COP290
Assignment 3

Harishchandra Patidar Rohit Sachin Kumar
Manoj Kataria

March 24, 2015

Battle and Chaos

Contents

1 Introduction

In this assignment, we will be building a multi-player game **Battle and Chaos**. Many variations of this type of game are available but none of them have the multi-player feature. This game is sort of mixture of real time strategy, action and tower defence.

2 Game Description

There will be two teams in the game. Each team will have equal number of players (bots if required).

Each team will have

- **Base** which it has to defend. The team which destroys the opponent team's base will be victorious.
- **Attacking units** to destroy opposition's base. The team can release attacking units(if available) at any point of time.
- **Defense units** to defend the base.

Our Game will have following basic units :

- **Hero** : Each player chooses a hero from a set of available heroes at the start of the game. A hero can move anywhere in the arena and its motion is controlled by player himself. It can attack enemy heroes and creeps. If a hero dies, it re-spawns after some time (or instantly if the player wish to spend some gold).
- **Creeps** : Creeps are the basic attacking units. Each team send creeps towards opponent teams base. Creeps move only on certain paths (there are many). The set of paths used by two teams is disjoint, therefore creeps of two teams do not collide with each other. The base is destroyed if a set amount of enemy units pass through it. Creeps are of various types and can be upgraded by the player.
- **Towers** : Towers are the basic defense units. Towers can be placed anywhere on the arena except in front of base or on the path on which enemy units move. Towers attack enemy units and heroes, by shooting bullets towards them, which adds some amount of gold to our bank depending upon initial health points of dead units. Towers are indestructible and of various types. Placing and upgrading towers costs gold.

Each team starts with some of these basic units and a fixed amount of gold. Each of the above units can be upgraded by spending gold. The cost of up-gradation increases as the units become more powerful.

3 Components

The implementation of this game is broadly divided into the following components:

- Game Attributes
- Graphics and Sounds
- Networking
- Artificial Intelligence

4 Game Attributes

Different units have different attributes. Description of general attributes of units:

- **Base** : has some initial HP(health points) which will decrease every time a creep passes through it or on attack by opposition heroes. The decrement will determined by following formula:

  $HP_f = HP_i - HP_{Unit} * d$, where d is the decrement factor of unit

- **Tower** :
  - range : in which it can shoot
  - damage : which it does to creeps
  - attack speed
  - cost
  - upgrade state : whether fully upgraded or can be upgraded further
- **Creep** :
  - movement speed
  - HP
  - cost
  - upgrade state
- **Hero** :
  - HP
  - damage : which it does to opposition heroes and their attacking units
  - attack speed
  - movement speed
  - range : in which it can attack
  - upgrade state

These attributes can be changed by spending gold.

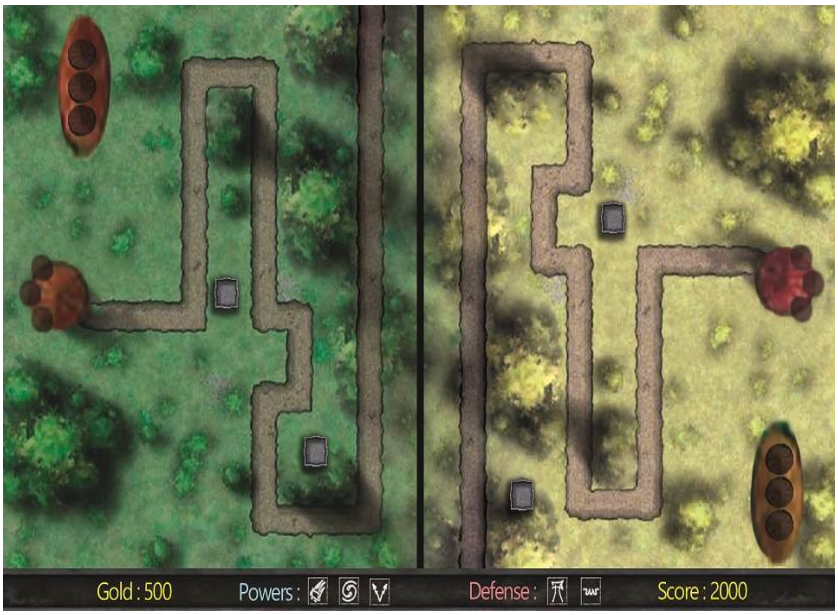
5 Graphics and Sounds

Overall, we will be implementing following things:

- Arena of the game
- Different types of enemy units
- Towers
- Sound Effects
- User Interaction using mouse

5.1 Gameplay view

The game will start with an arena in which there will be two bases one of each team, which they have to protect. The top view of the arena will be displayed.



Now, to implement this we will be using SFML library for C++. We have to show towers, enemy units, base and make them interactive. User can click them and do some actions. Background will be a simple rectangle with an image texture mapped on it. Above it there will be another layer consisting of path. Creeps and towers will be represented on the next layer.They will be rectangles, with alpha mapping and texture mapping. Alpha mapping will account for the custom shapes of the units.

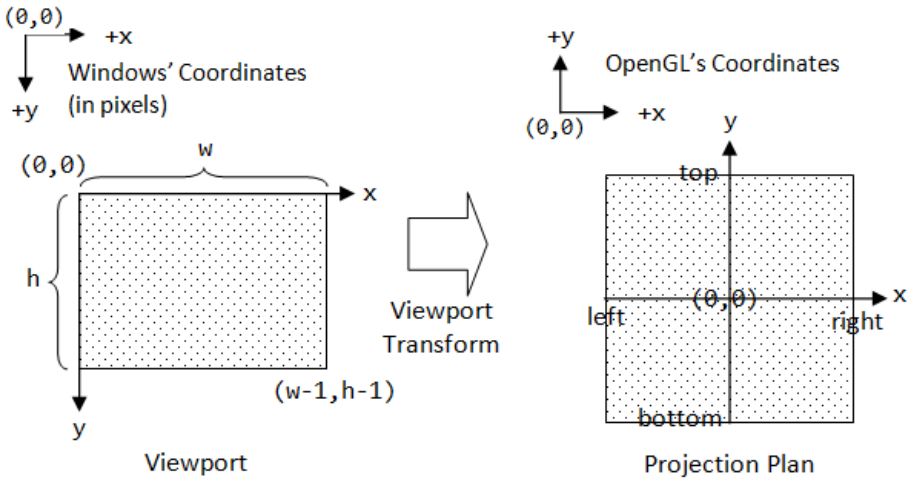
Apart from the basic design, we need to make it look more beautiful, so there will be some texture mapping and more effects which are available in SFML library.

5.2 Sound

A mute game is no fun, so we will add sounds in the game. Sound effects will be very realistic with a fine background music to give the feel of the game. We will be using SFML library for sound. For playing small sounds we will use .wav files in sound buffer and for long music we will stream .ogg files in music buffer.

5.3 Mouse Interaction

Whenever a player clicks on the game screen, using the co-ordinates of the clicked point we will calculate the unit which is being clicked and accordingly a menu for that unit will appear. The player can now choose one of the options from menu and change the attributes of selected unit.



6 Networking

We describe here networking portion of game :

- The game will be server free.
- It will be based on Peer-to-Peer network protocol.
- One of the player starts the game and other player joins the game by connecting to all the other players.
- The connection between two player will be established using TCP sockets.
- Any person can leave the game during the game play.

6.1 Connection

We will be having following classes to implement networking in our game :

- class socket : open(),close(),send(),receive() methods
- class connection : listen(),connect() methods

Since every player will be connected to all the other players, connections on one computer will be managed using threads for better synchronization between players.
If a player leaves the game mid-way or connection is lost then we will replace the player by a bot and game will remain balanced.

6.2 Message Transfer

Messages will be transferred when player,

- introduces new defense or attacking units.
- upgrade/remove an existing unit.
- changes direction of attacking units.
- move its hero unit, basically these messages will be sent on every frame of the game since all the players should know that position of each other in sync.

Messages will be stored in two queues, one for the messages send by the player and one for the messages received. At each frame, each player will

- first process the incoming messages
- send messages back to the network
- render

7 Artificial Intelligence

To make the game more interesting, we will require our basic units to have some intelligence in them. This will be a significant challenge.

7.1 AI of basic units

We describe here the AI component of each unit

- **Creeps** : As mentioned earlier, a wave of creeps will aim to reach opposition's base via a path which causes minimum damage to themselves. Creeps won't always take the same route. To find the optimal path, we will use **Dijkstra's algorithm**. In the algorithm, the weight of an edge between two nodes(successive tiles of path) will be taken as damage that will be done to creep if it moves along that edge.
  We will be using following formula to calculate weight of edge as:

  $W = X * damage + 1$
  where X is a large constant. We need to minimize $\sum W$.

- **Towers** : Towers will be shooting bullets on a wave of creeps in its range. In a wave, it will first shoot the creep which is closest to the base then try to shoot others if the first one is destroyed or goes out of range of tower. This can be implemented using a simple data structure like queue. Whenever a creep comes in the range of the tower, it will be pushed into the queue of the tower and will be popped when the unit leaves the range of tower or destroyed. There will be some modification when the units are controlled freely by the player.

7.2 AI of Bot

Since the game will be a team game, so to have equal number of players on each side, we may require a bot. The bot will have to control its defense and attack units and deal with upgrading of powers. Following are the 3 components of AI of bot

- **Control Defense unit** : To defend its base, bot has to intelligently decide where to place towers. Also it has to decide which type of tower will cause maximal damage to opponent's enemy unit. So type of tower will be according to strength of opponent's attack unit.
- **Control Attack unit** : Bot will be controlling creep's strength and size of creep unit to attack on opponent's base. It will be sending creeps after regular intervals of time. And it will decide the strength and number of creeps according to opponent's defense. To implement this, **FSM** will be used. Input to this FSM will be opponent's towers attributes and states will be strength and size of of creep unit.

- **Purchasing Items** : The Bot will earn gold coins after destroying creeps or heroes of opponent team. These gold coins will be used for purchasing and upgrading defense and attack units. The Bot will upgrade a tower/creep if it is more effective in neutralizing powers of opposition wave otherwise it will send waves or purchase new towers

8 Testing Components

We describe here how will we debug each component :

8.1 Graphics

The graphics should work perfectly at good fps. So we will be testing the graphics separately first with large number of units in game and at high speed.

8.2 Sound

Background music is fine. But sound effects should be in sync with the game actions. We will need to test that mainly in the sounds part.

8.3 Networking

We will test networking part separately then integrate with the main program. To test we'll send continuous messages between computers on a test network. Latency should be as small as possible otherwise the game will not run properly. All the bad cases (connection lost, low speed of network, etc.) should be handled, no crash/unwanted behaviour should occur.

8.4 Artificial Intelligence

To test Artificial Intelligence of each unit of game, we will try different permutations for testing both defensive and attacking units.
Defensive unit can be tested by sending enemy waves of different strength to opponent's side.
Attack unit's intelligence for path selection can be tested by placing different towers along path.