Documentación Técnica

Requisitos tecnológicos

Lenguaje de programación: c++ versión 11 o más
Cmake: version 3.16 o más
Qt: version 5 o más
Librería SLD2PP
Librería yaml-cpp

Descripción del proyecto

El proyecto se puede separar en 3 partes distintas.

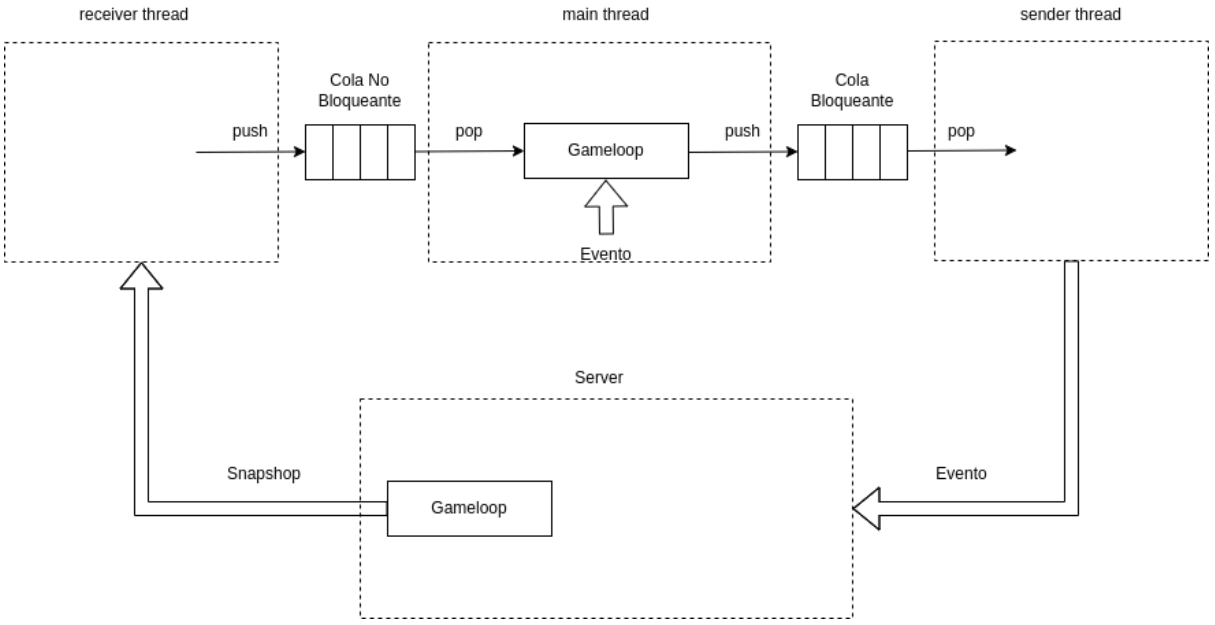
Server: encargado de la lógica del juego y enviarle al cliente los datos necesarios para graficar el juego.

Cliente: encargado de graficar el juego y enviarle los comandos al servidor

Editor: programa capaz de crear y modificar mapas

Cliente

El siguiente diagrama muestra cómo funciona el cliente:

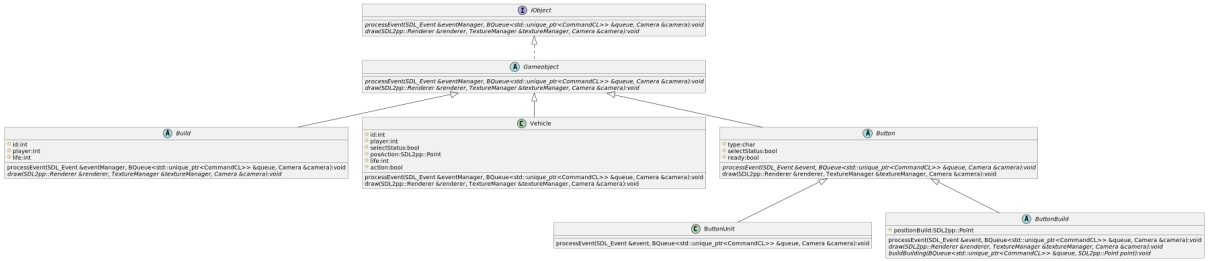


Tenemos un hilo recibidor el cual recibe del cliente un snapshot del modelo este se guarda en un vector de Gameobject el cual luego se pushea en una cola no bloqueante.

Tenemos un hilo main con un gameloop el cual popea de la cola no bloqueante y renderiza el modelo. Este cola debe ser no bloqueante ya que el hilo debe hacer varias cosas entre ellas renderizar el modelo y mirar los eventos de teclado mouse. Estos eventos son procesados por los objetos del juego y se pushea en una cola bloqueante.
Tenemos un hilo enviador el cual pushea de esa cola de comandos y envía este comando al servidor para actualizar el modelo.

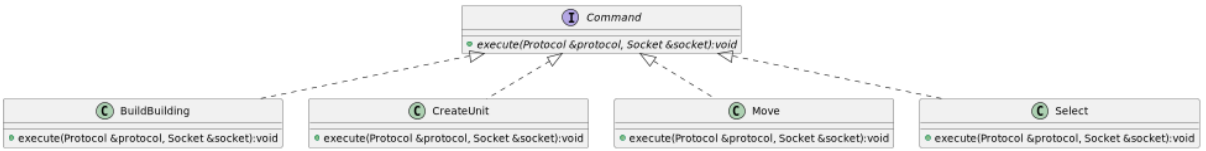
Diagrama de clases

GameObjects



Se decidió usar interfaz para los objetos del juego todos los objetos tienen que implementa el método processEvent y draw de forma que poder tener un vector de objetos y poder hacer que los objetos procesen un evento y también puedan ser renderizados

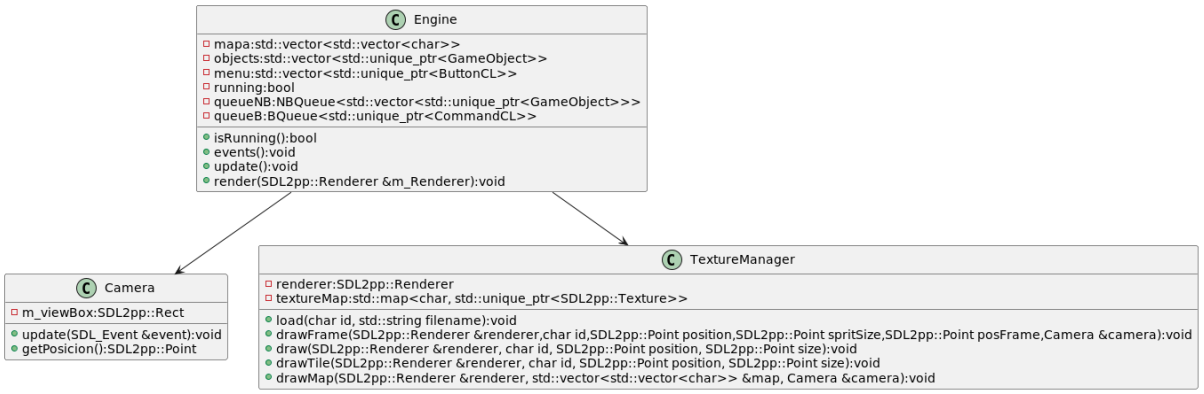
Commands



todos los objetos los objetos implementa el método execute el cual envia el comando al servidor

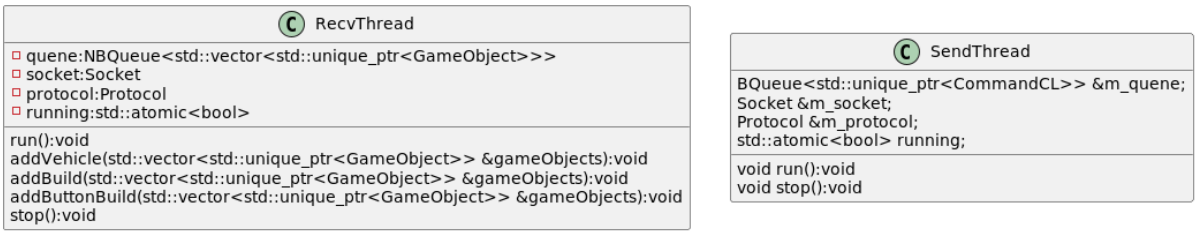
Engine

Esta clase es la que tiene los métodos de que usa el game loop
la de procesar los eventos actualizar el modelo y renderizar



Hilos Send y Recv

se encarga de enviar comandos y recibir el snapshot del servidor



Server

El servidor es el encargado de manejar toda la lógica del juego, de aceptar nuevas conexiones y de establecer múltiples partidas en simultáneo.
Para lograrlo se utilizaron sockets TCP y Multithreading, todo con el lenguaje de programación C + + y el estándar C++17.

Editor

El editor es un programa que va aparte de la arquitectura cliente-servidor, ya que no es necesario que se conecte al servidor.
Para realizarlo se utilizó la biblioteca de Qt y se aprovechó de las facilidades que provee del Qt Creator a la hora de realizar GUI.
El diseño de los botones, layouts y pantalla fue realizado por Qt creator, aprovechando la automatización que este provee mediante el uso de archivos ui y meta objetos.

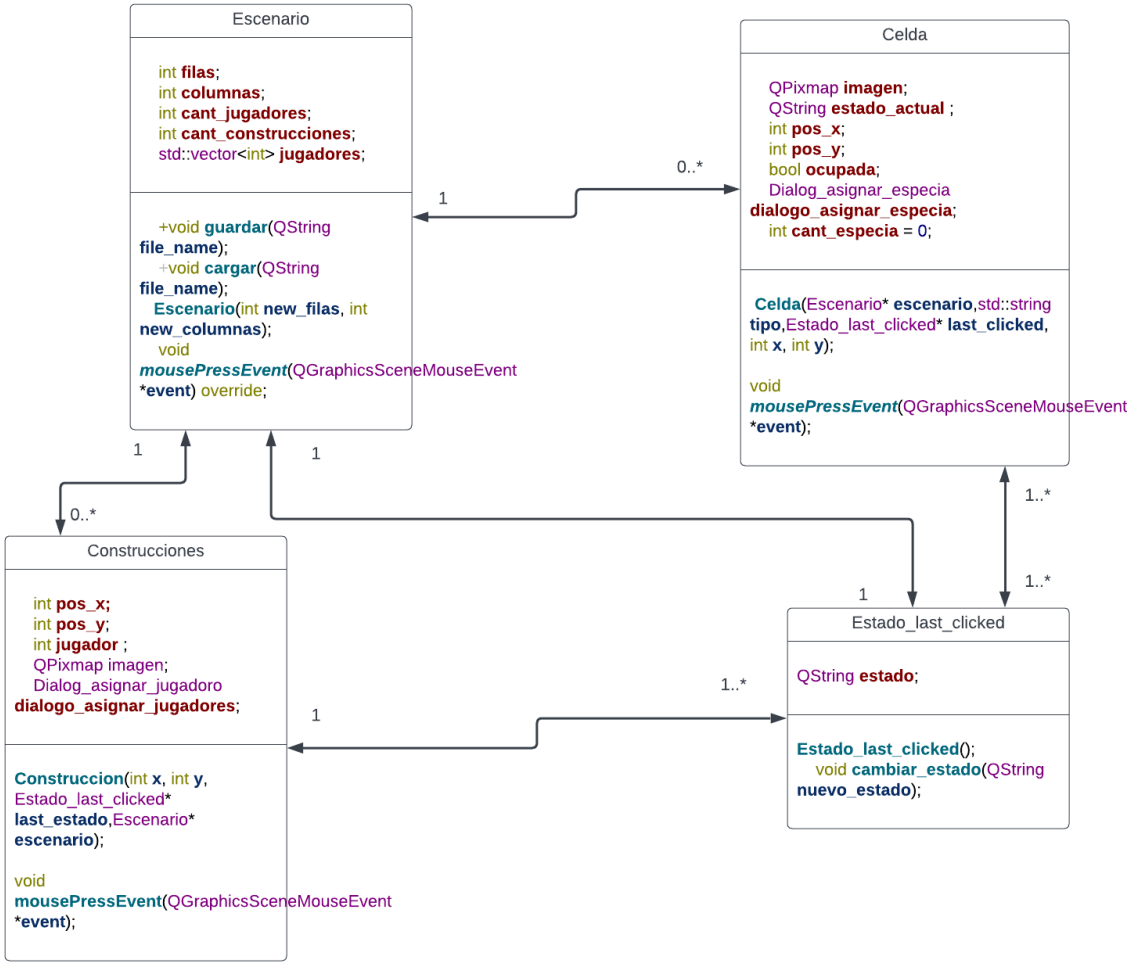
Los botones son pushButtons de Qt, las celdas del mapa heredan de QGraphicsPixmap y de QObject, las construcciones heredan de QGraphicsPixmap y de QObject, el escenario hereda de QGraphicsScene y es agregada a una QGraphicsView para poder desplegar el escenario.

Para poder ingresar datos se implementó un sistema de señales que eran enviadas a los objetos y cuando se reciben se levanta un QDialog para ingresar los datos.

Este sistema se usó para cambiar la cantidad de columnas y filas del mapa, para asignar la cantidad de jugadores, asignar un jugador a una construcción y para asignar una cantidad de especia a la celda Especia.

Además para poder parsear los clicks del usuario se creó la clase Estado_last_clicked el cual guardará el último botón clickeado. Este sistema se usa para cambiará el tipo de celda, crear y eliminar construcciones, asignar jugadores a las construcciones y asignar la cantidad de especia.

Diagrama de clase del Editor



Acciones que envía el Cliente al Servidor

PRIMEROS MENSAJES:

1* **Nombre** std::string
2* **Casa** uint8_t
3* **Crear partida**: uint16_t 1
4* **Unirse a partida**: uint16_t 2
5* **Listar partidas**: uint16_t

El cliente recibe una posición y una acción (seleccionar-desplazar-atacar).
El cliente mira que hay en esa posición:
Si hay un objeto del jugador la selecciona
Si hay un objeto del enemigo la ataca con los objetos seleccionados
Si está vacía mueve los objetos seleccionados a esa posición

Seleccionar Objeto
<accion><id>

Desplazar(Atacar)
<accion><id><posición>

Construir Edificio
<acción><Edificio><posicion>

Crear Unidad
<acción><unidad>

Accion: <uint8_t>

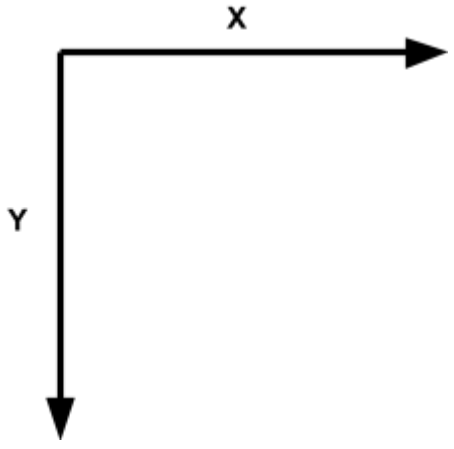
- seleccionar: **5**
- desplazar (atacar): **6**
- construir edificio: **7**
- crear unidad: **8**

Id Objeto: <id: uint16_t>

Unidad: <uint8_t>

Edificio: <uint8_t>

Posicion: <posX: uint16_t><posY: uint16_t>



El servidor responde con un snapshot del instante T

Snapshot

<cant objetos><objeto 1>....<objeto N>

Objetos

Unidad

<tipo><unidad><id><jugador><seleccionado><pos objeto><pos acción><ptos vida><acción>

Edificio

<tipo><edificio><id><jugador><pos objeto><ptos vida>

Botón Crear Unidad

<tipo><edificio><id><jugador><tiempo de construcción><seleccionado><ready>

Botón Crear Edificio

<tipo><edificio><id><jugador><tiempo de construcción><seleccionado><ready>

Gusano de Arena

<tipo><id><posX><posY><acción>

Tipo: <uint8_t>

- Vehiculo: 0
- Edificio: 1
- Botón: 2
- Gusano de Arena: 3

Unidad: <uint8_t>

Edificio: <uint8_t>

Jugador: <uint8_t>

Seleccionado: <uint8_t> 1: Seleccionado 0: no Seleccionado

Pos Objeto - Pos Accion: <posX: uint16_t><posY: uint16_t>

Puntos Vida: <uint16_t>

Accion: <uint8_t>

- mover: 0
- atacar: 1