Team RealTime- Jeffrey Weng, Yuanchu Liu (Leo Liu), Allard Peng

APCS2 Pd 3

UML DIAGRAMS

RevolutionGeometry (Driver class)

INSTANCE VARS

ArrayList<Unit> playerTeam;

ArrayList<Unit> computerTeam ;

Graveyard playerGraveyard; //Graveyards as a data structure are STACKS

Graveyard computerGraveyard ;;

Menu menu; //stores the means of managing units and currency through the interface

Boolean doOnce //a check to make sure only one unit is being trained at once

METHODS/FUNCTIONS

**Function of the class is to set up the field that the units battle on, and the menu that is the means of receiving commands from the player.

**Regulates the training and movement of all units on the field via draw()

**Increases rate at which player gains currency based on the number of miners

Menu

INSTANCE VARS

Button[] buttons; //stores the positions of every individual button, which comes from the Button class

int currency; //this is basically the amount of money you have

METHODS/FUNCTIONS

void loadMenu(); //puts the menu on the interface of processing

int buttonPressed(); //detects the button that was pressed with the mouseclick, returns the index that the button is in inside the Button[] buttons

int getCurrency(); //returns how much money you have

Button

INSTANCE VARS

```
 int xpos;
 int ypos;
 int button_r = 60; //size of the button on the screen
 String text; //what it says on the button
 int index = 0;//index of the button inside the Button[] buttons in Menu class
```

METHODS/FUNCTIONS

```
void display(); //displays the button on the screen according to size
//all accessors below
int getRad();
int x_position();
int y_position();
int getIndex();
```

Graveyard

INSTANCE VARS

```
   ArrayList<Unit> _stack; //all dead units go here
```

METHODS/FUNCTIONS

```
Unit dequeue(); //remove the unit that was last put in
void enqueue(Unit a); //put a unit into the stack
int size(); //accessor for size of _stack
```

Unit (superclass)

INSTANCE VARS

protected int life;

protected float speed;

protected int trainingTime;

protected int cost;

protected PVector position;

protected int damage;

protected boolean team; //is the unit on the player's team or computers

protected int y; //offset value to create a visual effect of depth

protected float attackRange;

METHODS/FUNCTIONS

abstract void drawUnit();

abstract void updateHealth();

abstract int identifier(); //what unit is this one specifically? Right now it should be 0, which means swordsmen, but there will be a lot more units soon

boolean isAlive() //very simply says whether this unit is dead or alive

int getLife() //accessor for life

void attack(Unit target) //the current target of the Unit that is being attacked

Unit selectTarget(ArrayList<Unit> enemyTeam) //finds closest enemy Unit

void move (Unit target) //the means of allowing a unit to move towards its closest enemy unit

void updateHealth() //updates a life bar that decreases in green area when life is lost

Swordsmen (extends Unit)

INSTANCE VARS

Static final int COST = 5; (subject to change as of now)

METHODS/FUNCTIONS

void attack(Unit target) // as stated in superclass Unit

Archer (extends Unit)

INSTANCE VARS

Static final int COST = 15; (subject to change as of now)

METHODS/FUNCTIONS

void attack(Unit target) //as stated in superclass Unit

Wizard (extends Unit)

INSTANCE VARS

Static final int COST = 5; (subject to change as of now)

METHODS/FUNCTIONS

void attack(Unit target) //attacks enemies using a priorityQ

Miner (extends Unit)

INSTANCE VARS

Static final int COST = 10; (subject to change as of now)

METHODS/FUNCTIONS

void attack(Unit target) //miner's attack method is just to walk around

Commander (extends Unit)

INSTANCE VARS

METHODS/FUNCTIONS

void attack(Unit target) //this is a special unit, the attack and movement of the Commander is guided by the player's controls. This will all be shown to the player in the gameplay

AI

**This is the class that makes the decisions for the computer on what to do to make the game challenging

METHODS/FUNCTIONS

static int choose(ArrayList<Unit> playerTeam, ArrayList<Unit> computerTeam, int currency) //has to determine what is the smartest move: train soldier (non-miner unit) or train miner

static int unitStrength(ArrayList<Unit> team) //calculates overall strength of a team

static boolean saveUp(ArrayList<Unit> playerTeam, ArrayList<Unit> computerTeam)

ArrayPriorityQueue

INSTANCE VARS

int start;

ArrayList<Unit> _priorityQ;

METHODS/FUNCTIONS

void add(Unit x) // _priorityQ.add(x);

int peekMin()//helper method for removeMin

void changeTime()

Unit removeMin() //used by the algo for queuing troops for training

Unit removeMinH(Unit firstTarget) //find the unit w the lowest life and is within strike range: used by wizard for targeting

void setArr(ArrayList<Unit> a) //sets _priorityQ to the parameter: a method of copying lists into the priority queue

Giant (extends Unit)

INSTANCE VARS

Static final int COST = 20; (subject to change as of now)

METHODS/FUNCTIONS

void attack(Unit target) //as stated in superclass Unit

Nexus (extends Unit)

-represents the “castle” in the game

-set as a Unit to aid in targeting process

Methods:

Void updateHealth() //updates a life bar that represents remaining life of the castle