Destroy or Defend

Introduction

Destroy or Defend (DorD) is a tower defense multi-player board game. On a massive 10,000 by 10,000 square arena, a defender aspires to protect their main base and an attacker aspires to destroy that base. The defender uses a combination of stationary towers (supporting turrets, cannons, sentinels and guns) and patrolling forces (consisting of soldiers, tanks and other vehicles) to protect its base. An attacker uses marching armies (consisting of soldiers, tanks and other vehicles) to attack the defender's units, towers and base. Given a preset time, the attacker must complete their task by successfully destroying the base and win; otherwise, the defender receives backup units, destroys the attacker units and successfully defends the base and win.

There are two teams in the arena, an attacking team and a defending team. Each team comprises one or more players. You are one of those players and must successfully *destroy or defend* the base, depending on which team you are on. The game consists of only one run. Before the beginning of the run, each player receives a preset number of "points" which they can use to buy and setup their defending or attacking units. Each unit is also given an autonomous tactic which will be followed throughout the run. A player is allowed to setup their units on certain regions of the arena. Once the run starts, the game units behave autonomously (without any intervention from the player). The units attack or defend following their assigned tactics. The run continues until one of the following occurs where one victor is declared:

- The base is destroyed, regardless of how many defending units are still alive;
- The attacker units are destroyed and no more attacking units are available; or
- The preset time is up, which means the attack has failed.

Specifics

- Before the beginning of the run, players receive "**Points**" which are used to setup their units;
- Players buy and upgrade units and other skills;
- Players deploy units in designated regions in the arena;
- Units have health values which are lost due to being attacked; once its health is depleted, a unit is destroyed;
- Attackers team units' move autonomously in the arena generally towards the defender base following preset tactic;
- Defenders team units' might be stationary or patrolling, they also behave autonomously by deciding which attacking units to target.
- Units can behave following one of these tactics:
    - Attack units with lowest health;
    - Attack units with highest damage;
    - Attack randomly;
    - Attack units with a predefined priority (ex: the player will say: attack the "Car bomb" first and then focus on the RBG soldiers and so on...).

- Units have attack damages, target types, attack speeds and bleeding sources[1];
    - Ex: Tesla Tank: has 300 attack damage and it can attack other tanks and soldiers, but it can't attack structures. It can at attack 3 tanks each 10 seconds. Snipers can't attack it.
- Units have attack speed which tells them how often they can attack:
    - Ex: sniper attack speed's is 1 which means they can perform one attack per second.
- Unit have armor: a discounting factor of received damage:
    - Ex: attack dmg = 300, armor = 0.4 => total damage dealt to the unit is (300 – (300*0.4) =180).
- Units can't move over each other:
    - Ex: if there is a tank wants to move and there is another unit in its path, then this unit should find another path, if it's stuck then it will stand in its place.

The only parameter of the game is the initial "**Points**" offered to each player at the beginning of the game.

Temporary Restrictions (for the first milestone)

- You may assume that the initial state of the game is a static and is already given to you. This includes the arena, the units and their deployed regions. This means there will **NOT be any input from the user**. You may code these objects statically; or simply obtain them from a file.
- Once a ran starts, you will need a pause functionality which shows the state of the game.

A proposal of the game arena:

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td colspan="3" rowspan="2">Attacker player 1 units</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="2" rowspan="2">Attacker player 2 units</td><td></td><td></td><td rowspan="2">Attacker player 1 units</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td rowspan="2">Defender player 1 units</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3">Defender player 1 units</td></tr>
<tr><td></td><td rowspan="2">Attacker player 2 units</td><td></td><td></td><td></td><td colspan="2" rowspan="3">Defender player 2 units</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>Defender Base</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

As shown on the proposed arena, we will not show all the detailed squares. We will print a summary of each group of cells and we can query (zoom in) on any group to look into these squares units.

[1] Complete list of each unit's properties are listed in Appendix A.

Game Implementation

In this first milestone, we will focus on implementing the core functionality of the game. But, **before** delivering the first milestone, you will be asked to implement a batch of changes. At a later stage of this project, you will be asked to implement **a second and a third batch** of changes improving the gameplay.

Functionality 1 (6 marks)

You should build the core of the game, which is the game environment. This functionality requires console interaction only (no GUI).

- The arena, with zoom in;
- The units: movements' and attacks;
- Use the random attack “tactic”;
- You should provide a valid case of using inner classes and exceptions handling.

Functionality 2 (3 marks)

- Implement the unit attack’s tactics (as presented above).
- Print a detailed log for each unit (its movements and attacks).

Functionality 3 (2 marks)

Apply the units attack speed concept.

Functionality 4 (4 marks)

Develop a GUI for the game with zoom in/out.

Additional functionalities

- Implement an algorithm to build the team units (best choice based on its price, power and what it can target) **two marks**;
- (GUI only): Implement an interface to build the team: **three marks**
    1. Each player will make one transaction each time:
    2. This transaction is buying one type of units and how many copies of it, then put them on the arena (the player will pick a square and the units will spread near to it)
    3. This interface will end when the players click on “Start game button”;
    4. This interface should respect the “total unit points” that the player has.

Project organization

- Each 4 – 5 students will work together;
- Groups of six students are allowed but must implement the additional functionalities;
- Additional functionalities have 0 marks if any of the main functionalities are not implemented;
- Functionalities evaluation has four levels:
    1. The functionality is working and the application works: full marks if the evaluator does not have any concerns or questions.
    2. The functionality is not completely finished but the application works: if there is a serious try then the group will take the half.
    3. Both the functionality and the applications are not working: if the team has deep understanding of how it should develop then the group will take few marks.
    4. There is no serious try to implement and the team does not understand how to do it then they will take zero.
- Delivery report must show the tasks and efforts of all students;
- Delivery report must be **two pages** at most and must contain the **data structure** and the algorithms;
- On the evaluation session, the group should provide the evaluation form (see appendix B);
- Project delivery will be by running the application, and then the evaluator asks any questions about the project and the implementation.

Appendix B

Delivery Form

First Milestone

Class number

Delivery date: 29-Nov-2020

Number of students

Student's names

Functionalities	Evaluation	Marks
Functionality 1		
Functionality 2		
Functionality 3		
Functionality 4		
Additional functionality		
Final Marks		