**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО «БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**Кафедра «Информатика и программное обеспечение»**

**«Утверждаю»**

Зав. кафедрой «ИиПО», к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Копелиович Д.И.

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

**КУРСОВАЯ РАБОТА**

**по дисциплине «Объектно-ориентированное программирование»**

**Тема: «Разработка графического приложения**

**с использованием парадигмы ООП»**

**Вариант 13**

Студент гр. О-21-ИВТ-2-по-Б

\_\_\_\_\_\_\_\_\_\_\_Марченков М.И.

№ зач. книжки 21.0296

Преподаватель

\_\_\_\_\_\_\_\_\_\_\_Радченко А.О.

Защищено с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**Брянск 2023 г.**

Содержание

[Введение 3](#_Toc134069690)

[1. Аналитическая часть 4](#_Toc134069691)

[1.1. Анализ предметной области 4](#_Toc134069692)

[1.1.1. Frostpunk 4](#_Toc134069693)

[1.1.2. They Are Billions 5](#_Toc134069694)

[1.1.3. The Last Spell 6](#_Toc134069695)

[1.2. Выводы 8](#_Toc134069696)

[2. Конструкторская часть 9](#_Toc134069697)

[2.1. Архитектура проекта 9](#_Toc134069698)

[2.2. Классы на основе шаблона Singleton 9](#_Toc134069699)

[2.3. Основные методы 13](#_Toc134069700)

[3. Экспериментальная часть 17](#_Toc134069701)

[3.1. Руководство пользователя 17](#_Toc134069702)

[3.2. Тестирование приложения 18](#_Toc134069703)

[Заключение 21](#_Toc134069704)

[Список литературы 22](#_Toc134069705)

# Введение

Градостроительный симулятор ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) city-building game) — [компьютерная игра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), в которой симулируется [строительство и управление](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%B0_%D0%B8_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F) городом. Как правило, относится к [экономическим стратегиям](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D1%81%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80). В играх такого рода обычно нет определённой цели, по достижении которой игра заканчивается, целью является сам процесс обустройства города.

Целью курсового проекта является закрепление навыков в объектно-ориентированном программировании на языке С++ с помощью программного средства Visual Studio, также обязательным пунктом является использование графической библиотеки SFML.

В качестве данной курсовой работы было принято решение разработать компьютерную игру, так как при разработке игр разработчику необходимо задействовать множество различных знаний, как в математике, так и в программировании.

# Аналитическая часть

## Анализ предметной области

Компьютерная игра в жанре градостроительный симулятор – представляет собой приложение, созданное для развлечения игроков. В основе этого жанра игр заключена механика строительства разнообразных зданий и скрупулёзный менеджмент всеми видами ресурсов.

Жанр зародился в [1968](https://ru.wikipedia.org/wiki/1968) с созданием [Дугом Дайментом](https://ru.wikipedia.org/wiki/%D0%94%D1%83%D0%B3_%D0%94%D0%B0%D0%B9%D0%BC%D0%B5%D0%BD%D1%82" \o "Дуг Даймент) игры [Hamurabi](https://ru.wikipedia.org/wiki/Hamurabi" \o "Hamurabi) на языке [Фокал](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D0%BA%D0%B0%D0%BB" \o "Фокал). Игра изначально была разработана для компьютеров [PDP-8](https://ru.wikipedia.org/wiki/PDP-8), но позже была переписана на языке [BASIC](https://ru.wikipedia.org/wiki/BASIC) и стала мультиплатформенной. Именно [Hamurabi](https://ru.wikipedia.org/wiki/Hamurabi" \o "Hamurabi) стала первой игрой по управлению городом, но при этом была проще последователей жанра (Simcity), например не позволяла строить новые здания и развивать инфраструктуру.

В ходе анализа предметной области были выбраны данные игры: Frostpunk, They Are Billions, The Last Spell.

### Frostpunk

Frostpunk  – [компьютерная игра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0)  в жанре [градостроительного симулятора](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%BE%D1%81%D1%82%D1%80%D0%BE%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%81%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80) и [симулятор выживания](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D1%83%D0%BB%D1%8F%D1%82%D0%BE%D1%80_%D0%B2%D1%8B%D0%B6%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), разработанная компанией [11 bit studios](https://ru.wikipedia.org/wiki/11_bit_studios) (рис. 1). Игра включает в себя элементы [стимпанка](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D0%BC%D0%BF%D0%B0%D0%BD%D0%BA) и [альтернативной истории](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D1%8C%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F). Название игры является [неологизмом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D0%B7%D0%BC), придуманным её создателями, и получено путём объединения слов frost ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) мороз) и steampunk (стимпанк).

Игра была выпущена 24 апреля 2018 года для платформы [Windows](https://ru.wikipedia.org/wiki/Windows), 11 октября 2019 года — для платформ [Xbox One](https://ru.wikipedia.org/wiki/Xbox_One), [PlayStation 4](https://ru.wikipedia.org/wiki/PlayStation_4). Версия для платформы [macOS](https://ru.wikipedia.org/wiki/MacOS) планировалась к выходу в феврале 2019 года, но из-за технических трудностей релиз был отложен и был выпущен 24 февраля 2021 года.

В игре присутствует несколько сюжетных компаний, которые в ходе своего прохождения потребуют от игрока решить множество проблем, связанных с обеспечением всего необходимого города, который находиться под управлением игрока. Карта города статична и никак не расширяется по ходу прохождения игры, потребности житель стоит учитывать, так как их удовлетворённость влияет на моральный дух и мотивацию от которых, зависит производительность работников и настроение в городе.



Рис. . Frostpunk

### They Are Billions

They Are Billions –  [компьютерная игра](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0) в жанре [стратегии в реальном времени](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D1%8F_%D0%B2_%D1%80%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%BC_%D0%B2%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%B8), разработанная испанской студией [Numantian Games](https://ru.wikipedia.org/w/index.php?title=Numantian_Games&action=edit&redlink=1). 18 июня 2019 года игра вышла для ПК, 5 июля состоялся релиз для [Xbox One](https://ru.wikipedia.org/wiki/Xbox_One), 9 июля — для [PlayStation 4](https://ru.wikipedia.org/wiki/PlayStation_4).

Действие игры происходит в [постапокалиптическом](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%B0%D0%BF%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%BF%D1%82%D0%B8%D0%BA%D0%B0) мире в стилистике [стимпанка](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B8%D0%BC%D0%BF%D0%B0%D0%BD%D0%BA), населённом зомби (рис. 2). Целью является постройка базы, изучение карты и сбор ресурсов, и одновременная борьба с зомби. Заражённые поселения, являющиеся идеальной целью для получения ресурсов и освоения близлежащих залежей [полезных ископаемых](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B5%D0%B7%D0%BD%D1%8B%D0%B5_%D0%B8%D1%81%D0%BA%D0%BE%D0%BF%D0%B0%D0%B5%D0%BC%D1%8B%D0%B5).

Основа игры – это оборона от волн врагов о которых вы знаете заранее, для уничтожения пылищей зомби у вас представлен большой выбор построек и юнитов. Для добычи ресурсов используются только специальные здания, люди в этой игре не умирают, а пополняют ряды ходячих мертвецов. Ваше игровое пространство в ходе прохождения может увеличится если вы построите для этого специальное здание, захват территории в этой игре очень важен так как, источники ресурсов редко находятся в удобном для вас месте.



Рис. . Tre Are Billions

### The Last Spell

The Last Spell –  [тактическая ролевая игра 2021 года](https://en.m.wikipedia.org/wiki/Roguelike#Rogue-lites_and_procedural_death_labyrinths)[,](https://en.m.wikipedia.org/wiki/Tactical_role-playing_game)разработанная французской студией Ishtar Games и изданная The Arcade Crew. Игра вышла в [ранний доступ](https://en.m.wikipedia.org/wiki/Early_access) в [Steam](https://en.m.wikipedia.org/wiki/Steam_(service)" \o "Steam (сервис)) в июне 2021 года. Финальная версия игры, выпущенная 9 марта 2023 года, в тот же день, что и запуск консольных версий для [Nintendo Switch](https://en.m.wikipedia.org/wiki/Nintendo_Switch), [PlayStation 4](https://en.m.wikipedia.org/wiki/PlayStation_4) и [PlayStation 5](https://en.m.wikipedia.org/wiki/PlayStation_5).

В игре вам необходимо оборонять деревню от врагов ночью и развивать её днем, с помощью постройки новых зданий и оборонительных сооружений. В игре не так много ресурсов, но от того, как тщательно вы проработаете свою систему их добычи, очень сильно влияет на то победите вы или нет.

В отличии от двух вышеперечисленных игр в The Last Spell, жители играют не очень большую роль, в этой игре вам нужно будет всё своё внимание уделять вашим героям, их умениям и экипировке (рис. 3). Многие здания станут доступны вам только по ходу прохождения, а в начале игры у вас будет лишь ограниченный выбор из нескольких производственных и оборонительных построек.



Рис. . The Last Spell

В табл. 1 приведены результаты для сравнения:

Таблица

Сравнение программ

|  |  |  |  |
| --- | --- | --- | --- |
| Линии сравнения | Frostpunk | They Are Billions | The Last Spell |
| Система морального духа | Да | Нет | Да |
| Доступный интерфейс | Да | Да | Нет |
| Горожане умирают | Да | Да | Нет |
| Потребность горожан в еде | Да | Да | Нет |
| Размеры доступной карты | Не изменяются | Можно увеличить | Не изменяются |
| Сложность игры | Сложно | Сложно | Очень сложно |

## Выводы

Делая выводы исходя из проанализированной предметной области, можно выделить следующие пункты:

1. Создать простой и доступный интерфейс для пользователя;
2. Разработка системы морального духа и потребностей горожан в еде, жилье и лечении;
3. Создание игровой карты для размещения на ней разнообразных зданий с различными бонусами и предназначением.
4. Разработка системы сохранения игрового процесса для продолжения игровой сессии.

# Конструкторская часть

## Архитектура проекта

Данная игра написана на языке C++ с использованием графической библиотеки SFML в среде разработки Visual Studio 2022.

Архитектура данного приложения представлена на рис. 4.

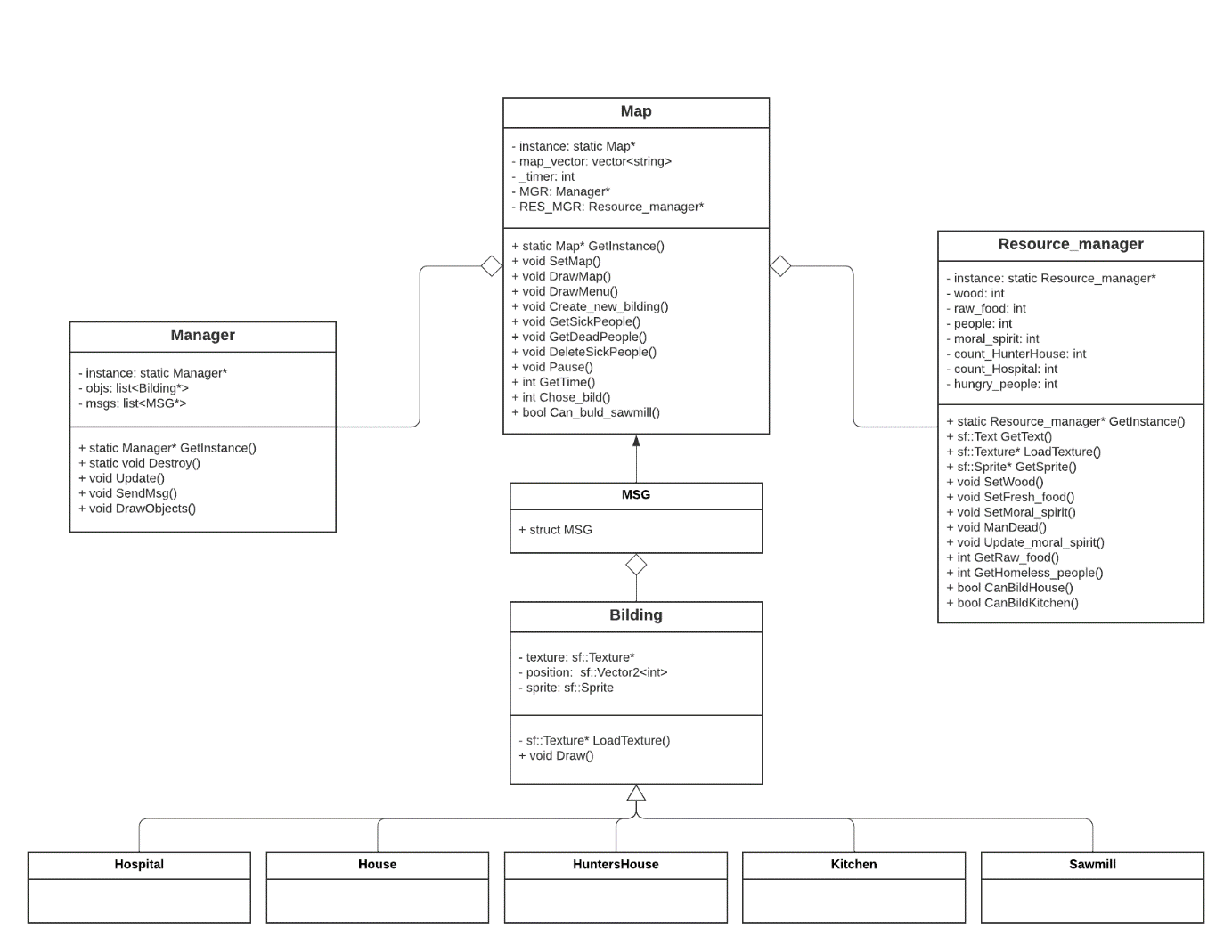


Рис. . Архитектура приложения

Файлами ресурсов являются изображения формата png, txt файл с содержимым игрового поля и шрифт формата ttf.

## Классы на основе шаблона Singleton

В данном проекте значительную часть архитектуры занимают классы типа Singleton. Класс Manager служит для создания и отрисовки здания. С помощью Resource\_manager реализована вся система взаимодействия с ресурсами в игре. Map, как следует из названия отвечает за все аспекты работы с картой. Рассмотрим каждый из них поподробнее.

Manager с помощью контейнеров типа list хранит в себе все существующие здания, добавление новых реализовано с помощью отправки и обработки сообщений. В классе присутствуют функции для обработки отправленных сообщений, а также функция с помощью которой обрисовываются все здания. Класс Manager представлен в листинге 1.

Листинг 1

Manager

class Manager

{

private:

static Manager\* instance;

std::list<Bilding\*> objs;

std::list<MSG\*> msgs;

Manager();

Manager(const Manager&);

~Manager();

public:

static Manager\* GetInstance();

static void Destroy();

void Update(float dt);

void SendMsg(MSG\* m);

void DrawObjects(sf::RenderWindow& win);

};

Класс Resource\_manager необходим для работы с ресурсами, которые добываются и используются в игре. Сам класс представляет из себя набор счетчиков и множество методов типа set и get для их изменения, также в классе есть методы типа bool, которые используются при попытке постройки новых зданий. Класс Resourse\_manager представлен в листинге 2.

Листинг 2

Resourse\_manager

class Resource\_manager

{

private:

static Resource\_manager\* instance;

int wood = 150;

int raw\_food = 100;

int fresh\_food = 50;

int people = 100;

int moral\_spirit = 90;

int count\_Sawmill = 0;

Окончание листинга 2

int count\_HunterHouse = 0;

int count\_Kitchen = 0;

int count\_Hospital = 0;

int homeless\_people = people;

int hungry\_people = 0;

public:

static Resource\_manager\* GetInstance();

sf::Text GetText(int x, int y, string text, int text\_size);

sf::Texture\* LoadTexture(const char\* filename, const sf::IntRect& src\_box);

sf::Sprite\* GetSprite(sf::Texture\* texture, int x, int y);

void SetWood(int \_wood);

void SetRaw\_food(int \_raw\_food);

void SetFresh\_food(int \_fresh\_food);

void SetMoral\_spirit(int \_moral\_spirit);

void SetHomeless\_people(int \_homeless\_people);

int GetWood();

int GetRaw\_food();

int GetFresh\_food();

int GetPeople();

int GetHomeless\_people();

int GetHungry\_people();

int GetMoral\_spirit();

int GetCount\_Hospital();

void ResourceExtraction();

bool CanBildHouse();

bool CanBildSawmill();

bool CanBildHunterHouse();

bool CanBildKitchen();

bool CanBildHospital();

void SetCountSawmill(int \_count\_Sawmill);

void SetCount\_HunterHouse(int \_count\_HunterHouse);

void SetCount\_Kitchen(int \_count\_HunterKitchen);

void SetCount\_Hospital(int \_count\_Hospital);

void Update\_moral\_spirit();

void ManDead();

};

Так как Manager и Resource\_manager классы типа Singleton, то для использования их функционала класс Map хранит в себя указатели на оба этих класса, которые инициализируются при его создании. Вся логика игры строится на взаимодействии с текстовым файлом карты. При старте игры все данные с файла считываются в класс Map, все данные о ресурсах хранятся в файле save\_resourse.txt которые также считывает класс Map и передаёт их Resource\_manager. В классе есть функции для рисования карты, меню, меню паузы и экрана поражения. Также в классе представлены функции для взаимодействия с больными людьми: их появления, выздоровление или смерть.

Не малую часть функционала класса, занимают функции для проверки возможности постройки нового здания в определённом месте на карте. Класс Map представлен в листинге 3.



Рис. 5. Пример карты в текстовом представлении

Листинг 3

Map

class Map

{

private:

static Map\* instance;

std::vector<string> map\_vector;

Manager\* MGR = nullptr;

Resource\_manager\* RES\_MGR = nullptr;

int \_timer = 24;

public:

static Map\* GetInstance();

void SetMap(const char\* filename);

void DrawMap(sf::RenderWindow& window, int days, float timer);

void DrawMenu(sf::RenderWindow& window);

void DrawSickPeople(sf::RenderWindow& window);

void Pause(sf::RenderWindow& window, int days,int timer);

Окончание листинга 3

void Create\_new\_bilding(sf::RenderWindow& window, int days, float timer);

int Chose\_bild(sf::RenderWindow& window);

bool Can\_buld\_sawmill(int x, int y);

void SetTime(int \_timer);

int GetTime();

void EndGame(sf::RenderWindow& window);

void GetSickPeople();

void GetDeadPeople();

void DeleteSickPeople(int x, int y);

};

## Основные методы

Основной игровой цикл реализован в функции main представлен в листинге 4. В начале цикла инициализируются экземпляры классов типа Singleton. В основном игровом цикле происходит обновление таймера и на основе его изменений происходят основные события в игре: изменение времени, добыча ресурсов или появление новых больных людей. Также в основном игровом цикле отслеживаются нажатия игрока на игровое поле для корректной реакции игры на действия игрока. Игрок нажатием может: изменить скорость игрового времени, поставить игру на паузу, построить новое здание или вылечить больного человека.

Листинг 4

main

int main()

{

RenderWindow window(sf::VideoMode(1216, 928), "BUILD TO SURVIVE");

window.setPosition(sf::Vector2i(300, 50));

Manager\* MGR = Manager::GetInstance();

Map\* MAP = Map::GetInstance();

Resource\_manager\* RES\_MGR = Resource\_manager::GetInstance();

MAP->DrawMenu(window);

float timer = 24;

timer = MAP->GetTime();

int days = timer / 24;

int get\_resorse = 0;

int get\_hill = 0;

int count\_spped = 1;

Clock clock;

clock.restart();

Продолжение листинга 4

while (window.isOpen())

{

timer += clock.restart().asSeconds() \* count\_spped;

if (int(timer) / 24 > days)

{

days += 1;

get\_resorse = 0;

get\_hill = 0;

RES\_MGR->Update\_moral\_spirit();

MAP->GetDeadPeople();

}

if (int(timer) % 24 == 12 && get\_resorse == 0

|| int(timer) % 24 == 16 && get\_resorse == 1

|| int(timer) % 24 == 20 && get\_resorse == 2)

{

RES\_MGR->ResourceExtraction();

get\_resorse = get\_resorse + 1;

}

if (int(timer) % 24 == 0 && get\_hill == 0 || int(timer) % 24 == 2 && get\_hill == 1

|| int(timer) % 24 == 4 && get\_hill == 2 || int(timer) % 24 == 8 && get\_hill == 3

|| int(timer) % 24 == 12 && get\_hill == 4 || int(timer) % 24 == 15 && get\_hill == 5)

{

get\_hill += 1;

MAP->GetSickPeople();

}

if (RES\_MGR->GetMoral\_spirit() == 0) MAP->EndGame(window);

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

{

window.close();

}

if (event.type == sf::Event::Resized)

{

window.setSize(sf::Vector2u(1216, 928));

}

if (event.type == sf::Event::MouseButtonPressed)

{

if (event.mouseButton.button == sf::Mouse::Left)

{

if (event.mouseButton.x >= 1024 && event.mouseButton.y >= 0

&& event.mouseButton.x <= 1088 && event.mouseButton.y <= 64 && count\_spped != 1) count\_spped -= 1;

if (event.mouseButton.x >= 1088 && event.mouseButton.y >= 0

&& event.mouseButton.x <= 1152 && event.mouseButton.y <= 64 && count\_spped != 4) count\_spped += 1;

if (event.mouseButton.x >= 960 && event.mouseButton.y >= 0

&& event.mouseButton.x <= 1024 && event.mouseButton.y <= 64) MAP->Pause(window, days, timer);

if (event.mouseButton.x >= 1152 && event.mouseButton.y <= 64)

Окончание листинга 4

MAP->Create\_new\_bilding(window, days, timer);

MAP->DeleteSickPeople(Mouse::getPosition(window).x, Mouse::getPosition(window).y);

}

}

}

window.clear();

MGR->Update(clock.restart().asSeconds());

MAP->DrawMap(window, days, timer);

MGR->DrawObjects(window);

MAP->DrawSickPeople(window);

window.display();

}

return 0;

}

Метод, с помощью которого каждый день обновляется счетчик морального духа представлен в листинге 5. Если после всех вычислений моральный дух примет значение меньше 0, игрок проиграет.

Листинг 5

Update\_moral\_spirit

void Resource\_manager::Update\_moral\_spirit()

{

if (moral\_spirit + (people \* 0.1) > 100) moral\_spirit = 100;

else moral\_spirit = moral\_spirit + (people \* 0.1);

if (raw\_food > 25 \* count\_Kitchen)

{

raw\_food -= 25 \* count\_Kitchen;

fresh\_food += 50 \* count\_Kitchen;

}

if (fresh\_food > people) fresh\_food - people;

else {

hungry\_people = people - fresh\_food;

fresh\_food = 0;

if (moral\_spirit - hungry\_people \* 0.25 < 0) moral\_spirit = 0;

else moral\_spirit -= hungry\_people \* 0.25;

}

if (moral\_spirit - homeless\_people \* 0.2 < 0) moral\_spirit = 0;

else moral\_spirit -= homeless\_people \* 0.2;

if (moral\_spirit + (10 \* count\_Hospital) > 100) moral\_spirit = 100;

else moral\_spirit + (10 \* count\_Hospital);

}

С помощью метода, представленного на листинге 6 реализовано появление больных людей. Чем в поселении больше домов, тем выше шанс появления больного человека, но наличии больниц уменьшает эти шансы.

Листинг 6

GetSickPeople

void Map::GetSickPeople()

{

int number\_house = (RES\_MGR->GetPeople() - RES\_MGR->GetHomeless\_people()) / 5;

if (RES\_MGR->GetCount\_Hospital() != 0) number\_house = number\_house / RES\_MGR->GetCount\_Hospital();

bool sick\_people = false;

for (int i = 0; i < number\_house; i++) {

int num = rand() % 100;

if (num <= 8)

{

sick\_people = true;

number\_house = i;

break;

}

}

int number\_house\_for\_sick = -1;

bool have\_sick\_people = false;

if (sick\_people) {

for (int i = 0; i < map\_vector.size() - 4; i++)

{

for (int j = 0; j < map\_vector[0].length(); j++)

{

if (map\_vector[i][j] == '1')

{

number\_house\_for\_sick += 1;

if (map\_vector[i + 1][j + 1] != 's' and !have\_sick\_people and number\_house\_for\_sick == number\_house)

{

map\_vector[i + 1][j + 1] = 's';

have\_sick\_people = true;

}

}

}

}

}

}

# Экспериментальная часть

## Руководство пользователя

После запуска приложения игрок может начать игру нажав на кнопку Начать новую игру или продолжить уже существующую через кнопку Продолжить игру, Выход соответственно закрывает приложение. Если нажать на кнопку Обучение или об авторе вы перейдете в окна с соответствующей информаций (рис. 6).





Рис. 6. Меню приложения и раздел Обучение

## Тестирование приложения

В первую очередь необходимо проверить корректность работы игры при выборе продолжить игру. Все данные с файла о количестве ресурсов, населения и времени загрузились верно. Также корректно загрузились все данные о карте: наличие в определённых домах больных, количество и расстановка зданий. Исходя из данного теста, можно сделать вывод, что продолжение игры работает корректно (рис. 7).



Рис. 7. Загрузка игрового процесса

Во вторую очередь необходимо проверить корректность обработки нажатия на левую клавишу мыши. Для теста попробуем нажать на кнопку пауза и сохранить игровой процесс. При нажатии на кнопку паузы игровой процесс останавливается, а мы попадаем на экран паузы. После нажатия на кнопку сохранить и выйти игра закрывается, а данные об игре в файлах с сохранениями перезаписываются (рис. 8). Для второго теста попробуем выбрать новое здание для постройки. После нажатия на кнопку выбор нового здания открывается меню с списком зданий. Попробуем постоять дом лесоруба в неправильном месте. После выбора здания для постройки на карте появляется силуэт будущего здания. Если хотя бы одна из клеток загорелась красным здание в этом месте построить невозможно (рис. 9). В результате проделанных тестов, можно сказать, что обработки нажатия на левую клавишу мыши – корректна.





Рис. 8. Меню паузы и выбора здания для постройки

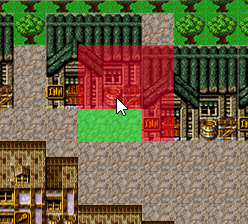




Рис. 9. Правильный и неправильный выбор места для постройки дама лесоруба

# Заключение

В качестве данной темы курсовой работы была выбрана разработка игры в жанре градостроительный симулятор на языке программирования C++ с использованием графической библиотеки SFML. Проведя анализ предметной области, были найдены главные особенности этого жанра и использованы в данном курсовом проекте.

Была разработана игра, соответствующая функциональным требованиям. В процессе написания программы были получены практические навыки разработки архитектуры игры в жанре градостроительный симулятор, различных механик для ее создания, а также графической библиотеки SFML и использования объектно-ориентированного программирования.

Данное приложение было протестировано по основным механикам, особенно важных ошибок в ходе данного процесса не выявлено. Проект можно считать успешно законченным.

В дальнейшем планируется добавить новые здания, карту мира и возможность расширять карту.

# Список литературы

1. Понамарев, В. Программирование на C++/C# в Visual Studio .NET 2003 / В. Понамарев. - М.: БХВ-Петербург, 2015. – 917 c.
2. Бьерн Страуструп Дизайн и эволюция языка С++ / Бьерн Страуструп. - М.: ДМК Пресс, 2014. - 446 c.
3. Пахомов, Борис C/C++ и MS Visual C++ 2012 для начинающих / Борис Пахомов. - Москва: СИНТЕГ, 2015. – 51 с.
4. Израилев, В.Я. Программирование на языке Си: Объектно- ориентированное программирование [Текст] + [Электронный ресурс]: практикум / В.Я. Израилев – Брянск: БГТУ, 2018. – 100 с.
5. Как настроить библиотеку SFML в Visual Studio [Электронный ресурс]. Режим доступа: https://habr.com/ru/post/703500/ (Дата обращения 30.12.2022)
6. Уроки по SFML [Электронный ресурс]. Режим доступа: https://kychka-pc.ru/ (Дата обращения 30.12.2022)