Multi-Player Pac-Man

GROUP 14 (E/13/183, E/13/420)

ABSTRACT

This report contains the design of the multiplayer Pac-Man game. It includes game logic (Back end) and front end implementations.

CONTENTS

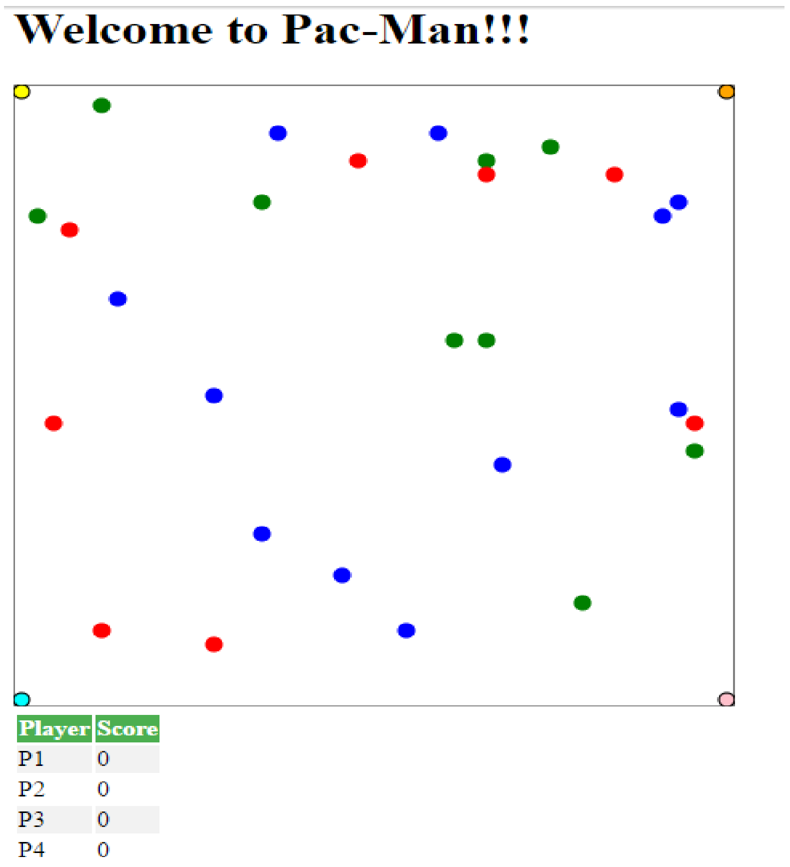
FIGURES

INTRODUCTION

Here, we developed a multiplayer game similar to Pac-Man. Play begins once four players have join in. Grid contains colored dots. Red, green and blue dots count for one, two and, four points. When two players collide, they lose three points each and collided players positions are changed to their initial positions. Players move on a two-dimensional grid using arrow keys. Here, we made a backend logic to maintain the current state of the game.

As each player logs in, the server initiates an SSE response to their browser. Once all four players have joined they are initially placed at the four corners of the grid. The server also generates a random distribution of red, green and blue dots on the grid.

On each keystroke the client sends an HTTP POST with the keystrokes represented by the strings “UP”, “DOWN”, “LEFT” and “RIGHT.” On receiving a keystroke, the server broadcasts a new event to all clients giving the updated state of the game. Event data consists of a JSON object containing integer grid coordinates denoting the position of the remaining dots and, players’ scores and coordinates. Player and dot coordinates may appear in any order, but the game front-end may assume that they are valid.



Welcome to Pac-Man!!!

Player	Score
P1	0
P2	0
P3	0
P4	0

EXISTING CLASSES

There are four classes in our game such as Dot.java, GameBoard.java, Player.java and UpdateGame.java(Servlet).

In Dot class, it sets the parameters of dots which are color and coordinates of dots.

In player class, it contains information for particular player. According to the keystroke moving, there is a function(public void move(int keystroke)) to calculate x ,y coordinates accordingly.

In UpdateGame class, it contains session handling part for four players (In doGet method, it allocates each session for particular user). In doPost method, it detects the keypress and update the players locations accordingly.

In GameBoard class, It contains collision handling part, score updating and updating the game grid according to the game logic. It also initiate the initial positions of four players, randomly generate food points and also it creates the complete json object.

CONCLUSION

Here, we generated a similar response format from server side and client side for parse the JSON and draw a canvas based on the coordinates from server. Game begins only four players are connected. Players move on the game board using keystroke. When two players collide, they lose three points each those collided players positions are set to the initial position. The game ends when all dots on the board have been collected.

REFERENCES

[1] http://docs.oracle.com/javaee/7/api/javax/json/package-summary.html

[2] https://en.wikipedia.org/wiki/Pac-Man

[3] http://gafferongames.com/networking-for-game-programmers/what-every-programmer-needs-to-know-about-game-networking/

[4] http://ithare.com/tcp-and-websockets-for-games/