Exploring Empirical Modelling and Web-Enabled EM Tools

1060969

Abstract

This paper explores web-enabled EM tools through a classic game development (Battleship using JsEden). In this scenario, empirical modeling principles introduced three new concepts: observable, dependency and agency. These concepts give a way of "thinking-with-computer". The basic aim of this paper is to explore how to adapt EM with web-enabled EM tools and develop a classic game such as Battleship using JsEden

1 Introduction

The principles of empirical modeling provide a way to go beyond classical programming for developing computer programs. In addition, the key concepts of EM are: observables, dependencies and agents. In this paper, the concepts of EM with web-enabled EM tools will be investigated in accordance with a classic game 'Battleship'. Web EDEN is the first web-enabled version of EDEN introduced in 2007, but there were few potential disadvantages of Web EDEN. However, a newer version of web-enabled EM tools called JsEden, has been developed in 2010. It is based on a JavaScript implementation of EDEN and it eliminated the major disadvantage introduced by Web Eden. This paper will explore JsEden and its advantages.

2 Web-enabled EM Tools

The first web-enabled EM tool is called Web-EDEN, it was introduced in 2007. Web EDEN has several disadvantages. The major disadvantage is load-balancing between client and server. This is because EDEM models run at the clients are sent to the server for interpretation which produces a large network load. Thus the interpretation is slow. However, JsEden has resolved this issue by implementing EDEN with JavaScript, although this is still in prototype stage, but it deals with all the issues of Web-Eden and provides several advantages over Web-EDEN. Given that JsEden is a browser based implementation of EDEN, thus, it addresses the issues of portability and provides capability of cross-platform. JsEden has improved the portability to a greater extent by implementing EDEN using JavaScript. However, portability is addressed by following the principles of EM. In this scenario, the major concepts such as observables, dependency and agency are not machine-specific terms; they are related to experience that comes from interacting with computer during the construction of models.

In addition, the traditional implementation of EDEN has relied on invoking procedural actions triggered by observable changes. Additionally, drawing interface in JsEden has to be explicitly programmed and point on the display has to be redrawn to maintain the dependency between internal state and their visual counterpart. Since JavaScript is primarily used for visualization management in browsers, thus maintaining dependency in JsEden is easy to implement. In is provided while drawing the interface and visual display of the game (Battleship).

3 Battleship

Battleship is a well-known guessing game that is played by two players. It was first introduced as pencil and paper game. The game is played on four grids, two for each player. Usually it is 10x10 square grids. Each player has a grid to arrange ships and records shots of the opponent and the other grid records his own grid. Before the game begins, players arrange their ships. They can be arranged vertically or horizontally. Each ship occupies at least two consecutive squares on the grid, number of consecutive squares representing different type of ships, for example a destroyer occupies two consecutive squares and a battle ship occupies four consecutive ships. This is to simulate the real-world situation for

the size of the ships. In each turn, players can fire at any one of the position, and tell the opponents which cell he fired at, and the other player will report if it hits part of his ship or missed the shot. When one of the players looses all their ships, the other player wins the game (Wikipedia, 2012). An image of the game is shown below:

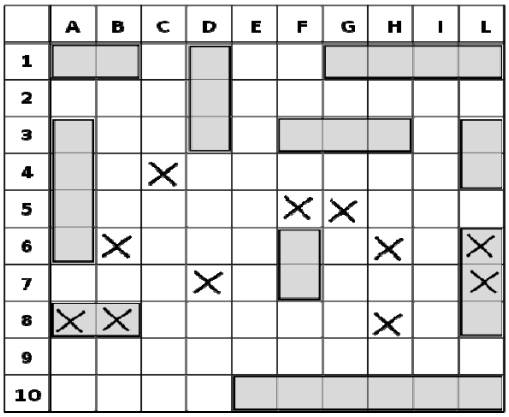

Figure 1Battleship Game, Image Source: http://en.wikipedia.org/wiki/File:Battleship_game_board.svg

In this project, the Battleship game will be implemented using JsEden. When the game starts, players can choose their opponents, it can be another player or versus computer using Artificial Intelligence (AI) (we will discuss AI of this game later). Once players selected their opponents, ships are preset, thus player cannot choose where they want to put their ship at. In each turn players can fire at any location that is not been shot at. Similarly, the display will update the board accordingly.

Moreover, each board in the game is represented by means of a 2D list in EDEN. The state of each cell is represented by numbers. With the advantage of JsEden, visual representation of the game state can be separated with the internal game state. In this scenario, a JavaScript function is responsible for updating graphical display of the boards. In addition, any changes to the game state will not be automatically updated in the display. Additionally, the model uses a set of buttons to act as agents, linking internal game state and visual representation of the game state by invoking functions from both sides. This is major advantage of JsEden over traditional implementation of EDEN such as tkeden. The reason behind this is, the actual internal values are more interesting to observe than those values used for graphical representation, because the entire visual representation depends on the actual game state. Hence JsEden provides an easier way to represent the internal values of the model and maintains the conceptual framework of EM. We can still observe changes of observables and dependency between them in a much better way.

4 AI algorithms

In this model, there is an option for the player to play the game against computer AI. The algorithm behind this is fairly simple. In each turn, there are two types of shots that can be used depending on previous shot. Trail shot will select randomly available cells to shot at. In this scenario, a list is used to keep track of all the available cells. This is to prevent firing at a cell that has already been fired. If the cell is part of opponent's ship, the location will be recorded. Once there is a hit recorded, in the following turn, the computer firing type will change to sunk shot. For this purpose an algorithm will be used to keep the record of location of previous turn, and shot at four nearby cells which are north, east, south and west of the recorded cell in the following turns. This is a search procedure that searches for the next part of opponent's ship. Once the second location is found, the direction is found which means in the following turns, we know which cell it is to shot at because ships are arranged either vertically or horizontally. Once we reach the end of the ship, number of hit cell is recorded and compared to the size of ships that will be on the board. For example, if only three cells are recorded as hit that means there will be one more cell left in the opposite direction because no ships are represented by three cells, only four cells or two cells. Hence, next firing location will be the opposite direction of the first recorded hit cells. Once the sunk hit is complete, trial shot mode will be resumed to search for the next ship.

5 Conclusion and future work

Although we have implemented the battleship game using JsEden, but we still need to make some changes in an attempt to improve the AI algorithm. If we make it more competitive, trial shot algorithm would be able to adapt a sequential scan of the board diagonally. Because the smallest ship occupies two consecutive cells, it means while scanning the board diagonally with trial shot, four cells can be scanned in two cells, which would improve the efficiency of trial shot.

In future, we also aim to make changes to game mode, for instance an AI vs. AI mode can be implemented and we can observe and examine the

behavior of the AI with different algorithms. Also a remote connection of game player can also be implemented by using JavaScript and HTML5.

The development of battleship game has uncovered some potential benefits of JsEden. As we discussed above, models can be simply visualized by using JavaScript and maintain the EM concepts by separating observables for the model and for visual representation of the model. This makes it easier to observe the dependencies and agencies between observables of the model. There are much more to explore in JsEden, the potential is huge because it takes the advantages of JavaScript and HTML5, thus distributed construction of model development is possible in JsEden as well as Peer-to-Peer connections for games like Battleship.

References

Wikipedia. (2012, January 22). *Battleship (game)*. Retrieved January 28, 2012, from http://en.wikipedia.org/wiki/Battleship_(game)

Battleship Game Algorithm Explained. Retrieved January 28, 2012, from http://arnosoftwaredev.blogspot.com/2008/06/battleship-game-algorithm-explained.html

Humt the Wumpus : An Empricical Approach Retrieved January 28, 2012, from http://www2.warwick.ac.uk/fac/sci/dcs/research/em/publications/web-em/01/

W. M. Beynon, *Reflections on JsEden and issues for the implementation of EM Tools*

Timothy Monks, *A Definitive System for The Browser,* 15th September, 2011