McMaster University

Final Project
Classic Battleship Game

Humaa Ambreen 1408109
Tanveer Ghatourha 400043866
Yury Stanev 400105984

SFWR TECH 3PR3

Dr. Peter Basl

5 April 2017

Contents

Scope

The Battleship project scope provides the scope framework for this project. This plan documents roles and responsibilities within the scope.

Project Scope

This project scope is for designing, identifying, analyzing and applying Visual Studio and libraries to create a classic Battleship Game. This includes planning of the rules for Battleship, creating compatible ships and grids, configuring and implementing network settings to ensure interactive player experience.

The project is utilizing the SFML libraries within Visual Studio in C++ to create graphics and implement rules and network settings for the gameplay. The communications management scope of this project includes and is limited to: in-person meetings, TeamViewers, Skype, Google Documents, emails, phone and class meetings.

The time management scope of this project includes and is limited to: Setting milestones within Microsoft Project and creating reminders on Google Calendar.

Product Scope

The Battleship Game will consist of 5 battleships (Carrier, Battleship, Cruiser, Submarine, Destroyer, 10x10 grid per player, 2 players against each other, 1 turn per player and it will utilize 1 library: SFML.

Requirements

The approach that is be used for requirements management for the Battleship project is broken down into three areas: technical requirements, functional requirements and project management requirements.

Technical Requirements

The technical requirements include defining the graphics, rules, networking and testing. The graphics will be created using the SFML library as this will eliminate the number of errors, simplify the code and make debugging and troubleshooting easier. Various methods will be used to understand and utilize SFML features which may include: accessing the SFML developer pages to be aware of the latest upgrades and understand code blocks/syntax, YouTube tutorials, trial and error techniques. Since the version of this Battleship game is classic, the number of turns is limited to five and the rules are predefined according to the game released under the patent of Milton Bradley in 1967. The networking and testing will be done using SFML library within Visual Studio (C++).

Functional Requirements

The project team will design a Battleship game that will be interactive. This means that the game will be able to be played on two separate end devices; one will act as the server and the other will act as a client. This will be done with the use of the SFML/Network.hpp header. The Battleships will include 5 separate ships and their corresponding grids which will utilize the SFML/Graphics.hpp header. These graphics must correspond to the gameplay by ensuring that when the enemy player places the ships on the correct coordinates of the enemy grid, the grid cells will turn red. If the player misses the coordinates, the grid cells will turn yellow. When hit, there will be an audio generated to mimic the sound of a ship sinking to enhance gameplay. The audio has been downloaded from a CC-free and royalty-free website called AudioMicro. This will be done with the use of the SFML/Audio.hpp header.

Both players will take turns in places ships on the enemy grid, the first individual to sink the entire opponents battleship wins the game. This portion will utilize core C++ principles and concepts in Visual Studio including: control structures, functions, pointers, references, arrays, structures, file input/output and classes.

Project Management Requirements

Throughout the project lifecycle, the project manager will ensure all team members are reporting requirement status and raising any issues or concerns with their assigned requirements as appropriate. Microsoft Project and TeamViewer will be the core component of planning and communication.

Requirement Resources

This is a summary are the resources needed to achieve the requirements. For a more detailed explanation on these resources, please refer to the Requirements section.

Requirement	Type	Resources
Technical	Graphics	SFML library (developer webpage for code blocks)
	Rules	Classic game according to rules defined by 1967 game.
	Networking	SFML library (developer webpage for code blocks)
	Testing	Visual Studio (C++) compiler
Functional	Graphics	SFML/Graphics.hpp header
	Networking	SFML/Network.hpp header
	Audio	SFML/Audio.hpp header Audiomicro.com downloads
	Gameplay	C++ principles and code
Project Management	Time management	Microsoft Project/Google Calendar

	Communication management	Skype, TeamViewer, Google Documents, emails, phone and class/in-person meeting

Third party library

SFML

The 3rd party library utilized within the Battleship game was SFML: Simple and Fast Multimedia library. The decision of which library to utilize was analyzed between Qt and SFML. The decision to use SFML was a result of the numerous modules provided, resources available to understand and compatibility with game code compared to Qt.

The library uses the following modules: system, window, graphics, audio and network. The modules allow the code to be seamless, simple, compatible and easy to implement. The system module utilizes vectors and Unicode for positioning and color change. The window module supports the window and input management of the code. The graphics module creates the 10x10 grid and the ships graphics. The audio module allowed the project to consist of an audio file that generates a sinking sound when the opponent ship gets hit. The networking module enabled the game to have gameplay between 2 separate players.

Resources utilized to understand the SFML library code blocks and syntax included: SFML Developer website – code blocks, updated version changed, support forums, YouTube courses and SFML tutorials files created by various instructors. In addition, trial and error was the core method of learning the functionality of the SFML library.

Challenges

Networking

The networking portion of the project was challenging as there was minimal documentation to support the understanding of the concept. The difficulty in the networking portion was sending multiple packets. This was overcome with minimal code understanding and trial and errors. This portion required extensive testing and external resources such as YouTube courses.

Graphics

The original plan was to generate 2D graphics from the 2DTorque database. This was not achieved as the website had been closed down. In addition, general external graphics caused many errors in the code and made it longer which resulted in bugs. To overcome this challenge, we first tried to implement Qt. The purpose of Qt is to create a compatible GUI for the players. This caused even greater errors as the code required to implement these graphics were much more complex and caused even more errors when testing and debugging. The solution was found when the removal of Qt library and implementation of SFML library occurred. SFML made the code seamless and error-free.

Additions and Enhancement

Possible future additions and enhancement include the use of improvement in graphics and audio. The enhanced version would most likely implement the Qt library alongside the SFML library. The Qt library would utilize complex 2D graphics and additional audio that correspond and are relevant to the game.

The enhanced gameplay would have additional rules that differ from the classic game that would give a twist and confusion to the player. The more complex the game, the more addicting it is.