Date: 15-12-2021

Supervising assistant: Sohlman Aleksi

ELEC-A7151-Object oriented programming with C++

Tower Defense Game 7

Navid Ahmad 796974

Xin Zhong, 875167

Martin Koskinen, 789567

Narayan Ghimire, 876470

Contents

Overview

The program works as a 2D tower defense game. The game is designed in the Martian theme. In 2024 during his first visit on Mars, our space pioneer Elon Musk experimented on cockroaches to see if they would start adapting into their new surroundings. The tests provided no significant results. While leaving back to earth the research team had left something behind, but the consequences of that error would be bigger than anyone would have thought.

Few years after humans had settled Mars, a new terror appeared seemingly from nowhere. Human sized alien cockroaches had infested the surface of mars and they started to attack the settlers.

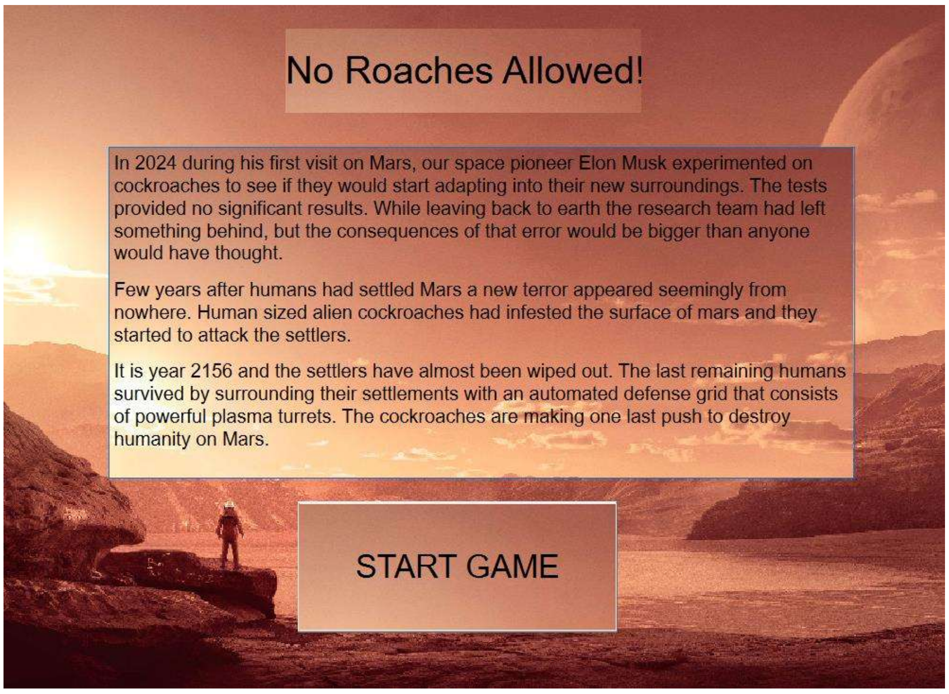
It is year 2156 and the settlers have almost been wiped out. The last remaining humans survived by surrounding their settlements with an automated defense grid that consists of powerful plasma turrets. The cockroaches are making one last push to destroy humanity on Mars.

Humans have built a defense system at entrance of their settlement. The defense system has three types of towers available for destroying cockroaches. They are basic tower, shotgun tower, and machin e gun tower. Basic tower is a semi-automatic turret with a slow rate of fire. The turret shoots plasma rounds once at a time when the cockroaches are within its standard range. Shot gun tower fires multiple rounds at once and has shortest target range of all. The machine gun tower is the most powerful one with a high fire rate and range. The high rate of fire decreases its accuracy by adding spread to the plasma rounds.

There are three distinct types of cockroaches, namely child cockroaches, male cockroaches, and mother cockroaches with distinct attributes. They all have different health and different speeds. The child cockroaches are of the highest speed and low health. The mother cockroaches are with lowest speed and highest health. Once the mother cockroaches are destroyed, it can give birth to four child cockroaches.

In this game, the hostile cockroaches move in waves from one specific place to the entrance of the human settlement. The goal of the player is to place the towers strategically along their path to attack or destroy the cockroaches before they reach the entrance. The game is over if a cockroach enters the entrance of the human settlement. The player has the default amount of money known as Elon (named after the pioneer of Mars colonizer) to buy the towers at the start of the game. As the cockroaches are destroyed, more Elons will be added to the store as a reward. The primary object of the game is survival of human base from mutated cockroaches.

The User Interface (UI) and the main game window are designed exactly as planned initially which is illustrated in the figures below.


No Roaches Allowed!

In 2024 during his first visit on Mars, our space pioneer Elon Musk experimented on cockroaches to see if they would start adapting into their new surroundings. The tests provided no significant results. While leaving back to earth the research team had left something behind, but the consequences of that error would be bigger than anyone would have thought.

Few years after humans had settled Mars a new terror appeared seemingly from nowhere. Human sized alien cockroaches had infested the surface of mars and they started to attack the settlers.

It is year 2156 and the settlers have almost been wiped out. The last remaining humans survived by surrounding their settlements with an automated defense grid that consists of powerful plasma turrets. The cockroaches are making one last push to destroy humanity on Mars.

Figure 1. User Interface

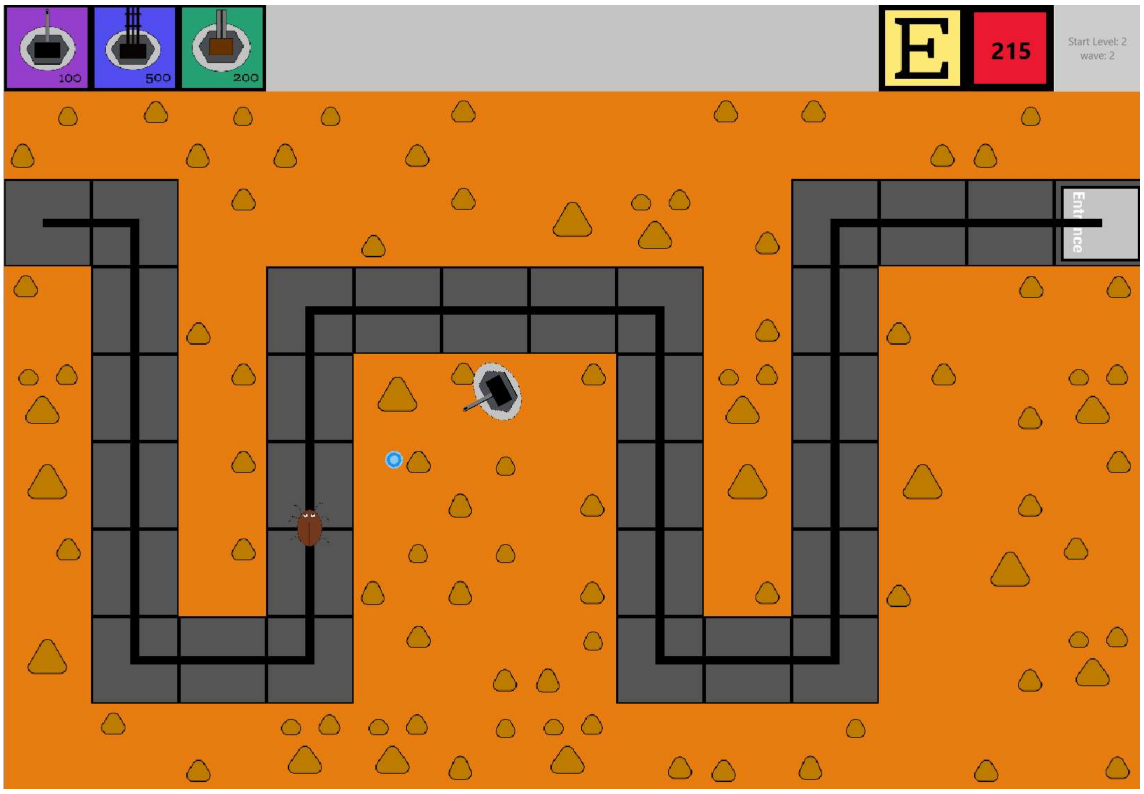


Figure 2. Main Window

Compared to the initial plans, most of the core features are placed in the final game with few additional features.

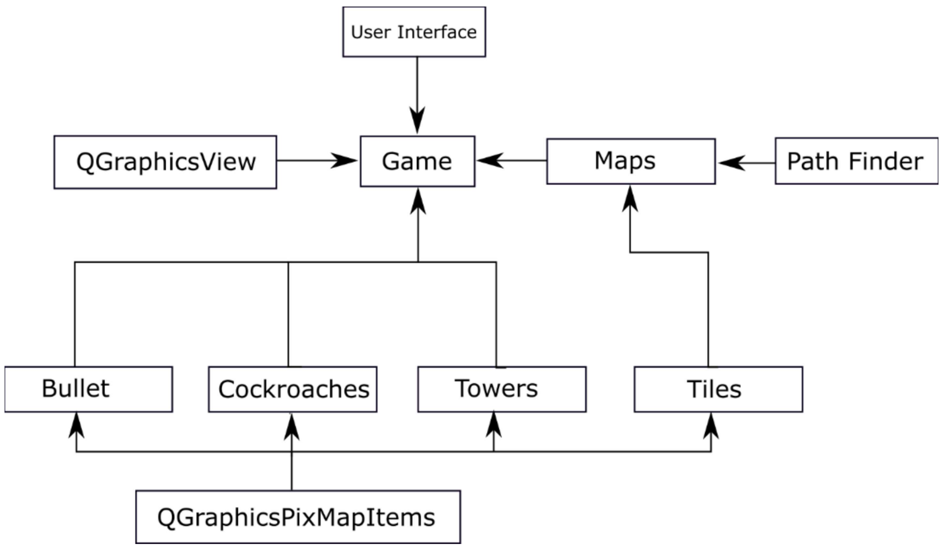
Minimum Requirements

- A functioning tower defense game with basic graphics
- At least three different types of towers
- At least three different types of enemies
- Controlling the game by mouse
- User interface that shows resources, number of waves, available towers etc.

Additional Features

- Randomly generated map
- Sound effect

Software Structure



The start button in the user interface leads to the game. Game state, which is inherited from QGraphicsView, is the engine of the program. Game manages all the graphics such as maps, towers, cockroaches, shops, wallet in the window and responds to player action. The tiles used to generate maps and all the other objects in the game are inherited from QGraphicsPixMapItems. The maps which are generated by the 2D vector of tiles also create a path for the cockroaches to follow.

Qt provides the mouse tracking events, which holds the position of the mouse all the time. This makes it possible for a player to grab and place the towers on the map.

Communication between different components, for example bullets and enemies, towers and road tiles are made possible because of collision detect method of QGraphicsPixMapItem.

Instruction for Building and Using the Software

To build the software, we need to install Qt6 software and CMake.

Steps to run the software:

1. Fetch the source code from the link below:

cpp-autumn-2021 / tower-defense / Tower Defense Group 7 · GitLab (aalto.fi)

Open terminal and type command

“git clone git@version.aalto.fi:cpp-autumun-2021/tower-defense/tower-defense-7.git”

2. Run Qt and open the project folder
3. Configure the project and run the game

Testing

The project has not been tested using unit test due to time constraints, however it has been manually tested by running the application after the addition of each new implementation. This method was exceptionally beneficial since the testing of application was possible in a different phase. Most of the time actual testing was performed by printing debugging information to the console.

During the development phase, functionality of the various classes and functions are often tested by implementing a manual test program which helped in further development and addition of improved functionality. Memcheck() function was often used to check segmentation fault.

In the final stage of the development, the difficulty level and robustness of the game was tested by running the game repeatedly and examined its functionality. This led to the improvement of minor changes to the attack range of the different towers, speed of the different cockroaches, damage, health, cost of towers, amount of rewarded money etc.

Work Log

Since early November, Tower Defense Game 7 has had a group meeting each week to distribute and track the member task progress. Originally, team members agreed to have averagely 8 hours spent on programming each week to make progress.

First week meeting

The project started with the first week meeting which made a draft of project plan. The initial conceptualization of software structure visualized overall architecture and class relationship diagrams. Meanwhile, each team member installed QT creator with CMake.

Second week meeting

In the second meeting, since we had a clear idea of what to do after basic training session, the responsibilities were modified, and the work was divided by the basic elements of game and assigned to each member as below:

- Enemy -> Xin Zhong
- Pathing finding and Map generation -> Navid Ahmad
- Tower -> Martin Koskinen
- Player and wallet -> Narayan Ghimire
- Main UI -> Navid Ahmad

In this meeting, we agreed to merge the source code into master only in face-to-face meetings.

Third week meeting

During the third meeting, we did the first-time git merging of software from map, enemy and tower. The game had built a map and enemy path. The enemies could be spawned and moved along the path. Basic on the simulation, the meeting raises the demand to build different towers variable shooting mechanics. Also, some issues were concluded for the further debug, for example the map scaled in screens with different specifications cannot be displayed well. The cockroach movement cannot follow the center of path line. To fix these issues, we made an action plan and assigned the topics to each member for the next work week as below:

1. Clean up cockroaches and update some features -> Xin Zhong
2. Implement towers to shoot cockroaches down.
3. Build shop basic features and UI design.

More specific requirements were added into the Todo list in the meeting notes.

Fourth week meeting

The fourth meeting only merged the tower branch into master. During the meeting, we found that Enemy class had problems as below:

1. Functions and variables naming was against the Google naming rules.
2. Subclass constructor shall do initialization of base class after the explicit initialization of parameters.
3. Function definition is verbose and unreadable, should be split into small functions.
4. Unneeded functions in public slots

The meeting required to do frequent commit for changes in software and gave suggestions to the structure of Enemies class objects as well as the Google C++ style check.

The final meeting met challenges like the cockroach pathing unsmooth way, and the implementation of towers shows that the tower icon didn’t follow the cursor. Our team took several hours to find the simple solution of cockroach movement and make the tower icon follow the cursor. In this meeting, most features were synthesized and tested. Meanwhile, we got better ideas to add sound effects and notifying messages.

Total contribution:

Note: A lot of work were implemented in groups.

- Enemy -> Xin Zhong
- Pathing finding and Map generation -> Navid Ahmad
- Towers and projectiles -> Martin Koskinen
- Shop, wallet and tower building -> Narayan Ghimire
- Main UI -> Navid Ahmad
- Game mechanics and logic -> Group collaboration
- Documentation - > Group collaboration