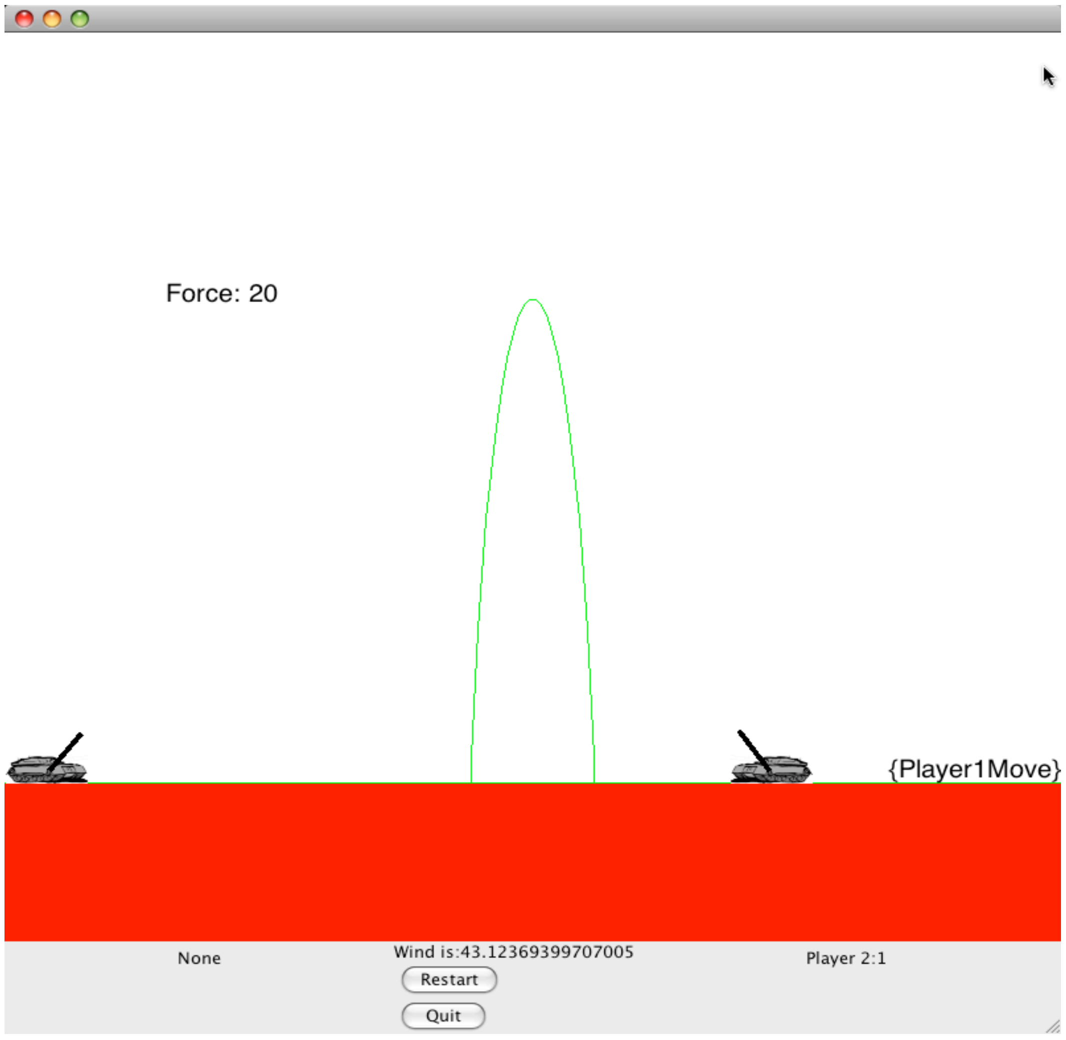
CS 211 Project 1 Assignment

Instructor: Dan Fleck, Ricci Heishman
Project: JMortarWar using JGame

Overview

In this project you will use JGame to create a small game. This game consists of two players firing at each other over a hill. This is an example of one of the earliest types of computer games. See http://en.wikipedia.org/wiki/Artillery_(computer_game) for more examples of these types of games from very early to recent.

In our game we'll have a simple flat ground and a hill in the middle. The hill will be drawn at different heights each round. Additionally, wind will be present that changes how the mortars (bullets) move through the air.



This project will build upon labs 2,3,4 which you have completed. You may (and should) reuse code from those labs to complete this assignment.

In this project we will give you many of the classes you will need to complete the project.

Game Play

This is a turn based game where each player sets a specific force and angle. Once both are set the tanks fire. During each round both players may score, one may score, or neither score. As seen in lab 4, JGame moves through a series of states. Each state does something and eventually transitions to a new state.

Game States

Start
This state begins the game. All it should do is wait 30 milliseconds and transition to the player 1 state. You could put up a nice message here if you want, but it's not needed.

Player1
Left-Right arrow keys adjust the angle.
Up-down arrow keys adjust the force.
Space bar locks in settings and moves to player2 state

Player2
Similar to Player1 except the space bar locks in settings and moves to the firing state.

Firing
At the beginning of the firing state two MortarRounds are created using the settings for each tank. Every frame should update the mortar rounds' positions. After both MortarRounds are no longer alive, the game moves to either the NewRound state or Player1 state if no one was hit.

NewRound
A new round is begun by creating a new Hill and changing the wind speed then moving to the Player1 state.

Requirements

1. JMortarEngine

    a. You must have at least the 5 required states as described above.

    b. You must have one class JMortarEngine that extends JGEngine. This is the class that manages game states

    c. You must have a GUI that looks like the one shown above. This is composed of two parts. The bottom panel from Lab 3 and the top panel is your JMortarEngine.

    d. Quit and Restart buttons should work as designed in Lab 3.

    e. Space bar should end the player's turn and move either to the next player or to the firing state (as appropriate)

    f. The wind speed must affect the projectile as described in Lab 2.

    g. The wind speed must change randomly everytime a new round begins. The wind speed should be between -50 and 50.

    h. Audio should play when the mortar rounds are fired. An audio file will be given.

    i. Force text should update for each player during their turn as they press up/down arrow keys.

2. MortarRound

    a. Mortar rounds must stop if they contact the hill

3. Tank – **This is given to you, so do not worry about it**

    a. The turret on the tank should be centered on the tank and rotate as the player holds down the left or right arrow keys during their move.

    b. The turret should have a minimum angle of 0 and a maximum of 180. (It must not be able to point “down” for example.

    c. The tank on the left side of the screen must be facing RIGHT and the tank on the right side must be facing LEFT

    d. The tanks must be position randomly. Left tank between 0 and 40% of the playing field width (pfwidth). Right tank must be between 60% and 100% of the playing field width.

    e. Audio must fire when a tank is hit

Hints

Notes: See the JGame notes posted to Blackboard. You’ll also find files to download for setting up the media and a Main.java file that is complete if you want to use it. (It is from Prof. Fleck’s implementation… use it if it helps.)

The FULL source framework in Netbeans can be downloaded from here. This fully runs, BUT does not include much of the code you need to do to make this work! You must un-Jar it into the appropriate place in your Netbeans project.

http://cs.gmu.edu/~dfleck/classes/cs211/fall08/projects/project1/src.jar

Classes: You can use however many file you need. I had these classes:
JMortarEngine – the engine. **You need to develop**.
MortarRound --- This is the bullet. **You need to develop**
ProjectileVelocity – from lab 2. **You need to modify (most likely)**
GameState – from lab 3. **You may need to modify.**

Files given

Main.java – Given in download
media.tbl – Given in download
Hill – Given in download
Player – Given in download

Turret – Given in download

Bonus Points

Bonus points will be awarded for each feature you add from the list below:

1. Display the windspeed graphically as an arrow at the top of the screen where the length of the arrow is determined by the wind speed and the direction of the arrow is the wind direction.
2. Restart button actually creates a new round (changes hill and windspeed)
3. Create a game mode where the bullets “bounce” off the left/right/top of the screen. (This must be selectable… I want to play the game with and without this feature.)
4. Any cool features from the best artillery game ever – Scorched Earth -- http://www.dosgamesarchive.com/download/game/144 (WARNING: Prof Fleck lost all of 10th grade due to this game). Seriously though, if you see fun features… let me know and we’ll probably give you credit for implementing them. There is also a new 3D version, but I’ve never played it.

What to turn in:

1. A Jar file containing all Java source code and compiled code. All files you create must contain the standard class header for CS211.
2. A list of any resources you used. This should include any sample files you got from the JGame website. It’s okay to use them, but I want to know which ones you reference and copied code from. (I assume you will use all the resources provided in this assignment document and the JGame notes posted by Prof. Fleck. No need to reference those.)
3. A list of any discrepancies between your code and the requirements. What could you not complete?
4. A list of any bonus features you completed

NOTE: This is an individual project. You must work alone on this and should not consult any unapproved resources. If you have the slightest doubt about what is allowed, please ASK YOUR PROFESSOR! Don’t take a chance! **Cheating will be dealt with harshly!**

Approved resources:

- JGame website (http://www.13thmonkey.org/~boris/jgame/)
- Blackboard for CS211
- Any files/information posted on Prof. Fleck or Prof. Heishman’s websites
- Discussions with CS211 TAs or professors are allowed and encouraged.
- Sun’s Javadoc’s API
- Our textbook

Grading Rubric: This assignment is worth 10 points and will be graded based on the following rubric:

Area	Exemplary	Competent	Developing	Points
Class Header	All header components are present, with references and comments that accurately support the state of the file.	All header components are present, but references and comments are incomplete or nonspecific.	Header is missing or only partially present, and references and comments are vague or unmeaningful.	__ / 2
Coding Style	Code implementation utilizes appropriate white space, self-documentation techniques and non-obvious comments.	Code implementation exhibits minor alignment or spacing problems, some comments are missing or redundant.	Significant alignment and spacing problems, comments are generally missing or sporadic.	__ / 2
GUI Layout	The GUI looks exactly as described in the assignment with no visible differences.	The GUI has some visible differences from the assignment, but does contain all required components.	The GUI does not look like the assignment states or is missing components.	__ / 10
Game Play	The game has all required functionality and does not have unintended functionality due to errors in implementation.	The game mostly works, but there are some missing /non-working features. Examples of non-working features are things like the projectile doesn't move correctly. The windspeed does not change every round, etc… Or there are parts of the game that should not happen (e.g. bullet goes through hill, etc…)	Major game features are missing or function incorrectly. Overall the game is generally unplayable. For example, the screen is able to be shown, but nothing happens, no game states are entered, etc…	__ / 24
Supporting information	Everything from the section "What to turn in:" above is present and clearly written.	Parts of "What to turn in" are missing or incomplete.	Most of "What to turn in" is missing or all of it is incomplete.	__ / 2