Data-Structure Project

Semester 16’ - Team #15

Tuesday, December 20, 2016

## Simulator Structure:

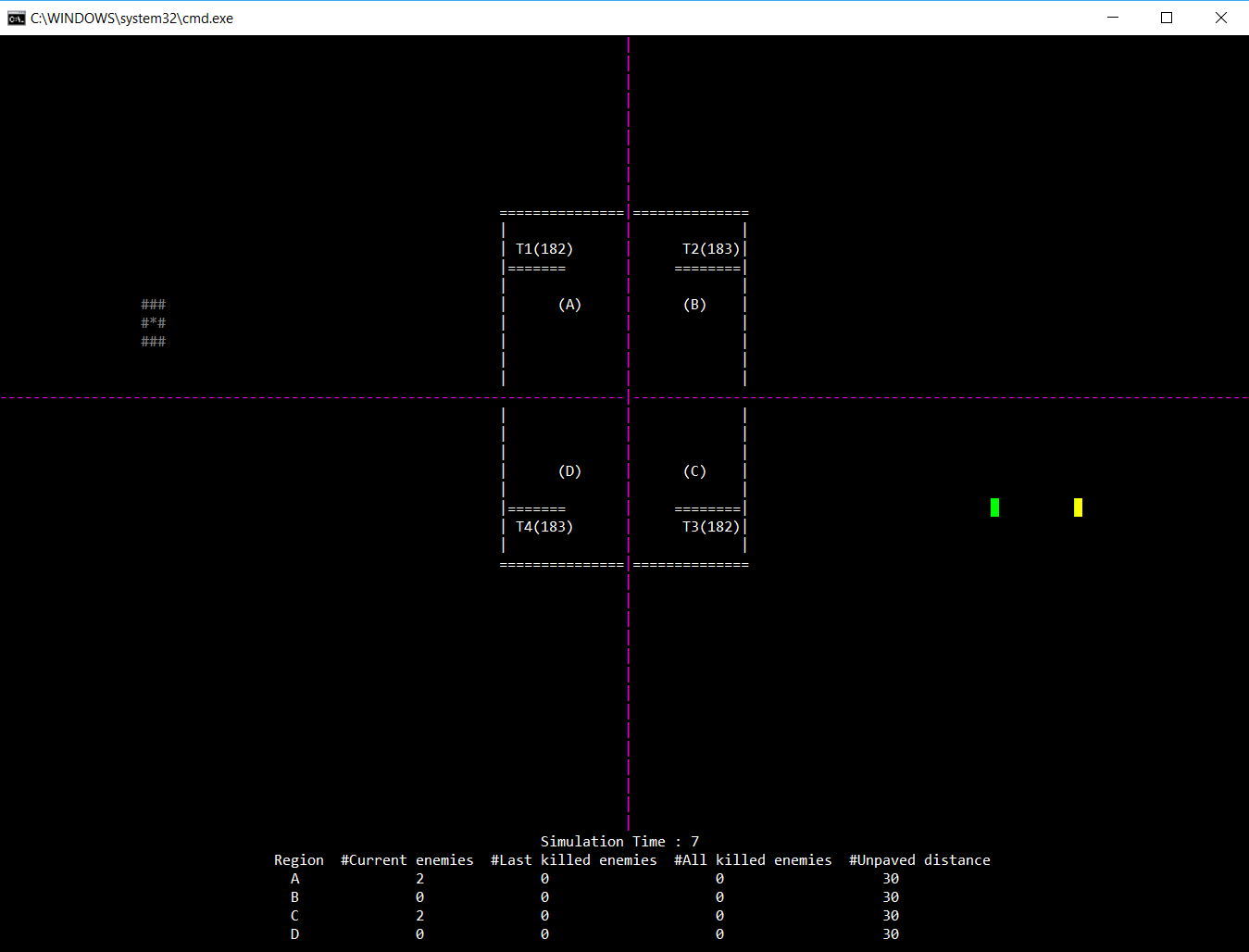
The project is separated into 3 main modules: Control, Data and Graph, each has its header and source files

Header files contain prototypes and source files contain the implementation

Each module contains different namespaces to represent submodules with their functions. e.g ENEMY, TOWER, Control etc..

The use of namespaces is to be able to use multiple functions with the same name for different modules, this approach does increase readability, maintenance and modularity of the project

NOTE: enemies have different speeds



screenshot of the simulator showing the Tank and two enemies

## To Increase Performance:

1. While looping in enemies, function ENEMY::Loop stops when it finds inactive enemy.
2. Function Control::Read() that loads data to memory, doesn't traverse the whole list of enemies to add one, it saves the last enemy assess in a pointer.
3. DrawEnemies() is overloaded to be able to take the whole castle instead of array of enemies, in order to avoid allocating special array for each region.
4. To Sort shielded enemies depending of priority, we stored their addresses ,instead of storing all of their data, in array, and sorted the addresses.
5. When passing big data to functions, we used constant reference to avoid passing huge amounts of value or the addresses themselves.

## Main Modules:

### Control:

Contains main functions that control the flow of the simulator. e.g loop, start, get mood and refresh screen etc…

NOTE: some overwritten graph functions are in Control not in Graph

### Data:

Contains data structures: Enemy, Castle and Tower, each contains its attributes (variables).

Also it has namespaces for each data module, each one has functions/helper-functions for that module

### Graph:

Contains provided graphing (drawing) functions. It is a rename of utility

We added some functionality in it to support drawing Tank enemy

Overloaded functions of DrawEnemies are written in Control to avoid changing the original.

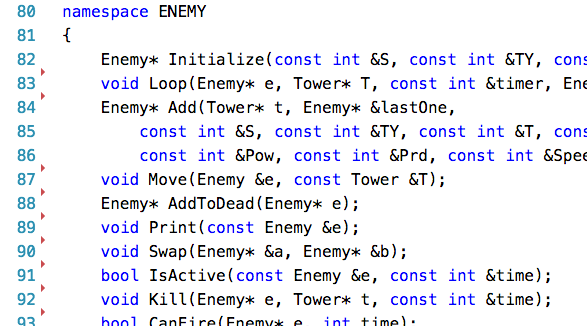
## Data Structure:

We avoided using OOP, instead we made structs to hold variables and exported its functionality in namespaces

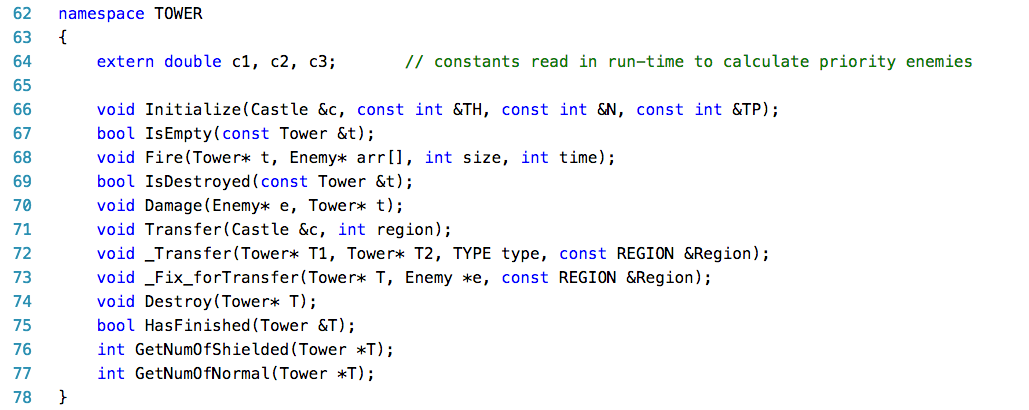
We didn’t modify given variables in structs, but added new

Here provided screenshots of the modifications:

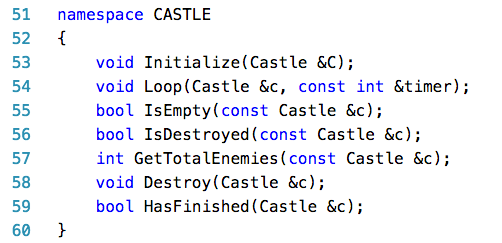
### Enemy

ENEMEY namespace:

### Tower:

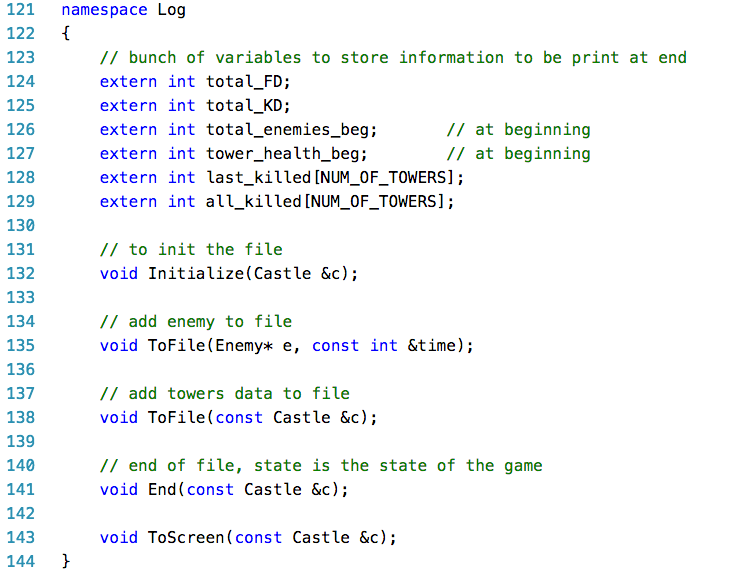
tower namespace:

### Castle:

Castle namespace:

### other:

Project included other modules like:

* **Log** (handle output file data and printing statistics to screen)
* **Shielded** (contains functions to handle priority of Shielded enemies)
* **Doctor** (contains Heal() function that increases health of two (if available) enemies around him)
* **Paver**
* **Tank**

## New Enemies:

We added 2 new enemies: Doctor and Tank, they only take action when their fire period finish

* *Doctor*:

Heals the two enemies near to him, each enemy’s health is increased by doctor fight power / 2

Doctor can't heal a tank

colour is WHITE

* *Tank*:

Generates new random enemies to the list behind him, each one has arrival time = Tank arrival time + 1

Tank doesn’t generate Tanks

NOTE: tanks leads sometimes to unpredicted behaviour, you cant predict exactly how will be the results.

Shape is 8 grey # with \* in the middle