Tower Defense Game Project - 1st Report

Theodore Petrick Reimmer and Huseynzade Huseyn

September 26, 2025

Project Description

The aim of this project is to develop a simple Tower Defense game using C++. The game will allow players to place towers to defend against waves of enemies. At this stage, the focus is on planning the project and setting up a clean folder structure for future development.

Project Start

We officially started our project “Tower Defense Game” on 16 September. Our first action was to plan meetings, define our goals, and choose the development environment.

Why We Chose Visual Studio

After comparing options like Code Blocks, CLion, and DevC++, we selected Visual Studio for the following reasons:

- It provides better integration with C++ compilers and modern standards (C++17).
- Strong debugging tools and IntelliSense make it easier to find and fix errors.
- It offers built in desktop and game development packages, which are exactly what we need.
- It is widely used in industry and therefore aligns with professional practice.

Installation Choices

When downloading and installing Visual Studio, we selected:

- Desktop Development with C++ (required for standard console and window applications)
- Game Development with C++ (adds libraries and tools relevant for building graphical and interactive projects)

We chose these options to make sure Visual Studio supports both basic C++ development and graphical or game frameworks such as SFML and TGUI.

First Project – Console App

We began with a Console Application. The reasons were:

- It is the simplest way to test whether our C++ environment was correctly configured.
- It allows us to start with text based output before moving to graphics.
- Visual Studio by default names these projects “ConsoleApplicationN”, making it easier to locate them.

We named our project “Tower Defense Game” and carefully noted the file location, since this would be important later when linking SFML and TGUI libraries.

First Example – “Hello World”

To verify everything worked, we wrote a Hello World program:

```
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

Result: The console displayed ”Hello World!” — confirming the compiler and runtime setup.

Integration of SFML 3.0.0 and TGUI 1.11.0

On 17 September, we compared our progress with the teacher’s tutorial PDF and then prepared to integrate SFML 3.0.0 and TGUI 1.11.0.

Configuration Steps in Visual Studio:

1. Open Project Properties (Alt + F7).
2. Select All Configurations (ensures both Debug and Release settings stay consistent).
3. C/C++ → Language → C++ Language Standard → set to ISO C++17 Standard.
4. C/C++ → General → Additional Include Directories → add include folders of SFML and TGUI.
5. Linker → General → Additional Library Directories → add lib folders of SFML and TGUI.
6. Linker → Input → Additional Dependencies → add: sfml-graphics.lib, sfml-window.lib, sfml-system.lib, tgui.lib.
7. Copy Required DLLs into output folders (Debug and Release).

DLL Files Needed:
Debug Mode: sfml-graphics-d-3.dll, sfml-window-d-3.dll, sfml-system-d-3.dll, tgui-d.dll
Release Mode: sfml-graphics-3.dll, sfml-window-3.dll, sfml-system-3.dll, tgui.dll

First SFML Example – Drawing a Green Circle

Once the libraries were linked, we tested with a simple SFML example:

```
// this which is a liberary, lets us use SFML to draw things
#include <SFML/Graphics.hpp>

// this is here so i dont need to write sf over and over again
using namespace sf;

// this is also here so i dont have to write std over and ove again
using namespace std;

int main() {
// it is to like make a window that is 200x200 pixels and call it "Green Circle"
    RenderWindow window(VideoMode({ 200, 200 }), "Green Circle");

    // this will make a circle with radius 100
    CircleShape circle(100.f);

    // this also will make the circle green
    circle.setFillColor(Color::Green);

    // this keeps the window open until we close it
    while (window.isOpen()) {

        // this line code checks for events like clicking the close button
        while (const optional event = window.pollEvent()) {

            // this line code helps if we click close, to stop the program
            if (event->is<Event::Closed>())
                window.close();
        }

        // this will clear the window (make it empty)
        window.clear();

        // draw the circle on the window
        window.draw(circle);

        //  this line will show what we drew
        window.display();
    }
}
```

Result: A window opened with a green circle, proving SFML was successfully installed and running.

Project Planning and Map Design

Parallel with lab work, we began planning our Tower Defense game design:

- We sketched maps with predefined paths and open zones for towers.
- We discussed shortest-path algorithms for enemy movement.
- We then defined towers and enemy behavior: automatic attacks, wave based spawning, and loss or win conditions.
- Considered OOP structure with classes for creatures, towers, bullets, and map zones.

Team Workflow

To stay organized, we:

- we usually held team discussions 3 times per week to share progress and refine our ideas.
- We use tutorials(the slide ”Guide to build and run sfml + tgui in visual studio 2022(windows) and online resources like ”YouTube” to solve problems during configuration and coding.
- We Work iteratively , which is , first with small test codes (Hello World, green circle), and then scaling to the game mechanics.

GitHub Repository and README

We created a GitHub account and repository to manage our project. The steps we used are as follows:

- We signed up for GitHub and created a new repository named “TOWER-DEFENSE-GAME”.
- We then initialized the repository with a README file describing the project and its objectives.
- We Wrote details in the README about what the game will look like, the planned features, and how to build the project in the future.

This process ensures our work is version-controlled and easy to collaborate on.

External Resources Consulted

In addition to the provided PDFs, we used the following resources:

- SFML Official Tutorials: `https://www.sfml-dev.org/tutorials/3.0/`
- TGUI Official Documentation: `https://tgui.eu/tutorials/`
- YouTube: SFML Setup in Visual Studio (`https://www.youtube.com/watch?v=I-hZkUa9mIs`)

Conclusion

So far, we have:

- Created a GitHub repository with a README file to document the project, track versions, and manage collaboration efficiently.
- Chosen Visual Studio as our development environment.
- Successfully installed and tested SFML 3.0.0 and TGUI 1.11.0.
- Written first example codes (Hello World and Green Circle).
- Started planning map design, tower placement, and shortest-path logic.
- Created a GitHub repository and README to manage code and document project progress.
- Organized a workflow with regular discussions and step-by-step progress.

This structured approach prepares us for the next phase: implementing the core game mechanics and refining our Tower Defense gameplay.