MET cS665 Summer 2 2018

Abel Gancsos

Abel Gancsos productions | Hawthorne, NJ

AMGBattleship v. 2

2018

Contents

1 Application Description 3

1.1 Implementation Details 3

1.2 Assumptions 3

1.3 Prerequisites 3

2 Assignment Tasks 4

2.1 Refactoring 4

2.2 Implementation Description 5

2.3 UML Class Diagram 6

2.4 UML Sequence Diagram 7

Bibliography 8

Appendices 9

Appendix A – Setup 9

Appendix A – Building project 9

Appendix B – Running executable 10

# 1 Application Description

## 1.1 Implementation Details

*The following details are also available in the README.md file*

This command-line utility serves as a text-based console version of the classic Battleship board game. Typically there would be two users that would have the same number of ships on the board at different locations and they both take turns trying to sink the other users ships. In the real game, each ship would have a different length and thus a different number of required shots to sink it; however, in this version of the game, every ship is represented by the same size, but in different locations. After each shot, the current player will only see there board, locations of their ships, what shots were taken against their ships, and which ships were hit/sunk. The board is then cleared in preparation for the next player.

## 1.2 Assumptions

* Users play the game on the same command line, but their own context during their turn.
* The game may add other attacks in the future.
* There will only be 2 players at a time on the board.
* A ship will only take up one block on the board. This may change in future releases.
* The ship count can not and should not be larger than the map size.
* Both players will start with the same number of ships.
* The board width and height are the same for each player.
* The values for the board height and board width are for each side/player (not the total board).
* The current player will only see their own board.

## 1.3 Prerequisites

* C++ compiler with C++11 capability
* CMake version 3.11 or higher
* XCode command-line tools (MacOS only)
* Visual Studio 2017 or higher (Windows only)

# 2 Assignment Tasks

## 2.1 Refactoring

## 2.2 Implementation Description

*The following details are also available in the README.md file*

## 2.3 UML Class Diagram

## 2.4 UML Sequence Diagram

# Bibliography

Döring, A. (2016, 11 07). *minimal\_cmake\_example*. Retrieved 07 03, 2018, from GitHub Web site: https://github.com/krux02/minimal\_cmake\_example

Google Corporation. (2018, 06 19). *Google C++ Style Guide*. Retrieved 07 03, 2018, from Google GitHub styles: https://google.github.io/styleguide/cppguide.html

Kitware Corporation. (n.d.). *Download*. Retrieved 07 03, 2018, from CMake Web site: https://cmake.org/download/

Lucid Software Corporation. (2017). *Class Diagram Tutorial*. Retrieved 7 25, 2017, from Lucid Software Corporation Web site: https://www.lucidchart.com/pages/uml/class-diagram

Microsoft Corporation. (2010). Visio. (2010). Redmond, WA, USA.

Oracle Corporation. (2017). VirtualBox. (5.1.18). Redwood City, CA, USA.

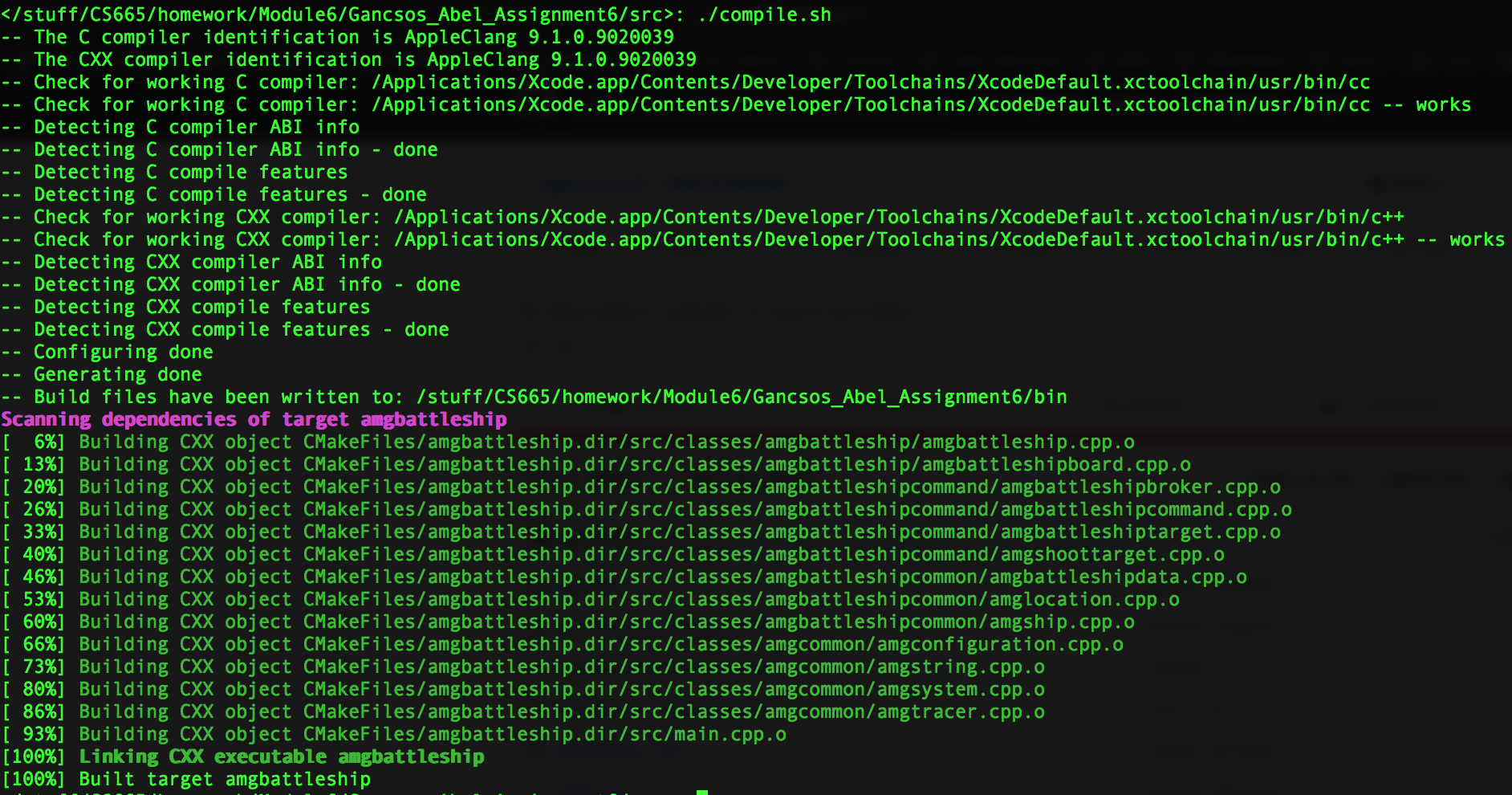
Tutorials Point. (2018, 07 14). *Design Patterns Command Pattern*. Retrieved 07 24, 2018, from Tutorials Point Web site: https://www.tutorialspoint.com/design\_pattern/command\_pattern.htm

# Appendices

## Appendix A – Setup

The project can be built using the run.sh or run.bat scripts, depending on the platform. Both of these scripts will run cmake to generate the CMakeFiles and then run the make command in the bin directory. Alternatively, the project can be built manually using the CMakeLists.txt file that comes with the package.

## Appendix A – Building project



## Appendix B – Running executable

