117	float GameStatsConfig::getRecoveryManaMeditation(RaceID race, GameClassID gameCl	
118	ass) {	
119	return (races.at(race).intelligent * gameClasses.at(gameClass).meditation) /	
120	(45 * 10);	
121	}	
122	float GameStatsConfig::getGoldDrop(CreatureID creatureId, uint level){	
123	return Random::getFloat(goldRandMin, goldRandMax) * GameStatsConfig::getMaxH	

53/217

jul 21, 20 15:20	GameStatsConfig.cpp	Page 3/5
123	health(creatureId, level);	
124	}	
125	float GameStatsConfig::getMaxGold(uint level){	
126	return goldMaxMult * pow(level, goldMaxPot);	
127	}	
128		
129	float GameStatsConfig::getNextLevelLimit(uint level){	
130	return expMaxMult * pow(level, expMaxPot);	
131	}	
132		
133	float GameStatsConfig::getExp(float damage, uint level, uint enemyLevel) {	
134	return damage * std::max((int)(enemyLevel - level + levelDifference), 0);	
135	}	
136		
137	float GameStatsConfig::getAdditionalExp(float damage, float enemyMaxLife, uint	
138	level, uint enemyLevel){	
139	return Random::getFloat(expRandMin, expRandMax) * enemyMaxLife * std::max((int)(enemyLevel - level + levelDifference), 0);	
140	}	
141	float GameStatsConfig::getDamage(RaceID raceId, WeaponID weaponId){	
142	RaceInfo aRaceInfo = races.at(raceId);	
143	if (weaponId == WeaponID::Nothing) {	
144	return aRaceInfo.strength;	
145	}	
146	ItemInfo aWeapon = items.at(ItemTranslator::weaponToItem(weaponId));	
147	return aRaceInfo.strength * Random::get(aWeapon.minDamage, aWeapon.maxDamage	
148	};	
149		
150	bool GameStatsConfig::canEvade(RaceID race){	
151	double base = Random::getFloat(evadeRandMin, evadeRandMax);	
152	return pow(base, races.at(race).agility) < evadeProbability;	
153	}	
154		
155	uint8_t GameStatsConfig::getAmountMovement(RaceID raceId) {	
156	return std::max(races.at(raceId).agility * 1.5f, minAgility);	
157	}	
158		
159	uint8_t GameStatsConfig::getCreaturesLimit() {	
160	return creaturesLimit;	
161	}	
162		
163	uint8_t GameStatsConfig::getNestCreatureLimit() {	
164	return nestCreaturesLimit;	
165	}	
166		
167	std::string GameStatsConfig::getPort() {	
168	return port;	
169	}	
170		
171	std::map<ItemsInventoryID, ItemInfo> GameStatsConfig::getItems() {	
172	return items;	
173	}	
174		
175	float GameStatsConfig::getDistance(){	
176	return distance;	
177	}	
178		
179	uint8_t GameStatsConfig::getAmountMovement(CreatureID creatureId) {	
180	return std::max(creatures.at(creatureId).agility, minAgility) / 2;	
181	}	
182		
183	float GameStatsConfig::getDamage(CreatureID creatureId){	
184	return creatures.at(creatureId).strength;	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	GameStatsConfig.cpp	Page 4/5
185	}	
186		
187	float GameStatsConfig::getDefense(BodyID bodyId, ShieldID shieldId, HelmetID hel	
188	metId) {	
189	float defense = 0.0;	
190	ItemInfo aItemInfo;	
191	if (bodyId != BodyID::Nothing) {	
192	aItemInfo = items.at(ItemTranslator::bodyToItem(bodyId));	
193	defense += Random::get(aItemInfo.minDefense, aItemInfo.maxDefense);	
194	}	
195	if (shieldId != ShieldID::Nothing) {	
196	aItemInfo = items.at(ItemTranslator::shieldToItem(shieldId));	
197	defense += Random::get(aItemInfo.minDefense, aItemInfo.maxDefense);	
198	}	
199	if (helmetId != HelmetID::Nothing) {	
200	aItemInfo = items.at(ItemTranslator::helmetToItem(helmetId));	
201	defense += Random::get(aItemInfo.minDefense, aItemInfo.maxDefense);	
202	}	
203	return defense;	
204	}	
205	float GameStatsConfig::getDefense(CreatureID creatureId) {	
206	return Random::getFloat(0.0, creatures.at(creatureId).constitution);	
207	}	
208		
209	bool GameStatsConfig::canEvade(CreatureID creatureId) {	
210	double base = Random::getFloat(evadeRandMin, evadeRandMax);	
211	return pow(base, creatures.at(creatureId).agility) < evadeProbability;	
212	}	
213		
214	float GameStatsConfig::getMaxHealth(CreatureID creatureId, uint level) {	
215	return creatures.at(creatureId).health * level;	
216	}	
217		
218	int GameStatsConfig::getInventoryLimit() {	
219	return GameStatsConfig::inventoryLimit;	
220	}	
221		
222	int GameStatsConfig::getWeaponDistance(WeaponID aWeaponId) {	
223	if (aWeaponId == WeaponID::Nothing) {	
224	return 1;	
225	}	
226	ItemInfo aItemInfo = items.at(ItemTranslator::weaponToItem(aWeaponId));	
227	return aItemInfo.range;	
228	}	
229		
230	int GameStatsConfig::getWeaponCost(WeaponID aWeaponId) {	
231	return items.at(ItemTranslator::weaponToItem(aWeaponId)).manaUsed;	
232	}	
233		
234	bool GameStatsConfig::canAttack(int level, int enemyLevel) {	
235	if (isNewbie(level) isNewbie(enemyLevel))	
236	return false;	
237	int diff = level - enemyLevel;	
238	if (diff < -GameStatsConfig::levelDifference diff > GameStatsConfig::leve	
239	lDifference)	
240	return false;	
241	return true;	
242	}	
243		
244	bool GameStatsConfig::isNewbie(int level){	
245	return level ≤ GameStatsConfig::newbieLevel;	
246	}	
247		
248	float GameStatsConfig::restoreHealth(WeaponID aWeaponId) {	
249	if (aWeaponId == WeaponID::Nothing) {	

54/217

jul 21, 20 15:20	GameStatsConfig.cpp	Page 5/5
249	return 0;	
250	}	
251	ItemInfo aItemInfo = items.at(ItemTranslator::weaponToItem(aWeaponId));	
252	return aItemInfo.healthRestored;	
253	}	
254		
255	float GameStatsConfig::getLoseExp() {	
256	return loseExp;	
257	}	
258		
259	GameStatsConfig::GameStatsConfig() = default ;	
260		
261	GameStatsConfig::~GameStatsConfig() = default ;	
262		
263	std::unordered_map<RaceID, RaceInfo, std::hash<RaceID>> GameStatsConfig::races;	
264	std::unordered_map<GameClassID, GameClassInfo, std::hash<GameClassID>> GameStatsConfig::gameClasses;	
265	std::map<ItemsInventoryID, ItemInfo> GameStatsConfig::items;	
266	std::unordered_map<CreatureID, CreatureInfo, std::hash<CreatureID>> GameStatsConfig::creatures;	
267	std::string GameStatsConfig::port{};	
268	float GameStatsConfig::goldRandMin = 0.0;	
269	float GameStatsConfig::goldRandMax = 0.0;	
270	float GameStatsConfig::goldMaxMult = 0.0;	
271	float GameStatsConfig::goldMaxPot = 0.0;	
272	float GameStatsConfig::expMaxMult = 0.0;	
273	float GameStatsConfig::expMaxPot = 0.0;	
274	float GameStatsConfig::evadeRandMin = 0.0;	
275	float GameStatsConfig::evadeRandMax = 0.0;	
276	float GameStatsConfig::evadeProbability = 0.0;	
277	float GameStatsConfig::expRandMin = 0.0;	
278	float GameStatsConfig::expRandMax = 0.0;	
279	uint8_t GameStatsConfig::levelDifference = 0;	
280	float GameStatsConfig::minAgility = 0.0;	
281	uint8_t GameStatsConfig::creaturesLimit = 0.0;	
282	uint8_t GameStatsConfig::nestCreaturesLimit = 0.0;	
283	float GameStatsConfig::distance = 0.0;	
284	int GameStatsConfig::inventoryLimit = 0;	
285	int GameStatsConfig::newbieLevel = 0;	
286	float GameStatsConfig::loseExp = 0;	

jul 21, 20 15:20	GameObjectsContainer.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_GAMEOBJECTSCONTAINER_H	
2	#define ARGENTUM_TALLER_GAMEOBJECTSCONTAINER_H	
3		
4		
5	#include <memory>	
6	#include <unordered_map>	
7	#include "GameObject.h"	
8		
9	class GameObjectsContainer {	
10	private:	
11	std::unordered_map<uint, std::shared_ptr<GameObject>, std::hash<uint>> gameObjects;	
12	public:	
13	GameObjectsContainer();	
14		
15	void addGameObject(std::shared_ptr<GameObject> aGameObject, uint id);	
16		
17	std::vector<GameObjectInfo> getUpdatedGameObjectsInfo();	
18		
19	std::vector<std::shared_ptr<GameObject>> getUpdatedGameObjects();	
20		
21	void update(Board& board);	
22		
23	std::shared_ptr<GameObject> getGameObject(uint id);	
24		
25	virtual ~GameObjectsContainer();	
26		
27	void deleteGameObject(const uint id, Board &board);	
28		
29	void removeCreaturesAndItems(Board& board);	
30	};	
31	#endif //ARGENTUM_TALLER_GAMEOBJECTSCONTAINER_H	

jul 21, 20 15:20	GameObjectsContainer.cpp	Page 1/1
1	#include <iostream>	
2	#include "GameObjectsContainer.h"	
3		
4	GameObjectsContainer::GameObjectsContainer() = default ;	
5		
6	GameObjectsContainer::~GameObjectsContainer() = default ;	
7		
8	void GameObjectsContainer::addGameObject(std::shared_ptr<GameObject> aGameObject	
9	, uint id) {	
10	gameObjects.insert(std::pair<uint, std::shared_ptr<GameObject>>(id, aGameObj	
11	ect));	
12	}	
13	std::vector<GameObjectInfo> GameObjectsContainer::getUpdatedGameObjectsInfo() {	
14	std::vector<GameObjectInfo> gameObjectsInfo;	
15	for (auto& gameObjectPair : gameObjects) {	
16	gameObjectsInfo.push_back(gameObjectPair.second->getGameObjectInfo());	
17	}	
18	return gameObjectsInfo;	
19	}	
20	std::vector<std::shared_ptr<GameObject>> GameObjectsContainer::getUpdatedGameObj	
21	ects() {	
22	std::vector<std::shared_ptr<GameObject>> objects;	
23	for (auto& gameObjectPair : gameObjects) {	
24	objects.push_back(gameObjectPair.second);	
25	}	
26	return objects;	
27	}	
28		
29	void GameObjectsContainer::update(Board& board) {	
30	for (auto& gameObjectPair : gameObjects) {	
31	gameObjectPair.second->update(gameObjects, board);	
32	}	
33	}	
34		
35	std::shared_ptr<GameObject> GameObjectsContainer::getGameObject(uint id) {	
36	return gameObjects.at(id);	
37	}	
38		
39	void GameObjectsContainer::deleteGameObject(uint id, Board &board) {	
40	this ->gameObjects.at(id)->remove(board);	
41	this ->gameObjects.erase(id);	
42	}	
43		
44	void GameObjectsContainer::removeCreaturesAndItems(Board &board) {	
45	auto iter = this ->gameObjects.begin();	
46	while (iter != this ->gameObjects.end()) {	
47	if ((*iter).second->isReadyToRemove()) {	
48	(*iter).second->remove(board);	
49	iter = this ->gameObjects.erase(iter);	
50	} else {	
51	iter++;	
52	}	
53	}	
54	}	

jul 21, 20 15:20	GameObject.h	Page 1/2
1	#ifndef OBJETOJUEGO_H	
2	#define OBJETOJUEGO_H	
3		
4	#include <string>	
5	#include "../common/Identificators.h"	
6	#include "../common/GameObjectInfo.h"	
7	#include "GameStatsConfig.h"	
8	#include "Board.h"	
9	#include "DropItem.h"	
10		
11	class GameObject{	
12	protected:	
13	Point point;	
14	std::shared_ptr<Cell> cell;	
15	uint id{};	
16	std::string textureHashId;	
17	Direction direction;	
18	NPCInfo infoInteracting;	
19	uint level;	
20	WeaponID interactWeapon;	
21	public:	
22	explicit GameObject(uint id, Point initialPoint, std::shared_ptr<Cell> initial	
23	Cell, Direction aDirection = Direction::down);	
24		
25	GameObjectInfo getGameObjectInfo();	
26		
27	virtual PlayerInfo getPlayerInfo();	
28		
29	uint getId() const ;	
30		
31	void setDirection(Direction direction);	
32		
33	void setCell(std::shared_ptr<Cell> aCell);	
34		
35	void setPoint(Point aPoint);	
36		
37	virtual bool isItem() = 0;	
38		
39	virtual bool canDropsItems() = 0;	
40		
41	virtual std::vector<DropItem> getDrop() = 0;	
42		
43	void setInteractWeapon(WeaponID interactWeapon);	
44		
45	std::shared_ptr<Cell> &getActualCell();	
46		
47	void setTextureHashId(const std::string &textureHashId);	
48		
49	virtual float getMaxLife() = 0;	
50		
51	uint getLevel();	
52	virtual void update(std::unordered_map<uint, std::shared_ptr<GameObject>> &g	
53	ameObjects, Board& board) = 0;	
54		
55	virtual CharacterStateID getStateId() = 0;	
56		
57	virtual bool isDead() = 0;	
58		
59	virtual void receiveDamage(float damage, WeaponID weaponId) = 0;	
60		
61	virtual NPCInfo interact(GameObject& character, InputInfo input) = 0;	
62		
63	NPCInfo getInteractInfo() const ;	
64	void setInteractInfo(NPCInfo info);	

jul 21, 20 15:20	GameObject.h	Page 2/2
65	virtual bool isReadyToRemove() = 0;	
66		
67	virtual void remove(Board &board) = 0;	
68		
69	virtual bool canBeAttacked(int enemyLevel) const = 0;	
70		
71	virtual bool hasAnInputInfo() = 0;	
72		
73	virtual InputInfo getNextInputInfo() = 0;	
74		
75	~GameObject();	
76	};	
77		
78		
79	#endif	

jul 21, 20 15:20	GameObject.cpp	Page 1/1
1	#include "GameObject.h"	
2		
3	#include <utility>	
4		
5	GameObjectInfo GameObject::getGameObjectInfo() {	
6	return GameObjectInfo(id, point, textureHashId, direction, getStateId(), isIt	
7	em(), interactWeapon);	
8	}	
9	uint GameObject::getId() const {	
10	return id;	
11	}	
12		
13	void GameObject::setDirection(Direction direction) {	
14	GameObject::direction = direction;	
15	}	
16		
17	void GameObject::setTextureHashId(const std::string &textureHashId) {	
18	GameObject::textureHashId = textureHashId;	
19	}	
20		
21	GameObject::GameObject(uint id, Point initialPoint, std::shared_ptr<Cell> initia	
22	lCell, Direction aDirection) :	
23	point(initialPoint), cell(std::move(initialCell)), id(id), textureHashId(),	
24	direction(aDirection),	
25	infoInteracting(), level(1), interactWeapon(WeaponID::Nothing) {}	
26		
27	std::shared_ptr<Cell> &GameObject::getActualCell() {	
28	return cell;	
29	}	
30		
31	void GameObject::setCell(std::shared_ptr<Cell> aCell) {	
32	cell = std::move(aCell);	
33	}	
34		
35	void GameObject::setPoint(Point aPoint) {	
36	point = aPoint;	
37	}	
38		
39	NPCInfo GameObject::getInteractInfo() const {	
40	return this->infoInteracting;	
41	}	
42		
43	void GameObject::setInteractInfo(NPCInfo info) {	
44	this->infoInteracting = info;	
45	}	
46		
47	uint GameObject::getLevel() {	
48	return level;	
49	}	
50		
51	PlayerInfo GameObject::getPlayerInfo() {	
52	return PlayerInfo();	
53	}	
54		
55	void GameObject::setInteractWeapon(WeaponID interactWeapon) {	
56	GameObject::interactWeapon = interactWeapon;	
57	}	
	GameObject::~GameObject()= default;	

jul 21, 20 15:20	GameCharacter.h	Page 1/2
1	#ifndef PERSONAJE_H	
2	#define PERSONAJE_H	
3		
4	#include "../common/PlayerInfo.h"	
5	#include "../common/StaticObject.h"	
6	#include "../common/InputQueue.h"	
7	#include "../common/Identificators.h"	
8	#include "Inventory.h"	
9	#include "GameObject.h"	
10	#include <vector>	
11	#include <memory>	
12	#include <states/StatePoolCharacter.h>	
13		
14	class GameCharacter : public GameObject{	
15	private:	
16	RaceID race{RaceID::Nothing};	
17	GameClassID gameClass{GameClassID::Nothing};	
18	uint goldAmount;	
19	float life;	
20	float mana;	
21	float exp;	
22	StatePoolCharacter statePool;	
23	InputQueue queueInputs;	
24	Inventory inventory;	
25	WeaponID weapon{WeaponID::Nothing};	
26	ShieldID shield{ShieldID::Nothing};	
27	HelmetID helmet{HelmetID::Nothing};	
28	BodyID body{BodyID::Nothing};	
29	ItemsInventoryID itemToDrop = ItemsInventoryID::Nothing;	
30		
31	std::string updateTextureHashId();	
32	void consumePotion(const ItemInfo& potion);	
33		
34	public:	
35	GameCharacter(uint id, RaceID aRace, GameClassID aClass, std::shared_ptr<Cell> initialCell, Point initialPoint);	
36		
37	PlayerInfo getPlayerInfo() override;	
38		
39	void consumeMana();	
40		
41	void upLevel();	
42		
43	bool canUseWeapon();	
44		
45	bool hasAnInputInfo() override;	
46		
47	InputInfo getNextInputInfo() override;	
48		
49	bool restoreHealth();	
50		
51	bool inventoryIsFull();	
52		
53	bool addItemToInventory(ItemsInventoryID aItemInventoryId);	
54		
55	ItemsInventoryID removeItemFromInventory(ItemsInventoryID aItemToFind);	
56		
57	void gainGold(int aGoldAmount);	
58		
59	bool isReadyToRemove() override;	
60		
61	WeaponID getWeapon();	
62		
63	std::vector<DropItem> getDrop() override;	
64		
65	void gainExp(float newExp);	

jul 21, 20 15:20	GameCharacter.h	Page 2/2
66	bool isItem() override;	
67		
68	bool canDropsItems() override;	
69		
70	float getMaxLife() override;	
71		
72	void cure();	
73		
74	void update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjec ts, Board &board) override;	
75		
76	void receiveDamage(float damage, WeaponID weaponId) override;	
77		
78	bool isDead() override;	
79		
80	~GameCharacter();	
81		
82	RaceID getRace() const;	
83		
84	void remove(Board &board) override;	
85		
86	uint getGoldAmount();	
87		
88	void setGoldAmount(uint goldAmount);	
89		
90	CharacterStateID getStateId() override;	
91		
92	float getExp() const;	
93		
94	std::string getStringInventory() const;	
95		
96	InputQueue &getQueueInputs();	
97		
98	NPCInfo interact(GameObject& character, InputInfo input) override;	
99		
100	void equipItem(int itemToEquip);	
101		
102	void unequipItem(int itemToUnequip);	
103		
104	bool takeItem(ItemsInventoryID anItemId, int amount);	
105		
106	void dropItem(int index);	
107		
108	void updateHealthAndMana();	
109		
110	bool canBeAttacked(int enemyLevel) const override;	
111	};	
112		
113		
114	#endif	

jul 21, 20 15:20	GameCharacter.cpp	Page 1/6
1	<code>#include <iostream></code>	
2	<code>#include <utility></code>	
3	<code>#include <states/StateTranslator.h></code>	
4	<code>#include "GameCharacter.h"</code>	
5	<code>#include "GameStatsConfig.h"</code>	
6	<code>#include "../common/Random.h"</code>	
7	<code>#include "ItemTranslator.h"</code>	
8		
9	<code>PlayerInfo GameCharacter::getPlayerInfo() {</code>	
10	<code> return PlayerInfo(id, point, goldAmount, life, mana, textureHashId, direction,</code>	
11	<code> GameStatsConfig::getMaxHealth(race, gameClass, level),</code>	
12	<code> GameStatsConfig::getMaxMana(race, gameClass, level),</code>	
13	<code> exp, GameStatsConfig::getNextLevelLimit(level), level,</code>	
14	<code> getStringInventory(), StateTranslator::stateToCharacterState(statePool.getStateId()), interactWeapon);</code>	
15	<code>}</code>	
16		
17	<code>GameCharacter::GameCharacter(uint id, RaceID aRace, GameClassID aClass, std::shared_ptr<Cell> initialCell, Point initialPoint):</code>	
18	<code> GameObject(id, initialPoint, std::move(initialCell)), race(aRace), gameClass(aClass), statePool(*this), queueInputs(true), inventory() {</code>	
19		
20	<code> this->life = GameStatsConfig::getMaxHealth(race, gameClass, level);</code>	
21	<code> this->mana = GameStatsConfig::getMaxMana(race, gameClass, level);</code>	
22	<code> this->goldAmount = 100;</code>	
23	<code> this->exp = 0;</code>	
24	<code> this->direction = Direction::down;</code>	
25	<code> this->textureHashId = updateTextureHashId(); //Solo deberÃ-a tener la cabeza correspondiente y su cuerpo. "ht00/h03/b01/s00/w00"</code>	
26	<code>}</code>	
27		
28	<code>void GameCharacter::update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjects, Board &board) {</code>	
29	<code> this->infoInteracting.type = 0;</code>	
30	<code> statePool.updateState();</code>	
31	<code> statePool.performTask(gameObjects, board);</code>	
32	<code> this->textureHashId = updateTextureHashId();</code>	
33	<code> updateHealthAndMana();</code>	
34	<code>}</code>	
35		
36	<code>std::string GameCharacter::updateTextureHashId() {</code>	
37	<code> std::string equipment;</code>	
38	<code> std::string idHelmet = std::to_string((int)this->helmet);</code>	
39	<code> equipment += "h";</code>	
40	<code> if (idHelmet.size() == 1)</code>	
41	<code> equipment += "0";</code>	
42	<code> equipment += idHelmet + " ";</code>	
43	<code> equipment += "h";</code>	
44	<code> if (!isDead()) {</code>	
45	<code> std::string idHead = std::to_string((int)this->race);</code>	
46	<code> if (idHead.size() == 1)</code>	
47	<code> equipment += "0";</code>	
48	<code> equipment += idHead + " ";</code>	
49	<code> } else {</code>	
50	<code> equipment += "00";</code>	
51	<code> }</code>	
52	<code> equipment += "b";</code>	
53	<code> if (this->body == BodyID::Nothing (this->body == BodyID::Ghost ^ !isDead()))</code>	
54	<code> {</code>	
55	<code> this->body = (BodyID)(Random::get(1,3));</code>	
56	<code> }</code>	
57	<code> std::string idBody = std::to_string((int)this->body);</code>	
58	<code> if (idBody.size() == 1)</code>	
59	<code> equipment += "0";</code>	
60	<code> equipment += idBody + " ";</code>	

jul 21, 20 15:20	GameCharacter.cpp	Page 2/6
60	<code> std::string idShield = std::to_string((int)this->shield);</code>	
61	<code> equipment += "s";</code>	
62	<code> if (idShield.size() == 1)</code>	
63	<code> equipment += "0";</code>	
64	<code> equipment += idShield + " ";</code>	
65	<code> equipment += "w";</code>	
66	<code> std::string idWeapon = std::to_string((int)this->weapon);</code>	
67	<code> if (idWeapon.size() == 1)</code>	
68	<code> equipment += "0";</code>	
69	<code> equipment += idWeapon;</code>	
70	<code> return equipment;</code>	
71	<code>}</code>	
72		
73	<code>std::string GameCharacter::getStringInventory() const {</code>	
74	<code> return inventory.getStringInventory();</code>	
75	<code>}</code>	
76		
77	<code>void GameCharacter::equipItem(int itemToEquip) {</code>	
78	<code> ItemsInventoryID idItem = inventory.getItem(itemToEquip-1);</code>	
79	<code> if (idItem == ItemsInventoryID::Nothing)</code>	
80	<code> return;</code>	
81	<code> ItemInfo info = GameStatsConfig::getItem(idItem);</code>	
82	<code> ItemsInventoryID item = ItemsInventoryID::Nothing;</code>	
83	<code> if (info.type == "Weapon") {</code>	
84	<code> WeaponID newWeapon = ItemTranslator::itemToWeapon(idItem);</code>	
85	<code> item = ItemTranslator::weaponToItem(this->weapon);</code>	
86	<code> this->weapon = newWeapon;</code>	
87	<code> } else if (info.type == "Body") {</code>	
88	<code> BodyID newBody = ItemTranslator::itemToBody(idItem);</code>	
89	<code> item = ItemTranslator::bodyToItem(this->body);</code>	
90	<code> this->body = newBody;</code>	
91	<code> } else if (info.type == "Shield") {</code>	
92	<code> ShieldID newShield = ItemTranslator::itemToShield(idItem);</code>	
93	<code> item = ItemTranslator::shieldToItem(this->shield);</code>	
94	<code> this->shield = newShield;</code>	
95	<code> } else if (info.type == "Helmet") {</code>	
96	<code> HelmetID newHelmet = ItemTranslator::itemToHelmet(idItem);</code>	
97	<code> item = ItemTranslator::helmetToItem(this->helmet);</code>	
98	<code> this->helmet = newHelmet;</code>	
99	<code> } else if (info.type == "Potion") {</code>	
100	<code> this->consumePotion(info);</code>	
101	<code> }</code>	
102	<code> this->inventory.removeItem(idItem);</code>	
103	<code> this->inventory.addItem(item);</code>	
104	<code>}</code>	
105		
106	<code>void GameCharacter::unequipItem(int itemToUnequip) {</code>	
107	<code> ItemsInventoryID item = ItemsInventoryID::Nothing;</code>	
108	<code> if (!inventoryIsFull()) {</code>	
109	<code> switch (itemToUnequip) {</code>	
110	<code> case 0:</code>	
111	<code> item = ItemTranslator::helmetToItem(this->helmet);</code>	
112	<code> this->helmet = HelmetID::Nothing;</code>	
113	<code> break;</code>	
114	<code> case 1:</code>	
115	<code> item = ItemTranslator::weaponToItem(this->weapon);</code>	
116	<code> this->weapon = WeaponID::Nothing;</code>	
117	<code> break;</code>	
118	<code> case 2:</code>	
119	<code> item = ItemTranslator::shieldToItem(this->shield);</code>	
120	<code> this->shield = ShieldID::Nothing;</code>	
121	<code> break;</code>	
122	<code> }</code>	
123	<code> }</code>	
124	<code> this->inventory.removeItem(ItemsInventoryID::Nothing);</code>	
125	<code> this->inventory.addItem(item);</code>	

Page 3/6	GameCharacter.cpp	Page 3/6
126	}	
127		
128	void GameCharacter::consumePotion(const ItemInfo& potion) {	
129	uint maxMana = GameStatsConfig::getMaxMana(this->race, this->gameClass, this->	
130	level);	
131	uint maxHealth = GameStatsConfig::getMaxHealth(this->race, this->gameClass, t	
132	his->level);	
133	mana = mana + potion.manaRestored > maxMana ? maxMana : mana + potion.manaRe	
134	stored;	
135	life = life + potion.healthRestored > maxHealth ? maxHealth : life + potion.	
136	healthRestored;	
137	}	
138		
139	RaceID GameCharacter::getRace() const {	
140	return race;	
141	}	
142		
143	uint GameCharacter::getGoldAmount() {	
144	return goldAmount;	
145	}	
146		
147	float GameCharacter::getExp() const {	
148	return exp;	
149	}	
150		
151	InputQueue &GameCharacter::getQueueInputs() {	
152	return queueInputs;	
153	}	
154	CharacterStateID GameCharacter::getStateId() {	
155	return StateTranslator::stateToCharacterState(statePool.getStateId());	
156	}	
157		
158	void GameCharacter::receiveDamage(float damage, WeaponID weaponId) {	
159	setInteractWeapon(weaponId);	
160	if (GameStatsConfig::canEvade(race)) {	
161	std::cout << "Enemy fail attack" << std::endl;	
162	} else {	
163	float defense = GameStatsConfig::getDefense(body, shield, helmet);	
164	float realDamage = damage - defense;	
165	if (realDamage > 0) {	
166	life = (life - realDamage > 0) ? life - realDamage : 0;	
167	std::cout << "Enemy attack damage: " << damage << std::endl;	
168	std::cout << "Character defense: " << defense << std::endl;	
169	std::cout << "Enemy real damage: " << realDamage << std::endl;	
170	}	
171	if (isDead()) {	
172	this->mana = 0;	
173	body = BodyID::Ghost;	
174	shield = ShieldID::Nothing;	
175	weapon = WeaponID::Nothing;	
176	helmet = HelmetID::Nothing;	
177	exp = exp - GameStatsConfig::getLoseExp() < 0 ? 0 : exp - GameStatsConf	
178	g::getLoseExp();	
179	}	
180	}	
181	bool GameCharacter::isDead() {	
182	return life == 0;	
183	}	
184	bool GameCharacter::hasAnInputInfo() {	
185	return !queueInputs.empty();	
186	}	
187	InputInfo GameCharacter::getNextInputInfo() {	

Page 4/6	GameCharacter.cpp	Page 4/6
187	return queueInputs.pop();	
188	}	
189		
190	WeaponID GameCharacter::getWeapon() {	
191	return weapon;	
192	}	
193		
194	void GameCharacter::cure(){	
195	this->life = GameStatsConfig::getMaxHealth(race, gameClass, level);	
196	this->mana = GameStatsConfig::getMaxMana(race, gameClass, level);	
197	}	
198		
199		
200	NPCInfo GameCharacter::interact(GameObject& character, InputInfo input) {	
201	NPCInfo info;	
202	return info;	
203	}	
204		
205	bool GameCharacter::isReadyToRemove() {	
206	return false;	
207	}	
208		
209	void GameCharacter::remove(Board &board) {	
210	cell->free();	
211	}	
212		
213	void GameCharacter::gainExp(float newExp) {	
214	exp += newExp;	
215	}	
216		
217	float GameCharacter::getMaxLife() {	
218	return GameStatsConfig::getMaxHealth(race, gameClass, level);	
219	}	
220		
221	bool GameCharacter::inventoryIsFull() {	
222	return inventory.isFull();	
223	}	
224		
225	bool GameCharacter::addItemToInventory(ItemsInventoryID aItemInventoryId) {	
226	return inventory.addItem(aItemInventoryId);	
227	}	
228		
229	void GameCharacter::setGoldAmount(uint aGoldAmount) {	
230	GameCharacter::goldAmount = aGoldAmount;	
231	}	
232		
233	ItemsInventoryID GameCharacter::removeItemFromInventory(ItemsInventoryID aItemTo	
234	Remove) {	
235	return inventory.removeItem(aItemToRemove);	
236	}	
237		
238	void GameCharacter::gainGold(int aGoldAmount) {	
239	goldAmount += aGoldAmount;	
240	}	
241		
242	std::vector<DropItem> GameCharacter::getDrop() {	
243	std::vector<DropItem> dropsItems;	
244	if (isDead()) {	
245	for (auto &aInventoryItem : inventory.getInventoryItems()) {	
246	if (aInventoryItem != ItemsInventoryID::Nothing) {	
247	dropsItems.emplace_back(aInventoryItem, 1);	
248	}	
249	}	
250	int diffGold = goldAmount - GameStatsConfig::getMaxGold(level);	
251	if (diffGold > 0) {	
	goldAmount = GameStatsConfig::getMaxGold(level);	

jul 21, 20 15:20	GameCharacter.cpp	Page 5/6
252	dropsItems.emplace_back(ItemsInventoryID::Gold, diffGold);	
253	inventory.clear();	
254	} else {	
255	dropsItems.emplace_back(itemToDrop, 1);	
256	itemToDrop = ItemsInventoryID::Nothing;	
257	}	
258	return dropsItems;	
259	}	
260		
261	bool GameCharacter::isItem() {	
262	return false;	
263	}	
264		
265	bool GameCharacter::canDropItems() {	
266	return (isDead() ^ !inventory.isEmpty()) ^ itemToDrop != ItemsInventoryID::N	
267	othing;	
268	}	
269		
270	void GameCharacter::consumeMana() {	
271	if (weapon != WeaponID::Nothing) {	
272	mana -= GameStatsConfig::getWeaponCost(weapon);	
273	}	
274	}	
275		
276	void GameCharacter::upLevel() {	
277	level++;	
278	life = getMaxLife();	
279	mana = GameStatsConfig::getMaxMana(race, gameClass, level);	
280	}	
281		
282	bool GameCharacter::canUseWeapon() {	
283	return weapon != WeaponID::Nothing ^ GameStatsConfig::getWeaponCost(weapon) ≤	
284	mana;	
285	}	
286		
287	void GameCharacter::updateHealthAndMana() {	
288	if (!isDead()) {	
289	float manaMax = GameStatsConfig::getMaxMana(race, gameClass, level);	
290	float lifeIncrement = GameStatsConfig::getRecoveryHealth(race);	
291	life = life + lifeIncrement > getMaxLife() ? getMaxLife() : lifeIncremen	
292	t + life;	
293	float manaIncrement = statePool.isMeditating() ?	
294	GameStatsConfig::getRecoveryManaMeditation(race, g	
295	ameClass) :	
296	GameStatsConfig::getRecoveryMana(race);	
297	mana = mana + manaIncrement > manaMax ? manaMax : mana + manaIncrement;	
298	}	
299	}	
300		
301	bool GameCharacter::takeItem(ItemsInventoryID anItemId, int amount) {	
302	bool canTake;	
303	if (anItemId == ItemsInventoryID::Gold) {	
304	goldAmount += amount;	
305	canTake = true;	
306	} else {	
307	canTake = inventory.addItem(anItemId);	
308	}	
309	return canTake;	
310	}	
311		
312	void GameCharacter::dropItem(int index) {	
313	ItemsInventoryID itemToRemove = inventory.getItem(index);	
314	if (itemToRemove != ItemsInventoryID::Nothing) {	
315	itemToDrop = this->inventory.removeItem(itemToRemove);	
316	}	

jul 21, 20 15:20	GameCharacter.cpp	Page 6/6
314	}	
315		
316	bool GameCharacter::canBeAttacked(int enemyLevel) const {	
317	return GameStatsConfig::canAttack(this->level, enemyLevel);	
318	}	
319		
320	bool GameCharacter::restoreHealth() {	
321	bool canRestoreHealth = false;	
322	float healthRestore = 0;	
323	if (canUseWeapon()) {	
324	setInteractWeapon(weapon);	
325	healthRestore = GameStatsConfig::restoreHealth(weapon);	
326	canRestoreHealth = true;	
327	consumeMana();	
328	life = healthRestore + life > getMaxLife() ? getMaxLife() : healthRestor	
329	e + life;	
330	}	
331	return canRestoreHealth;	
332	}	
333	GameCharacter::~GameCharacter() = default;	

jul 21, 20 15:20	Dropltem.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_DROPITEM_H	
2	#define ARGENTUM_TALLER_DROPITEM_H	
3		
4	#include "../common/Identificators.h"	
5		
6	class DropItem {	
7	private:	
8	ItemsInventoryID id;	
9	int amount{};	
10		
11	public:	
12	DropItem(ItemsInventoryID id, int amount);	
13		
14	ItemsInventoryID getId() const ;	
15		
16	int getAmount() const ;	
17		
18	virtual ~DropItem();	
19	};	
20		
21		
22	#endif //ARGENTUM_TALLER_DROPITEM_H	

jul 21, 20 15:20	Dropltem.cpp	Page 1/1
1	#include "Dropltem.h"	
2		
3	DropItem::DropItem(ItemsInventoryID id, int amount) : id(id), amount(amount) {}	
4		
5	ItemsInventoryID DropItem::getId() const {	
6	return id;	
7	}	
8		
9	int DropItem::getAmount() const {	
10	return amount;	
11	}	
12		
13	DropItem::~DropItem() = default ;	

jul 21, 20 15:20	Creature.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_CREATURE_H</code>	
2	<code>#define ARGENTUM_TALLER_CREATURE_H</code>	
3		
4	<code>#include <states/StatePoolCreature.h></code>	
5	<code>#include "GameObject.h"</code>	
6	<code>#include "states/State.h"</code>	
7		
8	<code>class Creature : public GameObject {</code>	
9	<code>private:</code>	
10	<code> uint life{};</code>	
11	<code> CreatureID creatureId;</code>	
12	<code> std::unique_ptr<State> state;</code>	
13	<code> bool itemDrop = false;</code>	
14	<code> uint8_t timeToRemove = 10;</code>	
15	<code> StatePoolCreature statePool;</code>	
16		
17	<code> InputInfo generateRandomInputInfo();</code>	
18	<code>public:</code>	
19	<code> Creature(uint id, CreatureID creatureId, std::shared_ptr<Cell> initialCell,</code>	
20	<code> Point initialPoint);</code>	
21	<code> void update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gameObjec</code>	
22	<code>ts, Board &board) override;</code>	
23	<code> void notify(uint pursuitId);</code>	
24	<code> CreatureID getCreatureId() const;</code>	
25		
26	<code> PlayerInfo getPlayerInfo() override;</code>	
27		
28	<code> bool hasAnInputInfo() override;</code>	
29		
30	<code> InputInfo getNextInputInfo() override;</code>	
31		
32	<code> bool isDead();</code>	
33		
34	<code> bool canDropsItems() override;</code>	
35		
36	<code> bool isItem() override;</code>	
37		
38	<code> std::vector<DropItem> getDrop() override;</code>	
39		
40	<code> float getMaxLife() override;</code>	
41		
42	<code> void receiveDamage(float damage, WeaponID weaponId) override;</code>	
43		
44	<code> CharacterStateID getStateId() override;</code>	
45		
46	<code> virtual NPCInfo interact(GameObject& character, InputInfo input);</code>	
47		
48	<code> virtual ~Creature();</code>	
49		
50	<code> void remove(Board &board) override;</code>	
51		
52	<code> bool isReadyToRemove() override;</code>	
53		
54	<code> bool canBeAttacked(int enemyLevel) const;</code>	
55		
56	<code>};</code>	
57		
58		
59	<code>#endif //ARGENTUM_TALLER_CREATURE_H</code>	
60		

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Creature.cpp	Page 1/3
1	<code>#include <iostream></code>	
2	<code>#include <utility></code>	
3	<code>#include <states/StateTranslator.h></code>	
4	<code>#include "Creature.h"</code>	
5	<code>#include "../common/Random.h"</code>	
6		
7	<code>void Creature::update(std::unordered_map<uint, std::shared_ptr<GameObject>> &gam</code>	
8	<code>eObjects, Board &board) {</code>	
9	<code> statePool.updateState();</code>	
10	<code> statePool.performTask(gameObjects, board);</code>	
11	<code>}</code>	
12	<code>Creature::Creature(uint id, CreatureID creatureId, std::shared_ptr<Cell> initial</code>	
13	<code>Cell, Point initialPoint) :</code>	
14	<code> GameObject(id, initialPoint, std::move(initialCell)), creatureId(creatur</code>	
15	<code>eId), statePool(*this) {</code>	
16	<code> switch (creatureId) {</code>	
17	<code> case CreatureID::Goblin:</code>	
18	<code> this->textureHashId = "h00 h00 b1 s00 w00";</code>	
19	<code> break;</code>	
20	<code> case CreatureID::Skeleton:</code>	
21	<code> this->textureHashId = "h00 h00 b12 s00 w00";</code>	
22	<code> break;</code>	
23	<code> case CreatureID::Spider:</code>	
24	<code> this->textureHashId = "h00 h00 b13 s00 w00";</code>	
25	<code> break;</code>	
26	<code> case CreatureID::Zombie:</code>	
27	<code> this->textureHashId = "h00 h06 b14 s00 w00";</code>	
28	<code> break;</code>	
29	<code> default:</code>	
30	<code> this->textureHashId = nullptr;</code>	
31	<code> break;</code>	
32	<code> }</code>	
33	<code> life = GameStatsConfig::getMaxHealth(creatureId, level);</code>	
34	<code>}</code>	
35	<code>void Creature::notify(uint pursuitId) {</code>	
36	<code> if (statePool.startChasing(pursuitId)) {</code>	
37	<code> InputInfo aInputInfo;</code>	
38	<code> aInputInfo.additional = pursuitId;</code>	
39	<code> statePool.setNextState(StateID::Pursuit, aInputInfo);</code>	
40	<code> }</code>	
41	<code>}</code>	
42		
43	<code>InputInfo Creature::generateRandomInputInfo() {</code>	
44	<code> uint8_t randomId = Random::get(2,5);</code>	
45	<code> InputInfo inputInfo;</code>	
46	<code> inputInfo.input = InputID(randomId);</code>	
47	<code> return inputInfo;</code>	
48	<code>}</code>	
49		
50	<code>CharacterStateID Creature::getStateId() {</code>	
51	<code> return StateTranslator::stateToCharacterState(statePool.getStateId());</code>	
52	<code>}</code>	
53		
54	<code>CreatureID Creature::getCreatureId() const {</code>	
55	<code> return creatureId;</code>	
56	<code>}</code>	
57		
58	<code>void Creature::receiveDamage(float damage, WeaponID weaponId) {</code>	
59	<code> setInteractWeapon(weaponId);</code>	
60	<code> if (GameStatsConfig::canEvade(creatureId)) {</code>	
61	<code> std::cout << "Enemy fail attack" << std::endl;</code>	
62	<code> } else {</code>	
63	<code> float defense = GameStatsConfig::getDefense(creatureId);</code>	

63/217

jul 21, 20 15:20	Creature.cpp	Page 2/3
64	float realDamage = damage - defense;	
65	if (realDamage > 0) {	
66	life = (life - realDamage > 0) ? life - realDamage : 0;	
67	}	
68	std::cout << "Character attack damage: " << damage << std::endl;	
69	std::cout << "Enemy defense: " << defense << std::endl;	
70	std::cout << "Character real damage: " << realDamage << std::endl;	
71	std::cout << "Enemy life is: " << life << std::endl;	
72	if (isDead()) {	
73	itemDrop = true;	
74	std::cout << "Enemy is dead" << realDamage << std::endl;	
75	}	
76	}	
77	}	
78		
79	bool Creature::isDead() {	
80	return life == 0;	
81	}	
82		
83	NPCInfo Creature::interact(GameObject& character, InputInfo input) {	
84	NPCInfo info;	
85	return info;	
86	}	
87		
88	bool Creature::isReadyToRemove() {	
89	if (isDead()) {	
90	timeToRemove--;	
91	}	
92	return timeToRemove == 0;	
93	}	
94		
95	void Creature::remove(Board &board) {	
96	board.removeCreatureFromNest(cell);	
97	cell->free();	
98	}	
99		
100	float Creature::getMaxLife() {	
101	return GameStatsConfig::getMaxHealth(creatureId, level);	
102	}	
103		
104	std::vector<DropItem> Creature::getDrop() {	
105	std::vector<DropItem> dropItems;	
106	int randomDrop = Random::get(0,100);	
107	ItemsInventoryID aItemInventoryId = ItemsInventoryID::Nothing;	
108	float amount = 1;	
109	if ((randomDrop > 80) ^ randomDrop ≤ 96) {	
110	aItemInventoryId = ItemsInventoryID::Gold;	
111	amount = GameStatsConfig::getGoldDrop(creatureId, level);	
112	} else if (randomDrop > 96 ^ randomDrop ≤ 98) {	
113	aItemInventoryId = ItemsInventoryID(Random::get(21, 22));	
114	} else if (randomDrop > 98 ^ randomDrop ≤ 100) {	
115	aItemInventoryId = ItemsInventoryID(Random::get(1, 20));	
116	}	
117	if (aItemInventoryId ≠ ItemsInventoryID::Nothing) {	
118	dropItems.emplace_back(aItemInventoryId, amount);	
119	}	
120	itemDrop = false;	
121	return dropItems;	
122	}	
123		
124	bool Creature::isItem() {	
125	return false;	
126	}	
127		
128	bool Creature::canDropItems() {	
129	return itemDrop;	

jul 21, 20 15:20	Creature.cpp	Page 3/3
130	}	
131		
132	bool Creature::canBeAttacked(int enemyLevel) const {	
133	return true;	
134	}	
135		
136	PlayerInfo Creature::getPlayerInfo() {	
137	return GameObject::getPlayerInfo();	
138	}	
139		
140	bool Creature::hasAnInputInfo() {	
141	return true;	
142	}	
143		
144	InputInfo Creature::getNextInputInfo() {	
145	return generateRandomInputInfo();	
146	}	
147		
148		
149	Creature::~Creature() = default;	

jul 21, 20 15:20	Cell.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_CELL_H</code>	
2	<code>#define ARGENTUM_TALLER_CELL_H</code>	
3		
4	<code>#include <zconf.h></code>	
5		
6	<code>class Cell {</code>	
7	<code>private:</code>	
8	<code> uint x;</code>	
9	<code> uint y;</code>	
10	<code> bool empty;</code>	
11	<code> uint gameIdObjectId;</code>	
12	<code> bool city;</code>	
13	<code> uint nestId;</code>	
14	<code> uint itemId;</code>	
15	<code> bool priest;</code>	
16	<code>public:</code>	
17	<code> Cell(uint x, uint y);</code>	
18		
19	<code> void occupied(uint id);</code>	
20		
21	<code> void free();</code>	
22		
23	<code> std::tuple<int, int> getCoord();</code>	
24		
25	<code> friend bool operator==(const Cell& c1, const Cell& c2);</code>	
26		
27	<code> friend bool operator!=(const Cell& c1, const Cell& c2);</code>	
28		
29	<code> void removeItem();</code>	
30		
31	<code> bool hasItem() const;</code>	
32		
33	<code> void setItemId(uint itemId);</code>	
34		
35	<code> uint getItemId() const;</code>	
36		
37	<code> bool isEmpty() const;</code>	
38		
39	<code> bool isPriest() const;</code>	
40		
41	<code> void addPriest();</code>	
42		
43	<code> void setEmpty(bool empty);</code>	
44		
45	<code> uint gameIdObjectId() const;</code>	
46		
47	<code> bool isCity() const;</code>	
48		
49	<code> void setCity(bool city);</code>	
50		
51	<code> uint getNestId() const;</code>	
52		
53	<code> void setNestId(uint nestId);</code>	
54		
55	<code> uint getX() const;</code>	
56		
57	<code> uint getY() const;</code>	
58		
59	<code> virtual ~Cell();</code>	
60	<code>};</code>	
61		
62		
63		
64	<code>#endif //ARGENTUM_TALLER_CELL_H</code>	

jul 21, 20 15:20	Cell.cpp	Page 1/2
1	<code>#include <tuple></code>	
2	<code>#include "Cell.h"</code>	
3		
4	<code>Cell::Cell(uint x, uint y) : x(x), y(y), empty(true), gameIdObjectId(0), city(false), nestId(0), itemId(0), priest(false) {}</code>	
5		
6	<code>bool Cell::isEmpty() const {</code>	
7	<code> return empty;</code>	
8	<code>}</code>	
9		
10	<code>void Cell::setEmpty(bool empty) {</code>	
11	<code> Cell::empty = empty;</code>	
12	<code>}</code>	
13		
14	<code>uint Cell::getGameObjectId() const {</code>	
15	<code> return gameIdObjectId;</code>	
16	<code>}</code>	
17		
18	<code>bool Cell::isCity() const {</code>	
19	<code> return city;</code>	
20	<code>}</code>	
21		
22	<code>void Cell::setCity(bool city) {</code>	
23	<code> Cell::city = city;</code>	
24	<code>}</code>	
25		
26	<code>uint Cell::getNestId() const {</code>	
27	<code> return nestId;</code>	
28	<code>}</code>	
29		
30	<code>void Cell::setNestId(uint nestId) {</code>	
31	<code> Cell::nestId = nestId;</code>	
32	<code>}</code>	
33		
34	<code>uint Cell::getX() const {</code>	
35	<code> return x;</code>	
36	<code>}</code>	
37		
38	<code>uint Cell::getY() const {</code>	
39	<code> return y;</code>	
40	<code>}</code>	
41		
42	<code>void Cell::occupied(uint id) {</code>	
43	<code> gameIdObjectId = id;</code>	
44	<code> empty = false;</code>	
45	<code>}</code>	
46		
47	<code>void Cell::free() {</code>	
48	<code> gameIdObjectId = 0;</code>	
49	<code> empty = true;</code>	
50	<code>}</code>	
51		
52	<code>std::tuple<int, int> Cell::getCoord() {</code>	
53	<code> return {x, y};</code>	
54	<code>}</code>	
55		
56	<code>bool operator==(const Cell& c1, const Cell& c2) {</code>	
57	<code> return c1.x == c2.x & c1.y == c2.y;</code>	
58	<code>}</code>	
59		
60	<code>bool operator!=(const Cell& c1, const Cell& c2) {</code>	
61	<code> return !(c1==c2);</code>	
62	<code>}</code>	
63		
64	<code>void Cell::removeItem() {</code>	
65	<code> itemId = 0;</code>	

jul 21, 20 15:20	Cell.cpp	Page 2/2
66	}	
67		
68	uint Cell::getItemId() const {	
69	return itemId;	
70	}	
71		
72	bool Cell::hasItem() const {	
73	return itemId != 0;	
74	}	
75		
76	void Cell::setItemId(uint itemId) {	
77	Cell::itemId = itemId;	
78	}	
79		
80	bool Cell::isPriest() const {	
81	return priest;	
82	}	
83		
84	void Cell::addPriest() {	
85	priest = true;	
86	}	
87		
88		
89	Cell::~Cell() = default;	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Board.h	Page 1/2
1	#ifndef ARGENTUM_TALLER_BOARD_H	
2	#define ARGENTUM_TALLER_BOARD_H	
3		
4		
5	#include <vector>	
6	#include <memory>	
7	#include <queue>	
8	#include "../common/Point.h"	
9	#include "../common/StaticObject.h"	
10	#include "../common/ObjectLayer.h"	
11	#include "../common/PlayerInfo.h"	
12	#include "GameStats.h"	
13	#include "GameStatsConfig.h"	
14	#include "Nest.h"	
15	#include "NestContainer.h"	
16	#include "Cell.h"	
17	#include "../common/TiledMap.h"	
18		
19	class Board {	
20	private:	
21	uint width{};	
22	uint height{};	
23	uint cols{}, rows{};	
24	Point initialPoint;	
25	NestContainer nestContainer;	
26	std::vector<std::vector<std::shared_ptr<Cell>>> cells;	
27		
28	void addCity(StaticObject &city);	
29	std::tuple<int, int> convertPoint(const Point &point);	
30	void addCollisionObject(StaticObject &object);	
31	std::pair<int, int> getCorrectPosition(const std::shared_ptr<Cell>& aCell, Direction aDirection);	
32	public:	
33	Board();	
34	Board(TiledMap &map, uint8_t nestCreaturesLimit);	
35		
36	virtual ~Board();	
37		
38	std::shared_ptr<Cell> getCellFromPoint(const Point& aPoint);	
39		
40	Point getPointFromCell(const std::shared_ptr<Cell>& aCell);	
41		
42	std::shared_ptr<Cell> getCell(uint x, uint y);	
43		
44	std::shared_ptr<Cell> getInitialCell();	
45		
46	std::vector<std::pair<uint8_t, std::shared_ptr<Cell>>> getAdjacents(std::tuple<uint, uint>, uint8_t distance);	
47		
48	uint8_t getDistance(const std::shared_ptr<Cell>& firstCell, const std::shared_ptr<Cell>& secondCell);	
49		
50	std::shared_ptr<Cell> getInitialCellInNest(Nest& nest);	
51		
52	std::vector<uint> getCreaturesInNest(uint nestId);	
53		
54	Nest& getAvailableNest();	
55		
56	std::vector<std::shared_ptr<Cell>> setCellsInNest(const std::shared_ptr<Cell>& aNestCell, uint nestId);	
57		
58	bool characterCanMove(const std::shared_ptr<Cell>& aCell, Direction aDirection);	
59		
60		
61	bool creatureCanMove(const std::shared_ptr<Cell>& aCell, Direction aDirection);	

66/217

jul 21, 20 15:20	Board.h	Page 2/2
	<pre> n); 62 63 std::shared_ptr<Cell> getNextCell(const std::shared_ptr<Cell>& aCell, Direct ion aDirection); 64 65 std::shared_ptr<Cell> getBestCell(const std::shared_ptr<Cell>& actualCell, c onst std::shared_ptr<Cell>& DestinationCell, bool overstepNest); 66 67 Direction getDirection(const std::shared_ptr<Cell>& actualCell, const std::s hared_ptr<Cell>& DestinationCell); 68 69 void removeCreatureFromNest(const std::shared_ptr<Cell>& aCell); 70 71 int getAmountCreatures(); 72 73 std::shared_ptr<Cell> getNextEmptyCell(const std::shared_ptr<Cell>& aCell); 74 75 std::shared_ptr<Cell> getCloserPriest(const std::shared_ptr<Cell>& aCell); 76 77 }; 78 79 #endif // ARGENTUM_TALLER_BOARD_H 80 </pre>	

jul 21, 20 15:20	Board.cpp	Page 1/5
	<pre> 1 #include "Board.h" 2 #include "Node.h" 3 #include "NodeContainer.h" 4 #include <memory> 5 #include <cstdlib> 6 7 Board::~Board() = default; 8 9 Board::Board() = default; 10 11 Board::Board(TiledMap &map, uint8_t nestCreaturesLimit) : initialPoint(1125, 550) { 12 cols = map.getWidth(); 13 rows = map.getHeight(); 14 height = map.getHeight() * map.getTileHeight(); 15 width = map.getWidth() * map.getTileWidth(); 16 uint nestIdCounter = 1; 17 for (size_t i = 0; i < rows; ++i) { 18 std::vector<std::shared_ptr<Cell>> cellsRows; 19 for (size_t j = 0; j < cols; ++j) { 20 cellsRows.push_back(std::make_shared<Cell>(i, j)); 21 } 22 cells.push_back(cellsRows); 23 } 24 for (auto& anObjectLayer : map.getObjectLayers()) { 25 if (anObjectLayer.getName() == "cities") { 26 for (StaticObject& aCity : anObjectLayer.getObjects()) { 27 addCity(aCity); 28 } 29 } else if (anObjectLayer.getName() == "collisionObjects") { 30 for (StaticObject &aCollisionObject : anObjectLayer.getObjects()) { 31 addCollisionObject(aCollisionObject); 32 } 33 } else if (anObjectLayer.getName() == "nestPoints") { 34 std::vector<Nest> nests; 35 std::shared_ptr<Cell> aCell; 36 uint nestId; 37 for (StaticObject& aNest : anObjectLayer.getObjects()) { 38 nestId = nestIdCounter++; 39 aCell = getCellFromPoint(aNest.getTopLeft()); 40 nests.emplace_back(nestCreaturesLimit, nestId, setCellsInNest(aC ell, nestId)); 41 } 42 nestContainer = NestContainer(nests); 43 } 44 } 45 } 46 47 Nest& Board::getAvailableNest() { 48 return nestContainer.getNextNestAvailable(); 49 } 50 51 std::vector<uint> Board::getCreaturesInNest(uint nestId) { 52 Nest& nest = nestContainer.getNest(nestId); 53 return nest.getCreatures(); 54 } 55 56 void Board::addCity(StaticObject &city) { 57 std::tuple<int, int> topLeft = convertPoint(city.getTopLeft()); 58 std::tuple<int, int> bottomRight = convertPoint(city.getBottomRight()); 59 for (int i = std::get<0>(topLeft); i < std::get<0>(bottomRight); ++i) { 60 for (int j = std::get<1>(topLeft); j < std::get<1>(bottomRight); ++j) { 61 cells[i][j] -> setCity(true); 62 } 63 } 64 } </pre>	

jul 21, 20 15:20	Board.cpp	Page 2/5
65	<code>std::tuple<int, int> Board::convertPoint(const Point &point) {</code>	
66	<code>int x = point.x / (width / cols);</code>	
67	<code>int y = point.y / (height / rows);</code>	
68	<code>return std::make_tuple(x, y);</code>	
69	<code>}</code>	
70		
71		
72	<code>void Board::addCollisionObject(StaticObject &collisionObject) {</code>	
73	<code>std::tuple<int, int> topLeft = convertPoint(collisionObject.getTopLeft());</code>	
74	<code>std::tuple<int, int> bottomRight = convertPoint(collisionObject.getBottomRight());</code>	
75	<code>int diffX = std::get<0>(bottomRight) - std::get<0>(topLeft);</code>	
76	<code>int diffY = std::get<1>(bottomRight) - std::get<1>(topLeft);</code>	
77	<code>for (int i = std::get<0>(topLeft); diffX > 0 ? i < std::get<0>(bottomRight) :</code>	
78	<code>i ≤ std::get<0>(bottomRight); ++i) {</code>	
79	<code>for (int j = std::get<1>(topLeft); diffY > 0 ? j < std::get<1>(bottomRight) :</code>	
80	<code>j ≤ std::get<1>(bottomRight); ++j) {</code>	
81	<code>cells[i][j]→setEmpty(false);</code>	
82	<code>}</code>	
83	<code>}</code>	
84	<code>std::shared_ptr<Cell> Board::getCellFromPoint(const Point &aPoint) {</code>	
85	<code>std::tuple<uint, uint> position = convertPoint(aPoint);</code>	
86	<code>return cells[std::get<0>(position)][std::get<1>(position)];</code>	
87	<code>}</code>	
88		
89	<code>std::shared_ptr<Cell> Board::getCell(uint x, uint y) {</code>	
90	<code>return cells[x][y];</code>	
91	<code>}</code>	
92		
93	<code>std::shared_ptr<Cell> Board::getInitialCell() {</code>	
94	<code>std::shared_ptr<Cell> cell = getCellFromPoint(initialPoint);</code>	
95	<code>if (!cell→isEmpty()) {</code>	
96	<code>for (auto &adjacent : getAdjacents(convertPoint(initialPoint), 4)) {</code>	
97	<code>if (adjacent.second→isEmpty()) {</code>	
98	<code>cell = adjacent.second;</code>	
99	<code>break;</code>	
100	<code>}</code>	
101	<code>}</code>	
102	<code>return cell;</code>	
103	<code>}</code>	
104		
105		
106	<code>std::vector<std::pair<uint8_t, std::shared_ptr<Cell>>> Board::getAdjacents(std::tuple<uint, uint> position, uint8_t distance) {</code>	
107	<code>std::vector<std::pair<uint8_t, std::shared_ptr<Cell>>> adjacents;</code>	
108	<code>uint x = std::get<0>(position);</code>	
109	<code>uint y = std::get<1>(position);</code>	
110	<code>std::shared_ptr<Cell> originCell = getCell(x, y);</code>	
111	<code>uint leftLimitX = x - distance ≥ 0 ? x - distance : 0;</code>	
112	<code>uint rightLimitX = x + distance < cols ? x + distance : cols - 1;</code>	
113	<code>uint leftLimitY = y - distance ≥ 0 ? y - distance : 0;</code>	
114	<code>uint rightLimitY = y + distance < rows ? y + distance : rows - 1;</code>	
115	<code>std::shared_ptr<Cell> aCell;</code>	
116	<code>for (size_t i = leftLimitX; i ≤ rightLimitX; ++i) {</code>	
117	<code>for (size_t j = leftLimitY; j ≤ rightLimitY; ++j) {</code>	
118	<code>if (i == x & j == y) {</code>	
119	<code>continue;</code>	
120	<code>}</code>	
121	<code>aCell = getCell(i, j);</code>	
122	<code>uint8_t aDistance = getDistance(originCell, aCell);</code>	
123	<code>if (aDistance ≤ distance) {</code>	
124	<code>adjacents.emplace_back(aDistance, aCell);</code>	
125	<code>}</code>	
126	<code>}</code>	

jul 21, 20 15:20	Board.cpp	Page 3/5
127	<code>}</code>	
128	<code>std::sort_heap(adjacents.begin(), adjacents.end());</code>	
129	<code>return adjacents;</code>	
130	<code>}</code>	
131		
132	<code>uint8_t Board::getDistance(const std::shared_ptr<Cell> &firstCell, const std::shared_ptr<Cell> &secondCell) {</code>	
133	<code>return std::abs(int(firstCell→getX() - secondCell→getX())) + std::abs(int(firstCell→getY() - secondCell→getY()));</code>	
134	<code>}</code>	
135		
136	<code>Point Board::getPointFromCell(const std::shared_ptr<Cell> &aCell) {</code>	
137	<code>return Point(aCell→getX() * (width/cols), aCell→getY() * (height/rows));</code>	
138	<code>}</code>	
139		
140	<code>std::shared_ptr<Cell> Board::getInitialCellInNest(Nest &nest) {</code>	
141	<code>return nest.getFreeCell();</code>	
142	<code>}</code>	
143		
144	<code>std::vector<std::shared_ptr<Cell>> Board::setCellsInNest(const std::shared_ptr<Cell> &aNestCell, uint nestId) {</code>	
145	<code>std::vector<std::shared_ptr<Cell>> cellInsideNest;</code>	
146	<code>uint distance = 8;</code>	
147	<code>uint leftLimitX = int(aNestCell→getX() - distance) ≥ 0 ? aNestCell→getX() - distance : 0;</code>	
148	<code>uint rightLimitX = aNestCell→getX() + distance < cols ? aNestCell→getX() + distance : cols - 1;</code>	
149	<code>uint leftLimitY = int(aNestCell→getY() - distance) ≥ 0 ? aNestCell→getY() - distance : 0;</code>	
150	<code>uint rightLimitY = aNestCell→getY() + distance < rows ? aNestCell→getY() + distance : rows - 1;</code>	
151	<code>std::shared_ptr<Cell> aCell;</code>	
152	<code>for (size_t i = leftLimitY; i ≤ rightLimitY; ++i) {</code>	
153	<code>for (size_t j = leftLimitX; j ≤ rightLimitX; ++j) {</code>	
154	<code>aCell = getCell(i, j);</code>	
155	<code>if (!aCell→isCity()) {</code>	
156	<code>aCell→setNestId(nestId);</code>	
157	<code>cellInsideNest.push_back(aCell);</code>	
158	<code>}</code>	
159	<code>}</code>	
160	<code>return cellInsideNest;</code>	
161	<code>}</code>	
162		
163		
164	<code>bool Board::characterCanMove(const std::shared_ptr<Cell> &aCell, Direction aDirection) {</code>	
165	<code>bool canMove = false;</code>	
166	<code>std::pair<uint, uint> aPosition = getCorrectPosition(aCell, aDirection);</code>	
167	<code>if (aPosition.first ≥ 0 ∧ aPosition.first < cols ∧ aPosition.second ≥ 0 ∧ aPosition.second < rows) {</code>	
168	<code>canMove = getCell(aPosition.first, aPosition.second)→isEmpty();</code>	
169	<code>}</code>	
170	<code>return canMove;</code>	
171	<code>}</code>	
172		
173	<code>std::shared_ptr<Cell> Board::getNextCell(const std::shared_ptr<Cell> &aCell, Direction aDirection) {</code>	
174	<code>std::pair<uint, uint> aPosition = getCorrectPosition(aCell, aDirection);</code>	
175	<code>return getCell(aPosition.first, aPosition.second);</code>	
176	<code>}</code>	
177		
178	<code>bool Board::creatureCanMove(const std::shared_ptr<Cell> &aCell, Direction aDirection) {</code>	
179	<code>bool canMove = false;</code>	
180	<code>std::pair<uint, uint> aPosition = getCorrectPosition(aCell, aDirection);</code>	
181	<code>if (aPosition.first ≥ 0 ∧ aPosition.first < cols ∧ aPosition.second ≥ 0 ∧ aPosition.second < rows) {</code>	

Page 4/5	Board.cpp	Page 4/5
182	osition.second < rows) {	
183	std::shared_ptr<Cell> nestCell = getCell(aPosition.first, aPosition.seco	
184	nd);	
185	canMove = nestCell->isEmpty() ^ nestCell->getNestId() == aCell->getNestId(
186);	
187	return canMove;	
188	std::pair<int, int> Board::getCorrectPosition(const std::shared_ptr<Cell>& aCell	
189	, Direction aDirection) {	
190	int x = aCell->getX();	
191	int y = aCell->getY();	
192	switch(aDirection) {	
193	case Direction::up:	
194	y --;	
195	break;	
196	case Direction::down:	
197	y ++;	
198	break;	
199	case Direction::left:	
200	x --;	
201	break;	
202	case Direction::right:	
203	x ++;	
204	break;	
205	return {x, y};	
206	}	
207		
208	std::shared_ptr<Cell> Board::getBestCell(const std::shared_ptr<Cell>& actualCell	
209	, const std::shared_ptr<Cell>& destinationCell, bool overstepNest) {	
210	NodeContainer close;	
211	NodeContainer open;	
212	Node firstNode = Node(0, 0, getDistance(actualCell, destinationCell), actual	
213	Cell);	
214	Node destinationNode = Node(0, 0, 0, destinationCell);	
215	Node& pos = firstNode;	
216	open.insert(firstNode);	
217	while (!open.has(destinationNode)) {	
218	pos = open.getBestNode();	
219	for (auto &adjacent : getAdjacents(pos.getCell()->getCoord(), 1)) {	
220	Node aNode = Node(pos.getId(), pos.getG() + 1, getDistance(adjacent.	
221	second, destinationCell, adjacent.second);	
222	if (overstepNest ^ adjacent.second->getNestId() == actualCell->getNes	
223	tId()) {	
224	if (adjacent.second == destinationCell) {	
225	open.insert(aNode);	
226	else if (adjacent.second->isEmpty() ^ !close.has(aNode)) {	
227	if (open.has(aNode)) {	
228	open.modifyNode(aNode.getId(), pos);	
229	else {	
230	open.insert(aNode);	
231	}	
232	}	
233	close.insert(pos);	
234	Node &selectedNode = close.get(open.get(destinationNode.getId()).getParent()	
235);	
236	while (close.get(selectedNode.getParent()).getParent() != 0) {	
237	selectedNode = close.get(selectedNode.getParent());	
238	return selectedNode.getCell();	

Page 5/5	Board.cpp	Page 5/5
239	Direction Board::getDirection(const std::shared_ptr<Cell>& actualCell, const std	
240	::shared_ptr<Cell>& destinationCell) {	
241	Direction aDirection;	
242	if (actualCell->getX() > destinationCell->getX()) {	
243	aDirection = Direction::left;	
244	}	
245	if (actualCell->getX() < destinationCell->getX()) {	
246	aDirection = Direction::right;	
247	}	
248	if (actualCell->getY() > destinationCell->getY()) {	
249	aDirection = Direction::up;	
250	}	
251	if (actualCell->getY() < destinationCell->getY()) {	
252	aDirection = Direction::down;	
253	}	
254	return aDirection;	
255	}	
256		
257	void Board::removeCreatureFromNest(const std::shared_ptr<Cell>& aCell) {	
258	Nest &nest = nestContainer.getNest(aCell->getNestId());	
259	nest.removeCreature(aCell->getGameObjectId());	
260	}	
261		
262	int Board::getAmountCreatures() {	
263	return nestContainer.getAmountCreatures();	
264	}	
265		
266	std::shared_ptr<Cell> Board::getNextEmptyCell(const std::shared_ptr<Cell> &aCell	
267) {	
268	std::shared_ptr<Cell> cell = aCell;	
269	uint distance = 0;	
270	while (!cell->isEmpty() ^ cell->hasItem()) {	
271	distance++;	
272	for (auto &adjacent : getAdjacents(aCell->getCoord(), distance)) {	
273	if (adjacent.second->isEmpty() ^ !adjacent.second->hasItem()) {	
274	cell = adjacent.second;	
275	break;	
276	}	
277	}	
278	return cell;	
279	}	
280		
281	std::shared_ptr<Cell> Board::getCloserPriest(const std::shared_ptr<Cell>& aCell)	
282	{	
283	std::shared_ptr<Cell> chosenCell;	
284	std::shared_ptr<Cell> aCellPriest;	
285	uint8_t minDistance = 0;	
286	for (size_t i = 0; i < rows; ++i) {	
287	for (size_t j = 0; j < cols; ++j) {	
288	aCellPriest = getCell(i, j);	
289	if (aCellPriest->isPriest()) {	
290	if (minDistance == 0 ^ minDistance > getDistance(aCellPriest, aCe	
291	ll)) {	
292	minDistance = getDistance(aCellPriest, aCell);	
293	chosenCell = aCellPriest;	
294	}	
295	}	
296	return chosenCell;	
297	}	

jul 21, 20 15:20	Banker.h	Page 1/1
1	<code>#ifndef BANKER_H</code>	
2	<code>#define BANKER_H</code>	
3		
4	<code>#include <unordered_map></code>	
5	<code>#include <vector></code>	
6	<code>#include "../common/Identificators.h"</code>	
7	<code>#include "Profession.h"</code>	
8		
9	<code>//Se optÃ por realizar esta clase como un Singleton debido a que</code>	
10	<code>//los diversos jugadores(Threads) tendrÃn solo acceso a sus items</code>	
11	<code>//que estÃen depositados dentro del Banco, sin poder acceder a ningun</code>	
12	<code>//otra "cuenta" que pertenezca al usuario.</code>	
13	<code>class Banker: public Profession {</code>	
14	<code>private:</code>	
15	<code>Banker();</code>	
16	<code>std::unordered_map<uint, std::vector<ItemsInventoryID>> accountsItems;</code>	
17	<code>std::unordered_map<uint, uint> accountsGold;</code>	
18	<code>static Banker* banker;</code>	
19		
20	<code>void createNewAccount(uint accountHolder);</code>	
21		
22	<code>std::vector<ItemsInventoryID> getMyItems(uint accountHolder);</code>	
23		
24	<code>uint checkBalance(uint accountHolder);</code>	
25		
26	<code>ItemsInventoryID retireItem(uint accountHolder, uint item);</code>	
27		
28	<code>void depositItem(uint accountHolder, ItemsInventoryID idItem);</code>	
29		
30	<code>void depositGold(uint accountHolder, uint amountGold);</code>	
31		
32	<code>uint retireGold(uint accountHolder, uint amountGold);</code>	
33		
34	<code>public:</code>	
35	<code>static Banker* getInstance();</code>	
36		
37	<code>virtual NPCInfo getInfo(uint id);</code>	
38		
39	<code>void processInput(GameCharacter &character, InputInfo inputInfo) override;</code>	
40		
41	<code>virtual ~Banker();</code>	
42	<code>};</code>	
43		
44	<code>#endif</code>	

jul 21, 20 15:20	Banker.cpp	Page 1/3
1	<code>#include "Banker.h"</code>	
2		
3	<code>Banker* Banker::banker = nullptr;</code>	
4		
5	<code>Banker::Banker() :accountsItems(), accountsGold() {</code>	
6	<code> this->actions.push_back(ActionsProfessionID::DepositGold);</code>	
7	<code> this->actions.push_back(ActionsProfessionID::DepositItem);</code>	
8	<code> this->actions.push_back(ActionsProfessionID::RetireItem);</code>	
9	<code> this->actions.push_back(ActionsProfessionID::RetireGold);</code>	
10	<code>}</code>	
11		
12	<code>Banker* Banker::getInstance() {</code>	
13	<code> if (banker == nullptr)</code>	
14	<code> banker = new Banker();</code>	
15	<code> return banker;</code>	
16	<code>}</code>	
17		
18	<code>std::vector<ItemsInventoryID> Banker::getMyItems(uint accountHolder) {</code>	
19	<code> auto iter = accountsItems.find(accountHolder);</code>	
20	<code> if (iter == accountsItems.end()) {</code>	
21	<code> createNewAccount(accountHolder);</code>	
22	<code> iter = accountsItems.find(accountHolder);</code>	
23	<code> }</code>	
24	<code> return (*iter).second;</code>	
25	<code>}</code>	
26		
27	<code>uint Banker::checkBalance(uint accountHolder) {</code>	
28	<code> auto iter = accountsGold.find(accountHolder);</code>	
29	<code> if (iter == accountsGold.end()) {</code>	
30	<code> createNewAccount(accountHolder);</code>	
31	<code> return 0;</code>	
32	<code> }</code>	
33	<code> return accountsGold[accountHolder];</code>	
34	<code>}</code>	
35		
36	<code>void Banker::createNewAccount(uint accountHolder) {</code>	
37	<code> uint gold = 0;</code>	
38	<code> accountsGold.insert({accountHolder, gold});</code>	
39	<code> std::vector<ItemsInventoryID> emptyAccount;</code>	
40	<code> emptyAccount.push_back(ItemsInventoryID::HealthPotion);</code>	
41	<code> accountsItems.insert({accountHolder, emptyAccount});</code>	
42	<code>}</code>	
43		
44	<code>ItemsInventoryID Banker::retireItem(uint accountHolder, uint item) {</code>	
45	<code> auto iter = accountsItems.find(accountHolder);</code>	
46	<code> if (iter == accountsItems.end()) {</code>	
47	<code> createNewAccount(accountHolder);</code>	
48	<code> return ItemsInventoryID::Nothing;</code>	
49	<code> }</code>	
50	<code> if (accountsItems[accountHolder].size() == 0)</code>	
51	<code> return ItemsInventoryID::Nothing;</code>	
52	<code> ItemsInventoryID itemToRetire = (*iter).second.at(item);</code>	
53	<code> (*iter).second.at(item) = ItemsInventoryID::Nothing;</code>	
54	<code> auto it = (*iter).second.begin();</code>	
55	<code> while (it != (*iter).second.end()) {</code>	
56	<code> if (*it == ItemsInventoryID::Nothing) {</code>	
57	<code> it = (*iter).second.erase(it);</code>	
58	<code> } else {</code>	
59	<code> it++;</code>	
60	<code> }</code>	
61	<code> }</code>	
62	<code> return itemToRetire;</code>	
63	<code>}</code>	
64		
65	<code>NPCInfo Banker::getInfo(uint id) {</code>	
66	<code> NPCInfo info;</code>	

jul 21, 20 15:20	Banker.cpp	Page 2/3
67	info.type = 3;	
68	info.actions = actions;	
69	info.gold = checkBalance(id);	
70	info.itemsInBank = getMyItems(id);	
71	return info;	
72	}	
73		
74		
75	void Banker::depositItem(uint accountHolder, ItemsInventoryID idItem) {	
76	auto iter = accountsItems.find(accountHolder);	
77	if (iter == accountsItems.end()) {	
78	createNewAccount(accountHolder);	
79	iter = accountsItems.find(accountHolder);	
80	}	
81	(*iter).second.push_back(idItem);	
82	}	
83		
84	void Banker::depositGold(uint accountHolder, uint amountGold) {	
85	auto iter = accountsGold.find(accountHolder);	
86	if (iter == accountsGold.end()) {	
87	createNewAccount(accountHolder);	
88	accountsGold[accountHolder] += amountGold;	
89	}	
90		
91	uint Banker::retireGold(uint accountHolder, uint amountGold) {	
92	auto iter = accountsGold.find(accountHolder);	
93	if (iter == accountsGold.end()) {	
94	createNewAccount(accountHolder);	
95	return 0;	
96	}	
97	if (accountsGold[accountHolder] < amountGold) {	
98	amountGold = accountsGold[accountHolder];	
99	accountsGold[accountHolder] = 0;	
100	} else {	
101	accountsGold[accountHolder] -= amountGold;	
102	}	
103	return amountGold;	
104	}	
105		
106	void Banker::processInput(GameCharacter &character, InputInfo inputInfo) {	
107	bool addedToInventory;	
108	ItemsInventoryID aItem;	
109	int gold;	
110	switch (inputInfo.input) {	
111	case InputID::depositItem:	
112	character.removeItemFromInventory(ItemsInventoryID(inputInfo.adition	
113	al));	
114	depositItem(character.getId(), ItemsInventoryID(inputInfo.additional))	
115	break;	
116	case InputID::retireItem:	
117	aItem = retireItem(character.getId(), inputInfo.additional);	
118	if (aItem != ItemsInventoryID::Nothing) {	
119	addedToInventory = character.addItemToInventory(aItem);	
120	if (!addedToInventory)	
121	depositItem(character.getId(), aItem);	
122	break;	
123	case InputID::depositGold:	
124	gold = character.getGoldAmount();	
125	if (inputInfo.additional > gold)	
126	inputInfo.additional = gold;	
127	depositGold(character.getId(), inputInfo.additional);	
128	character.setGoldAmount(character.getGoldAmount() - inputInfo.aditio	
129	nal);	
130	break;	

jul 21, 20 15:20	Banker.cpp	Page 3/3
130	case InputID::retireGold:	
131	gold = retireGold(character.getId(), inputInfo.additional);	
132	character.gainGold(gold);	
133	break;	
134	default:	
135	break;	
136	}	
137	}	
138	}	
139	Banker::~Banker() {}	
140		

jul 21, 20 15:20	TileSet.h	Page 1/1
1	<code>#ifndef ARGENTUM_TILESET_H</code>	
2	<code>#define ARGENTUM_TILESET_H</code>	
3		
4	<code>#include <string></code>	
5	<code>#include <rapidjson/document.h></code>	
6		
7	<code>class TileSet {</code>	
8	<code>private:</code>	
9	<code> uint16_t firstgid;</code>	
10	<code> std::string image;</code>	
11	<code> uint8_t id;</code>	
12		
13	<code>public:</code>	
14	<code> explicit TileSet(rapidjson::Value&, uint8_t id);</code>	
15	<code> TileSet(uint16_t firstGid, uint8_t id);</code>	
16		
17	<code> virtual ~TileSet();</code>	
18		
19	<code> int getFirstgid() const;</code>	
20		
21	<code> const std::string &getImage() const;</code>	
22		
23	<code> uint8_t getId() const;</code>	
24	<code>};</code>	
25		
26		
27	<code>#endif //ARGENTUM_TILESET_H</code>	

jul 21, 20 15:20	TileSet.cpp	Page 1/1
1	<code>#include "TileSet.h"</code>	
2	<code>#include "Identificators.h"</code>	
3		
4	<code>TileSet::TileSet(rapidjson::Value & tilesetDoc, uint8_t id): id(id) {</code>	
5	<code> firstgid = tilesetDoc["firstgid"].GetInt();</code>	
6	<code> image = tilesetDoc["image"].GetString();</code>	
7	<code>}</code>	
8		
9	<code>int TileSet::getFirstgid() const {</code>	
10	<code> return firstgid;</code>	
11	<code>}</code>	
12		
13	<code>const std::string &TileSet::getImage() const {</code>	
14	<code> return image;</code>	
15	<code>}</code>	
16		
17	<code>uint8_t TileSet::getId() const {</code>	
18	<code> return id;</code>	
19	<code>}</code>	
20		
21	<code>TileSet::TileSet(uint16_t firstGid, uint8_t id): firstgid(firstGid), id(id) {</code>	
22	<code> std::string path(ROOT_DIR);</code>	
23	<code> image = path + "/assets/img/map/" + std::to_string(id) + ".png";</code>	
24	<code>}</code>	
25		
26	<code>TileSet::~TileSet() = default;</code>	

jul 21, 20 15:20	TileLayer.h	Page 1/1
1	<code>#ifndef ARGENTUM_LAYER_H</code>	
2	<code>#define ARGENTUM_LAYER_H</code>	
3		
4	<code>#include <vector></code>	
5	<code>#include <string></code>	
6	<code>#include "rapidjson/document.h"</code>	
7		
8	<code>class TileLayer {</code>	
9	<code>private:</code>	
10	<code>std::vector<uint16_t> data;</code>	
11	<code>bool isGround;</code>	
12		
13	<code>public:</code>	
14	<code>TileLayer();</code>	
15	<code>explicit TileLayer(rapidjson::Value&);</code>	
16	<code>TileLayer(std::vector<uint16_t> data, bool isGround);</code>	
17		
18	<code>virtual ~TileLayer();</code>	
19		
20	<code>const std::vector<uint16_t> &getData() const;</code>	
21		
22	<code>bool isGroundLayer() const;</code>	
23	<code>};</code>	
24		
25		
26	<code>#endif //ARGENTUM_LAYER_H</code>	

jul 21, 20 15:20	TileLayer.cpp	Page 1/1
1	<code>#include "TileLayer.h"</code>	
2		
3	<code>#include <utility></code>	
4		
5	<code>TileLayer::TileLayer() = default;</code>	
6		
7	<code>TileLayer::~TileLayer() = default;</code>	
8		
9	<code>TileLayer::TileLayer(rapidjson::Value & value) {</code>	
10	<code>if (value.HasMember("data")) {</code>	
11	<code>for (auto& a : value["data"].GetArray()) {</code>	
12	<code>data.push_back(a.GetInt64());</code>	
13	<code>}</code>	
14	<code>}</code>	
15	<code>std::string layerName = value["name"].GetString();</code>	
16	<code>isGround = "groundLayer" == layerName;</code>	
17	<code>}</code>	
18		
19	<code>const std::vector<uint16_t> &TileLayer::getData() const {</code>	
20	<code>return data;</code>	
21	<code>}</code>	
22		
23	<code>bool TileLayer::isGroundLayer() const {</code>	
24	<code>return isGround;</code>	
25	<code>}</code>	
26		
27	<code>TileLayer::TileLayer(std::vector<uint16_t> data, bool isGround): data(std::move(data)), isGround(isGround) {}</code>	

jul 21, 20 15:20	TiledMap.h	Page 1/1
1	<code>#ifndef ARGENTUM_TILEDMAP_H</code>	
2	<code>#define ARGENTUM_TILEDMAP_H</code>	
3		
4		
5	<code>#include <vector></code>	
6	<code>#include <string></code>	
7	<code>#include "TileLayer.h"</code>	
8	<code>#include "TileSet.h"</code>	
9	<code>#include "ObjectLayer.h"</code>	
10		
11	<code>class TiledMap {</code>	
12	<code>private:</code>	
13	<code> uint16_t width{};</code>	
14	<code> uint16_t height{};</code>	
15	<code> uint8_t tileWidth{};</code>	
16	<code> uint8_t tileHeight{};</code>	
17	<code> std::vector<TileLayer> tileLayers;</code>	
18	<code> std::vector<ObjectLayer> objectLayers;</code>	
19	<code> std::vector<TileSet> tilesets;</code>	
20	<code>public:</code>	
21	<code> TiledMap();</code>	
22	<code> explicit TiledMap(rapidjson::Document & json);</code>	
23		
24	<code> TiledMap(uint16_t width, uint16_t height, uint8_t tileWidth, uint8_t tileHeight,</code>	
25	<code> std::vector<TileLayer> tileLayers, std::vector<TileSet> tilesets);</code>	
26		
27	<code> ~TiledMap();</code>	
28		
29	<code> TiledMap(TiledMap& other) noexcept ;</code>	
30	<code> TiledMap& operator=(TiledMap& other) noexcept ;</code>	
31		
32	<code> std::vector<ObjectLayer> getObjectLayers();</code>	
33		
34	<code> const std::vector<TileLayer> &getTileLayers() const;</code>	
35		
36	<code> const std::vector<TileSet> &getTilesets() const;</code>	
37		
38	<code> uint16_t getHeight() const;</code>	
39		
40	<code> uint16_t getWidth() const;</code>	
41		
42	<code> uint8_t getTileHeight() const;</code>	
43		
44	<code> uint8_t getTileWidth() const;</code>	
45	<code>};</code>	
46		
47		
48	<code>#endif //ARGENTUM_TILEDMAP_H</code>	

jul 21, 20 15:20	TiledMap.cpp	Page 1/2
1	<code>#include "TiledMap.h"</code>	
2		
3	<code>#include <utility></code>	
4		
5	<code>TiledMap::TiledMap(rapidjson::Document & json) {</code>	
6	<code> width = json["width"].GetInt();</code>	
7	<code> height = json["height"].GetInt();</code>	
8	<code> tileWidth = json["tilewidth"].GetInt();</code>	
9	<code> tileHeight = json["tileheight"].GetInt();</code>	
10		
11	<code> rapidjson::Value::Array layers = json["layers"].GetArray();</code>	
12	<code> for (auto &aLayer : layers) {</code>	
13	<code> std::string type = aLayer["type"].GetString();</code>	
14	<code> if ("tilelayer" == type) {</code>	
15	<code> TileLayer tileSet(aLayer);</code>	
16	<code> tileLayers.push_back(tileSet);</code>	
17	<code> } else if ("objectgroup" == type) {</code>	
18	<code> ObjectLayer objectLayer(aLayer);</code>	
19	<code> objectLayers.push_back(objectLayer);</code>	
20	<code> }</code>	
21		
22	<code> uint8_t idCounter = 0;</code>	
23	<code> rapidjson::Value::Array tileSetArray = json["tilesets"].GetArray();</code>	
24	<code> for (auto &aTileSet : tileSetArray) {</code>	
25	<code> TileSet tileset(aTileSet, ++idCounter);</code>	
26	<code> tilesets.push_back(tileset);</code>	
27	<code> }</code>	
28	<code>}</code>	
29		
30	<code>TiledMap::TiledMap() = default;</code>	
31		
32	<code>const std::vector<TileLayer> &TiledMap::getTileLayers() const {</code>	
33	<code> return tileLayers;</code>	
34	<code>}</code>	
35		
36	<code>std::vector<ObjectLayer> TiledMap::getObjectLayers() {</code>	
37	<code> return objectLayers;</code>	
38	<code>}</code>	
39		
40	<code>const std::vector<TileSet> &TiledMap::getTilesets() const {</code>	
41	<code> return tilesets;</code>	
42	<code>}</code>	
43		
44	<code>TiledMap::TiledMap(TiledMap &other) noexcept {</code>	
45	<code> std::swap(height, other.height);</code>	
46	<code> std::swap(width, other.width);</code>	
47	<code> std::swap(tileHeight, other.tileHeight);</code>	
48	<code> std::swap(tileWidth, other.tileWidth);</code>	
49	<code> std::swap(tileLayers, other.tileLayers);</code>	
50	<code> std::swap(objectLayers, other.objectLayers);</code>	
51	<code> std::swap(tilesets, other.tilesets);</code>	
52	<code>}</code>	
53		
54	<code>TiledMap &TiledMap::operator=(TiledMap &other) noexcept {</code>	
55	<code> if (this == &other) {</code>	
56	<code> return *this;</code>	
57	<code> }</code>	
58	<code> std::swap(height, other.height);</code>	
59	<code> std::swap(width, other.width);</code>	
60	<code> std::swap(tileHeight, other.tileHeight);</code>	
61	<code> std::swap(tileWidth, other.tileWidth);</code>	
62	<code> std::swap(tileLayers, other.tileLayers);</code>	
63	<code> std::swap(objectLayers, other.objectLayers);</code>	
64	<code> std::swap(tilesets, other.tilesets);</code>	
65	<code> return *this;</code>	
66	<code>}</code>	

jul 21, 20 15:20	TiledMap.cpp	Page 2/2
67	uint16_t TiledMap::getHeight() const {	
68	return height;	
69	}	
70		
71	uint16_t TiledMap::getWidth() const {	
72	return width;	
73	}	
74		
75	uint8_t TiledMap::getTileHeight() const {	
76	return tileHeight;	
77	}	
78		
79	uint8_t TiledMap::getTileWidth() const {	
80	return tileWidth;	
81	}	
82		
83	TiledMap::TiledMap(uint16_t width, uint16_t height, uint8_t tileWidth, uint8_t tileHeight,	
84	std::vector<TileLayer> tileLayers, std::vector<TileSet> tiles	
85	ets) : width(width), height(height), tileWidth(tileWidth), tileHeight(tileHeight),	
86	tileLayers(std::move(tileLayers)), tilesets(std::move(tilesets)) {}	
87		
88	TiledMap::~TiledMap() = default;	
89		

jul 21, 20 15:20	Thread.h	Page 1/1
1	#ifndef THREAD_H	
2	#define THREAD_H	
3		
4	#include <thread>	
5		
6	class Thread {	
7	private:	
8	std::thread thread;	
9		
10	public:	
11	Thread();	
12	Thread(Thread ^other);	
13	Thread& operator=(Thread^ other);	
14		
15	void start();	
16	void join();	
17	virtual void run() = 0;	
18		
19	virtual ~Thread();	
20	Thread(const Thread&) = delete;	
21	Thread &operator=(const Thread&) = delete;	
22	};	
23		
24	#endif	
25		

jul 21, 20 15:20	Thread.cpp	Page 1/1
1	<code>#include "Thread.h"</code>	
2	<code>#include <utility></code>	
3		
4	<code>Thread::Thread() = default;</code>	
5		
6	<code>Thread::~Thread() = default;</code>	
7		
8	<code>void Thread::start() {</code>	
9	<code> this->thread = std::thread(&Thread::run, this);</code>	
10	<code>}</code>	
11		
12	<code>void Thread::join() {</code>	
13	<code> this->thread.join();</code>	
14	<code>}</code>	
15		
16	<code>Thread::Thread(Thread &other) {</code>	
17	<code> this->thread = std::move(other.thread);</code>	
18	<code>}</code>	
19		
20	<code>Thread& Thread::operator=(Thread &other) {</code>	
21	<code> this->thread = std::move(other.thread);</code>	
22	<code> return *this;</code>	
23	<code>}</code>	

jul 21, 20 15:20	StaticObject.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_STATICOBJECT_H</code>	
2	<code>#define ARGENTUM_TALLER_STATICOBJECT_H</code>	
3		
4	<code>#include <rapidjson/document.h></code>	
5	<code>#include "Point.h"</code>	
6		
7	<code>class StaticObject {</code>	
8	<code>private:</code>	
9	<code> Point topLeft;</code>	
10	<code> Point topRight;</code>	
11	<code> Point bottomLeft;</code>	
12	<code> Point bottomRight;</code>	
13	<code> std::string name;</code>	
14		
15	<code>public:</code>	
16	<code> explicit StaticObject(rapidjson::Value &value);</code>	
17		
18	<code> ~StaticObject();</code>	
19		
20	<code> const Point &getTopLeft() const;</code>	
21		
22	<code> const Point &getTopRight() const;</code>	
23		
24	<code> const Point &getBottomLeft() const;</code>	
25		
26	<code> const Point &getBottomRight() const;</code>	
27	<code> const std::string &getName() const;</code>	
28	<code>};</code>	
29		
30		
31		
32	<code>#endif // ARGENTUM_TALLER_STATICOBJECT_H</code>	

jul 21, 20 15:20	StaticObject.cpp	Page 1/1
1	#include "StaticObject.h"	
2		
3	StaticObject::~StaticObject() = default;	
4		
5	StaticObject::StaticObject(rapidjson::Value &value) {	
6	int x = value["x"].GetFloat();	
7	int y = value["y"].GetFloat();	
8	int width = value["width"].GetFloat();	
9	int height = value["height"].GetFloat();	
10	topLeft = Point(x, y);	
11	topRight = Point(x + width, y);	
12	bottomLeft = Point(x, y + height);	
13	bottomRight = Point(x + width, y + height);	
14	name = value["name"].GetString();	
15	}	
16		
17	const std::string &StaticObject::getName() const {	
18	return name;	
19	}	
20		
21	const Point &StaticObject::getTopLeft() const {	
22	return topLeft;	
23	}	
24		
25	const Point &StaticObject::getTopRight() const {	
26	return topRight;	
27	}	
28		
29	const Point &StaticObject::getBottomLeft() const {	
30	return bottomLeft;	
31	}	
32		
33	const Point &StaticObject::getBottomRight() const {	
34	return bottomRight;	
35	}	

jul 21, 20 15:20	Socket.h	Page 1/1
1	#ifndef SOCKET_H	
2	#define SOCKET_H	
3		
4	#include <stdint>	
5		
6	class Socket {	
7	private:	
8	int fd;	
9	//Resuelve la conexi3n con el host (si es indicado) y el puerto al cual el	
10	//socket se conecta. El file descriptor que es utilizado queda almacenado	
11	//en el Socket.	
12	void resolve_address(struct addrinfo* hints,	
13	const char* host, const char* port);	
14		
15	//Constructor solamente utilizado en accept.	
16	explicit Socket(int fd);	
17		
18	public:	
19	//Constructores de Socket. Se elimina el constructor por copia.	
20	//En caso de los constructores por movimiento se invalida a other	
21	//tomando el ownership del mismo.	
22	Socket();	
23	Socket(Socket& other);	
24	Socket& operator=(Socket& other);	
25		
26	//Se encarga de conectar el socket aceptador (del lado del servidor)	
27	//con el puerto indicado como par3metro e indicarle la cantidad m3xima	
28	//de conexiones en espera que puede haber a la vez. Lanza una	
29	//SocketException en caso de error.	
30	void bind_and_listen(const char* port, std::uint32_t max_waiting);	
31		
32	//Se encarga de conectar el cliente con el (host, port) indicando como	
33	//par3metro. Lanza una SocketException en caso de error.	
34	void connect(const char* host, const char* port);	
35		
36	//Acepta la conexi3n de un cliente. Devuelve al Socket por movimiento.	
37	//Lanza una SocketException en caso de error.	
38	Socket accept();	
39		
40	//Env3a a un stream que comienza en buffer de longitud length.	
41	//Devuelve la cantidad de bytes enviados.	
42	//Lanza una SocketException en caso de error.	
43	int send(const void* buffer, std::uint32_t length) const ;	
44		
45	//Almacena en un buffer de longitud length todos los bytes	
46	//recibidos en el socket self. devuelve la cantidad de bytes recibidos.	
47	//Lanza una SocketException en caso de error.	
48	int recieve(void* buffer, std::uint32_t length) const ;	
49		
50	//channel: SHUT_WR, SHUT_RD, SHUT_RDWR	
51	void shutdown(int channel);	
52		
53	//Realiza un cierre del Socket, invalidandolo.	
54	void close();	
55		
56	~Socket();	
57		
58	Socket(const Socket& other) = delete;	
59	Socket& operator=(const Socket& other) = delete;	
60	};	
61		
62	#endif	

jul 21, 20 15:20	SocketException.h	Page 1/1
1	#ifndef SOCKETEXCEPTION_H	
2	#define SOCKETEXCEPTION_H	
3		
4	#include <typeinfo>	
5	#define BUFF_SIZE 256	
6		
7	class SocketException : public std::exception {	
8	private:	
9	char msg_error[BUFF_SIZE];	
10	public:	
11	explicit SocketException(const char* msg, ...) noexcept;	
12		
13	virtual const char* what() const noexcept;	
14		
15	virtual ~SocketException() noexcept;	
16	};	
17		
18	#endif	

jul 21, 20 15:20	SocketException.cpp	Page 1/1
1	#include "SocketException.h"	
2	#include <errno.h>	
3	#include <cstdio>	
4	#include <cstring>	
5	#include <cstdlib>	
6		
7	SocketException::SocketException(const char* msg, ...) noexcept {	
8	int _errno = errno;	
9		
10	va_list args;	
11	va_start(args, msg);	
12	int s = vsprintf(msg_error, BUFF_SIZE, msg, args);	
13	va_end(args);	
14		
15	strncpy(msg_error+s, strerror(_errno), BUFF_SIZE-s);	
16	msg_error[BUFF_SIZE-1] = 0;	
17	}	
18		
19	const char* SocketException::what() const noexcept {	
20	return msg_error;	
21	}	
22		
23	SocketException::~SocketException() noexcept {}	

jul 21, 20 15:20	Socket.cpp	Page 1/3
1	<code>#include "Socket.h"</code>	
2	<code>#include "SocketException.h"</code>	
3	<code>#include <utility></code>	
4	<code>#include <cstring></code>	
5	<code>#include <unistd.h></code>	
6	<code>#include <stdexcept></code>	
7	<code>#include <netdb.h></code>	
8	<code>#include <sys/socket.h></code>	
9		
10	<code>#define FAIL_FD "No se pudo conectar a ning�n file descriptor."</code>	
11	<code>#define FAIL_GETADDR "Error en getaddrinfo."</code>	
12	<code>#define FAIL_BIND "Error al realizar el bind al puerto %s."</code>	
13	<code>#define FAIL_CONNECT "Error al conectarse al host %s y puerto %s."</code>	
14	<code>#define FAIL_ACCEPT "Error al intentar aceptar un nuevo cliente."</code>	
15	<code>#define FAIL_SEND "Error al enviar desde el socket %d."</code>	
16	<code>#define FAIL_RECV "Error al recibir desde el socket %d."</code>	
17		
18	<code>Socket::Socket() : fd(-1) {}</code>	
19		
20	<code>Socket::Socket(int fd) : fd(fd) {}</code>	
21		
22	<code>void Socket::resolve_address(struct addrinfo* hints,</code>	
23	<code>const char* host, const char* port) {</code>	
24	<code>struct addrinfo *results, *iter;</code>	
25	<code>int val = 1;</code>	
26	<code>if (getaddrinfo(host, port, hints, &results) != 0) {</code>	
27	<code>freeaddrinfo(results);</code>	
28	<code>throw SocketException(FAIL_GETADDR);</code>	
29	<code>}</code>	
30		
31	<code>for (iter = results; iter != nullptr; iter = iter->ai_next) {</code>	
32	<code>this->fd = ::socket(iter->ai_family, iter->ai_socktype, 0);</code>	
33		
34	<code>if (this->fd == -1)</code>	
35	<code>continue;</code>	
36		
37	<code>if (host == nullptr) {</code>	
38	<code>if (::bind(this->fd, iter->ai_addr, iter->ai_addrlen) == 0) {</code>	
39	<code>setsockopt(this->fd, SOL_SOCKET, SO_REUSEADDR, &val, sizeof(val));</code>	
40	<code>break;</code>	
41	<code>}</code>	
42	<code>else {</code>	
43	<code>if (::connect(this->fd, iter->ai_addr, iter->ai_addrlen) != -1)</code>	
44	<code>break;</code>	
45	<code>}</code>	
46	<code>::close(this->fd);</code>	
47	<code>this->fd = -1;</code>	
48	<code>}</code>	
49		
50	<code>freeaddrinfo(results);</code>	
51	<code>if (this->fd == -1)</code>	
52	<code>throw SocketException(FAIL_FD);</code>	
53	<code>return;</code>	
54	<code>}</code>	
55		
56	<code>void Socket::bind_and_listen(const char* port, std::uint32_t max_waiting) {</code>	
57	<code>struct addrinfo hints;</code>	
58		
59	<code>memset(&hints, 0, sizeof(struct addrinfo));</code>	
60	<code>hints.ai_family = AF_INET; //IPv4</code>	
61	<code>hints.ai_socktype = SOCK_STREAM;</code>	
62	<code>hints.ai_flags = AI_PASSIVE;</code>	
63	<code>try{</code>	
64	<code>resolve_address(&hints, nullptr, port);</code>	
65	<code>catch (const SocketException& e) {</code>	
66	<code>throw SocketException(FAIL_BIND, port);</code>	

jul 21, 20 15:20	Socket.cpp	Page 2/3
67	<code>}</code>	
68	<code>::listen(this->fd, max_waiting);</code>	
69		
70	<code>return;</code>	
71	<code>}</code>	
72		
73	<code>void Socket::connect(const char* host, const char* port) {</code>	
74	<code>struct addrinfo hints;</code>	
75		
76	<code>memset(&hints, 0, sizeof(struct addrinfo));</code>	
77	<code>hints.ai_family = AF_INET; //IPv4</code>	
78	<code>hints.ai_socktype = SOCK_STREAM;</code>	
79	<code>hints.ai_flags = 0;</code>	
80	<code>try{</code>	
81	<code>resolve_address(&hints, host, port);</code>	
82	<code>catch (const SocketException& e) {</code>	
83	<code>throw SocketException(FAIL_CONNECT, host, port);</code>	
84	<code>}</code>	
85	<code>return;</code>	
86	<code>}</code>	
87		
88	<code>Socket Socket::accept() {</code>	
89	<code>int new_fd = ::accept(this->fd, nullptr, nullptr);</code>	
90	<code>if (new_fd == -1)</code>	
91	<code>throw SocketException(FAIL_ACCEPT);</code>	
92	<code>return std::move(Socket(new_fd));</code>	
93	<code>}</code>	
94		
95	<code>int Socket::send(const void* buffer, std::uint32_t length) const {</code>	
96	<code>std::uint32_t send_bytes = 0;</code>	
97	<code>int result_send;</code>	
98	<code>std::uint32_t remaining_bytes = length;</code>	
99	<code>const char* char_buffer = (const char*) buffer;</code>	
100	<code>while(send_bytes < length) {</code>	
101	<code>result_send = ::send(this->fd, &char_buffer[send_bytes],</code>	
102	<code>remaining_bytes, MSG_NOSIGNAL);</code>	
103	<code>if(result_send == -1 � result_send == 0)</code>	
104	<code>throw SocketException(FAIL_SEND, this->fd);</code>	
105	<code>send_bytes += result_send;</code>	
106	<code>remaining_bytes -= result_send;</code>	
107	<code>}</code>	
108	<code>return send_bytes;</code>	
109	<code>}</code>	
110		
111	<code>int Socket::recieve(void* buffer, std::uint32_t length) const {</code>	
112	<code>std::uint32_t received_bytes = 0;</code>	
113	<code>int result_recv;</code>	
114	<code>std::uint32_t remaining_bytes = length;</code>	
115	<code>char* char_buffer = (char*) buffer;</code>	
116		
117	<code>while(received_bytes < length) {</code>	
118	<code>result_recv = ::recv(this->fd, &char_buffer[received_bytes],</code>	
119	<code>remaining_bytes, 0);</code>	
120	<code>if(result_recv == -1 � result_recv == 0)</code>	
121	<code>throw SocketException(FAIL_RECV, this->fd);</code>	
122	<code>received_bytes += result_recv;</code>	
123	<code>remaining_bytes -= result_recv;</code>	
124	<code>}</code>	
125	<code>return received_bytes;</code>	
126	<code>}</code>	
127		
128	<code>Socket::Socket(Socket& other) : fd(other.fd) {</code>	
129	<code>other.fd = -1;</code>	
130	<code>}</code>	
131		
132	<code>Socket& Socket::operator=(Socket& other) {</code>	

jul 21, 20 15:20	Socket.cpp	Page 3/3
133	<code>this->fd = other.fd;</code>	
134	<code>other.fd = -1;</code>	
135	<code>return this;</code>	
136	<code>}</code>	
137		
138	<code>void Socket::shutdown(int channel) {</code>	
139	<code> ::shutdown(this->fd, channel);</code>	
140	<code>}</code>	
141		
142	<code>void Socket::close() {</code>	
143	<code> if (this->fd != -1){</code>	
144	<code> shutdown(SHUT_RDWR);</code>	
145	<code> ::close(this->fd);</code>	
146	<code> }</code>	
147	<code> this->fd = -1;</code>	
148	<code>}</code>	
149		
150	<code>Socket::~Socket() {</code>	
151	<code> this->close();</code>	
152	<code>}</code>	

jul 21, 20 15:20	Random.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_RANDOM_H</code>	
2	<code>#define ARGENTUM_TALLER_RANDOM_H</code>	
3		
4		
5	<code>class Random {</code>	
6	<code>public:</code>	
7	<code> static int get(int minRand, int maxRand);</code>	
8	<code> static float getFloat(float minRand, float maxRand);</code>	
9	<code>};</code>	
10		
11		
12	<code>#endif //ARGENTUM_TALLER_RANDOM_H</code>	

jul 21, 20 15:20	Random.cpp	Page 1/1
	<pre> 1 #include "Random.h" 2 #include <random> 3 4 float Random::getFloat(float minRand, float maxRand) { 5 std::random_device rd; 6 std::mt19937 mt(rd()); 7 std::uniform_real_distribution<float> dist(minRand, maxRand); 8 return dist(mt); 9 } 10 11 int Random::get(int minRand, int maxRand) { 12 std::random_device rd; 13 std::mt19937 mt(rd()); 14 std::uniform_int_distribution<int> dist(minRand, maxRand); 15 return dist(mt); 16 } </pre>	

jul 21, 20 15:20	Point.h	Page 1/1
	<pre> 1 #ifndef POINT_H 2 #define POINT_H 3 #include <cmath> 4 class Point { 5 public: 6 float x,y; 7 8 explicit Point(float x=0, float y=0); 9 10 Point& operator=(const Point& p2); 11 12 friend Point operator+(Point p1, const Point& p2); 13 14 Point& operator +=(const Point& p2); 15 16 friend Point operator-(Point p1, const Point& p2); 17 18 Point& operator-=(const Point& p2); 19 20 friend Point operator*(const Point& p1, float n); 21 22 friend bool operator==(Point p1, const Point& p2); 23 24 friend bool operator!=(Point p1, const Point& p2); 25 26 float distance(const Point& p2); 27 }; 28 29 #endif </pre>	

jul 21, 20 15:20	Point.cpp	Page 1/1
	<pre> 1 #include "Point.h" 2 3 #include <cmath> 4 5 Point::Point(float x, float y) : x(x), y(y) {} 6 7 Point& Point::operator=(const Point& p2) { 8 this->x = p2.x; 9 this->y = p2.y; 10 return *this; 11 } 12 13 Point operator+(Point p1, const Point& p2) { 14 return Point(p1.x+p2.x, p1.y+p2.y); 15 } 16 17 Point& Point::operator+=(const Point& p2) { 18 this->x += p2.x; 19 this->y += p2.y; 20 return *this; 21 } 22 23 Point& Point::operator-=(const Point& p2) { 24 this->x -= p2.x; 25 this->y -= p2.y; 26 return *this; 27 } 28 29 Point operator-(Point p1, const Point& p2) { 30 return Point(p1.x-p2.x, p1.y-p2.y); 31 } 32 33 bool operator==(Point p1, const Point& p2) { 34 return (p1.x==p2.x) ^ (p1.y == p2.y); 35 } 36 37 Point operator*(const Point& p1, float n) { 38 return Point(p1.x*n, p1.y*n); 39 } 40 41 bool operator!=(Point p1, const Point& p2) { 42 return !(p1==p2); 43 } 44 45 float Point::distance(const Point& p2) { 46 Point dist = *this - p2; 47 return std::sqrt(dist.x * dist.x + dist.y * dist.y); 48 } </pre>	

jul 21, 20 15:20	PlayerInfo.h	Page 1/1
	<pre> 1 #ifndef ARGENTUM_TALLER_PLAYERINFO_H 2 #define ARGENTUM_TALLER_PLAYERINFO_H 3 4 #include <zconf.h> 5 #include <string> 6 #include "GameObjectInfo.h" 7 #include "Identificators.h" 8 9 class PlayerInfo : public GameObjectInfo{ 10 private: 11 uint goldAmount{}; 12 uint life{}; 13 uint mana{}; 14 uint level{}; 15 uint exp{}; 16 uint maxLife{}; 17 uint maxMana{}; 18 uint maxExp{}; 19 uint safeGold{}; 20 std::string inventory; 21 std::string name; 22 23 public: 24 PlayerInfo(); 25 PlayerInfo(uint id, Point point, uint goldAmount, uint life, uint mana, 26 const std::string& textureHashId, Direction direction, uint safeGold, 27 uint maxLife, uint maxMana, uint exp, uint maxExp, uint level, 28 std::string inventory, CharacterStateID state, WeaponID attackBy); 29 30 ~PlayerInfo() override; 31 32 uint getGoldAmount() const; 33 34 uint getLife() const; 35 36 uint getMana() const; 37 38 Direction getDirection() const; 39 40 uint getExp() const; 41 42 uint getLevel() const; 43 44 uint getSafeGold() const; 45 46 uint getMaxLife() const; 47 48 uint getMaxMana() const; 49 50 uint getMaxExp() const; 51 52 std::string getName() const; 53 54 std::string getInventory() const; 55 }; 56 57 #endif // ARGENTUM_TALLER_PLAYERINFO_H </pre>	

jul 21, 20 15:20	PlayerInfo.cpp	Page 1/2
1	<code>#include "PlayerInfo.h"</code>	
2		
3	<code>#include <utility></code>	
4		
5	<code>PlayerInfo::~PlayerInfo() = default;</code>	
6		
7	<code>uint PlayerInfo::getGoldAmount() const {</code>	
8	<code> return goldAmount;</code>	
9	<code>}</code>	
10		
11	<code>uint PlayerInfo::getLife() const {</code>	
12	<code> return life;</code>	
13	<code>}</code>	
14		
15	<code>uint PlayerInfo::getMana() const {</code>	
16	<code> return mana;</code>	
17	<code>}</code>	
18		
19	<code>PlayerInfo::PlayerInfo(uint id, Point point, uint goldAmount, uint life, uint ma</code>	
20	<code>na, const std::string& textureHashId,</code>	
21	<code>Direction direction, uint safeGold, uint maxLife, uint maxMana, uint exp, ui</code>	
22	<code>nt maxExp,</code>	
23	<code>uint level, std::string inventory, CharacterStateID state, WeaponID attackBy</code>	
24	<code>)</code>	
25	<code> :GameObjectInfo(id, point, textureHashId, direction, state, false, attackBy),</code>	
26	<code>goldAmount(goldAmount), life(life), mana(mana), level(level), exp(exp), maxL</code>	
27	<code>ife(maxLife),</code>	
28	<code>maxMana(maxMana), maxExp(maxExp), safeGold(safeGold) , inventory(std::move(i</code>	
29	<code>nventory)) {}</code>	
30		
31	<code>PlayerInfo::PlayerInfo() : GameObjectInfo() {}</code>	
32		
33	<code>Direction PlayerInfo::getDirection() const {</code>	
34	<code> return this->direction;</code>	
35	<code>}</code>	
36		
37	<code>uint PlayerInfo::getMaxExp() const {</code>	
38	<code> return this->maxExp;</code>	
39	<code>}</code>	
40		
41	<code>uint PlayerInfo::getMaxLife() const {</code>	
42	<code> return this->maxLife;</code>	
43	<code>}</code>	
44		
45	<code>uint PlayerInfo::getMaxMana() const {</code>	
46	<code> return this->maxMana;</code>	
47	<code>}</code>	
48		
49	<code>uint PlayerInfo::getSafeGold() const {</code>	
50	<code> return this->safeGold;</code>	
51	<code>}</code>	
52		
53	<code>uint PlayerInfo::getExp() const {</code>	
54	<code> return this->exp;</code>	
55	<code>}</code>	
56		
57	<code>uint PlayerInfo::getLevel() const {</code>	
58	<code> return this->level;</code>	
59	<code>}</code>	
60		
61	<code>std::string PlayerInfo::getName() const {</code>	
	<code> return this->name;</code>	
	<code>}</code>	
	<code>std::string PlayerInfo::getInventory() const {</code>	
	<code> return this->inventory;</code>	

jul 21, 20 15:20	PlayerInfo.cpp	Page 2/2
62	<code>}</code>	

jul 21, 20 15:20

ObjectLayer.h

Page 1/1

```

1  #ifndef ARGENTUM_TALLER_OBJECTLAYER_H
2  #define ARGENTUM_TALLER_OBJECTLAYER_H
3
4  #include <string>
5  #include <vector>
6  #include <rapidjson/pointer.h>
7  #include "StaticObject.h"
8
9  class ObjectLayer {
10 private:
11     std::string name;
12     std::vector<StaticObject> objects;
13
14 public:
15     explicit ObjectLayer(rapidjson::Value&);
16
17     ~ObjectLayer();
18
19     const std::string &getName() const;
20
21     std::vector<StaticObject> getObjects();
22 };
23
24
25 #endif //ARGENTUM_TALLER_OBJECTLAYER_H

```

jul 21, 20 15:20

ObjectLayer.cpp

Page 1/1

```

1  #include "ObjectLayer.h"
2
3  ObjectLayer::ObjectLayer(rapidjson::Value &value) {
4      name = value["name"].GetString();
5      rapidjson::Value::Array objectsArray = value["objects"].GetArray();
6      for (auto &objectValue : objectsArray) {
7          StaticObject mapObject(objectValue);
8          objects.push_back(std::move(mapObject));
9      }
10 }
11
12 ObjectLayer::~ObjectLayer() = default;
13
14 const std::string &ObjectLayer::getName() const {
15     return name;
16 }
17
18 std::vector<StaticObject> ObjectLayer::getObjects() {
19     return objects;
20 }
21
22
23

```

jul 21, 20 15:20	Message.h	Page 1/1
1	<code>#ifndef ARGENTUM_TALLER_MESSAGE_H</code>	
2	<code>#define ARGENTUM_TALLER_MESSAGE_H</code>	
3		
4		
5	<code>#include <vector></code>	
6	<code>#include <stdint></code>	
7		
8	<code>//Wrapper del mensaje recibido por el socket.</code>	
9	<code>//Se puede obtener la longitud del mismo, su tipo de mensaje</code>	
10	<code>//y la informaci3n propiamente del mensaje enviado.</code>	
11	<code>//Adem3s provee un mecanismo de lectura de distintos bytes.</code>	
12	<code>class Message {</code>	
13	<code>private:</code>	
14	<code>std::vector<uint8_t> data;</code>	
15	<code>uint32_t length{};</code>	
16	<code>uint8_t type{};</code>	
17	<code>uint32_t pos = 0;</code>	
18	<code>public:</code>	
19	<code>Message();</code>	
20	<code>Message(std::vector<uint8_t>& data, uint32_t length, uint8_t type);</code>	
21		
22	<code>~Message();</code>	
23	<code>uint8_t getType() const;</code>	
24		
25	<code>void clear();</code>	
26		
27	<code>uint8_t read8();</code>	
28		
29	<code>uint16_t read16();</code>	
30		
31	<code>uint32_t read32();</code>	
32	<code>};</code>	
33		
34	<code>#endif //ARGENTUM_TALLER_MESSAGE_H</code>	
35		
36		

jul 21, 20 15:20	Message.cpp	Page 1/2
1	<code>#include "Message.h"</code>	
2		
3	<code>#include <utility></code>	
4	<code>#include "Exception.h"</code>	
5		
6	<code>#define ERRORREAD "Se quiere ingresar a una posicion no valida"</code>	
7		
8	<code>Message::~Message() = default;</code>	
9		
10	<code>Message::Message() = default;</code>	
11	<code>uint8_t Message::getType() const {</code>	
12	<code>return type;</code>	
13	<code>}</code>	
14		
15		
16	<code>Message::Message(std::vector<uint8_t>& data, uint32_t length, uint8_t type) :</code>	
17	<code>data(std::move(data)), length(length), type(type) {}</code>	
18		
19	<code>void Message::clear() {</code>	
20	<code>pos = 0;</code>	
21	<code>}</code>	
22		
23	<code>uint32_t conversorTo32(const uint8_t* value) {</code>	
24	<code>uint8_t temp[4];</code>	
25	<code>uint32_t* temp32;</code>	
26	<code>for (int j = 0; j < 4; j++) {</code>	
27	<code>temp[j] = value[j];</code>	
28	<code>}</code>	
29	<code>temp32 = (uint32_t*) temp;</code>	
30	<code>return *temp32;</code>	
31	<code>}</code>	
32		
33	<code>uint16_t conversorTo16(const uint8_t* value) {</code>	
34	<code>uint8_t temp[4];</code>	
35	<code>uint16_t* temp16;</code>	
36	<code>for (int j = 0; j < 2; j++) {</code>	
37	<code>temp[j] = value[j];</code>	
38	<code>}</code>	
39	<code>temp16 = (uint16_t*) temp;</code>	
40	<code>return *temp16;</code>	
41	<code>}</code>	
42		
43	<code>uint8_t Message::read8() {</code>	
44	<code>if (pos > length) {</code>	
45	<code>throw Exception(ERRORREAD);</code>	
46	<code>}</code>	
47	<code>uint8_t value = *(data.data() + pos);</code>	
48	<code>pos++;</code>	
49	<code>return value;</code>	
50	<code>}</code>	
51		
52	<code>uint16_t Message::read16() {</code>	
53	<code>if (pos + 2 > length) {</code>	
54	<code>throw Exception(ERRORREAD);</code>	
55	<code>}</code>	
56	<code>uint16_t value = conversorTo16(data.data() + pos);</code>	
57	<code>pos += 2;</code>	
58	<code>return value;</code>	
59	<code>}</code>	
60		
61	<code>uint32_t Message::read32() {</code>	
62	<code>if (pos + 4 > length) {</code>	
63	<code>throw Exception(ERRORREAD);</code>	
64	<code>}</code>	
65	<code>uint32_t value = conversorTo32(data.data() + pos);</code>	
66	<code>pos += 4;</code>	

jul 21, 20 15:20	Message.cpp	Page 2/2
67	return value;	
68	}	

jul 21, 20 15:20	JsonReader.h	Page 1/1
1	#ifndef ARGENTUM_JSONREADER_H	
2	#define ARGENTUM_JSONREADER_H	
3		
4		
5	#include <rapidjson/document.h>	
6		
7	class JsonReader {	
8	public:	
9	JsonReader();	
10		
11	virtual ~JsonReader();	
12		
13	static rapidjson::Document read(const std::string&);	
14		
15	};	
16		
17		
18	#endif //ARGENTUM_JSONREADER_H	

jul 21, 20 15:20	JsonReader.cpp	Page 1/1
1	#include <fstream>	
2	#include <rapidjson/istreamwrapper.h>	
3	#include "JsonReader.h"	
4	#include <rapidjson/error/en.h>	
5	#include "Exception.h"	
6		
7	JsonReader::JsonReader() = default ;	
8		
9	JsonReader::~JsonReader() = default ;	
10		
11	rapidjson::Document JsonReader::read(const std::string & filename) {	
12	std::ifstream file(filename);	
13	rapidjson::IStreamWrapper isw(file);	
14	rapidjson::Document json;	
15	json.ParseStream(isw);	
16	if (json.HasParseError()) {	
17	throw Exception("There are some problems with parsing", rapidjson::GetParseError_E	
18	n(json.GetParseError()));	
19	}	
20	return json;	

jul 21, 20 15:20	InputQueue.h	Page 1/1
1	#ifndef INPUTQUEUE_H	
2	#define INPUTQUEUE_H	
3		
4	#include "BlockingQueue.h"	
5	#include "Identificators.h"	
6		
7	using InputQueue = BlockingQueue<InputInfo>;	
8		
9	#endif	

jul 21, 20 15:20	Identificators.h	Page 1/4
1	<code>#ifndef CHARACTERSTATESID_H</code>	
2	<code>#define CHARACTERSTATESID_H</code>	
3		
4	<code>#include "Point.h"</code>	
5	<code>#include <vector></code>	
6	<code>#include <unordered_map></code>	
7	<code>#include <iostream></code>	
8		
9	<code>#ifdef DEV</code>	
10	<code>#define ROOT_DIR "/"</code>	
11	<code>#define CONFIG_DIR ".json"</code>	
12	<code>#else</code>	
13	<code>#define ROOT_DIR "/var/Argentum"</code>	
14	<code>#define CONFIG_DIR "/etc/Argentum"</code>	
15	<code>#endif</code>	
16	<code>enum class CharacterStateID {</code>	
17	<code>Still, //Aplica a NPCServer</code>	
18	<code>Move, //Aplica a NPCServer</code>	
19	<code>Attack, //Aplica a NPCServer</code>	
20	<code>Meditate,</code>	
21	<code>Interact,</code>	
22	<code>Resurrect,</code>	
23	<code>};</code>	
24		
25	<code>enum class InputID {</code>	
26	<code>nothing,</code>	
27	<code>stopMove,</code>	
28	<code>up, //w</code>	
29	<code>down, //s</code>	
30	<code>left, //a</code>	
31	<code>right, //d</code>	
32	<code>meditate, //q</code>	
33	<code>resurrect, //r</code>	
34	<code>cure, //h</code>	
35	<code>buy,</code>	
36	<code>sell,</code>	
37	<code>depositItem,</code>	
38	<code>retireItem,</code>	
39	<code>depositGold,</code>	
40	<code>retireGold,</code>	
41	<code>selectTarget, //click</code>	
42	<code>equipItem,</code>	
43	<code>dropItem,</code>	
44	<code>takeItem, //e</code>	
45	<code>unequipItem,</code>	
46	<code>};</code>	
47		
48	<code>enum class HelmetID {</code>	
49	<code>Nothing,</code>	
50	<code>Hood,</code>	
51	<code>IronHelmet,</code>	
52	<code>MagicHat,</code>	
53	<code>};</code>	
54		
55	<code>enum class BodyID {</code>	
56	<code>Nothing</code>	
57	<code>RedCommon,</code>	
58	<code>BlueCommon,</code>	
59	<code>GreenCommon,</code>	
60	<code>BlueTunic,</code>	
61	<code>LeatherArmor,</code>	
62	<code>PlateArmor,</code>	
63	<code>Ghost,</code>	
64	<code>Banker,</code>	
65	<code>Merchant,</code>	
66		

jul 21, 20 15:20	Identificators.h	Page 2/4
67	<code>Priest,</code>	
68	<code>Goblin,</code>	
69	<code>Skeleton,</code>	
70	<code>Spider,</code>	
71	<code>Zombie,</code>	
72	<code>};</code>	
73		
74	<code>enum class HeadID {</code>	
75	<code>Nothing,</code>	
76	<code>Human,</code>	
77	<code>Elf,</code>	
78	<code>Dwarf,</code>	
79	<code>Gnome,</code>	
80	<code>Priest,</code>	
81	<code>Zombie,</code>	
82	<code>};</code>	
83		
84	<code>enum class ShieldID {</code>	
85	<code>Nothing,</code>	
86	<code>TurtleShield,</code>	
87	<code>IronShield,</code>	
88	<code>};</code>	
89		
90	<code>enum class WeaponID {</code>	
91	<code>Nothing,</code>	
92	<code>SimpleArc,</code>	
93	<code>CompoundArc,</code>	
94	<code>LongSword,</code>	
95	<code>Hammer,</code>	
96	<code>Ax,</code>	
97	<code>ElficFlaute,</code>	
98	<code>AshStick,</code>	
99	<code>GnarledStick,</code>	
100	<code>Crosier,</code>	
101	<code>};</code>	
102		
103	<code>enum class ItemsInventoryID {</code>	
104	<code>Nothing,</code>	
105	<code>SimpleArc,</code>	
106	<code>CompoundArc,</code>	
107	<code>LongSword,</code>	
108	<code>Hammer,</code>	
109	<code>Ax,</code>	
110	<code>ElficFlaute,</code>	
111	<code>AshStick,</code>	
112	<code>GnarledStick,</code>	
113	<code>Crosier,</code>	
114	<code>TurtleShield,</code>	
115	<code>IronShield,</code>	
116	<code>Hood,</code>	
117	<code>IronHelmet,</code>	
118	<code>MagicHat,</code>	
119	<code>RedCommon,</code>	
120	<code>BlueCommon,</code>	
121	<code>GreenCommon,</code>	
122	<code>BlueTunic,</code>	
123	<code>LeatherArmor,</code>	
124	<code>PlateArmor,</code>	
125	<code>HealthPotion,</code>	
126	<code>ManaPotion,</code>	
127	<code>Gold,</code>	
128	<code>};</code>	
129		
130	<code>enum class Direction{</code>	
131	<code>down,</code>	
132	<code>up,</code>	

jul 21, 20 15:20	Identificators.h	Page 3/4
133	left,	
134	right,	
135	};	
136		
137	enum class RaceID {	
138	Nothing,	
139	Human,	
140	Elf,	
141	Dwarf,	
142	Gnome	
143	};	
144		
145	enum class GameClassID {	
146	Nothing,	
147	Mage,	
148	Cleric,	
149	Paladin,	
150	Warrior	
151	};	
152		
153	enum class CreatureID {	
154	Nothing,	
155	Zombie,	
156	Skeleton,	
157	Spider,	
158	Goblin,	
159	};	
160		
161	enum class ActionsProfessionID {	
162	Nothing,	
163	DepositItem,	
164	DepositGold,	
165	RetireItem,	
166	RetireGold,	
167	Buy,	
168	Sell,	
169	Resurrect,	
170	Cure,	
171	};	
172		
173	struct NPCInfo {	
174	uint8_t type = 0;	
175	std::vector<ActionsProfessionID> actions;	
176	std::unordered_map<ItemsInventoryID,uint> items;	
177	uint32_t gold;	
178	std::vector<ItemsInventoryID> itemsInBank;	
179	};	
180		
181	struct InputInfo {	
182	InputID input = InputID::nothing;	
183	Point position;	
184	int additional = 0;	
185	};	
186		
187	enum class StateID {	
188	Still,	
189	Move,	
190	Equip,	
191	Transition,	
192	Interact,	
193	Attack,	
194	Pursuit,	
195	Meditate,	
196	TakeDrop,	
197	Resurrect,	
198	};	

jul 21, 20 15:20	Identificators.h	Page 4/4
199		
200	#endif	

jul 21, 20 15:20	GameObjectInfo.h	Page 1/1
1	#ifndef ARGENTUM_TALLER_GAMEOBJECTINFO_H	
2	#define ARGENTUM_TALLER_GAMEOBJECTINFO_H	
3		
4		
5	#include <zconf.h>	
6	#include <string>	
7	#include "Identificators.h"	
8		
9	class GameObjectInfo {	
10	protected:	
11	uint id{};	
12	Point point;	
13	Direction direction;	
14	std::string textureHashId;	
15	CharacterStateID state;	
16	bool item;	
17	WeaponID attackBy;	
18	public:	
19	GameObjectInfo();	
20	GameObjectInfo(uint id, const Point &point, std::string textureHashId, Direc	
21	tion direction,	
22	CharacterStateID state, bool item, WeaponID attackBy);	
23		
24	const Point &getPoint() const ;	
25	const std::string &getTextureHashId() const ;	
26		
27	virtual ~GameObjectInfo();	
28	uint getId() const ;	
29		
30	float getX() const ;	
31		
32	float getY() const ;	
33		
34	Direction getDirection() const ;	
35		
36	HelmetID getHelmetID() const ;	
37	HeadID getHeadID() const ;	
38	BodyID getBodyID() const ;	
39	WeaponID getWeaponID() const ;	
40	ShieldID getShieldID() const ;	
41	ItemsInventoryID getItemID() const ;	
42	CharacterStateID getState() const ;	
43	WeaponID getAttackWeapon() const ;	
44		
45	bool isItem() const ;	
46	};	
47		
48		
49	#endif //ARGENTUM_TALLER_GAMEOBJECTINFO_H	
50		

jul 21, 20 15:20	GameObjectInfo.cpp	Page 1/2
1	#include "GameObjectInfo.h"	
2	#include <utility>	
3		
4	GameObjectInfo::~GameObjectInfo() = default ;	
5		
6	GameObjectInfo::GameObjectInfo(uint id, const Point &point, std::string textureH	
7	ashId, Direction aDirection,	
8	CharacterStateID state, bool item, WeaponID attackBy) : id(id), point(point	
9), direction(aDirection),	
10	textureHashId(std::move(textureHashId)), state(state), item(item), attackB	
11	y(attackBy) {}	
12		
13	uint GameObjectInfo::getId() const {	
14	return id;	
15	}	
16		
17	const Point &GameObjectInfo::getPoint() const {	
18	return point;	
19	}	
20	const std::string &GameObjectInfo::getTextureHashId() const {	
21	return textureHashId;	
22	}	
23	float GameObjectInfo::getX() const {	
24	return point.x;	
25	}	
26	float GameObjectInfo::getY() const {	
27	return point.y;	
28	}	
29		
30	HelmetID GameObjectInfo::getHelmetID() const {	
31	std::string stringId = this ->textureHashId.substr(2,2);	
32	int id = std::stoi(stringId);	
33	return (HelmetID)id;	
34	}	
35		
36	HeadID GameObjectInfo::getHeadID() const {	
37	std::string stringId = this ->textureHashId.substr(6,2);	
38	int id = std::stoi(stringId);	
39	return (HeadID)id;	
40	}	
41		
42	BodyID GameObjectInfo::getBodyID() const {	
43	std::string stringId = this ->textureHashId.substr(10,2);	
44	int id = std::stoi(stringId);	
45	return (BodyID)id;	
46	}	
47	WeaponID GameObjectInfo::getWeaponID() const {	
48	std::string stringId = this ->textureHashId.substr(18,2);	
49	int id = std::stoi(stringId);	
50	return (WeaponID)id;	
51	}	
52	ShieldID GameObjectInfo::getShieldID() const {	
53	std::string stringId = this ->textureHashId.substr(14,2);	
54	int id = std::stoi(stringId);	
55	return (ShieldID)id;	
56	}	
57		
58	ItemsInventoryID GameObjectInfo::getItemID() const {	
59	std::string stringId = this ->textureHashId.substr(22,2);	
60	int id = std::stoi(stringId);	
61	return (ItemsInventoryID)id;	
62	}	
63		

jul 21, 20 15:20	GameObjectInfo.cpp	Page 2/2
64	Direction GameObjectInfo::getDirection() const {	
65	return direction;	
66	}	
67		
68	bool GameObjectInfo::isItem() const {	
69	return this->item;	
70	}	
71		
72	CharacterStateID GameObjectInfo::getState() const {	
73	return this->state;	
74	}	
75		
76	WeaponID GameObjectInfo::getAttackWeapon() const {	
77	return this->attackBy;	
78	}	
79		
80	GameObjectInfo::GameObjectInfo() = default;	

jul 21, 20 15:20	Exception.h	Page 1/1
1	#ifndef EXCEPTION_H	
2	#define EXCEPTION_H	
3		
4	#include <typeinfo>	
5	#define BUFF_SIZE 256	
6		
7	class Exception : public std::exception {	
8	private:	
9	char msg_error[BUFF_SIZE];	
10	public:	
11	explicit Exception(const char* msg, ...) noexcept;	
12		
13	virtual const char* what() const noexcept;	
14		
15	virtual ~Exception() noexcept;	
16	};	
17		
18	#endif	

jul 21, 20 15:20	Exception.cpp	Page 1/1
	<pre> 1 #include "Exception.h" 2 #include <errno.h> 3 #include <stdio.h> 4 #include <string> 5 #include <stdarg.h> 6 7 Exception::Exception(const char* msg, ...) noexcept { 8 int _errno = errno; 9 10 va_list args; 11 va_start(args, msg); 12 int s = vsnprintf(msg_error, BUFF_SIZE, msg, args); 13 va_end(args); 14 15 strncpy(msg_error+s, strerror(_errno), BUFF_SIZE-s); 16 msg_error[BUFF_SIZE-1] = 0; 17 } 18 19 const char* Exception::what() const noexcept { 20 return msg_error; 21 } 22 23 Exception::~Exception() noexcept {} </pre>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Decoder.h	Page 1/1
	<pre> 1 #ifndef ARGENTUM_TALLER_DECODER_H 2 #define ARGENTUM_TALLER_DECODER_H 3 4 5 #include <stdint.h> 6 #include <vector> 7 #include "PlayerInfo.h" 8 #include "TiledMap.h" 9 #include "Message.h" 10 11 //Este protocolo binario de comunicaci3n est3 dise1ado para enviar el mapa, l 12 //informaci3n del jugador, los objetos renderizables del mapa(items, criaturas, 13 //la informaci3n del NPC con el que interactua y los comandos que ingresa el ju 14 gador. 15 16 class Decoder { 17 private: 18 static void encodeStatsPlayer(const PlayerInfo &info, std::vector<uint8_t>& 19 t); 20 static void encodeEquipmentPlayer(const PlayerInfo &info, std::vector<uint8_ 21 t>&); 22 static void encodeInventory(const PlayerInfo &info, std::vector<uint8_t>&); 23 static void encodeStatePlayer(const GameObjectInfo &info, std::vector<uint8_ 24 t>&); 25 static void convertorTo8(uint32_t value, uint8_t from, std::vector<uint8_t>& 26 encodeMsg); 27 static std::string decodeEquipment(Message& msg, bool isGameObject=false); 28 static std::string decodeInventory(Message& msg); 29 static void decodeItem(GameObjectInfo object, std::vector<uint8_t>&); 30 static void decodeCharacter(GameObjectInfo object, std::vector<uint8_t>&); 31 public: 32 Decoder(); 33 //De un PlayerInfo genera una codificaci3n correspondiente 34 static std::vector<uint8_t> encodePlayerInfo(const PlayerInfo &info); 35 36 static std::vector<uint8_t> encodeCommand(InputInfo input); 37 38 static std::vector<uint8_t> encodeInit(RaceID race, GameClassID gameClass); 39 40 static std::vector<uint8_t> encodeMap(const TiledMap&); 41 42 static std::vector<uint8_t> encodeNPCInfo(const NPCInfo& info); 43 44 static std::vector<uint8_t> encodeGameObjects(const std::vector<GameObjectIn 45 fo> &objects); 46 47 //Se recibe como par3metro el mensaje recibido desde el servidor 48 //Devuelve un PlayerInfo con toda la informaci3n del jugador 49 static PlayerInfo decodePlayerInfo(Message msg); 50 51 static InputInfo decodeCommand(Message msg); 52 53 static std::vector<GameObjectInfo> decodeGameObjects(Message msg); 54 55 static TiledMap decodeMap(Message msg); 56 57 static NPCInfo decodeNPCInfo(Message msg); 58 59 virtual ~Decoder(); 60 }; 61 62 #endif //ARGENTUM_TALLER_DECODER_H </pre>	

92/217

jul 21, 20 15:20	Decoder.cpp	Page 1/8
1	<code>#include <netinet/in.h></code>	
2	<code>#include "Decoder.h"</code>	
3		
4	<code>#define ZERO 0x00</code>	
5	<code>#define MAPMSG 0x00</code>	
6	<code>#define PLAYERINFMSG 0x01</code>	
7	<code>#define OBJECTSINFMSG 0x02</code>	
8	<code>#define COMMANDMSG 0x03</code>	
9	<code>#define INITMSG 0x04</code>	
10	<code>#define INTERACTMSG 0x05</code>	
11	<code>#define INITLENGTH 2</code>	
12	<code>#define OBJECTLENGTH 18</code>	
13	<code>#define PLAYERINFOLENGTH 47</code>	
14	<code>#define COMMANDLENGTH 7</code>	
15	<code>#define MAPLENGTH 240046</code>	
16	<code>#define ITEM 0x01</code>	
17	<code>#define CHARACTER 0x00</code>	
18	<code>#define LIMITINVENTORY 9</code>	
19		
20		
21	<code>Decoder::Decoder() = default;</code>	
22		
23	<code>void Decoder::convertorTo8(uint32_t value, uint8_t from, std::vector<uint8_t>& encodeMsg) {</code>	
24	<code> int max = 0;</code>	
25	<code> if (from == 16) {</code>	
26	<code> max = 2;</code>	
27	<code> } else if (from == 32) {</code>	
28	<code> max = 4;</code>	
29	<code> }</code>	
30	<code> auto* ptr = (uint8_t*) &value;</code>	
31	<code> for(int i = 0; i < max; i++)</code>	
32	<code> encodeMsg.push_back(*(ptr+i));</code>	
33	<code>}</code>	
34		
35	<code>void Decoder::encodeStatsPlayer(const PlayerInfo &info, std::vector<uint8_t>& encodeMsg) {</code>	
36	<code> uint16_t life, maxLife, mana, maxMana, level, gold, safeGold;</code>	
37	<code> uint32_t maxExp, exp;</code>	
38	<code> life = htons(info.getLife());</code>	
39	<code> maxLife = htons(info.getMaxLife());</code>	
40	<code> mana = htons(info.getMana());</code>	
41	<code> maxMana = htons(info.getMaxMana());</code>	
42	<code> exp = htonl(info.getExp());</code>	
43	<code> maxExp = htonl(info.getMaxExp());</code>	
44	<code> level = htons(info.getLevel());</code>	
45	<code> gold = htons(info.getGoldAmount());</code>	
46	<code> safeGold = htons(info.getSafeGold());</code>	
47	<code> convertorTo8(life, 16, encodeMsg);</code>	
48	<code> convertorTo8(maxLife, 16, encodeMsg);</code>	
49	<code> convertorTo8(mana, 16, encodeMsg);</code>	
50	<code> convertorTo8(maxMana, 16, encodeMsg);</code>	
51	<code> convertorTo8(exp, 32, encodeMsg);</code>	
52	<code> convertorTo8(maxExp, 32, encodeMsg);</code>	
53	<code> convertorTo8(level, 16, encodeMsg);</code>	
54	<code> convertorTo8(gold, 16, encodeMsg);</code>	
55	<code> convertorTo8(safeGold, 16, encodeMsg);</code>	
56	<code>}</code>	
57		
58	<code>void Decoder::encodeInventory(const PlayerInfo &info, std::vector<uint8_t>& encodeMsg) {</code>	
59	<code> std::string inventory = info.getInventory();</code>	
60	<code> std::string item;</code>	
61	<code> uint8_t idItem;</code>	
62	<code> for (int i = 0; i < LIMITINVENTORY; i++) {</code>	
63	<code> item = inventory.substr(2*i, 2);</code>	

jul 21, 20 15:20	Decoder.cpp	Page 2/8
64	<code>idItem = std::stoi(item);</code>	
65	<code>encodeMsg.push_back(idItem);</code>	
66	<code>}</code>	
67	<code>}</code>	
68		
69	<code>void Decoder::encodeEquipmentPlayer(const PlayerInfo &info, std::vector<uint8_t> & encodeMsg) {</code>	
70	<code> uint8_t idHelmet = (uint8_t) info.getHelmetID();</code>	
71	<code> uint8_t idHead = (uint8_t) info.getHeadID();</code>	
72	<code> uint8_t idBody = (uint8_t) info.getBodyID();</code>	
73	<code> uint8_t idShield = (uint8_t) info.getShieldID();</code>	
74	<code> uint8_t idWeapon = (uint8_t) info.getWeaponID();</code>	
75		
76	<code> encodeMsg.push_back(idHelmet);</code>	
77	<code> encodeMsg.push_back(idHead);</code>	
78	<code> encodeMsg.push_back(idBody);</code>	
79	<code> encodeMsg.push_back(idShield);</code>	
80	<code> encodeMsg.push_back(idWeapon);</code>	
81	<code>}</code>	
82		
83	<code>void Decoder::encodeStatePlayer(const GameObjectInfo &info, std::vector<uint8_t> & encodeMsg) {</code>	
84	<code> uint16_t state = htons((uint16_t) info.getState());</code>	
85	<code> uint16_t dir = htons((uint16_t) info.getDirection());</code>	
86	<code> uint16_t posX = htons((info.getPoint().x));</code>	
87	<code> uint16_t posY = htons((info.getPoint().y));</code>	
88	<code> convertorTo8(state, 16, encodeMsg);</code>	
89	<code> convertorTo8(dir, 16, encodeMsg);</code>	
90	<code> convertorTo8(posX, 16, encodeMsg);</code>	
91	<code> convertorTo8(posY, 16, encodeMsg);</code>	
92	<code>}</code>	
93		
94	<code>std::vector<uint8_t> Decoder::encodePlayerInfo(const PlayerInfo &info) {</code>	
95	<code> std::vector<uint8_t> encodeMsg;</code>	
96	<code> uint32_t length = PLAYERINFOLENGTH;</code>	
97	<code> length = htonl(length);</code>	
98	<code> convertorTo8(length, 32, encodeMsg);</code>	
99	<code> uint8_t type = PLAYERINFMSG;</code>	
100	<code> encodeMsg.push_back(type);</code>	
101	<code> uint16_t id = info.getId();</code>	
102	<code> id = htons(id);</code>	
103	<code> convertorTo8(id, 16, encodeMsg);</code>	
104	<code> encodeStatsPlayer(info, encodeMsg);</code>	
105	<code> encodeEquipmentPlayer(info, encodeMsg);</code>	
106	<code> encodeInventory(info, encodeMsg);</code>	
107	<code> encodeStatePlayer(info, encodeMsg);</code>	
108	<code> uint8_t attackBy = (uint8_t) info.getAttackWeapon();</code>	
109	<code> encodeMsg.push_back(attackBy);</code>	
110	<code> return encodeMsg;</code>	
111	<code>}</code>	
112		
113	<code>std::string Decoder::decodeEquipment(Message& msg, bool isGameObject) {</code>	
114	<code> std::string equipment;</code>	
115	<code> uint8_t id;</code>	
116	<code> std::string temp;</code>	
117	<code> equipment += "h";</code>	
118	<code> id = msg.read8();</code>	
119	<code> temp = std::to_string(id);</code>	
120	<code> if (temp.size() == 1)</code>	
121	<code> equipment += "0";</code>	
122	<code> equipment += temp + " ";</code>	
123	<code> equipment += "h";</code>	
124	<code> id = msg.read8();</code>	
125	<code> temp = std::to_string(id);</code>	
126	<code> if (temp.size() == 1)</code>	
127	<code> equipment += "0";</code>	

jul 21, 20 15:20	Decoder.cpp	Page 3/8
128	equipment += temp + " ";	
129	equipment += "b";	
130	id = msg.read8();	
131	temp = std::to_string(id);	
132	if (temp.size() == 1)	
133	equipment += "0";	
134	equipment += temp + " ";	
135	equipment += "s";	
136	id = msg.read8();	
137	temp = std::to_string(id);	
138	if (temp.size() == 1)	
139	equipment += "0";	
140	equipment += temp + " ";	
141	equipment += "w";	
142	id = msg.read8();	
143	temp = std::to_string(id);	
144	if (temp.size() == 1)	
145	equipment += "0";	
146	equipment += temp;	
147	if (isGameObject) {	
148	equipment += "i";	
149	id = msg.read8();	
150	temp = std::to_string(id);	
151	if (temp.size() == 1)	
152	equipment += "0";	
153	equipment += temp;	
154	}	
155	return equipment;	
156	}	
157		
158	std::string Decoder::decodeInventory(Message& msg) {	
159	std::string inventory;	
160	uint8_t id;	
161	std::string temp;	
162	for (int i=0; i<LIMITINVENTORY;i++) {	
163	id = msg.read8();	
164	temp = std::to_string(id);	
165	if (temp.size() == 1)	
166	inventory += "0";	
167	inventory += temp;	
168	if (i != LIMITINVENTORY-1)	
169	inventory += " ";	
170	}	
171	return inventory;	
172	}	
173		
174	PlayerInfo Decoder::decodePlayerInfo(Message msg) {	
175	msg.clear();	
176	uint16_t id = ntohs(msg.read16());	
177	uint16_t life = ntohs(msg.read16());	
178	uint16_t maxLife = ntohs(msg.read16());	
179	uint16_t mana = ntohs(msg.read16());	
180	uint16_t maxMana = ntohs(msg.read16());	
181	uint32_t exp = ntohl(msg.read32());	
182	uint32_t maxExp = ntohl(msg.read32());	
183	uint16_t level = ntohs(msg.read16());	
184	uint16_t gold = ntohs(msg.read16());	
185	uint16_t safeGold = ntohs(msg.read16());	
186	std::string equipment = decodeEquipment(msg, false);	
187	std::string inventory = decodeInventory(msg);	
188	auto state = (CharacterStateID) (ntohs(msg.read16()));	
189	auto dir = (Direction) (ntohs(msg.read16()));	
190	uint16_t x = ntohs(msg.read16());	
191	uint16_t y = ntohs(msg.read16());	
192	auto attackBy = (WeaponID) msg.read8();	
193	PlayerInfo info(id, Point(x,y), gold, life, mana, equipment, dir,	

jul 21, 20 15:20	Decoder.cpp	Page 4/8
194	safeGold,maxLife,maxMana,exp,maxExp,level,inventory,state,at	
195	ackBy);	
196	return info;	
197	}	
198	void Decoder::encodeItem(GameObjectInfo object, std::vector<uint8_t>& encodeMsg)	
199	{	
200	encodeMsg.push_back(ITEM);	
201	auto idHelmet = (uint8_t) object.getHelmetID();	
202	auto idHead = (uint8_t) object.getHeadID();	
203	auto idBody = (uint8_t) object.getBodyID();	
204	auto idShield = (uint8_t) object.getShieldID();	
205	auto idWeapon = (uint8_t) object.getWeaponID();	
206	auto idItem = (uint8_t) object.getItemID();	
207		
208	encodeMsg.push_back(idHelmet);	
209	encodeMsg.push_back(idHead);	
210	encodeMsg.push_back(idBody);	
211	encodeMsg.push_back(idShield);	
212	encodeMsg.push_back(idWeapon);	
213	encodeMsg.push_back(idItem);	
214	//Encodeo el estado y direccion del item con 0 dado que no me interesa su va	
215	lor	
216	for (int i=0; i<4;i++){	
217	encodeMsg.push_back(ZERO);	
218	}	
219	uint16_t posX = htons(object.getPoint().x);	
220	uint16_t posY = htons(object.getPoint().y);	
221	converterTo8(posX,16, encodeMsg);	
222	converterTo8(posY,16, encodeMsg);	
223	encodeMsg.push_back(ZERO); //Un item no puede ser atacado	
224	}	
225	void Decoder::encodeCharacter(GameObjectInfo object, std::vector<uint8_t>& encod	
226	eMsg) {	
227	encodeMsg.push_back(CHARACTER);	
228	auto idHelmet = (uint8_t) object.getHelmetID();	
229	auto idHead = (uint8_t) object.getHeadID();	
230	auto idBody = (uint8_t) object.getBodyID();	
231	auto idShield = (uint8_t) object.getShieldID();	
232	auto idWeapon = (uint8_t) object.getWeaponID();	
233	auto attackBy = (uint8_t) object.getAttackWeapon();	
234	encodeMsg.push_back(idHelmet);	
235	encodeMsg.push_back(idHead);	
236	encodeMsg.push_back(idBody);	
237	encodeMsg.push_back(idShield);	
238	encodeMsg.push_back(idWeapon);	
239	encodeMsg.push_back(ZERO); //Correspondiente al valor del item que no es	
240	encodeStatePlayer(object, encodeMsg);	
241	encodeMsg.push_back(attackBy);	
242	}	
243	std::vector<uint8_t> Decoder::encodeGameObjects(const std::vector<GameObjectInfo	
244	> &objects) {	
245	std::vector<uint8_t> encodeMsg;	
246	uint32_t cantObjects = objects.size();	
247	uint32_t length = cantObjects * OBJECTLENGTH + 4;	
248	length = htonl(length);	
249	cantObjects = htonl(cantObjects);	
250	uint8_t type = OBJECTSINFOMSG;	
251	converterTo8(length, 32, encodeMsg);	
252	encodeMsg.push_back(type);	
253	converterTo8(cantObjects, 32, encodeMsg);	
254	uint16_t id;	
255	for(auto& object: objects) {	

jul 21, 20 15:20	Decoder.cpp	Page 5/8
255	id = htons(object.getId());	
256	conversorTo8(id,16, encodeMsg);	
257	if(object.isItem()) {	
258	encodeItem(object, encodeMsg);	
259	} else {	
260	encodeCharacter(object, encodeMsg);	
261	}	
262	}	
263	return encodeMsg;	
264	}	
265		
266	std::vector<GameObjectInfo> Decoder::decodeGameObjects(Message msg) {	
267	std::vector<GameObjectInfo> objects;	
268	msg.clear();	
269	uint32_t cantObjects = ntohs(msg.read32());	
270	std::string equipment;	
271	for (uint i=0; i < cantObjects ; i++) {	
272	uint16_t id = ntohs(msg.read16());	
273	uint8_t type = msg.read8();	
274	equipment = decodeEquipment(msg, true);	
275	auto state = (CharacterStateID) (ntohs(msg.read16()));	
276	auto dir = (Direction) (ntohs(msg.read16()));	
277	uint16_t x = ntohs(msg.read16());	
278	uint16_t y = ntohs(msg.read16());	
279	auto attackBy = (WeaponID) msg.read8();	
280	GameObjectInfo info(id,Point(x,y),equipment,dir,state,type,attackBy);	
281	objects.push_back(info);	
282	}	
283	return objects;	
284	}	
285		
286	std::vector<uint8_t> Decoder::encodeCommand(InputInfo input) {	
287	std::vector<uint8_t> encodeMsg;	
288	uint32_t length = COMMANDLENGTH;	
289	length = htonl(length);	
290	conversorTo8(length,32, encodeMsg);	
291	uint8_t type = COMMANDMSG;	
292	encodeMsg.push_back(type);	
293	auto inputId = (uint8_t) input.input;	
294	encodeMsg.push_back(inputId);	
295		
296	uint16_t x = htons(input.position.x);	
297	conversorTo8(x,16, encodeMsg);	
298	uint16_t y = htons(input.position.y);	
299	conversorTo8(y,16, encodeMsg);	
300	uint16_t adition = htons(input.additional);	
301	conversorTo8(adition,16, encodeMsg);	
302		
303	return encodeMsg;	
304		
305	}	
306		
307	InputInfo Decoder::decodeCommand(Message msg) {	
308	msg.clear();	
309	InputInfo input;	
310	input.input = (InputID) msg.read8();	
311	input.position.x = ntohs(msg.read16());	
312	input.position.y = ntohs(msg.read16());	
313	input.additional = ntohs(msg.read16());	
314	return input;	
315	}	
316		
317	std::vector<uint8_t> Decoder::encodeInit(RaceID race, GameClassID gameClass) {	
318	std::vector<uint8_t> encodeMsg;	
319	uint32_t length = INITLENGTH;	
320	conversorTo8(htonl(length),32, encodeMsg);	

jul 21, 20 15:20	Decoder.cpp	Page 6/8
321	encodeMsg.push_back(INITMSG);	
322	auto breed = (uint8_t) race;	
323	auto playerClass = (uint8_t) gameClass;	
324	encodeMsg.push_back(breed);	
325	encodeMsg.push_back(playerClass);	
326	return encodeMsg;	
327	}	
328		
329	std::vector<uint8_t> Decoder::encodeMap(const TiledMap &tiledMap) {	
330	std::vector<uint8_t> encodeMsg;	
331	uint32_t length = MAPLENGTH;	
332	conversorTo8(htonl(length), 32, encodeMsg);	
333	encodeMsg.push_back(MAPMSG);	
334	conversorTo8(htonl(tiledMap.getWidth()), 16, encodeMsg);	
335	conversorTo8(htonl(tiledMap.getHeight()), 16, encodeMsg);	
336	encodeMsg.push_back(tiledMap.getWidth());	
337	encodeMsg.push_back(tiledMap.getHeight());	
338	std::vector<TileLayer> tilesLayers = tiledMap.getTileLayers();	
339	uint8_t amountLayers = tilesLayers.size();	
340	uint16_t dataSize = tilesLayers.at(0).getData().size();	
341	encodeMsg.push_back(amountLayers);	
342	conversorTo8(htonl(dataSize), 16, encodeMsg);	
343	for (auto &aTileLayer : tilesLayers) {	
344	encodeMsg.push_back(aTileLayer.isGroundLayer() ? 0x01 : ZERO);	
345	for (auto &aDataTile : aTileLayer.getData()) {	
346	conversorTo8(htonl(aDataTile), 16, encodeMsg);	
347	}	
348	}	
349	std::vector<TileSet> tilesSet = tiledMap.getTileSets();	
350	uint8_t amountTileSets = tilesSet.size();	
351	encodeMsg.push_back(amountTileSets);	
352	for (auto &aTileSet : tilesSet) {	
353	conversorTo8(htonl(aTileSet.getFirstgid()), 16, encodeMsg);	
354	encodeMsg.push_back(aTileSet.getId());	
355	}	
356	return encodeMsg;	
357	}	
358		
359	TiledMap Decoder::decodeMap(Message msg) {	
360	uint16_t width = ntohs(msg.read16());	
361	uint16_t height = ntohs(msg.read16());	
362	uint8_t tileWidth = msg.read8();	
363	uint8_t tileHeight = msg.read8();	
364	uint8_t amountLayers = msg.read8();	
365	uint16_t dataSize = ntohs(msg.read16());	
366	std::vector<TileLayer> tileLayers;	
367	for (size_t i = 0; i < amountLayers; ++i) {	
368	std::vector<uint16_t> data;	
369	bool isGround = msg.read8();	
370	for (size_t j = 0; j < dataSize; ++j) {	
371	data.push_back(ntohs(msg.read16()));	
372	}	
373	tileLayers.emplace_back(data, isGround);	
374	}	
375	std::vector<TileSet> tileSets;	
376	uint8_t amountTileSets = msg.read8();	
377	for (size_t i = 0; i < amountTileSets; ++i) {	
378	uint16_t firstGid = ntohs(msg.read16());	
379	uint8_t id = msg.read8();	
380	tileSets.emplace_back(firstGid, id);	
381	}	
382	return TiledMap(width, height, tileWidth, tileHeight, tileLayers, tileSets);	
383	}	
384		
385	std::vector<uint8_t> Decoder::encodeNPCInfo(const NPCInfo& info) {	
386	std::vector<uint8_t> encodeMsg;	

Page 7/8	Decoder.cpp	Page 7/8
387	uint32_t length = 0;	
388	convectorTo8(htonl(length), 32, encodeMsg);	
389	encodeMsg.push_back(INTERACTMSG);	
390	length += 1;	
391	encodeMsg.push_back(info.type);	
392	uint8_t cantAccions = info.actions.size();	
393	length += 1;	
394	encodeMsg.push_back(cantAccions);	
395	for (uint j = 0; j < info.actions.size(); j++) {	
396	length += 1;	
397	encodeMsg.push_back(uint8_t(info.actions[j]));	
398	}	
399	if (info.type != 0) {	
400	uint8_t cantItems = info.items.size();	
401	length += 1;	
402	encodeMsg.push_back(cantItems);	
403	for (auto iter: info.items) {	
404	length += 3;	
405	encodeMsg.push_back(uint8_t(iter.first));	
406	convectorTo8(htonl(iter.second), 16, encodeMsg);	
407	}	
408	} else {	
409	uint8_t cantItems = ZERO;	
410	length += 1;	
411	encodeMsg.push_back(cantItems);	
412	}	
413	uint32_t gold = info.gold;	
414	length += 4;	
415	convectorTo8(htonl(gold), 32, encodeMsg);	
416	if (info.type == 3) {	
417	uint8_t cantItems = info.itemsInBank.size();	
418	length += 1;	
419	encodeMsg.push_back(cantItems);	
420	for (auto iter: info.itemsInBank) {	
421	length += 1;	
422	encodeMsg.push_back(uint8_t(iter));	
423	}	
424	} else {	
425	uint8_t cantItems = ZERO;	
426	length += 1;	
427	encodeMsg.push_back(cantItems);	
428	}	
429	length = htonl(length);	
430	auto* ptr = (uint8_t*) &length;	
431	for (int i = 0; i < 4; i++)	
432	encodeMsg.at(i) = *(ptr+i);	
433		
434	return encodeMsg;	
435	}	
436		
437	NPCInfo Decoder::decodeNPCInfo(Message msg){	
438	NPCInfo info;	
439	info.type = msg.read8();	
440	uint8_t cantAccions = msg.read8();	
441	for (uint8_t i=0; i < cantAccions; i++) {	
442	info.actions.push_back(ActionsProfessionID(msg.read8()));	
443	}	
444	uint8_t cantItems = msg.read8();	
445	for (uint8_t i=0; i < cantItems; i++) {	
446	ItemsInventoryID id = ItemsInventoryID(msg.read8());	
447	uint16_t cost = ntohs(msg.read16());	
448	info.items.insert({id,cost});	
449	}	
450	info.gold = ntohl(msg.read32());	
451	uint8_t cantItemsInBank = msg.read8();	
452	for (uint8_t i=0; i < cantItemsInBank; i++) {	

Page 8/8	Decoder.cpp	Page 8/8
453	info.itemsInBank.push_back(ItemsInventoryID(msg.read8()));	
454	}	
455	return info;	
456	}	
457		
458	Decoder::~Decoder() = default;	

jul 21, 20 15:20	DataQueue.h	Page 1/1
1	#ifndef DATAQUEUE_H	
2	#define DATAQUEUE_H	
3		
4	#include "BlockingQueue.h"	
5	#include "Message.h"	
6	#include <utility>	
7	#include <vector>	
8		
9	using DataQueue = BlockingQueue<Message>;	
10		
11	#endif	

mar 21 jul 2020 15:20:01 ART

jul 21, 20 15:20	CommunicationProtocol.h	Page 1/1
1	#ifndef COMMUNICATIONPROTOCOL_H	
2	#define COMMUNICATIONPROTOCOL_H	
3		
4	#include <vector>	
5	#include "PlayerInfo.h"	
6	#include <cstdint>	
7	#include "GameObjectInfo.h"	
8	#include "../client/GameMap.h"	
9	#include "Socket.h"	
10	#include "Message.h"	
11		
12	<i>//Wrapper del socket utilizado para la comunicacion.s</i>	
13	class CommunicationProtocol {	
14	private:	
15	Socket socket;	
16	public:	
17	CommunicationProtocol();	
18	explicit CommunicationProtocol(Socket socket);	
19		
20	void connect(const char* host, const char* port);	
21		
22	void send(std::vector<uint8_t> msg) const;	
23		
24	Message receive() const;	
25		
26	void stop();	
27		
28	~CommunicationProtocol();	
29	};	
30		
31	#endif	

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

97/217

jul 21, 20 15:20	CommunicationProtocol.cpp	Page 1/1
	<pre> 1 #include "CommunicationProtocol.h" 2 #include <arpa/inet.h> 3 #include <utility> 4 #include <iostream> 5 6 CommunicationProtocol::CommunicationProtocol() : socket() {} 7 8 CommunicationProtocol::CommunicationProtocol(Socket socket) : 9 socket(std::move(socket)) {} 10 11 12 void CommunicationProtocol::connect(const char* host, const char* port) { 13 this->socket.connect(host, port); 14 } 15 16 void CommunicationProtocol::send(std::vector<uint8_t> msg) const { 17 this->socket.send(msg.data(), msg.size()); 18 } 19 20 Message CommunicationProtocol::receive() const { 21 uint8_t length[5]; 22 this->socket.receive(&length, 5); 23 uint32_t* temp32 = (uint32_t*) length; 24 uint32_t length_message = ntohl(*temp32); 25 uint8_t type = length[4]; 26 27 std::vector<uint8_t> buffer(length_message, 0); 28 uint8_t* buf = buffer.data(); 29 this->socket.receive(buf, length_message); 30 return Message(buffer, length_message, type); 31 } 32 33 void CommunicationProtocol::stop() { 34 this->socket.shutdown(SHUT_RDWR); 35 this->socket.close(); 36 } 37 38 CommunicationProtocol::~CommunicationProtocol() = default;</pre>	

jul 21, 20 15:20	Chrono.h	Page 1/1
	<pre> 1 #ifndef CHRONO_H 2 #define CHRONO_H 3 #include <chrono> 4 5 class Chrono { 6 private: 7 std::chrono::high_resolution_clock::time_point time; 8 public: 9 Chrono(); 10 11 //Devuelve la cantidad de milisegundos que hubo entre 2 llamadas a esta 12 //funcion o con la creaci3n del cron3metro. 13 double lap(); 14 15 ~Chrono(); 16 }; 17 18 #endif</pre>	

jul 21, 20 15:20	Chrono.cpp	Page 1/1
	<pre> 1 #include "Chrono.h" 2 3 Chrono::Chrono() { 4 this->time = std::chrono::high_resolution_clock::now(); 5 } 6 7 double Chrono::lap() { 8 std::chrono::high_resolution_clock::time_point now = 9 std::chrono::high_resolution_clock::now(); 10 11 double dt = std::chrono::duration_cast<std::chrono::duration<double>>(now - 12 time).count(); 13 this->time = now; 14 return dt; 15 } 16 Chrono::~Chrono() {} </pre>	

jul 21, 20 15:20	BlockingQueue.h	Page 1/2
	<pre> 1 #ifndef BLOCKING_QUEUE_H 2 #define BLOCKING_QUEUE_H 3 #include <mutex> 4 #include <condition_variable> 5 #include <queue> 6 #include <atomic> 7 #include "Exception.h" 8 9 #define ERRORMSG "No hay elementos en la cola y no se bloquea al realizar el pop" 10 #define CLOSEDQUEUE "La queue se encuentra cerrada" 11 12 //La clase Blocking Queue permite un acceso controlado a los recursos 13 //encolados para que cada uno de los jugadores saquen un numero a la vez. 14 template<typename ITEM> 15 class BlockingQueue { 16 private: 17 std::queue<ITEM> blocking_queue; 18 mutable std::mutex m; 19 std::condition_variable cv; 20 bool blockAtPop; 21 bool isClosed; 22 23 public: 24 //Constructores para la clase Blocking Queue, se permite el constructor 25 //por movimiento, no así por copia dado que no nos interesa que una queue 26 //sea copiada. 27 BlockingQueue(bool blockPop) : blockAtPop(blockPop), isClosed(false){} 28 29 BlockingQueue(BlockingQueue &other) { 30 std::unique_lock<std::mutex> lock(m); 31 this->blocking_queue = std::move(other.blocking_queue); 32 } 33 34 BlockingQueue& operator=(BlockingQueue &other) { 35 std::unique_lock<std::mutex> lock(m); 36 this->blocking_queue = std::move(other.blocking_queue); 37 return *this; 38 } 39 40 //Se realiza el pop del primer elemento de la queue y lo retorna. En caso 41 //de que está vacía, se queda esperando a que aparezca una nueva unidad 42 //del recurso. En caso de que la queue está cerrada lanza una excepción. 43 ITEM pop() { 44 std::unique_lock<std::mutex> lock(m); 45 while(blocking_queue.empty() ^ blockAtPop){ 46 if(isClosed) { 47 throw Exception(CLOSEDQUEUE); 48 } 49 cv.wait(lock); 50 } 51 if (blocking_queue.empty() ^ !blockAtPop) 52 throw Exception(ERRORMSG); 53 ITEM value(std::move(blocking_queue.front())); 54 blocking_queue.pop(); 55 return value; 56 } 57 58 //Se pusha el value en la cola bloqueante. Luego notifica a todos aquellos 59 //que están esperando. 60 void push(ITEM value) { 61 std::unique_lock<std::mutex> lock(m); 62 blocking_queue.push(std::move(value)); 63 cv.notify_all(); 64 } 65 66 bool empty() const { </pre>	

jul 21, 20 15:20	BlockingQueue.h	Page 2/2
67	std::unique_lock<std::mutex> lock(m);	
68	return this->blocking_queue.empty();	
69	}	
70		
71	void close() {	
72	std::unique_lock<std::mutex> lock(m);	
73	isClosed = true;	
74	cv.notify_all();	
75	}	
76		
77	~BlockingQueue(){}	
78		
79	BlockingQueue(const BlockingQueue&) = delete;	
80	BlockingQueue &operator=(const BlockingQueue&) = delete;	
81	};	
82		
83	#endif	

jul 21, 20 15:20	Window.h	Page 1/1
1	#ifndef WINDOW_H	
2	#define WINDOW_H	
3		
4	class SDL_Window;	
5	class SDL_Renderer;	
6	union SDL_Event;	
7		
8	class Window {	
9	private:	
10	SDL_Window* window;	
11	SDL_Renderer* renderer;	
12	bool isMinimized;	
13	int height;	
14	int width;	
15	void init();	
16	const char* title;	
17	public:	
18	Window(const char* title);	
19	Window(const int height,const int width,const char* title);	
20		
21	void clearScreen();	
22		
23	void render();	
24		
25	void handleEvent(SDL_Event& event);	
26		
27	SDL_Renderer& getRenderer() const;	
28		
29	int getWidth() const;	
30	int getHeight() const;	
31		
32	~Window();	
33		
34	Window(const Window&) = delete;	
35	Window &operator=(const Window&) = delete;	
36	};	
37		
38	#endif	

Page 1/3	Window.cpp	Page 1/3
1	#include "Window.h"	
2	#include <exception>	
3	#include <SDL2/SDL.h>	
4	#include <SDL2/SDL_image.h>	
5	#include <SDL2/SDL_mixer.h>	
6	#include <SDL2/SDL_ttf.h>	
7	#include "../common/Exception.h"	
8		
9	#define CHANNELS 2	
10	#define MIX_CHUNKSIZE 1024	
11	#define WINDOW_HEIGHT 600	
12	#define WINDOW_WIDTH 800	
13		
14	Window::Window(const int height, const int width, const char* title) :	
15	isMinimized(false), height(height), width(width), title(title) {	
16		
17	init();	
18		
19	Uint32 flags = SDL_WINDOW_SHOWN SDL_WINDOW_RESIZABLE SDL_RENDERER_ACCELE	
20	TED;	
21	if (SDL_CreateWindowAndRenderer(this->width, this->height, flags, &this->>window	
22	, &this->renderer))	
23	throw Exception("Error with SDL_CreateWindowAndRenderer: %s", SDL_GetError());	
24	SDL_SetWindowTitle(this->>window, this->title);	
25	}	
26	Window::Window(const char* title) : isMinimized(false), height(WINDOW_HEIGHT),	
27	width(WINDOW_WIDTH), title(title) {	
28	init();	
29		
30	Uint32 flags = SDL_WINDOW_SHOWN SDL_WINDOW_RESIZABLE SDL_RENDERER_ACCELE	
31	TED;	
32	if (SDL_CreateWindowAndRenderer(this->width, this->height, flags, &this->>window	
33	, &this->renderer))	
34	throw Exception("Error with SDL_CreateWindowAndRenderer: %s", SDL_GetError());	
35	SDL_SetWindowTitle(this->>window, this->title);	
36	}	
37	void Window::init() {	
38	if (SDL_Init(SDL_INIT_VIDEO SDL_INIT_AUDIO) < 0)	
39	throw Exception("Error with SDL_Init: %s", SDL_GetError());	
40		
41	if (Mix_OpenAudio(MIX_DEFAULT_FREQUENCY, MIX_DEFAULT_FORMAT,	
42	CHANNELS, MIX_CHUNKSIZE) == -1) {	
43	throw Exception("Fail Mix_OpenAudio: %s", Mix_GetError());	
44	}	
45		
46	if (!(IMG_Init(IMG_INIT_PNG) & IMG_INIT_PNG)) {	
47	throw Exception("Fail IMG_Init: %s", IMG_GetError());	
48	}	
49		
50	if (TTF_Init() == -1)	
51	throw Exception("Fail TTF_Init: %s", TTF_GetError());	
52	}	
53		
54	void Window::clearScreen() {	
55	SDL_SetRenderDrawColor(this->renderer, 0x00, 0x00, 0x00, 0x00);	
56	SDL_RenderClear(this->renderer);	
57	}	
58		
59	void Window::handleEvent(SDL_Event& event) {	
60	if(event.type == SDL_WINDOWEVENT) {	
61	switch(event.window.event) {	
62	//Redimensiona el tamaño de la ventana.	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

Page 2/3	Window.cpp	Page 2/3
63	case SDL_WINDOWEVENT_SIZE_CHANGED:	
64	this->width = event.window.data1;	
65	this->height = event.window.data2;	
66	render();	
67	break;	
68	//Repaint on exposure	
69	case SDL_WINDOWEVENT_EXPOSED:	
70	this->isMinimized = false;	
71	render();	
72	break;	
73	//Window minimized	
74	case SDL_WINDOWEVENT_MINIMIZED:	
75	this->isMinimized = true;	
76	break;	
77	//Window maxized	
78	case SDL_WINDOWEVENT_MAXIMIZED:	
79	this->isMinimized = false;	
80	render();	
81	break;	
82	//Window restored	
83	case SDL_WINDOWEVENT_RESTORED:	
84	this->isMinimized = false;	
85	render();	
86	break;	
87	}	
88	}	
89	}	
90		
91	void Window::render() {	
92	if (!this->isMinimized) {	
93	SDL_RenderPresent(this->renderer);	
94	}	
95	}	
96		
97	SDL_Renderer& Window::getRenderer() const {	
98	return *this->renderer;	
99	}	
100		
101	int Window::getWidth() const {	
102	int w, h;	
103	SDL_GL_GetDrawableSize(this->window, &w, &h);	
104	return w;	
105	}	
106		
107		
108	int Window::getHeight() const {	
109	int w, h;	
110	SDL_GL_GetDrawableSize(this->window, &w, &h);	
111	return h;	
112	}	
113		
114	Window::~Window() {	
115	if (this->renderer) {	
116	SDL_DestroyRenderer(this->renderer);	
117	this->renderer = nullptr;	
118	}	
119	if (this->window) {	
120	SDL_DestroyWindow(this->window);	
121	this->window = nullptr;	
122	}	
123		
124	TTF_Quit();	
125	IMG_Quit();	
126	int numtimesopened, frequency, channels;	
127	Uint16 format;	
128	numtimesopened = Mix_QuerySpec(&frequency, &format, &channels);	

101/217

jul 21, 20 15:20	Window.cpp	Page 3/3
129	for (<i>int</i> i = 0; i < numtimesopened; i++)	
130	Mix_CloseAudio();	
131	while (Mix_Init(0))	
132	Mix_Quit();	
133	SDL_Quit();	
134	}	

jul 21, 20 15:20	UnequipButton.h	Page 1/1
1	#ifndef UNEQUIPBUTTON_H	
2	#define UNEQUIPBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class UnequipButton : public RaisedButton {	
8	Player* player;	
9	public :	
10	UnequipButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
11	const TextureManager& manager, Player* player);	
12		
13	virtual bool inside(<i>int</i> x, <i>int</i> y);	
14		
15	virtual void render();	
16		
17	virtual InputInfo onClick(<i>int</i> item);	
18		
19	void setViewport(SDL_Rect viewport);	
20		
21	virtual ~UnequipButton();	
22	};	
23		
24	#endif	

jul 21, 20 15:20	UnequipButton.cpp	Page 1/1
1	#include "UnequipButton.h"	
2		
3	UnequipButton::UnequipButton(SDL_Renderer* renderer, Font& font, std::string text	
4	SDL_Rect position, const TextureManager& manager, Player* player) :	
5	RaisedButton(renderer, font, text, position, manager), player(player) {}	
6		
7	InputInfo UnequipButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→unequipItem(item);	
10	}	
11		
12	bool UnequipButton::inside(int x, int y) {	
13	return RaisedButton::inside(x,y);	
14	}	
15		
16	void UnequipButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void UnequipButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	UnequipButton::~UnequipButton(){}	

mar 21 jul 2020 15:20:01 ART

jul 21, 20 15:20	UI.h	Page 1/2
1	#ifndef UI_H	
2	#define UI_H	
3		
4	#include <SDL2/SDL.h>	
5	#include "../Window.h"	
6	#include "../common/Point.h"	
7	#include "../MusicManager.h"	
8	#include "../Player.h"	
9	#include "../Font.h"	
10	#include "SelectButton.h"	
11	#include "RaisedButton.h"	
12	#include "ArrowButton.h"	
13	#include <vector>	
14	#include "NPCInterface.h"	
15	union SDL_Event;	
16		
17	<i>//Clase encargada de renderizar el estado general de todo el jugador.</i>	
18		
19	class UI {	
20	private:	
21	Player* playerTarget = nullptr; <i>//Jugador del cual debe mostrar estadísticas</i>	
22	Window& window;	
23	const TextureManager& manager;	
24	const MusicManager& mixer;	
25	Font font;	
26	<i>//Interfaz del NPC con el que se está interactuando</i>	
27	std::shared_ptr<NPCInterface> npc{nullptr};	
28		
29	std::vector<SDL_Texture*> texts; <i>//Textos que se muestran en la interfaz</i>	
30	std::vector<SDL_Texture*> info; <i>//Informacion de los stats del jugador</i>	
31	std::vector<std::shared_ptr<RaisedButton>> buttonsInventory; <i>//Botones del i</i>	
32	std::vector<ItemsInventoryID> itemsID; <i>//Id de los items que están disponibles</i>	
33	std::vector<std::shared_ptr<SelectButton>> buttonsItems; <i>//Items mostrados e</i>	
34	std::vector<std::shared_ptr<SelectButton>> buttonsBuild; <i>//Items mostrados e</i>	
35	std::shared_ptr<RaisedButton> unequipButton;	
36		
37	NPCInfo informationNPC;	
38	int widthSegment;	
39	int itemSelected{-1};	
40	int buildSelected{-1};	
41	int maxExpPreviousLevel{0};	
42	int maxExpActualLevel{0};	
43		
44	<i>//Son las funciones encargadas de actualizar el estado general del jugador</i>	
45	void updateStates();	
46	void updateHealth();	
47	void updateMana();	
48	void updateGold();	
49	void updateLevelAndExpirience();	
50	void deleteInfo();	
51		
52	<i>//Actualiza el inventario del jugador, mostrando por pantalla los items y</i>	
53	<i>//botones que este tiene disponible para equiparse, vender o tirar.</i>	
54	void updateInventory();	
55	void updateItems();	
56		
57	<i>//Actualiza el equipamiento del jugador en caso de que este no está interactuando</i>	
58	<i>//En caso de que su estado sea de interactuar, graficar, lo que el profesio</i>	
59	<i>nal</i>	
	<i>//tenga para mostrar y permitira realizar las acciones disponibles.</i>	

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

103/217

jul 21, 20 15:20	UI.h	Page 2/2
60	<code>void updateEquipment();</code>	
61	<code>void updateInteract();</code>	
62	<code>void updateBuild();</code>	
63		
64	<code>void createTexts();</code>	
65		
66	<code>public:</code>	
67	<code>UI(Window& window, Player* player, const TextureManager& manager, const Musi</code>	
68	<code>cManager& mixer);</code>	
69	<code>void render();</code>	
70		
71	<code>//FunciÃ³n llamada para actualizar la informaciÃ³n del NPC con el que se est</code>	
72	<code>Ã;</code>	
73	<code>//interactuando.</code>	
74	<code>void setNPCInfo(NPCInfo info);</code>	
75	<code>//Realiza el manejo de los eventos de click.</code>	
76	<code>//ActualizarÃ; el item seleccionado en el inventario, permitirÃ; realizar la</code>	
77	<code>//acciÃ³n correspondiente a cada botÃ³n o delegarÃ; ese manejo al NPCInterfa</code>	
78	<code>ce</code>	
79	<code>//correspondiente</code>	
80	<code>InputInfo handleClick(SDL_Event& event);</code>	
81	<code>~UI();</code>	
82	<code>};</code>	
83		
84	<code>#endif</code>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	UI.cpp	Page 1/8
1	<code>#include <iostream></code>	
2	<code>#include "UI.h"</code>	
3	<code>#include "EquipButton.h"</code>	
4	<code>#include "DropButton.h"</code>	
5	<code>#include "UnequipButton.h"</code>	
6	<code>#include "PriestInterface.h"</code>	
7	<code>#include "MerchantInterface.h"</code>	
8	<code>#include "BankerInterface.h"</code>	
9	<code>#include "../Effect.h"</code>	
10		
11	<code>#define WIDTHBUTTON 70</code>	
12	<code>#define HEIGHTBUTTON 25</code>	
13	<code>#define ITEMSMERCHANT 12</code>	
14	<code>#define MERCHANTTYPE 1</code>	
15	<code>#define PRIESTTYPE 2</code>	
16	<code>#define BANKERTYPE 3</code>	
17	<code>#define LIMITINVENTORY 9</code>	
18	<code>#define TOPBARHEIGHT 60</code>	
19		
20	<code>UI::UI(Window& window, Player* player, const TextureManager& manager, const Musi</code>	
21	<code>cManager& mixer) :</code>	
22	<code>playerTarget(player),window(window), manager(manager), mixer(mixer),</code>	
23	<code>font("assets/font/Prince Valiant.ttf",18,{0xA4, 0xA4, 0xA4}), texts(), info(),</code>	
24	<code>itemsID(), buttonsBuild() {</code>	
25	<code>createTexts();</code>	
26	<code>SDL_Rect buttonRect;</code>	
27	<code>for (int i = 0; i< LIMITINVENTORY; i++) {</code>	
28	<code>buttonRect = {9+(i*3)*50,50+(i/3)*50,32,32};</code>	
29	<code>std::shared_ptr<SelectButton> button = std::shared_ptr<SelectButton>(new</code>	
30	<code>SelectButton(&(window.getRenderer()),buttonRect,manager,i));</code>	
31	<code>this->buttonsItems.push_back(button);</code>	
32	<code>} std::shared_ptr<RaisedButton> equipButton = std::shared_ptr<RaisedButton>(ne</code>	
33	<code>w EquipButton(&(window.getRenderer()),font,</code>	
34	<code>"Equipar",{9,205,WIDTHBUTTON,HEIGHTBUTTON}, manager,playerTarget));</code>	
35	<code>std::shared_ptr<RaisedButton> dropButton = std::shared_ptr<RaisedButton>(new</code>	
36	<code>DropButton(&(window.getRenderer()),font,"Tirar",{109,205,WIDTHBUTTON,HEIG</code>	
37	<code>THBUTTON}, manager,playerTarget));</code>	
38	<code>this->buttonsInventory.push_back(equipButton);</code>	
39	<code>this->buttonsInventory.push_back(dropButton);</code>	
40	<code>this->informationNPC.type = 0;</code>	
41	<code>} void UI::updateHealth(){</code>	
42	<code>int w, h,width_text,height_text;</code>	
43	<code>SDL_QueryTexture(this->texts[0], NULL, NULL, &w, &h);</code>	
44		
45	<code>int playerHealth = this->playerTarget->getHealth();</code>	
46	<code>int playerMaxHealth = this->playerTarget->getMaxHealth();</code>	
47	<code>int health = (playerHealth * (widthSegment-2))/playerMaxHealth;</code>	
48	<code>std::string life = "("+ std::to_string(playerHealth) + "/" + std::to_string(</code>	
49	<code>playerMaxHealth) + ")";</code>	
50	<code>SDL_Texture* lifeTexture = font.createText(life,&(window.getRenderer()), &wi</code>	
51	<code>dth_textT, &height_text);</code>	
52	<code>this->info.push_back(lifeTexture);</code>	
53	<code>SDL_Rect healthText = {9+widthSegment*0,10,w,h};</code>	
54	<code>SDL_Rect lifeRect = {12+widthSegment*0+w,10,width_text,height_text};</code>	
55	<code>SDL_RenderCopy(&(window.getRenderer()), this->texts[0], NULL, &healthText);</code>	
56	<code>SDL_RenderCopy(&(window.getRenderer()), this->info[0],NULL,&lifeRect);</code>	
57	<code>SDL_Rect healthRect = {9+widthSegment*0,30,widthSegment,20};</code>	
58	<code>SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0xA4, 0xA4, 0xAA</code>	
	<code>);</code>	

104/217

jul 21, 20 15:20	UI.cpp	Page 2/8
59	SDL_RenderDrawRect(&(this->window.getRenderer()), &healthRect);	
60	SDL_Rect fillHealth = {10+widthSegment*0,31, health,18};	
61	SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0xFF, 0x00, 0x00, 0xFF);	
62	SDL_RenderFillRect(&(this->window.getRenderer()), &fillHealth);	
63	}	
64		
65	void UI::updateMana(){	
66	int w, h,width_text,height_text;	
67	SDL_QueryTexture(this->texts[1], NULL, NULL, &w, &h);	
68		
69	int playerMana = this->playerTarget->getMana();	
70	int playerMaxMana = this->playerTarget->getMaxMana();	
71	int mana = playerMaxMana > 0 ? (playerMana * (widthSegment-2))/playerMaxMana	
72	: 0;	
73	std::string manaTotal = "(" + std::to_string(playerMana) + "/" + std::to_string(playerMaxMana) + ")";	
74	SDL_Texture* manaTexture = font.createText(manaTotal,&(window.getRenderer()), &width_text, &height_text);	
75	this->info.push_back(manaTexture);	
76		
77	SDL_Rect manaText = {9+widthSegment*2,10,w,h};	
78	SDL_Rect manaTotalRect = {12+widthSegment*2+w,10,width_text,height_text};	
79	SDL_RenderCopy(&(window.getRenderer()), this->texts[1], NULL, &manaText);	
80	SDL_RenderCopy(&(window.getRenderer()), this->info[1],NULL,&manaTotalRect);	
81		
82	SDL_Rect manaRect = {9+widthSegment*2,30,widthSegment,20};	
83	SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0xA4, 0xA4, 0xA4, 0xAA	
84);	
85	SDL_RenderDrawRect(&(this->window.getRenderer()), &manaRect);	
86		
87	SDL_Rect fillMana = {10+widthSegment*2,31, mana,18};	
88	SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0x01, 0xDF, 0xD7, 0xFF);	
89	SDL_RenderFillRect(&(this->window.getRenderer()), &fillMana);	
90	}	
91	void UI::updateGold() {	
92	int width_text,height_text;	
93	const Texture& goldTexture = manager.getTexture(TextureID::Gold);	
94		
95	int safeGold = this->playerTarget->getSafeGold();	
96	int actualGold = this->playerTarget->getGold();	
97	std::string gold = std::to_string(actualGold) + "/" + std::to_string(safeGold);	
98	SDL_Texture* goldText = font.createText(gold,&(window.getRenderer()), &width_text, &height_text);	
99	this->info.push_back(goldText);	
100		
101	SDL_Rect srcGold = {0,0,32,32};	
102	SDL_Rect dstGold = {widthSegment*4,10,32,32};	
103	goldTexture.render(srcGold,dstGold);	
104		
105	SDL_Rect goldRect = {36+widthSegment*4,10,width_text,height_text};	
106	SDL_RenderCopy(&(window.getRenderer()), this->info[2],NULL,&goldRect);	
107	}	
108		
109	void UI::updateLevelAndExpirience() {	
110	int w, h,width_text,height_text;	
111	SDL_QueryTexture(this->texts[2], NULL, NULL, &w, &h);	
112		
113	const Texture& star = manager.getTexture(TextureID::Star);	
114	SDL_Rect srcStar = {0,0,15,16};	
115	SDL_Rect dstStar = {20+widthSegment*5,15,15,16};	
116	star.render(srcStar,dstStar);	

jul 21, 20 15:20	UI.cpp	Page 3/8
117	std::string level = std::to_string(this->playerTarget->getLevel());	
118	SDL_Texture* levelTexture = font.createText(level,&(window.getRenderer()), &width_text, &height_text);	
119	this->info.push_back(levelTexture);	
120		
121		
122	SDL_Rect levelText = {50+widthSegment*5,10,w,h};	
123	SDL_Rect levelRect = {50+w+widthSegment*5,10,width_text,height_text};	
124	SDL_RenderCopy(&(window.getRenderer()), this->texts[2], NULL, &levelText);	
125	SDL_RenderCopy(&(window.getRenderer()), this->info[3],NULL,&levelRect);	
126		
127	int playerExp = this->playerTarget->getExp();	
128	int playerMaxExp = this->playerTarget->getMaxExp();	
129	int exp = ((playerExp-this->maxExpPreviousLevel) * (widthSegment-2))/(playerMaxExp-this->maxExpPreviousLevel);	
130	std::string expTotal = "(" + std::to_string(playerExp) + "/" + std::to_string(playerMaxExp) + ")";	
131	SDL_Texture* expTexture = font.createText(expTotal,&(window.getRenderer()), &width_text, &height_text);	
132	this->info.push_back(expTexture);	
133		
134	SDL_QueryTexture(this->texts[3], NULL, NULL, &w, &h);	
135	SDL_Rect expText = {25+widthSegment*6,8,w,h};	
136	SDL_Rect expTotalRect = {25+widthSegment*6+w,8,width_text,height_text};	
137	SDL_RenderCopy(&(window.getRenderer()), this->texts[3], NULL, &expText);	
138	SDL_RenderCopy(&(window.getRenderer()), this->info[4],NULL,&expTotalRect);	
139		
140	SDL_Rect expRect = {25+widthSegment*6,30,widthSegment,20};	
141	SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0xA4, 0xA4, 0xA4, 0xAA	
142);	
143	SDL_RenderDrawRect(&(this->window.getRenderer()), &expRect);	
144		
145	SDL_Rect fillExp = {26+widthSegment*6,31, exp,18};	
146	SDL_SetRenderDrawColor(&(this->window.getRenderer()), 0x00, 0xFF, 0x00, 0xFF);	
147	SDL_RenderFillRect(&(this->window.getRenderer()), &fillExp);	
148		
149	if (this->maxExpActualLevel != playerMaxExp) {	
150	this->maxExpPreviousLevel = this->maxExpActualLevel;	
151	this->maxExpActualLevel = playerMaxExp;	
152	}	
153		
154		
155	void UI::deleteInfo(){	
156	for (auto& text: info) {	
157	font.deleteText(text);	
158	this->info.pop_back();	
159	}	
160		
161		
162	void UI::updateStates() {	
163	const Texture& topBar = manager.getTexture(TextureID::TopBar);	
164	SDL_Rect topBarViewport = {0,0,this->window.getWidth(),TOPBARHEIGHT};	
165	SDL_RenderSetViewport(&(this->window.getRenderer()), &topBarViewport);	
166	SDL_RenderCopy(&(this->window.getRenderer()), topBar.getTexture(), NULL, NULL	
167	L);	
168	deleteInfo();	
169	updateHealth();	
170	updateMana();	
171	updateGold();	
172	updateLevelAndExpirience();	
173	}	
174		
175	void UI::updateItems() {	
176	itemsID.clear();	

jul 21, 20 15:20	UI.cpp	Page 4/8
176	int w,h;	
177	SDL_QueryTexture(this →texts[4], NULL, NULL, &w, &h);	
178	SDL_Rect invText = {15,15,w,h};	
179	SDL_RenderCopy(&(window.getRenderer()), this →texts[4], NULL, &invText);	
180		
181	std::string items = this →playerTarget→getInventory();	
182	std::string item;	
183	int idItem;	
184		
185	SDL_Rect src = {0,0,52,52};	
186	SDL_Rect dst;	
187	for (int i = 0; i < LIMITINVENTORY; i++) {	
188	item = items.substr(2*i,i,2);	
189	idItem = std::stoi(item);	
190	const Texture& item = manager.getTexture((ItemsInventoryID)idItem);	
191	itemsID.push_back((ItemsInventoryID)idItem);	
192		
193	buttonsItems[i]→setViewport({0,TOPBARHEIGHT,widthSegment*2,(this →windo	
194	w.getHeight()-TOPBARHEIGHT)/2});	
195	dst = {35+(i*3)*widthSegment/2,50+(i/3)*((this →window.getHeight()-TOPBA	
196	RHEIGHT)/2)/5,widthSegment/3,widthSegment/3};	
197	buttonsItems[i]→updatePosition(dst);	
198	if (idItem > 0) {	
199	buttonsItems[i]→render();	
200	item.render(src,dst);	
201	}	
202		
203	void UI::updateInventory() {	
204	const Texture& statBackground = manager.getTexture(TextureID::StatsBackgroun	
205	d);	
206	SDL_Rect inventoryViewPort = {0,TOPBARHEIGHT,widthSegment*2,(this →window.ge	
207	tHeight()-TOPBARHEIGHT)/2};	
208	SDL_RenderSetViewport(&(this→window.getRenderer()), &inventoryViewPort);	
209	SDL_Rect dst = {0,0,widthSegment*2,(this→window.getHeight()-TOPBARHEIGHT)/2	
210	};	
211	SDL_RenderCopy(&(this→window.getRenderer()), statBackground.getTexture(), N	
212	ULL, &dst);	
213	updateItems();	
214	int i = 0;	
215	for (auto& button: buttonsInventory){	
216	button→setViewport({0,TOPBARHEIGHT,widthSegment*2,(this→window.getHeig	
217	ht()-TOPBARHEIGHT)/2});	
218	button→updatePosition({20+(i*2)*widthSegment,int(((this→window.getHeig	
219	ht()-TOPBARHEIGHT)/2)-HEIGHTBUTTON*2.5),WIDTHBUTTON,HEIGHTBUTTON});	
220	button→render();	
221	i++;	
222	}	
223		
224	void UI::updateBuild() {	
225	int w,h;	
226	SDL_QueryTexture(this →texts[5], NULL, NULL, &w, &h);	
227	SDL_Rect equipment = {15,15,w,h};	
228	SDL_RenderCopy(&(window.getRenderer()), this →texts[5], NULL, &equipment);	
229		
230	PlayerInfo info = this →playerTarget→getInfo();	
231	const Texture& itemBody = manager.getTexture(info.getBodyID());	
232	const Texture& itemHead = manager.getTexture(info.getHeadID());	
233	const Texture& itemHelmet = manager.getTexture(info.getHelmetID());	
234	const Texture& itemWeapon = manager.getTexture(info.getWeaponID());	
235	const Texture& itemShield = manager.getTexture(info.getShieldID());	
236		
237	if (this →buttonsBuild.size() == 0) {	
238	std::shared_ptr<SelectButton> button;	

jul 21, 20 15:20	UI.cpp	Page 5/8
234	button = std::shared_ptr<SelectButton>(new SelectButton(
235	&(window.getRenderer()),{widthSegment-widthSegment/6,50,widthSegment	
236	/3,widthSegment/3},manager,0));	
237	this →buttonsBuild.push_back(button);	
238	button = std::shared_ptr<SelectButton>(new SelectButton(
239	&(window.getRenderer()),{widthSegment-(widthSegment/3)*2,140,widthSe	
240	gment/3,widthSegment/3},manager,1));	
241	this →buttonsBuild.push_back(button);	
242	button = std::shared_ptr<SelectButton>(new SelectButton(
243	&(window.getRenderer()),{widthSegment+widthSegment/3,140,widthSegmen	
244	t/3,widthSegment/3},manager,2));	
245	this →buttonsBuild.push_back(button);	
246	this →unequipButton = std::shared_ptr<RaisedButton> (new UnequipButton(
247	&(window.getRenderer()),font,"Guardar",{9,205,WIDTHBUTTON,HEIGHTBUTTO	
248	N}, manager,playerTarget));	
249		
250	for(uint i = 0; i < this →buttonsBuild.size(); i++){	
251	buttonsBuild[i]→setViewport({0,(this→window.getHeight())/2,widthSegmen	
252	t*2, (this→window.getHeight())/2});	
253	if (i == 0) {	
254	buttonsBuild[i]→updatePosition({widthSegment-widthSegment/6,50,width	
255	hSegment/3,widthSegment/3});	
256	} else if (i == 1) {	
257	buttonsBuild[i]→updatePosition({widthSegment-(widthSegment/3)*2,140	
258	,widthSegment/3,widthSegment/3});	
259	} else if (i == 2) {	
260	buttonsBuild[i]→updatePosition({widthSegment+widthSegment/3,140,wid	
261	thSegment/3,widthSegment/3});	
262	buttonsBuild[i]→render();	
263		
264	SDL_Rect src = {0,0,52,52};	
265	itemHelmet.render(src,{widthSegment-widthSegment/6,50,widthSegment/3,widthSe	
266	gment/3});	
267	itemBody.render(src,{widthSegment-widthSegment/3,117+widthSegment/5,(widthSe	
268	gment/3)*2,(widthSegment/3)*2});	
269	if (this →playerTarget→getHealth() == 0) {	
270	itemHead.render({0,0,52,52},{widthSegment-widthSegment/6,85+widthSegment	
271	/6,widthSegment/3,widthSegment/3});	
272	} else {	
273	itemHead.render({0,0,17,17},{widthSegment-widthSegment/6,85+widthSegment	
274	/6,widthSegment/3,widthSegment/3});	
275	itemWeapon.render(src,{widthSegment-(widthSegment/3)*2,140,widthSegment/3,wi	
276	dthSegment/3});	
277	itemShield.render(src,{widthSegment+widthSegment/3,140,widthSegment/3,widthS	
278	egment/3});	
279	this →unequipButton→setViewport({0,(this→window.getHeight())/2,widthSegment	
280	*2, (this→window.getHeight())/2});	
281	this →unequipButton→updatePosition({20,int(((this→window.getHeight())/2)-HE	
282	IGHTBUTTON*2.5),WIDTHBUTTON,HEIGHTBUTTON});	
283	this →unequipButton→render();	
284		
285		
286		
287		
288	void UI::updateInteract() {	
289	if (this →npc == nullptr) {	
290	if (this →informationNPC.type == MERCHANTTYPE) {	
291	this →npc = std::shared_ptr<NPCInterface>(new MerchantInterface(info	
292	rmationNPC,&window,manager,playerTarget));	
293	const Effect& effect = mixer.getEffect(MusicID::Merchant);	

jul 21, 20 15:20	UI.cpp	Page 6/8
283	effect.playEffect();	
284	else if (this->informationNPC.type == PRIESTTYPE) {	
285	this->npc = std::shared_ptr<NPCInterface>(new PriestInterface(inform	
286	ationNPC,&window,manager,playerTarget));	
287	const Effect& effect = mixer.getEffect(MusicID::Priest);	
288	effect.playEffect();	
289	else if (this->informationNPC.type == BANKERTYPE) {	
290	this->npc = std::shared_ptr<NPCInterface>(new BankerInterface(inform	
291	ationNPC,&window,manager,playerTarget));	
292	const Effect& effect = mixer.getEffect(MusicID::Banker);	
293	effect.playEffect();	
294	else {	
295	updateBuild();	
296	} else {	
297	npc->update(this->informationNPC);	
298	if (this->npc != nullptr)	
299	this->npc->render();	
300	}	
301		
302	void UI::updateEquipment() {	
303	const Texture& statBackground = manager.getTexture(TextureID::StatsBackgroun	
304	d);	
305	SDL_Rect equipmentViewPort = {0,(this->window.getHeight())/2,widthSegment*2,	
306	(this->window.getHeight())/2};	
307	SDL_RenderSetViewport(&(this->window.getRenderer()), &equipmentViewPort);	
308	SDL_Rect dst = {0,0,widthSegment*2,(this->window.getHeight())/2};	
309	SDL_RenderCopy(&(this->window.getRenderer()), statBackground.getTexture(), N	
310	ULL, &dst);	
311	if (this->playerTarget->getState() == CharacterStateID::Interact) {	
312	updateInteract();	
313	this->buttonsBuild.clear();	
314	this->unequipButton = nullptr;	
315	this->buildSelected = -1;	
316	else {	
317	this->npc = nullptr;	
318	this->informationNPC.type = 0;	
319	updateBuild();	
320	}	
321	void UI::render() {	
322	this->widthSegment = this->window.getWidth()/8;	
323	updateInventory();	
324	updateEquipment();	
325	updateStates();	
326	}	
327	void UI::setNPCInfo(NPCInfo info) {	
328	this->informationNPC = info;	
329	}	
330		
331	InputInfo UI::handleClick(SDL_Event& event) {	
332	int x,y;	
333	InputInfo info;	
334	info.input = InputID::nothing;	
335	bool newItemSelected = false;	
336	if (event.type == SDL_MOUSEMOTION ∨ event.type == SDL_MOUSEBUTTONDOWN ∨ event.	
337	type == SDL_MOUSEBUTTONUP)	
338	SDL_GetMouseState(&x, &y);	
339	switch(event.type) {	
340	case SDL_MOUSEBUTTONDOWN:	
341	for(auto& button: buttonsItems){	
342	newItemSelected = button->inside(x,y);	
	if (newItemSelected) {	

jul 21, 20 15:20	UI.cpp	Page 7/8
343	if(itemSelected == -1) {	
344	itemSelected = button->getId();	
345	} else if (itemSelected == button->getId()) {	
346	itemSelected = -1;	
347	} else if (itemSelected != button->getId()) {	
348	buttonsItems[itemSelected]->onClick();	
349	itemSelected = button->getId();	
350	}	
351	}	
352	for (auto& button: buttonsInventory) {	
353	if (button->inside(x,y) ^ itemSelected != -1)	
354	info = button->onClick(itemSelected+1);	
355	}	
356	for(auto& button: buttonsBuild){	
357	newItemSelected = button->inside(x,y);	
358	if (newItemSelected) {	
359	if(buildSelected == -1) {	
360	buildSelected = button->getId();	
361	} else if (buildSelected == button->getId()) {	
362	buildSelected = -1;	
363	} else if (buildSelected != button->getId()) {	
364	buttonsBuild[buildSelected]->onClick();	
365	buildSelected = button->getId();	
366	}	
367	}	
368	}	
369		
370	if (this->npc != nullptr) {	
371	int itemId = 0;	
372	if (itemSelected != -1)	
373	itemId = int(itemsID[itemSelected]);	
374	info = this->npc->handleClick(x,y,itemId);	
375	} else if (this->unequipButton != nullptr) {	
376	if (this->unequipButton->inside(x,y) ^ buildSelected > -1)	
377	info = this->unequipButton->onClick(buildSelected);	
378	break;	
379	}	
380		
381		
382	}	
383	return info;	
384	}	
385		
386	void UI::createTexts() {	
387	int width_text, height_text;	
388	SDL_Texture* health = font.createText("Vida",	
389	&(window.getRenderer()), &width_text, &height_text);	
390	this->texts.push_back(health);	
391	SDL_Texture* mana = font.createText("Mana",	
392	&(window.getRenderer()), &width_text, &height_text);	
393	this->texts.push_back(mana);	
394	SDL_Texture* level = font.createText("Nivel",	
395	&(window.getRenderer()), &width_text, &height_text);	
396	this->texts.push_back(level);	
397	SDL_Texture* expirience = font.createText("Experiencia",	
398	&(window.getRenderer()), &width_text, &height_text);	
399	this->texts.push_back(expirience);	
400	SDL_Texture* inventory = font.createText("Inventario",	
401	&(window.getRenderer()), &width_text, &height_text);	
402	this->texts.push_back(inventory);	
403	SDL_Texture* equipment = font.createText("Equipacion",	
404	&(window.getRenderer()), &width_text, &height_text);	
405	this->texts.push_back(equipment);	
406	}	
407		
408	UI::~UI() {	

jul 21, 20 15:20	UI.cpp	Page 8/8
409	for (auto & text: texts) {	
410	font.deleteText(text);	
411	}	
412	}	

mar 21 jul 2020 15:20:01 ART

jul 21, 20 15:20	SellButton.h	Page 1/1
1	#ifndef SELLBUTTON_H	
2	#define SELLBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class SellButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	SellButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
12	const TextureManager& manager, Player* player);	
13		
14	virtual bool inside(int x, int y);	
15		
16	virtual void render();	
17		
18	virtual InputInfo onClick(int item);	
19	void setViewport(SDL_Rect viewport);	
20		
21	virtual ~SellButton();	
22		
23	};	
24		
25	#endif	

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

108/217

jul 21, 20 15:20	SellButton.cpp	Page 1/1
1	#include "SellButton.h"	
2		
3	SellButton::SellButton(SDL_Renderer* renderer, Font& font, std::string text,	
4	SDL_Rect position, const TextureManager& manager, Player* player) :	
5	RaisedButton(renderer, font, text, position, manager), player(player) {}	
6		
7	InputInfo SellButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→sell(item);	
10	}	
11		
12	bool SellButton::inside(int x, int y) {	
13	return RaisedButton::inside(x,y);	
14	}	
15		
16	void SellButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void SellButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	SellButton::~SellButton(){}	

jul 21, 20 15:20	SelectButton.h	Page 1/1
1	#ifndef SELECTBUTTON_H	
2	#define SELECTBUTTON_H	
3		
4	#include "Button.h"	
5		
6	class SelectButton: public Button {	
7	private:	
8	int id;	
9	public:	
10	SelectButton(SDL_Renderer* renderer, SDL_Rect position, const TextureManager&	
11	manager, int id);	
12	virtual bool inside(int x, int y);	
13		
14	virtual void render();	
15		
16	void onClick();	
17		
18	int getId();	
19		
20	virtual void updatePosition(SDL_Rect position);	
21		
22	void setViewport(SDL_Rect viewport);	
23		
24	~SelectButton();	
25	};	
26		
27	#endif	

jul 21, 20 15:20	SelectButton.cpp	Page 1/1
1	#include "SelectButton.h"	
2		
3	SelectButton::SelectButton(SDL_Renderer* renderer, SDL_Rect position, const TextureManager& manager, int id) :	
4	Button(renderer,position,manager), id(id){}	
5		
6	void SelectButton::render(){	
7	int w,h;	
8	const Texture& background = manager.getTexture(TextureID::Button);	
9	SDL_QueryTexture(background.getTexture(), NULL , NULL , &w, &h);	
10	SDL_Rect src = {0,0,w,h};	
11	background.render(src, this →button);	
12	if (this →clicked) {	
13	SDL_Rect click = {button.x-1,button.y-1,button.w+2,button.h+2};	
14	SDL_SetRenderDrawColor(this →renderer, 0xA4, 0xA4, 0xA4, 0xAA);	
15	SDL_RenderDrawRect(this →renderer, &click);	
16	}	
17	if (this →texture ≠ nullptr){	
18	SDL_QueryTexture(this →texture, NULL , NULL , &w, &h);	
19	src = {0,0,w,h};	
20	SDL_Rect dest = {button.x+w/4, button.y,w,h};	
21	SDL_RenderCopy(this →renderer, this →texture,&src,&dest);	
22	}	
23	}	
24		
25	void SelectButton::onClick() {	
26	this →clicked = !this →clicked;	
27	}	
28		
29	void SelectButton::updatePosition(SDL_Rect position) {	
30	Button::updatePosition(position);	
31	}	
32		
33	bool SelectButton::inside(int x, int y) {	
34	bool inside = true ;	
35	if (x < this →button.x+viewport.x)	
36	inside = false ;	
37	if (x > this →button.x+ this →button.w+viewport.x)	
38	inside = false ;	
39	if (y < this →button.y+viewport.y)	
40	inside = false ;	
41	if (y > this →button.y + this →button.h+viewport.y)	
42	inside = false ;	
43	if (inside)	
44	onClick();	
45	return inside;	
46	}	
47		
48	int SelectButton::getId() {	
49	return this →id;	
50	}	
51		
52	void SelectButton::setViewport(SDL_Rect viewport) {	
53	this →viewport = viewport;	
54	}	
55		
56	SelectButton::~SelectButton(){} 	

jul 21, 20 15:20	RetireItemButton.h	Page 1/1
1	#ifndef RETIREITEMBUTTON_H	
2	#define RETIREITEMBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class RetireItemButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	RetireItemButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
12	const TextureManager& manager, Player* player);	
13		
14	virtual bool inside(int x, int y);	
15		
16	virtual void render();	
17		
18	virtual InputInfo onClick(int item);	
19		
20	void setViewport(SDL_Rect viewport);	
21		
22	virtual ~RetireItemButton ();	
23	};	
24		
25	#endif	

jul 21, 20 15:20	RetireItemButton.cpp	Page 1/1
1	#include "RetireItemButton.h"	
2		
3	RetireItemButton::RetireItemButton(SDL_Renderer* renderer, Font& font, std::string text,	
4	SDL_Rect position, const TextureManager& manager, Player* player) :	
5	RaisedButton(renderer, font, text, position, manager), player(player) {}	
6		
7	InputInfo RetireItemButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→retire(item, true);	
10	}	
11		
12	bool RetireItemButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void RetireItemButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void RetireItemButton::setViewport(SDL_Rect viewport) {	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	RetireItemButton::~RetireItemButton() {}	

jul 21, 20 15:20	RetireGoldButton.h	Page 1/1
1	#ifndef RETIREGOLDBUTTON_H	
2	#define RETIREGOLDBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class RetireGoldButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	RetireGoldButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
12	const TextureManager& manager, Player* player);	
13		
14	virtual bool inside(int x, int y);	
15		
16	virtual void render();	
17		
18	virtual InputInfo onClick(int item);	
19		
20	void setViewport(SDL_Rect viewport);	
21		
22	virtual ~RetireGoldButton();	
23	};	
24		
25	#endif	

jul 21, 20 15:20	RetireGoldButton.cpp	Page 1/1
1	#include "RetireGoldButton.h"	
2		
3	RetireGoldButton::RetireGoldButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position, const TextureManager& manager, Player* player) :	
4	RaisedButton(renderer, font, text, position, manager), player(player) {}	
5		
6		
7	InputInfo RetireGoldButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→retire(item, false);	
10	}	
11		
12	bool RetireGoldButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void RetireGoldButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void RetireGoldButton::setViewport(SDL_Rect viewport) {	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	RetireGoldButton::~RetireGoldButton() {}	

jul 21, 20 15:20	ResurrectButton.h	Page 1/1
1	#ifndef RESURRECTBUTTON_H	
2	#define RESURRECTBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class ResurrectButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	ResurrectButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
12	const TextureManager& manager, Player* player);	
13		
14	virtual bool inside(int x, int y);	
15		
16	virtual void render();	
17		
18	virtual InputInfo onClick(int item);	
19		
20	void setViewport(SDL_Rect viewport);	
21		
22	virtual ~ResurrectButton();	
23	};	
24		
25	#endif	

jul 21, 20 15:20	ResurrectButton.cpp	Page 1/1
1	#include "ResurrectButton.h"	
2		
3	ResurrectButton::ResurrectButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position, const TextureManager& manager, Player* player) :	
4	RaisedButton(renderer, font, text, position, manager), player(player) {}	
5		
6		
7	InputInfo ResurrectButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→resurrect();	
10	}	
11		
12	bool ResurrectButton::inside(int x, int y) {	
13	return RaisedButton::inside(x,y);	
14	}	
15		
16	void ResurrectButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void ResurrectButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	ResurrectButton::~ResurrectButton(){}	

jul 21, 20 15:20	RaisedButton.h	Page 1/1
1	#ifndef RAISEDBUTTON_H	
2	#define RAISEDBUTTON_H	
3		
4	#include "Button.h"	
5		
6	class RaisedButton : public Button {	
7	public:	
8	RaisedButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position, const TextureManager& manager);	
9		
10	virtual bool inside(int x, int y);	
11		
12	virtual void render();	
13		
14	virtual InputInfo onClick(int item) = 0;	
15		
16	void setViewport(SDL_Rect viewport);	
17		
18	virtual void updatePosition(SDL_Rect position);	
19		
20	virtual ~RaisedButton();	
21		
22	};	
23		
24	#endif	

jul 21, 20 15:20	RaisedButton.cpp	Page 1/1
1	#include "RaisedButton.h"	
2		
3	RaisedButton::RaisedButton(SDL_Renderer* renderer, Font& font, std::string text,	
4	SDL_Rect position, const TextureManager& manager) {	
5	Button(renderer, font, text, position, manager) {}	
6		
7	void RaisedButton::render(){	
8	int w,h;	
9	const Texture& background = manager.getTexture(TextureID::Button);	
10	SDL_QueryTexture(background.getTexture(), NULL , NULL , &w, &h);	
11	SDL_Rect src = {0,0,w,h};	
12	background.render(src, this →button);	
13	if (this →clicked) {	
14	SDL_Rect click = {button.x-1, button.y-1, button.w+2, button.h+2};	
15	SDL_SetRenderDrawColor(this →renderer, 0xA4, 0xA4, 0xA4, 0xAA);	
16	SDL_RenderDrawRect(this →renderer, &click);	
17	this →clicked = !this →clicked;	
18		
19	if (this →texture != nullptr){	
20	SDL_QueryTexture(this →texture, NULL , NULL , &w, &h);	
21	src = {0,0,w,h};	
22	SDL_Rect dest = {button.x*(w/6), button.y,w,h};	
23	SDL_RenderCopy(this →renderer, this →texture, &src, &dest);	
24		
25	}	
26		
27	void RaisedButton::updatePosition(SDL_Rect position) {	
28	Button::updatePosition(position);	
29	}	
30		
31	bool RaisedButton::inside(int x, int y) {	
32	bool inside = true ;	
33	if (x < this →button.x+viewport.x)	
34	inside = false ;	
35	if (x > this →button.x+ this →button.w+viewport.x)	
36	inside = false ;	
37	if (y < this →button.y+viewport.y)	
38	inside = false ;	
39	if (y > this →button.y + this →button.h+viewport.y)	
40	inside = false ;	
41	return inside;	
42	}	
43		
44	void RaisedButton::setViewport(SDL_Rect viewport){	
45	this →viewport = viewport;	
46	}	
47		
48	RaisedButton::~RaisedButton(){}	

jul 21, 20 15:20	PriestInterface.h	Page 1/1
1	#ifndef PRIESTINTERFACE_H	
2	#define PRIESTINTERFACE_H	
3		
4	#include "NPCInterface.h"	
5	#include <vector>	
6	#include <memory>	
7	#include "RaisedButton.h"	
8	#include "SelectButton.h"	
9		
10	class PriestInterface: public NPCInterface {	
11	private :	
12	std::vector<std::shared_ptr<RaisedButton>> buttonsNPC; //Botones que nos mue	
13	stra el NPC	
14	std::vector<std::shared_ptr<SelectButton>> buttonsItemsNPC; //Items que nos	
15	muestra el NPC	
16	std::vector<SDL_Texture*> gold; //El oro que se salen cada uno de los items	
17	std::vector<ItemsInventoryID> itemsPriest;	
18	int itemSelectedNPC{-1};	
19		
20	void deleteGoldValues();	
21	public :	
22	explicit PriestInterface(NPCInfo info, Window* window, const TextureManager&	
23	manager, Player* player);	
24		
25	virtual void render();	
26		
27	virtual InputInfo handleClick(int x, int y, int itemSelected);	
28		
29	~PriestInterface();	
30		
31	};	
32		
33	#endif	

Page 1/3	PriestInterface.cpp	Page 1/3
1	<code>#include "PriestInterface.h"</code>	
2		
3	<code>#include <memory></code>	
4		
5	<code>#define WIDTHBUTTON 70</code>	
6	<code>#define HEIGHTBUTTON 25</code>	
7		
8	<code>PriestInterface::PriestInterface(NPCInfo info, Window* window, const TextureManager& manager, Player* player) :</code>	
9	<code> NPCInterface(info, window, manager, player), buttonsNPC(), buttonsItemsNPC(), go</code>	
10	<code>ld() {</code>	
11	<code> int width_text, height_text;</code>	
12	<code> SDL_Texture* priest = font.createText("Sacerdote",</code>	
13	<code> &(window->getRenderer()), &width_text, &height_text);</code>	
14	<code> this->texture = priest;</code>	
15	<code>}</code>	
16	<code>void PriestInterface::render() {</code>	
17	<code> int w,h;</code>	
18	<code> int i = 0;</code>	
19	<code> int width = 0;</code>	
20	<code> int height = 0;</code>	
21	<code> SDL_QueryTexture(this->texture, NULL, NULL, &w, &h);</code>	
22	<code> SDL_Rect priest = {15,15,w,h};</code>	
23	<code> SDL_RenderCopy(&(window->getRenderer()), this->texture, NULL, &priest);</code>	
24		
25	<code> SDL_Rect src;</code>	
26	<code> SDL_Rect dst;</code>	
27	<code> std::shared_ptr<RaisedButton> button;</code>	
28	<code> std::shared_ptr<SelectButton> selection;</code>	
29	<code> SDL_Texture* textureGold;</code>	
30	<code> bool loadButtons = false;</code>	
31		
32	<code> std::vector<std::pair<ItemsInventoryID, uint>> items(information.items.begin</code>	
33	<code> (), information.items.end());</code>	
34	<code> if (this->itemsPriest.size() == 0) {</code>	
35	<code> for (auto iter: items) {</code>	
36	<code> this->itemsPriest.push_back(iter.first);</code>	
37	<code> }</code>	
38	<code> }</code>	
39		
40	<code> if (this->buttonsItemsNPC.size() == 0)</code>	
41	<code> loadButtons = true;</code>	
42		
43	<code> int widthSegment = this->window->getWidth()/8;</code>	
44		
45	<code> for (uint j = 0; j < items.size(); j++) {</code>	
46	<code> const Texture& item = manager.getTexture(items[j].first);</code>	
47	<code> src = {0,0,52,52};</code>	
48	<code> dst = {20+(i%3)*widthSegment/2,50+(i/3)*((this->window->getHeight())/2)/</code>	
49	<code>5,widthSegment/3,widthSegment/3};</code>	
50	<code> if (loadButtons) {</code>	
51	<code> selection = std::make_shared<SelectButton>(&(window->getRenderer()),</code>	
52	<code> dst,manager,j);</code>	
53	<code> this->buttonsItemsNPC.push_back(selection);</code>	
54	<code> }</code>	
55	<code> this->buttonsItemsNPC[i]->setViewport({0,(this->window->getHeight())/2,(t</code>	
56	<code>his->window->getWidth()/8)*2,(this->window->getHeight())/2});</code>	
57	<code> this->buttonsItemsNPC[i]->updatePosition(dst);</code>	
58	<code> this->buttonsItemsNPC[i]->render();</code>	
59	<code> item.render(src,dst);</code>	
60	<code> textureGold = font.createText(std::to_string(items[j].second),&(window-></code>	
61	<code>getRenderer()),&w,&h);</code>	
62	<code> this->gold.push_back(textureGold);</code>	
63	<code> src = {0,0,w,h};</code>	

Page 2/3	PriestInterface.cpp	Page 2/3
60	<code> dst = {widthSegment/4+(i%3)*widthSegment/2,15+(i/3)*((this->window->getH</code>	
61	<code>eight())/2)/5+widthSegment/2,w,h);</code>	
62	<code> SDL_RenderCopy(&(window->getRenderer()), textureGold,&src,&dst);</code>	
63	<code> i++;</code>	
64	<code> }</code>	
65	<code> i = 0;</code>	
66	<code> loadButtons = false;</code>	
67	<code> if (this->buttonsNPC.size() == 0)</code>	
68	<code> loadButtons = true;</code>	
69	<code> for (auto& action: this->information.actions) {</code>	
70	<code> if (i % 2 == 0) {</code>	
71	<code> width = 0;</code>	
72	<code> } else {</code>	
73	<code> width = 100;</code>	
74	<code> }</code>	
75	<code> if (i < this->information.actions.size()/2.0) {</code>	
76	<code> height = 0;</code>	
77	<code> } else {</code>	
78	<code> height = 50;</code>	
79	<code> }</code>	
80	<code> if (loadButtons) {</code>	
81	<code> button = createButtonAction(action, {12+width, ((this->window->getHeig</code>	
82	<code>ht()-60)/2)-height-HEIGHTBUTTON, WIDTHBUTTON, HEIGHTBUTTON});</code>	
83	<code> this->buttonsNPC.push_back(button);</code>	
84	<code> }</code>	
85	<code> this->buttonsNPC[i]->setViewport({0,(this->window->getHeight())/2,(this-></code>	
86	<code>window->getWidth()/8)*2,(this->window->getHeight())/2});</code>	
87	<code> this->buttonsNPC[i]->updatePosition({12+width, ((this->window->getHeight()</code>	
88	<code>-60)/2)-height-HEIGHTBUTTON, WIDTHBUTTON, HEIGHTBUTTON});</code>	
89	<code> this->buttonsNPC[i]->render();</code>	
90	<code> i++;</code>	
91	<code> }</code>	
92	<code> }</code>	
93	<code>InputInfo PriestInterface::handleClick(int x, int y, int itemSelected) {</code>	
94	<code> InputInfo info;</code>	
95	<code> info.input = InputID::nothing;</code>	
96	<code> bool newItemSelected = false;</code>	
97	<code> for (auto& button: buttonsItemsNPC) {</code>	
98	<code> newItemSelected = button->inside(x,y);</code>	
99	<code> if (newItemSelected) {</code>	
100	<code> if (itemSelectedNPC == -1) {</code>	
101	<code> itemSelectedNPC = button->getId();</code>	
102	<code> } else if (itemSelectedNPC == button->getId()) {</code>	
103	<code> itemSelectedNPC = -1;</code>	
104	<code> } else if (itemSelectedNPC != button->getId()) {</code>	
105	<code> buttonsItemsNPC[itemSelectedNPC]->onClick();</code>	
106	<code> itemSelectedNPC = button->getId();</code>	
107	<code> }</code>	
108	<code> }</code>	
109	<code> for (uint i = 0; i < this->buttonsNPC.size(); i++) {</code>	
110	<code> if (buttonsNPC[i]->inside(x,y)) {</code>	
111	<code> if (i == 2 ^ itemSelectedNPC == -1)</code>	
112	<code> continue;</code>	
113	<code> info = buttonsNPC[i]->onClick(int(itemsPriest[itemSelectedNPC]));</code>	
114	<code> }</code>	
115	<code> }</code>	
116	<code> return info;</code>	
117	<code>void PriestInterface::deleteGoldValues() {</code>	
118	<code> std::vector<SDL_Texture*>::iterator iter;</code>	
119	<code> iter = this->gold.begin();</code>	
120	<code> while (iter != this->gold.end()) {</code>	
121	<code> SDL_DestroyTexture(*iter);</code>	

jul 21, 20 15:20	PriestInterface.cpp	Page 3/3
122	iter = gold.erase(iter);	
123	}	
124	}	
125		
126		
127	PriestInterface::~PriestInterface() {	
128	deleteGoldValues();	
129	}	

jul 21, 20 15:20	NPCInterface.h	Page 1/1
1	#ifndef NPCINTERFACE_H	
2	#define NPCINTERFACE_H	
3		
4	#include "../common/Identificators.h"	
5	#include <SDL2/SDL.h>	
6	#include "../Window.h"	
7	#include "../TextureManager.h"	
8	#include <memory>	
9	#include "../Player.h"	
10	#include "../Font.h"	
11	#include "RaisedButton.h"	
12		
13	class NPCInterface {	
14	protected:	
15	NPCInfo information;	
16	SDL_Texture* texture = nullptr;	
17	Window* window;	
18	const TextureManager& manager;	
19	Player* player;	
20	Font font;	
21	std::shared_ptr<RaisedButton> createButtonAction(ActionsProfessionID action,	
22	SDL_Rect rect);	
23	public:	
24	NPCInterface(NPCInfo info, Window* window, const TextureManager& manager, Play	
25	er* player);	
26	virtual void render() = 0;	
27	void update(NPCInfo npcInfo);	
28	virtual InputInfo handleClick(int x, int y, int itemSelected) = 0;	
29		
30	~NPCInterface();	
31		
32	};	
33		
34		
35	#endif	
36		

jul 21, 20 15:20	NPCInterface.cpp	Page 1/2
1	<code>#include "NPCInterface.h"</code>	
2	<code>#include "SellButton.h"</code>	
3	<code>#include "ResurrectButton.h"</code>	
4	<code>#include "BuyButton.h"</code>	
5	<code>#include "CureButton.h"</code>	
6	<code>#include "RetireGoldButton.h"</code>	
7	<code>#include "DepositGoldButton.h"</code>	
8	<code>#include "DepositItemButton.h"</code>	
9	<code>#include "RetireItemButton.h"</code>	
10		
11	<code>NPCInterface::NPCInterface(NPCInfo info, Window* window, const TextureManager& manager, Player* player)</code>	
12	<code>{ information(info), window(window), manager(manager), player(player),</code>	
13	<code>font("assets/font/Prince Valiant.ttf", 18, {0xA4, 0xA4, 0xA4}) }</code>	
14		
15	<code>std::shared_ptr<RaisedButton> NPCInterface::createButtonAction(ActionsProfession</code>	
16	<code>ID action, SDL_Rect rect) {</code>	
17	<code>std::string text;</code>	
18	<code>std::shared_ptr<RaisedButton> button;</code>	
19	<code>switch (action) {</code>	
20	<code>case ActionsProfessionID::Buy:</code>	
21	<code>text = "Comprar";</code>	
22	<code>button = std::shared_ptr<RaisedButton>(new BuyButton(&(window->getRender</code>	
23	<code>er()), font, text, rect, manager, player));</code>	
24	<code>break;</code>	
25	<code>case ActionsProfessionID::Sell:</code>	
26	<code>text = "Vender";</code>	
27	<code>button = std::shared_ptr<RaisedButton>(new SellButton(&(window->getRende</code>	
28	<code>rer()), font, text, rect, manager, player));</code>	
29	<code>break;</code>	
30	<code>case ActionsProfessionID::Resurrect:</code>	
31	<code>text = "Resucitar";</code>	
32	<code>button = std::shared_ptr<RaisedButton>(new ResurrectButton(&(window->get</code>	
33	<code>Renderer()), font, text, rect, manager, player));</code>	
34	<code>break;</code>	
35	<code>case ActionsProfessionID::Cure:</code>	
36	<code>text = "Curar";</code>	
37	<code>button = std::shared_ptr<RaisedButton>(new CureButton(&(window->getRende</code>	
38	<code>rer()), font, text, rect, manager, player));</code>	
39	<code>break;</code>	
40	<code>case ActionsProfessionID::DepositGold:</code>	
41	<code>text = "Depositar";</code>	
42	<code>button = std::shared_ptr<RaisedButton>(new DepositGoldButton(&(window->g</code>	
43	<code>etRenderer()), font, text, rect, manager, player));</code>	
44	<code>break;</code>	
45	<code>case ActionsProfessionID::DepositItem:</code>	
46	<code>text = "Depositar";</code>	
47	<code>button = std::shared_ptr<RaisedButton>(new DepositItemButton(&(window->g</code>	
48	<code>etRenderer()), font, text, rect, manager, player));</code>	
49	<code>break;</code>	
50	<code>case ActionsProfessionID::RetireItem:</code>	
51	<code>text = "Retirar";</code>	
52	<code>button = std::shared_ptr<RaisedButton>(new RetireItemButton(&(window->ge</code>	
53	<code>tRenderer()), font, text, rect, manager, player));</code>	
54	<code>break;</code>	
55	<code>case ActionsProfessionID::RetireGold:</code>	
56	<code>text = "Retirar";</code>	
57	<code>button = std::shared_ptr<RaisedButton>(new RetireGoldButton(&(window->ge</code>	
58	<code>tRenderer()), font, text, rect, manager, player));</code>	
59	<code>break;</code>	
60	<code>case ActionsProfessionID::Nothing:</code>	
61	<code>text = "";</code>	
62	<code>button = nullptr;</code>	
63	<code>break;</code>	
64	<code>}</code>	
65	<code>return button;</code>	
66		

jul 21, 20 15:20	NPCInterface.cpp	Page 2/2
57	<code>}</code>	
58		
59		
60	<code>NPCInterface::~NPCInterface() {</code>	
61	<code>SDL_DestroyTexture(this->texture);</code>	
62	<code>}</code>	
63		
64	<code>void NPCInterface::update(NPCInfo npcInfo) {</code>	
65	<code>information = npcInfo;</code>	
66	<code>}</code>	

jul 21, 20 15:20	MerchantInterface.h	Page 1/1
1	<code>#ifndef MERCHANTINTERFACE_H</code>	
2	<code>#define MERCHANTINTERFACE_H</code>	
3		
4	<code>#include "NPCInterface.h"</code>	
5	<code>#include <vector></code>	
6	<code>#include <memory></code>	
7	<code>#include "RaisedButton.h"</code>	
8	<code>#include "SelectButton.h"</code>	
9	<code>#include "ArrowButton.h"</code>	
10		
11	<code>class MerchantInterface: public NPCInterface {</code>	
12	<code>private:</code>	
13	<code>std::vector<std::shared_ptr<RaisedButton>> buttonsNPC; //Botones que nos mue</code>	
14	<code>stra el NPC</code>	
15	<code>std::vector<std::shared_ptr<SelectButton>> buttonsItemsNPC; //Items que nos</code>	
16	<code>muestra el NPC</code>	
17	<code>std::vector<SDL_Texture*> gold; //El oro que se salen cada uno de los items</code>	
18	<code>std::vector<ItemsInventoryID> itemsMerchant;</code>	
19	<code>int itemSelectedNPC{-1};</code>	
20	<code>uint pagItems{0}; //La página que nos está mostrando el NPC de los items</code>	
21	<code>uint pagMax{0}; //La cantidad de páginas que se va a tener con el NPC</code>	
22	<code>std::shared_ptr<ArrowButton> arrow{nullptr};</code>	
23	<code>void deleteGoldValues();</code>	
24	<code>public:</code>	
25	<code>explicit MerchantInterface(NPCInfo info, Window* window, const TextureManager</code>	
26	<code>& manager, Player* player);</code>	
27	<code>virtual void render();</code>	
28	<code>virtual InputInfo handleClick(int x, int y, int itemSelected);</code>	
29	<code>~MerchantInterface();</code>	
30	<code>};</code>	
31		
32		
33	<code>#endif</code>	

jul 21, 20 15:20	MerchantInterface.cpp	Page 1/3
1	<code>#include "MerchantInterface.h"</code>	
2		
3	<code>#define WIDTHBUTTON 70</code>	
4	<code>#define HEIGHTBUTTON 25</code>	
5	<code>#define ITEMSMERCHANT 9</code>	
6		
7	<code>MerchantInterface::MerchantInterface(NPCInfo info, Window* window, const TextureM</code>	
8	<code>anager& manager, Player* player) :</code>	
9	<code>NPCInterface(info, window, manager, player), buttonsNPC(), buttonsItemsNPC(), g</code>	
10	<code>old(), itemsMerchant()) {</code>	
11	<code>int width_text, height_text;</code>	
12	<code>SDL_Texture* merchant = font.createText("Comerciante",</code>	
13	<code>&(window->getRenderer()), &width_text, &height_text);</code>	
14	<code>this->texture = merchant;</code>	
15		
16	<code>void MerchantInterface::render() {</code>	
17	<code>int w,h;</code>	
18	<code>int i = 0;</code>	
19	<code>int width = 0;</code>	
20	<code>SDL_QueryTexture(this->texture, NULL, NULL, &w, &h);</code>	
21	<code>SDL_Rect merchant = {15,15,w,h};</code>	
22	<code>SDL_RenderCopy(&(window->getRenderer()), this->texture, NULL, &merchant);</code>	
23	<code>SDL_Rect src;</code>	
24	<code>SDL_Rect dst;</code>	
25	<code>std::shared_ptr<RaisedButton> button;</code>	
26	<code>std::shared_ptr<SelectButton> selection;</code>	
27	<code>SDL_Texture* textureGold;</code>	
28	<code>bool loadButtons = false;</code>	
29		
30	<code>this->pagMax = information.items.size() / ITEMSMERCHANT + 1;</code>	
31	<code>std::vector<std::pair<ItemsInventoryID, uint>> items(information.items.begin</code>	
32	<code>(), information.items.end());</code>	
33		
34	<code>if (this->itemsMerchant.size() == 0) {</code>	
35	<code>for (auto iter: items) {</code>	
36	<code>this->itemsMerchant.push_back(iter.first);</code>	
37	<code>}</code>	
38		
39	<code>if (this->buttonsItemsNPC.size() == 0)</code>	
40	<code>loadButtons = true;</code>	
41		
42	<code>uint max;</code>	
43	<code>if ((pagItems+1)*ITEMSMERCHANT > information.items.size()){</code>	
44	<code>max = information.items.size();</code>	
45	<code>} else {</code>	
46	<code>max = (pagItems+1)*ITEMSMERCHANT;</code>	
47	<code>}</code>	
48	<code>int widthSegment = this->window->getWidth()/8;</code>	
49	<code>for (uint j = pagItems*ITEMSMERCHANT; j < max; j++) {</code>	
50	<code>const Texture& item = manager.getTexture(items[j].first);</code>	
51	<code>src = {0,0,52,52};</code>	
52	<code>dst = {20+(i%3)*widthSegment/2,50+(i/3)*((this->window->getHeight())/2)/</code>	
53	<code>5,widthSegment/3,widthSegment/3};</code>	
54	<code>if (loadButtons) {</code>	
55	<code>selection = std::shared_ptr<SelectButton>(new</code>	
56	<code>SelectButton(&(window->getRenderer()),dst,manager,j));</code>	
57	<code>this->buttonsItemsNPC.push_back(selection);</code>	
58	<code>this->buttonsItemsNPC[i]->setViewport({0,(this->window->getHeight())/2,</code>	
59	<code>(this->window->getWidth()/8)*2,(this->window->get</code>	
60	<code>Height())/2});</code>	
61	<code>this->buttonsItemsNPC[i]->updatePosition(dst);</code>	
62	<code>this->buttonsItemsNPC[i]->render();</code>	

jul 21, 20 15:20	MerchantInterface.cpp	Page 2/3
62	item.render(src,dst);	
63	textureGold = font.createText(std::to_string(items[j].second), &(window→getRenderer()), &w, &h);	
64	this→gold.push_back(textureGold);	
65	src = {0,0,w,h};	
66	dst = {widthSegment/4+(i%3)*widthSegment/2, 15+(i/3)*((this→window→getHeight())/2)/5+widthSegment/2, w, h};	
67	SDL_RenderCopy(&(window→getRenderer()), textureGold, &src, &dst);	
68	i++;	
69	if (i == ITEMSMERCHANT)	
70	break;	
71	}	
72	i = 0;	
73		
74	loadButtons = false;	
75	if (this→buttonsNPC.size() == 0)	
76	loadButtons = true;	
77	for (auto& action: this→information.actions) {	
78	if (i == 0) {	
79	width = 0;	
80	} else {	
81	width = 100;	
82	}	
83	if (loadButtons){	
84	button = createAction(action, {12+width,	
85	((this→window→getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
86	HEIGHTBUTTON);	
87	this→buttonsNPC.push_back(button);	
88	this→buttonsNPC[i]→setViewport({0, (this→window→getHeight())/2,	
89	(this→window→getWidth()/8)*2, (this→window→getHeight()	
90)/2});	
91	this→buttonsNPC[i]→updatePosition({12+width,	
92	((this→window→getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
93	HEIGHTBUTTON);	
94	this→buttonsNPC[i]→render();	
95	i++;	
96	if (loadButtons) {	
97	SDL_Rect rect = {(this→window→getWidth()/8)*2-40, ((this→window→getHei	
98	ght()-60)/2)-HEIGHTBUTTON*4, 20, 20};	
99	this→arrow = std::shared_ptr<ArrowButton>(new ArrowButton(&(window→get	
100	Renderer()), font, "+", rect, manager);	
101	}	
102	if (this→arrow != nullptr){	
103	this→arrow→setViewport({0, (this→window→getHeight())/2,	
104	(this→window→getWidth()/8)*2, (this→window→getHeight())/2	
105	});	
106	this→arrow→updatePosition({(this→window→getWidth()/8)*2-40,	
107	((this→window→getHeight()-60)/2)-HEIGHTBUTTON*4, 20, 20});	
108	this→arrow→render();	
109	}	
110		
111	InputInfo MerchantInterface::handleClick(int x, int y, int itemSelected) {	
112	InputInfo info;	
113	info.input = InputID::nothing;	
114	bool newItemSelected = false;	
115	for(auto& button: buttonsItemsNPC){	
116	newItemSelected = button→inside(x,y);	
117	if (newItemSelected) {	
118	if (itemSelectedNPC == -1) {	
119	itemSelectedNPC = button→getId();	
120	} else if (itemSelectedNPC == button→getId()) {	
121	itemSelectedNPC = -1;	
122	} else if (itemSelectedNPC != button→getId()) {	

jul 21, 20 15:20	MerchantInterface.cpp	Page 3/3
120	buttonsItemsNPC[itemSelectedNPC]→onClick();	
121	itemSelectedNPC = button→getId();	
122	}	
123	}	
124	}	
125	if (buttonsNPC[0]→inside(x,y) ^ itemSelectedNPC > -1)	
126	info = buttonsNPC[0]→onClick(int(itemsMerchant[itemSelectedNPC+pagItems	
127	*ITEMSMERCHANT));	
128	if (buttonsNPC[1]→inside(x,y))	
129	info = buttonsNPC[1]→onClick(itemSelected);	
130		
131	if (this→arrow != nullptr ^ this→arrow→inside(x,y)) {	
132	this→arrow→onClick();	
133	this→pagItems = (pagItems+1) % this→pagMax;	
134	}	
135	return info;	
136	}	
137	void MerchantInterface::deleteGoldValues() {	
138	std::vector<SDL_Texture*>::iterator iter;	
139	iter = this→gold.begin();	
140	while (iter != this→gold.end()) {	
141	SDL_DestroyTexture(*iter);	
142	iter = gold.erase(iter);	
143	}	
144	}	
145		
146		
147	MerchantInterface::~MerchantInterface() {	
148	deleteGoldValues();	
149	}	

jul 21, 20 15:20	EquipButton.h	Page 1/1
1	#ifndef EQUIPBUTTON_H	
2	#define EQUIPBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class EquipButton : public RaisedButton {	
8	Player* player;	
9	public:	
10	EquipButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
11	const TextureManager& manager, Player* player);	
12	virtual bool inside(int x, int y);	
13	virtual void render();	
14	virtual void render();	
15	virtual InputInfo onClick(int item);	
16	virtual void setViewport(SDL_Rect viewport);	
17	virtual ~EquipButton();	
18	};	
19		
20		
21		
22		
23		
24	#endif	

jul 21, 20 15:20	EquipButton.cpp	Page 1/1
1	#include "EquipButton.h"	
2		
3	EquipButton::EquipButton(SDL_Renderer* renderer, Font& font, std::string text,	
4	SDL_Rect position, const TextureManager& manager, Player* player) :	
5	RaisedButton(renderer, font, text, position, manager), player(player) {}	
6		
7	InputInfo EquipButton::onClick(int item) {	
8	this →clicked = !this →clicked;	
9	return this →player→selectItem(item);	
10	}	
11		
12	bool EquipButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void EquipButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void EquipButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	EquipButton::~EquipButton(){}	

jul 21, 20 15:20	DropButton.h	Page 1/1
1	<code>#ifndef DROPBUTTON_H</code>	
2	<code>#define DROPBUTTON_H</code>	
3		
4	<code>#include "RaisedButton.h"</code>	
5	<code>#include "../Player.h"</code>	
6		
7	<code>class DropButton : public RaisedButton {</code>	
8	<code>private:</code>	
9	<code> Player* player;</code>	
10	<code>public:</code>	
11	<code> DropButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,</code>	
12	<code> const TextureManager& manager, Player* player);</code>	
13		
14	<code> virtual bool inside(int x, int y);</code>	
15		
16	<code> virtual void render();</code>	
17		
18	<code> virtual InputInfo onClick(int item);</code>	
19		
20	<code> void setViewport(SDL_Rect viewport);</code>	
21		
22	<code> virtual ~DropButton();</code>	
23	<code>};</code>	
24		
25	<code>#endif</code>	

jul 21, 20 15:20	DropButton.cpp	Page 1/1
1	<code>#include "DropButton.h"</code>	
2		
3	<code>DropButton::DropButton(SDL_Renderer* renderer, Font& font, std::string text,</code>	
4	<code> SDL_Rect position, const TextureManager& manager, Player* player) :</code>	
5	<code> RaisedButton(renderer, font, text, position, manager), player(player) {}</code>	
6		
7	<code>InputInfo DropButton::onClick(int item) {</code>	
8	<code> this->clicked = ~this->clicked;</code>	
9	<code> return this->player->dropItem(item);</code>	
10	<code>}</code>	
11		
12	<code>bool DropButton::inside(int x, int y) {</code>	
13	<code> return RaisedButton::inside(x, y);</code>	
14	<code>}</code>	
15		
16	<code>void DropButton::render() {</code>	
17	<code> RaisedButton::render();</code>	
18	<code>}</code>	
19		
20	<code>void DropButton::setViewport(SDL_Rect viewport){</code>	
21	<code> RaisedButton::setViewport(viewport);</code>	
22	<code>}</code>	
23		
24	<code>DropButton::~DropButton(){}</code>	

jul 21, 20 15:20	DepositItemButton.h	Page 1/1
1	#ifndef DEPOSITITEMBUTTON_H	
2	#define DEPOSITITEMBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class DepositItemButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	DepositItemButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_R	
12	ect position,	
13	const TextureManager& manager, Player* player);	
14	virtual bool inside(int x, int y);	
15	virtual void render();	
16	virtual InputInfo onClick(int item);	
17	void setViewport(SDL_Rect viewport);	
18	virtual ~DepositItemButton();	
19	};	
20		
21	#endif	
22		
23		
24		
25		

jul 21, 20 15:20	DepositItemButton.cpp	Page 1/1
1	#include "DepositItemButton.h"	
2		
3	DepositItemButton::DepositItemButton(SDL_Renderer* renderer, Font& font, std::str	
4	ing text,	
5	SDL_Rect position, const TextureManager& manager, Player* player) :	
6	RaisedButton(renderer, font, text, position, manager), player(player) {}	
7	InputInfo DepositItemButton::onClick(int item) {	
8	this →clicked = ~ this →clicked;	
9	return this →player→deposit(item, true);	
10	}	
11		
12	bool DepositItemButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void DepositItemButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void DepositItemButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	DepositItemButton::~DepositItemButton(){}	

jul 21, 20 15:20	DepositGoldButton.h	Page 1/1
1	#ifndef DEPOSITGOLDBUTTON_H	
2	#define DEPOSITGOLDBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class DepositGoldButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	DepositGoldButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_R	
12	ect position,	
13	const TextureManager& manager, Player* player);	
14	virtual bool inside(int x, int y);	
15	virtual void render();	
16	virtual InputInfo onClick(int item);	
17	virtual void setViewport(SDL_Rect viewport);	
18	virtual ~DepositGoldButton();	
19	};	
20		
21	#endif	
22		
23		
24		
25		

jul 21, 20 15:20	DepositGoldButton.cpp	Page 1/1
1	#include "DepositGoldButton.h"	
2		
3	DepositGoldButton::DepositGoldButton(SDL_Renderer* renderer, Font& font, std::str	
4	ing text,	
5	SDL_Rect position, const TextureManager& manager, Player* player) :	
6	RaisedButton(renderer, font, text, position, manager), player(player) {}	
7	InputInfo DepositGoldButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→deposit(item, false);	
10	}	
11		
12	bool DepositGoldButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void DepositGoldButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void DepositGoldButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	DepositGoldButton::~DepositGoldButton(){}	

jul 21, 20 15:20	CureButton.h	Page 1/1
1	#ifndef CUREBUTTON_H	
2	#define CUREBUTTON_H	
3		
4	#include "RaisedButton.h"	
5	#include "../Player.h"	
6		
7	class CureButton: public RaisedButton {	
8	private:	
9	Player* player;	
10	public:	
11	CureButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position,	
12	const TextureManager& manager, Player* player);	
13		
14	virtual bool inside(int x, int y);	
15		
16	virtual void render();	
17		
18	virtual InputInfo onClick(int item);	
19		
20	void setViewport(SDL_Rect viewport);	
21		
22	virtual ~CureButton();	
23	};	
24		
25	#endif	

jul 21, 20 15:20	CureButton.cpp	Page 1/1
1	#include "CureButton.h"	
2		
3	CureButton::CureButton(SDL_Renderer* renderer, Font& font, std::string text,	
4	SDL_Rect position, const TextureManager& manager, Player* player) :	
5	RaisedButton(renderer, font, text, position, manager), player(player) {}	
6		
7	InputInfo CureButton::onClick(int item) {	
8	this →clicked = ~this →clicked;	
9	return this →player→cure();	
10	}	
11		
12	bool CureButton::inside(int x, int y) {	
13	return RaisedButton::inside(x, y);	
14	}	
15		
16	void CureButton::render() {	
17	RaisedButton::render();	
18	}	
19		
20	void CureButton::setViewport(SDL_Rect viewport){	
21	RaisedButton::setViewport(viewport);	
22	}	
23		
24	CureButton::~CureButton(){} }	

jul 21, 20 15:20	BuyButton.h	Page 1/1
1	<code>#ifndef BUYBUTTON_H</code>	
2	<code>#define BUYBUTTON_H</code>	
3		
4	<code>#include "RaisedButton.h"</code>	
5	<code>#include "../Player.h"</code>	
6		
7	<code>class BuyButton: public RaisedButton {</code>	
8	<code>private:</code>	
9	<code> Player* player;</code>	
10	<code>public:</code>	
11	<code> BuyButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect posi</code>	
12	<code>tion,</code>	
13	<code> const TextureManager& manager, Player* player);</code>	
14	<code> virtual bool inside(int x, int y);</code>	
15		
16	<code> virtual void render();</code>	
17		
18	<code> virtual InputInfo onClick(int item);</code>	
19		
20	<code> void setViewport(SDL_Rect viewport);</code>	
21		
22	<code> virtual ~BuyButton();</code>	
23	<code>};</code>	
24		
25	<code>#endif</code>	

jul 21, 20 15:20	BuyButton.cpp	Page 1/1
1	<code>#include "BuyButton.h"</code>	
2		
3	<code>BuyButton::BuyButton(SDL_Renderer* renderer, Font& font, std::string text,</code>	
4	<code> SDL_Rect position, const TextureManager& manager, Player* player) :</code>	
5	<code> RaisedButton(renderer, font, text, position, manager), player(player) {}</code>	
6		
7	<code>InputInfo BuyButton::onClick(int item) {</code>	
8	<code> this->clicked = ~this->clicked;</code>	
9	<code> return this->player->buy(item);</code>	
10	<code>}</code>	
11		
12	<code>bool BuyButton::inside(int x, int y) {</code>	
13	<code> return RaisedButton::inside(x, y);</code>	
14	<code>}</code>	
15		
16	<code>void BuyButton::render() {</code>	
17	<code> RaisedButton::render();</code>	
18	<code>}</code>	
19		
20	<code>void BuyButton::setViewport(SDL_Rect viewport){</code>	
21	<code> RaisedButton::setViewport(viewport);</code>	
22	<code>}</code>	
23		
24	<code>BuyButton::~BuyButton(){}</code>	

jul 21, 20 15:20	Button.h	Page 1/1
1	<code>#ifndef BUTTON_H</code>	
2	<code>#define BUTTON_H</code>	
3		
4	<code>#include "../Fonh"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class Button {</code>	
8	<code>protected:</code>	
9	<code> std::string text;</code>	
10	<code> SDL_Rect button;</code>	
11	<code> int textSize{8};</code>	
12	<code> SDL_Texture* texture = nullptr;</code>	
13	<code> const TextureManager& manager;</code>	
14	<code> SDL_Renderer* renderer;</code>	
15	<code> SDL_Rect viewport{0,0,0,0};</code>	
16	<code> bool clicked;</code>	
17		
18	<code>public:</code>	
19	<code> Button(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position, const TextureManager& manager);</code>	
20	<code> Button(SDL_Renderer* renderer, SDL_Rect position, const TextureManager& manager);</code>	
21	<code> virtual bool inside(int x, int y) = 0;</code>	
22	<code> virtual void render() = 0;</code>	
23	<code> void onClick();</code>	
24	<code> virtual void updatePosition(SDL_Rect position);</code>	
25	<code> void setViewport(SDL_Rect viewport);</code>	
26	<code> ~Button();</code>	
27	<code>};</code>	
28	<code>#endif</code>	

jul 21, 20 15:20	Button.cpp	Page 1/1
1	<code>#include "Button.h"</code>	
2		
3	<code>Button::Button(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect position, const TextureManager& manager) : text(text), button(position), manager(manager),</code>	
4	<code> renderer(renderer), clicked(false) {</code>	
5	<code> int w,h;</code>	
6	<code> font.setSize(this->textSize);</code>	
7	<code> this->texture = font.createText(text, renderer, &w, &h);</code>	
8	<code>}</code>	
9		
10		
11	<code>Button::Button(SDL_Renderer* renderer, SDL_Rect position, const TextureManager& manager) :</code>	
12	<code> button(position), manager(manager), renderer(renderer), clicked(false){}</code>	
13		
14	<code>void Button::onClick() {</code>	
15	<code> this->clicked = !this->clicked;</code>	
16	<code>}</code>	
17		
18	<code>void Button::updatePosition(SDL_Rect position) {</code>	
19	<code> this->button = position;</code>	
20	<code>}</code>	
21		
22	<code>void Button::setViewport(SDL_Rect viewport) {</code>	
23	<code> this->viewport = viewport;</code>	
24	<code>}</code>	
25		
26	<code>Button::~Button() {</code>	
27	<code> SDL_DestroyTexture(this->texture);</code>	
28	<code>}</code>	

jul 21, 20 15:20	BankerInterface.h	Page 1/1
1	#ifndef BANKERINTERFACE_H	
2	#define BANKERINTERFACE_H	
3		
4	#include "NPCInterface.h"	
5	#include <vector>	
6	#include <memory>	
7	#include "RaisedButton.h"	
8	#include "SelectButton.h"	
9	#include "ArrowButton.h"	
10		
11	class BankerInterface: public NPCInterface {	
12	private:	
13	std::vector<std::shared_ptr<RaisedButton>> buttonsNPC; //Botones que nos mue	
14	std::vector<std::shared_ptr<SelectButton>> buttonsItemsNPC; //Items que nos	
15	std::shared_ptr<ArrowButton> changeScreen{nullptr};	
16	std::vector<std::shared_ptr<ArrowButton>> arrows;	
17	bool pageGold{true};	
18	int amountGold{0};	
19	SDL_Texture* goldValue{nullptr};	
20	SDL_Texture* inBank{nullptr};	
21	SDL_Texture* goldInBank{nullptr};	
22	int itemSelectedNPC{-1};	
23	uint pagItemsInBank{0}; //La página que nos está; mostrando el NPC de los i	
24	tems depositados	
25	uint pagMaxInBank{0}; //La cantidad de páginas que se va a tener con el NPC	
26	std::shared_ptr<ArrowButton> arrow{nullptr};	
27		
28	void renderGoldManagment();	
29	void renderItems();	
30	public:	
31	explicit BankerInterface(NPCInfo info, Window* window, const TextureManager&	
32	manager, Player* player);	
33		
34	virtual void render();	
35		
36	virtual InputInfo handleClick(int x, int y, int itemSelected);	
37		
38	~BankerInterface();	
39		
40	#endif	

jul 21, 20 15:20	BankerInterface.cpp	Page 1/5
1	#include "BankerInterface.h"	
2	#include <SDL2/SDL.h>	
3		
4	#include <memory>	
5	#include <utility>	
6		
7	#define WIDTHBUTTON 70	
8	#define HEIGHTBUTTON 25	
9	#define ITEMSBANKER 9	
10		
11	BankerInterface::BankerInterface(NPCInfo info, Window* window, const TextureMana	
12	ger& manager, Player* player) :	
13	NPCInterface(std::move(info), window, manager, player), buttonsNPC(), button	
14	sItemsNPC(), arrows() {	
15	int width_text, height_text;	
16	SDL_Texture* banker = font.createText("Banquero",	
17	&(window->getRenderer()), &width_text, &height_text);	
18	this->texture = banker;	
19	this->inBank = font.createText("Oro en Banco: ",	
20	&(window->getRenderer()), &width_text, &height_text);	
21	SDL_Rect rect = {(this->window->getWidth())/8}*2-40, ((this->window->getHeight(
22)-60)/2)-HEIGHTBUTTON*3,20,20);	
23	this->changeScreen = std::make_shared<ArrowButton>(&(window->getRenderer()),	
24	font, ">", rect, manager);	
25	}	
26		
27	void BankerInterface::renderGoldManagment() {	
28	int w,h;	
29	int i =0;	
30	SDL_QueryTexture(this->inBank, NULL, NULL, &w, &h);	
31	SDL_Rect bank = {15, 45, w, h};	
32	SDL_RenderCopy(&(window->getRenderer()), this->inBank, NULL, &bank);	
33	bank = {15, 45 + h, w, h};	
34	this->goldInBank = font.createText(std::to_string(information.gold),	
35	&(window->getRenderer()), &w, &h);	
36	bank.w = w;	
37	bank.h = h;	
38	SDL_RenderCopy(&(window->getRenderer()), this->goldInBank, NULL, &bank);	
39		
40	const Texture& goldTexture = manager.getTexture(TextureID::Gold);	
41	SDL_Rect srcGold = {0,0,32,32};	
42	SDL_Rect dstGold = {25,140,32,32};	
43	goldTexture.render(srcGold,dstGold);	
44	std::string inputText;	
45	if (this->amountGold == 0) {	
46	int amount = this->player->getGold() - this->player->getSafeGold();	
47	if (amount < 0)	
48	amount = 0;	
49	inputText = std::to_string(amount);	
50	}	
51	inputText = std::to_string(this->amountGold);	
52	this->goldValue = font.createText(inputText,	
53	&(window->getRenderer()), &w, &h);	
54	SDL_Rect gold = {100,140,w,h};	
55	SDL_RenderCopy(&(window->getRenderer()), this->goldValue, NULL, &gold);	
56		
57	bool loadButtons =false;	
58		
59	if(this->arrows.empty()) {	
60	SDL_Rect rect = {100,120,20,20};	
61	this->arrows.push_back(std::make_shared<ArrowButton>(&(window->getRender	
62	er()),font,"+",rect, manager));	
63	rect = {100,160,20,20};	
64	this->arrows.push_back(std::make_shared<ArrowButton>(&(window->getRender	
65	er()),font,"-",rect, manager));	
66	}	

jul 21, 20 15:20	BankerInterface.cpp	Page 2/5
61	for (auto& button: arrows) {	
62	button->setViewport({0, (this->>window->getHeight())/2,	
63	(this->>window->getWidth()/8)*2, (this->>window->getHeight())/2}	
64	};	
65	button->render();	
66	loadButtons = false;	
67	if (this->buttonsNPC.size() == 0)	
68	loadButtons = true;	
69	int width = 0;	
70	for (auto& action: information.actions){	
71	if (action == ActionsProfessionID::DepositGold ∨ action == ActionsProfessi	
72	onID::RetireGold) {	
73	if (i == 0) {	
74	width = 0;	
75	} else {	
76	width = 100;	
77	if (loadButtons){	
78	this->buttonsNPC.push_back(createButtonAction(action, {12+width,	
79	((this->>window->getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
80	HEIGHTBUTTON));	
81	this->buttonsNPC[i]->setViewport({0, (this->>window->getHeight())/2,	
82	(this->>window->getWidth()/8)*2, (this->>window->getHeight()	
83)/2});	
84	this->buttonsNPC[i]->updatePosition({12+width,	
85	((this->>window->getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
86	HEIGHTBUTTON));	
87	this->buttonsNPC[i]->render();	
88	i++;	
89	}	
90	}	
91		
92	void BankerInterface::renderItems() {	
93	bool loadButtons = false;	
94	SDL_Rect src;	
95	SDL_Rect dst;	
96	int i = 0;	
97	int width = 0;	
98	std::shared_ptr<RaisedButton> button;	
99	std::shared_ptr<SelectButton> selection;	
100		
101	this->pagMaxInBank = information.itemsInBank.size()/ITEMSBANKER+1;	
102		
103	if (this->buttonsItemsNPC.size() != information.itemsInBank.size()) {	
104	loadButtons = true;	
105	this->buttonsItemsNPC.clear();	
106	}	
107	uint max;	
108	if ((pagItemsInBank+1)*ITEMSBANKER > information.itemsInBank.size()){	
109	max = information.itemsInBank.size();	
110	} else {	
111	max = (pagItemsInBank+1)*ITEMSBANKER;	
112	}	
113	int widthSegment = this->>window->getWidth()/8;	
114	for (uint j = pagItemsInBank*ITEMSBANKER; j < max; j++) {	
115	const Texture& item = manager.getTexture(information.itemsInBank[j]);	
116	src = {0,0,52,52};	
117	dst = {20+(i%3)*widthSegment/2,50+(i/3)*((this->>window->getHeight())/2)/	
118	5,widthSegment/3,widthSegment/3};	
119	if (loadButtons) {	
120	selection = std::make_shared<SelectButton>(&(window->getRenderer()),	
121	dst,manager,j);	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	BankerInterface.cpp	Page 3/5
120	this->buttonsItemsNPC.push_back(selection);	
121	}	
122	this->buttonsItemsNPC[i]->setViewport({0, (this->>window->getHeight())/2,	
123	(this->>window->getWidth()/8)*2, (this->>window->get	
124	Height())/2});	
125	this->buttonsItemsNPC[i]->updatePosition(dst);	
126	this->buttonsItemsNPC[i]->render();	
127	item.render(src,dst);	
128	i++;	
129	if (i == ITEMSBANKER)	
130	break ;	
131	}	
132	i = 0;	
133	loadButtons = false;	
134	if (this->buttonsNPC.size() == 0)	
135	loadButtons = true;	
136	for (auto& action: information.actions){	
137	if (action == ActionsProfessionID::DepositItem ∨ action == ActionsProfessi	
138	onID::RetireItem) {	
139	if (i == 0) {	
140	width = 0;	
141	} else {	
142	width = 100;	
143	if (loadButtons){	
144	this->buttonsNPC.push_back(createButtonAction(action, {12+width,	
145	((this->>window->getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
146	HEIGHTBUTTON));	
147	this->buttonsNPC[i]->setViewport({0, (this->>window->getHeight())/2,	
148	(this->>window->getWidth()/8)*2, (this->>window->getHeight()	
149)/2});	
150	this->buttonsNPC[i]->updatePosition({12+width,	
151	((this->>window->getHeight()-60)/2)-HEIGHTBUTTON, WIDTHBUTTON,	
152	HEIGHTBUTTON));	
153	this->buttonsNPC[i]->render();	
154	i++;	
155	}	
156	}	
157	if (loadButtons) {	
158	SDL_Rect rect = {(this->>window->getWidth()/8)*2-40, ((this->>window->getHei	
159	ght()-60)/2)-HEIGHTBUTTON*4,20,20};	
160	this->arrow = std::make_shared<ArrowButton>(&(window->getRenderer()),fon	
161	t,"+",rect, manager);	
162	if (this->arrow != nullptr){	
163	this->arrow->setViewport({0, (this->>window->getHeight())/2,	
164	(this->>window->getWidth()/8)*2, (this->>window->getHeight())/2}	
165);	
166	this->arrow->updatePosition({(this->>window->getWidth()/8)*2-40,	
167	((this->>window->getHeight()-60)/2)-HEIGHTBUTTON*4,20,20});	
168	this->arrow->render();	
169	}	
170	}	
171	void BankerInterface::render() {	
172	int w,h;	
173	SDL_QueryTexture(this->texture, NULL, NULL, &w, &h);	
174	SDL_Rect banker = {15,15,w,h};	
175	SDL_RenderCopy(&(window->getRenderer()), this->texture, NULL, &banker);	
176	if (this->goldValue != nullptr) {	
177	SDL_DestroyTexture(this->goldValue);	
178	this->goldValue = nullptr;	

128/217

jul 21, 20 15:20	BankerInterface.cpp	Page 4/5
177		
178	if (this->goldInBank != nullptr){	
179	SDL_DestroyTexture(this->goldInBank);	
180	this->goldInBank = nullptr;	
181	}	
182	if (pageGold) {	
183	renderGoldManagment();	
184	} else {	
185	renderItems();	
186	}	
187	if (this->changeScreen != nullptr){	
188	this->changeScreen->getViewPort({0,(this->window->getHeight())/2,	
189	(this->window->getWidth())/8*2,(this->window->getHeight())/2}	
190);	
191	this->changeScreen->updatePosition({(this->window->getWidth())/8*2-40,	
192	((this->window->getHeight()-60)/2)-HEIGHTBUTTON*3,20,20});	
193	this->changeScreen->render();	
194	}	
195		
196	InputInfo BankerInterface::handleClick(int x, int y, int itemSelected) {	
197	InputInfo info;	
198	info.input = InputID::nothing;	
199	bool newItemSelected = false;	
200	for(auto& button: buttonsItemsNPC){	
201	newItemSelected = button->inside(x,y);	
202	if (newItemSelected) {	
203	if(itemSelectedNPC == -1) {	
204	itemSelectedNPC = button->getId();	
205	} else if (itemSelectedNPC == button->getId()) {	
206	itemSelectedNPC = -1;	
207	} else if (itemSelectedNPC != button->getId()) {	
208	buttonsItemsNPC[itemSelectedNPC]->onClick();	
209	itemSelectedNPC = button->getId();	
210	}	
211	}	
212	if (pageGold) {	
213	if (buttonsNPC[0]->inside(x,y))	
214	info = buttonsNPC[0]->onClick(amountGold);	
215	if (buttonsNPC[1]->inside(x,y))	
216	info = buttonsNPC[1]->onClick(amountGold);	
217	} else {	
218	if (buttonsNPC[0]->inside(x,y) ^ itemSelected > 0)	
219	info = buttonsNPC[0]->onClick(itemSelected);	
220	if (buttonsNPC[1]->inside(x,y) ^ itemSelectedNPC > -1) {	
221	info = buttonsNPC[1]->onClick(itemSelectedNPC);	
222	itemSelectedNPC = -1;	
223	}	
224	}	
225		
226	if (this->changeScreen != nullptr ^ this->changeScreen->inside(x,y)) {	
227	this->changeScreen->onClick();	
228	this->pageGold = -this->pageGold;	
229	this->buttonsItemsNPC.clear();	
230	this->buttonsNPC.clear();	
231	}	
232	if (this->arrows.size() != 0) {	
233	if (this->arrows[0]->inside(x,y)){	
234	this->amountGold +=10;	
235	}	
236	if (this->arrows[1]->inside(x,y)) {	
237	this->amountGold -=10;	
238	if (this->amountGold < 0)	
239	this->amountGold = 0;	
240	}	
241	}	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	BankerInterface.cpp	Page 5/5
242	return info;	
243	}	
244		
245	BankerInterface::~BankerInterface() {	
246	SDL_DestroyTexture(this->inBank);	
247	}	

129/217

jul 21, 20 15:20	ArrowButton.h	Page 1/1
1	<code>#ifndef ARROWBUTTON_H</code>	
2	<code>#define ARROWBUTTON_H</code>	
3		
4	<code>#include "Button.h"</code>	
5	<code>#include "../Player.h"</code>	
6		
7	<code>class ArrowButton: public Button {</code>	
8	<code>public:</code>	
9	<code> ArrowButton(SDL_Renderer* renderer, Font& font, std::string text, SDL_Rect po</code>	
10	<code>sition,</code>	
11	<code> const TextureManager& manager);</code>	
12	<code> virtual bool inside(int x, int y);</code>	
13	<code> virtual void render();</code>	
14	<code> virtual void onClick();</code>	
15	<code> void setViewport(SDL_Rect viewport);</code>	
16	<code> virtual ~ArrowButton();</code>	
17	<code>};</code>	
18		
19		
20		
21		
22		
23	<code>#endif</code>	

jul 21, 20 15:20	ArrowButton.cpp	Page 1/1
1	<code>#include "ArrowButton.h"</code>	
2		
3	<code>ArrowButton::ArrowButton(SDL_Renderer* renderer, Font& font, std::string text,</code>	
4	<code> SDL_Rect position, const TextureManager& manager) :</code>	
5	<code> Button(renderer, font, text, position, manager) {}</code>	
6		
7	<code>bool ArrowButton::inside(int x, int y) {</code>	
8	<code> bool inside = true;</code>	
9	<code> if (x < this->button.x+viewport.x)</code>	
10	<code> inside = false;</code>	
11	<code> if (x > this->button.x+this->button.w+viewport.x)</code>	
12	<code> inside = false;</code>	
13	<code> if (y < this->button.y+viewport.y)</code>	
14	<code> inside = false;</code>	
15	<code> if (y > this->button.y + this->button.h+viewport.y)</code>	
16	<code> inside = false;</code>	
17	<code> return inside;</code>	
18	<code>}</code>	
19		
20	<code>void ArrowButton::onClick() {</code>	
21	<code> this->clicked = !this->clicked;</code>	
22	<code>}</code>	
23		
24	<code>void ArrowButton::render() {</code>	
25	<code> int w,h;</code>	
26	<code> const Texture& background = manager.getTexture(TextureID::Button);</code>	
27	<code> SDL_QueryTexture(background.getTexture(), NULL, NULL, &w, &h);</code>	
28	<code> SDL_Rect src = {0,0,w,h};</code>	
29	<code> background.render(src, this->button);</code>	
30	<code> if(this->clicked) {</code>	
31	<code> SDL_Rect click = {button.x-1,button.y-1,button.w+2,button.h+2};</code>	
32	<code> SDL_SetRenderDrawColor(this->renderer, 0xA4, 0xA4, 0xA4, 0xAA);</code>	
33	<code> SDL_RenderDrawRect(this->renderer, &click);</code>	
34	<code> this->clicked = !this->clicked;</code>	
35	<code> }</code>	
36	<code> if(this->texture != nullptr){</code>	
37	<code> SDL_QueryTexture(this->texture, NULL, NULL, &w, &h);</code>	
38	<code> SRC = {0,0,w,h};</code>	
39	<code> SDL_Rect dest = {button.x, button.y-button.h,w*2,h*2};</code>	
40	<code> SDL_RenderCopy(this->renderer, this->texture, &src, &dest);</code>	
41	<code> }</code>	
42	<code>}</code>	
43		
44	<code>void ArrowButton::setViewport(SDL_Rect viewport){</code>	
45	<code> Button::setViewport(viewport);</code>	
46	<code>}</code>	
47		
48	<code>ArrowButton::~~ArrowButton(){}</code>	

jul 21, 20 15:20	Tile.h	Page 1/1
1	<code>#ifndef ARGENTUM_TILE_H</code>	
2	<code>#define ARGENTUM_TILE_H</code>	
3		
4		
5	<code>#include "Texture.h"</code>	
6	<code>#include "Camera.h"</code>	
7	<code>#include "../common/Point.h"</code>	
8		
9	<code>class Tile {</code>	
10	<code>private:</code>	
11	<code>int x;</code>	
12	<code>int y;</code>	
13	<code>int tx;</code>	
14	<code>int ty;</code>	
15	<code>int width;</code>	
16	<code>int height;</code>	
17	<code>Texture& texture;</code>	
18	<code>SDL_Rect src, dest;</code>	
19	<code>public:</code>	
20	<code>Tile(int x, int y, int tx, int ty, int width, int height, Texture &texture);</code>	
21		
22	<code>virtual ~Tile();</code>	
23		
24	<code>void destinyAdjust(Camera& camera);</code>	
25		
26	<code>void draw(SDL_Renderer&, Camera& camera);</code>	
27		
28	<code>};</code>	
29		
30		
31	<code>#endif //ARGENTUM_TILE_H</code>	

jul 21, 20 15:20	Tile.cpp	Page 1/1
1	<code>#include "Tile.h"</code>	
2		
3	<code>Tile::Tile(int x, int y, int tx, int ty,</code>	
4	<code>int width, int height, Texture &texture) :</code>	
5	<code>x(x), y(y), tx(tx), ty(ty),</code>	
6	<code>width(width), height(height),</code>	
7	<code>texture(texture) {}</code>	
8		
9	<code>void Tile::draw(SDL_Renderer &renderer, Camera& camera) {</code>	
10	<code>src.w = width;</code>	
11	<code>src.h = height;</code>	
12	<code>src.x = tx;</code>	
13	<code>src.y = ty;</code>	
14	<code>destinyAdjust(camera);</code>	
15	<code>texture.render(src, dest);</code>	
16	<code>}</code>	
17		
18	<code>void Tile::destinyAdjust(Camera& camera) {</code>	
19	<code>this->dest.w = width*camera.getScale();</code>	
20	<code>this->dest.h = height*camera.getScale();</code>	
21	<code>this->dest.x = (x - camera.getCameraPosition().x)-(width/2)*camera.getScale(</code>	
22	<code>);</code>	
23	<code>this->dest.y = (y - camera.getCameraPosition().y)-(height/2)*camera.getScale</code>	
24	<code>();</code>	
25	<code>}</code>	
26	<code>Tile::~Tile() = default;</code>	

jul 21, 20 15:20	TextureManager.h	Page 1/1
1	#ifndef TEXTUREMANAGER_H	
2	#define TEXTUREMANAGER_H	
3	#include "Texture.h"	
4	#include "TextureID.h"	
5	#include <unordered_map>	
6	#include <string>	
7	#include "../common/Identificators.h"	
8		
9	class SDL_Renderer;	
10	class SDL_Color;	
11		
12	class TextureManager {	
13	private:	
14	std::unordered_map<TextureID, Texture, std::hash<TextureID>> textures;	
15	SDL_Renderer& renderer;	
16		
17	public:	
18	explicit TextureManager(SDL_Renderer& renderer);	
19		
20	void loadTextures();	
21		
22	void createTexture(TextureID id, const std::string& path, SDL_Color color);	
23	void createTexture(TextureID id, const std::string& path) ;	
24		
25	void dropTexture(TextureID id);	
26		
27	const Texture& getTexture(TextureID id) const ;	
28		
29	const Texture& getTexture(ItemsInventoryID id) const ;	
30	const Texture& getTexture(BodyID id) const ;	
31	const Texture& getTexture(HeadID id) const ;	
32	const Texture& getTexture(WeaponID id) const ;	
33	const Texture& getTexture(ShieldID id) const ;	
34	const Texture& getTexture(HelmetID id) const ;	
35	~TextureManager();	
36		
37	TextureManager(const TextureManager&) = delete;	
38	TextureManager &operator=(const TextureManager&) = delete;	
39	};	
40		
41	#endif	

jul 21, 20 15:20	TextureManager.cpp	Page 1/6
1	#include "TextureManager.h"	
2	#include <utility>	
3		
4	TextureManager::TextureManager(SDL_Renderer& renderer) : textures(),	
5	renderer(renderer) {}	
6		
7	void TextureManager::createTexture(TextureID id, const std::string& path) {	
8	Texture newTexture(path, this →renderer);	
9	this →textures.insert(std::make_pair(id, std::move(newTexture)));	
10	}	
11		
12	void TextureManager::createTexture(TextureID id, const std::string& path, SDL_Color color) {	
13	Texture newTexture(path, this →renderer, color);	
14	this →textures.insert(std::make_pair(id, std::move(newTexture)));	
15	}	
16		
17	void TextureManager::dropTexture(TextureID id) {	
18	this →textures.erase(id);	
19	}	
20		
21	const Texture& TextureManager::getTexture(TextureID id) const {	
22	return this →textures.at(id);	
23	}	
24		
25	const Texture& TextureManager::getTexture(BodyID id) const {	
26	TextureID idItem = TextureID::ItemNothing;	
27	switch(id) {	
28	case BodyID::BlueCommon:	
29	idItem = TextureID::ItemBlueCommonBody;	
30	break;	
31	case BodyID::GreenCommon:	
32	idItem = TextureID::ItemGreenCommonBody;	
33	break;	
34	case BodyID::RedCommon:	
35	idItem = TextureID::ItemRedCommonBody;	
36	break;	
37	case BodyID::LeatherArmor:	
38	idItem = TextureID::ItemLeatherArmor;	
39	break;	
40	case BodyID::BlueTunic:	
41	idItem = TextureID::ItemBlueTunic;	
42	break;	
43	case BodyID::PlateArmor:	
44	idItem = TextureID::ItemPlateArmor;	
45	break;	
46	case BodyID::Ghost:	
47	idItem = TextureID::ItemNothing;	
48	break;	
49	default:	
50	idItem = TextureID::ItemNothing;	
51	break;	
52	}	
53	const Texture& texture = getTexture(idItem);	
54	return texture;	
55	}	
56		
57	const Texture& TextureManager::getTexture(HeadID id) const {	
58	TextureID idItem = TextureID::ItemNothing;	
59	switch(id) {	
60	case HeadID::Elf:	
61	idItem = TextureID::ElfHead;	
62	break;	
63	case HeadID::Gnome:	
64	idItem = TextureID::GnomeHead;	
65	break;	

jul 21, 20 15:20	TextureManager.cpp	Page 2/6
66	case HeadID::Human:	
67	idItem = TextureID::HumanHead;	
68	break;	
69	case HeadID::Dwarf:	
70	idItem = TextureID::DwarfHead;	
71	break;	
72	case HeadID::Nothing:	
73	idItem = TextureID::ItemNothing;	
74	break;	
75	default:	
76	idItem = TextureID::ItemNothing;	
77	break;	
78	}	
79	const Texture& texture = getTexture(idItem);	
80	return texture;	
81	}	
82	const Texture& TextureManager::getTexture(HelmetID id) const {	
83	TextureID idItem = TextureID::ItemNothing;	
84	switch(id) {	
85	case HelmetID::Hood:	
86	idItem = TextureID::ItemHood;	
87	break;	
88	case HelmetID::MagicHat:	
89	idItem = TextureID::ItemMagicHat;	
90	break;	
91	case HelmetID::IronHelmet:	
92	idItem = TextureID::ItemIronHelmet;	
93	break;	
94	case HelmetID::Nothing:	
95	idItem = TextureID::ItemNothing;	
96	break;	
97	default:	
98	idItem = TextureID::ItemNothing;	
99	break;	
100	}	
101	const Texture& texture = getTexture(idItem);	
102	return texture;	
103	}	
104		
105	const Texture& TextureManager::getTexture(WeaponID id) const {	
106	TextureID idItem = TextureID::ItemNothing;	
107	switch(id) {	
108	case WeaponID::SimpleArc:	
109	idItem = TextureID::ItemSimpleArc;	
110	break;	
111	case WeaponID::CompoundArc:	
112	idItem = TextureID::ItemCompoundArc;	
113	break;	
114	case WeaponID::LongSword:	
115	idItem = TextureID::ItemLongSword;	
116	break;	
117	case WeaponID::Hammer:	
118	idItem = TextureID::ItemHammer;	
119	break;	
120	case WeaponID::Ax:	
121	idItem = TextureID::ItemAx;	
122	break;	
123	case WeaponID::ElficFlaute:	
124	idItem = TextureID::ItemElficFlaute;	
125	break;	
126	case WeaponID::AshStick:	
127	idItem = TextureID::ItemAshStick;	
128	break;	
129	case WeaponID::Crosier:	
130	idItem = TextureID::ItemCrosier;	
131		

jul 21, 20 15:20	TextureManager.cpp	Page 3/6
132	break;	
133	case WeaponID::GnarledStick:	
134	idItem = TextureID::ItemGnarledStick;	
135	break;	
136	case WeaponID::Nothing:	
137	idItem = TextureID::ItemNothing;	
138	break;	
139	default:	
140	idItem = TextureID::ItemNothing;	
141	break;	
142	}	
143	const Texture& texture = getTexture(idItem);	
144	return texture;	
145	}	
146		
147	const Texture& TextureManager::getTexture(ShieldID id) const {	
148	TextureID idItem = TextureID::ItemNothing;	
149	switch(id) {	
150	case ShieldID::IronShield:	
151	idItem = TextureID::ItemIronShield;	
152	break;	
153	case ShieldID::TurtleShield:	
154	idItem = TextureID::ItemTurtleShield;	
155	break;	
156	case ShieldID::Nothing:	
157	idItem = TextureID::ItemNothing;	
158	break;	
159	}	
160	const Texture& texture = getTexture(idItem);	
161	return texture;	
162	}	
163		
164	const Texture& TextureManager::getTexture(ItemsInventoryID id) const {	
165	TextureID idText = TextureID::ItemNothing;	
166	switch(id) {	
167	case ItemsInventoryID::SimpleArc:	
168	idText = TextureID::ItemSimpleArc;	
169	break;	
170	case ItemsInventoryID::CompoundArc:	
171	idText = TextureID::ItemCompoundArc;	
172	break;	
173	case ItemsInventoryID::LongSword:	
174	idText = TextureID::ItemLongSword;	
175	break;	
176	case ItemsInventoryID::Hammer:	
177	idText = TextureID::ItemHammer;	
178	break;	
179	case ItemsInventoryID::Ax:	
180	idText = TextureID::ItemAx;	
181	break;	
182	case ItemsInventoryID::ElficFlaute:	
183	idText = TextureID::ItemElficFlaute;	
184	break;	
185	case ItemsInventoryID::AshStick:	
186	idText = TextureID::ItemAshStick;	
187	break;	
188	case ItemsInventoryID::Crosier:	
189	idText = TextureID::ItemCrosier;	
190	break;	
191	case ItemsInventoryID::GnarledStick:	
192	idText = TextureID::ItemGnarledStick;	
193	break;	
194	case ItemsInventoryID::IronShield:	
195	idText = TextureID::ItemIronShield;	
196	break;	
197	case ItemsInventoryID::TurtleShield:	

jul 21, 20 15:20	TextureManager.cpp	Page 4/6
198	idText = TextureID::ItemTurtleShield;	
199	break;	
200	case ItemsInventoryID::Hood:	
201	idText = TextureID::ItemHood;	
202	break;	
203	case ItemsInventoryID::MagicHat:	
204	idText = TextureID::ItemMagicHat;	
205	break;	
206	case ItemsInventoryID::IronHelmet:	
207	idText = TextureID::ItemIronHelmet;	
208	break;	
209	case ItemsInventoryID::BlueCommon:	
210	idText = TextureID::ItemBlueCommonBody;	
211	break;	
212	case ItemsInventoryID::GreenCommon:	
213	idText = TextureID::ItemGreenCommonBody;	
214	break;	
215	case ItemsInventoryID::RedCommon:	
216	idText = TextureID::ItemRedCommonBody;	
217	break;	
218	case ItemsInventoryID::LeatherArmor:	
219	idText = TextureID::ItemLeatherArmor;	
220	break;	
221	case ItemsInventoryID::BlueTunic:	
222	idText = TextureID::ItemBlueTunic;	
223	break;	
224	case ItemsInventoryID::PlateArmor:	
225	idText = TextureID::ItemPlateArmor;	
226	break;	
227	case ItemsInventoryID::HealthPotion:	
228	idText = TextureID::HealthPotion;	
229	break;	
230	case ItemsInventoryID::ManaPotion:	
231	idText = TextureID::ManaPotion;	
232	break;	
233	case ItemsInventoryID::Gold:	
234	idText = TextureID::Gold;	
235	break;	
236	case ItemsInventoryID::Nothing:	
237	idText = TextureID::ItemNothing;	
238	break;	
239	default:	
240	idText = TextureID::ItemNothing;	
241	break;	
242	}	
243	const Texture& texture = getTexture(idText);	
244	return texture;	
245	}	
246		
247	void TextureManager::loadTextures() {	
248	SDL_Color textColor = {0x0, 0x0, 0x0};	
249	std::string path(ROOT_DIR);	
250	createTexture(TextureID::PresentationImage, path + "/assets/img/ImagenPresentacion.jpg")	
251	;	
252	createTexture(TextureID::LobbyBackground, path + "/assets/img/Fondo Inicio.jpg");	
253	createTexture(TextureID::ZombieHead, path + "/assets/img/Zombie Cabeza.png", textColor)	
254	;	
255	createTexture(TextureID::ElfHead, path + "/assets/img/Cabeza Elfo.png", textColor);	
256	createTexture(TextureID::HumanHead, path + "/assets/img/Cabeza Humano.png", textColor)	
257	;	
258	createTexture(TextureID::DwarfHead, path + "/assets/img/Cabeza Enano.png", textColor);	
259	createTexture(TextureID::GnomeHead, path + "/assets/img/Cabeza Gnomo.png", textColor)	
	;	
	createTexture(TextureID::PriestHead, path + "/assets/img/Sacerdote Cabeza Sprite.png", textColor);	
	createTexture(TextureID::ZombieBody, path + "/assets/img/Zombie Cuerpo Sprite.png", textColor);	

jul 21, 20 15:20	TextureManager.cpp	Page 5/6
259	olor);	
260	createTexture(TextureID::BlueTunic, path + "/assets/img/Tunica Azul Sprite.png", textColor);	
261	createTexture(TextureID::PriestBody, path + "/assets/img/Sacerdote Sprite.png", textColor);	
262	createTexture(TextureID::LeatherArmor, path + "/assets/img/Armadura de Cuero Sprite.png", textColor);	
263	createTexture(TextureID::PlateArmor, path + "/assets/img/Armadura de Placas Sprite.png", textColor);	
264	createTexture(TextureID::BlueCommonBody, path + "/assets/img/Vestimenta Comun azul Sprite.png", textColor);	
265	createTexture(TextureID::GreenCommonBody, path + "/assets/img/Vestimenta Comun verde Sprite.png", textColor);	
266	createTexture(TextureID::RedCommonBody, path + "/assets/img/Vestimenta Comun roja Sprite.png", textColor);	
267	createTexture(TextureID::TurtleShield, path + "/assets/img/Escudo de Tortuga Sprite.png", textColor);	
268	createTexture(TextureID::IronShield, path + "/assets/img/Escudo de Hierro Sprite.png", textColor);	
269	createTexture(TextureID::IronHelmet, path + "/assets/img/Casco de Hierro Sprite.png", textColor);	
270	createTexture(TextureID::Hood, path + "/assets/img/Capucha Sprite.png", textColor);	
271	createTexture(TextureID::MagicHat, path + "/assets/img/Sombrero Magico Sprite.png", textColor);	
272	createTexture(TextureID::AshStick, path + "/assets/img/Vara de Fresno Sprite.png", textColor);	
273	createTexture(TextureID::LongSword, path + "/assets/img/Espada Larga Sprite.png", textColor);	
274	createTexture(TextureID::Hammer, path + "/assets/img/Martillo Sprite.png", textColor);	
275	createTexture(TextureID::SimpleArc, path + "/assets/img/Arco Simple Sprite.png", textColor);	
276	createTexture(TextureID::CompoundArc, path + "/assets/img/Arco Compuesto Sprite.png", textColor);	
277	createTexture(TextureID::GnarledStick, path + "/assets/img/Baston Nudoso Sprite.png", textColor);	
278	createTexture(TextureID::Crosier, path + "/assets/img/Baculo Engarzado Sprite.png", textColor);	
279	createTexture(TextureID::Ax, path + "/assets/img/Hacha Sprite.png", textColor);	
280	createTexture(TextureID::ElficFlaute, path + "/assets/img/Flauta Elfica Sprite.png", textColor);	
281	createTexture(TextureID::Spider, path + "/assets/img/Araña Sprite.png", textColor);	
282	createTexture(TextureID::Skeleton, path + "/assets/img/Esqueleto Sprite.png", textColor);	
283	createTexture(TextureID::Banker, path + "/assets/img/Banquero Sprite.png", textColor);	
284	createTexture(TextureID::Merchant, path + "/assets/img/Comerciante Sprite.png", textColor);	
285	createTexture(TextureID::Goblin, path + "/assets/img/Goblin.png", textColor);	
286	createTexture(TextureID::Ghost, path + "/assets/img/Fantasma Sprite.png", textColor);	
287	createTexture(TextureID::ManaPotion, path + "/assets/img/Pocion Mana.png", textColor);	
288	createTexture(TextureID::HealthPotion, path + "/assets/img/Pocion Vida.png", textColor);	
289	createTexture(TextureID::Gold, path + "/assets/img/Oro.png", textColor);	
290	createTexture(TextureID::ItemHood, path + "/assets/img/Capucha.png", textColor);	
291	createTexture(TextureID::ItemIronHelmet, path + "/assets/img/Casco de Hierro.png", textColor);	
292	createTexture(TextureID::ItemMagicHat, path + "/assets/img/Sombrero Magico.png", textColor);	
293	createTexture(TextureID::ItemIronShield, path + "/assets/img/Escudo de Hierro.png", textColor);	
294	createTexture(TextureID::ItemTurtleShield, path + "/assets/img/Escudo de Tortuga.png", textColor);	
295	createTexture(TextureID::ItemAx, path + "/assets/img/Hacha.png", textColor);	
296	createTexture(TextureID::ItemSimpleArc, path + "/assets/img/Arco Simple.png", textColor);	
	createTexture(TextureID::ItemCompoundArc, path + "/assets/img/Arco Compuesto.png", textColor);	

jul 21, 20 15:20	TextureManager.cpp	Page 6/6
297	xtColor);	
298	createTexture(TextureID::ItemHammer, path + "/assets/img/Martillo.png", textColor);	
299	createTexture(TextureID::ItemLongSword, path + "/assets/img/Espada Larga.png", textColor);	
299	createTexture(TextureID::ItemAshStick, path + "/assets/img/Vara de Fresno.png", textColor);	
300	createTexture(TextureID::ItemCrosier, path + "/assets/img/Baculo Engarzado.png", textColor);	
301	createTexture(TextureID::ItemGnarledStick, path + "/assets/img/Baston Nudoso.png", textColor);	
302	createTexture(TextureID::ItemElficFlaute, path + "/assets/img/Flauta Elfica.png", textColor);	
303	createTexture(TextureID::ItemGreenCommonBody, path + "/assets/img/Vestimenta Comun verde.png", textColor);	
304	createTexture(TextureID::ItemBlueCommonBody, path + "/assets/img/Vestimenta Comun azul.png", textColor);	
305	createTexture(TextureID::ItemRedCommonBody, path + "/assets/img/Vestimenta Comun roja.png", textColor);	
306	createTexture(TextureID::ItemBlueTunic, path + "/assets/img/Tunica Azul.png", textColor);	
307	createTexture(TextureID::ItemLeatherArmor, path + "/assets/img/Armadura de Cuero.png", textColor);	
308	createTexture(TextureID::ItemPlateArmor, path + "/assets/img/Armadura de Placas.png", textColor);	
309	createTexture(TextureID::TopBar, path + "/assets/img/Fondo Barra Superior.jpg");	
310	createTexture(TextureID::Star, path + "/assets/img/Estrella.jpg", textColor);	
311	createTexture(TextureID::StatsBackground, path + "/assets/img/StatsBackground.jpg");	
312	createTexture(TextureID::ItemNothing, path + "/assets/img/RecuadroItems.jpg");	
313	createTexture(TextureID::MeditateEffect, path + "/assets/img/Efecto Meditar.png", textColor);	
314	createTexture(TextureID::HitEffect, path + "/assets/img/Efecto Golpe.png", textColor);	
315	createTexture(TextureID::MagicArrowEffect, path + "/assets/img/Efecto Flecha Magica Sprite.png", textColor);	
316	createTexture(TextureID::MissileEffect, path + "/assets/img/Efecto Misil Sprite.png", textColor);	
317	createTexture(TextureID::ExplosionEffect, path + "/assets/img/Efecto Explosion Sprite.png", textColor);	
318	createTexture(TextureID::HealthEffect, path + "/assets/img/Efecto Curar Sprite.png", textColor);	
319	createTexture(TextureID::Button, path + "/assets/img/Boton.bmp");	
320	}	
321		
322	TextureManager::~TextureManager() = default;	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	TextureID.h	Page 1/2
1	#ifndef TEXTUREID_H	
2	#define TEXTUREID_H	
3		
4	enum class TextureID {	
5	LobbyBackground,	
6	PresentationImage,	
7	TopBar,	
8	StatsBackground,	
9	Button,	
10	Star,	
11	ZombieHead,	
12	ElfHead,	
13	HumanHead,	
14	DwarfHead,	
15	GnomeHead,	
16	PriestHead,	
17	ZombieBody,	
18	PriestBody,	
19	BlueTunic,	
20	LeatherArmor,	
21	PlateArmor,	
22	BlueCommonBody,	
23	GreenCommonBody,	
24	RedCommonBody,	
25	TurtleShield,	
26	IronShield,	
27	IronHelmet,	
28	MagicHat,	
29	Hood,	
30	AshStick,	
31	SimpleArc,	
32	CompoundArc,	
33	LongSword,	
34	Hammer,	
35	Crosier,	
36	GnarledStick,	
37	Ax,	
38	ElficFlaute,	
39	Spider,	
40	Skeleton,	
41	Goblin,	
42	Ghost,	
43	Merchant,	
44	Banker,	
45	HealthEffect,	
46	ExplosionEffect,	
47	MagicArrowEffect,	
48	MissileEffect,	
49	MeditateEffect,	
50	HitEffect,	
51	ManaPotion,	
52	HealthPotion,	
53	Gold,	
54	ItemSimpleArc,	
55	ItemCompoundArc,	
56	ItemLongSword,	
57	ItemHammer,	
58	ItemCrosier,	
59	ItemGnarledStick,	
60	ItemAx,	
61	ItemAshStick,	
62	ItemElficFlaute,	
63	ItemTurtleShield,	
64	ItemIronShield,	
65	ItemIronHelmet,	
66	ItemMagicHat,	

135/217

jul 21, 20 15:20	TextureID.h	Page 2/2
67	ItemHood,	
68	ItemBlueTunic,	
69	ItemLeatherArmor,	
70	ItemPlateArmor,	
71	ItemBlueCommonBody,	
72	ItemGreenCommonBody,	
73	ItemRedCommonBody,	
74	ItemNothing,	
75	};	
76		
77	#endif	

jul 21, 20 15:20	Texture.h	Page 1/1
1	#ifndef TEXTURE_H	
2	#define TEXTURE_H	
3	#include <string>	
4	#include <SDL2/SDL.h>	
5		
6	class Texture {	
7	private:	
8	SDL_Texture* texture = nullptr;	
9	int width;	
10	int height;	
11	SDL_Renderer* renderer = nullptr;	
12	SDL_Texture* loadTexture(const std::string &fileName, SDL_Color colorKey);	
13		
14	public:	
15	Texture(const std::string &fileName, SDL_Renderer& renderer, SDL_Color colorKey);	
16	Texture(const std::string &fileName, SDL_Renderer& renderer);	
17	Texture(Texture^ other);	
18	~Texture();	
19		
20	int render(const SDL_Rect& source, const SDL_Rect& destiny) const;	
21		
22	int render() const;	
23		
24	int getWidth() const;	
25		
26	SDL_Texture* getTexture() const;	
27	};	
28		
29	#endif	

jul 21, 20 15:20	Texture.cpp	Page 1/2
1	<code>#include "Texture.h"</code>	
2	<code>#include "Window.h"</code>	
3	<code>#include <SDL2/SDL_image.h></code>	
4	<code>#include "../common/Exception.h"</code>	
5	<code>#include <iostream></code>	
6		
7	<code>Texture::Texture(const std::string &fileName, SDL_Renderer& renderer, SDL_Color colorKey) {</code>	
8	<code> this->renderer = &renderer;</code>	
9	<code> this->texture = loadTexture(fileName, colorKey);</code>	
10	<code>}</code>	
11		
12	<code>Texture::Texture(const std::string &fileName, SDL_Renderer& renderer) {</code>	
13	<code> this->renderer = &renderer;</code>	
14	<code> if (!this->texture) {</code>	
15	<code> this->texture = IMG_LoadTexture(this->renderer, fileName.c_str());</code>	
16	<code> }</code>	
17	<code> if (!this->texture) {</code>	
18	<code> throw Exception("Fail IMG_LoadTexture: %s ", IMG_GetError());</code>	
19	<code> }</code>	
20	<code> int h,w;</code>	
21	<code> SDL_QueryTexture(this->texture, NULL, NULL, &w, &h);</code>	
22	<code> this->width = w;</code>	
23	<code> this->height = h;</code>	
24	<code>}</code>	
25		
26	<code>Texture::Texture(Texture &other) {</code>	
27	<code> std::swap(this->texture, other.texture);</code>	
28	<code> std::swap(this->renderer, other.renderer);</code>	
29	<code> std::swap(this->height, other.height);</code>	
30	<code> std::swap(this->width, other.width);</code>	
31	<code>}</code>	
32		
33		
34	<code>SDL_Texture* Texture::loadTexture(const std::string &fileName, SDL_Color colorKey) {</code>	
35	<code> SDL_Surface* surface = IMG_Load(fileName.c_str());</code>	
36	<code> if (!surface) {</code>	
37	<code> throw Exception("Fail IMG_Load: %s", IMG_GetError());</code>	
38	<code> }</code>	
39	<code> SDL_SetColorKey(surface, SDL_TRUE,</code>	
40	<code> SDL_MapRGB(surface->format, colorKey.r, colorKey.g, colorKey.b));</code>	
41	<code> this->width = surface->w;</code>	
42	<code> this->height = surface->h;</code>	
43	<code> this->texture = SDL_CreateTextureFromSurface(this->renderer, surface);</code>	
44	<code> if (!this->texture) {</code>	
45	<code> throw Exception("Fail SDL_CreateTextureFromSurface: %s", SDL_GetError());</code>	
46	<code> }</code>	
47	<code> SDL_FreeSurface(surface);</code>	
48	<code> return this->texture;</code>	
49	<code>}</code>	
50		
51		
52	<code>int Texture::render() const {</code>	
53	<code> return SDL_RenderCopy(this->renderer, this->texture, NULL, NULL);</code>	
54	<code>}</code>	
55		
56	<code>int Texture::render(const SDL_Rect& source, const SDL_Rect& destiny) const {</code>	
57	<code> return SDL_RenderCopy(this->renderer, this->texture, &source, &destiny);</code>	
58	<code>}</code>	
59		
60	<code>Texture::~Texture() {</code>	
61	<code> if (this->texture) {</code>	
62	<code> SDL_DestroyTexture(this->texture);</code>	
63	<code> this->texture = nullptr;</code>	
64	<code>}</code>	

jul 21, 20 15:20	Texture.cpp	Page 2/2
65	<code>}</code>	
66		
67	<code>int Texture::getWidth() const {</code>	
68	<code> return width;</code>	
69	<code>}</code>	
70		
71	<code>SDL_Texture* Texture::getTexture() const {</code>	
72	<code> return this->texture;</code>	
73	<code>}</code>	

jul 21, 20 15:20	Receiver.h	Page 1/1
1	<code>#ifndef RECEIVER_H</code>	
2	<code>#define RECEIVER_H</code>	
3		
4	<code>#include "../common/Thread.h"</code>	
5	<code>#include "../common/DataQueue.h"</code>	
6	<code>#include "../common/CommunicationProtocol.h"</code>	
7	<code>class Receiver: public Thread {</code>	
8	<code>private:</code>	
9	<code> DataQueue& queue;</code>	
10	<code> std::atomic<bool> keepTalking;</code>	
11	<code> CommunicationProtocol& protocol;</code>	
12	<code>public:</code>	
13		
14	<code> Receiver(CommunicationProtocol& protocol, DataQueue& queue);</code>	
15	<code> virtual void run();</code>	
16		
17	<code> bool is_alive() const;</code>	
18		
19	<code> void stop();</code>	
20		
21	<code> virtual ~Receiver();</code>	
22	<code>};</code>	
23	<code>#endif</code>	
24		
25		
26		
27		

jul 21, 20 15:20	Receiver.cpp	Page 1/1
1	<code>#include "Receiver.h"</code>	
2	<code>#include <vector></code>	
3	<code>#include <iostream></code>	
4	<code>#include "../common/Message.h"</code>	
5	<code>#include "../common/SocketException.h"</code>	
6		
7	<code>#define UNKNOWN_ERROR "Unknow Error"</code>	
8	<code>#define ERRORSOCKET "Error en la comunicaci3n en Receiver::run() "</code>	
9	<code>#define ERRORRECEIVER "Error en Receiver::run() "</code>	
10		
11	<code>Receiver::Receiver(CommunicationProtocol& protocol, DataQueue& queue) :</code>	
12	<code> queue(queue), keepTalking(true), protocol(protocol) {}</code>	
13		
14	<code>void Receiver::run() {</code>	
15	<code> Message msg;</code>	
16	<code> while (this->keepTalking){</code>	
17	<code> try {</code>	
18	<code> msg = this->protocol.receive();</code>	
19	<code> this->queue.push(msg);</code>	
20	<code> } catch (const SocketException& e) {</code>	
21	<code> std::cerr << ERRORSOCKET << e.what() << std::endl;</code>	
22	<code> this->keepTalking = false;</code>	
23	<code> } catch (const std::exception& e) {</code>	
24	<code> std::cerr << ERRORRECEIVER << e.what() << std::endl;</code>	
25	<code> this->keepTalking = false;</code>	
26	<code> } catch (...) {</code>	
27	<code> this->keepTalking = false;</code>	
28	<code> std::cerr << UNKNOWN_ERROR << std::endl;</code>	
29	<code> }</code>	
30	<code> }</code>	
31	<code>}</code>	
32		
33	<code>bool Receiver::is_alive() const {</code>	
34	<code> return this->keepTalking;</code>	
35	<code>}</code>	
36		
37	<code>void Receiver::stop() {</code>	
38	<code> this->keepTalking = false;</code>	
39	<code> this->protocol.stop();</code>	
40	<code>}</code>	
41		
42	<code>Receiver::~Receiver() {}</code>	

jul 21, 20 15:20	Presentation.h	Page 1/1
1	<code>#ifndef PRESENTATION_H</code>	
2	<code>#define PRESENTATION_H</code>	
3		
4	<code>#include "Window.h"</code>	
5	<code>#include "TextureManager.h"</code>	
6		
7	<code>class Presentation {</code>	
8	<code>private:</code>	
9	<code>Window& window;</code>	
10	<code>TextureManager& manager;</code>	
11	<code>public:</code>	
12	<code>Presentation(Window& window, TextureManager& manager);</code>	
13		
14	<code>bool run();</code>	
15		
16	<code>~Presentation();</code>	
17	<code>};</code>	
18		
19		
20	<code>#endif</code>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Presentation.cpp	Page 1/1
1	<code>#include "Presentation.h"</code>	
2	<code>#include <SDL2/SDL.h></code>	
3	<code>#include "Texture.h"</code>	
4	<code>#include "Font.h"</code>	
5		
6	<code>Presentation::Presentation(Window& window, TextureManager& manager) :</code>	
7	<code>window(window), manager(manager) {}</code>	
8		
9	<code>bool Presentation::run() {</code>	
10	<code>bool quit = false;</code>	
11	<code>bool closeProgram = false;</code>	
12	<code>int width_text, height_text;</code>	
13	<code>SDL_Event event;</code>	
14	<code>SDL_Rect dest;</code>	
15	<code>const Texture& fondo = this->manager.getTexture(TextureID::PresentationImage</code>	
16	<code>);</code>	
17	<code>std::string fuente = "assets/font/Prince Valiant.ttf";</code>	
18	<code>SDL_Color textColor = {0xFF, 0xFF, 0xFF};</code>	
19	<code>Font font(fuente, 18, textColor);</code>	
20	<code>font.setColor(textColor);</code>	
21	<code>SDL_Texture* messageTexture = font.createText("Presionar ENTER para continuar",</code>	
22	<code>&(window.getRenderer()), &width_text, &height_text);</code>	
23		
24	<code>while (!quit) {</code>	
25		
26	<code>while (SDL_PollEvent(&event) != 0) {</code>	
27	<code>if (event.type == SDL_QUIT) {</code>	
28	<code>quit = true;</code>	
29	<code>closeProgram = true;</code>	
30	<code>} else if (event.type == SDL_KEYDOWN) {</code>	
31	<code>if (event.key.keysym.sym == SDLK_RETURN){</code>	
32	<code>quit = true;</code>	
33	<code>}</code>	
34	<code>}</code>	
35	<code>window.clearScreen();</code>	
36	<code>fondo.render();</code>	
37	<code>dest = {(window.getWidth() - width_text) / 2, (window.getHeight() - height_text) * 6/7, width_text, height_text};</code>	
38	<code>SDL_RenderCopy(&(window.getRenderer()), messageTexture, NULL, &dest);</code>	
39	<code>window.render();</code>	
40	<code>}</code>	
41	<code>}</code>	
42	<code>font.deleteText(messageTexture);</code>	
43	<code>return closeProgram;</code>	
44	<code>}</code>	
45		
46	<code>Presentation::~Presentation(){}</code>	

139/217

jul 21, 20 15:20	Player.h	Page 1/1
1	#ifndef PLAYER_H	
2	#define PLAYER_H	
3		
4	#include "Character.h"	
5	#include "Camera.h"	
6	#include "TextureManager.h"	
7	#include "MusicManager.h"	
8	#include "../common/Point.h"	
9	#include "characterStates/CharacterState.h"	
10	#include "../common/Identificators.h"	
11	#include "../common/PlayerInfo.h"	
12	#include <memory>	
13	#include <string>	
14		
15	union SDL_Event;	
16		
17		
18	class Player : public Character {	
19	private :	
20	Point center;	
21	PlayerInfo playerInfo;	
22	std::shared_ptr<CharacterState> state = nullptr;	
23	bool playLowLife{false};	
24		
25	void setState(CharacterStateID newState);	
26	void playEffectLowLife();	
27		
28	public :	
29	Player(const TextureManager& manager, const PlayerInfo& playerInfo, const MusicManager& mixer);	
30		
31	virtual void render(Camera& camera);	
32	virtual void update(double dt);	
33		
34	void updatePlayerInfo(const PlayerInfo& info);	
35		
36	InputInfo handleEvent(SDL_Event& event, Camera& camera);	
37	InputInfo dropItem(int itemNumber);	
38	InputInfo selectItem(int itemNumber);	
39	InputInfo resurrect();	
40	InputInfo cure();	
41	InputInfo buy(int itemNumber);	
42	InputInfo sell(int itemNumber);	
43	InputInfo deposit(int information, bool isItem);	
44	InputInfo retire(int information, bool isItem);	
45	InputInfo unequipItem(int itemNumber);	
46	Point* getCenter();	
47		
48	uint getLevel() const ;	
49	uint getHealth() const ;	
50	uint getMana() const ;	
51	uint getGold() const ;	
52	uint getMaxMana() const ;	
53	uint getMaxHealth() const ;	
54	uint getSafeGold() const ;	
55	uint getExp() const ;	
56	uint getMaxExp() const ;	
57	std::string getInventory() const ;	
58	CharacterStateID& getState() const ;	
59	PlayerInfo getInfo() const ;	
60		
61	~Player();	
62	};	
63		
64	#endif	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Player.cpp	Page 1/5
1	#include "Player.h"	
2	#include <SDL2/SDL.h>	
3	#include <iostream>	
4	#include "characterStates/StillState.h"	
5	#include "characterStates/MoveState.h"	
6	#include "characterStates/MeditateState.h"	
7	#include "characterStates/InteractState.h"	
8	#include "characterStates/AttackState.h"	
9	#include "characterStates/ResurrectState.h"	
10	#include "Items/Animation.h"	
11	#include "Items/MeditateAnimation.h"	
12		
13		
14	Player::Player(const TextureManager& manager, const PlayerInfo& playerInfo,	
15	const MusicManager& mixer) : Character(playerInfo.getX(),playerInfo.getY(),	
16	playerInfo.getId(),manager,mixer), center(playerInfo.getX(),playerInfo.getY())	
17	{	
18	playerInfo(playerInfo){	
19	this →direction = playerInfo.getDirection();	
20	this →frameHead = 0;	
21	this →state = std::shared_ptr<CharacterState>(new StillState());	
22	setArmor(playerInfo.getBodyID());	
23	setHead(playerInfo.getHeadID());	
24	setHelmet(playerInfo.getHelmetID());	
25	setShield(playerInfo.getShieldID());	
26	setWeapon(playerInfo.getWeaponID());	
27	}	
28		
29	void Player::render(Camera& camera) {	
30	Character::render(camera);	
31	playEffectLowLife();	
32	}	
33		
34		
35	void Player::update(double dt) {	
36	if (this →state ≠ nullptr ^	
37	(this →state→getState() == CharacterStateID::Move v this →state→getSta	
38	te() == CharacterStateID::Resurrect)) {	
39	Point aux(posX, posY);	
40	this →center = aux;	
41	this →body→update(dt,direction);	
42	if (this →weapon ≠ nullptr)	
43	this →weapon→update(dt,direction);	
44	if (this →shield ≠ nullptr)	
45	this →shield→update(dt,direction);	
46	}	
47	if (this →animation ≠ nullptr)	
48	this →animation→update();	
49	}	
50	void Player::updatePlayerInfo(const PlayerInfo& info) {	
51	this →posX = info.getX();	
52	this →posY = info.getY();	
53	this →direction = info.getDirection();	
54	this →playerInfo = info;	
55	setState(info.getState());	
56	setArmor(info.getBodyID());	
57	setHead(info.getHeadID());	
58	setHelmet(info.getHelmetID());	
59	setShield(info.getShieldID());	
60	setWeapon(info.getWeaponID());	
61	setFrameHead();	
62	setAnimation(info.getAttackWeapon());	
63	}	
64		

140/217

jul 21, 20 15:20	Player.cpp	Page 2/5
65	void Player::playEffectLowLife() {	
66	if(this->playerInfo.getLife() < this->playerInfo.getMaxLife()*0.1 ^	
67	this->playerInfo.getLife() != 0 ^ playLowLife) {	
68	const Effect& effect = mixer.getEffect(MusicID::Heart);	
69	effect.playEffect(-1);	
70	playLowLife = false;	
71	} else if (this->playerInfo.getLife() > this->playerInfo.getMaxLife()*0.1 ^	
72	!playLowLife ^ this->playerInfo.getLife() == 0) {	
73	const Effect& effect = mixer.getEffect(MusicID::Heart);	
74	effect.pause();	
75	playLowLife = true;	
76	}	
77	}	
78		
79	InputInfo Player::dropItem(int itemNumber) {	
80	InputInfo info = this->state->dropItem(*this, itemNumber);	
81	return info;	
82	}	
83		
84	InputInfo Player::selectItem(int itemNumber) {	
85	InputInfo info = this->state->selectItem(*this, itemNumber);	
86	return info;	
87	}	
88		
89	InputInfo Player::resurrect() {	
90	InputInfo info = this->state->resurrect(*this);	
91	return info;	
92	}	
93		
94	InputInfo Player::cure() {	
95	InputInfo info = this->state->cure(*this);	
96	return info;	
97	}	
98		
99	InputInfo Player::buy(int itemNumber) {	
100	InputInfo info = this->state->buyItem(*this, itemNumber);	
101	return info;	
102	}	
103		
104	InputInfo Player::deposit(int information, bool isItem) {	
105	InputInfo info = this->state->deposit(*this, information, isItem);	
106	return info;	
107	}	
108		
109	InputInfo Player::retire(int information, bool isItem) {	
110	InputInfo info = this->state->retire(*this, information, isItem);	
111	return info;	
112	}	
113		
114	InputInfo Player::sell(int itemNumber) {	
115	InputInfo info = this->state->sellItem(*this, itemNumber);	
116	return info;	
117	}	
118		
119	InputInfo Player::unequipItem(int itemNumber) {	
120	InputInfo info = this->state->unequipItem(*this, itemNumber);	
121	return info;	
122	}	
123		
124	InputInfo Player::handleEvent(SDL_Event& event, Camera& camera) {	
125	InputInfo input;	
126	if(event.type == SDL_KEYDOWN) {	
127	switch (event.key.keysym.sym) {	
128	case SDLK_w:	
129	input = this->state->moveUp(*this);	
130	break;	

jul 21, 20 15:20	Player.cpp	Page 3/5
131	case SDLK_s:	
132	input = this->state->moveDown(*this);	
133	break;	
134	case SDLK_a:	
135	input = this->state->moveLeft(*this);	
136	break;	
137	case SDLK_d:	
138	input = this->state->moveRight(*this);	
139	break;	
140	case SDLK_r:	
141	input = this->state->resurrect(*this);	
142	break;	
143	case SDLK_q:	
144	input = this->state->takeItem(*this);	
145	break;	
146	case SDLK_h:	
147	input = this->state->cure(*this);	
148	break;	
149	case SDLK_e:	
150	input = this->state->meditate(*this);	
151	break;	
152	case SDLK_1:	
153	input = this->state->selectItem(*this, 1);	
154	break;	
155	case SDLK_2:	
156	input = this->state->selectItem(*this, 2);	
157	break;	
158	case SDLK_3:	
159	input = this->state->selectItem(*this, 3);	
160	break;	
161	case SDLK_4:	
162	input = this->state->selectItem(*this, 4);	
163	break;	
164	case SDLK_5:	
165	input = this->state->selectItem(*this, 5);	
166	break;	
167	case SDLK_6:	
168	input = this->state->selectItem(*this, 6);	
169	break;	
170	case SDLK_7:	
171	input = this->state->selectItem(*this, 7);	
172	break;	
173	case SDLK_8:	
174	input = this->state->selectItem(*this, 8);	
175	break;	
176	case SDLK_9:	
177	input = this->state->selectItem(*this, 9);	
178	break;	
179	}	
180	} else if (event.type == SDL_KEYUP) {	
181	switch(event.key.keysym.sym) {	
182	case SDLK_w:	
183	input = this->state->stopMove(*this);	
184	break;	
185	case SDLK_s:	
186	input = this->state->stopMove(*this);	
187	break;	
188	case SDLK_a:	
189	input = this->state->stopMove(*this);	
190	break;	
191	case SDLK_d:	
192	input = this->state->stopMove(*this);	
193	break;	
194	}	
195	} else if (event.type == SDL_MOUSEBUTTONDOWN) {	
196	int x,y;	

jul 21, 20 15:20	Player.cpp	Page 4/5
197	SDL_GetMouseState(&x, &y);	
198	if (camera.clickInMap(Point(x,y))) {	
199	Point coord = camera.calculateGlobalPosition(Point(x,y));	
200	input = this->state->selectTarget(*this, coord);	
201	}	
202	} return input;	
203	}	
204		
205		
206	void Player::setState(CharacterStateID newState) {	
207	if(this->state != nullptr ^	
208	this->state->getState() == CharacterStateID::Meditate ^	
209	newState != CharacterStateID::Meditate)	
210	this->animation = nullptr;	
211	if(this->state == nullptr v this->state->getState() != newState) {	
212	switch (newState) {	
213	case CharacterStateID::Still:	
214	this->state = std::shared_ptr<CharacterState>(new StillState());	
215	break;	
216	case CharacterStateID::Interact:	
217	this->state = std::shared_ptr<CharacterState>(new InteractState());	
218	break;	
219	case CharacterStateID::Move:	
220	this->state = std::shared_ptr<CharacterState>(new MoveState());	
221	break;	
222	case CharacterStateID::Meditate:	
223	this->state = std::shared_ptr<CharacterState>(new MeditateState());	
224	this->animation = std::shared_ptr<Animation>(new MeditateAnimation(this	
225	->manager, mixer.getEffect(MusicID::Meditation));	
226	break;	
227	case CharacterStateID::Attack:	
228	this->state = std::shared_ptr<CharacterState>(new AttackState());	
229	break;	
230	case CharacterStateID::Resurrect:	
231	this->state = std::shared_ptr<CharacterState>(new ResurrectState());	
232	break;	
233	}	
234	}	
235		
236	Point* Player::getCenter() {	
237	return ¢er;	
238	}	
239		
240	uint Player::getLevel() const {	
241	return this->playerInfo.getLevel();	
242	}	
243		
244	uint Player::getHealth() const {	
245	return this->playerInfo.getLife();	
246	}	
247		
248	uint Player::getMana() const {	
249	return this->playerInfo.getMana();	
250	}	
251		
252	uint Player::getGold() const {	
253	return this->playerInfo.getGoldAmount();	
254	}	
255		
256	uint Player::getMaxHealth() const {	
257	return this->playerInfo.getMaxLife();	
258	}	
259		
260	uint Player::getMaxMana() const {	
261	return this->playerInfo.getMaxMana();	

jul 21, 20 15:20	Player.cpp	Page 5/5
262	}	
263		
264	uint Player::getSafeGold() const {	
265	return this->playerInfo.getSafeGold();	
266	}	
267		
268	uint Player::getExp() const {	
269	return this->playerInfo.getExp();	
270	}	
271		
272	uint Player::getMaxExp() const {	
273	return this->playerInfo.getMaxExp();	
274	}	
275		
276	std::string Player::getInventory() const {	
277	return this->playerInfo.getInventory();	
278	}	
279		
280	CharacterStateID& Player::getState() const {	
281	return this->state->getState();	
282	}	
283		
284	PlayerInfo Player::getInfo() const {	
285	return this->playerInfo;	
286	}	
287		
288	Player::~~Player() = default;	

jul 21, 20 15:20	NPC.h	Page 1/1
1	<code>#ifndef NPC_H</code>	
2	<code>#define NPC_H</code>	
3		
4	<code>#include "Character.h"</code>	
5	<code>#include "Items/Item.h"</code>	
6	<code>#include "TextureManager.h"</code>	
7	<code>#include "MusicManager.h"</code>	
8	<code>#include "../common/Identificators.h"</code>	
9	<code>#include "../common/GameObjectInfo.h"</code>	
10	<code>#include "characterStates/CharacterState.h"</code>	
11	<code>#include <memory></code>	
12		
13	<code>class NPC : public Character {</code>	
14	<code>private:</code>	
15	<code> std::shared_ptr<CharacterState> state = nullptr;</code>	
16		
17	<code> bool aItem{false};</code>	
18	<code> std::shared_ptr<Item> item = nullptr;</code>	
19	<code> void setState(CharacterStateID newState);</code>	
20	<code> void setItem(ItemsInventoryID itemId);</code>	
21	<code> MusicID selectSound();</code>	
22		
23	<code>public:</code>	
24	<code> NPC(const TextureManager& manager, const GameObjectInfo& gameObjectInfo, con</code>	
25	<code>st MusicManager& mixer);</code>	
26	<code> virtual void render(Camera& camera);</code>	
27	<code> virtual void update(double dt);</code>	
28	<code> bool isItem() const;</code>	
29	<code> void updatePlayerInfo(const GameObjectInfo& info);</code>	
30	<code> CharacterStateID& getState() const;</code>	
31	<code> ~NPC();</code>	
32		
33	<code>};</code>	
34		
35	<code>#endif</code>	

jul 21, 20 15:20	NPC.cpp	Page 1/3
1	<code>#include "NPC.h"</code>	
2		
3	<code>#include <memory></code>	
4	<code>#include "characterStates/StillState.h"</code>	
5	<code>#include "characterStates/MoveState.h"</code>	
6	<code>#include "characterStates/MeditateState.h"</code>	
7	<code>#include "characterStates/InteractState.h"</code>	
8	<code>#include "characterStates/AttackState.h"</code>	
9	<code>#include "characterStates/ResurrectState.h"</code>	
10	<code>#include "../common/Random.h"</code>	
11	<code>#include "Items/Animation.h"</code>	
12	<code>#include "Items/MeditateAnimation.h"</code>	
13		
14	<code>NPC::NPC(const TextureManager& manager, const GameObjectInfo& gameObjectInfo,</code>	
15	<code>const MusicManager& mixer): Character(gameObjectInfo.getX(), gameObjectInfo.ge</code>	
16	<code>ty(),</code>	
17	<code>gameObjectInfo.getId(),manager,mixer){</code>	
18	<code> this->direction = gameObjectInfo.getDirection();</code>	
19	<code> setState(gameObjectInfo.getState());</code>	
20	<code> setFrameHead();</code>	
21	<code> setArmor(gameObjectInfo.getBodyID());</code>	
22	<code> setHead(gameObjectInfo.getHeadID());</code>	
23	<code> setHelmet(gameObjectInfo.getHelmetID());</code>	
24	<code> setShield(gameObjectInfo.getShieldID());</code>	
25	<code> setWeapon(gameObjectInfo.getWeaponID());</code>	
26	<code>}</code>	
27	<code>void NPC::render(Camera& camera) {</code>	
28	<code> if (!aItem) {</code>	
29	<code> int distance = camera.distanceFromTarget(this->getPosition());</code>	
30	<code> Character::render(camera);</code>	
31	<code> if(distance < 400){</code>	
32	<code> MusicID effectId = selectSound();</code>	
33	<code> if (Random::get(0,600) == 1 ^ effectId != MusicID::Nothing) {</code>	
34	<code> const Effect& effect = mixer.getEffect(effectId);</code>	
35	<code> if (distance > 255)</code>	
36	<code> distance = 255;</code>	
37	<code> effect.setDistance(distance);</code>	
38	<code> effect.playEffect(0,64);</code>	
39	<code> }</code>	
40	<code> }</code>	
41	<code> } else {</code>	
42	<code> if (this->item != nullptr)</code>	
43	<code> this->item->render(int(posX-camera.getCameraPosition().x), int(posY-camera</code>	
44	<code>.getCameraPosition().y));</code>	
45	<code> }</code>	
46		
47	<code>void NPC::update(double dt) {</code>	
48	<code> if(this->state != nullptr ^ this->state->getState() == CharacterStateID::Move) {</code>	
49	<code> if(this->body != nullptr)</code>	
50	<code> this->body->update(dt,direction);</code>	
51	<code> if (this->weapon != nullptr)</code>	
52	<code> this->weapon->update(dt,direction);</code>	
53	<code> if (this->shield != nullptr)</code>	
54	<code> this->shield->update(dt,direction);</code>	
55	<code> }</code>	
56	<code> if (this->animation != nullptr)</code>	
57	<code> this->animation->update();</code>	
58	<code> }</code>	
59		
60	<code>void NPC::updatePlayerInfo(const GameObjectInfo &info) {</code>	
61	<code> this->posX = info.getX();</code>	
62	<code> this->posY = info.getY();</code>	
63	<code> this->direction = info.getDirection();</code>	
64	<code> setState(info.getState());</code>	

jul 21, 20 15:20	NPC.cpp	Page 2/3
65	setArmor(info.getBodyID());	
66	setHead(info.getHeadID());	
67	setHelmet(info.getHelmetID());	
68	setShield(info.getShieldID());	
69	setWeapon(info.getWeaponID());	
70	setFrameHead();	
71	setAnimation(info.getAttackWeapon());	
72	if (info.isItem())	
73	setItem(info.getItemID());	
74	}	
75		
76	bool NPC::isItem() const {	
77	return this->aItem;	
78	}	
79		
80	void NPC::setItem(ItemsInventoryID itemId) {	
81	this->aItem = true;	
82	this->item = std::make_shared<Item>(manager.getTexture(itemId),32,32);	
83	}	
84		
85	MusicID NPC::selectSound() {	
86	MusicID effectId = MusicID::Nothing;	
87	if (body != nullptr) {	
88	switch (this->body->getId()) {	
89	case BodyID::Goblin:	
90	effectId = MusicID::Goblin;	
91	break;	
92	case BodyID::Skeleton:	
93	effectId = MusicID::Skeleton;	
94	break;	
95	case BodyID::Zombie:	
96	effectId = MusicID::Zombie;	
97	break;	
98	default:	
99	effectId = MusicID::Nothing;	
100	}	
101	}	
102	return effectId;	
103	}	
104		
105	void NPC::setState(CharacterStateID newState) {	
106	if(this->state != nullptr ^	
107	this->state->getState() == CharacterStateID::Meditate ^	
108	newState == CharacterStateID::Meditate)	
109	this->animation = nullptr;	
110	if(this->state == nullptr v this->state->getState() != newState) {	
111	switch (newState) {	
112	case CharacterStateID::Still:	
113	this->state = std::shared_ptr<CharacterState>(new StillState());	
114	break;	
115	case CharacterStateID::Interact:	
116	this->state = std::shared_ptr<CharacterState>(new InteractState());	
117	break;	
118	case CharacterStateID::Move:	
119	this->state = std::shared_ptr<CharacterState>(new MoveState());	
120	break;	
121	case CharacterStateID::Meditate:	
122	this->state = std::shared_ptr<CharacterState>(new MeditateState());	
123	this->animation = std::shared_ptr<Animation>(new MeditateAnimation(this	
124	->manager, mixer.getEffect(MusicID::Meditation));	
125	break;	
126	case CharacterStateID::Attack:	
127	this->state = std::shared_ptr<CharacterState>(new AttackState());	
128	break;	
129	case CharacterStateID::Resurrect:	
130	this->state = std::shared_ptr<CharacterState>(new ResurrectState());	

jul 21, 20 15:20	NPC.cpp	Page 3/3
130	break;	
131	}	
132	}	
133	}	
134		
135	CharacterStateID& NPC::getState() const {	
136	return this->state->getState();	
137	}	
138		
139	NPC::~NPC()= default;	

jul 21, 20 15:20	MusicManager.h	Page 1/1
	<pre> 1 #ifndef MUSICMANAGER_H 2 #define MUSICMANAGER_H 3 #include "Music.h" 4 #include "Effect.h" 5 #include "MusicID.h" 6 #include <unordered_map> 7 #include <string> 8 9 class MusicManager { 10 private: 11 std::unordered_map<MusicID, Music, std::hash<MusicID>> songs; 12 std::unordered_map<MusicID, Effect, std::hash<MusicID>> effects; 13 14 void createMusic(MusicID id, const std::string& path); 15 void createEffect(MusicID id, const std::string& path); 16 public: 17 MusicManager(); 18 19 void dropMusic(MusicID id); 20 void dropEffect(MusicID id); 21 22 const Music& getMusic(MusicID id) const; 23 24 const Effect& getEffect(MusicID id) const; 25 26 void loadSounds(); 27 28 ~MusicManager(); 29 30 MusicManager(const MusicManager&) = delete; 31 MusicManager &operator=(const MusicManager&) = delete; 32 }; 33 34 #endif </pre>	

jul 21, 20 15:20	MusicManager.cpp	Page 1/1
	<pre> 1 #include "MusicManager.h" 2 #include <utility> 3 #include "MusicID.h" 4 #include "../common/Identificators.h" 5 6 MusicManager::MusicManager() : songs(), effects() {} 7 8 9 void MusicManager::createMusic(MusicID id, const std::string& path) { 10 Music newSong(path); 11 this->songs.insert(std::make_pair(id, std::move(newSong))); 12 } 13 14 void MusicManager::createEffect(MusicID id, const std::string& path) { 15 Effect newEffect(path); 16 this->effects.insert(std::make_pair(id, std::move(newEffect))); 17 } 18 19 void MusicManager::dropMusic(MusicID id) { 20 this->songs.erase(id); 21 } 22 23 void MusicManager::dropEffect(MusicID id) { 24 this->effects.erase(id); 25 } 26 27 const Music& MusicManager::getMusic(MusicID id) const { 28 return this->songs.at(id); 29 } 30 31 const Effect& MusicManager::getEffect(MusicID id) const { 32 return this->effects.at(id); 33 } 34 35 void MusicManager::loadSounds() { 36 std::string path(ROOT_DIR); 37 createMusic(MusicID::BackGround, path + "/assets/sound/Musica Inicio.mp3"); 38 createEffect(MusicID::Eagle, path + "/assets/sound/Aguila.wav"); 39 createEffect(MusicID::Walk, path + "/assets/sound/Caminata.wav"); 40 createEffect(MusicID::Cure, path + "/assets/sound/Curar.wav"); 41 createEffect(MusicID::Sword, path + "/assets/sound/Espada.wav"); 42 createEffect(MusicID::Skeleton, path + "/assets/sound/Esqueleto.wav"); 43 createEffect(MusicID::Explosion, path + "/assets/sound/Explosion.wav"); 44 createEffect(MusicID::MagicArrow, path + "/assets/sound/Flecha Magica.wav"); 45 createEffect(MusicID::Arrow, path + "/assets/sound/Flecha.wav"); 46 createEffect(MusicID::Goblin, path + "/assets/sound/Goblin.wav"); 47 createEffect(MusicID::Ax, path + "/assets/sound/Hacha.wav"); 48 createEffect(MusicID::Hammer, path + "/assets/sound/Martillo.wav"); 49 createEffect(MusicID::Meditation, path + "/assets/sound/Meditar.wav"); 50 createEffect(MusicID::Misil, path + "/assets/sound/Misil.wav"); 51 createEffect(MusicID::Death, path + "/assets/sound/Muerte.wav"); 52 createEffect(MusicID::Wind, path + "/assets/sound/Viento.wav"); 53 createEffect(MusicID::Zombie, path + "/assets/sound/Zombie.wav"); 54 createEffect(MusicID::Wolf, path + "/assets/sound/Lobo.wav"); 55 createEffect(MusicID::Raven, path + "/assets/sound/Cuervo.wav"); 56 createEffect(MusicID::Heart, path + "/assets/sound/Latidos.wav"); 57 createEffect(MusicID::Merchant, path + "/assets/sound/Comerciante.wav"); 58 createEffect(MusicID::Banker, path + "/assets/sound/Banquero.wav"); 59 createEffect(MusicID::Priest, path + "/assets/sound/Sacerdote.wav"); 60 } 61 62 MusicManager::~MusicManager() {} </pre>	

jul 21, 20 15:20	MusicID.h	Page 1/1
1	#ifndef MUSICID_H	
2	#define MUSICID_H	
3		
4	enum class MusicID {	
5	Nothing,	
6	BackGround,	
7	Eagle,	
8	Wind,	
9	Wolf,	
10	Walk,	
11	Cure,	
12	Sword,	
13	Skeleton,	
14	Explosion,	
15	MagicArrow,	
16	Arrow,	
17	Goblin,	
18	Ax,	
19	Hammer,	
20	Meditation,	
21	Missil,	
22	Death,	
23	Zombie,	
24	Raven,	
25	Heart,	
26	Banker,	
27	Merchant,	
28	Priest,	
29	};	
30		
31	#endif	

jul 21, 20 15:20	Music.h	Page 1/1
1	#ifndef MUSIC_H	
2	#define MUSIC_H	
3	#include <string>	
4		
5	typedef struct _Mix_Music Mix_Music;	
6		
7	class Music {	
8	private:	
9	Mix_Music* music = nullptr;	
10	bool reproduce;	
11	Mix_Music* loadMusic(const std::string &fileName);	
12		
13	public:	
14	explicit Music(const std::string &fileName);	
15	Music(Music^ other);	
16		
17	int playMusic(int times) const ;	
18		
19	void pauseMusic() const ;	
20		
21	void setVolume(int volume) const ;	
22		
23	void stopMusic() const ;	
24		
25	~Music();	
26	};	
27		
28	#endif	

jul 21, 20 15:20	Music.cpp	Page 1/1
	<pre> 1 #include "Music.h" 2 #include <SDL2/SDL_mixer.h> 3 #include "../common/Exception.h" 4 5 6 Music::Music(const std::string& fileName) : reproduce(false) { 7 this->music = loadMusic(fileName); 8 } 9 10 int Music::playMusic(int times) const { 11 return Mix_PlayMusic(this->music, times); 12 } 13 14 Music::Music(Music ^other) { 15 std::swap(this->music, other.music); 16 } 17 18 19 Mix_Music* Music::loadMusic(const std::string &fileName) { 20 if (!this->music) { 21 this->music = Mix_LoadMUS(fileName.c_str()); 22 } 23 if (!this->music) { 24 throw Exception("Fail loading music: %s", Mix_GetError()); 25 } 26 return this->music; 27 } 28 29 void Music::setVolume(int volume) const { 30 Mix_VolumeMusic(volume); 31 } 32 33 void Music::pauseMusic() const { 34 if (Mix_PausedMusic() == 1) 35 Mix_ResumeMusic(); 36 else 37 Mix_PauseMusic(); 38 } 39 40 void Music::stopMusic() const { 41 Mix_HaltMusic(); 42 } 43 44 Music::~Music() { 45 if (this->music) { 46 Mix_FreeMusic(this->music); 47 this->music = nullptr; 48 } 49 } </pre>	

jul 21, 20 15:20	ZombieHead.h	Page 1/1
	<pre> 1 #ifndef ZOMBIEHEAD_H 2 #define ZOMBIEHEAD_H 3 4 #include "Head.h" 5 #include "../TextureManager.h" 6 7 class ZombieHead: public Head { 8 public: 9 ZombieHead(const TextureManager& manager); 10 11 virtual void render(int posX, int posY); 12 13 virtual void update(int dir); 14 15 ~ZombieHead(); 16 17 }; 18 19 #endif </pre>	

jul 21, 20 15:20	ZombieHead.cpp	Page 1/1
1	<code>#include "ZombieHead.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>ZombieHead::ZombieHead(const TextureManager& manager) :</code>	
8	<code>Head(manager.getTexture(TextureID::ZombieHead),WIDTH_HEAD,HEIGHT_HEAD, HeadID::</code>	
9	<code>Zombie) {}</code>	
10	<code>void ZombieHead::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*direction, this->height*0, this->width, this</code>	
12	<code>->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void ZombieHead::update(int dir){</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>ZombieHead::~ZombieHead(){}</code>	

jul 21, 20 15:20	ZombieBody.h	Page 1/1
1	<code>#ifndef ZOMBIEBODY_H</code>	
2	<code>#define ZOMBIEBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class ZombieBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>ZombieBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~ZombieBody();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	ZombieBody.cpp	Page 1/1
1	<code>#include "ZombieBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 25</code>	
4	<code>#define HEIGHT_BODY 45</code>	
5		
6	<code>ZombieBody::ZombieBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::ZombieBody),WIDTH_BODY,HEIGHT_BODY,BodyID::Z</code>	
8	<code>ombie) {}</code>	
9	<code>void ZombieBody::render(int posX, int posY) {</code>	
10	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
11	<code>th, this->height};</code>	
12	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
13	<code>this->texture.render(srcBody, dstBody);</code>	
14	<code>}</code>	
15	<code>void ZombieBody::update(double dt,Direction dir) {</code>	
16	<code>setDirection(int(dir));</code>	
17	<code>Body::update(dt,dir);</code>	
18	<code>}</code>	
19		
20	<code>void ZombieBody::setDirection(int direction) {</code>	
21	<code>if (direction == 0 v direction == 1)</code>	
22	<code>this->totalFrames = 6;</code>	
23	<code>if (direction == 2v direction == 3)</code>	
24	<code>this->totalFrames = 5;</code>	
25	<code>}</code>	
26		
27	<code>ZombieBody::~ZombieBody(){}</code>	

jul 21, 20 15:20	Weapon.h	Page 1/1
1	<code>#ifndef WEAPON_H</code>	
2	<code>#define WEAPON_H</code>	
3		
4	<code>#include "Item.h"</code>	
5	<code>#include "../Texture.h"</code>	
6	<code>#include "../common/Identificators.h"</code>	
7		
8	<code>class Weapon: public Item {</code>	
9	<code>protected:</code>	
10	<code>int frame{0};</code>	
11	<code>float elapsed{0.0};</code>	
12	<code>float animationSpeed{30.0f};</code>	
13	<code>int totalFrames{5};</code>	
14	<code>WeaponID id{WeaponID::Nothing};</code>	
15	<code>Direction direction{Direction::down};</code>	
16	<code>public:</code>	
17	<code>Weapon(const Texture& texture, const int width, const int height, WeaponID i</code>	
18	<code>d = WeaponID::Nothing);</code>	
19		
20	<code>virtual void render(int posX, int posY) = 0;</code>	
21	<code>void update(double dt, Direction dir);</code>	
22		
23	<code>void setAnimationSpeed(float speed);</code>	
24		
25	<code>int getHeight() const;</code>	
26		
27	<code>WeaponID getId() const;</code>	
28	<code>~Weapon();</code>	
29	<code>};</code>	
30		
31		
32	<code>#endif</code>	

jul 21, 20 15:20	Weapon.cpp	Page 1/1
1	<code>#include "Weapon.h"</code>	
2		
3	<code>Weapon::Weapon(const Texture& texture, const int width, const int height, Weapon</code>	
4	<code>ID id) : Item(texture, width, height), id(id) {}</code>	
5		
6	<code>void Weapon::update(double dt, Direction dir) {</code>	
7	<code> if (this->direction != dir) {</code>	
8	<code> this->direction = dir;</code>	
9	<code> this->elapsed = 0;</code>	
10	<code> }</code>	
11	<code> this->elapsed += dt;</code>	
12	<code> this->frame = int(this->elapsed/this->animationSpeed) % this->totalFrames</code>	
13	<code> ;</code>	
14	<code>}</code>	
15	<code>void Weapon::setAnimationSpeed(float speed) {</code>	
16	<code> this->animationSpeed = speed;</code>	
17	<code>}</code>	
18		
19	<code>int Weapon::getHeight() const {</code>	
20	<code> return this->height;</code>	
21	<code>}</code>	
22		
23	<code>WeaponID Weapon::getId() const {</code>	
24	<code> return this->id;</code>	
25	<code>}</code>	
26		
27	<code>Weapon::~Weapon() {}</code>	

jul 21, 20 15:20	TurtleShield.h	Page 1/1
1	<code>#ifndef TURTLESHIELD_H</code>	
2	<code>#define TURTLESHIELD_H</code>	
3		
4	<code>#include "Shield.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class TurtleShield: public Shield {</code>	
8	<code>private:</code>	
9	<code> void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code> TurtleShield(const TextureManager& manager);</code>	
12		
13	<code> virtual void render(int posX, int posY);</code>	
14		
15	<code> virtual void update(double dt, Direction dir);</code>	
16		
17	<code> ~TurtleShield();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	TurtleShield.cpp	Page 1/1
1	<code>#include "TurtleShield.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>TurtleShield::TurtleShield(const TextureManager& manager) :</code>	
8	<code>Shield(manager.getTexture(TextureID::TurtleShield),WIDTH_BODY,HEIGHT_BODY,ShieldID::TurtleShield) {}</code>	
9		
10	<code>void TurtleShield::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcBody, dstBody);</code>	
15	<code>}</code>	
16	<code>void TurtleShield::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Shield::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void TurtleShield::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28	<code>TurtleShield::~TurtleShield(){}</code>	

jul 21, 20 15:20	SpiderBody.h	Page 1/1
1	<code>#ifndef SPIDERBODY_H</code>	
2	<code>#define SPIDERBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class SpiderBody: public Body {</code>	
8	<code>public:</code>	
9	<code>SpiderBody(const TextureManager& manager);</code>	
10		
11	<code>virtual void render(int posX, int posY);</code>	
12		
13	<code>virtual void update(double dt,Direction dir);</code>	
14		
15	<code>~SpiderBody();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	SpiderBody.cpp	Page 1/1
1	<code>#include "SpiderBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 53</code>	
4	<code>#define HEIGHT_BODY 35</code>	
5		
6	<code>SpiderBody::SpiderBody(const TextureManager& manager):</code>	
7	<code>Body(manager.getTexture(TextureID::Spider),WIDTH_BODY,HEIGHT_BODY,BodyID::Sp</code>	
8	<code>ider) { this->totalFrames = 4;</code>	
9	<code>}</code>	
10		
11	<code>void SpiderBody::render(int posX, int posY) {</code>	
12	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
13	<code>th, this->height};</code>	
14	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
15	<code>this->texture.render(srcBody, dstBody);</code>	
16	<code>}</code>	
17	<code>void SpiderBody::update(double dt,Direction dir) {</code>	
18	<code>Body::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>SpiderBody::~SpiderBody(){}</code>	

jul 21, 20 15:20	SkeletonBody.h	Page 1/1
1	<code>#ifndef SKELETONBODY_H</code>	
2	<code>#define SKELETONBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class SkeletonBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>SkeletonBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~SkeletonBody();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	SkeletonBody.cpp	Page 1/1
1	<code>#include "SkeletonBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 25</code>	
4	<code>#define HEIGHT_BODY 52</code>	
5		
6	<code>SkeletonBody::SkeletonBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::Skeleton),WIDTH_BODY,HEIGHT_BODY,BodyID::Ske</code>	
8	<code>leton) {}</code>	
9	<code>void SkeletonBody::render(int posX, int posY) {</code>	
10	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
11	<code>th, this->height};</code>	
12	<code>SDL_Rect dstBody = {posX, posY, int(this->width*1.2), int(this->height*1.2)};</code>	
13	<code>this->texture.render(srcBody, dstBody);</code>	
14	<code>}</code>	
15	<code>void SkeletonBody::update(double dt,Direction dir) {</code>	
16	<code>setDirection(int(dir));</code>	
17	<code>Body::update(dt,dir);</code>	
18	<code>}</code>	
19		
20	<code>void SkeletonBody::setDirection(int direction) {</code>	
21	<code>if (direction == 0 v direction == 1)</code>	
22	<code>this->totalFrames = 6;</code>	
23	<code>if (direction == 2v direction == 3)</code>	
24	<code>this->totalFrames = 5;</code>	
25	<code>}</code>	
26		
27	<code>SkeletonBody::~SkeletonBody(){}</code>	

jul 21, 20 15:20	SimpleArc.h	Page 1/1
1	<code>#ifndef SIMPLEARC_H</code>	
2	<code>#define SIMPLEARC_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class SimpleArc: public Weapon {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>SimpleArc(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~SimpleArc();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	SimpleArc.cpp	Page 1/1
1	<code>#include "SimpleArc.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>SimpleArc::SimpleArc(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::SimpleArc),WIDTH_BODY,HEIGHT_BODY, Weap</code>	
9	<code>onID::SimpleArc) {}</code>	
10	<code>void SimpleArc::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void SimpleArc::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void SimpleArc::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2 v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>SimpleArc::~SimpleArc(){}</code>	

jul 21, 20 15:20	Shield.h	Page 1/1
1	<code>#ifndef SHIELD_H</code>	
2	<code>#define SHIELD_H</code>	
3		
4	<code>#include "Item.h"</code>	
5	<code>#include "../Texture.h"</code>	
6	<code>#include "../common/Identificators.h"</code>	
7		
8	<code>class Shield: public Item {</code>	
9	<code>protected:</code>	
10	<code>int frame{0};</code>	
11	<code>float elapsed{0.0};</code>	
12	<code>float animationSpeed{30.0f};</code>	
13	<code>int totalFrames{5};</code>	
14	<code>ShieldID id{ShieldID::Nothing};</code>	
15	<code>Direction direction{Direction::down};</code>	
16	<code>public:</code>	
17	<code>Shield(const Texture& texture, const int width, const int height, ShieldID i</code>	
18	<code>d = ShieldID::Nothing);</code>	
19		
20	<code>virtual void render(int posX, int posY) = 0;</code>	
21	<code>void update(double dt, Direction dir);</code>	
22		
23	<code>void setAnimationSpeed(float speed);</code>	
24		
25	<code>ShieldID getId() const;</code>	
26		
27	<code>~Shield();</code>	
28	<code>};</code>	
29		
30	<code>#endif</code>	

jul 21, 20 15:20	Shield.cpp	Page 1/1
1	#include "Shield.h"	
2		
3	Shield::Shield(const Texture& texture, const int width, const int height, Shield ID id) : Item(texture, width, height), id(id){}	
4		
5	void Shield::update(double dt, Direction dir) {	
6	if (this ->direction != dir) {	
7	this ->direction = dir;	
8	this ->elapsed = 0;	
9	}	
10	this ->elapsed += dt;	
11	this ->frame = int(this ->elapsed/ this ->animationSpeed) % this ->totalFrames	
12	};	
13		
14	void Shield::setAnimationSpeed(float speed) {	
15	this ->animationSpeed = speed;	
16	}	
17		
18	ShieldID Shield::getId() const{	
19	return this ->id;	
20	}	
21		
22	Shield::~Shield() {}	
23		

jul 21, 20 15:20	RedCommonBody.h	Page 1/1
1	#ifndef REDCOMMONBODY_H	
2	#define REDCOMMONBODY_H	
3		
4	#include "Body.h"	
5	#include "../TextureManager.h"	
6		
7	class RedCommonBody: public Body {	
8	private:	
9	void setDirection(int direction);	
10	public:	
11	RedCommonBody(const TextureManager& manager);	
12		
13	virtual void render(int posX, int posY);	
14		
15	virtual void update(double dt, Direction dir);	
16		
17	~RedCommonBody();	
18		
19	};	
20		
21	#endif	

jul 21, 20 15:20	RedCommonBody.cpp	Page 1/1
1	#include "RedCommonBody.h"	
2	#include "../TextureID.h"	
3		
4	#define WIDTH_BODY 25	
5	#define HEIGHT_BODY 45	
6		
7	RedCommonBody::RedCommonBody(const TextureManager& manager):	
8	Body(manager.getTexture(TextureID::RedCommonBody),WIDTH_BODY,HEIGHT_BODY,Bod	
9	yID::RedCommon) {}	
10	void RedCommonBody::render(int posX, int posY) {	
11	SDL_Rect srcHead = { this ->width*frame, this ->height*int(direction), this -wid	
12	th, this ->height};	
13	SDL_Rect dstHead = {posX, posY, this ->width, this ->height};	
14	this ->texture.render(srcHead, dstHead);	
15	}	
16	void RedCommonBody::update(double dt,Direction dir) {	
17	setDirection(int(dir));	
18	Body::update(dt,dir);	
19	}	
20		
21	void RedCommonBody::setDirection(int direction) {	
22	if (direction == 0 v direction == 1)	
23	this ->totalFrames = 6;	
24	if (direction == 2v direction == 3)	
25	this ->totalFrames = 5;	
26	}	
27		
28		
29	RedCommonBody::~RedCommonBody(){} 	

jul 21, 20 15:20	PriestHead.h	Page 1/1
1	#ifndef PRIESTHEAD_H	
2	#define PRIESTHEAD_H	
3		
4	#include "Head.h"	
5	#include "../TextureManager.h"	
6		
7	class PriestHead: public Head {	
8	public:	
9	PriestHead(const TextureManager& manager);	
10		
11	virtual void render(int posX, int posY);	
12		
13	virtual void update(int dir);	
14		
15	~PriestHead();	
16		
17	};	
18		
19	#endif	

jul 21, 20 15:20	PriestHead.cpp	Page 1/1
1	<code>#include "PriestHead.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 17</code>	
6		
7	<code>PriestHead::PriestHead(const TextureManager& manager) :</code>	
8	<code>Head(manager.getTexture(TextureID::PriestHead),WIDTH_HEAD,HEIGHT_HEAD,HeadID::P</code>	
9	<code>riest) {}</code>	
10	<code>void PriestHead::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width * direction, this->height * 0, this->width, t</code>	
12	<code>his->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void PriestHead::update(int dir){</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>PriestHead::~PriestHead(){}</code>	

jul 21, 20 15:20	PriestBody.h	Page 1/1
1	<code>#ifndef PRIESTBODY_H</code>	
2	<code>#define PRIESTBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class PriestBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>PriestBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~PriestBody();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	PriestBody.cpp	Page 1/1
1	<code>#include "PriestBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 25</code>	
4	<code>#define HEIGHT_BODY 45</code>	
5		
6	<code>PriestBody::PriestBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::PriestBody),WIDTH_BODY,HEIGHT_BODY,BodyID::P</code>	
8	<code>riest) {}</code>	
9	<code>void PriestBody::render(int posX, int posY) {</code>	
10	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
11	<code>th, this->height};</code>	
12	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
13	<code>this->texture.render(srcBody, dstBody);</code>	
14	<code>}</code>	
15	<code>void PriestBody::update(double dt,Direction dir) {</code>	
16	<code>setDirection(int(dir));</code>	
17	<code>Body::update(dt,dir);</code>	
18	<code>}</code>	
19		
20	<code>void PriestBody::setDirection(int direction) {</code>	
21	<code>if (direction == 0 v direction == 1)</code>	
22	<code>this->totalFrames = 6;</code>	
23	<code>if (direction == 2v direction == 3)</code>	
24	<code>this->totalFrames = 5;</code>	
25	<code>}</code>	
26		
27	<code>PriestBody::~PriestBody() {}</code>	

jul 21, 20 15:20	PlateArmor.h	Page 1/1
1	<code>#ifndef PLATEARMOR_H</code>	
2	<code>#define PLATEARMOR_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class PlateArmor: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>PlateArmor(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~PlateArmor();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	PlateArmor.cpp	Page 1/1
1	<code>#include "PlateArmor.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>PlateArmor::PlateArmor(const TextureManager& manager) :</code>	
8	<code>Body(manager.getTexture(TextureID::PlateArmor),WIDTH_BODY,HEIGHT_BODY,BodyID::P</code>	
9	<code>lateArmor) {}</code>	
10	<code>void PlateArmor::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void PlateArmor::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Body::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void PlateArmor::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>PlateArmor::~PlateArmor(){} </code>	

jul 21, 20 15:20	MissileAnimation.h	Page 1/1
1	<code>#ifndef MISSILEANIMATION_H</code>	
2	<code>#define MISSILEANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class MissileAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code>MissileAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code>virtual void render(int x, int y);</code>	
13		
14	<code>~MissileAnimation();</code>	
15		
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	MissileAnimation.cpp	Page 1/1
1	<code>#include "MissileAnimation.h"</code>	
2		
3	<code>#define WIDTHFRAME 85</code>	
4	<code>#define HEIGHTFRAME 70</code>	
5	<code>#define FRAMES 7</code>	
6		
7	<code>MissileAnimation::MissileAnimation(const TextureManager& manager, const Effect& effect) :</code>	
8	<code>Animation(manager.getTexture(TextureID::MissileEffect), WIDTHFRAME, HEIGHTFRAME, effect, FRAMES) {}</code>	
9		
10	<code>void MissileAnimation::render(int x, int y) {</code>	
11	<code>if (this->finish)</code>	
12	<code>return;</code>	
13	<code>if (!this->effectPlayed) {</code>	
14	<code> this->effect.playEffect(0, 64);</code>	
15	<code> this->effectPlayed = true;</code>	
16	<code>}</code>	
17	<code>SDL_Rect src = {this->width*this->frame, this->height*0, this->width, this->height};</code>	
18	<code>SDL_Rect dst = {x-int(this->width/3), y-int(this->height/3), this->width, this->height};</code>	
19	<code> this->texture.render(src, dst);</code>	
20	<code>}</code>	
21		
22	<code>MissileAnimation::~MissileAnimation() {}</code>	

jul 21, 20 15:20	MerchantBody.h	Page 1/1
1	<code>#ifndef MERCHANTBODY_H</code>	
2	<code>#define MERCHANTBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class MerchantBody: public Body {</code>	
8	<code>private:</code>	
9	<code> void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code> MerchantBody(const TextureManager& manager);</code>	
12		
13	<code> virtual void render(int posX, int posY);</code>	
14		
15	<code> virtual void update(double dt, Direction dir);</code>	
16		
17	<code> ~MerchantBody();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	MerchantBody.cpp	Page 1/1
1	<code>#include "MerchantBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 25</code>	
4	<code>#define HEIGHT_BODY 45</code>	
5		
6	<code>MerchantBody::MerchantBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::Merchant),WIDTH_BODY,HEIGHT_BODY, BodyID::Me</code>	
8	<code>rchant) {}</code>	
9	<code>void MerchantBody::render(int posX, int posY) {</code>	
10	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
11	<code>th, this->height};</code>	
12	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
13	<code>this->texture.render(srcBody, dstBody);</code>	
14	<code>}</code>	
15	<code>void MerchantBody::update(double dt,Direction dir) {</code>	
16	<code>setDirection(int(dir));</code>	
17	<code>Body::update(dt,dir);</code>	
18	<code>}</code>	
19	<code>void MerchantBody::setDirection(int direction) {</code>	
20	<code>if (direction == 0 v direction == 1)</code>	
21	<code>this->totalFrames = 6;</code>	
22	<code>if (direction == 2v direction == 3)</code>	
23	<code>this->totalFrames = 5;</code>	
24	<code>}</code>	
25		
26	<code>MerchantBody::~MerchantBody(){}</code>	

jul 21, 20 15:20	MeditateAnimation.h	Page 1/1
1	<code>#ifndef MEDITATEANIMATION_H</code>	
2	<code>#define MEDITATEANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class MeditateAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code>MeditateAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code>virtual void render(int x, int y);</code>	
13		
14	<code>~MeditateAnimation();</code>	
15		
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	MeditateAnimation.cpp	Page 1/1
	<pre> 1 #include "MeditateAnimation.h" 2 3 #define WIDTHFRAME 80 4 #define HEIGHTFRAME 170 5 #define FRAMES 10 6 7 MeditateAnimation::MeditateAnimation(const TextureManager& manager, const Effect& effect) : 8 Animation(manager.getTexture(TextureID::MeditateEffect), WIDTHFRAME, HEIGHTFRAME , effect, FRAMES) {} 9 10 void MeditateAnimation::render(int x, int y) { 11 if(!this->effectPlayed) { 12 this->effect.playEffect(0, 64); 13 this->effectPlayed = true; 14 } 15 int row = this->totalFrames/2; 16 SDL_Rect src = {this->width*(this->frame % row), 17 this->height*int(floor(this->frame/row)), this->width, this ->height}; 18 SDL_Rect dst = {x-int(this->width/3), y-int(this->height/1.5), this->width, thi s->height}; 19 this->texture.render(src, dst); 20 this->finish=false; 21 } 22 23 MeditateAnimation::~MeditateAnimation() {} </pre>	

jul 21, 20 15:20	MagicHat.h	Page 1/1
	<pre> 1 #ifndef MAGICCHAT_H 2 #define MAGICCHAT_H 3 4 #include "Helmet.h" 5 #include "../TextureManager.h" 6 7 class MagicHat: public Helmet { 8 public: 9 MagicHat(const TextureManager& manager); 10 11 virtual void render(int posX, int posY); 12 13 virtual void update(int dir); 14 15 ~MagicHat(); 16 17 }; 18 19 #endif </pre>	

jul 21, 20 15:20	MagicHat.cpp	Page 1/1
1	<code>#include "MagicHat.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>MagicHat::MagicHat(const TextureManager& manager) :</code>	
8	<code>Helmet(manager.getTexture(TextureID::MagicHat),WIDTH_HEAD,HEIGHT_HEAD,HelmetID:</code>	
9	<code>:MagicHat) {}</code>	
10	<code>void MagicHat::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY-this->height/4+6, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void MagicHat::update(int dir) {</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>MagicHat::~MagicHat(){}</code>	

jul 21, 20 15:20	MagicArrowAnimation.h	Page 1/1
1	<code>#ifndef MAGICARROWANIMATION_H</code>	
2	<code>#define MAGICARROWANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class MagicArrowAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code>MagicArrowAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code>virtual void render(int x, int y);</code>	
13		
14	<code>~MagicArrowAnimation();</code>	
15		
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	MagicArrowAnimation.cpp	Page 1/1
1	<code>#include "MagicArrowAnimation.h"</code>	
2		
3	<code>#define WIDTHFRAME 62</code>	
4	<code>#define HEIGHTFRAME 60</code>	
5	<code>#define FRAMES 5</code>	
6		
7	<code>MagicArrowAnimation::MagicArrowAnimation(const TextureManager& manager, const Effect& effect) :</code>	
8	<code>Animation(manager, getTexture(TextureID::MagicArrowEffect), WIDTHFRAME, HEIGHTFRAME, effect, FRAMES) {}</code>	
9		
10	<code>void MagicArrowAnimation::render(int x, int y) {</code>	
11	<code>if (this->finish)</code>	
12	<code>return;</code>	
13	<code>if (!this->effectPlayed) {</code>	
14	<code> this->effect.playEffect(0, 64);</code>	
15	<code> this->effectPlayed = true;</code>	
16	<code>}</code>	
17	<code>SDL_Rect src = {this->width*this->frame, this->height*0, this->width, this->height};</code>	
18	<code>SDL_Rect dst = {x-int(this->width/3), y-int(this->height/3), this->width, this->height};</code>	
19	<code> this->texture.render(src, dst);</code>	
20	<code>}</code>	
21		
22	<code>MagicArrowAnimation::~MagicArrowAnimation() {}</code>	

jul 21, 20 15:20	LongSword.h	Page 1/1
1	<code>#ifndef LONGSWORD_H</code>	
2	<code>#define LONGSWORD_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class LongSword: public Weapon {</code>	
8	<code>private:</code>	
9	<code> void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code> LongSword(const TextureManager& manager);</code>	
12		
13	<code> virtual void render(int posX, int posY);</code>	
14		
15	<code> virtual void update(double dt, Direction dir);</code>	
16		
17	<code> ~LongSword();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	LongSword.cpp	Page 1/1
1	<code>#include "LongSword.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>LongSword::LongSword(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::LongSword),WIDTH_BODY,HEIGHT_BODY, Weap</code>	
9	<code>onID::LongSword) {}</code>	
10	<code>void LongSword::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void LongSword::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void LongSword::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>LongSword::~LongSword(){}</code>	

jul 21, 20 15:20	LeatherArmor.h	Page 1/1
1	<code>#ifndef LEATHERARMOR_H</code>	
2	<code>#define LEATHERARMOR_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class LeatherArmor: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>LeatherArmor(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~LeatherArmor();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	LeatherArmor.cpp	Page 1/1
1	<code>#include "LeatherArmor.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>LeatherArmor::LeatherArmor(const TextureManager& manager) :</code>	
8	<code>Body(manager.getTexture(TextureID::LeatherArmor),WIDTH_BODY,HEIGHT_BODY,BodyID:</code>	
9	<code>:LeatherArmor) {}</code>	
10	<code>void LeatherArmor::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void LeatherArmor::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Body::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void LeatherArmor::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28	<code>LeatherArmor::~LeatherArmor(){}</code>	

jul 21, 20 15:20	Item.h	Page 1/1
1	<code>#ifndef ITEM_H</code>	
2	<code>#define ITEM_H</code>	
3		
4	<code>#include "../Texture.h"</code>	
5		
6	<code>class Item {</code>	
7	<code>protected:</code>	
8	<code>const Texture& texture;</code>	
9	<code>const int width; //Ancho del Frame</code>	
10	<code>const int height; //Alto del Frame</code>	
11	<code>public:</code>	
12	<code>Item(const Texture& texture, const int width, const int height);</code>	
13		
14	<code>virtual void render(int posX, int posY);</code>	
15		
16	<code>~Item();</code>	
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	Item.cpp	Page 1/1
1	#include "Item.h"	
2		
3	Item::Item(const Texture& texture, const int width, const int height) :	
4	texture(texture), width(width), height(height) {}	
5		
6	void Item::render(int posX, int posY) {	
7	SDL_Rect srcItem = {0,0, this->width, this->height};	
8	SDL_Rect dstItem = {posX, posY, this->width, this->height};	
9	this->texture.render(srcItem, dstItem);	
10	}	
11		
12	Item::~Item() {}	

jul 21, 20 15:20	IronShield.h	Page 1/1
1	#ifndef IRONSHIELD_H	
2	#define IRONSHIELD_H	
3		
4	#include "Shield.h"	
5	#include "../TextureManager.h"	
6		
7	class IronShield: public Shield {	
8	private:	
9	void setDirection(int direction);	
10	public:	
11	IronShield(const TextureManager& manager);	
12		
13	virtual void render(int posX, int posY);	
14		
15	virtual void update(double dt, Direction dir);	
16		
17	~IronShield();	
18		
19	};	
20		
21	#endif	

jul 21, 20 15:20	IronShield.cpp	Page 1/1
1	<code>#include "IronShield.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>IronShield::IronShield(const TextureManager& manager) :</code>	
8	<code>Shield(manager.getTexture(TextureID::IronShield),WIDTH_BODY,HEIGHT_BODY, Shield</code>	
9	<code>ID::IronShield) {}</code>	
10	<code>void IronShield::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcBody, dstBody);</code>	
15	<code>}</code>	
16	<code>void IronShield::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Shield::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void IronShield::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28	<code>IronShield::~IronShield(){}</code>	

jul 21, 20 15:20	IronHelmet.h	Page 1/1
1	<code>#ifndef IRONHELMET_H</code>	
2	<code>#define IRONHELMET_H</code>	
3		
4	<code>#include "Helmet.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class IronHelmet: public Helmet {</code>	
8	<code>public:</code>	
9	<code>IronHelmet(const TextureManager& manager);</code>	
10		
11	<code>virtual void render(int posX, int posY);</code>	
12		
13	<code>virtual void update(int dir);</code>	
14		
15	<code>~IronHelmet();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	IronHelmet.cpp	Page 1/1
1	<code>#include "IronHelmet.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>IronHelmet::IronHelmet(const TextureManager& manager) :</code>	
8	<code>Helmet(manager.getTexture(TextureID::IronHelmet),WIDTH_HEAD,HEIGHT_HEAD, Helmet</code>	
9	<code>ID::IronHelmet) {}</code>	
10	<code>void IronHelmet::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void IronHelmet::update(int dir) {</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>IronHelmet::~IronHelmet(){}</code>	

jul 21, 20 15:20	HumanHead.h	Page 1/1
1	<code>#ifndef HUMANHEAD_H</code>	
2	<code>#define HUMANHEAD_H</code>	
3		
4	<code>#include "Head.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class HumanHead: public Head {</code>	
8	<code>public:</code>	
9	<code>HumanHead(const TextureManager& manager);</code>	
10		
11	<code>virtual void render(int posX, int posY);</code>	
12		
13	<code>virtual void update(int dir);</code>	
14		
15	<code>~HumanHead();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	HumanHead.cpp	Page 1/1
1	#include "HumanHead.h"	
2	#include "../TextureID.h"	
3		
4	#define WIDTH_HEAD 17	
5	#define HEIGHT_HEAD 16	
6		
7	HumanHead::HumanHead(const TextureManager& manager) :	
8	Head(manager.getTexture(TextureID::HumanHead),WIDTH_HEAD,HEIGHT_HEAD,HeadID::Hu	
9	man) {}	
10	void HumanHead::render(int posX, int posY) {	
11	SDL_Rect srcHead = { this ->width* this ->direction, this ->height*0, this ->width,	
12	this ->height};	
13	SDL_Rect dstHead = {posX, posY+ this ->height-2, this ->width, this ->height};	
14	this ->texture.render(srcHead, dstHead);	
15	}	
16	void HumanHead::update(int dir){	
17	this ->direction = dir;	
18	}	
19		
20		
21	HumanHead::~HumanHead() {}	

jul 21, 20 15:20	Hood.h	Page 1/1
1	#ifndef HOOD_H	
2	#define HOOD_H	
3		
4	#include "Helmet.h"	
5	#include "../TextureManager.h"	
6		
7	class Hood: public Helmet {	
8	public:	
9	Hood(const TextureManager& manager);	
10		
11	virtual void render(int posX, int posY);	
12		
13	virtual void update(int dir);	
14		
15	~Hood();	
16		
17	};	
18		
19	#endif	

jul 21, 20 15:20	Hood.cpp	Page 1/1
1	<code>#include "Hood.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>Hood::Hood(const TextureManager& manager) :</code>	
8	<code>Helmet(manager.getTexture(TextureID::Hood),WIDTH_HEAD,HEIGHT_HEAD, HelmetID::Ho</code>	
9	<code>od) {}</code>	
10	<code>void Hood::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void Hood::update(int dir) {</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>Hood::~Hood() {}</code>	

jul 21, 20 15:20	HitAnimation.h	Page 1/1
1	<code>#ifndef HITANIMATION_H</code>	
2	<code>#define HITANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class HitAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code>HitAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code>virtual void render(int x, int y);</code>	
13		
14	<code>~HitAnimation();</code>	
15		
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	HitAnimation.cpp	Page 1/1
	<pre> 1 #include "HitAnimation.h" 2 #include "../common/Random.h" 3 #define WIDTHFRAME 15 4 #define HEIGHTFRAME 15 5 #define FRAMES 10 6 7 HitAnimation::HitAnimation(const TextureManager& manager, const Effect& effect) : 8 Animation(manager.getTexture(TextureID::HitEffect), WIDTHFRAME, HEIGHTFRAME, effect, FRAMES) {} 9 10 void HitAnimation::render(int x, int y) { 11 if (this->finish) 12 return; 13 if (!this->effectPlayed) { 14 this->effect.playEffect(); 15 this->effectPlayed = true; 16 } 17 int randX = Random::get(0, this->width); 18 int randY = Random::get(0, this->height*2); 19 SDL_Rect src = {this->width*this->frame, this->height*0, this->width, this->height}; 20 SDL_Rect dst = {x+randX, y+randY, this->width, this->height}; 21 this->texture.render(src, dst); 22 } 23 24 HitAnimation::~HitAnimation() {} </pre>	

jul 21, 20 15:20	Helmet.h	Page 1/1
	<pre> 1 #ifndef HELMET_H 2 #define HELMET_H 3 4 #include "Item.h" 5 #include "../Texture.h" 6 #include "../common/Identificators.h" 7 8 class Helmet: public Item { 9 protected: 10 HelmetID id{HelmetID::Nothing}; 11 int direction{0}; 12 public: 13 Helmet(const Texture& texture, const int width, const int height, HelmetID id) : 14 =HelmetID::Nothing : 15 Item(texture, width, height), id(id) {} 16 17 virtual void render(int posX, int posY) = 0; 18 19 virtual void update(int dir) = 0; 20 21 HelmetID getId() { 22 return this->id; 23 } 24 25 ~Helmet() {} 26 }; 27 #endif </pre>	

jul 21, 20 15:20	Head.h	Page 1/1
1	#ifndef HEAD_H	
2	#define HEAD_H	
3		
4	#include "Item.h"	
5	#include "../Texture.h"	
6	#include "../common/Identificators.h"	
7		
8	class Head: public Item {	
9	protected:	
10	HeadID id{HeadID::Nothing};	
11	int direction{0};	
12	public:	
13	Head(const Texture& texture, const int width, const int height, HeadID id = H	
14	eadID::Nothing) :	
15	Item(texture, width, height), id(id){}	
16	virtual void render(int posX, int posY) = 0;	
17	virtual void update(int dir) = 0;	
18	HeadID getId(){	
19	return this->id;	
20	}	
21	~Head() {};	
22	};	
23		
24	#endif	

jul 21, 20 15:20	Hammer.h	Page 1/1
1	#ifndef HAMMER_H	
2	#define HAMMER_H	
3		
4	#include "Weapon.h"	
5	#include "../TextureManager.h"	
6		
7	class Hammer: public Weapon {	
8	private:	
9	void setDirection(int direction);	
10	public:	
11	Hammer(const TextureManager& manager);	
12		
13	virtual void render(int posX, int posY);	
14	virtual void update(double dt, Direction dir);	
15	~Hammer();	
16		
17	};	
18		
19	#endif	
20		
21		

jul 21, 20 15:20	Hammer.cpp	Page 1/1
1	<code>#include "Hammer.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>Hammer::Hammer(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::Hammer),WIDTH_BODY,HEIGHT_BODY, WeaponI</code>	
9	<code>D::Hammer) {}</code>	
10	<code>void Hammer::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void Hammer::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void Hammer::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>Hammer::~Hammer(){}</code>	

jul 21, 20 15:20	GreenCommonBody.h	Page 1/1
1	<code>#ifndef GREENCOMMONBODY_H</code>	
2	<code>#define GREENCOMMONBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class GreenCommonBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>GreenCommonBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~GreenCommonBody();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	GreenCommonBody.cpp	Page 1/1
1	<code>#include "GreenCommonBody.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>GreenCommonBody::GreenCommonBody(const TextureManager& manager):</code>	
8	<code>Body(manager.getTexture(TextureID::GreenCommonBody),WIDTH_BODY,HEIGHT_BODY,</code>	
9	<code>BodyID::GreenCommon) {}</code>	
10	<code>void GreenCommonBody::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*frame, this->height*int(direction), this->wid</code>	
12	<code>th, this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void GreenCommonBody::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Body::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void GreenCommonBody::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>GreenCommonBody::~GreenCommonBody(){}</code>	

jul 21, 20 15:20	GoblinBody.h	Page 1/1
1	<code>#ifndef GOBLINBODY_h</code>	
2	<code>#define GOBLINBODY_h</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class GoblinBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>GoblinBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~GoblinBody();</code>	
18	<code>};</code>	
19		
20		
21	<code>#endif</code>	

jul 21, 20 15:20	GoblinBody.cpp	Page 1/1
1	<code>#include "GoblinBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 24</code>	
4	<code>#define HEIGHT_BODY 31</code>	
5		
6	<code>GoblinBody::GoblinBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::Goblin),WIDTH_BODY,HEIGHT_BODY, BodyID::Gobl</code>	
8	<code>in) {</code>	
9	<code> this->totalFrames = 8;</code>	
10	<code>}</code>	
11	<code>void GoblinBody::render(int posX, int posY) {</code>	
12	<code> SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
13	<code>th, this->height};</code>	
14	<code> SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
15	<code> this->texture.render(srcBody, dstBody);</code>	
16	<code>}</code>	
17	<code>void GoblinBody::update(double dt,Direction dir) {</code>	
18	<code> setDirection(int(dir));</code>	
19	<code> Body::update(dt,dir);</code>	
20	<code>}</code>	
21		
22	<code>void GoblinBody::setDirection(int direction) {}</code>	
23		
24	<code>GoblinBody::~GoblinBody(){}</code>	

jul 21, 20 15:20	GnomeHead.h	Page 1/1
1	<code>#ifndef GNOMEHEAD_H</code>	
2	<code>#define GNOMEHEAD_H</code>	
3		
4	<code>#include "Head.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class GnomeHead: public Head {</code>	
8	<code>public:</code>	
9	<code> GnomeHead(const TextureManager& manager);</code>	
10		
11	<code> virtual void render(int posX, int posY);</code>	
12		
13	<code> virtual void update(int dir);</code>	
14		
15	<code> ~GnomeHead();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	GnomeHead.cpp	Page 1/1
1	<code>#include "GnomeHead.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>GnomeHead::GnomeHead(const TextureManager& manager) :</code>	
8	<code>Head(manager.getTexture(TextureID::GnomeHead),WIDTH_HEAD,HEIGHT_HEAD,HeadID::Gn</code>	
9	<code>ome) {}</code>	
10	<code>void GnomeHead::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void GnomeHead::update(int dir){</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>GnomeHead::~GnomeHead() {}</code>	

jul 21, 20 15:20	GnarledStick.h	Page 1/1
1	<code>#ifndef GNARLEDSTICK_H</code>	
2	<code>#define GNARLEDSTICK_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class GnarledStick: public Weapon {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>GnarledStick(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~GnarledStick();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	GnarledStick.cpp	Page 1/1
1	<code>#include "GnarledStick.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>GnarledStick::GnarledStick(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::GnarledStick),WIDTH_BODY,HEIGHT_BODY, W</code>	
9	<code>eaonID::GnarledStick) {}</code>	
10	<code>void GnarledStick::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void GnarledStick::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void GnarledStick::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>GnarledStick::~GnarledStick(){}</code>	

jul 21, 20 15:20	GhostBody.h	Page 1/1
1	<code>#ifndef GHOSTBODY_H</code>	
2	<code>#define GHOSTBODY_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class GhostBody: public Body {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>GhostBody(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~GhostBody();</code>	
18	<code>};</code>	
19		
20		
21	<code>#endif</code>	

jul 21, 20 15:20	GhostBody.cpp	Page 1/1
1	<code>#include "GhostBody.h"</code>	
2		
3	<code>#define WIDTH_BODY 29</code>	
4	<code>#define HEIGHT_BODY 32</code>	
5		
6	<code>GhostBody::GhostBody(const TextureManager& manager) :</code>	
7	<code>Body(manager.getTexture(TextureID::Ghost),WIDTH_BODY,HEIGHT_BODY,BodyID::Ghost)</code>	
8	<code>{</code>	
9	<code> this->totalFrames = 3;</code>	
10	<code>}</code>	
11	<code>void GhostBody::render(int posX, int posY) {</code>	
12	<code> SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
13	<code>th, this->height};</code>	
14	<code> SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
15	<code> this->texture.render(srcBody, dstBody);</code>	
16	<code>}</code>	
17	<code>void GhostBody::update(double dt,Direction dir) {</code>	
18	<code> setDirection(int(dir));</code>	
19	<code> Body::update(dt,dir);</code>	
20	<code>}</code>	
21		
22	<code>void GhostBody::setDirection(int direction) {}</code>	
23		
24	<code>GhostBody::~GhostBody(){}</code>	

jul 21, 20 15:20	ExplosionAnimation.h	Page 1/1
1	<code>#ifndef EXPLOSIONANIMATION_H</code>	
2	<code>#define EXPLOSIONANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class ExplosionAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code> ExplosionAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code> virtual void render(int x, int y);</code>	
13		
14	<code> ~ExplosionAnimation();</code>	
15	<code>}</code>	
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	ElficFlaute.cpp	Page 1/1
1	<code>#include "ElficFlaute.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 42</code>	
5	<code>#define HEIGHT_BODY 64</code>	
6		
7	<code>ElficFlaute::ElficFlaute(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::ElficFlaute),WIDTH_BODY,HEIGHT_BODY, We</code>	
9	<code>aponID::ElficFlaute) {}</code>	
10	<code>void ElficFlaute::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void ElficFlaute::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void ElficFlaute::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>ElficFlaute::~ElficFlaute(){}</code>	

jul 21, 20 15:20	ElfHead.h	Page 1/1
1	<code>#ifndef ELFHEAD_H</code>	
2	<code>#define ELFHEAD_H</code>	
3		
4	<code>#include "Head.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class ElfHead: public Head {</code>	
8	<code>public:</code>	
9	<code>ElfHead(const TextureManager& manager);</code>	
10		
11	<code>virtual void render(int posX, int posY);</code>	
12		
13	<code>virtual void update(int dir);</code>	
14		
15	<code>~ElfHead();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	ElfHead.cpp	Page 1/1
1	<code>#include "ElfHead.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 16</code>	
6		
7	<code>ElfHead::ElfHead(const TextureManager& manager) :</code>	
8	<code>Head(manager.getTexture(TextureID::ElfHead),WIDTH_HEAD,HEIGHT_HEAD,HeadID::Elf)</code>	
9	<code>{}</code>	
10	<code>void ElfHead::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void ElfHead::update(int dir){</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>ElfHead::~ElfHead(){}</code>	

jul 21, 20 15:20	DwarfHead.h	Page 1/1
1	<code>#ifndef DWARFHEAD_H</code>	
2	<code>#define DWARFHEAD_H</code>	
3		
4	<code>#include "Head.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class DwarfHead: public Head {</code>	
8	<code>public:</code>	
9	<code>DwarfHead(const TextureManager& manager);</code>	
10		
11	<code>virtual void render(int posX, int posY);</code>	
12		
13	<code>virtual void update(int dir);</code>	
14		
15	<code>~DwarfHead();</code>	
16		
17	<code>};</code>	
18		
19	<code>#endif</code>	

jul 21, 20 15:20	DwarfHead.cpp	Page 1/1
1	<code>#include "DwarfHead.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_HEAD 17</code>	
5	<code>#define HEIGHT_HEAD 19</code>	
6		
7	<code>DwarfHead::DwarfHead(const TextureManager& manager) :</code>	
8	<code>Head(manager.getTexture(TextureID::DwarfHead),WIDTH_HEAD,HEIGHT_HEAD,HeadID::DwarfHead) {}</code>	
9		
10	<code>void DwarfHead::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcHead = {this->width*this->direction, this->height*0, this->width,</code>	
12	<code>this->height};</code>	
13	<code>SDL_Rect dstHead = {posX, posY+this->height-2, this->width, this->height};</code>	
14	<code>this->texture.render(srcHead, dstHead);</code>	
15	<code>}</code>	
16	<code>void DwarfHead::update(int dir){</code>	
17	<code>this->direction = dir;</code>	
18	<code>}</code>	
19		
20	<code>DwarfHead::~DwarfHead(){}</code>	

jul 21, 20 15:20	CureAnimation.h	Page 1/1
1	<code>#ifndef CUREANIMATION_H</code>	
2	<code>#define CUREANIMATION_H</code>	
3		
4	<code>#include "Animation.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6	<code>#include "../Effect.h"</code>	
7		
8	<code>class CureAnimation: public Animation {</code>	
9	<code>public:</code>	
10	<code>CureAnimation(const TextureManager& manager,const Effect& effect);</code>	
11		
12	<code>virtual void render(int x, int y);</code>	
13		
14	<code>~CureAnimation();</code>	
15		
16	<code>};</code>	
17		
18	<code>#endif</code>	

jul 21, 20 15:20	CureAnimation.cpp	Page 1/1
1	<code>#include "CureAnimation.h"</code>	
2		
3	<code>#define WIDTHFRAME 113</code>	
4	<code>#define HEIGHTFRAME 113</code>	
5	<code>#define FRAMES 20</code>	
6		
7	<code>CureAnimation::CureAnimation(const TextureManager& manager, const Effect& effect)</code>	
8	<code>: Animation(manager.getTexture(TextureID::HealthEffect), WIDTHFRAME, HEIGHTFRAME, effect, FRAMES) {}</code>	
9		
10	<code>void CureAnimation::render(int x, int y) {</code>	
11	<code>if (this->finish)</code>	
12	<code>return;</code>	
13	<code>if (!this->effectPlayed) {</code>	
14	<code> this->effect.playEffect(0, 64);</code>	
15	<code> this->effectPlayed = true;</code>	
16	<code>}</code>	
17	<code>int row = this->totalFrames/4;</code>	
18	<code>SDL_Rect src = {this->width*(this->frame % row),</code>	
19	<code> this->height*int(floor(this->frame/row)), this->width, this</code>	
20	<code>->height};</code>	
21	<code>SDL_Rect dst = {x-int(this->width/2.5), y-int(this->height/3), this->width, this</code>	
22	<code>->height};</code>	
23	<code> this->texture.render(src, dst);</code>	
24	<code>}</code>	
25	<code>CureAnimation::~CureAnimation() {}</code>	

jul 21, 20 15:20	Crosier.h	Page 1/1
1	<code>#ifndef CROSIER_H</code>	
2	<code>#define CROSIER_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class Crosier: public Weapon {</code>	
8	<code>private:</code>	
9	<code> void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code> Crosier(const TextureManager& manager);</code>	
12		
13	<code> virtual void render(int posX, int posY);</code>	
14		
15	<code> virtual void update(double dt, Direction dir);</code>	
16		
17	<code> ~Crosier();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	Crosier.cpp	Page 1/1
1	<code>#include "Crosier.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>Crosier::Crosier(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::Crosier),WIDTH_BODY,HEIGHT_BODY, Weapon</code>	
9	<code>ID::Crosier) {}</code>	
10	<code>void Crosier::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void Crosier::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void Crosier::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>Crosier::~Crosier(){}</code>	

jul 21, 20 15:20	CompoundArc.h	Page 1/1
1	<code>#ifndef COMPOUNDARC_H</code>	
2	<code>#define COMPOUNDARC_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class CompoundArc: public Weapon {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>CompoundArc(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~CompoundArc();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	CompoundArc.cpp	Page 1/1
1	<code>#include "CompoundArc.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>CompoundArc::CompoundArc(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::CompoundArc),WIDTH_BODY,HEIGHT_BODY, We</code>	
9	<code>aponID::CompoundArc) {}</code>	
10	<code>void CompoundArc::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void CompoundArc::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void CompoundArc::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2 v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>CompoundArc::~CompoundArc(){}</code>	

jul 21, 20 15:20	Body.h	Page 1/1
1	<code>#ifndef BODY_H</code>	
2	<code>#define BODY_H</code>	
3		
4	<code>#include "Item.h"</code>	
5	<code>#include "../Texture.h"</code>	
6	<code>#include "../common/Identificators.h"</code>	
7		
8	<code>class Body: public Item {</code>	
9	<code>protected:</code>	
10	<code>int frame{0};</code>	
11	<code>float elapsed{0.0};</code>	
12	<code>float animationSpeed{30.0};</code>	
13	<code>int totalFrames{5};</code>	
14	<code>BodyID id{BodyID::Nothing};</code>	
15	<code>Direction direction{Direction::down};</code>	
16	<code>public:</code>	
17	<code>Body(const Texture& texture, const int width, const int height, BodyID id =</code>	
18	<code>BodyID::Nothing);</code>	
19		
20	<code>virtual void render(int posX, int posY) = 0;</code>	
21	<code>void update(double dt, Direction dir);</code>	
22		
23	<code>void setAnimationSpeed(float speed);</code>	
24		
25	<code>int getHeight() const;</code>	
26		
27	<code>BodyID getId()const;</code>	
28		
29	<code>~Body();</code>	
30	<code>};</code>	
31		
32	<code>#endif</code>	

jul 21, 20 15:20	Body.cpp	Page 1/1
1	<code>#include "Body.h"</code>	
2		
3	<code>Body::Body(const Texture& texture, const int width, const int height, BodyID id)</code>	
4	<code>{</code>	
5	<code>Item(texture, width, height), id(id) {}</code>	
6	<code>void Body::update(double dt, Direction dir) {</code>	
7	<code>if (this->direction != dir) {</code>	
8	<code> this->direction = dir;</code>	
9	<code> this->elapsed = 0;</code>	
10	<code>}</code>	
11	<code> this->elapsed += dt;</code>	
12	<code> this->frame = int(this->elapsed/this->animationSpeed) % this->totalFrames</code>	
13	<code>;</code>	
14	<code>}</code>	
15	<code>void Body::setAnimationSpeed(float speed) {</code>	
16	<code> this->animationSpeed = speed;</code>	
17	<code>}</code>	
18		
19	<code>int Body::getHeight() const{</code>	
20	<code> return this->height;</code>	
21	<code>}</code>	
22		
23	<code>BodyID Body::getId() const{</code>	
24	<code> return this->id;</code>	
25	<code>}</code>	
26		
27	<code>Body::~Body() {}</code>	

jul 21, 20 15:20	BlueTunic.h	Page 1/1
1	<code>#ifndef BLUETONIC_H</code>	
2	<code>#define BLUETONIC_H</code>	
3		
4	<code>#include "Body.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class BlueTunic: public Body {</code>	
8	<code>private:</code>	
9	<code> void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code> BlueTunic(const TextureManager& manager);</code>	
12		
13	<code> virtual void render(int posX, int posY);</code>	
14		
15	<code> virtual void update(double dt, Direction dir);</code>	
16		
17	<code> ~BlueTunic();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	BlueTunic.cpp	Page 1/1
1	#include "BlueTunic.h"	
2	#include "../TextureID.h"	
3		
4	#define WIDTH_BODY 25	
5	#define HEIGHT_BODY 45	
6		
7	BlueTunic::BlueTunic(const TextureManager& manager) :	
8	Body(manager.getTexture(TextureID::BlueTunic),WIDTH_BODY,HEIGHT_BODY,BodyID::BlueTunic) {}	
9		
10	void BlueTunic::render(int posX, int posY) {	
11	SDL_Rect srcHead = { this ->width*frame, this ->height*int(direction), this ->width, this ->height};	
12	SDL_Rect dstHead = {posX, posY, this ->width, this ->height};	
13	this ->texture.render(srcHead, dstHead);	
14	}	
15		
16	void BlueTunic::update(double dt,Direction dir) {	
17	setDirection(int(dir));	
18	Body::update(dt,dir);	
19	}	
20		
21		
22	void BlueTunic::setDirection(int direction) {	
23	if (direction == 0 ∨ direction == 1)	
24	this ->totalFrames = 6;	
25	if (direction == 2 ∨ direction == 3)	
26	this ->totalFrames = 5;	
27	}	
28		
29	BlueTunic::~BlueTunic(){}	

jul 21, 20 15:20	BlueCommonBody.h	Page 1/1
1	#ifndef BLUECOMMONBODY_H	
2	#define BLUECOMMONBODY_H	
3		
4	#include "Body.h"	
5	#include "../TextureManager.h"	
6		
7	class BlueCommonBody: public Body {	
8	private:	
9	void setDirection(int direction);	
10	public:	
11	BlueCommonBody(const TextureManager& manager);	
12		
13	virtual void render(int posX, int posY);	
14		
15	virtual void update(double dt,Direction dir);	
16		
17	~BlueCommonBody();	
18		
19	};	
20		
21	#endif	

jul 21, 20 15:20	BlueCommonBody.cpp	Page 1/1
	<pre> 1 #include "BlueCommonBody.h" 2 #include "../TextureID.h" 3 4 #define WIDTH_BODY 25 5 #define HEIGHT_BODY 45 6 7 BlueCommonBody::BlueCommonBody(const TextureManager& manager): 8 Body(manager.getTexture(TextureID::BlueCommonBody),WIDTH_BODY,HEIGHT_BODY,Bo 9 dyID::BlueCommon) {} 10 11 void BlueCommonBody::render(int posX, int posY) { 12 SDL_Rect srcHead = {this->width*frame, this->height*int(direction), this->wid 13 th, this->height}; 14 SDL_Rect dstHead = {posX, posY, this->width, this->height}; 15 this->texture.render(srcHead, dstHead); 16 } 17 18 void BlueCommonBody::update(double dt,Direction dir) { 19 setDirection(int(dir)); 20 Body::update(dt,dir); 21 } 22 23 void BlueCommonBody::setDirection(int direction) { 24 if (direction == 0 v direction == 1) 25 this->totalFrames = 6; 26 if (direction == 2v direction == 3) 27 this->totalFrames = 5; 28 } 29 30 BlueCommonBody::~BlueCommonBody(){} </pre>	

jul 21, 20 15:20	BankerBody.h	Page 1/1
	<pre> 1 #ifndef BANKERBODY_H 2 #define BANKERBODY_H 3 4 #include "Body.h" 5 #include "../TextureManager.h" 6 7 class BankerBody: public Body { 8 private: 9 void setDirection(int direction); 10 public: 11 BankerBody(const TextureManager& manager); 12 13 virtual void render(int posX, int posY); 14 15 virtual void update(double dt,Direction dir); 16 17 ~BankerBody(); 18 }; 19 20 #endif </pre>	

jul 21, 20 15:20	BankerBody.cpp	Page 1/1
1	<code>#include "BankerBody.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3	<code>#include "../common/Identificators.h"</code>	
4		
5	<code>#define WIDTH_BODY 25</code>	
6	<code>#define HEIGHT_BODY 45</code>	
7		
8	<code>BankerBody::BankerBody(const TextureManager& manager) :</code>	
9	<code>Body(manager.getTexture(TextureID::Banker),WIDTH_BODY,HEIGHT_BODY,BodyID::Banke</code>	
10	<code>r) {}</code>	
11	<code>void BankerBody::render(int posX, int posY) {</code>	
12	<code>SDL_Rect srcBody = {this->width*frame, this->height*int(direction), this->wid</code>	
13	<code>th, this->height};</code>	
14	<code>SDL_Rect dstBody = {posX, posY, this->width, this->height};</code>	
15	<code>this->texture.render(srcBody, dstBody);</code>	
16	<code>}</code>	
17	<code>void BankerBody::update(double dt,Direction dir) {</code>	
18	<code>setDirection(int(dir));</code>	
19	<code>Body::update(dt,dir);</code>	
20	<code>}</code>	
21		
22	<code>void BankerBody::setDirection(int direction) {</code>	
23	<code>if (direction == 0 v direction == 1)</code>	
24	<code>this->totalFrames = 6;</code>	
25	<code>if (direction == 2v direction == 3)</code>	
26	<code>this->totalFrames = 5;</code>	
27	<code>}</code>	
28		
29	<code>BankerBody::~BankerBody()= default;</code>	

jul 21, 20 15:20	Ax.h	Page 1/1
1	<code>#ifndef AX_H</code>	
2	<code>#define AX_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class Ax: public Weapon {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>Ax(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~Ax();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	Ax.cpp	Page 1/1
1	<code>#include "Ax.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>Ax::Ax(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::Ax),WIDTH_BODY,HEIGHT_BODY, WeaponID::A</code>	
9	<code>x) {}</code>	
10	<code>void Ax::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void Ax::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void Ax::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>Ax::~Ax(){}</code>	

jul 21, 20 15:20	AshStick.h	Page 1/1
1	<code>#ifndef ASHSTICK_H</code>	
2	<code>#define ASHSTICK_H</code>	
3		
4	<code>#include "Weapon.h"</code>	
5	<code>#include "../TextureManager.h"</code>	
6		
7	<code>class AshStick: public Weapon {</code>	
8	<code>private:</code>	
9	<code>void setDirection(int direction);</code>	
10	<code>public:</code>	
11	<code>AshStick(const TextureManager& manager);</code>	
12		
13	<code>virtual void render(int posX, int posY);</code>	
14		
15	<code>virtual void update(double dt,Direction dir);</code>	
16		
17	<code>~AshStick();</code>	
18		
19	<code>};</code>	
20		
21	<code>#endif</code>	

jul 21, 20 15:20	AshStick.cpp	Page 1/1
1	<code>#include "AshStick.h"</code>	
2	<code>#include "../TextureID.h"</code>	
3		
4	<code>#define WIDTH_BODY 25</code>	
5	<code>#define HEIGHT_BODY 45</code>	
6		
7	<code>AshStick::AshStick(const TextureManager& manager):</code>	
8	<code>Weapon(manager.getTexture(TextureID::AshStick),WIDTH_BODY,HEIGHT_BODY, Weapo</code>	
9	<code>nID::AshStick) {}</code>	
10	<code>void AshStick::render(int posX, int posY) {</code>	
11	<code>SDL_Rect srcWeapon = {this->width*frame, this->height*int(direction), this->w</code>	
12	<code>idth, this->height};</code>	
13	<code>SDL_Rect dstWeapon = {posX, posY, this->width, this->height};</code>	
14	<code>this->texture.render(srcWeapon, dstWeapon);</code>	
15	<code>}</code>	
16	<code>void AshStick::update(double dt,Direction dir) {</code>	
17	<code>setDirection(int(dir));</code>	
18	<code>Weapon::update(dt,dir);</code>	
19	<code>}</code>	
20		
21	<code>void AshStick::setDirection(int direction) {</code>	
22	<code>if (direction == 0 v direction == 1)</code>	
23	<code>this->totalFrames = 6;</code>	
24	<code>if (direction == 2 v direction == 3)</code>	
25	<code>this->totalFrames = 5;</code>	
26	<code>}</code>	
27		
28		
29	<code>AshStick::~AshStick(){}</code>	

jul 21, 20 15:20	Animation.h	Page 1/1
1	<code>#ifndef ANIMATION_H</code>	
2	<code>#define ANIMATION_H</code>	
3		
4	<code>#include "../Effect.h"</code>	
5	<code>#include "../Texture.h"</code>	
6		
7	<code>class Animation {</code>	
8	<code>protected:</code>	
9	<code>const Effect& effect;</code>	
10	<code>const Texture& texture;</code>	
11	<code>const int width; //Ancho del Frame</code>	
12	<code>const int height; //Alto del Frame</code>	
13	<code>int totalFrames{0};</code>	
14	<code>int frame{0};</code>	
15	<code>float elapsed{0.0};</code>	
16	<code>float animationSpeed{45.0f};</code>	
17	<code>bool finish{false};</code>	
18	<code>bool effectPlayed{false};</code>	
19	<code>public:</code>	
20		
21	<code>Animation(const Texture& texture, const int width, const int height, const E</code>	
22	<code>ffect& effect,int totalFrames);</code>	
23	<code>virtual void render(int x, int y) = 0;</code>	
24		
25	<code>void update();</code>	
26		
27	<code>bool finished() const;</code>	
28		
29	<code>~Animation();</code>	
30	<code>};</code>	
31		
32		
33	<code>#endif</code>	

jul 21, 20 15:20	Animation.cpp	Page 1/1
	<pre> 1 #include "Animation.h" 2 3 Animation::Animation(const Texture& texture, const int width, const int height, 4 const Effect& effect, int totalFrames) : effect(effect), texture(texture), 5 width(width), height(height), totalFrames(totalFrames) {} 6 7 void Animation::update() { 8 this->frame++; 9 if (this->frame ≥ this->totalFrames) 10 this->finish = true; 11 this->frame = this->frame % this->totalFrames; 12 } 13 14 bool Animation::finished() const { 15 return this->finish; 16 } 17 18 Animation::~Animation() {} </pre>	

jul 21, 20 15:20	GameMap.h	Page 1/1
	<pre> 1 #ifndef ARGENTUM_GAMEMAP_H 2 #define ARGENTUM_GAMEMAP_H 3 4 #include "Tile.h" 5 #include "../common/TiledMap.h" 6 #include "Texture.h" 7 #include "Camera.h" 8 #include "../common/Chrono.h" 9 #include "ui/Ulh.h" 10 #include <vector> 11 #include <map> 12 13 class GameMap { 14 private: 15 std::vector<Tile> tiles; 16 std::vector<Tile> groundTiles; 17 std::map<int, Texture> tileSetMap; 18 SDL_Renderer& renderer; 19 uint16_t rows, columns; 20 uint16_t width, height; 21 void _loadTileSets(const std::vector<TileSet>&, SDL_Renderer&); 22 public: 23 explicit GameMap(const TiledMap&, SDL_Renderer&); 24 25 virtual ~GameMap(); 26 27 void drawHighLayers(Camera& camera); 28 void drawGround(Camera& camera); 29 30 int getMapWidth() const; 31 int getMapHeight() const; 32 33 }; 34 35 36 #endif // ARGENTUM_GAMEMAP_H </pre>	

jul 21, 20 15:20	GameMap.cpp	Page 1/2
	<pre> 1 #include <iostream> 2 #include "GameMap.h" 3 4 GameMap::GameMap(const TiledMap & tiledMap, SDL_Renderer& renderer) : renderer(r 5 ender) { 6 this->rows = tiledMap.getHeight(); 7 this->columns = tiledMap.getWidth(); 8 this->width = tiledMap.getWidth() * tiledMap.getTileWidth(); 9 this->height = tiledMap.getHeight() * tiledMap.getTileHeight(); 10 _loadTileSets(tiledMap.getTileSets(), renderer); 11 12 uint8_t tileWidth = tiledMap.getTileWidth(); 13 uint8_t tileHeight = tiledMap.getTileHeight(); 14 15 for (auto& layer : tiledMap.getTileLayers()) { 16 std::vector<uint16_t> layerData = layer.getData(); 17 18 for (int y = 0; y < rows; ++y) { 19 for (int x = 0; x < columns; ++x) { 20 int tileIndex = x + (y * columns); 21 int curGid = layerData[tileIndex]; 22 23 if (curGid == 0) { 24 continue; 25 } 26 27 int tileSetGid = -1; 28 for (auto& ts : tileSetMap) { 29 if (ts.first <= curGid) { 30 tileSetGid = ts.first; 31 } 32 } 33 34 if (tileSetGid == -1) { 35 continue; 36 } 37 Texture& loadTexture = tileSetMap.at(tileSetGid); 38 39 curGid -= tileSetGid; 40 41 int regionX = (curGid % (loadTexture.getWidth() / tileWidth)) * 42 tileWidth; 43 int regionY = (curGid / (loadTexture.getWidth() / tileWidth)) * 44 tileHeight; 45 46 int xPos = x * tileWidth; 47 int yPos = y * tileHeight; 48 49 Tile aTile(xPos, yPos, regionX, regionY, tileWidth, tileHeight, 50 loadTexture); 51 if (layer.isGroundLayer()) { 52 groundTiles.push_back(aTile); 53 } else { 54 tiles.push_back(aTile); 55 } 56 } 57 } 58 } 59 int GameMap::getMapHeight() const { 60 return this->height; 61 } 62 int GameMap::getMapWidth() const { </pre>	

jul 21, 20 15:20	GameMap.cpp	Page 2/2
	<pre> 63 return this->width; 64 } 65 66 void GameMap::_loadTileSets(const std::vector<TileSet> &tileSets, SDL_Renderer & 67 renderer) { 68 SDL_Color textColor = {0x0, 0x0, 0x0}; 69 for (auto& aTileSet : tileSets) { 70 Texture aTexture(aTileSet.getImage(), renderer, textColor); 71 tileSetMap.insert(std::pair<int, Texture>(aTileSet.getFirstgid(), std::m 72 ove(aTexture))); 73 } 74 75 void GameMap::drawHighLayers(Camera& camera) { 76 for (Tile& tile : tiles) { 77 tile.draw(renderer, camera); 78 } 79 } 80 81 void GameMap::drawGround(Camera& camera) { 82 for (Tile& tile : groundTiles) { 83 tile.draw(renderer, camera); 84 } 85 } 86 87 GameMap::~GameMap() = default; 88 89 </pre>	

jul 21, 20 15:20	Game.h	Page 1/1
1	#ifndef GAME_H	
2	#define GAME_H	
3		
4	#include <memory>	
5	#include <unordered_map>	
6	#include "Window.h"	
7	#include "TextureManager.h"	
8	#include "MusicManager.h"	
9	#include "GameMap.h"	
10	#include "Player.h"	
11	#include "NPC.h"	
12	#include "Camera.h"	
13	#include "ui/UL.h"	
14	#include "../common/InputQueue.h"	
15	#include "../common/DataQueue.h"	
16	#include "../common/Socket.h"	
17	#include "../common/Identificators.h"	
18	#include "../common/CommunicationProtocol.h"	
19	#include "Dispatcher.h"	
20	#include "Receiver.h"	
21		
22	<i>//Clase destinada a coordinar el juego del lado del cliente</i>	
23	<i>//Realiza la conexiÃ³n inicial y luego, irÃ¡ realizando los updates</i>	
24	<i>//recibidos desde el server para renderizarlos y enviando</i>	
25	<i>//comandos al mismo. A su vez tendrÃ¡ otros 2 Threads que</i>	
26	<i>//establecerÃ¡n la comunicaciÃ³n con el server</i>	
27		
28	class Game {	
29	private:	
30	Window window;	
31	CommunicationProtocol protocol;	
32	TextureManager textureManager;	
33	MusicManager musicManager;	
34	std::unordered_map<uint, NPC> npcs;	
35	std::shared_ptr<Player> player = nullptr;	
36	std::shared_ptr<GameMap> map = nullptr;	
37	InputQueue commandQueue;	
38	DataQueue dataQueue;	
39	Dispatcher dispatcher;	
40	Receiver receiver;	
41	std::shared_ptr<Camera> camera{nullptr};	
42	std::shared_ptr<UI> ui{nullptr};	
43		
44	RaceID translateRace(const std::string& race);	
45	GameClassID translateGameClass(const std::string& gameClass);	
46	void recieveMapAndPlayer();	
47		
48	void update();	
49	void render();	
50	void sounds();	
51	void close();	
52	public:	
53	Game();	
54	<i>//Se conecta con el servidor, enviando la raza y clase elegida</i>	
55	<i>//Recibe el mapa estÃ¡tico y el primer playerInfo que tendrÃ¡ la</i>	
56	<i>//informaciÃ³n inicial para que el jugador pueda comenzar el juego.</i>	
57	bool init(char* argv[]);	
58		
59	<i>//Gameloop principal del cliente.</i>	
60	int run();	
61		
62	~Game();	
63		
64	};	
65		
66	#endif	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Game.cpp	Page 1/5
1	#include "Game.h"	
2	#include <memory>	
3	#include <stdexcept>	
4	#include "Presentation.h"	
5	#include <arpa/inet.h>	
6	#include <iostream>	
7	#include "../common/Decoder.h"	
8	#include "../common/SocketException.h"	
9	#include <algorithm>	
10	#include "../common/Random.h"	
11		
12	#define WRONGRACE "Raza invalida. Seleccione entre: elfo, gnomo, humano, enano."	
13	#define WRONGCLASS "Clase invalida. Seleccione entre: mago, clerigo, paladin, guerrero"	
14	#define ARGENTUM "Argentum Taller"	
15	#define INITERROR "Error en Game::init: "	
16	#define GAMELOOPTIME 1000000/45.0	
17	#define PLAYERINFOMSG 0x01	
18	#define OBJECTSINFOMSG 0x02	
19	#define INTERACTMSG 0x05	
20		
21		
22	Game::Game() : window(ARGENTUM), protocol(), textureManager(window.getRenderer()	
23), musicManager(), npcs(), commandQueue(true), dataQueue(false),	
24	dispatcher(protocol,commandQueue), receiver(protocol,dataQueue) {}	
25		
26	void Game::recieveMapAndPlayer() {	
27		
28	Message msgMap = this ->protocol.receive();	
29	TiledMap tiledMap = Decoder::decodeMap(msgMap);	
30		
31	Message msgPlayerInfo = this ->protocol.receive();	
32	PlayerInfo info = Decoder::decodePlayerInfo(msgPlayerInfo);	
33		
34	this ->map = std::make_shared<GameMap>(tiledMap, this ->window.getRenderer());	
35	this ->player = std::make_shared<Player>(this ->textureManager, info, this ->mus	
36	icManager);	
37	}	
38	bool Game::init(char* argv[]) {	
39	try{	
40	this ->protocol.connect(argv[3], argv[4]);	
41	std::cout << "me conecta " << argv[4] << std::endl;	
42	std::vector<uint8_t> initMsg;	
43	initMsg = Decoder::encodeInit(translateRace(argv[1]),translateGameClass(
44	argv[2]));	
45	this ->protocol.send(initMsg);	
46	this ->textureManager.loadTextures();	
47	this ->musicManager.loadSounds();	
48	recieveMapAndPlayer();	
49		
50	this ->camera = std::make_shared<Camera>(this ->window, this ->map->getMapWi	
51	dth(), this ->map->getMapHeight());	
52	this ->ui = std::make_shared<UI>(this ->window, &(this ->player), this ->tex	
53	tureManager, this ->musicManager);	
54		
55	} catch (const SocketException& e) {	
56	std::cout << INITERROR << e.what() << std::endl;	
57	return false;	
58	}	
59	return true;	
60		
61	int Game::run() {	
62	const Music& musica = musicManager.getMusic(MusicID::BackGround);	
63	musica.playMusic(-1);	

195/217

jul 21, 20 15:20	Game.cpp	Page 2/5
62	musica.setVolume(MIX_MAX_VOLUME/4);	
63	Presentation presentation(window, textureManager);	
64	if (presentation.run())	
65	return 0;	
66		
67	this->dispatcher.start();	
68	this->receiver.start();	
69	bool quit = false;	
70	SDL_Event event;	
71		
72	Chrono chrono;	
73	double initLoop, endLoop, sleep;	
74	InputInfo input;	
75	while (!quit) {	
76	try{	
77	initLoop = chrono.lap();	
78	while (SDL_PollEvent(&event) != 0) {	
79	if (event.type == SDL_QUIT) {	
80	quit = true;	
81	break;	
82	}	
83	if(event.type == SDL_KEYDOWN) {	
84	if (event.key.keysym.sym == SDLK_m) {	
85	musica.pauseMusic();	
86	}	
87	input = player->handleEvent(event,*camera);	
88	this->commandQueue.push(input);	
89	input = ui->handleClick(event);	
90	this->commandQueue.push(input);	
91	window.handleEvent(event);	
92	}	
93		
94		
95	this->update();	
96	this->render();	
97	this->sounds();	
98	endLoop = chrono.lap();	
99	sleep = GAMELOOPTIME - (endLoop - initLoop);	
100	if (sleep > 0)	
101	usleep(sleep);	
102	} catch (const std::exception& e) {	
103	quit = true;	
104	std::cerr << e.what() << std::endl;	
105	} catch (...) {	
106	quit = true;	
107	std::cerr << "Unkown Error in Game::run()" << std::endl;	
108	}	
109	}	
110	close();	
111	return 0;	
112	}	
113		
114		
115	void Game::update() {	
116	Message msg;	
117	PlayerInfo playerInfo;	
118	std::vector<GameObjectInfo> objects;	
119	std::unordered_map<uint,NPC> newNpcs;	
120	NPCInfo items;	
121	while (!this->dataQueue.empty()){	
122	msg = this->dataQueue.pop();	
123	if (msg.getType() == PLAYERINFOFMSG) {	
124	playerInfo = Decoder::decodePlayerInfo(msg);	
125	this->player->updatePlayerInfo(playerInfo);	
126	} else if (msg.getType() == OBJECTSINFOFMSG) {	
127	objects = Decoder::decodeGameObjects(msg);	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Game.cpp	Page 3/5
128	newNpcs.clear();	
129	for (GameObjectInfo& npc : objects) {	
130	auto iter = this->npcs.find(npc.getId());	
131	if (iter == this->npcs.end()){	
132	NPC aNPC(this->textureManager,npc,this->musicManager);	
133	newNpcs.insert({npc.getId(),aNPC});	
134	} else {	
135	(*iter).second.updatePlayerInfo(npc);	
136	newNpcs.insert({(*iter).first,(*iter).second});	
137	}	
138		
139	this->npcs.swap(newNpcs);	
140	} else if (msg.getType() == INTERACTMSG) {	
141	items = Decoder::decodeNPCInfo(msg);	
142	this->ui->setNPCInfo(items);	
143	}	
144		
145	this->player->update(GAMELOOPTIME);	
146	for (auto& npc: this->npcs)	
147	npc.second.update(GAMELOOPTIME);	
148	}	
149		
150		
151	bool comparation(Character* c1, Character* c2) {	
152	return c1->getPosition().y < c2->getPosition().y;	
153	}	
154		
155	void Game::render() {	
156	window.clearScreen();	
157		
158	ui->render();	
159	Point* center = this->player->getCenter();	
160	camera->setPlayer(center);	
161	camera->render(*center);	
162	this->map->drawGround(*camera);	
163		
164	std::vector<Character*> characters;	
165	std::vector<Character*> items;	
166		
167	characters.push_back(&(*this->player));	
168		
169	for (auto& aNPC : this->npcs) {	
170	if(aNPC.second.isItem()){	
171	items.push_back(&(aNPC.second));	
172	} else {	
173	characters.push_back(&(aNPC.second));	
174	}	
175	}	
176		
177	std::sort(characters.begin(),characters.end(),comparation);	
178	std::sort(items.begin(),items.end(),comparation);	
179		
180	for (auto& aItem : items) {	
181	aItem->render(*camera);	
182	}	
183		
184	for (auto& aCharacter : characters) {	
185	aCharacter->render(*camera);	
186	}	
187		
188	this->map->drawHighLayers(*camera);	
189		
190	this->window.render();	
191		
192	}	
193		

196/217

jul 21, 20 15:20	Game.cpp	Page 4/5
194	void Game::sounds() {	
195	int random = Random::get(1,5000);	
196	MusicID effectId = MusicID::Nothing;	
197	switch (random) {	
198	case 1:	
199	effectId = MusicID::Wind;	
200	break;	
201	case 2:	
202	effectId = MusicID::Eagle;	
203	break;	
204	case 3:	
205	effectId = MusicID::Wolf;	
206	break;	
207	case 4:	
208	effectId = MusicID::Raven;	
209	break;	
210	default:	
211	break;	
212	} if (effectId != MusicID::Nothing){	
213	const Effect& effect = this->musicManager.getEffect(effectId);	
214	effect.playEffect();	
215	}	
216	}	
217	}	
218		
219	void Game::close() {	
220	std::cout << "Cierro todo" << std::endl;	
221	this->dataQueue.close();	
222	this->commandQueue.close();	
223	this->receiver.stop();	
224	this->receiver.join();	
225	this->dispatcher.stop();	
226	this->dispatcher.join();	
227	}	
228		
229	RaceID Game::translateRace(const std::string& race){	
230	RaceID selectedRace;	
231	if (race == "elfo") {	
232	selectedRace = RaceID::Elf;	
233	} else if (race == "enano") {	
234	selectedRace = RaceID::Dwarf;	
235	} else if (race == "humano") {	
236	selectedRace = RaceID::Human;	
237	} else if (race == "gnomo") {	
238	selectedRace = RaceID::Gnome;	
239	} else {	
240	throw std::invalid_argument(WRONGRACE);	
241	}	
242	return selectedRace;	
243	}	
244		
245	GameClassID Game::translateGameClass(const std::string& gameClass){	
246	GameClassID selectedClass;	
247	if (gameClass == "mago") {	
248	selectedClass = GameClassID::Mage;	
249	} else if (gameClass == "clerigo") {	
250	selectedClass = GameClassID::Cleric;	
251	} else if (gameClass == "guerrero") {	
252	selectedClass = GameClassID::Warrior;	
253	} else if (gameClass == "paladin") {	
254	selectedClass = GameClassID::Paladin;	
255	} else {	
256	throw std::invalid_argument(WRONGCLASS);	
257	}	
258	return selectedClass;	
259	}	

jul 21, 20 15:20	Game.cpp	Page 5/5
260		
261	Game::~Game() {}	

jul 21, 20 15:20	Font.h	Page 1/1
	<pre> 1 #ifndef FONT_H 2 #define FONT_H 3 #include <SDL2/SDL_ttf.h> 4 #include <string> 5 6 class SDL_Texture; 7 class SDL_Color; 8 class SDL_Renderer; 9 10 class Font { 11 private: 12 TTF_Font* font; 13 int size; 14 SDL_Color color; 15 public: 16 Font(const std::string& path, const int size, SDL_Color color); 17 18 void setColor(SDL_Color color); 19 20 SDL_Texture* createText(const std::string& text, SDL_Renderer* renderer, int* w idht, int* height); 21 22 TTF_Font* getFont() const; 23 24 void setSize(int size); 25 26 void deleteText(SDL_Texture* text); 27 28 }; 29 ~Font(); 30 31 #endif </pre>	

jul 21, 20 15:20	Font.cpp	Page 1/1
	<pre> 1 #include "Font.h" 2 #include "../common/Exception.h" 3 #include "../common/Identificators.h" 4 5 Font::Font(const std::string& path, const int size, SDL_Color color) : size(size), 6 color(color) { 7 std::string root(ROOT_DIR+std::string("/") + path); 8 this->font = TTF_OpenFont(root.c_str(), this->size); 9 if (!this->font) 10 throw Exception("Fail TTF_OpenFont: %s", TTF_GetError()); 11 } 12 13 14 void Font::setColor(SDL_Color color) { 15 this->color = color; 16 } 17 18 void Font::setSize(int size) { 19 this->size = size; 20 } 21 22 SDL_Texture* Font::createText(const std::string& text, SDL_Renderer* renderer, i nt* width, int* height) { 23 SDL_Surface* textSurface = TTF_RenderText_Solid(this->font, text.c_str(), this ->color); 24 if (!textSurface) 25 throw Exception("Fail TTF_RenderText_Solid: %s", TTF_GetError()); 26 27 SDL_Texture* texture = SDL_CreateTextureFromSurface(renderer, textSurface); 28 if (!texture) 29 throw Exception("Fail SDL_CreateTextureFromSurface in createText: %s", SDL_GetError()); 30 31 *height = textSurface->h; 32 *width = textSurface->w; 33 SDL_FreeSurface(textSurface); 34 return texture; 35 } 36 37 TTF_Font* Font::getFont() const { 38 return this->font; 39 } 40 41 void Font::deleteText(SDL_Texture* text) { 42 SDL_DestroyTexture(text); 43 } 44 45 Font::~Font(){ 46 if(this->font) { 47 TTF_CloseFont(this->font); 48 this->font = nullptr; 49 } 50 } </pre>	

jul 21, 20 15:20	Effect.h	Page 1/1
1	<code>#ifndef EFFECT_H</code>	
2	<code>#define EFFECT_H</code>	
3		
4	<code>#include <SDL2/SDL_mixer.h></code>	
5	<code>#include <string></code>	
6		
7	<code>class Effect {</code>	
8	<code>private:</code>	
9	<code> Mix_Chunk* effect = nullptr;</code>	
10	<code> mutable int channel;</code>	
11	<code>public:</code>	
12	<code> explicit Effect(const std::string &fileName);</code>	
13	<code> Effect(Effect^ other);</code>	
14		
15	<code> void playEffect(int times = 0, int volume=-1) const;</code>	
16		
17	<code> void setDistance(int distance) const;</code>	
18		
19	<code> void pause() const;</code>	
20		
21	<code> ~Effect();</code>	
22	<code>};</code>	
23		
24		
25	<code>#endif</code>	

jul 21, 20 15:20	Effect.cpp	Page 1/1
1	<code>#include "Effect.h"</code>	
2	<code>#include "../common/Exception.h"</code>	
3	<code>#include <iostream></code>	
4		
5	<code>Effect::Effect(const std::string& fileName) {</code>	
6	<code> this->effect = Mix_LoadWAV(fileName.c_str());</code>	
7	<code> if(this->effect == NULL) {</code>	
8	<code> std::cout << "ERROR al cargar el archivo" << fileName << std::endl;</code>	
9	<code> }</code>	
10	<code> this->channel = -1;</code>	
11	<code> }</code>	
12		
13	<code>void Effect::playEffect(int times, int volume) const {</code>	
14	<code> Mix_VolumeChunk(this->effect, volume);</code>	
15	<code> this->channel = Mix_PlayChannel(-1, this->effect, times);</code>	
16	<code>}</code>	
17		
18	<code>void Effect::pause() const {</code>	
19	<code> Mix_Pause(this->channel);</code>	
20	<code>}</code>	
21		
22	<code>Effect::Effect(Effect ^other) {</code>	
23	<code> std::swap(this->effect, other->effect);</code>	
24	<code> std::swap(this->channel, other->channel);</code>	
25	<code>}</code>	
26		
27	<code>void Effect::setDistance(int distance) const {</code>	
28	<code> Mix_SetDistance(this->channel, distance);</code>	
29	<code>}</code>	
30		
31	<code>Effect::~Effect() {</code>	
32	<code> if (this->effect) {</code>	
33	<code> Mix_FreeChunk(this->effect);</code>	
34	<code> this->effect = nullptr;</code>	
35	<code> }</code>	
36	<code>}</code>	

jul 21, 20 15:20	Dispatcher.h	Page 1/1
1	<code>#ifndef DISPATCHER_H</code>	
2	<code>#define DISPATCHER_H</code>	
3		
4	<code>#include "../common/Thread.h"</code>	
5	<code>#include "../common/InputQueue.h"</code>	
6	<code>#include "../common/CommunicationProtocol.h"</code>	
7		
8	<code>//Thread encargado de enviar al servidor los comandos ingresados</code>	
9	<code>//por el jugador.</code>	
10	<code>class Dispatcher: public Thread {</code>	
11	<code>private:</code>	
12	<code>InputQueue& queue;</code>	
13	<code>std::atomic<bool> keepTalking;</code>	
14	<code>CommunicationProtocol& protocol;</code>	
15	<code>public:</code>	
16		
17	<code>Dispatcher(CommunicationProtocol& protocol, InputQueue& queue);</code>	
18		
19	<code>virtual void run();</code>	
20		
21	<code>bool is_alive() const;</code>	
22		
23	<code>void stop();</code>	
24		
25	<code>virtual ~Dispatcher();</code>	
26		
27	<code>};</code>	
28		
29	<code>#endif</code>	

jul 21, 20 15:20	Dispatcher.cpp	Page 1/1
1	<code>#include "Dispatcher.h"</code>	
2	<code>#include <vector></code>	
3	<code>#include <iostream></code>	
4	<code>#include "../common/SocketException.h"</code>	
5	<code>#include "../common/Decoder.h"</code>	
6		
7	<code>#define UNKNOWN_ERROR "Unknown Error"</code>	
8	<code>#define ERRORSOCKET "Error en la comunicaci3n en Dispatcher::run() "</code>	
9	<code>#define ERRORDISPATCHER "Error en Dispatcher::run() "</code>	
10		
11	<code>Dispatcher::Dispatcher(CommunicationProtocol& protocol, InputQueue& queue) :</code>	
12	<code>queue(queue), keepTalking(true), protocol(protocol) {}</code>	
13		
14	<code>void Dispatcher::run() {</code>	
15	<code>InputInfo info;</code>	
16	<code>std::vector<uint8_t> msg;</code>	
17	<code>while (this->keepTalking) {</code>	
18	<code>try{</code>	
19	<code>info = this->queue.pop();</code>	
20	<code>if (info.input != InputID::nothing) {</code>	
21	<code>msg = Decoder::encodeCommand(info);</code>	
22	<code>this->protocol.send(msg);</code>	
23	<code>}</code>	
24	<code>} catch (const SocketException& e) {</code>	
25	<code>std::cerr << ERRORSOCKET << e.what() << std::endl;</code>	
26	<code>this->keepTalking = false;</code>	
27	<code>} catch(const std::exception& e){</code>	
28	<code>std::cerr << ERRORDISPATCHER << e.what() << std::endl;</code>	
29	<code>this->keepTalking = false;</code>	
30	<code>}catch (...){</code>	
31	<code>this->keepTalking = false;</code>	
32	<code>std::cerr << UNKNOWN_ERROR << std::endl;</code>	
33	<code>}</code>	
34	<code>}</code>	
35	<code>}</code>	
36		
37		
38	<code>bool Dispatcher::is_alive() const {</code>	
39	<code>return this->keepTalking;</code>	
40	<code>}</code>	
41		
42	<code>void Dispatcher::stop() {</code>	
43	<code>this->keepTalking = false;</code>	
44	<code>this->protocol.stop();</code>	
45	<code>}</code>	
46		
47	<code>Dispatcher::~Dispatcher() {}</code>	

jul 21, 20 15:20	client_main.cpp	Page 1/1
1	<code>#include "../common/Socket.h"</code>	
2	<code>#include "../common/CommunicationProtocol.h"</code>	
3	<code>#include "Game.h"</code>	
4	<code>#include <iostream></code>	
5	<code>#define ARGENTUM "Argentum Online"</code>	
6	<code>#define INVALID_ARGUMENTS "USO: argentum_client <raza> <clase> <host> <puerto>"</code>	
7	<code>#define GAMELOOPTIME 1000000/30.0</code>	
8		
9	<code>int main(int argc, char* argv[]) {</code>	
10	<code> if (argc != 5) {</code>	
11	<code> std::cout << INVALID_ARGUMENTS << std::endl;</code>	
12	<code> return 1;</code>	
13	<code> }</code>	
14	<code> Game game;</code>	
15	<code> if (!game.init(argv)) {</code>	
16	<code> return 1;</code>	
17	<code> }</code>	
18	<code> game.run();</code>	
19	<code> return 0;</code>	
20		
21	<code>}</code>	
22		

jul 21, 20 15:20	StillState.h	Page 1/1
1	<code>#ifndef STILLSTATE_H</code>	
2	<code>#define STILLSTATE_H</code>	
3		
4	<code>#include "CharacterState.h"</code>	
5		
6	<code>class StillState: public CharacterState {</code>	
7	<code>public:</code>	
8	<code> StillState();</code>	
9	<code> ~StillState();</code>	
10		
11	<code> virtual InputInfo moveUp(Character& character);</code>	
12	<code> virtual InputInfo moveDown(Character& character);</code>	
13	<code> virtual InputInfo moveLeft(Character& character);</code>	
14	<code> virtual InputInfo moveRight(Character& character);</code>	
15	<code> virtual InputInfo stopMove(Character& character);</code>	
16		
17	<code> virtual InputInfo selectItem(Character& character, int item);</code>	
18	<code> virtual InputInfo dropItem(Character& character, int item);</code>	
19	<code> virtual InputInfo unequipItem(Character& character, int item);</code>	
20	<code> virtual InputInfo buyItem(Character& character, int item);</code>	
21	<code> virtual InputInfo sellItem(Character& character, int item);</code>	
22		
23	<code> virtual InputInfo deposit(Character& character, int item, bool isItem);</code>	
24	<code> virtual InputInfo retire(Character& character, int item, bool isItem);</code>	
25		
26	<code> virtual InputInfo meditate(Character& character);</code>	
27	<code> virtual InputInfo resurrect(Character& character);</code>	
28	<code> virtual InputInfo cure(Character& character);</code>	
29	<code> virtual InputInfo takeItem(Character& character);</code>	
30	<code> virtual InputInfo selectTarget(Character& character, Point position);</code>	
31	<code>};</code>	
32		
33	<code>#endif</code>	

jul 21, 20 15:20	StillState.cpp	Page 1/3
1	#include "StillState.h"	
2		
3	StillState::StillState() :	
4	CharacterState(CharacterStateID::Still){}	
5		
6	StillState::~StillState() {}	
7		
8	InputInfo StillState::moveUp(Character& character){	
9	InputInfo info;	
10	info.input = InputID::up;	
11	Point aux(0.0,0.0);	
12	info.position = aux;	
13	return info;	
14	}	
15		
16	InputInfo StillState::moveDown(Character& character) {	
17	InputInfo info;	
18	info.input = InputID::down;	
19	Point aux(0.0,0.0);	
20	info.position = aux;	
21	return info;	
22	}	
23		
24	InputInfo StillState::moveLeft(Character& character) {	
25	InputInfo info;	
26	info.input = InputID::left;	
27	Point aux(0.0,0.0);	
28	info.position = aux;	
29	return info;	
30	}	
31		
32	InputInfo StillState::moveRight(Character& character) {	
33	InputInfo info;	
34	info.input = InputID::right;	
35	Point aux(0.0,0.0);	
36	info.position = aux;	
37	return info;	
38	}	
39		
40	InputInfo StillState::stopMove(Character& character) {	
41	InputInfo info;	
42	info.input = InputID::stopMove;	
43	Point aux(0.0,0.0);	
44	info.position = aux;	
45	return info;	
46	}	
47		
48	InputInfo StillState::selectItem(Character& character, int item) {	
49	InputInfo info;	
50	Point aux(0.0,0.0);	
51	info.position = aux;	
52	info.input = InputID::equipItem;	
53	info.additional = item;	
54	return info;	
55	}	
56		
57		
58	InputInfo StillState::selectTarget(Character& character, Point position) {	
59	InputInfo info;	
60	info.position = position;	
61	info.input = InputID::selectTarget;	
62	return info;	
63	}	
64		
65	InputInfo StillState::meditate(Character& character) {	
66	InputInfo info;	

jul 21, 20 15:20	StillState.cpp	Page 2/3
67	Point aux(0.0,0.0);	
68	info.position = aux;	
69	info.input = InputID::meditate;	
70	return info;	
71	}	
72		
73	InputInfo StillState::resurrect(Character& character) {	
74	InputInfo info;	
75	Point aux(0.0,0.0);	
76	info.position = aux;	
77	info.input = InputID::resurrect;	
78	return info;	
79	}	
80		
81	InputInfo StillState::cure(Character& character) {	
82	InputInfo info;	
83	Point aux(0.0,0.0);	
84	info.position = aux;	
85	info.input = InputID::nothing;	
86	return info;	
87	}	
88		
89	InputInfo StillState::takeItem(Character& character) {	
90	InputInfo info;	
91	info.position = character.getPosition();	
92	info.input = InputID::takeItem;	
93	return info;	
94	}	
95		
96	InputInfo StillState::dropItem(Character& character, int item) {	
97	InputInfo info;	
98	Point aux(0.0,0.0);	
99	info.position = aux;	
100	info.input = InputID::dropItem;	
101	info.additional = item;	
102	return info;	
103	}	
104		
105	InputInfo StillState::buyItem(Character& character, int item) {	
106	InputInfo info;	
107	Point aux(0.0,0.0);	
108	info.position = aux;	
109	info.input = InputID::nothing;	
110	return info;	
111	}	
112		
113	InputInfo StillState::sellItem(Character& character, int item) {	
114	InputInfo info;	
115	Point aux(0.0,0.0);	
116	info.position = aux;	
117	info.input = InputID::nothing;	
118	return info;	
119	}	
120		
121	InputInfo StillState::retire(Character& character, int item, bool isItem) {	
122	InputInfo info;	
123	Point aux(0.0,0.0);	
124	info.position = aux;	
125	info.input = InputID::nothing;	
126	return info;	
127	}	
128		
129	InputInfo StillState::deposit(Character& character, int item, bool isItem) {	
130	InputInfo info;	
131	Point aux(0.0,0.0);	
132	info.position = aux;	

jul 21, 20 15:20	StillState.cpp	Page 3/3
133	info.input = InputID::nothing;	
134	return info;	
135	}	
136		
137	InputInfo StillState::unequipItem(Character& character, int item) {	
138	InputInfo info;	
139	Point aux(0.0,0.0);	
140	info.position = aux;	
141	info.input = InputID::unequipItem;	
142	info.additional = item;	
143	return info;	
144	}	

jul 21, 20 15:20	ResurrectState.h	Page 1/1
1	#ifndef RESURRECTSTATE_H	
2	#define RESURRECTSTATE_H	
3		
4	#include "CharacterState.h"	
5		
6	class ResurrectState: public CharacterState {	
7	public:	
8	ResurrectState();	
9	~ResurrectState();	
10		
11	virtual InputInfo moveUp(Character& character);	
12	virtual InputInfo moveDown(Character& character);	
13	virtual InputInfo moveLeft(Character& character);	
14	virtual InputInfo moveRight(Character& character);	
15	virtual InputInfo stopMove(Character& character);	
16		
17	virtual InputInfo selectItem(Character& character, int item);	
18	virtual InputInfo dropItem(Character& character, int item);	
19	virtual InputInfo unequipItem(Character& character, int item);	
20	virtual InputInfo buyItem(Character& character, int item);	
21	virtual InputInfo sellItem(Character& character, int item);	
22		
23	virtual InputInfo deposit(Character& character,int item, bool isItem);	
24	virtual InputInfo retire(Character& character,int item, bool isItem);	
25		
26	virtual InputInfo meditate(Character& character);	
27	virtual InputInfo resurrect(Character& character);	
28	virtual InputInfo cure(Character& character);	
29	virtual InputInfo takeItem(Character& character);	
30	virtual InputInfo selectTarget(Character& character, Point position);	
31	};	
32		
33	#endif	

jul 21, 20 15:20	ResurrectState.cpp	Page 1/3
1	#include "ResurrectState.h"	
2		
3	ResurrectState::ResurrectState() :	
4	CharacterState(CharacterStateID::Resurrect){}	
5		
6	ResurrectState::~ResurrectState() {}	
7		
8	InputInfo ResurrectState::moveUp(Character& character){	
9	InputInfo info;	
10	info.input = InputID::nothing;	
11	Point aux(0.0,0.0);	
12	info.position = aux;	
13	return info;	
14	}	
15		
16	InputInfo ResurrectState::moveDown(Character& character) {	
17	InputInfo info;	
18	info.input = InputID::nothing;	
19	Point aux(0.0,0.0);	
20	info.position = aux;	
21	return info;	
22	}	
23		
24	InputInfo ResurrectState::moveLeft(Character& character) {	
25	InputInfo info;	
26	info.input = InputID::nothing;	
27	Point aux(0.0,0.0);	
28	info.position = aux;	
29	return info;	
30	}	
31		
32	InputInfo ResurrectState::moveRight(Character& character) {	
33	InputInfo info;	
34	info.input = InputID::nothing;	
35	Point aux(0.0,0.0);	
36	info.position = aux;	
37	return info;	
38	}	
39		
40	InputInfo ResurrectState::stopMove(Character& character) {	
41	InputInfo info;	
42	info.input = InputID::nothing;	
43	Point aux(0.0,0.0);	
44	info.position = aux;	
45	return info;	
46	}	
47		
48	InputInfo ResurrectState::selectItem(Character& character, int item) {	
49	InputInfo info;	
50	Point aux(0.0,0.0);	
51	info.position = aux;	
52	info.input = InputID::nothing;	
53	return info;	
54	}	
55		
56	InputInfo ResurrectState::selectTarget(Character& character, Point position) {	
57	InputInfo info;	
58	info.position = position;	
59	info.input = InputID::nothing;	
60	return info;	
61	}	
62		
63	InputInfo ResurrectState::meditate(Character& character) {	
64	InputInfo info;	
65	Point aux(0.0,0.0);	
66	info.position = aux;	

jul 21, 20 15:20	ResurrectState.cpp	Page 2/3
67	info.input = InputID::nothing;	
68	return info;	
69	}	
70		
71	InputInfo ResurrectState::resurrect(Character& character) {	
72	InputInfo info;	
73	Point aux(0.0,0.0);	
74	info.position = aux;	
75	info.input = InputID::nothing;	
76	return info;	
77	}	
78		
79	InputInfo ResurrectState::cure(Character& character) {	
80	InputInfo info;	
81	Point aux(0.0,0.0);	
82	info.position = aux;	
83	info.input = InputID::nothing;	
84	return info;	
85	}	
86		
87	InputInfo ResurrectState::takeItem(Character& character) {	
88	InputInfo info;	
89	Point aux(0.0,0.0);	
90	info.position = aux;	
91	info.input = InputID::nothing;	
92	return info;	
93	}	
94		
95	InputInfo ResurrectState::dropItem(Character& character, int item) {	
96	InputInfo info;	
97	Point aux(0.0,0.0);	
98	info.position = aux;	
99	info.input = InputID::nothing;	
100	return info;	
101	}	
102		
103	InputInfo ResurrectState::buyItem(Character& character,int item) {	
104	InputInfo info;	
105	Point aux(0.0,0.0);	
106	info.position = aux;	
107	info.input = InputID::nothing;	
108	return info;	
109	}	
110		
111	InputInfo ResurrectState::sellItem(Character& character,int item) {	
112	InputInfo info;	
113	Point aux(0.0,0.0);	
114	info.position = aux;	
115	info.input = InputID::nothing;	
116	return info;	
117	}	
118		
119	InputInfo ResurrectState::retire(Character& character,int item, bool isItem) {	
120	InputInfo info;	
121	Point aux(0.0,0.0);	
122	info.position = aux;	
123	info.input = InputID::nothing;	
124	return info;	
125	}	
126		
127	InputInfo ResurrectState::deposit(Character& character,int item, bool isItem) {	
128	InputInfo info;	
129	Point aux(0.0,0.0);	
130	info.position = aux;	
131	info.input = InputID::nothing;	
132	return info;	

jul 21, 20 15:20	ResurrectState.cpp	Page 3/3
133	}	
134		
135	InputInfo ResurrectState::unequipItem(Character& character, int item) {	
136	InputInfo info;	
137	Point aux(0.0,0.0);	
138	info.position = aux;	
139	info.input = InputID::nothing;	
140	return info;	
141	}	
142		

jul 21, 20 15:20	MoveState.h	Page 1/1
1	#ifndef MOVESTATE_H	
2	#define MOVESTATE_H	
3		
4	#include "CharacterState.h"	
5		
6	class MoveState: public CharacterState {	
7	public:	
8	MoveState();	
9	~MoveState();	
10		
11	virtual InputInfo moveUp(Character& character);	
12	virtual InputInfo moveDown(Character& character);	
13	virtual InputInfo moveLeft(Character& character);	
14	virtual InputInfo moveRight(Character& character);	
15	virtual InputInfo stopMove(Character& character);	
16		
17	virtual InputInfo selectItem(Character& character, int item);	
18	virtual InputInfo dropItem(Character& character, int item);	
19	virtual InputInfo unequipItem(Character& character, int item);	
20	virtual InputInfo deposit(Character& character, int item, bool isItem);	
21	virtual InputInfo retire(Character& character, int item, bool isItem);	
22		
23	virtual InputInfo buyItem(Character& character, int item);	
24	virtual InputInfo sellItem(Character& character, int item);	
25		
26	virtual InputInfo meditate(Character& character);	
27	virtual InputInfo resurrect(Character& character);	
28	virtual InputInfo cure(Character& character);	
29	virtual InputInfo takeItem(Character& character);	
30	virtual InputInfo selectTarget(Character& character, Point position);	
31	};	
32		
33	#endif	

jul 21, 20 15:20	MoveState.cpp	Page 1/3
1	#include "MoveState.h"	
2		
3	MoveState::MoveState() :	
4	CharacterState(CharacterStateID::Move){}	
5		
6	MoveState::~MoveState() = default;	
7		
8	InputInfo MoveState::moveUp(Character& character){	
9	InputInfo info;	
10	info.input = InputID::up;	
11	if (character.getDirection() == Direction::up)	
12	info.input = InputID::nothing;	
13	Point aux(0.0,0.0);	
14	info.position = aux;	
15	return info;	
16	}	
17		
18	InputInfo MoveState::moveDown(Character& character) {	
19	InputInfo info;	
20	info.input = InputID::down;	
21	if (character.getDirection() == Direction::down)	
22	info.input = InputID::nothing;	
23	Point aux(0.0,0.0);	
24	info.position = aux;	
25	return info;	
26	}	
27		
28	InputInfo MoveState::moveLeft(Character& character) {	
29	InputInfo info;	
30	info.input = InputID::left;	
31	if (character.getDirection() == Direction::left)	
32	info.input = InputID::nothing;	
33	Point aux(0.0,0.0);	
34	info.position = aux;	
35	return info;	
36	}	
37		
38	InputInfo MoveState::moveRight(Character& character) {	
39	InputInfo info;	
40	info.input = InputID::right;	
41	if (character.getDirection() == Direction::right)	
42	info.input = InputID::nothing;	
43	Point aux(0.0,0.0);	
44	info.position = aux;	
45	return info;	
46	}	
47		
48	InputInfo MoveState::stopMove(Character& character) {	
49	InputInfo info;	
50	info.input = InputID::stopMove;	
51	Point aux(0.0,0.0);	
52	info.position = aux;	
53	return info;	
54	}	
55		
56	InputInfo MoveState::selectItem(Character& character, int item) {	
57	InputInfo info;	
58	Point aux(0.0,0.0);	
59	info.position = aux;	
60	info.input = InputID::nothing;	
61	return info;	
62	}	
63		
64	InputInfo MoveState::selectTarget(Character& character, Point position) {	
65	InputInfo info;	
66	info.position = position;	

jul 21, 20 15:20	MoveState.cpp	Page 2/3
67	info.input = InputID::selectTarget;	
68	return info;	
69	}	
70		
71	InputInfo MoveState::meditate(Character& character) {	
72	InputInfo info;	
73	Point aux(0.0,0.0);	
74	info.position = aux;	
75	info.input = InputID::nothing;	
76	return info;	
77	}	
78		
79	InputInfo MoveState::resurrect(Character& character) {	
80	InputInfo info;	
81	Point aux(0.0,0.0);	
82	info.position = aux;	
83	info.input = InputID::nothing;	
84	return info;	
85	}	
86		
87	InputInfo MoveState::cure(Character& character) {	
88	InputInfo info;	
89	Point aux(0.0,0.0);	
90	info.position = aux;	
91	info.input = InputID::nothing;	
92	return info;	
93	}	
94		
95	InputInfo MoveState::takeItem(Character& character) {	
96	InputInfo info;	
97	Point aux(0.0,0.0);	
98	info.position = aux;	
99	info.input = InputID::nothing;	
100	return info;	
101	}	
102		
103	InputInfo MoveState::dropItem(Character& character, int item) {	
104	InputInfo info;	
105	Point aux(0.0,0.0);	
106	info.position = aux;	
107	info.input = InputID::dropItem;	
108	info.additional = item;	
109	return info;	
110	}	
111		
112	InputInfo MoveState::buyItem(Character& character, int item) {	
113	InputInfo info;	
114	Point aux(0.0,0.0);	
115	info.position = aux;	
116	info.input = InputID::nothing;	
117	return info;	
118	}	
119		
120	InputInfo MoveState::sellItem(Character& character, int item) {	
121	InputInfo info;	
122	Point aux(0.0,0.0);	
123	info.position = aux;	
124	info.input = InputID::nothing;	
125	return info;	
126	}	
127		
128	InputInfo MoveState::retire(Character& character, int item, bool isItem) {	
129	InputInfo info;	
130	Point aux(0.0,0.0);	
131	info.position = aux;	
132	info.input = InputID::nothing;	

jul 21, 20 15:20	MoveState.cpp	Page 3/3
133	return info;	
134	}	
135		
136	InputInfo MoveState::deposit(Character& character,int item, bool isItem) {	
137	InputInfo info;	
138	Point aux(0.0,0.0);	
139	info.position = aux;	
140	info.input = InputID::nothing;	
141	return info;	
142	}	
143		
144	InputInfo MoveState::unequipItem(Character& character,int item) {	
145	InputInfo info;	
146	Point aux(0.0,0.0);	
147	info.position = aux;	
148	info.input = InputID::nothing;	
149	return info;	
150	}	

jul 21, 20 15:20	MeditateState.h	Page 1/1
1	#ifndef MEDITATESTATE_H	
2	#define MEDITATESTATE_H	
3		
4	#include "CharacterState.h"	
5		
6	class MeditateState: public CharacterState {	
7	public:	
8	MeditateState();	
9	~MeditateState();	
10		
11	virtual InputInfo moveUp(Character& character);	
12	virtual InputInfo moveDown(Character& character);	
13	virtual InputInfo moveLeft(Character& character);	
14	virtual InputInfo moveRight(Character& character);	
15	virtual InputInfo stopMove(Character& character);	
16		
17	virtual InputInfo selectItem(Character& character, int item);	
18	virtual InputInfo dropItem(Character& character, int item);	
19	virtual InputInfo unequipItem(Character& character, int item);	
20	virtual InputInfo buyItem(Character& character, int item);	
21	virtual InputInfo sellItem(Character& character, int item);	
22		
23	virtual InputInfo deposit(Character& character,int item, bool isItem);	
24	virtual InputInfo retire(Character& character,int item, bool isItem);	
25		
26	virtual InputInfo meditate(Character& character);	
27	virtual InputInfo resurrect(Character& character);	
28	virtual InputInfo cure(Character& character);	
29	virtual InputInfo takeItem(Character& character);	
30	virtual InputInfo selectTarget(Character& character, Point position);	
31	};	
32		
33	#endif	

jul 21, 20 15:20	MeditateState.cpp	Page 1/3
1	<code>#include "MeditateState.h"</code>	
2		
3	<code>MeditateState::MeditateState() :</code>	
4	<code>CharacterState(CharacterStateID::Meditate){}</code>	
5		
6	<code>MeditateState::~MeditateState() {}</code>	
7		
8	<code>InputInfo MeditateState::moveUp(Character& character){</code>	
9	<code>InputInfo info;</code>	
10	<code>info.input = InputID::up;</code>	
11	<code>Point aux(0.0,0.0);</code>	
12	<code>info.position = aux;</code>	
13	<code>return info;</code>	
14	<code>}</code>	
15		
16	<code>InputInfo MeditateState::moveDown(Character& character) {</code>	
17	<code>InputInfo info;</code>	
18	<code>info.input = InputID::down;</code>	
19	<code>Point aux(0.0,0.0);</code>	
20	<code>info.position = aux;</code>	
21	<code>return info;</code>	
22	<code>}</code>	
23		
24	<code>InputInfo MeditateState::moveLeft(Character& character) {</code>	
25	<code>InputInfo info;</code>	
26	<code>info.input = InputID::left;</code>	
27	<code>Point aux(0.0,0.0);</code>	
28	<code>info.position = aux;</code>	
29	<code>return info;</code>	
30	<code>}</code>	
31		
32	<code>InputInfo MeditateState::moveRight(Character& character) {</code>	
33	<code>InputInfo info;</code>	
34	<code>info.input = InputID::right;</code>	
35	<code>Point aux(0.0,0.0);</code>	
36	<code>info.position = aux;</code>	
37	<code>return info;</code>	
38	<code>}</code>	
39		
40	<code>InputInfo MeditateState::stopMove(Character& character) {</code>	
41	<code>InputInfo info;</code>	
42	<code>info.input = InputID::stopMove;</code>	
43	<code>Point aux(0.0,0.0);</code>	
44	<code>info.position = aux;</code>	
45	<code>return info;</code>	
46	<code>}</code>	
47		
48	<code>InputInfo MeditateState::selectItem(Character& character, int item) {</code>	
49	<code>InputInfo info;</code>	
50	<code>Point aux(0.0,0.0);</code>	
51	<code>info.position = aux;</code>	
52	<code>info.input = InputID::equipItem;</code>	
53	<code>info.additional = item;</code>	
54	<code>return info;</code>	
55	<code>}</code>	
56		
57	<code>InputInfo MeditateState::selectTarget(Character& character, Point position) {</code>	
58	<code>InputInfo info;</code>	
59	<code>info.position = position;</code>	
60	<code>info.input = InputID::selectTarget;</code>	
61	<code>return info;</code>	
62	<code>}</code>	
63		
64	<code>InputInfo MeditateState::meditate(Character& character) {</code>	
65	<code>InputInfo info;</code>	
66	<code>Point aux(0.0,0.0);</code>	

jul 21, 20 15:20	MeditateState.cpp	Page 2/3
67	<code>info.position = aux;</code>	
68	<code>info.input = InputID::meditate;</code>	
69	<code>return info;</code>	
70	<code>}</code>	
71		
72	<code>InputInfo MeditateState::resurrect(Character& character) {</code>	
73	<code>InputInfo info;</code>	
74	<code>Point aux(0.0,0.0);</code>	
75	<code>info.position = aux;</code>	
76	<code>info.input = InputID::nothing;</code>	
77	<code>return info;</code>	
78	<code>}</code>	
79		
80	<code>InputInfo MeditateState::cure(Character& character) {</code>	
81	<code>InputInfo info;</code>	
82	<code>Point aux(0.0,0.0);</code>	
83	<code>info.position = aux;</code>	
84	<code>info.input = InputID::nothing;</code>	
85	<code>return info;</code>	
86	<code>}</code>	
87		
88	<code>InputInfo MeditateState::takeItem(Character& character) {</code>	
89	<code>InputInfo info;</code>	
90	<code>Point aux(0.0,0.0);</code>	
91	<code>info.position = aux;</code>	
92	<code>info.input = InputID::nothing;</code>	
93	<code>return info;</code>	
94	<code>}</code>	
95		
96	<code>InputInfo MeditateState::dropItem(Character& character, int item) {</code>	
97	<code>InputInfo info;</code>	
98	<code>Point aux(0.0,0.0);</code>	
99	<code>info.position = aux;</code>	
100	<code>info.input = InputID::dropItem;</code>	
101	<code>info.additional = item;</code>	
102	<code>return info;</code>	
103	<code>}</code>	
104		
105	<code>InputInfo MeditateState::buyItem(Character& character,int item) {</code>	
106	<code>InputInfo info;</code>	
107	<code>Point aux(0.0,0.0);</code>	
108	<code>info.position = aux;</code>	
109	<code>info.input = InputID::nothing;</code>	
110	<code>return info;</code>	
111	<code>}</code>	
112		
113	<code>InputInfo MeditateState::sellItem(Character& character,int item) {</code>	
114	<code>InputInfo info;</code>	
115	<code>Point aux(0.0,0.0);</code>	
116	<code>info.position = aux;</code>	
117	<code>info.input = InputID::nothing;</code>	
118	<code>return info;</code>	
119	<code>}</code>	
120		
121	<code>InputInfo MeditateState::retire(Character& character,int item, bool isItem) {</code>	
122	<code>InputInfo info;</code>	
123	<code>Point aux(0.0,0.0);</code>	
124	<code>info.position = aux;</code>	
125	<code>info.input = InputID::nothing;</code>	
126	<code>return info;</code>	
127	<code>}</code>	
128		
129	<code>InputInfo MeditateState::deposit(Character& character,int item, bool isItem) {</code>	
130	<code>InputInfo info;</code>	
131	<code>Point aux(0.0,0.0);</code>	
132	<code>info.position = aux;</code>	

jul 21, 20 15:20	MeditateState.cpp	Page 3/3
133	info.input = InputID::nothing;	
134	return info;	
135	}	
136		
137	InputInfo MeditateState::unequipItem(Character& character, int item) {	
138	InputInfo info;	
139	Point aux(0.0,0.0);	
140	info.position = aux;	
141	info.input = InputID::unequipItem;	
142	info.additional = item;	
143	return info;	
144	}	
145		

jul 21, 20 15:20	InteractState.h	Page 1/1
1	#ifndef INTERACTSTATE_H	
2	#define INTERACTSTATE_H	
3		
4	#include "CharacterState.h"	
5		
6	class InteractState: public CharacterState {	
7	private:	
8	InputID beforeInput;	
9	public:	
10	InteractState();	
11	~InteractState();	
12		
13	virtual InputInfo moveUp(Character& character);	
14	virtual InputInfo moveDown(Character& character);	
15	virtual InputInfo moveLeft(Character& character);	
16	virtual InputInfo moveRight(Character& character);	
17	virtual InputInfo stopMove(Character& character);	
18		
19	virtual InputInfo selectItem(Character& character, int item);	
20	virtual InputInfo dropItem(Character& character, int item);	
21	virtual InputInfo unequipItem(Character& character, int item);	
22	virtual InputInfo buyItem(Character& character, int item);	
23	virtual InputInfo sellItem(Character& character, int item);	
24		
25	virtual InputInfo deposit(Character& character,int item, bool isItem);	
26	virtual InputInfo retire(Character& character,int item, bool isItem);	
27		
28	virtual InputInfo meditate(Character& character);	
29	virtual InputInfo resurrect(Character& character);	
30	virtual InputInfo cure(Character& character);	
31	virtual InputInfo takeItem(Character& character);	
32	virtual InputInfo selectTarget(Character& character, Point position);	
33	};	
34		
35	#endif	

jul 21, 20 15:20	InteractState.cpp	Page 1/3
------------------	--------------------------	----------

```

1  #include "InteractState.h"
2
3  InteractState::InteractState() :
4      CharacterState(CharacterStateID::Interact), beforeInput(InputID::nothing){}
5
6  InteractState::~InteractState() {}
7
8  InputInfo InteractState::moveUp(Character& character){
9      InputInfo info;
10     info.input = InputID::up;
11     if (info.input == beforeInput) {
12         info.input = InputID::nothing;
13     }
14     Point aux(0.0,0.0);
15     info.position = aux;
16     beforeInput = info.input;
17     return info;
18 }
19
20 InputInfo InteractState::moveDown(Character& character) {
21     InputInfo info;
22     info.input = InputID::down;
23     if (info.input == beforeInput) {
24         info.input = InputID::nothing;
25     }
26     Point aux(0.0,0.0);
27     info.position = aux;
28     beforeInput = info.input;
29     return info;
30 }
31
32 InputInfo InteractState::moveLeft(Character& character) {
33     InputInfo info;
34     info.input = InputID::left;
35     if (info.input == beforeInput) {
36         info.input = InputID::nothing;
37     }
38     Point aux(0.0,0.0);
39     info.position = aux;
40     beforeInput = info.input;
41     return info;
42 }
43
44 InputInfo InteractState::moveRight(Character& character) {
45     InputInfo info;
46     info.input = InputID::right;
47     if (info.input == beforeInput) {
48         info.input = InputID::nothing;
49     }
50     Point aux(0.0,0.0);
51     info.position = aux;
52     beforeInput = info.input;
53     return info;
54 }
55
56 InputInfo InteractState::stopMove(Character& character) {
57     InputInfo info;
58     info.input = InputID::stopMove;
59     Point aux(0.0,0.0);
60     info.position = aux;
61     beforeInput = info.input;
62     return info;
63 }
64
65 InputInfo InteractState::selectItem(Character& character, int item) {
66     InputInfo info;

```

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	InteractState.cpp	Page 2/3
------------------	--------------------------	----------

```

67     Point aux(0.0,0.0);
68     info.position = aux;
69     info.input = InputID::nothing;
70     beforeInput = info.input;
71     return info;
72 }
73
74 InputInfo InteractState::selectTarget(Character& character, Point position) {
75     InputInfo info;
76     info.position = position;
77     info.input = InputID::selectTarget;
78     beforeInput = info.input;
79     return info;
80 }
81
82 InputInfo InteractState::meditate(Character& character) {
83     InputInfo info;
84     Point aux(0.0,0.0);
85     info.position = aux;
86     info.input = InputID::nothing;
87     beforeInput = info.input;
88     return info;
89 }
90
91 InputInfo InteractState::resurrect(Character& character) {
92     InputInfo info;
93     Point aux(0.0,0.0);
94     info.position = aux;
95     info.input = InputID::resurrect;
96     beforeInput = info.input;
97     return info;
98 }
99
100 InputInfo InteractState::cure(Character& character) {
101     InputInfo info;
102     Point aux(0.0,0.0);
103     info.position = aux;
104     info.input = InputID::cure;
105     beforeInput = info.input;
106     return info;
107 }
108
109 InputInfo InteractState::takeItem(Character& character) {
110     InputInfo info;
111     Point aux(0.0,0.0);
112     info.position = aux;
113     info.input = InputID::nothing;
114     beforeInput = info.input;
115     return info;
116 }
117
118 InputInfo InteractState::dropItem(Character& character, int item) {
119     InputInfo info;
120     Point aux(0.0,0.0);
121     info.position = aux;
122     info.input = InputID::nothing;
123     beforeInput = info.input;
124     return info;
125 }
126
127 InputInfo InteractState::buyItem(Character& character, int item) {
128     InputInfo info;
129     Point aux(0.0,0.0);
130     info.position = aux;
131     info.input = InputID::buy;
132     info.additional = item;

```

210/217

jul 21, 20 15:20	InteractState.cpp	Page 3/3
133	beforeInput = info.input;	
134	return info;	
135	}	
136		
137	InputInfo InteractState::sellItem(Character& character,int item) {	
138	InputInfo info;	
139	Point aux(0.0,0.0);	
140	info.position = aux;	
141	info.input =InputID::sell;	
142	info.additional = item;	
143	beforeInput = info.input;	
144	return info;	
145	}	
146		
147	InputInfo InteractState::deposit(Character& character,int item, bool isItem) {	
148	InputInfo info;	
149	Point aux(0.0,0.0);	
150	info.position = aux;	
151	if (isItem) {	
152	info.input =InputID::depositItem;	
153	} else {	
154	info.input =InputID::depositGold;	
155	}	
156	info.additional = item;	
157	beforeInput = info.input;	
158	return info;	
159	}	
160		
161	InputInfo InteractState::retire(Character& character,int item, bool isItem) {	
162	InputInfo info;	
163	Point aux(0.0,0.0);	
164	info.position = aux;	
165	if (isItem) {	
166	info.input =InputID::retireItem;	
167	} else {	
168	info.input =InputID::retireGold;	
169	}	
170	info.additional = item;	
171	beforeInput = info.input;	
172	return info;	
173	}	
174		
175	InputInfo InteractState::unequipItem(Character& character,int item) {	
176	InputInfo info;	
177	Point aux(0.0,0.0);	
178	info.position = aux;	
179	info.input = InputID::nothing;	
180	beforeInput = info.input;	
181	return info;	
182	}	
183		

jul 21, 20 15:20	CharacterState.h	Page 1/1
1	#ifndef CHARACTERSTATE_H	
2	#define CHARACTERSTATE_H	
3		
4	#include "../common/Identificators.h"	
5	#include "../Character.h"	
6		
7	<i>//Esta clase es la utilizada por los personajes del juego para retratar su</i>	
8	<i>//estado. Las implementaciones de esta clase abstracta</i>	
9	<i>//permitir�n o no el efecto de ciertos comandos.</i>	
10		
11	class CharacterState {	
12	private:	
13	CharacterStateID id;	
14	public:	
15	explicit CharacterState(CharacterStateID state) : id(state) {}	
16	virtual ~CharacterState() = default;	
17	virtual CharacterStateID& getState() {	
18	return this->id;	
19	}	
20		
21	virtual InputInfo moveUp(Character& character) = 0;	
22	virtual InputInfo moveDown(Character& character) = 0;	
23	virtual InputInfo moveLeft(Character& character) = 0;	
24	virtual InputInfo moveRight(Character& character) = 0;	
25	virtual InputInfo stopMove(Character& character) = 0;	
26		
27	virtual InputInfo selectItem(Character& character, int item) = 0;	
28	virtual InputInfo unequipItem(Character& character, int item) = 0;	
29	virtual InputInfo dropItem(Character& character, int item) = 0;	
30	virtual InputInfo takeItem(Character& character) = 0;	
31	virtual InputInfo resurrect(Character& character) = 0;	
32	virtual InputInfo meditate(Character& character) = 0;	
33	virtual InputInfo cure(Character& character) = 0;	
34	virtual InputInfo buyItem(Character& character, int item) = 0;	
35	virtual InputInfo sellItem(Character& character, int item) = 0;	
36	virtual InputInfo deposit(Character& character,int item, bool isItem) = 0;	
37	virtual InputInfo retire(Character& character,int item, bool isItem) = 0;	
38	virtual InputInfo selectTarget(Character& character, Point position) = 0;	
39		
40	};	
41		
42	#endif	

jul 21, 20 15:20	AttackState.h	Page 1/1
1	<code>#ifndef ATTACKSTATE_H</code>	
2	<code>#define ATTACKSTATE_H</code>	
3		
4	<code>#include "CharacterState.h"</code>	
5		
6	<code>class AttackState: public CharacterState {</code>	
7	<code>private:</code>	
8	<code> InputID beforeInput;</code>	
9	<code>public:</code>	
10	<code> AttackState();</code>	
11	<code> ~AttackState();</code>	
12		
13	<code> virtual InputInfo moveUp(Character& character);</code>	
14	<code> virtual InputInfo moveDown(Character& character);</code>	
15	<code> virtual InputInfo moveLeft(Character& character);</code>	
16	<code> virtual InputInfo moveRight(Character& character);</code>	
17	<code> virtual InputInfo stopMove(Character& character);</code>	
18		
19	<code> virtual InputInfo selectItem(Character& character, int item);</code>	
20	<code> virtual InputInfo dropItem(Character& character, int item);</code>	
21	<code> virtual InputInfo unequipItem(Character& character, int item);</code>	
22	<code> virtual InputInfo buyItem(Character& character, int item);</code>	
23	<code> virtual InputInfo sellItem(Character& character, int item);</code>	
24		
25	<code> virtual InputInfo deposit(Character& character,int item, bool isItem);</code>	
26	<code> virtual InputInfo retire(Character& character,int item, bool isItem);</code>	
27		
28	<code> virtual InputInfo meditate(Character& character);</code>	
29	<code> virtual InputInfo resurrect(Character& character);</code>	
30	<code> virtual InputInfo cure(Character& character);</code>	
31	<code> virtual InputInfo takeItem(Character& character);</code>	
32	<code> virtual InputInfo selectTarget(Character& character, Point position);</code>	
33	<code>};</code>	
34		
35	<code>#endif</code>	

jul 21, 20 15:20	AttackState.cpp	Page 1/3
1	<code>#include "AttackState.h"</code>	
2		
3	<code>AttackState::AttackState() :</code>	
4	<code> CharacterState(CharacterStateID::Attack) , beforeInput(InputID::nothing) {}</code>	
5		
6	<code>AttackState::~AttackState() = default;</code>	
7		
8	<code>InputInfo AttackState::moveUp(Character& character){</code>	
9	<code> InputInfo info;</code>	
10	<code> info.input = InputID::up;</code>	
11	<code> if (character.getDirection() == Direction::up)</code>	
12	<code> info.input = InputID::nothing;</code>	
13	<code> Point aux(0.0,0.0);</code>	
14	<code> info.position = aux;</code>	
15	<code> beforeInput = info.input;</code>	
16	<code> return info;</code>	
17	<code> }</code>	
18		
19	<code>InputInfo AttackState::moveDown(Character& character) {</code>	
20	<code> InputInfo info;</code>	
21	<code> info.input = InputID::down;</code>	
22	<code> if (character.getDirection() == Direction::down)</code>	
23	<code> info.input = InputID::nothing;</code>	
24	<code> Point aux(0.0,0.0);</code>	
25	<code> info.position = aux;</code>	
26	<code> beforeInput = info.input;</code>	
27	<code> return info;</code>	
28	<code> }</code>	
29		
30	<code>InputInfo AttackState::moveLeft(Character& character) {</code>	
31	<code> InputInfo info;</code>	
32	<code> info.input = InputID::left;</code>	
33	<code> if (character.getDirection() == Direction::left)</code>	
34	<code> info.input = InputID::nothing;</code>	
35	<code> Point aux(0.0,0.0);</code>	
36	<code> info.position = aux;</code>	
37	<code> beforeInput = info.input;</code>	
38	<code> return info;</code>	
39	<code> }</code>	
40		
41	<code>InputInfo AttackState::moveRight(Character& character) {</code>	
42	<code> InputInfo info;</code>	
43	<code> info.input = InputID::right;</code>	
44	<code> if (character.getDirection() == Direction::right)</code>	
45	<code> info.input = InputID::nothing;</code>	
46	<code> Point aux(0.0,0.0);</code>	
47	<code> info.position = aux;</code>	
48	<code> beforeInput = info.input;</code>	
49	<code> return info;</code>	
50	<code> }</code>	
51		
52	<code>InputInfo AttackState::stopMove(Character& character) {</code>	
53	<code> InputInfo info;</code>	
54	<code> info.input = InputID::stopMove;</code>	
55	<code> Point aux(0.0,0.0);</code>	
56	<code> info.position = aux;</code>	
57	<code> beforeInput = info.input;</code>	
58	<code> return info;</code>	
59	<code> }</code>	
60		
61	<code>InputInfo AttackState::selectItem(Character& character, int item) {</code>	
62	<code> InputInfo info;</code>	
63	<code> Point aux(0.0,0.0);</code>	
64	<code> info.position = aux;</code>	
65	<code> info.input = InputID::nothing;</code>	
66	<code> beforeInput = info.input;</code>	

jul 21, 20 15:20	AttackState.cpp	Page 2/3
67	return info;	
68	}	
69		
70	InputInfo AttackState::selectTarget(Character& character, Point position) {	
71	InputInfo info;	
72	info.position = position;	
73	info.input = InputID::selectTarget;	
74	if (beforeInput == info.input)	
75	info.input = InputID::nothing;	
76	beforeInput = info.input;	
77	return info;	
78	}	
79		
80	InputInfo AttackState::meditate(Character& character) {	
81	InputInfo info;	
82	Point aux(0.0,0.0);	
83	info.position = aux;	
84	info.input =InputID::nothing;	
85	beforeInput = info.input;	
86	return info;	
87	}	
88		
89	InputInfo AttackState::resurrect(Character& character) {	
90	InputInfo info;	
91	Point aux(0.0,0.0);	
92	info.position = aux;	
93	info.input =InputID::nothing;	
94	beforeInput = info.input;	
95	return info;	
96	}	
97		
98	InputInfo AttackState::cure(Character& character) {	
99	InputInfo info;	
100	Point aux(0.0,0.0);	
101	info.position = aux;	
102	info.input =InputID::nothing;	
103	beforeInput = info.input;	
104	return info;	
105	}	
106		
107	InputInfo AttackState::takeItem(Character& character) {	
108	InputInfo info;	
109	Point aux(0.0,0.0);	
110	info.position = aux;	
111	info.input =InputID::nothing;	
112	beforeInput = info.input;	
113	return info;	
114	}	
115		
116	InputInfo AttackState::dropItem(Character& character, int item) {	
117	InputInfo info;	
118	Point aux(0.0,0.0);	
119	info.position = aux;	
120	info.input =InputID::nothing;	
121	beforeInput = info.input;	
122	return info;	
123	}	
124		
125	InputInfo AttackState::buyItem(Character& character,int item) {	
126	InputInfo info;	
127	Point aux(0.0,0.0);	
128	info.position = aux;	
129	info.input =InputID::nothing;	
130	beforeInput = info.input;	
131	return info;	
132	}	

jul 21, 20 15:20	AttackState.cpp	Page 3/3
133		
134	InputInfo AttackState::sellItem(Character& character,int item) {	
135	InputInfo info;	
136	Point aux(0.0,0.0);	
137	info.position = aux;	
138	info.input =InputID::nothing;	
139	beforeInput = info.input;	
140	return info;	
141	}	
142		
143	InputInfo AttackState::retire(Character& character,int item, bool isItem) {	
144	InputInfo info;	
145	Point aux(0.0,0.0);	
146	info.position = aux;	
147	info.input = InputID::nothing;	
148	beforeInput = info.input;	
149	return info;	
150	}	
151		
152	InputInfo AttackState::deposit(Character& character,int item, bool isItem) {	
153	InputInfo info;	
154	Point aux(0.0,0.0);	
155	info.position = aux;	
156	info.input = InputID::nothing;	
157	beforeInput = info.input;	
158	return info;	
159	}	
160		
161	InputInfo AttackState::unequipItem(Character& character,int item) {	
162	InputInfo info;	
163	Point aux(0.0,0.0);	
164	info.position = aux;	
165	info.input = InputID::nothing;	
166	beforeInput = info.input;	
167	return info;	
168	}	

jul 21, 20 15:20	Camera.h	Page 1/1
1	<code>#ifndef CAMERA_H</code>	
2	<code>#define CAMERA_H</code>	
3		
4	<code>#include <SDL2/SDL.h></code>	
5	<code>#include "Window.h"</code>	
6	<code>#include "../common/Point.h"</code>	
7		
8	<code>//Clase destinada a mantener centrada la camara del juego sobre</code>	
9	<code>//el jugador que es seteado como target.</code>	
10		
11	<code>class Camera {</code>	
12	<code>private:</code>	
13	<code>Window& window;</code>	
14	<code>float width, height; //Limites del mapa</code>	
15	<code>float scale;</code>	
16	<code>Point* playerTarget = nullptr; //Jugador en el cual se debe centrar la camar</code>	
17	<code>a.</code>	
18	<code>Point positionScreen; //</code>	
19	<code>SDL_Rect cam; //Dimensiones de la cámara.</code>	
20	<code>//Ajusta la posición a destiny para que nunca se pase de los límites del map</code>	
21	<code>a.</code>	
22	<code>void limits(Point* destiny);</code>	
23	<code>public:</code>	
24	<code>Camera(Window& window, float widthMap, float heightMap);</code>	
25	<code>SDL_Rect getCamera() const;</code>	
26	<code>float getCameraWidth() const;</code>	
27	<code>float getCameraHeight() const;</code>	
28	<code>Point getCameraPosition() const;</code>	
29	<code>float getScale() const;</code>	
30	<code>void setPlayer(Point* player);</code>	
31	<code>void render(Point destiny);</code>	
32		
33	<code>//Devuelve verdadero en caso de que el click haya sido realizado</code>	
34	<code>//dentro del Viewport de la camara.</code>	
35	<code>bool clickInMap(Point coordinates) const;</code>	
36		
37	<code>//Devuelve un Point de acuerdo a las coordenadas globales,</code>	
38	<code>//sacandole el relativo de la camara.</code>	
39	<code>Point calculateGlobalPosition(Point coordinates) const;</code>	
40		
41	<code>//Devuelve la distancia que hay entre el Point coordinates y</code>	
42	<code>//el target que tiene ajustado la cámara.</code>	
43	<code>int distanceFromTarget(Point coordinates) const;</code>	
44		
45	<code>-Camera();</code>	
46	<code>};</code>	
47	<code>#endif</code>	

mar 21 jul 2020 15:20:01 ART

Padron Grupo 14 (curso 2020.1.1) Ejercicio 4.2 (entrega 2020-07-21T15:19:48)

jul 21, 20 15:20	Camera.cpp	Page 1/2
1	<code>#include "Camera.h"</code>	
2	<code>#include <algorithm></code>	
3	<code>#include <SDL2/SDL.h></code>	
4		
5	<code>#define WIDTHSEGMENT 8</code>	
6	<code>#define TOPBARHEIGHT 60</code>	
7		
8	<code>Camera::Camera(Window& window, float widthMap, float heightMap) : window(window)</code>	
9	<code>{</code>	
10	<code>width(widthMap), height(heightMap), scale(1.0f) {</code>	
11	<code>this->positionScreen = Point(0.0, float(this->window.getHeight())/2.0);</code>	
12	<code>this->cam = {(this->window.getWidth()/WIDTHSEGMENT)*2, TOPBARHEIGHT, (this->win</code>	
13	<code>dow.getWidth()/WIDTHSEGMENT)*6, this->window.getHeight()-TOPBARHEIGHT};</code>	
14	<code>float Camera::getCameraWidth() const {</code>	
15	<code>return float(this->window.getWidth());</code>	
16	<code>}</code>	
17	<code>float Camera::getCameraHeight() const {</code>	
18	<code>return float(this->window.getHeight());</code>	
19	<code>}</code>	
20		
21	<code>SDL_Rect Camera::getCamera() const {</code>	
22	<code>return this->cam;</code>	
23	<code>}</code>	
24		
25	<code>float Camera::getScale() const {</code>	
26	<code>return this->scale;</code>	
27	<code>}</code>	
28		
29	<code>Point Camera::getCameraPosition() const {</code>	
30	<code>return this->positionScreen;</code>	
31	<code>}</code>	
32		
33	<code>void Camera::setPlayer(Point* player){</code>	
34	<code>this->playerTarget = player;</code>	
35	<code>}</code>	
36		
37	<code>void Camera::limits(Point* destiny) {</code>	
38	<code>if (this->playerTarget != nullptr) {</code>	
39	<code>float limitWidth = ((this->window.getWidth()/WIDTHSEGMENT)*6) / 2.0f;</code>	
40	<code>float limitHeight = (this->window.getHeight()-TOPBARHEIGHT) / 2.0f;</code>	
41		
42	<code>destiny->y = std::max(destiny->y, limitHeight);</code>	
43	<code>destiny->y = std::min(destiny->y, this->height-limitHeight);</code>	
44	<code>destiny->x = std::min(destiny->x, this->width-limitWidth);</code>	
45	<code>destiny->x = std::max(destiny->x, limitWidth);</code>	
46	<code>}</code>	
47		
48	<code>}</code>	
49		
50		
51	<code>Point Camera::calculateGlobalPosition(Point coordinates) const {</code>	
52	<code>float x = coordinates.x + (positionScreen.x-cam.x);</code>	
53	<code>float y = coordinates.y + (positionScreen.y-cam.y);</code>	
54	<code>return Point(x,y);</code>	
55	<code>}</code>	
56		
57	<code>bool Camera::clickInMap(Point coordinates) const {</code>	
58	<code>bool inMap = false;</code>	
59	<code>if (coordinates.x > cam.x ^ coordinates.x < cam.x+cam.w ^</code>	
60	<code>coordinates.y > cam.y ^ coordinates.y < cam.y+cam.h)</code>	
61	<code>inMap = true;</code>	
62	<code>return inMap;</code>	
63	<code>}</code>	
64		

214/217

jul 21, 20 15:20	Camera.cpp	Page 2/2
65	void Camera::render(Point destiny) {	
66	limits(&destiny);	
67	this->cam = {(this->window.getWidth()/WIDTHSEGMENT)*2, TOPBARHEIGHT, (this->win	
68	dow.getWidth()/WIDTHSEGMENT)*6,	
69	this->window.getHeight()-TOPBARHEIGHT};	
70	this->positionScreen.x = destiny.x - (((this->window.getWidth()/WIDTHSEGMENT	
71)*6) / 2.0f);	
72	this->positionScreen.y = destiny.y - (((this->window.getHeight() - TOPBARHEIG	
73	HT) / 2.0f);	
74	SDL_Rect display = {(this->window.getWidth()/WIDTHSEGMENT) * 2, TOPBARHEIGHT,	
75	(this->window.getWidth()/WIDTHSEGMENT) * 6, this->window.getHeight()	
76	- TOPBARHEIGHT};	
77	SDL_RenderSetViewport(&(this->window.getRenderer()), &display);	
78	}	
79		
80	int Camera::distanceFromTarget(Point coordinates) const {	
	return this->playerTarget->distance(coordinates);	
	}	
	Camera::~Camera()= default;	

jul 21, 20 15:20	Table of Content	Page 1/5
1	Table of Contents	
2	1 WorldInfoQueue.h... sheets 1 to 1 (1) pages 1- 1 10 lines	
3	2 WorldInfo.h... sheets 1 to 1 (1) pages 2- 2 28 lines	
4	3 WorldInfo.cpp... sheets 2 to 2 (1) pages 3- 3 33 lines	
5	4 World.h... sheets 2 to 3 (2) pages 4- 5 73 lines	
6	5 World.cpp... sheets 3 to 4 (2) pages 6- 8 174 lines	
7	6 ThPlayerReceiver.h... sheets 5 to 5 (1) pages 9- 9 30 lines	
8	7 ThPlayerReceiver.cpp sheets 5 to 5 (1) pages 10- 10 44 lines	
9	8 ThPlayer.h... sheets 6 to 6 (1) pages 11- 11 40 lines	
10	9 ThPlayer.cpp... sheets 6 to 6 (1) pages 12- 12 56 lines	
11	10 ThLobbyPlayer.h... sheets 7 to 7 (1) pages 13- 13 29 lines	
12	11 ThLobbyPlayer.cpp... sheets 7 to 7 (1) pages 14- 14 37 lines	
13	12 TakeAndDropStateCharacter.h sheets 8 to 8 (1) pages 15- 15 28 lines	
14	13 TakeAndDropStateCharacter.cpp sheets 8 to 8 (1) pages 16- 16 60 lines	
15	14 StillStateCreature.h sheets 9 to 9 (1) pages 17- 17 29 lines	
16	15 StillStateCreature.cpp sheets 9 to 9 (1) pages 18- 18 32 lines	
17	16 StillStateCharacter.h sheets 10 to 10 (1) pages 19- 19 28 lines	
18	17 StillStateCharacter.cpp sheets 10 to 10 (1) pages 20- 20 49 lines	
19	18 StateTranslator.h... sheets 11 to 11 (1) pages 21- 21 18 lines	
20	19 StateTranslator.cpp... sheets 11 to 11 (1) pages 22- 22 43 lines	
21	20 StatePool.h... sheets 12 to 12 (1) pages 23- 23 26 lines	
22	21 StatePoolCreature.h... sheets 12 to 12 (1) pages 24- 24 36 lines	
23	22 StatePoolCreature.cpp sheets 13 to 13 (1) pages 25- 26 74 lines	
24	23 StatePool.cpp... sheets 14 to 14 (1) pages 27- 27 6 lines	
25	24 StatePoolCharacter.h sheets 14 to 14 (1) pages 28- 28 36 lines	
26	25 StatePoolCharacter.cpp sheets 15 to 15 (1) pages 29- 30 101 lines	
27	26 State.h... sheets 16 to 16 (1) pages 31- 31 36 lines	
28	27 StateCreature.h... sheets 16 to 16 (1) pages 32- 32 17 lines	
29	28 StateCreature.cpp... sheets 17 to 17 (1) pages 33- 33 8 lines	
30	29 State.cpp... sheets 17 to 17 (1) pages 34- 34 27 lines	
31	30 StateCharacter.h... sheets 18 to 18 (1) pages 35- 35 17 lines	
32	31 StateCharacter.cpp... sheets 18 to 18 (1) pages 36- 36 7 lines	
33	32 ResurrectStateCharacter.h sheets 19 to 19 (1) pages 37- 37 32 lines	
34	33 ResurrectStateCharacter.cpp sheets 19 to 20 (2) pages 38- 39 69 lines	
35	34 PursuitStateCreature.h sheets 20 to 20 (1) pages 40- 40 36 lines	
36	35 PursuitStateCreature.cpp sheets 21 to 21 (1) pages 41- 42 80 lines	
37	36 MoveStateCreature.h... sheets 22 to 22 (1) pages 43- 43 32 lines	
38	37 MoveStateCreature.cpp sheets 22 to 23 (2) pages 44- 45 74 lines	
39	38 MoveStateCharacter.h sheets 23 to 23 (1) pages 46- 46 34 lines	
40	39 MoveStateCharacter.cpp sheets 24 to 24 (1) pages 47- 48 94 lines	
41	40 MeditateStateCharacter.h sheets 25 to 25 (1) pages 49- 49 29 lines	
42	41 MeditateStateCharacter.cpp sheets 25 to 25 (1) pages 50- 50 38 lines	
43	42 InteractStateCharacter.h sheets 26 to 26 (1) pages 51- 51 30 lines	
44	43 InteractStateCharacter.cpp sheets 26 to 27 (2) pages 52- 53 81 lines	
45	44 EquipStateCharacter.h sheets 27 to 27 (1) pages 54- 54 31 lines	
46	45 EquipStateCharacter.cpp sheets 28 to 28 (1) pages 55- 55 48 lines	
47	46 AttackStateCreature.h sheets 28 to 28 (1) pages 56- 56 33 lines	
48	47 AttackStateCreature.cpp sheets 29 to 29 (1) pages 57- 58 73 lines	
49	48 AttackStateCharacter.h sheets 30 to 30 (1) pages 59- 59 33 lines	
50	49 AttackStateCharacter.cpp sheets 30 to 31 (2) pages 60- 61 86 lines	
51	50 server_main.cpp... sheets 31 to 31 (1) pages 62- 62 37 lines	
52	51 ServerItem.h... sheets 32 to 32 (1) pages 63- 63 30 lines	
53	52 ServerItem.cpp... sheets 32 to 32 (1) pages 64- 64 30 lines	
54	53 Profession.h... sheets 33 to 33 (1) pages 65- 65 27 lines	
55	54 Priest.h... sheets 33 to 33 (1) pages 66- 66 34 lines	
56	55 Priest.cpp... sheets 34 to 34 (1) pages 67- 68 73 lines	
57	56 PlayerAcceptor.h... sheets 35 to 35 (1) pages 69- 69 30 lines	
58	57 PlayerAcceptor.cpp... sheets 35 to 35 (1) pages 70- 70 57 lines	
59	58 ObjectItem.h... sheets 36 to 36 (1) pages 71- 71 56 lines	
60	59 ObjectItem.cpp... sheets 36 to 37 (2) pages 72- 73 82 lines	
61	60 NPCServer.h... sheets 37 to 37 (1) pages 74- 74 49 lines	
62	61 NPCServer.cpp... sheets 38 to 38 (1) pages 75- 76 88 lines	
63	62 Node.h... sheets 39 to 39 (1) pages 77- 77 46 lines	
64	63 Node.cpp... sheets 39 to 39 (1) pages 78- 78 55 lines	
65	64 NodeContainer.h... sheets 40 to 40 (1) pages 79- 79 29 lines	
66	65 NodeContainer.cpp... sheets 40 to 40 (1) pages 80- 80 47 lines	

jul 21, 20 15:20	Table of Content	Page 2/5
67	66 Nest.h..... sheets	41 to 41 (1) pages 81- 81 40 lines
68	67 Nest.cpp..... sheets	41 to 41 (1) pages 82- 82 64 lines
69	68 NestContainer.h..... sheets	42 to 42 (1) pages 83- 83 33 lines
70	69 NestContainer.cpp..... sheets	42 to 42 (1) pages 84- 84 64 lines
71	70 Movement.h..... sheets	43 to 43 (1) pages 85- 85 38 lines
72	71 MovementCreature.h..... sheets	43 to 43 (1) pages 86- 86 22 lines
73	72 MovementCreature.cpp..... sheets	44 to 44 (1) pages 87- 87 19 lines
74	73 Movement.cpp..... sheets	44 to 44 (1) pages 88- 88 48 lines
75	74 MovementCharacter.h..... sheets	45 to 45 (1) pages 89- 89 23 lines
76	75 MovementCharacter.cpp..... sheets	45 to 45 (1) pages 90- 90 19 lines
77	76 Merchant.h..... sheets	46 to 46 (1) pages 91- 91 39 lines
78	77 Merchant.cpp..... sheets	46 to 47 (2) pages 92- 93 77 lines
79	78 ItemTranslator.h..... sheets	47 to 47 (1) pages 94- 94 24 lines
80	79 ItemTranslator.cpp..... sheets	48 to 49 (2) pages 95- 98 210 lines
81	80 Inventory.h..... sheets	50 to 50 (1) pages 99- 99 38 lines
82	81 Inventory.cpp..... sheets	50 to 51 (2) pages 100-101 75 lines
83	82 GameStats.h..... sheets	51 to 51 (1) pages 102-102 46 lines
84	83 GameStatsConfig.h..... sheets	52 to 52 (1) pages 103-104 82 lines
85	84 GameStatsConfig.cpp..... sheets	53 to 55 (3) pages 105-109 287 lines
86	85 GameObjectsContainer.h..... sheets	55 to 55 (1) pages 110-110 32 lines
87	86 GameObjectsContainer.cpp..... sheets	56 to 56 (1) pages 111-111 55 lines
88	87 GameObject.h..... sheets	56 to 57 (2) pages 112-113 80 lines
89	88 GameObject.cpp..... sheets	57 to 57 (1) pages 114-114 58 lines
90	89 GameCharacter.h..... sheets	58 to 58 (1) pages 115-116 115 lines
91	90 GameCharacter.cpp..... sheets	59 to 61 (3) pages 117-122 334 lines
92	91 DropItem.h..... sheets	62 to 62 (1) pages 123-123 23 lines
93	92 DropItem.cpp..... sheets	62 to 62 (1) pages 124-124 14 lines
94	93 Creature.h..... sheets	63 to 63 (1) pages 125-125 61 lines
95	94 Creature.cpp..... sheets	63 to 64 (2) pages 126-128 150 lines
96	95 Cell.h..... sheets	65 to 65 (1) pages 129-129 65 lines
97	96 Cell.cpp..... sheets	65 to 66 (2) pages 130-131 90 lines
98	97 Board.h..... sheets	66 to 67 (2) pages 132-133 81 lines
99	98 Board.cpp..... sheets	67 to 69 (3) pages 134-138 298 lines
100	99 Banker.h..... sheets	70 to 70 (1) pages 139-139 45 lines
101	100 Banker.cpp..... sheets	70 to 71 (2) pages 140-142 141 lines
102	101 TileSet.h..... sheets	72 to 72 (1) pages 143-143 28 lines
103	102 TileSet.cpp..... sheets	72 to 72 (1) pages 144-144 27 lines
104	103 TileLayer.h..... sheets	73 to 73 (1) pages 145-145 27 lines
105	104 TileLayer.cpp..... sheets	73 to 73 (1) pages 146-146 28 lines
106	105 TiledMap.h..... sheets	74 to 74 (1) pages 147-147 49 lines
107	106 TiledMap.cpp..... sheets	74 to 75 (2) pages 148-149 90 lines
108	107 Thread.h..... sheets	75 to 75 (1) pages 150-150 26 lines
109	108 Thread.cpp..... sheets	76 to 76 (1) pages 151-151 24 lines
110	109 StaticObject.h..... sheets	76 to 76 (1) pages 152-152 33 lines
111	110 StaticObject.cpp..... sheets	77 to 77 (1) pages 153-153 36 lines
112	111 Socket.h..... sheets	77 to 77 (1) pages 154-154 63 lines
113	112 SocketException.h..... sheets	78 to 78 (1) pages 155-155 19 lines
114	113 SocketException.cpp..... sheets	78 to 78 (1) pages 156-156 24 lines
115	114 Socket.cpp..... sheets	79 to 80 (2) pages 157-159 153 lines
116	115 Random.h..... sheets	80 to 80 (1) pages 160-160 13 lines
117	116 Random.cpp..... sheets	81 to 81 (1) pages 161-161 17 lines
118	117 Point.h..... sheets	81 to 81 (1) pages 162-162 30 lines
119	118 Point.cpp..... sheets	82 to 82 (1) pages 163-163 49 lines
120	119 PlayerInfo.h..... sheets	82 to 82 (1) pages 164-164 59 lines
121	120 PlayerInfo.cpp..... sheets	83 to 83 (1) pages 165-166 63 lines
122	121 ObjectLayer.h..... sheets	84 to 84 (1) pages 167-167 26 lines
123	122 ObjectLayer.cpp..... sheets	84 to 84 (1) pages 168-168 24 lines
124	123 Message.h..... sheets	85 to 85 (1) pages 169-169 37 lines
125	124 Message.cpp..... sheets	85 to 86 (2) pages 170-171 69 lines
126	125 JsonReader.h..... sheets	86 to 86 (1) pages 172-172 19 lines
127	126 JsonReader.cpp..... sheets	87 to 87 (1) pages 173-173 21 lines
128	127 InputQueue.h..... sheets	87 to 87 (1) pages 174-174 10 lines
129	128 Identifiers.h..... sheets	88 to 89 (2) pages 175-178 201 lines
130	129 GameObjectInfo.h..... sheets	90 to 90 (1) pages 179-179 51 lines
131	130 GameObjectInfo.cpp..... sheets	90 to 91 (2) pages 180-181 81 lines
132	131 Exception.h..... sheets	91 to 91 (1) pages 182-182 19 lines

jul 21, 20 15:20	Table of Content	Page 3/5
133	132 Exception.cpp..... sheets	92 to 92 (1) pages 183-183 24 lines
134	133 Decoder.h..... sheets	92 to 92 (1) pages 184-184 58 lines
135	134 Decoder.cpp..... sheets	93 to 96 (4) pages 185-192 459 lines
136	135 DataQueue.h..... sheets	97 to 97 (1) pages 193-193 12 lines
137	136 CommunicationProtocol.h..... sheets	97 to 97 (1) pages 194-194 32 lines
138	137 CommunicationProtocol.cpp..... sheets	98 to 98 (1) pages 195-195 39 lines
139	138 Chrono.h..... sheets	98 to 98 (1) pages 196-196 19 lines
140	139 Chrono.cpp..... sheets	99 to 99 (1) pages 197-197 17 lines
141	140 BlockingQueue.h..... sheets	99 to 100 (2) pages 198-199 84 lines
142	141 Window.h..... sheets	100 to 100 (1) pages 200-200 39 lines
143	142 Window.cpp..... sheets	101 to 102 (2) pages 201-203 135 lines
144	143 UnequipButton.h..... sheets	102 to 102 (1) pages 204-204 25 lines
145	144 UnequipButton.cpp..... sheets	103 to 103 (1) pages 205-205 25 lines
146	145 UI.h..... sheets	103 to 104 (2) pages 206-207 85 lines
147	146 UI.cpp..... sheets	104 to 108 (5) pages 208-215 413 lines
148	147 SellButton.h..... sheets	108 to 108 (1) pages 216-216 26 lines
149	148 SellButton.cpp..... sheets	109 to 109 (1) pages 217-217 25 lines
150	149 SelectButton.h..... sheets	109 to 109 (1) pages 218-218 28 lines
151	150 SelectButton.cpp..... sheets	110 to 110 (1) pages 219-219 57 lines
152	151 RetireItemButton.h..... sheets	110 to 110 (1) pages 220-220 26 lines
153	152 RetireItemButton.cpp..... sheets	111 to 111 (1) pages 221-221 25 lines
154	153 RetireGoldButton.h..... sheets	111 to 111 (1) pages 222-222 26 lines
155	154 RetireGoldButton.cpp..... sheets	112 to 112 (1) pages 223-223 25 lines
156	155 ResurrectButton.h..... sheets	112 to 112 (1) pages 224-224 26 lines
157	156 ResurrectButton.cpp..... sheets	113 to 113 (1) pages 225-225 25 lines
158	157 RaisedButton.h..... sheets	113 to 113 (1) pages 226-226 25 lines
159	158 RaisedButton.cpp..... sheets	114 to 114 (1) pages 227-227 50 lines
160	159 PriestInterface.h..... sheets	114 to 114 (1) pages 228-228 32 lines
161	160 PriestInterface.cpp..... sheets	115 to 116 (2) pages 229-231 130 lines
162	161 NPCInterface.h..... sheets	116 to 116 (1) pages 232-232 37 lines
163	162 NPCInterface.cpp..... sheets	117 to 117 (1) pages 233-234 67 lines
164	163 MerchantInterface.h..... sheets	118 to 118 (1) pages 235-235 36 lines
165	164 MerchantInterface.cpp..... sheets	118 to 119 (2) pages 236-238 150 lines
166	165 EquipButton.h..... sheets	120 to 120 (1) pages 239-239 25 lines
167	166 EquipButton.cpp..... sheets	120 to 120 (1) pages 240-240 25 lines
168	167 DropButton.h..... sheets	121 to 121 (1) pages 241-241 26 lines
169	168 DropButton.cpp..... sheets	121 to 121 (1) pages 242-242 25 lines
170	169 DepositItemButton.h..... sheets	122 to 122 (1) pages 243-243 26 lines
171	170 DepositItemButton.cpp..... sheets	122 to 122 (1) pages 244-244 25 lines
172	171 DepositGoldButton.h..... sheets	123 to 123 (1) pages 245-245 26 lines
173	172 DepositGoldButton.cpp..... sheets	123 to 123 (1) pages 246-246 25 lines
174	173 CureButton.h..... sheets	124 to 124 (1) pages 247-247 26 lines
175	174 CureButton.cpp..... sheets	124 to 124 (1) pages 248-248 25 lines
176	175 BuyButton.h..... sheets	125 to 125 (1) pages 249-249 26 lines
177	176 BuyButton.cpp..... sheets	125 to 125 (1) pages 250-250 25 lines
178	177 Button.h..... sheets	126 to 126 (1) pages 251-251 37 lines
179	178 Button.cpp..... sheets	126 to 126 (1) pages 252-252 29 lines
180	179 BankerInterface.h..... sheets	127 to 127 (1) pages 253-253 41 lines
181	180 BankerInterface.cpp..... sheets	127 to 129 (3) pages 254-258 248 lines
182	181 ArrowButton.h..... sheets	130 to 130 (1) pages 259-259 24 lines
183	182 ArrowButton.cpp..... sheets	130 to 130 (1) pages 260-260 49 lines
184	183 Tile.h..... sheets	131 to 131 (1) pages 261-261 32 lines
185	184 Tile.cpp..... sheets	131 to 131 (1) pages 262-262 27 lines
186	185 TextureManager.h..... sheets	132 to 132 (1) pages 263-263 42 lines
187	186 TextureManager.cpp..... sheets	132 to 135 (4) pages 264-269 323 lines
188	187 TextureID.h..... sheets	135 to 136 (2) pages 270-271 78 lines
189	188 Texture.h..... sheets	136 to 136 (1) pages 272-272 30 lines
190	189 Texture.cpp..... sheets	137 to 137 (1) pages 273-274 74 lines
191	190 Receiver.h..... sheets	138 to 138 (1) pages 275-275 28 lines
192	191 Receiver.cpp..... sheets	138 to 138 (1) pages 276-276 43 lines
193	192 Presentation.h..... sheets	139 to 139 (1) pages 277-277 21 lines
194	193 Presentation.cpp..... sheets	139 to 139 (1) pages 278-278 47 lines
195	194 Player.h..... sheets	140 to 140 (1) pages 279-279 65 lines
196	195 Player.cpp..... sheets	140 to 142 (3) pages 280-284 289 lines
197	196 NPC.h..... sheets	143 to 143 (1) pages 285-285 36 lines
198	197 NPC.cpp..... sheets	143 to 144 (2) pages 286-288 140 lines

jul 21, 20 15:20	Table of Content	Page 4/5
199	198 MusicManager.h.....	sheets 145 to 145 (1) pages 289-289 35 lines
200	199 MusicManager.cpp.....	sheets 145 to 145 (1) pages 290-290 63 lines
201	200 MusicTD.h.....	sheets 146 to 146 (1) pages 291-291 32 lines
202	201 Music.h.....	sheets 146 to 146 (1) pages 292-292 29 lines
203	202 Music.cpp.....	sheets 147 to 147 (1) pages 293-293 50 lines
204	203 ZombieHead.h.....	sheets 147 to 147 (1) pages 294-294 20 lines
205	204 ZombieHead.cpp.....	sheets 148 to 148 (1) pages 295-295 21 lines
206	205 ZombieBody.h.....	sheets 148 to 148 (1) pages 296-296 22 lines
207	206 ZombieBody.cpp.....	sheets 149 to 149 (1) pages 297-297 28 lines
208	207 Weapon.h.....	sheets 149 to 149 (1) pages 298-298 33 lines
209	208 Weapon.cpp.....	sheets 150 to 150 (1) pages 299-299 28 lines
210	209 TurtleShield.h.....	sheets 150 to 150 (1) pages 300-300 22 lines
211	210 TurtleShield.cpp.....	sheets 151 to 151 (1) pages 301-301 29 lines
212	211 SpiderBody.h.....	sheets 151 to 151 (1) pages 302-302 20 lines
213	212 SpiderBody.cpp.....	sheets 152 to 152 (1) pages 303-303 22 lines
214	213 SkeletonBody.h.....	sheets 152 to 152 (1) pages 304-304 22 lines
215	214 SkeletonBody.cpp.....	sheets 153 to 153 (1) pages 305-305 28 lines
216	215 SimpleArc.h.....	sheets 153 to 153 (1) pages 306-306 22 lines
217	216 SimpleArc.cpp.....	sheets 154 to 154 (1) pages 307-307 30 lines
218	217 Shield.h.....	sheets 154 to 154 (1) pages 308-308 31 lines
219	218 Shield.cpp.....	sheets 155 to 155 (1) pages 309-309 24 lines
220	219 RedCommonBody.h.....	sheets 155 to 155 (1) pages 310-310 22 lines
221	220 RedCommonBody.cpp.....	sheets 156 to 156 (1) pages 311-311 30 lines
222	221 PriestHead.h.....	sheets 156 to 156 (1) pages 312-312 20 lines
223	222 PriestHead.cpp.....	sheets 157 to 157 (1) pages 313-313 21 lines
224	223 PriestBody.h.....	sheets 157 to 157 (1) pages 314-314 22 lines
225	224 PriestBody.cpp.....	sheets 158 to 158 (1) pages 315-315 28 lines
226	225 PlateArmor.h.....	sheets 158 to 158 (1) pages 316-316 22 lines
227	226 PlateArmor.cpp.....	sheets 159 to 159 (1) pages 317-317 30 lines
228	227 MissileAnimation.h.....	sheets 159 to 159 (1) pages 318-318 19 lines
229	228 MissileAnimation.cpp.....	sheets 160 to 160 (1) pages 319-319 23 lines
230	229 MerchantBody.h.....	sheets 160 to 160 (1) pages 320-320 22 lines
231	230 MerchantBody.cpp.....	sheets 161 to 161 (1) pages 321-321 27 lines
232	231 MeditateAnimation.h.....	sheets 161 to 161 (1) pages 322-322 19 lines
233	232 MeditateAnimation.cpp.....	sheets 162 to 162 (1) pages 323-323 24 lines
234	233 MagicHat.h.....	sheets 162 to 162 (1) pages 324-324 20 lines
235	234 MagicHat.cpp.....	sheets 163 to 163 (1) pages 325-325 21 lines
236	235 MagicArrowAnimation.h.....	sheets 163 to 163 (1) pages 326-326 19 lines
237	236 MagicArrowAnimation.cpp.....	sheets 164 to 164 (1) pages 327-327 23 lines
238	237 LongSword.h.....	sheets 164 to 164 (1) pages 328-328 22 lines
239	238 LongSword.cpp.....	sheets 165 to 165 (1) pages 329-329 30 lines
240	239 LeatherArmor.h.....	sheets 165 to 165 (1) pages 330-330 22 lines
241	240 LeatherArmor.cpp.....	sheets 166 to 166 (1) pages 331-331 29 lines
242	241 Item.h.....	sheets 166 to 166 (1) pages 332-332 20 lines
243	242 Item.cpp.....	sheets 167 to 167 (1) pages 333-333 13 lines
244	243 IronShield.h.....	sheets 167 to 167 (1) pages 334-334 22 lines
245	244 IronShield.cpp.....	sheets 168 to 168 (1) pages 335-335 29 lines
246	245 IronHelmet.h.....	sheets 168 to 168 (1) pages 336-336 20 lines
247	246 IronHelmet.cpp.....	sheets 169 to 169 (1) pages 337-337 21 lines
248	247 HumanHead.h.....	sheets 169 to 169 (1) pages 338-338 20 lines
249	248 HumanHead.cpp.....	sheets 170 to 170 (1) pages 339-339 22 lines
250	249 Hood.h.....	sheets 170 to 170 (1) pages 340-340 20 lines
251	250 Hood.cpp.....	sheets 171 to 171 (1) pages 341-341 21 lines
252	251 HitAnimation.h.....	sheets 171 to 171 (1) pages 342-342 19 lines
253	252 HitAnimation.cpp.....	sheets 172 to 172 (1) pages 343-343 25 lines
254	253 Helmet.h.....	sheets 172 to 172 (1) pages 344-344 28 lines
255	254 Head.h.....	sheets 173 to 173 (1) pages 345-345 28 lines
256	255 Hammer.h.....	sheets 173 to 173 (1) pages 346-346 22 lines
257	256 Hammer.cpp.....	sheets 174 to 174 (1) pages 347-347 30 lines
258	257 GreenCommonBody.h.....	sheets 174 to 174 (1) pages 348-348 22 lines
259	258 GreenCommonBody.cpp.....	sheets 175 to 175 (1) pages 349-349 30 lines
260	259 GoblinBody.h.....	sheets 175 to 175 (1) pages 350-350 22 lines
261	260 GoblinBody.cpp.....	sheets 176 to 176 (1) pages 351-351 25 lines
262	261 GnomeHead.h.....	sheets 176 to 176 (1) pages 352-352 20 lines
263	262 GnomeHead.cpp.....	sheets 177 to 177 (1) pages 353-353 21 lines
264	263 GnarledStick.h.....	sheets 177 to 177 (1) pages 354-354 22 lines

jul 21, 20 15:20	Table of Content	Page 5/5
265	264 GnarledStick.cpp.....	sheets 178 to 178 (1) pages 355-355 30 lines
266	265 GhostBody.h.....	sheets 178 to 178 (1) pages 356-356 22 lines
267	266 GhostBody.cpp.....	sheets 179 to 179 (1) pages 357-357 25 lines
268	267 ExplosionAnimation.h.....	sheets 179 to 179 (1) pages 358-358 19 lines
269	268 ExplosionAnimation.cpp.....	sheets 180 to 180 (1) pages 359-359 25 lines
270	269 ElficFlaute.h.....	sheets 180 to 180 (1) pages 360-360 22 lines
271	270 ElficFlaute.cpp.....	sheets 181 to 181 (1) pages 361-361 30 lines
272	271 ElfHead.h.....	sheets 181 to 181 (1) pages 362-362 20 lines
273	272 ElfHead.cpp.....	sheets 182 to 182 (1) pages 363-363 21 lines
274	273 DwarfHead.h.....	sheets 182 to 182 (1) pages 364-364 20 lines
275	274 DwarfHead.cpp.....	sheets 183 to 183 (1) pages 365-365 21 lines
276	275 CureAnimation.h.....	sheets 183 to 183 (1) pages 366-366 19 lines
277	276 CureAnimation.cpp.....	sheets 184 to 184 (1) pages 367-367 25 lines
278	277 Crosier.h.....	sheets 184 to 184 (1) pages 368-368 22 lines
279	278 Crosier.cpp.....	sheets 185 to 185 (1) pages 369-369 30 lines
280	279 CompoundArc.h.....	sheets 185 to 185 (1) pages 370-370 22 lines
281	280 CompoundArc.cpp.....	sheets 186 to 186 (1) pages 371-371 30 lines
282	281 Body.h.....	sheets 186 to 186 (1) pages 372-372 33 lines
283	282 Body.cpp.....	sheets 187 to 187 (1) pages 373-373 28 lines
284	283 BlueTunic.h.....	sheets 187 to 187 (1) pages 374-374 22 lines
285	284 BlueTunic.cpp.....	sheets 188 to 188 (1) pages 375-375 30 lines
286	285 BlueCommonBody.h.....	sheets 188 to 188 (1) pages 376-376 22 lines
287	286 BlueCommonBody.cpp.....	sheets 189 to 189 (1) pages 377-377 29 lines
288	287 BankerBody.h.....	sheets 189 to 189 (1) pages 378-378 22 lines
289	288 BankerBody.cpp.....	sheets 190 to 190 (1) pages 379-379 30 lines
290	289 Ax.h.....	sheets 190 to 190 (1) pages 380-380 22 lines
291	290 Ax.cpp.....	sheets 191 to 191 (1) pages 381-381 30 lines
292	291 AshStick.h.....	sheets 191 to 191 (1) pages 382-382 22 lines
293	292 AshStick.cpp.....	sheets 192 to 192 (1) pages 383-383 30 lines
294	293 Animation.h.....	sheets 192 to 192 (1) pages 384-384 34 lines
295	294 Animation.cpp.....	sheets 193 to 193 (1) pages 385-385 19 lines
296	295 GameMap.h.....	sheets 193 to 193 (1) pages 386-386 37 lines
297	296 GameMap.cpp.....	sheets 194 to 194 (1) pages 387-388 90 lines
298	297 Game.h.....	sheets 195 to 195 (1) pages 389-389 67 lines
299	298 Game.cpp.....	sheets 195 to 197 (3) pages 390-394 262 lines
300	299 Font.h.....	sheets 198 to 198 (1) pages 395-395 32 lines
301	300 Font.cpp.....	sheets 198 to 198 (1) pages 396-396 51 lines
302	301 Effect.h.....	sheets 199 to 199 (1) pages 397-397 26 lines
303	302 Effect.cpp.....	sheets 199 to 199 (1) pages 398-398 37 lines
304	303 Dispatcher.h.....	sheets 200 to 200 (1) pages 399-399 30 lines
305	304 Dispatcher.cpp.....	sheets 200 to 200 (1) pages 400-400 48 lines
306	305 client_main.cpp.....	sheets 201 to 201 (1) pages 401-401 23 lines
307	306 StillState.h.....	sheets 201 to 201 (1) pages 402-402 34 lines
308	307 StillState.cpp.....	sheets 202 to 203 (2) pages 403-405 145 lines
309	308 ResurrectState.h.....	sheets 203 to 203 (1) pages 406-406 34 lines
310	309 ResurrectState.cpp.....	sheets 204 to 205 (2) pages 407-409 143 lines
311	310 MoveState.h.....	sheets 205 to 205 (1) pages 410-410 34 lines
312	311 MoveState.cpp.....	sheets 206 to 207 (2) pages 411-413 151 lines
313	312 MeditateState.h.....	sheets 207 to 207 (1) pages 414-414 34 lines
314	313 MeditateState.cpp.....	sheets 208 to 209 (2) pages 415-417 146 lines
315	314 InteractState.h.....	sheets 209 to 209 (1) pages 418-418 36 lines
316	315 InteractState.cpp.....	sheets 210 to 211 (2) pages 419-421 184 lines
317	316 CharacterState.h.....	sheets 211 to 211 (1) pages 422-422 43 lines
318	317 AttackState.h.....	sheets 212 to 212 (1) pages 423-423 36 lines
319	318 AttackState.cpp.....	sheets 212 to 213 (2) pages 424-426 169 lines
320	319 Character.h.....	sheets 213 to 213 (1) pages 426-426 1 lines
321	320 Character.cpp.....	sheets 213 to 213 (1) pages 426-426 1 lines
322	321 Camera.h.....	sheets 214 to 214 (1) pages 427-427 55 lines
323	322 Camera.cpp.....	sheets 214 to 215 (2) pages 428-429 81 lines