МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный университет геодезии и картографии»

(МИИГАиК)

Факультет геоинформатики и информационной безопасности

Кафедра геоинформационных систем и технологий

**Лабораторная работа №5 "ООП"**

Проверил: Выполнил:

Лебедев Е.Д. Студент группы: 2024-ФГИИБ-ПИ-16

Зазыкин Иван Дмитриевич

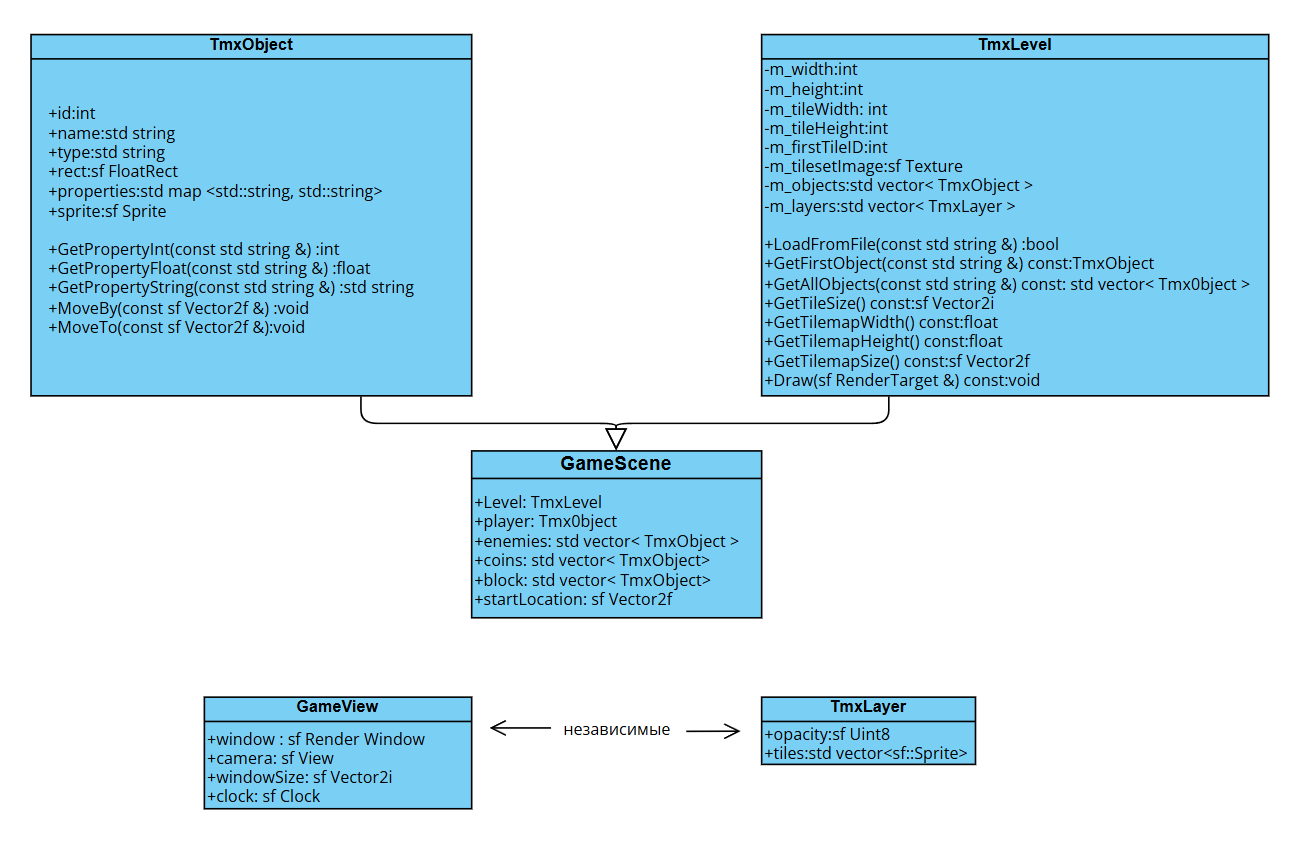
Москва 2024

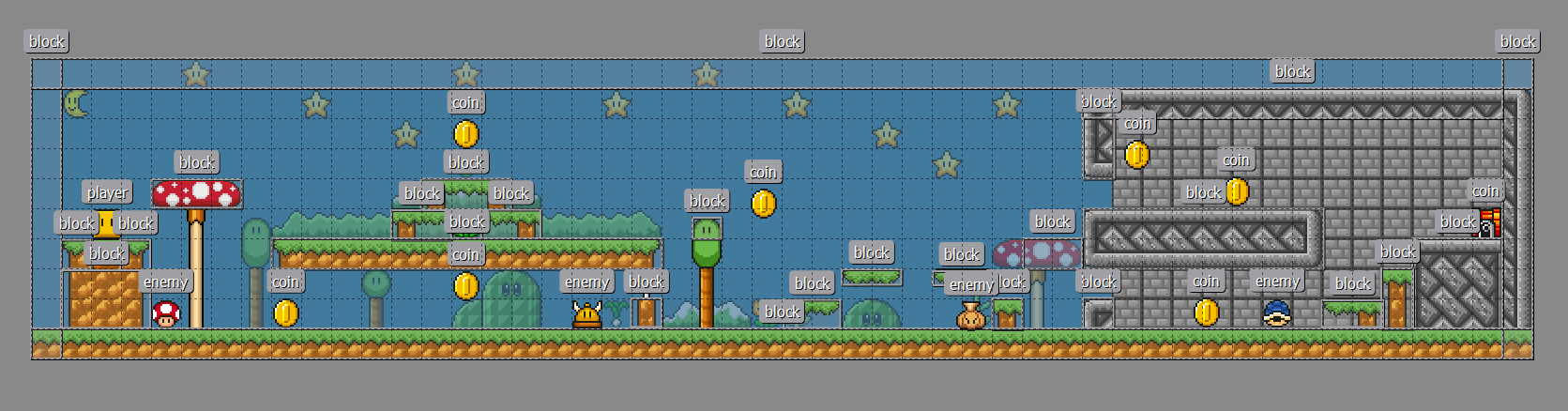
**Глава 1** **"Разработка средств загрузки уровня"**

Поставленные задачи:

* Нарисовать диаграмму классов по всем файлам проекта, кроме **tinyxml2.cpp** и **tinyxml2.h**
* Модифицировать уровень, используя **Tiled**. Показать результат на рисунке.
* Приложить код файлов хэдеров с описанием в виде листингов (кроме **tinyxml2.h**)

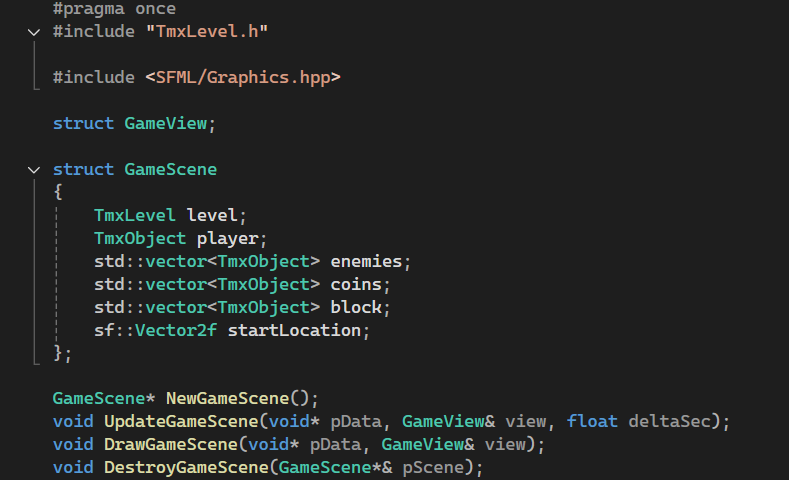
Диаграмма классов:

Модифицированный уровень **Tiled.**



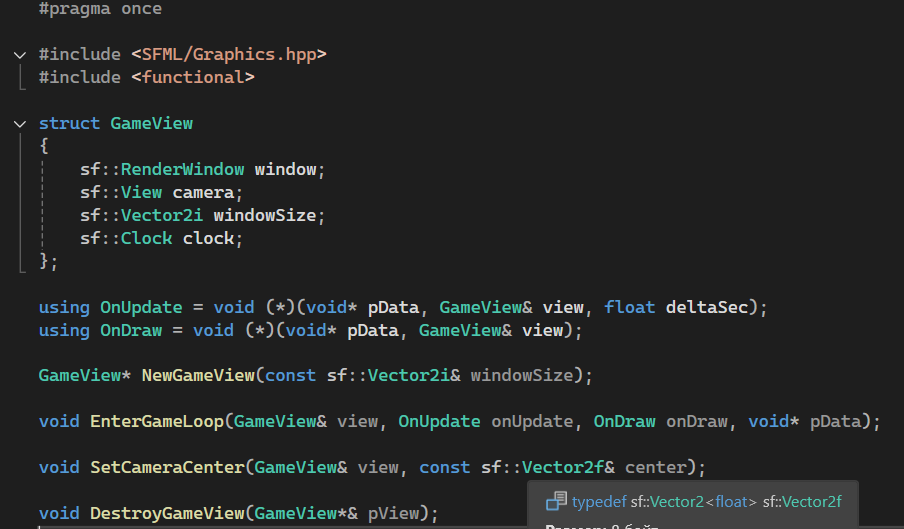
Листинг хедер файлов (кроме **tinyxml2.h**).

1.**GameScene.h**



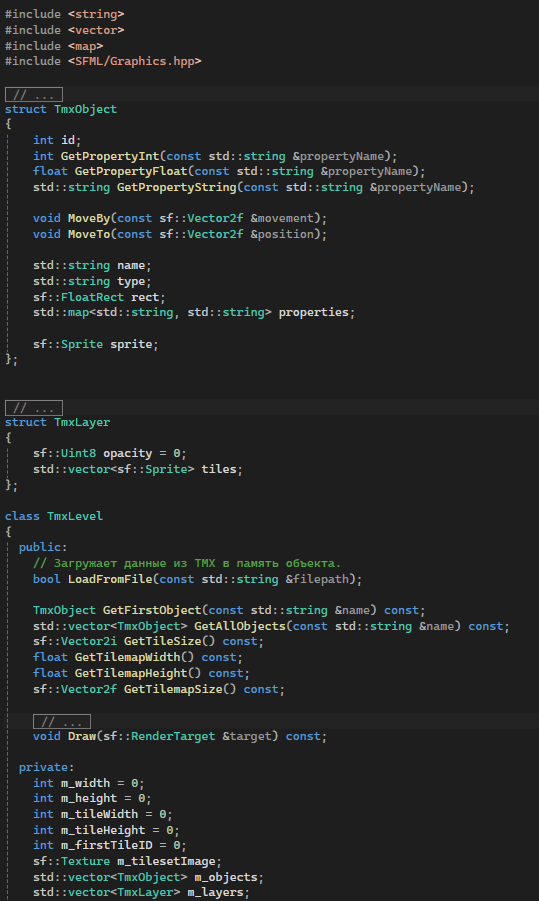
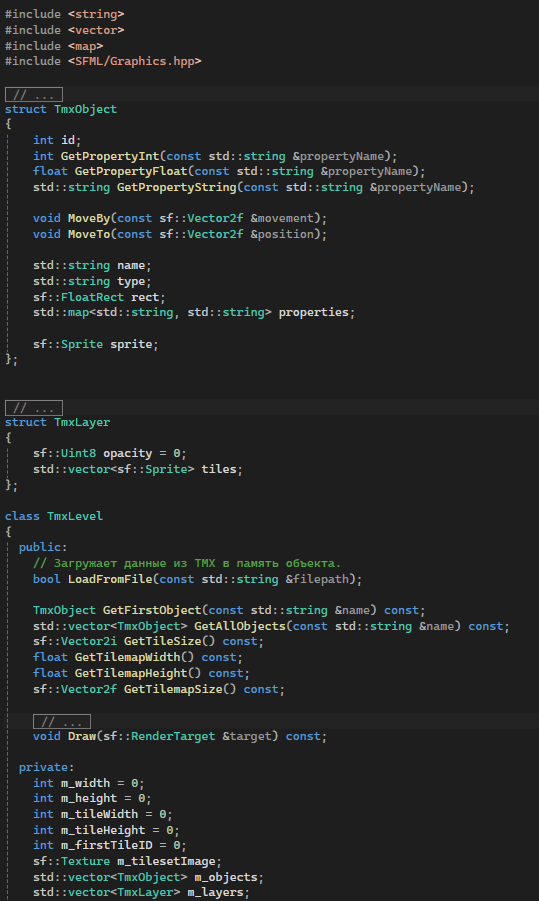
* **#include "TmxLevel.h"**: Подключает заголовочный файл.
* **#include <SFML/Graphics.hpp>**: Подключает библиотеку SFML, которая используется для работы с графикой в C++.
* **struct GameScene**: Определяет структуру, представляющую игровую сцену, содержащую уровень, игрока, врагов, монеты и блоки.
* **Функции**:
  + **UpdateGameScene**: Обновляет состояние игровой сцены.
  + **DrawGameScene**: Отрисовывает игровую сцену на экране.
  + **DestroyGameScene**: Освобождает ресурсы, связанные с игровой сценой.

2.**GameView.h**



* **struct GameView**: Определяет структуру, представляющую игровой интерфейс.
  + **sf::RenderWindow window**: Окно для отображения графики.
  + **sf::View camera**: Камера.
  + **sf::Vector2i windowSize**: Хранит размеры окна (ширина и высота).
* **using OnUpdate и using OnDraw**: Определяют типы указателей на функции для обновления и отрисовки, что позволяет передавать функции в качестве параметров.
* **Функции**:
  + **NewGameView**: Создает и возвращает указатель на новый объект GameView с заданными размерами окна.
  + **EnterGameLoop**: Запускает игровой цикл, принимая функции обновления и отрисовки, а также данные.
  + **SetCameraCenter**: Устанавливает центр камеры для отображения сцены.
  + **DestroyGameView**: Освобождает ресурсы, связанные с объектом GameView.

3.**TmxLevel.h**

* **struct TmxObject**: Определяет объект на карте, который содержит информацию о его свойствах и методах манипуляции.
  + Поля для хранения идентификатора, имени, типа, границ, пользовательских свойств и спрайта.
  + Методы для получения свойств и перемещения объекта.
* **struct TmxLayer**: Определяет слой на карте, который содержит тайлы.
  + Поля для хранения прозрачности слоя и вектора спрайтов (тайлов).
* **class TmxLevel**: Основной класс для работы с уровнем, загружающим данные из файла **tmx**.
  + Методы для загрузки данных, получения объектов, получения размеров тайлов и рисования слоев.
  + Приватные поля для хранения информации о размерах уровня, текстуре тайлового набора, объектах и слоях.

**Глава 2** **"Разработка логики взаимодействия с объектами"**

Поставленные задачи:

* Описать логику инциализации сцены (**GameScene \*NewGameScene()**) и логику обработки взаимодействия с объектами. Дать краткое описание каждого взаимодействия.
* Опиcать какими методами пользовался (какие из них стандартные методы библиотеки SFML, а какие реализованы в файле проекта)
* Изменить код Хэдера в приложении (**GameScene.h**) и код (**GameScene.cpp**)
* Выложите проект вместе с .exe файлом на гитхаб. Проверьте, что .exe работает!

**Логика инициализации:**

Процесс инициализации нового объекта **GameScene** включает создание структуры и установку указателя на ее фактическое значение в памяти. Далее осуществляется инициализация всех полей, определенных в структуре **GameScene**, таких как **player, coins, enemies, block** и **startLocation**. Также выполняется загрузка данных из файла формата **\*.tmx**.

**Логика взаимодействия:**

Основная идея заключается в проверке границ объектов и определении дальнейших действий в зависимости от результатов этих проверок.

* **Враги**:  
  Добавлено условие для проверки пересечения спрайта игрока (**player.sprite**) со спрайтом врага (**enemy.sprite**). Пересечение проверяется с помощью свойства **intersects**. В случае пересечения, игрок перемещается на свою начальную позицию.
* **Монеты**:  
  Реализовано условие для проверки пересечения спрайта игрока (**player.sprite**) со спрайтом монет (**coins.sprite**). Проверка пересечения выполняется через свойство **intersects**. Если пересечение произошло, объект монеты удаляется с помощью метода **erase**.
* **Препятствия**:  
  Введено условие для проверки пересечения спрайта игрока (**player.sprite**) с объектом препятствия (**sf::FloatRect rect**). Пересечение также проверяется с помощью свойства **intersects**.

Хранение монет реализовано с помощью встроенных инструментов языка, а именно структуры **std::vector<TmxObject**>, где **TmxObject** содержит общие характеристики всех объектов на сцене.

**Описание использованных методов:**

Объекты **Rectangle**, **Vector2f** и другие взаимодействуют с графикой благодаря библиотеки SFML. Остальные файлы реализованы внутри проекта и не относятся к SFML.

Проект на GitHub: [Лабораторная работа №5](https://github.com/re-side/Inf_university/tree/main/laboratory%20work%205/tiledgame)