CoKe
Kevin Bao, Connie Lei
Period 1
TLSFN

Overview

Our proposal for our final project is a strategy game that takes place in a town and the player's objective is to defend the town hall (which is situated in the center) from hordes of evil-doers through the means of constructing defensive buildings and training troops to fight the enemies. The player may also gain the ability to use spells, but only if we have enough time.

Buildings are built with money, which a player starts off with a base of certain amount (to be decided later and dependent on difficulty level). Buildings are instantaneously built and serve different purposes. Barracks allows soldiers to be trained, a wall can be built as a line of defense, a cannon shoots at enemies, etc. Each building will have attributes like health, damage, and range. The buildings will be stored in an ArrayList.

Troops are trained, but require money and time to complete their training. This will be implemented with a priority queue. Troops can be continuously added to the training list, and once a troop finishes its training, it's specific attribute will be increased and the rest of the troops will fill in the spot. If a higher level troop is taking too long to train, and you want to quickly train lower level troops, you can reorder them.

The enemies (controlled by the computer) will approach the town hall from random spots around the borders of the town. However, in order for them to reach the town hall they would have to destroy the buildings/walls surrounding it. A binary search tree/heap will be used to decide the path of an enemy unit. How you ask? A BST/heap will contain the values of the distances between an enemy and allied buildings/troops, where the shortest distance is located at the root. The enemy will always move in the shortest path between itself and a building/troop and the owner of the value at the root of the tree will be the enemy's target. Killing enemies grants gold and enemies get progressively harder as time progresses (also dependent on difficulty level).

Backend Structures:

BST/Heap:

- Stores distances between an enemy and every building/troop, where the shortest distance is the root.
- The building/troop whose distance to the enemy is the shortest will be the building/troop that the enemy will move to. Basically, a BST/Heap is used to find the shortest path for an enemy.

PriorityQueues:

- Troops are trained in the barracks where the first troop that goes in will be the first troop that comes out.
- However, the player can choose to bring another high-priority troop to the front of the queue to have to finish training first.

Stacks:

- Stores enemies depending on difficulty, with the boss/hardest added at the beginning of setup
- Pops enemies at increasingly short intervals
-

Frontend Structures:

- Buildings
    - Player should be able build and place buildings by clicking (to select a building from a menu), dragging (to move building to position), and clicking again (to place building).
- Barracks
    - Player should be able to train a troop by selecting a troop from a menu.
    - Player should also be able to drag a troop (that is waiting in queue) to the front of the queue, causing the this troop to be trained first.
- Special Spell
    - Clicks on a certain button that does something special, wait till the project is done to find out what...recharge time
- Troops
    - Player should be able to select a troop and then select an enemy to order the troop to pathfind to the enemy and attack it.

Features:

- Special player spells
- Difficulty levels
- Customizable troop attributes
- Different types of enemies
- Customizable terrain generation, color scheme
- Different types of buildings
    - Cannons, moats, turrets, walls, cannons, all that sha-bang
- Movable/reordering the training of troops

Algorithms:

- Spawning in/start of game, creation of the town, random world generation, background
- Detecting where buildings are allowed to be built
- Computer generated enemies, number, health, defense, spacing, timing
- Distance calculations for enemies
- Damage calculations for buildings/structures (scanning a range to find enemies and then damage them)
- Troop path finding

Classes:

***abstract* class Structure**	
Attributes	Methods
- defense points - health points - attack damage - width - height - color - coordinates	- accessors/modifiers for each attribute - attack ability - repair ability - special ability (freeze, slow, heal)

class Cannon, Turret, Wall, Moat, etc extends Structure	
Attributes	Methods
- everything in **Structure**	- everything in **Structure**

class Unit	
Attributes	Methods
- health points - defense points - weapon - attack damage - armor - speed - coordinates	- accessors/modifiers for each attribute - attack ability (enemy troops) - heal ability - find nearest enemy ability

class Enemy extends Unit	
Attributes	Methods
- everything in **Unit**	- everything in **Unit** - attack ability (enemy structures)

class Troop extends Unit	
Attributes	Methods
- everything in **Unit** - trainingTime	- everything in **Unit** - train ability - target ability