CS 211 Lab Assignment

Instructor: Dan Fleck, Ricci Heishman
Lab: Using JGame to implement Rock Paper Scissors game

Overview

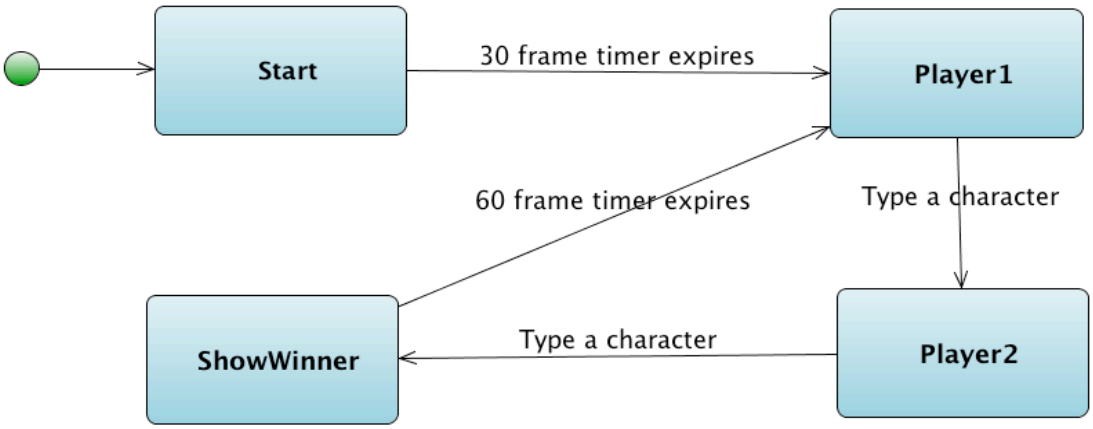
In this lab you will begin learning about JGame and state machines. JGame uses states to determine what methods get called and what happens during the each frame. See the JGame tutorial for information about how to use states in JGame.

In this lab you'll implement the game "Rock Paper Scissors". This is a very simple game with two players who each choose either Rock, Paper or Scissors. The rules are:

Rock wins against Scissors,
Scissors wins against Paper
Paper wins against Rock

If you both pick the same value, it's a tie and you play again. If you get good at this game (and truly I have no idea how anyone is better than anyone else) you can make it to the world championships! http://www.worldrps.com/ (Please mention me in your acceptance speech if you win).

In this game you should implement the following state diagram for game play. Each state is shown in a box, and the action that causes the system to transition from one state to another is noted on the arrow between states.



Start state: Displays text on the screen "Get ready…"

Player1 state: Displays text on the screen: Player 1: Choose (r)ock (p)aper (s)cissors

Player2 state: Displays text on the screen: Player 2: Choose (r)ock (p)aper (s)cissors

ShowWinner state: Displays text on the screen determined by the previous choices made by the players.

One of the following lines will appear:

Same answer: no score

Invalid choice: Player1: <<char player 1 chose>> Player2: <<char player 2 chose>>

Player1 wins: Paper beats rock

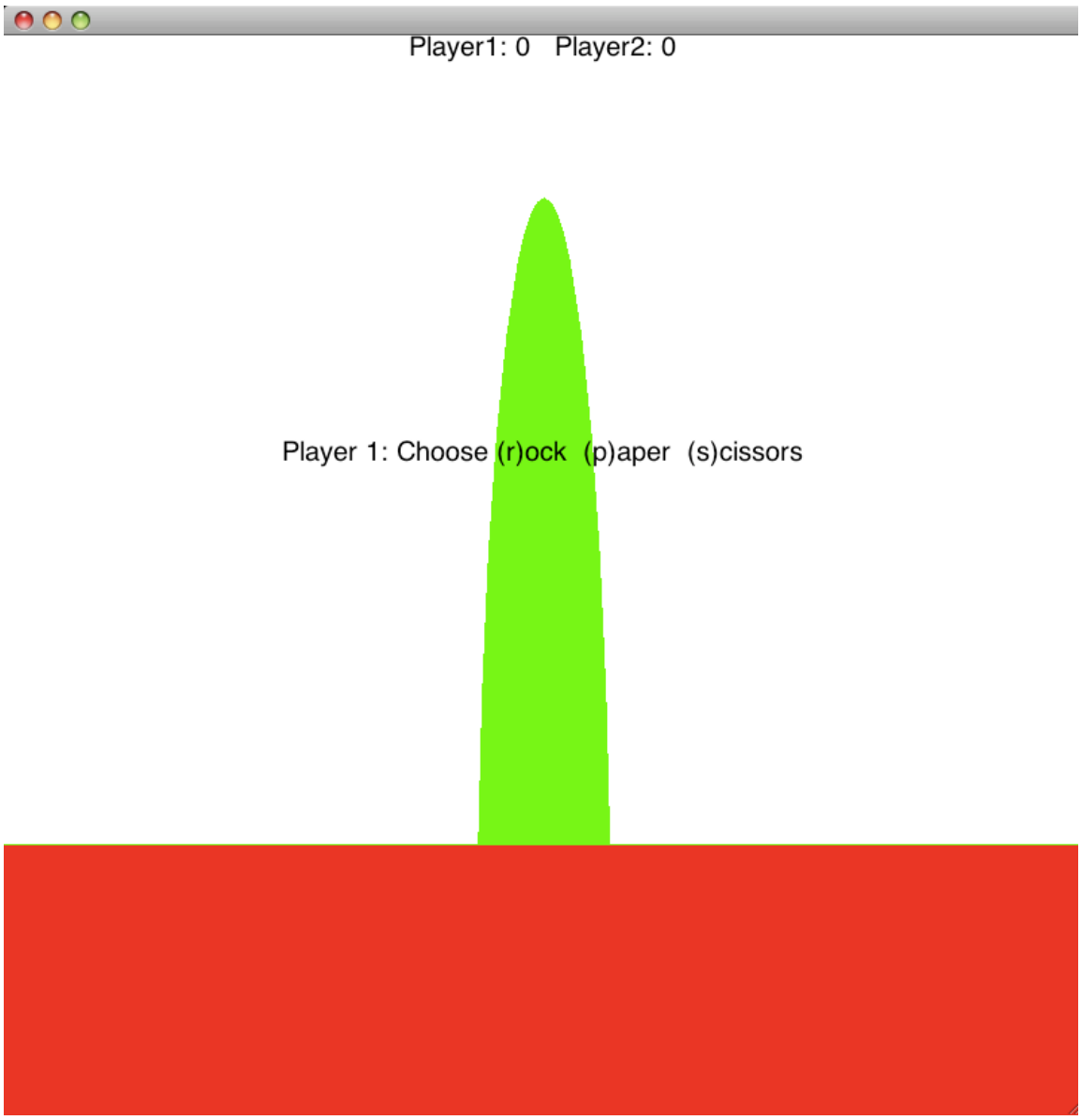
Player2 wins: Paper beats rock

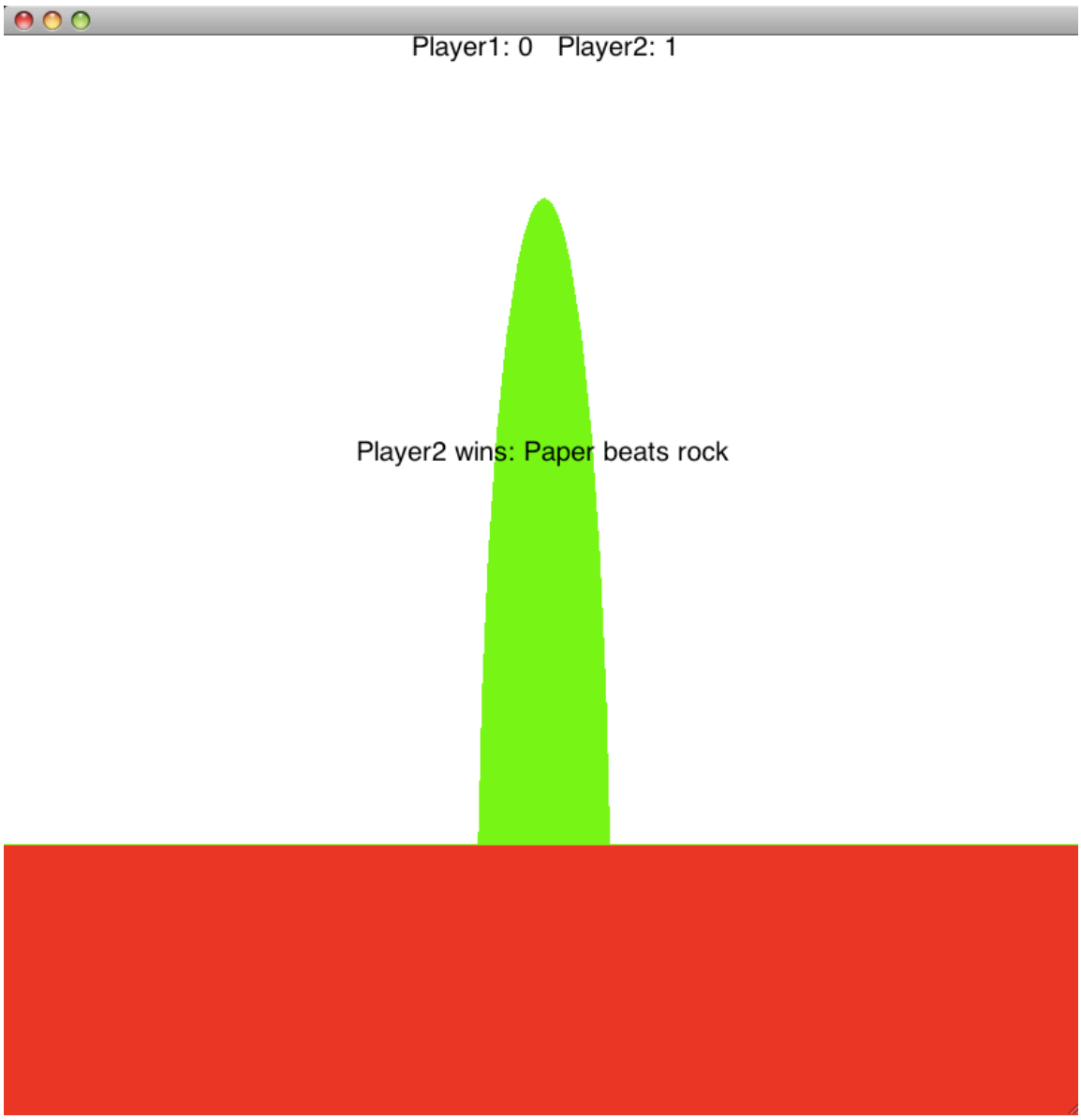
(repeat for all possible choice combinations)

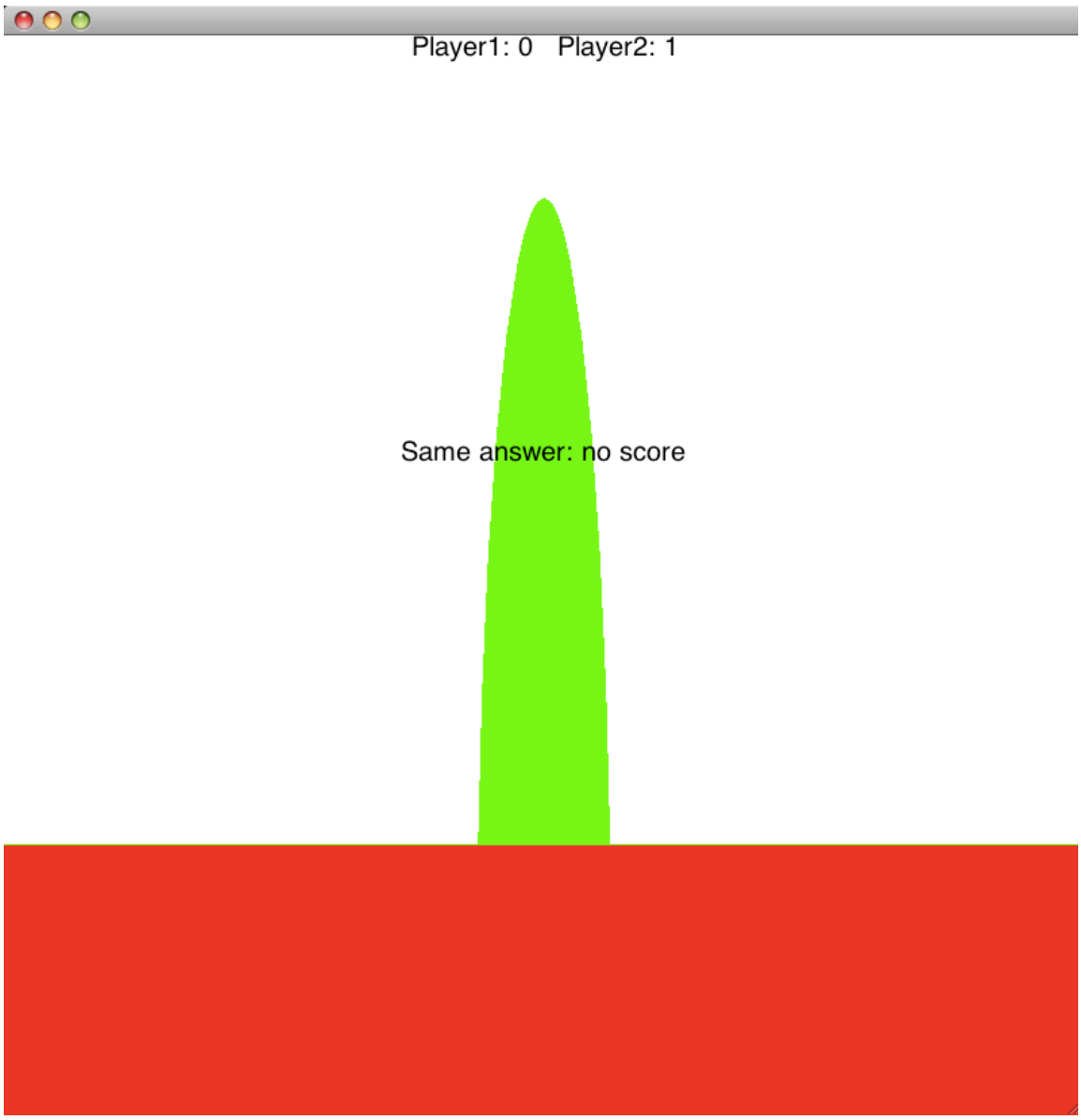
In the two lines above,

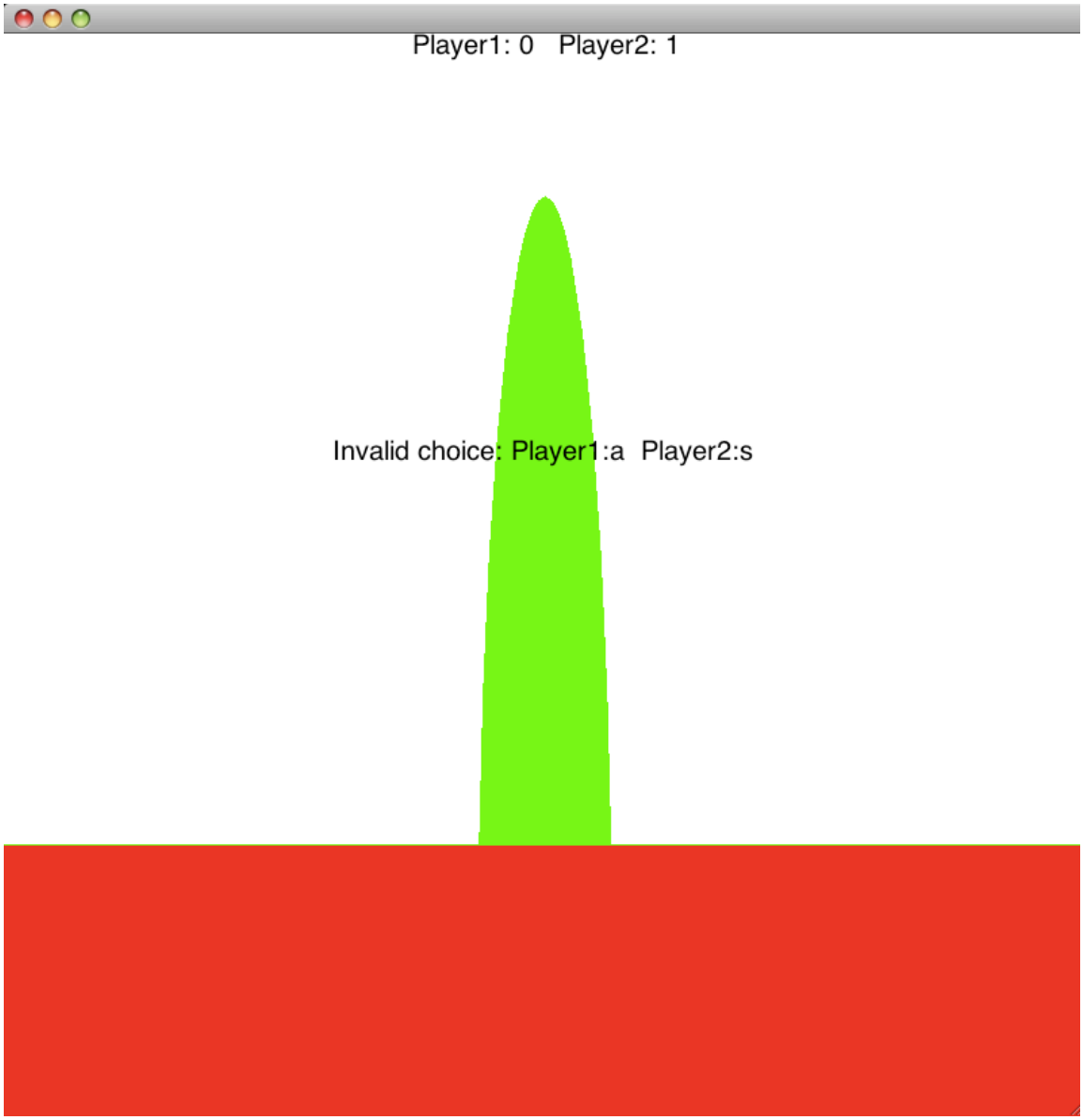
```
Player1 wins: Paper beats rock    // This means player 1 chose paper
                                  // and player 2 chose rock

Player2 wins: Paper beats rock    // This means player 2 chose paper
                                  // and player 1 chose rock
```

For this project you may (and probably should) start with the sample code given for the project. Some sample screens for some of the game states are below:


Player1: 0 Player2: 0
Player 1: Choose (r)ock (p)aper (s)cissors


Player1: 0 Player2: 1
Player2 wins: Paper beats rock


Player1: 0 Player2: 1
Same answer: no score



Requirements

1. The GUI should look as shown above.
2. The system shall include all states described in the state machine diagram above.
3. The system shall transition from one state to the next under the appropriate conditions
4. The system shall display and update the score of both users
5. The system shall tolerate invalid input by displaying a message to the user and moving back to the player1 state

Hints

- Reading the JGame tutorial will help you (the one posted in Blackboard)
- Using the code from the project will be VERY helpful
- If you can't see the application for some reason, try changing the boolean value in Main.java . It will create two windows instead of one but may work better. Let us know if you need to do that.

What to turn in:

1. A Jar file containing all Java source code and compiled code you used. (Include code that was given by the professor and your own code. You do NOT need to include the JGame library code (jgame-all.jar).

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.	__ / 1
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.	__ / 1
Game Play	The game functions as described including states changing appropriately and scores being updated correctly and invalid input being handled.	States change, but not in the correct order or correct time. Scores are missing or not updated. Invalid input crashes the program or allows one user to score.	States are missing. Game crashes under invalid user input. States do not transition.	__ / 5
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.	__ / 3