WEB1: RTS - Mid-Point Project Check

Team members: Kathryn McDonald, Manhing Lei, Tatyana Vlaskin

Main program:

http://web.engr.oregonstate.edu/~vlaskint/CS419_mid_quarter_progress/

- Choose "New Game" (Load Game is not implemented yet).
- Pick either Easy Game or Hard Game. Please NOTE Sprite sheet was updated and coordinates were not fixed in the java script file, so it does NOT work. **Only click HARD mode.**
- Both will have a control panel on the left and the map in the center.
- The control panel shows the gamestate: the progress of the towers for the player and the CPU, the amount of diamonds the player has accumulated, and the number of robots the player has.
- Under "Buy" on the control panel, a user buy robots by clicking the buttons. After buying a robot, the diamonds will be deducted and a new robot will appear.
- Use Arrow keys to move the the destructor around the map [left, right, up, down]
- Also use the spacebar to attack someone. We were having a hard time with the animation of the destructor, so the decision was to attack with the baseball bat and baseball ball. When the player clicks the spacebar, the baseball bat is lifted and a baseball ball is thrown on the enemy. If the baseball ball bits the enemy - the enemy dies. We have a hard time removing the enemy from the canvas - we are still working on this. To make sure that the space bar works and the baseball ball flows, the destructor needs to have the bat down. This can be accomplished if you press one of the arrows.
- We are detecting collision using the following function:

```
//collision detection
function collision(object1, object2) {
   return object1.drawX <= object2.drawX + object2.width && object1.drawX >=
object2.drawX &&
        object1.drawY <= object2.drawY + object2.height && object1.drawY >=
object2.drawY;
}
```

where object1 is the baseball ball and object2 is the enemy object

- We are having a hard time removing an object from the canvas. We tried to use this function:

```
function die(){
        //enemy2.ctxEntities =null;
        delete enemydestructor.ctxEntities;

}
```

but for some reason object is not removed

- However last minute before submission, we were able to accomplish collision detection - see Link: http://web.engr.oregonstate.edu/~leima/CS419/collisiontest/collisiontest.html

The functionality in this test program to be used in the main program are: having an object melee another object with a set interval between melees and keeping track of the opponent's health points with each melee.

Instructions:

- After page loads, hit the spacebar once to put the character into action. The collision detection and the ability to melee is based on proximity to the opponent. The meleeing will stop once the opponent's health points go down to zero.
- Additionally, you can move the character with the arrow keys to see when the character is far enough away from the opponent, no meleeing will occur.
- This needs to be incorporated in our code game.

Work in progress:

1. Incorporate collision detection and disappearance of the enemy from the screen.

2. Functionality: Save game variables to MySQL; and load game variables from MySQL
Link 1 of 2: http://web.engr.oregonstate.edu/~leima/CS419/database/test1.php
Link 2 of 2: http://web.engr.oregonstate.edu/~leima/CS419/database/loadtest1.php

The functionality in this test program to be used in the main program are: keeping tracking of new objects and getting the X & Y coordinates of each. This data will be saved to a MySQL database when the user saves a game.

Instructions:

- Go here: http://web.engr.oregonstate.edu/~leima/CS419/database/test1.php.

- Click the moving ball object to create new display objects (robots).
- After creating a few objects, click "Save Game". This mimics a situation where a user is in a middle of a game and wants to continue later on.
- To load previously saved data, go here: http://web.engr.oregonstate.edu/~leima/CS419/database/loadtest1.php
- Click "Load Game" to recreate the previous display objects at their saved positions.
- You may continue adding more robots and repeat the saving, but this part is a little buggy.

3. Control the time of how fast the building is made based on the number of builders
4. Control the positioning of the miners, builder and make sure that are are not placed on top of each other

Conclusion:

Most likely we will simplify the original AI side of the game, thus the whole game. Instead of competing for the tallest building, the user will try to build a building within a reasonable time and the AI side will keep producing enemy destructors to destroy players objects. The player will run around killing AIs destructors with the intention to build a building within an allocated period of time.

References:
Will be provided upon request