Astana IT University

Department of Intelligent Systems and Cybersecurity

Educational Practice Project Report on **2D Game**

Students Name: **Zhandarbek Saginov, Madiyar Kaliyev,**

, Dias Merekeev, Ali Orymbayev

Supervisor Name: **Balzhan Azibek**

November 2021

Abstract

After examining libraries for game development, the team concluded that SFML is the only option for newcomers to creation of a game. The controls in a 2D game are very simple that does not require much effort, moreover, it performs the function of a game, holding players' attention to playing it. Overall, the development was a very time-consuming process, but in the end, the team coped with the obstacles by constant self-studying from the available materials on the internet. This article will analyze each game process in detail to fully understand this library, in the following parts each step is described in detail for a complete acquaintance with the project creation.

Introduction

The topic which the Team has chosen is a 2D Game Development that is based on C++' Simple and Fast Multimedia Library. Interaction of a user and a game should be straightforward leading to enjoying and inducing various emotions that will hold the player's attention while playing it. Our Team took it all into account while creating the game. The main objective of this project is to attract a wider range of audiences and entertain them by fast downloading, a little storage, moreover, not requiring the fast computers systems to play it. The game is called "Caveman" and is based on the story of an old man who for some reason remained under the mine. The name of the game is not randomly selected. It has a connection with the plot of the game and the map that the main character must pass to complete the game. The next session includes a literature review, which will be more theoretical**,** oppositely, the part of the methodology that follows the first is more practical and contains detailed series of steps and actions for each phase of the project life cycle, following in the results session for better the visualization of our project screenshots will be shown of the "Caveman" game, after which these tutorial screenshots shall be described in simple language for a clear understanding of each step by the user.

Methodology

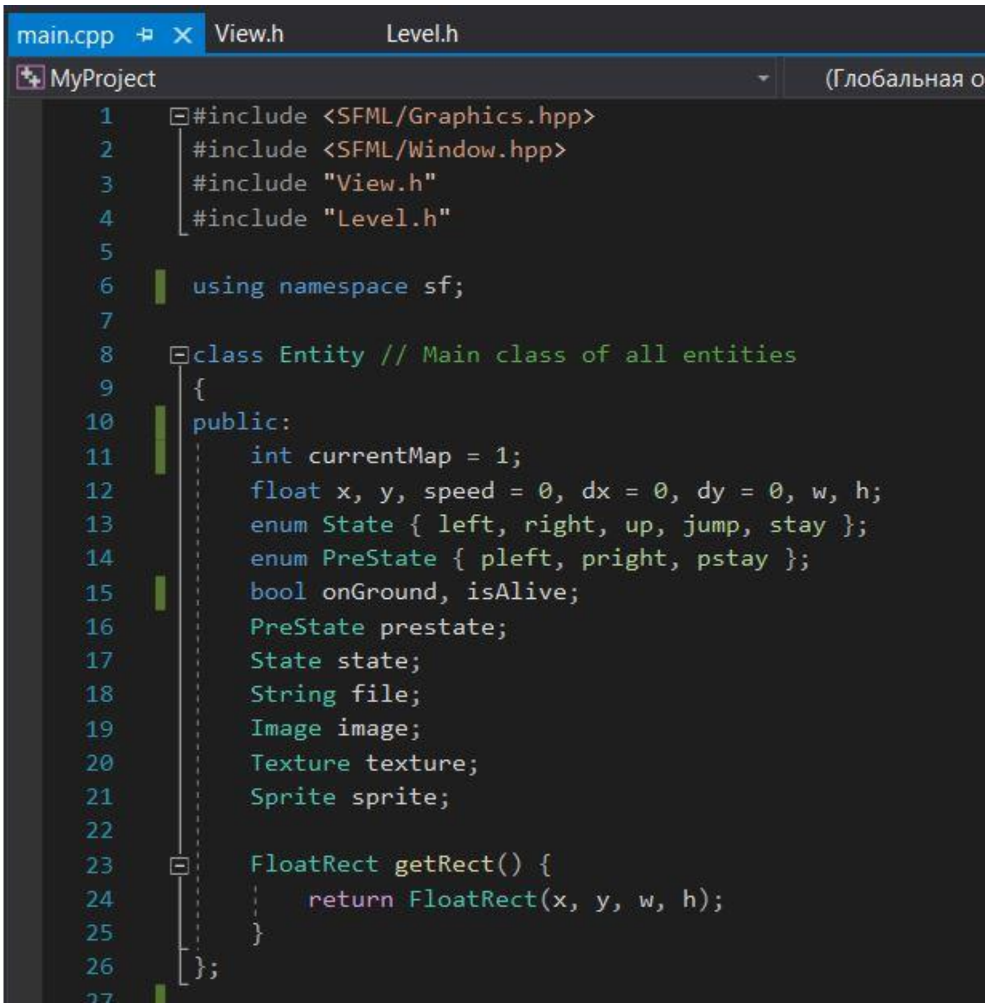
Developing a project is a rigid process that takes patience, discipline, and creativity. Before the final theme of our project, opinions were heard from each participant, and finally, 2D game development was chosen as the ultimate topic among proposed suggestions. The further step was to select a platform that will put planning into realization and by the team's research, C++' **Simple and Fast Multimedia Library (SFML)** was adopted as the primary platform of developing a game. Map's design in which the main character will pass the game created by the "Tiled" program. It will change according to the level to express different environments and complexity to make the transition noticeable.

SFML C ++ is a software development multimedia library designed to make game creation easy. It has a variety of methods that are related to types of media such as audio, fonts, motions, etc. The development on SFML is quick and comparably easy to understand for novice developers. SFML is applied widely in making game engines and games for developers. The advantage is window creation that assists programmers to see an instant result of coding. This library as previously stated is simple to use and portable Application Programming Interface (API) that’s written in C++ object-oriented programming language.

It can be used as a windowing system for the Open Graphics Library (OpenGL) or as a multimedia library full of various functions that can help you create video games or multimedia software. SFML is based on C ++, nevertheless, bindings (a third-party library or operating system service that is not native to that language). have been developed for other languages like C, C#, Java, Python, etc. SFML is available on Windows (10,8,7 and XP), macOS, and Linux. It runs on both 64- and 32-bit systems

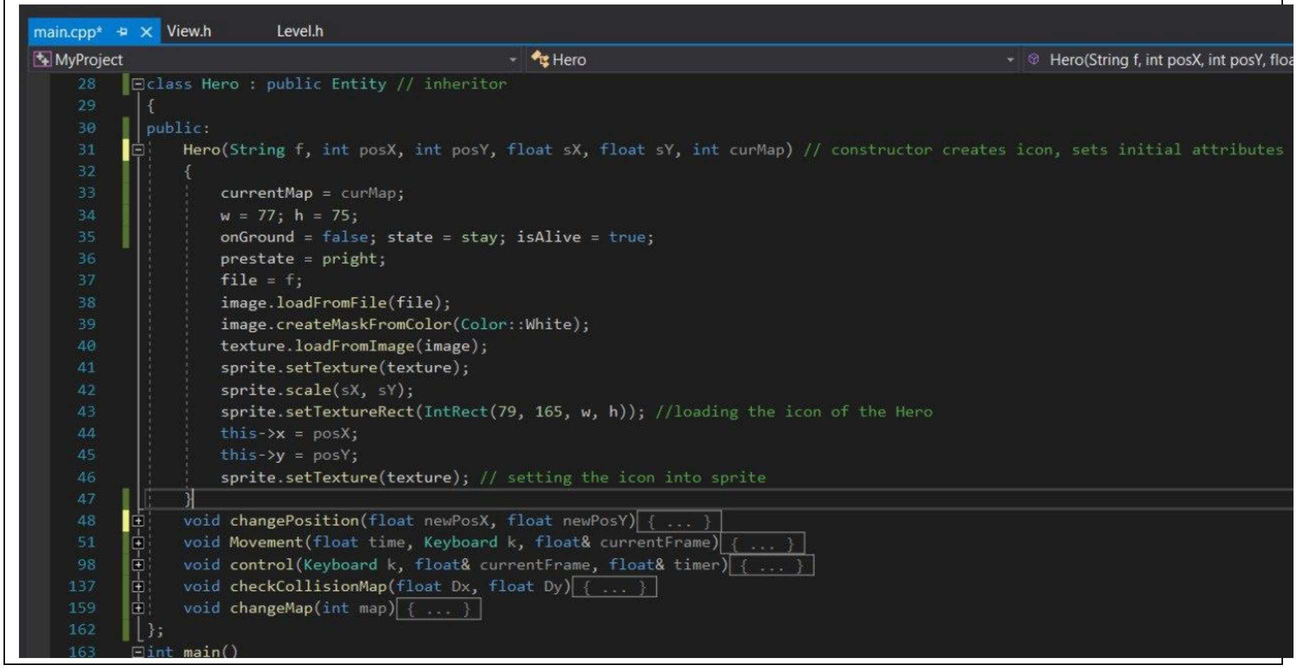
Explanation of code:

"SFML/Graphics.hpp" and "SFML/Windows.hpp" allow us to use various "classes" with different useful functions and methods, for example, "Texture" can take files from device memory and implement them into the code. “View. h" and "Level. h" its separated code, which later will be discussed in the documentation. Class "Entity" is the prominent class for all moving objects, such as the Main character, Enemies etc.



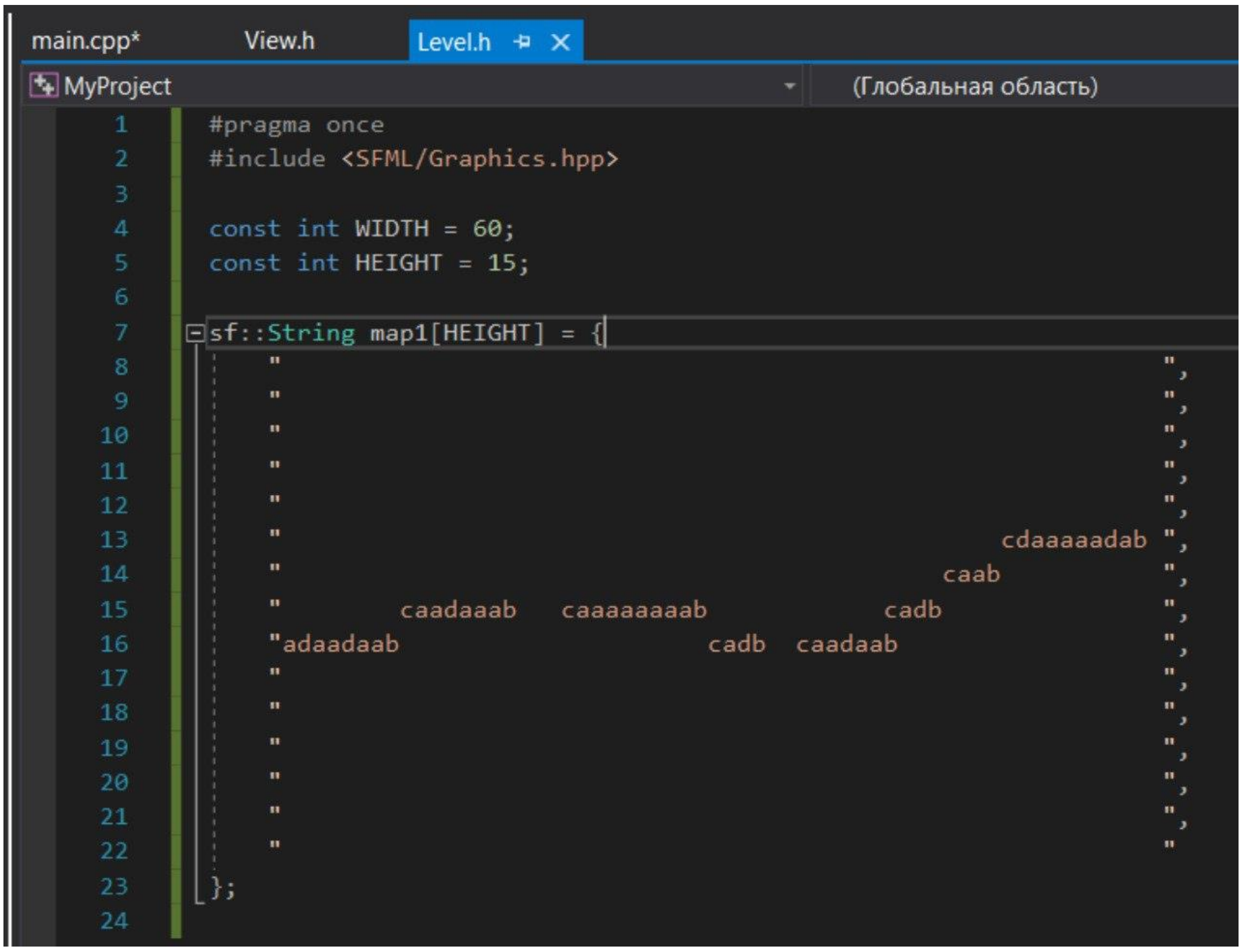
```cpp
#include <SFML/Graphics.hpp>
#include <SFML/Window.hpp>
#include "View.h"
#include "Level.h"

using namespace sf;

class Entity // Main class of all entities
{
public:
    int currentMap = 1;
    float x, y, speed = 0, dx = 0, dy = 0, w, h;
    enum State { left, right, up, jump, stay };
    enum PreState { pleft, pright, pstay };
    bool onGround, isAlive;
    PreState prestate;
    State state;
    String file;
    Image image;
    Texture texture;
    Sprite sprite;

    FloatRect getRect() {
        return FloatRect(x, y, w, h);
    }
};
```

Class 'Hero' creates the main Hero, with its functions and methods. This class inherits everything from the prominent class 'Entity’. It has a constructor, which sets up initial attributes. And divergent functions, which are responsible for movement, animation, control checking collisions with map.



```cpp
class Hero : public Entity // inheritor
{
public:
    Hero(String f, int posX, int posY, float sX, float sY, int curMap) // constructor creates icon, sets initial attributes
    {
        currentMap = curMap;
        w = 77; h = 75;
        onGround = false; state = stay; isAlive = true;
        prestate = pright;
        file = f;
        image.loadFromFile(file);
        image.createMaskFromColor(Color::White);
        texture.loadFromImage(image);
        sprite.setTexture(texture);
        sprite.scale(sX, sY);
        sprite.setTextureRect(IntRect(79, 165, w, h)); //loading the icon of the Hero
        this->x = posX;
        this->y = posY;
        sprite.setTexture(texture); // setting the icon into sprite
    }
    void changePosition(float newPosX, float newPosY){ ... }
    void Movement(float time, Keyboard k, float& currentFrame){ ... }
    void control(Keyboard k, float& currentFrame, float& timer){ ... }
    void checkCollisionMap(float Dx, float Dy){ ... }
    void changeMap(int map){ ... }
};
int main()
```

Header file "Level. h" contains simple maps in the game. The maps are 60 x 15 Tiles, each Tile is 32x32 pixels. Every symbol represents the textures, that should be placed on a specific point.



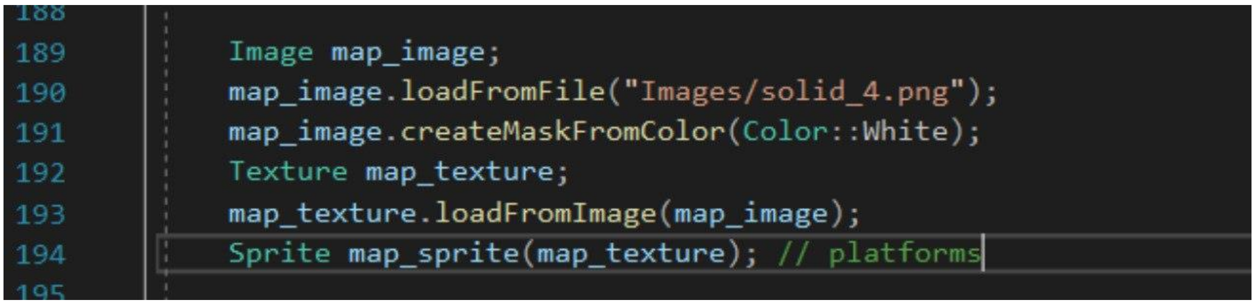
main.cpp*   View.h   Level.h

MyProject   (Глобальная область)

```cpp
#pragma once
#include <SFML/Graphics.hpp>

const int WIDTH = 60;
const int HEIGHT = 15;

sf::String map1[HEIGHT] = {
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                 cdaaaaadab ",
    "                                             caab           ",
    "        caadaaab   caaaaaaaab            cadb               ",
    "adaadaab                     cadb  caadaab                  ",
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                            ",
    "                                                            "
};

```

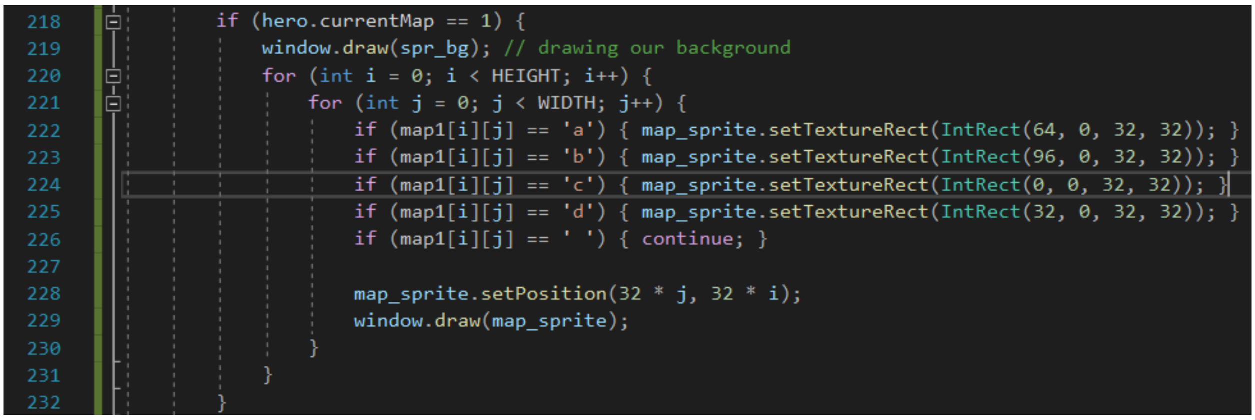
In the "main.cpp”:

Here tiles are loaded into the texture



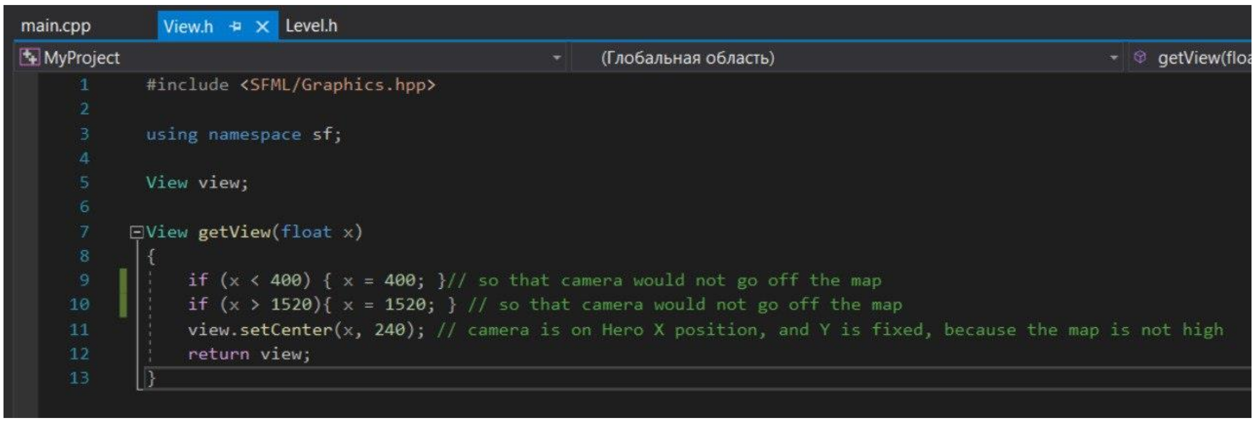
```cpp
    Image map_image;
    map_image.loadFromFile("Images/solid_4.png");
    map_image.createMaskFromColor(Color::White);
    Texture map_texture;
    map_texture.loadFromImage(map_image);
    Sprite map_sprite(map_texture); // platforms
```

Maps are being drawn in the window:



```
if (hero.currentMap == 1) {
    window.draw(spr_bg); // drawing our background
    for (int i = 0; i < HEIGHT; i++) {
        for (int j = 0; j < WIDTH; j++) {
            if (map1[i][j] == 'a') { map_sprite.setTextureRect(IntRect(64, 0, 32, 32)); }
            if (map1[i][j] == 'b') { map_sprite.setTextureRect(IntRect(96, 0, 32, 32)); }
            if (map1[i][j] == 'c') { map_sprite.setTextureRect(IntRect(0, 0, 32, 32)); }
            if (map1[i][j] == 'd') { map_sprite.setTextureRect(IntRect(32, 0, 32, 32)); }
            if (map1[i][j] == ' ') { continue; }

            map_sprite.setPosition(32 * j, 32 * i);
            window.draw(map_sprite);
        }
    }
}
```

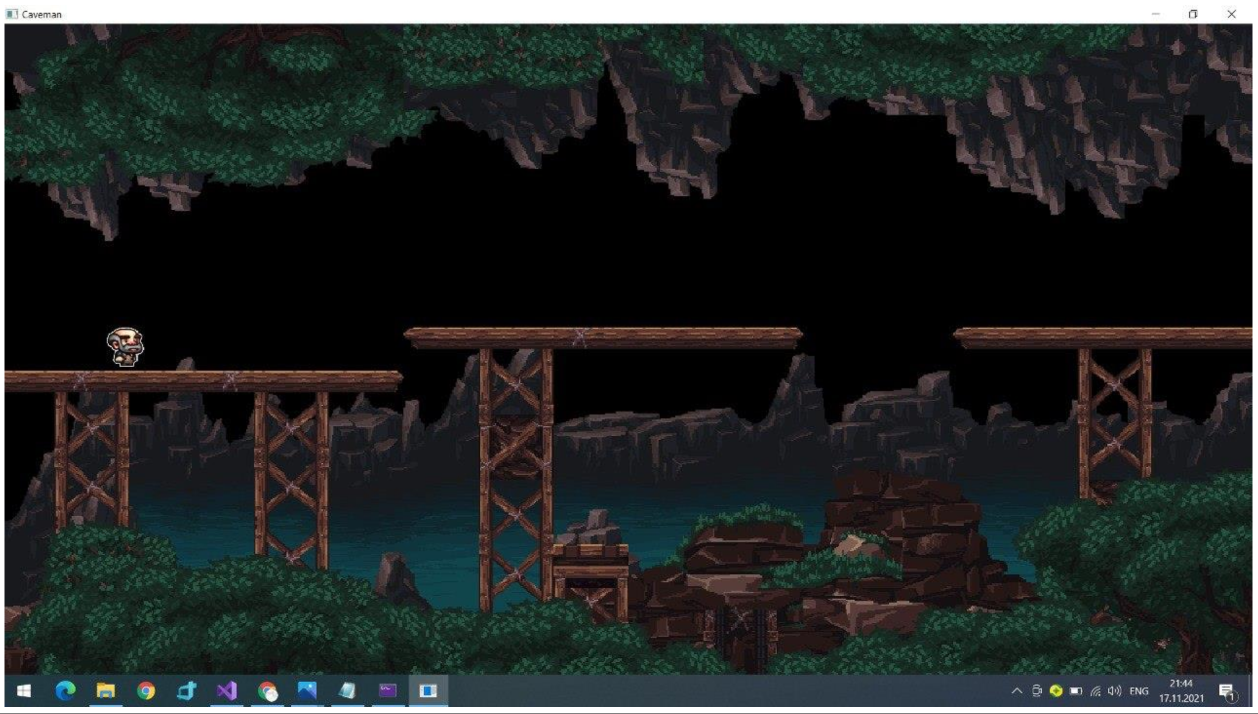
Header file "View. h" is responsible for setting the focus of the camera on the main hero. The camera focus is only moving horizontally according to the main character's X position.



main.cpp View.h Level.h
MyProject (Глобальная область) getView(floa

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

using namespace sf;

View view;

View getView(float x)
{
    if (x < 400) { x = 400; }// so that camera would not go off the map
    if (x > 1520){ x = 1520; } // so that camera would not go off the map
    view.setCenter(x, 240); // camera is on Hero X position, and Y is fixed, because the map is not high
    return view;
}
```

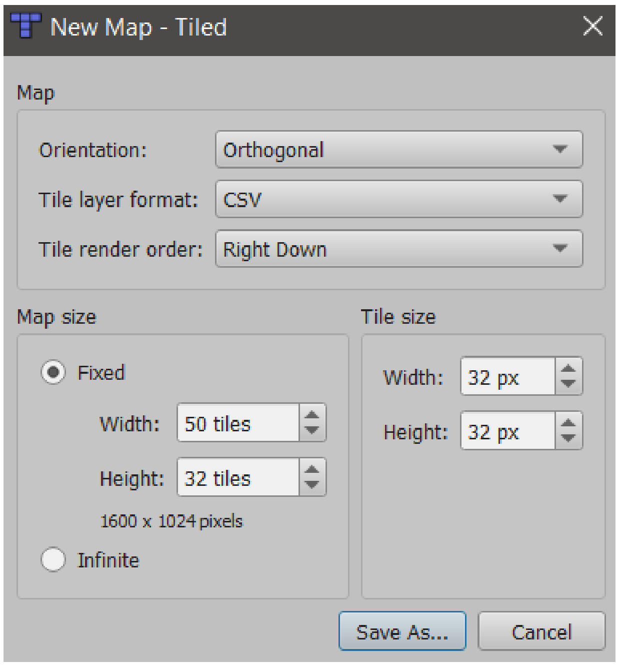
Result:



Tiled is a 2D-level editor that is used for developing the visuals of a game. Its main functionality is to edit tile maps of different forms, but it also supports free image placement as well as powerful ways to annotate your level with additional information. Tiled focuses on overall flexibility while trying to be understandable. Tiled is very convenient in terms of placing tiles in any part of the created map within one project. In addition, it is possible to insert inter-tile sprites. All this makes it possible to comfortably create levels, maps for 2d games.

The process of creation a map via Tiled:

Here we select the main parameters of the card. The default orientation is orthogonal, but there is also an isometric, hexagonal orientation. The layer format for new versions is CSV, but if you need XML, then this option is selected after creating the map. Drawing in many games starts at the bottom right corner, and this can also be changed. The map size has two options - fixed and infinite. And the size of the tiles has been adjusted to fit the pixels.


New Map - Tiled

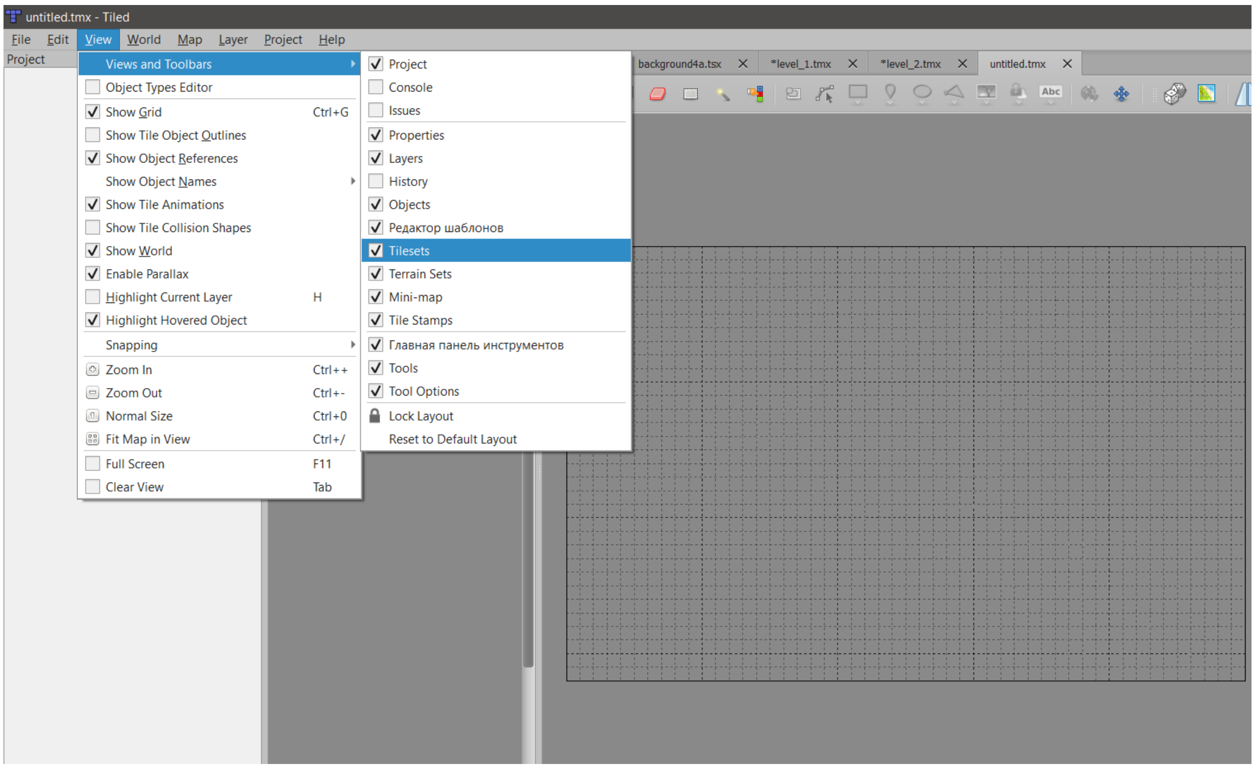
Map

Orientation: Orthogonal
Tile layer format: CSV
Tile render order: Right Down

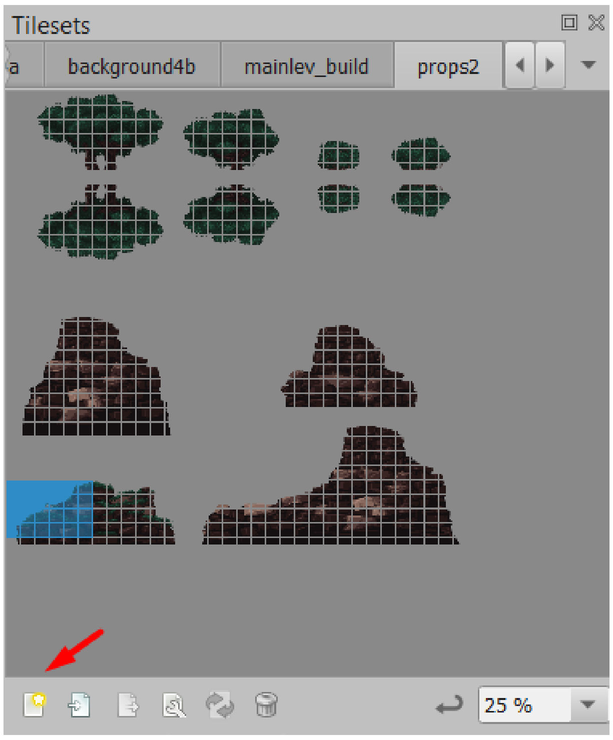
Map size
(● Fixed)
Width: 50 tiles
Height: 32 tiles
1600 x 1024 pixels
○ Infinite

Tile size
Width: 32 px
Height: 32 px

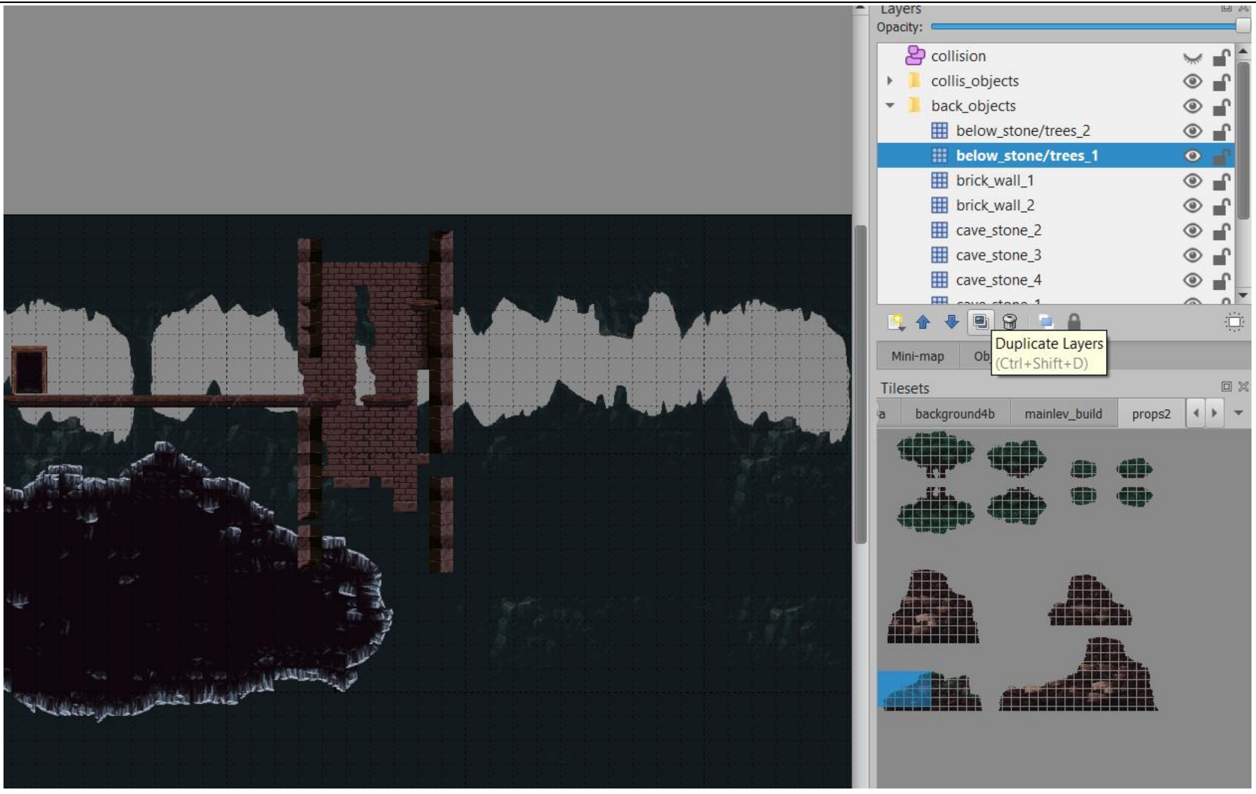
After creating a new map, we need a building material. Select "View" -> "Views and Toolbars" -> "Tile Set".



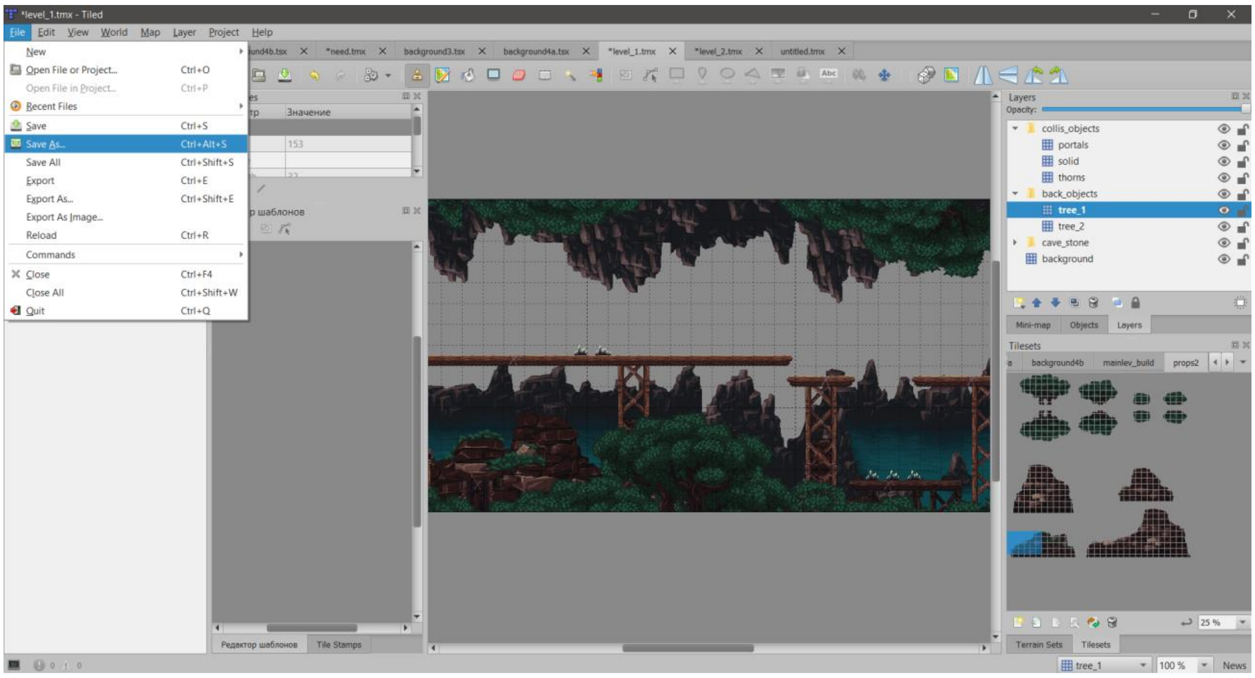
As we can see here, you can see the “tile set” loaded into the map. A new set of tiles can be downloaded through the marked folder.



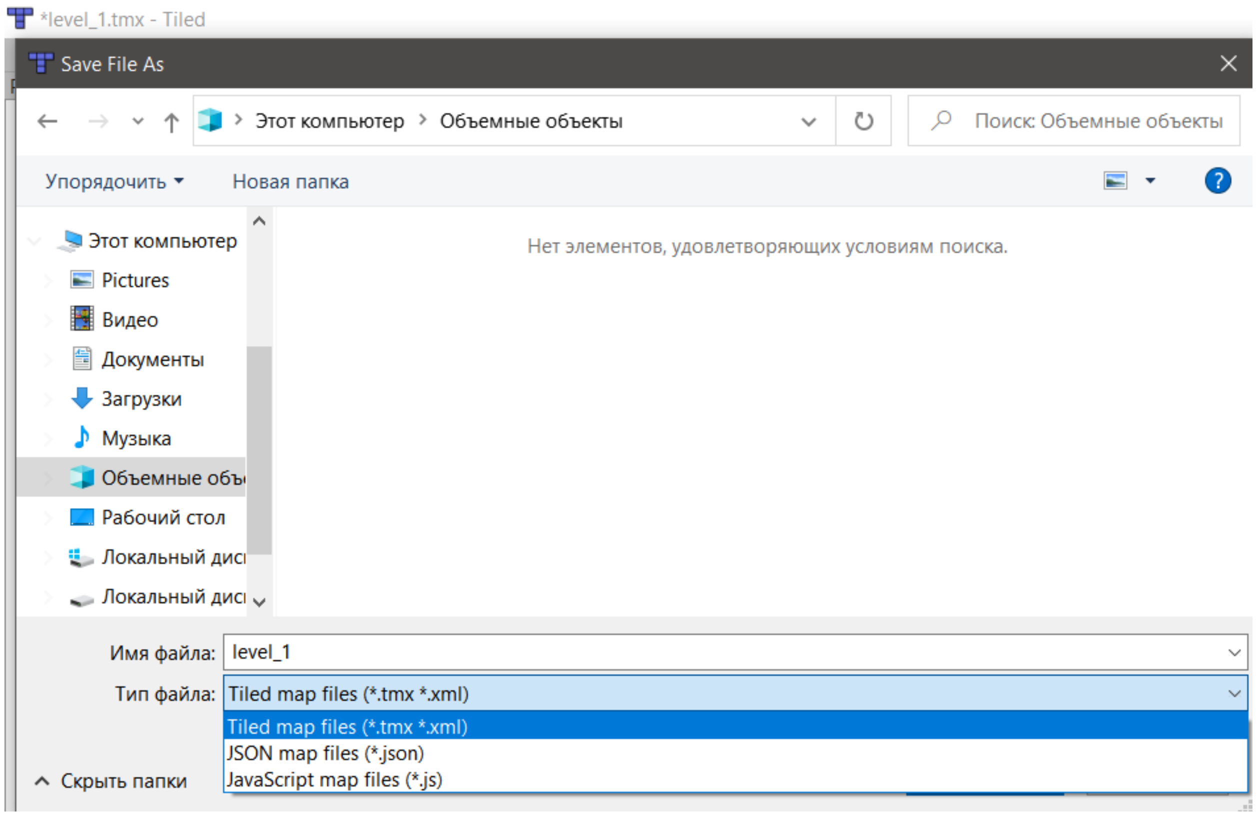
Then we start working on the layers. Layers are needed for convenience when designing a map.



If you have finished the whole process, it remains to save the project. And then we will have a choice.



We can save the map in the "tmx, xml", "json", "js" format. And it is very convenient for programmers if the card is added to the code itself.



Нет элементов, удовлетворяющих условиям поиска.

Имя файла: level_1

Тип файла: Tiled map files (*.tmx *.xml)

Tiled map files (*.tmx *.xml)
JSON map files (*.json)
JavaScript map files (*.js)

Limitation of resources (enough knowledge)

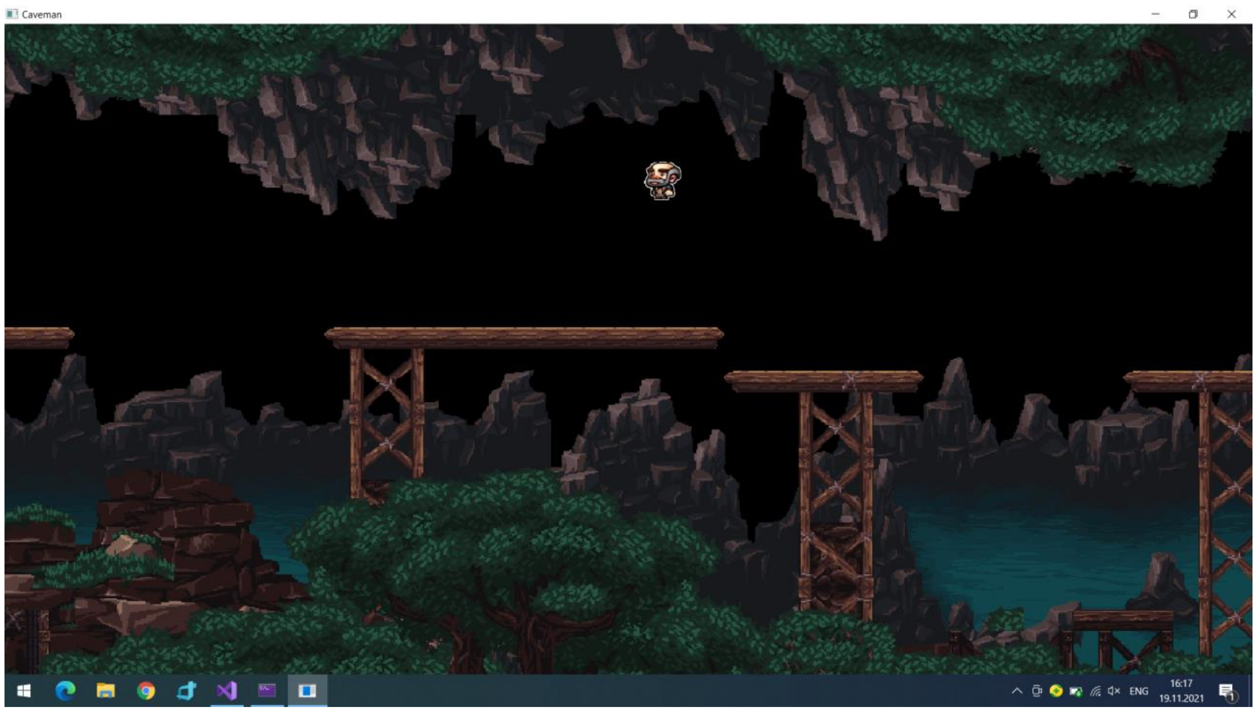
“Caveman” is not a flawless game, so it has some limitations that could be improved in future updates. The first is the lack of many levels that players go through, which subsequently affects the duration time of the game. To maintain the interest of users in the game, there must be at least 15 levels, which differ in difficulty, so that the players do not get bored.

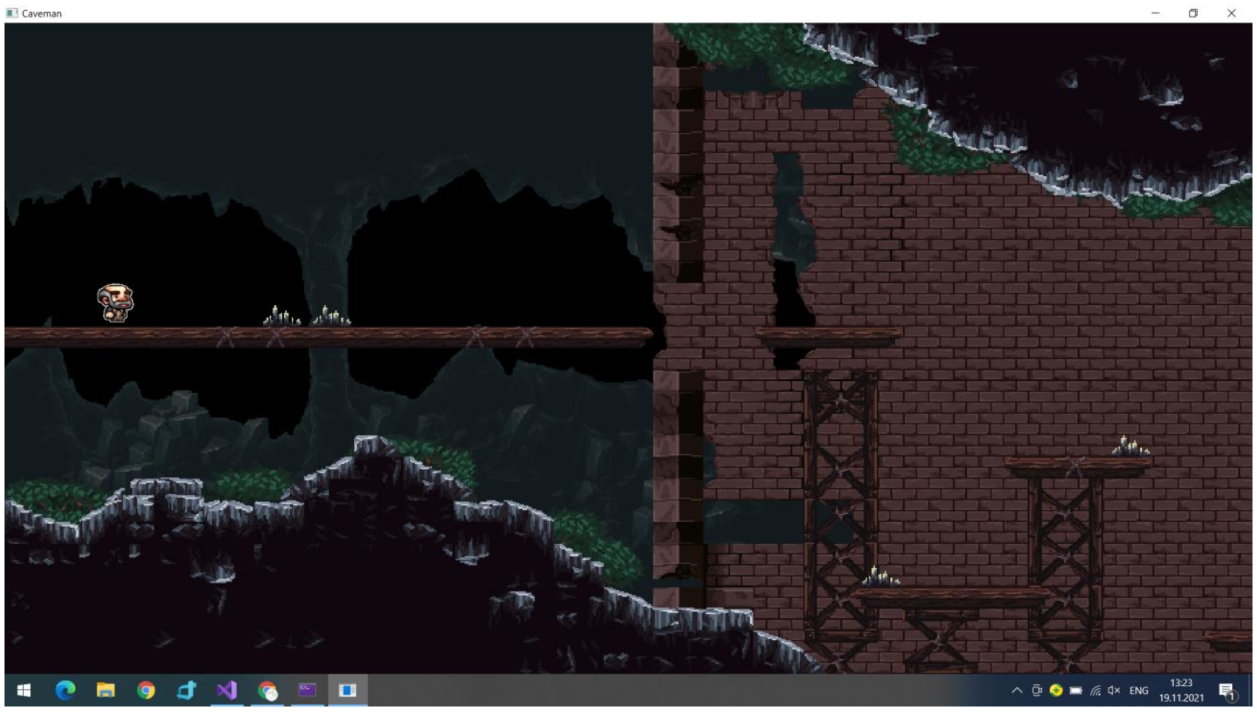
Another drawback of the game is that the map was created in the code itself. Objects that interact with the character are written to the "level.h" file. The rest of the map parameters are added as a background, and this is unprofessional in terms of creating maps. We have not one layer left, but several. In the future, with the help of an experienced programmer, we are thinking of fixing this defect.

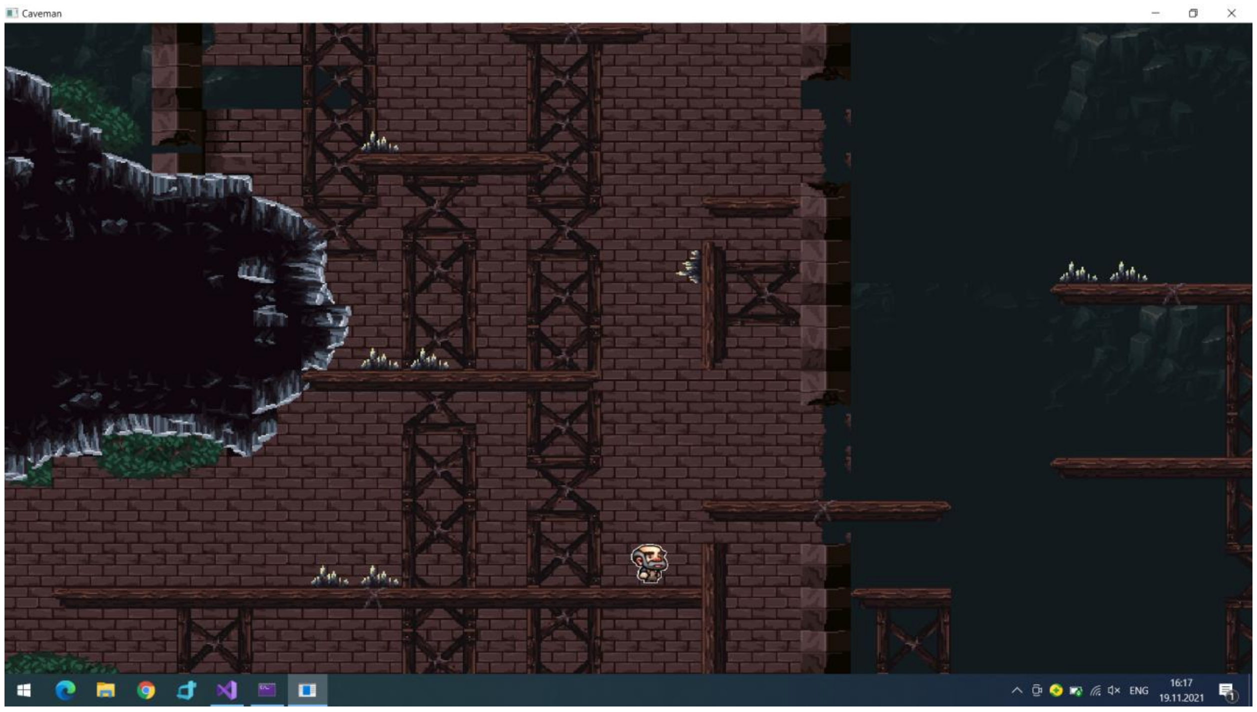
Weekly Progress

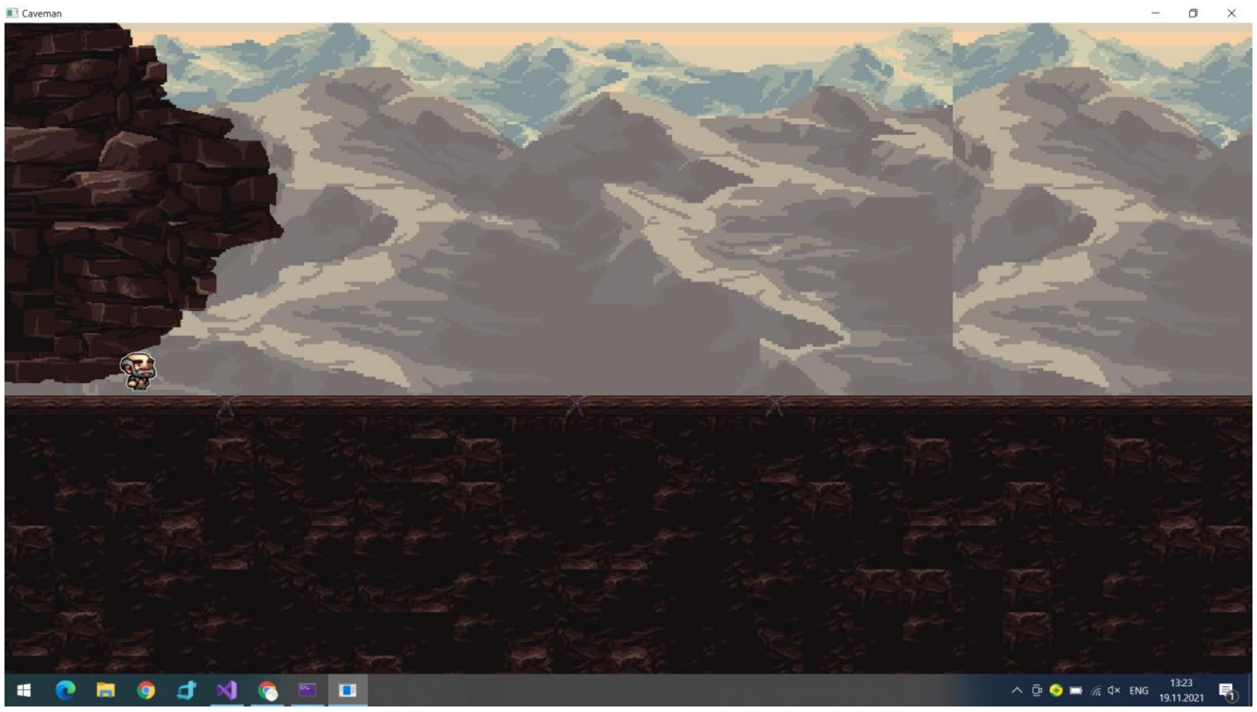
Date	What you have done
8.11.2021	Discussed a topic of project
9.11.2021	Wrote main class “Entity” and” Hero”
12.11.2021	Animation and gravity
13.11.2021	Background of Level 1
14.11.2021	Added collision and implemented the first level to the game
15.11.2021	Background of Level 2&3
17.11.2021	Documentation
18.11.2021	Release of project

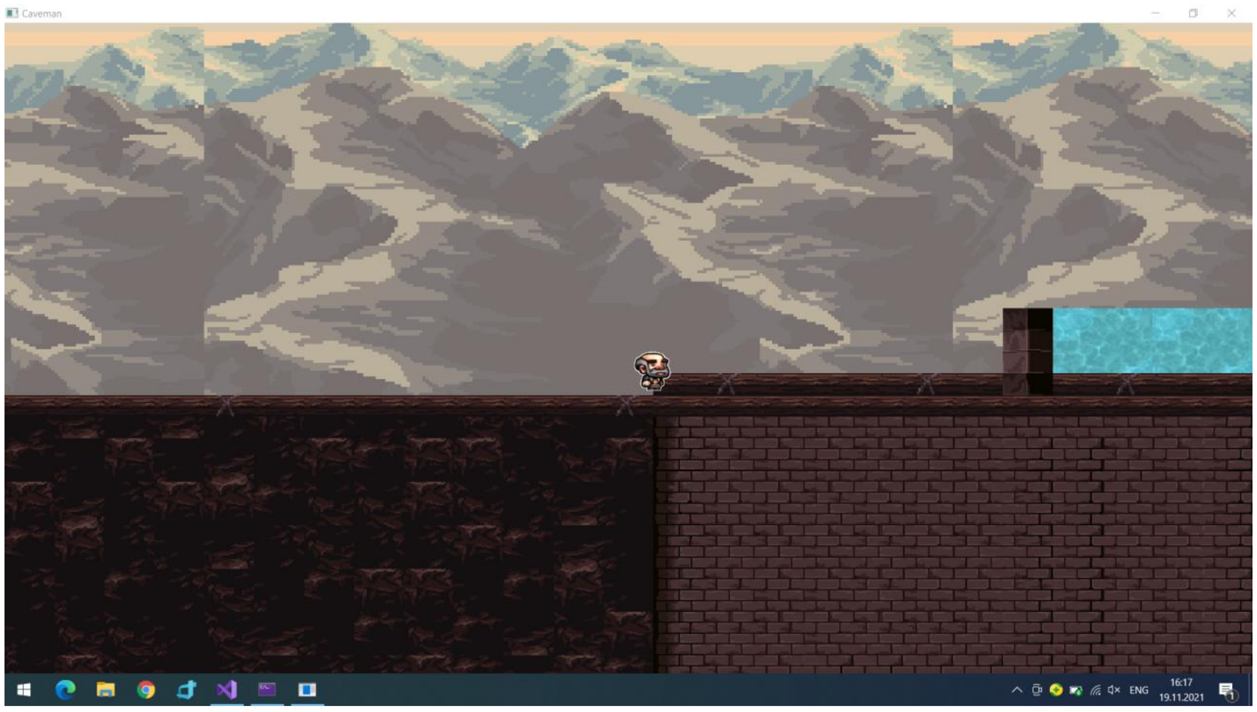
Results




Caveman
13:23
19.11.2021
ENG


Caveman
16:17
19.11.2021
ENG





As a result, it turned out to create a character with its own mechanics, with movements, with physics. It turned out to create a location with a landscape and obstacles in the style of caves.

Discussion

The development of the project taught us a lot of skills such as creativity, time and team management, critical thinking, etc. improved significantly during creation. Each participant made his own equal contribution to this project, for instance, if one were removed from the group, the project would not have been implemented, because everyone has their own role that moves closer to the implementation of the game. It is worth paying attention to the self-study because each member taught the platform separately from scratch in order to develop a project and apply theory successfully in practice.

Conclusion

In conclusion, the team was able to implement a project based on plans and ideas and with the help of the SFML multimedia library in C++, and a cross-platform open editor of tile maps for games named "Tiled'. The game is raw, so it's not flawless. The team started everything from scratch, studied new development environments for themselves, did things that had not been done before from the knees.

References list

Eden, M. (2020, May 6). *2D Vs 3D Games: Differences, Benefits and Costs*. Melior Games.

https://meliorgames.com/game-development/2d-vs-3d-games-differences-benefits-and-costs/

Frequently Asked Questions (SFML / Learn). (2021). SFML.

https://www.sfml-dev.org/faq.php#grl-whatis

Introduction — Tiled 1.7.2 documentation. (2021). Tiled.

https://doc.mapeditor.org/en/stable/manual/introduction/#setting-up-a-new-project