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:

Лебедев Е.Д.

Выполнил:

Студент группы: 2024-ФГИИБ-ПИ-16

Шамадаев Рустам Эльдарович

Москва 2024

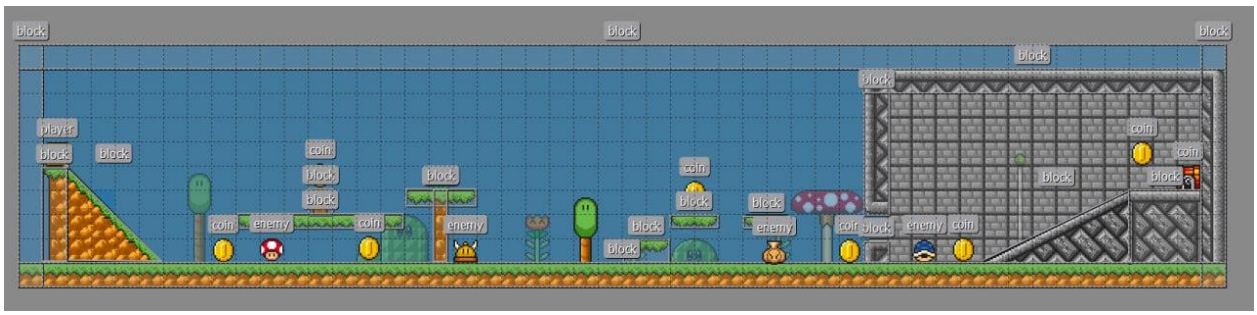
# **Глава 1** "Разработка средств загрузки уровня"

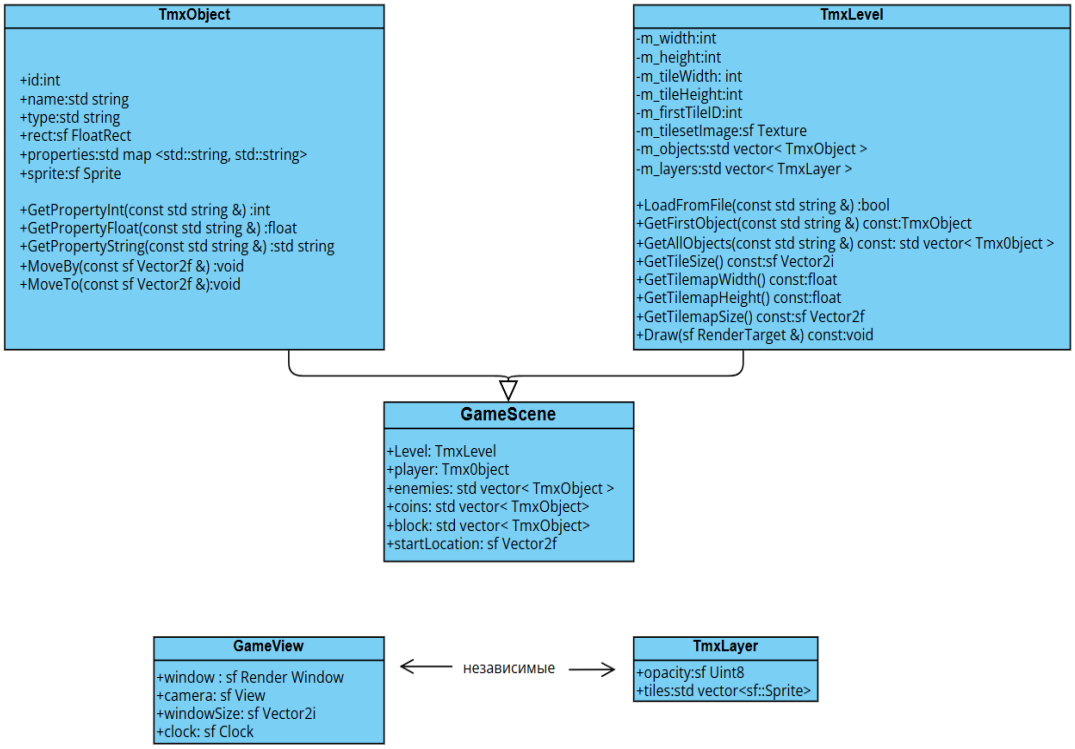
Поставленные задачи:

- 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**

```cpp
#pragma once
#include "TmxLevel.h"

#include <SFML/Graphics.hpp>

struct GameView;

struct GameScene
{
    TmxLevel level;
    TmxObject player;
    std::vector<TmxObject> enemies;
    std::vector<TmxObject> coins;
    std::vector<TmxObject> block;
    sf::Vector2f startLocation;
};

GameScene* NewGameScene();
void UpdateGameScene(void* pData, GameView& view, float deltaSec);
void DrawGameScene(void* pData, GameView& view);
void DestroyGameScene(GameScene*& pScene);
```

- **#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.

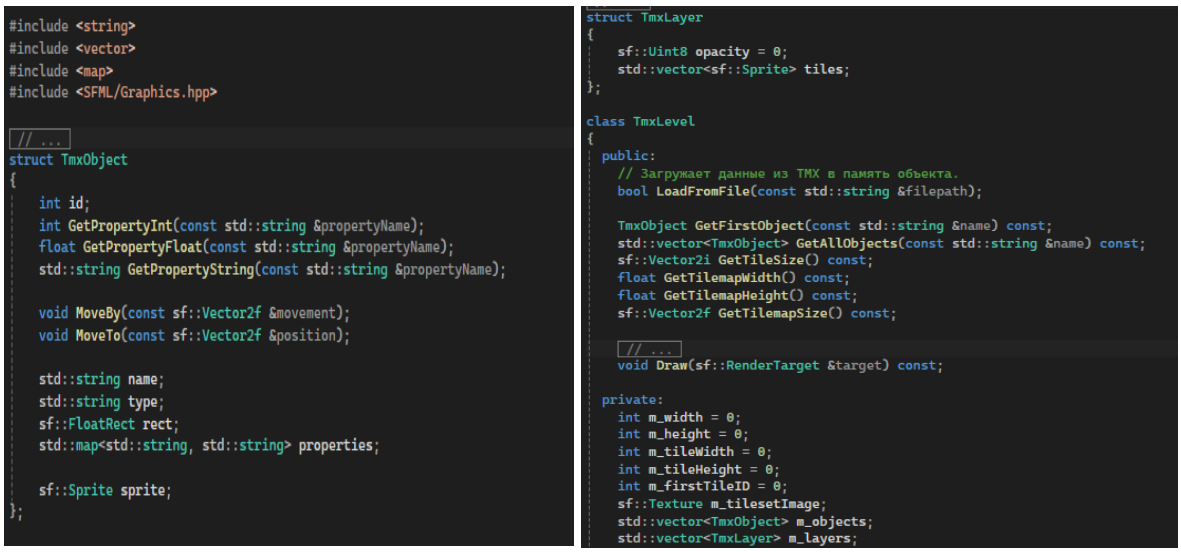
2.**GameView.h**

```cpp
#pragma once

#include <SFML/Graphics.hpp>
#include <functional>

struct GameView
{
    sf::RenderWindow window;
    sf::View camera;
    sf::Vector2i windowSize;
    sf::Clock clock;
};

using OnUpdate = void (*)(void* pData, GameView& view, float deltaSec);
using OnDraw = void (*)(void* pData, GameView& view);

GameView* NewGameView(const sf::Vector2i& windowSize);

void EnterGameLoop(GameView& view, OnUpdate onUpdate, OnDraw onDraw, void* pData);

void SetCameraCenter(GameView& view, const sf::Vector2f& center);

void DestroyGameView(GameView*& pView);
```



- **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**



```
#include <string>
#include <vector>
#include <map>
#include <SFML/Graphics.hpp>

// ...
struct TmxObject
{
    int id;
    int GetPropertyInt(const std::string &propertyName);
    float GetPropertyFloat(const std::string &propertyName);
    std::string GetPropertyString(const std::string &propertyName);

    void MoveBy(const sf::Vector2f &movement);
    void MoveTo(const sf::Vector2f &position);

    std::string name;
    std::string type;
    sf::FloatRect rect;
    std::map<std::string, std::string> properties;

    sf::Sprite sprite;
};
```

```
struct TmxLayer
{
    sf::Uint8 opacity = 0;
    std::vector<sf::Sprite> tiles;
};

class TmxLevel
{
  public:
    // Загружает данные из TMX в память объекта.
    bool LoadFromFile(const std::string &filepath);

    TmxObject GetFirstObject(const std::string &name) const;
    std::vector<TmxObject> GetAllObjects(const std::string &name) const;
    sf::Vector2i GetTileSize() const;
    float GetTilemapWidth() const;
    float GetTilemapHeight() const;
    sf::Vector2f GetTilemapSize() const;

    // ...
    void Draw(sf::RenderTarget &target) const;

  private:
    int m_width = 0;
    int m_height = 0;
    int m_tileWidth = 0;
    int m_tileHeight = 0;
    int m_firstTileID = 0;
    sf::Texture m_tilesetImage;
    std::vector<TmxObject> m_objects;
    std::vector<TmxLayer> m_layers;
```

**struct TmxObject**: Описание объекта на карте, включающее информацию о его свойствах и методах взаимодействия.

Поля для хранения уникального идентификатора, имени, типа, границ, пользовательских свойств и спрайта.

Методы для получения данных об объекте и манипуляции его позиционированием.

**struct TmxLayer**: Описание слоя карты, содержащего тайлы.

Поля для хранения прозрачности слоя и вектора спрайтов (тайлов).

**class TmxLevel**: Главный класс для работы с уровнем, загружающий данные из файла формата .tmx.

Методы для загрузки данных, получения объектов, вычисления размеров тайлов и отрисовки слоев.

Приватные поля для хранения информации о размере уровня, текстуре набора тайлов, объектах и слоях.

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

**Поставленные задачи:**

1. Описать логику инициализации сцены (GameScene *NewGameScene()) и логику обработки взаимодействий с объектами. Дать краткое описание каждого типа взаимодействия.
2. Описать, какие методы использовались (какие стандартные методы библиотеки SFML, а какие реализованы в проекте).
3. Изменить код заголовочного файла (GameScene.h) и исходного файла (GameScene.cpp).
4. Опубликовать проект с .exe файлом на GitHub. Убедитесь, что .exe файл работает!

**Логика инициализации:**

Процесс создания нового объекта GameScene включает выделение памяти под структуру и установку указателя на её местоположение. Затем инициализируются все поля структуры GameScene, такие как player, coins, enemies, block и startLocation. Загружаются данные из файла формата *.tmx.

**Логика взаимодействия:**

Основной принцип заключается в проверке границ объектов и принятии решений на основе результатов этих проверок.

**Враги:**

Добавлено условие для проверки столкновения спрайта игрока (player.sprite) с врагом (enemy.sprite). Столкновение проверяется с использованием метода intersects. В случае столкновения, игрок возвращается на свою начальную позицию.

**Монеты:**

Реализована проверка пересечения спрайта игрока (player.sprite) с монетами (coins.sprite). Столкновение также проверяется через intersects. При пересечении монета удаляется с помощью метода erase.

**Препятствия:**

Добавлено условие для проверки пересечения спрайта игрока (player.sprite) с объектом препятствия (sf::FloatRect rect), используя intersects.

Хранение монет осуществляется через стандартный контейнер std::vector, где TmxObject содержит общие данные о всех объектах на сцене.

**Описание использованных методов:**

Объекты типа Rectangle, Vector2f и другие взаимодействуют с графикой через библиотеку SFML. Остальные методы и файлы реализованы в проекте и не относятся к SFML.

```cpp
struct TmxLayer
{
    sf::Uint8 opacity = 0;
    std::vector<sf::Sprite> tiles;
};

class TmxLevel
{
  public:
    // Загружает данные из TMX в память объекта.
    bool LoadFromFile(const std::string &filepath);

    TmxObject GetFirstObject(const std::string &name) const;
    std::vector<TmxObject> GetAllObjects(const std::string &name) const;
    sf::Vector2i GetTileSize() const;
    float GetTilemapWidth() const;
    float GetTilemapHeight() const;
    sf::Vector2f GetTilemapSize() const;

    // ...
    void Draw(sf::RenderTarget &target) const;

  private:
    int m_width = 0;
    int m_height = 0;
    int m_tileWidth = 0;
    int m_tileHeight = 0;
    int m_firstTileID = 0;
    sf::Texture m_tilesetImage;
    std::vector<TmxObject> m_objects;
    std::vector<TmxLayer> m_layers;
```