Top-Down Shooter

C++ Game

Проект на C++ по компьютерным технологиям

Вехов Владимир

Ильюшенков Михаил

Б01-306

Поставленные задачи

1. сделать рабочий движок:
   - клиентский модуль
   - серверный модуль
   - обработка нажатия клавиш
   - обработка пересечений (со стенами, с пулями, с игроками)
2. Графика
   - спрайты персонажей
   - карта (стены, пол)
   - спрайты пулей
3. Завершение игры

Top-Down Shooter initialization

1. Создание сервера

`./bin/server` - запуск северного модуля

Сервер ждет клиентов, затем отсылает 'глобальное состояние' своим клиентам, обрабатывает пересечения объектов.

2. Создание клиентов

`./bin/client N` - запуск N'ого клиентского модуля

Клиент считывает нажатие клавиш, отсылает их на сервер, а также отрисовывает карту и персонажей.

3. Игровой процесс

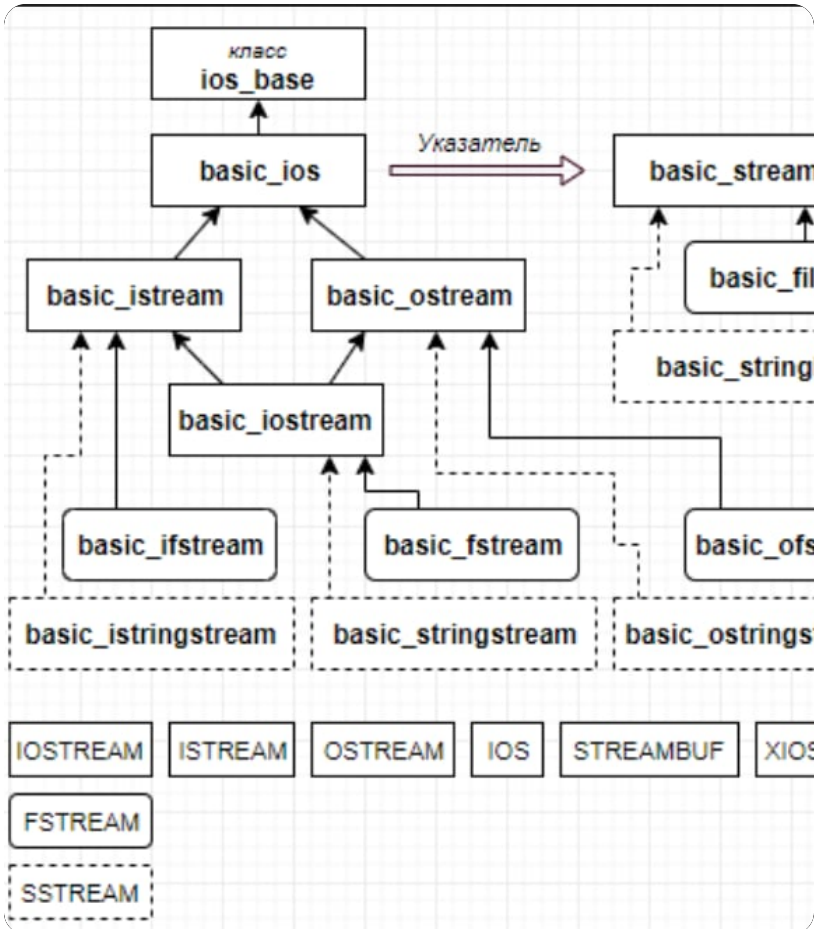
Клиенты отсылают состояние их игроков (нажатие клавиш), сервер их обрабатывает, следит за состоянием игроков (health), если кто-то из игроков 'умирает' отсылает это сообщение всем клиентам. На этом игра заканчивается.

`- GAME OVER -`

Иерархия классов

Namespace “game” (in `game_state.hpp`)

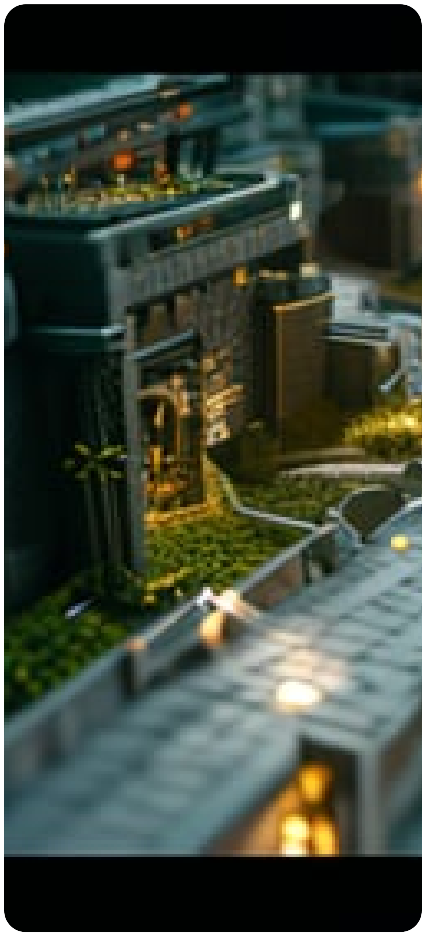
- mouse_input -
- control_struct -
- object -
- Wall -
- window_info -
- Map -
- game_state_client -
- game_state_server -



Структура проекта. Сервер

1. **Серверная часть (`TCP_server.hpp`)**

- **Сетевой модуль:**
  - `sf::TcpListener` – ожидание подключений.
  - `sf::SocketSelector` – мультиплексирование сокетов.
  - Очереди входящих/исходящих сообщений (`std::vector<sf::Packet>`).
- **Логика игры**:
  - Обработка коллизий (`resolve_collision`).
  - Обновление состояния игроков (`update_state`).
  - Синхронизация данных между клиентами (`create_messages`).



Структура проекта. Клиент

2. Клиентская часть (`client_tcp.cpp`)

- **Сетевой поток**:
  - Прием состояния игры от сервера (`network_handler`).
  - Отправка действий игрока (движение, выстрелы).
- **Рендеринг**:
  - Отрисовка игроков, стен, снарядов через SFML.
  - Камера, привязанная к герою (`sf::View`).

Структура проекта. Hpp

3. Общие модули (`game_state.hpp`, `game_objects.hpp`)

- **Состояние игры**:
  - `game_state_server` – управление игроками, стенами, пулями.
  - `game_state_client` – локальная копия состояния для рендеринга.
- **Гейм-объекты**:
  - `AABB` – система коллизий (проверка пересечений).
  - `projectile` – логика снарядов (урон, продолжительность жизни).

Движок

Программа считаывает команды с клавиатуры:

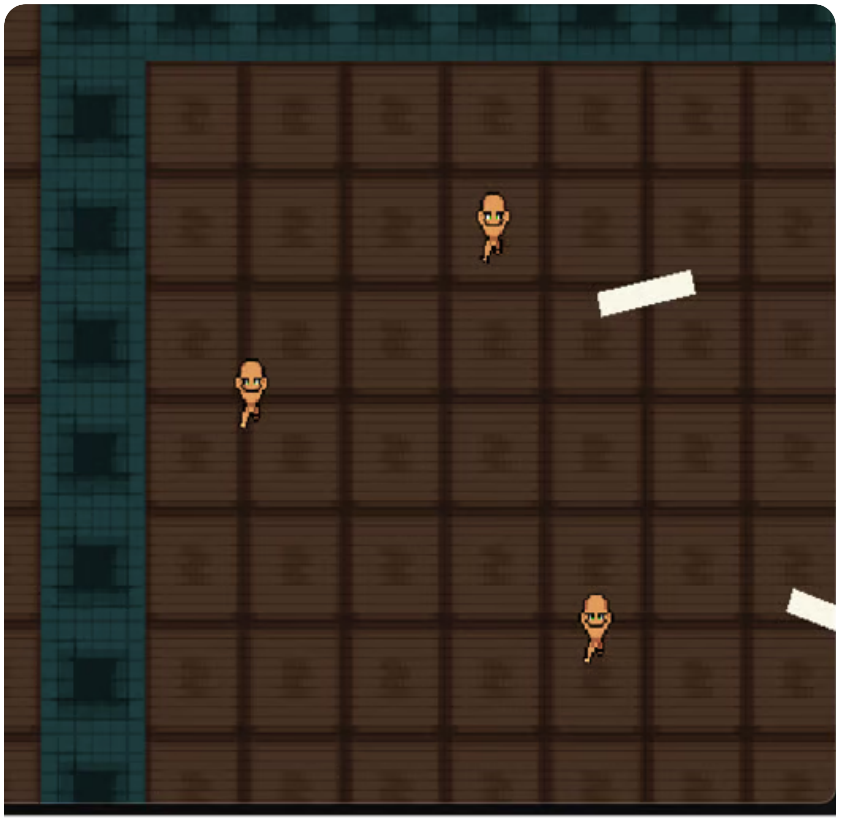
[w] - движение вверх

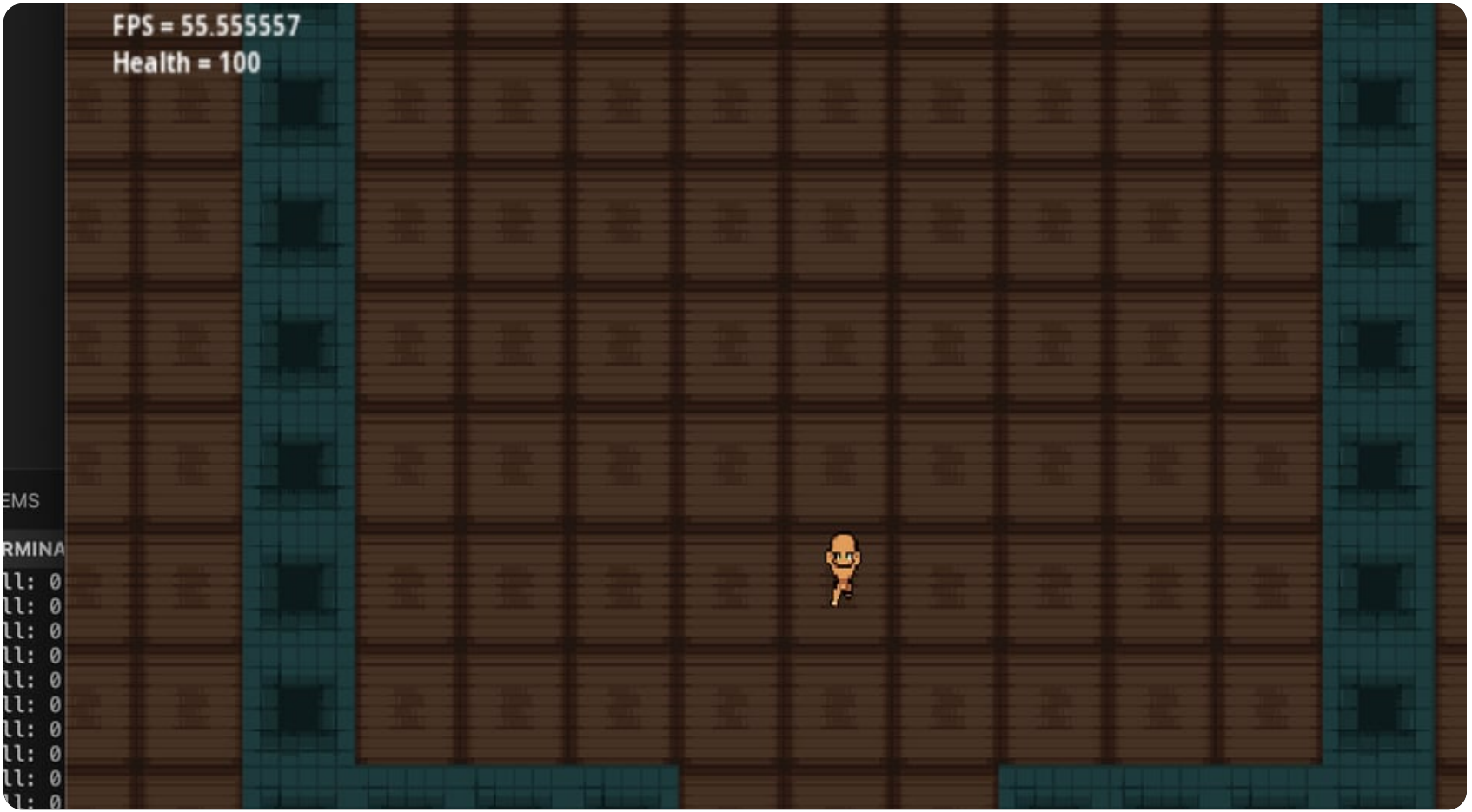
[s] - движение вниз

[a] - движение вправо

[d] - движение влево

[левая кнопка мыши] - выстрел




FPS = 55.555557
Health = 100
EMS
RMINA
ll: 0
ll: 0
ll: 0
ll: 0
ll: 0
ll: 0
ll: 0
ll: 0
ll: 0

Популярные движки на C++

- **Unreal Engine** – Blueprint + C++, используется в Fortnite, The Matrix Awakens.
- **Unity (DOTS)** – High-Performance C#/C++ (Burst Compiler).
- **Godot (C++ API)** – Опенсорс, поддержка GDScript/C++.
- **Самописные движки** – Например, Frostbite (EA), REDengine (CD Projekt).

Почему C++?

- Скорость, контроль, поддержка движков.
- Подходит для AAA и низкоуровневой разработки.
- удобно выстраивать ООП

Демо-проект:

- наш проект Top-down shooter

Зачем это нужно?

с точки зрения обучения очень полезно.

- Опыт работы с сетевыми приложениями и протоколами
- Опыт работы с библиотекой `SFML` - графика, сети, обработка клавиш
- Опыт работы с **AABB** - обработка пересечения прямоугольников
- Выстраиваивание логики игры

Дальнейшее развитие:

→ **Добавление мобов (искусственных игроков)**

→ **Добавление нового оружия**

→ **Улучшения карты и графики в целом**

→ **Взаимодействие с картой (сундуки, телепортация, двери)**

→ **Более оптимизированная работа движка**