CS 211 Project 2 Assignment

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

Overview

Project two will build upon project one. In project two you will start with project one and add four new features and diagram your system.

1. Creating an intro menu screen
2. Tracking and maintaining a high score list
3. Allowing players to move left/right
4. Making the games time based

In addition to the features you must provide a state machine drawing showing all the states in your game. This can be done in any drawing tool as long as it looks nice. If you'd like, you can install the UML plugin for Netbeans and use Netbeans to draw your diagram.

In this project you're expected to figure out a little more than the last project. A general specification is given below, but not specific coding guidelines (e.g. you need to determine any new states that are needed, etc…).

As with all problems, I would break it down into small chunks and try to finish each one in turn.

Feature Descriptions

Creating an Intro Screen

You should create an intro menu screen that looks like this:

Each option should allow uppercase or lowercase characters.

- P or p should begin a game.
- S or s should show the high score screen
- Q or q should exit the program

High Scores

Using Lab 6 you must create and maintain a high score screen. This mean you must

- Load the high scores data file when the program starts
- Track which player wins the game and what their score is

- Allow the user to type in a name for the list. They should be able to actually type, not select a character from a list or something like that. The typing should be in the game, not on the console window (not like you did in Lab 6).
- Save the name and score in the data file
- Display the high scores and names with the highest score at the top of the screen. You should only display the top 10 scores.

HINT: You will need to modify the classes from Lab 6 to implement this feature in JGame.

Making the games time based

In order to have different scores, the game must now be time based instead of ending at a specific score. You must implement timer to make the games last approximately 2 minutes.

At the end of two minutes the game should automatically end. To simplify the coding, the winner will always enter their name into the high score list and have it saved in the **scores.dat** file. It may not show up on the high score screen because you're only going to display the top 10 scores.

Allowing players to move left/right

To make the game more interesting, you must also add the ability for players to move left and right during their turn.

- The players should move left with the ',' (comma) key
- The players should move right with the '.' (period) key
- The players must not be allowed to move through the hill
- The players must not be allowed to move off the left/right of the screen

HINT: You'll need to modify the Player class for this and use methods in JGObject to check for collisions and if you are on the screen.

HINT 2: You do not need to be super-accurate with collision detection. Just checking if the bounding box of the tank hits the bounding box of the hill is fine.

NOTE: The keys were chosen because when you look at the key on the keyboard, it means left is < and right is > .

State Machine

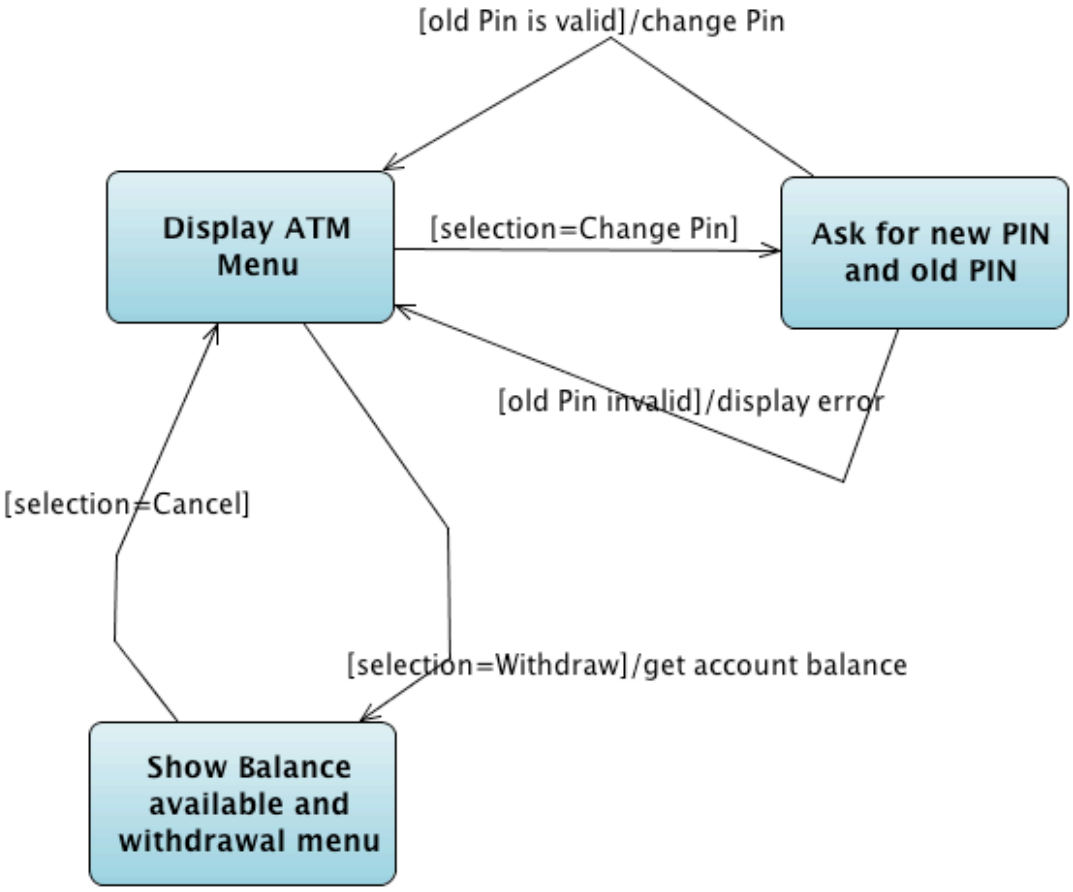
The state machine diagram should show each state in your system as a box (or oval). Between the states there should be transition arrows showing when you move from one state to the next. Transition labels may have a guard condition explaining why you transitioned and may have an action that is taken while you transition.

If more than one arrow leaves a state you must add a guard condition onto the arrows so we can tell which transition is followed.

Guard conditions are in [ ] and actions taken on the transition are after the condition. You do NOT need conditions and actions for every transition. Keep it simple and do what seems to make sense to you.

In JGame actions on the transition would be things in the startXYZ methods and the states themselves is the general idea of what is happening in doFrame/paintFrame.

See example for part of an ATM machine:



Bonus Points

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

1. Making the game simultaneous instead of turn-based. This means both players can be moving and firing all at the same time. To do this you'll need to use different keys for player 1 and player 2. (+7pts)

2. Make the terrain height random, so the players may be at different heights each game. They still need to be able to move left/right though! (+10pts)
3. Make the tank be able to either move or fire during a round, but not both, or in some way limit the ability to move so players are penalized for movement, to reduce movement every round. (+5)
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
5. A state machine drawing showing states for your implementation.

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 40 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
Intro Screen	The intro screen is present and the menu works correctly.	The intro screen is present, but some menu options don't work. (Or menu options do not support upper and lower case)	The intro screen is missing or all menu options don't work.	__ / 6
High Score List	The High score list functionality is exactly as specified above.	The High score list functionality is missing small parts, but still functions.	The high score functionality is not able to complete much (if any) of the features described.	__ / 13
Left/Right Movement	Players can move left/right and are blocked by the hill/screen edges.	Players can move left and right, but collisions are not checked (they can go through the hill or off the edge.	Players cannot move left or right.	__ / 13
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.	__ / 1
State Machine drawing	The state machine drawing is clear and depicts the game correctly.	The state machine drawing is clear, but is missing states that are present in the system.	The state machine drawing is missing, very unclear or very incomplete.	__ / 3