Generatrivia : board games generator

COLLABORATORS			
	TITLE : Generatrivia : board games generator		*REFERENCE* :
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Sebastián Gurin	October 26, 2011	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Abstract

Generatrivia : board games generator javascript framework for easily define, publish and play, board games like chess, trivias, ludo on he web.

1 About this document

This document is available in the following formats:

1. **pdf** Generatrivia : board games generator for the web
2. **html fragmented** Generatrivia : board games generator for the web
3. **docbook (sources)** Generatrivia : board games generator for the web
4. **download htmls** Generatrivia : board games generator for the web

2 Introduction

Generatrivia: define, publish and play your board games in the web.

Generatrivia is a html and javascript based application to define and generate boards games like chess, trivia, ludo, and games like that. It supports tools that game definitions can use, like dices, question and answers, pieces definition and initial positioning, etc. The users who uses generatrivia for defining a game, must define the game in a text file (javascript - json format), indicating the game board image, defining pieces (image, shape, size, color, name). If the game requires dices, the "dices tool" should be enabled. If the game require questions and answers like in a trivia, questions answers, categories should have to be defined in that file as well.

Once this game definition file is ready, the generatrivia javascript framework can be used for executing the game in a web page easily.

In generatrivia generated games, the computer do not control any game logic. The players are responsible moving their pieces and in some games perhaps creating the pieces too. Also players are responsible of controlling that others do not cheat while moving them!! This design has two main advantages:

- the games are defined by board image and rules outside computing. This gives a lot of flexibility for game definition.
- the computer ony replaces paper and dices, so the game experience is almost like the same as a paper made board.

Also, the framework contain a lot of known board game definitions, like trivias, different kind of chess, checkers, ludo, etc.

About the history of the project. The initial idea was to provide teachers with an easy mean of defining trivia games of school subject so students can play agains each other in a single computer in the classroom. I realice that no game logic programming is required for this, since students are able of counting points and moving their game pieces by theirself. Then I realice that several board games (the ones that do not require cards), can be sistematically defined for human players only.

3 Getting starting

No experience with programming languages is required for building your board games using generatrivia. The user should know show to work with files, directories and editing text files.

In this section, we will learn how to generate some simple game boards like ludo, trivia and chess. step by step.

3.1 Making the ludo game

in this section we will define our first game, the ludo game. Rules of the game can be fount at http://en.wikipedia.org/wiki/Ludo_(board_g
The ludo game consist on the following elements:

- A board
- 4 colored circular pieces of colors red, blue, yellow and green

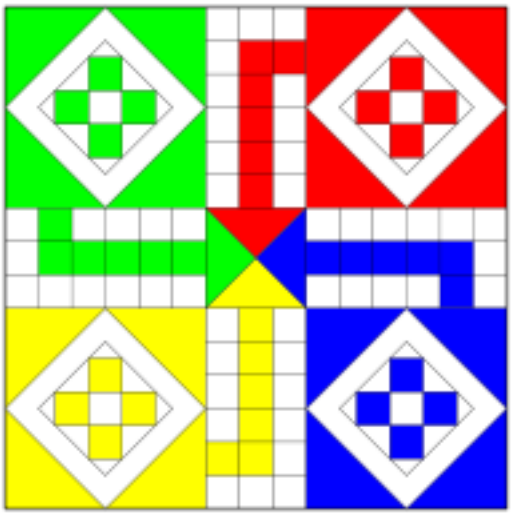
- A dice

- **Prepare files** First of all you have to download and decompress generatrivia somewhere. Your game will be use some of these files.

  Second, we will prepare your game's directory and files. For this, simply copy generatrivia's src directory to somewhere. Rename with your game's name, in this case "ludo". This directory will be where your trivia game will be stored. All following steps will be working in this dir.

- **The board image** Create or copy your game board's 'image from internet. Put in on your game's directory and call it ludoboard1.png.

  For our ludo we will use the following image:



- **Define the game in a text file** The last and more important step is to define all of our game's elements in a the game definition text file. For this, first create a file called data.js and open it with your favourite plain text editor [1].

  For our first version of the ludo game, this will be the file contents:

```
triviadata={
    gameName: "the Ludo game",

  boardImage: "ludoboard1.png",
  boardSize: "640,640",

  pieces: [
         {name: "red piece", code: "circle(10, 10, 10, 10)",
           size: "20,20", color: "#ff0000"},

         {name: "blue piece", code: "circle(10, 10, 10, 10)",
           size: "50,20", color: "#0000ff"},

         {name: "green piece", code: "circle(10, 20, 10, 10)",
           size: "20,20", color: "#00ff00"},

         {name: "green piece", code: "circle(10, 20, 10, 10)",
           size: "20,20", color: "#00ff00"}
    ]
}
```

  The file contents must be a valid javascript - json object string.

[1] In windows, you must use a plain text editor like notepad, and not ms word. Also, in be sure the file is called data.js and not something like data.js.txt

Let’s introduce the data.js file and explain what ’s inside briefly.

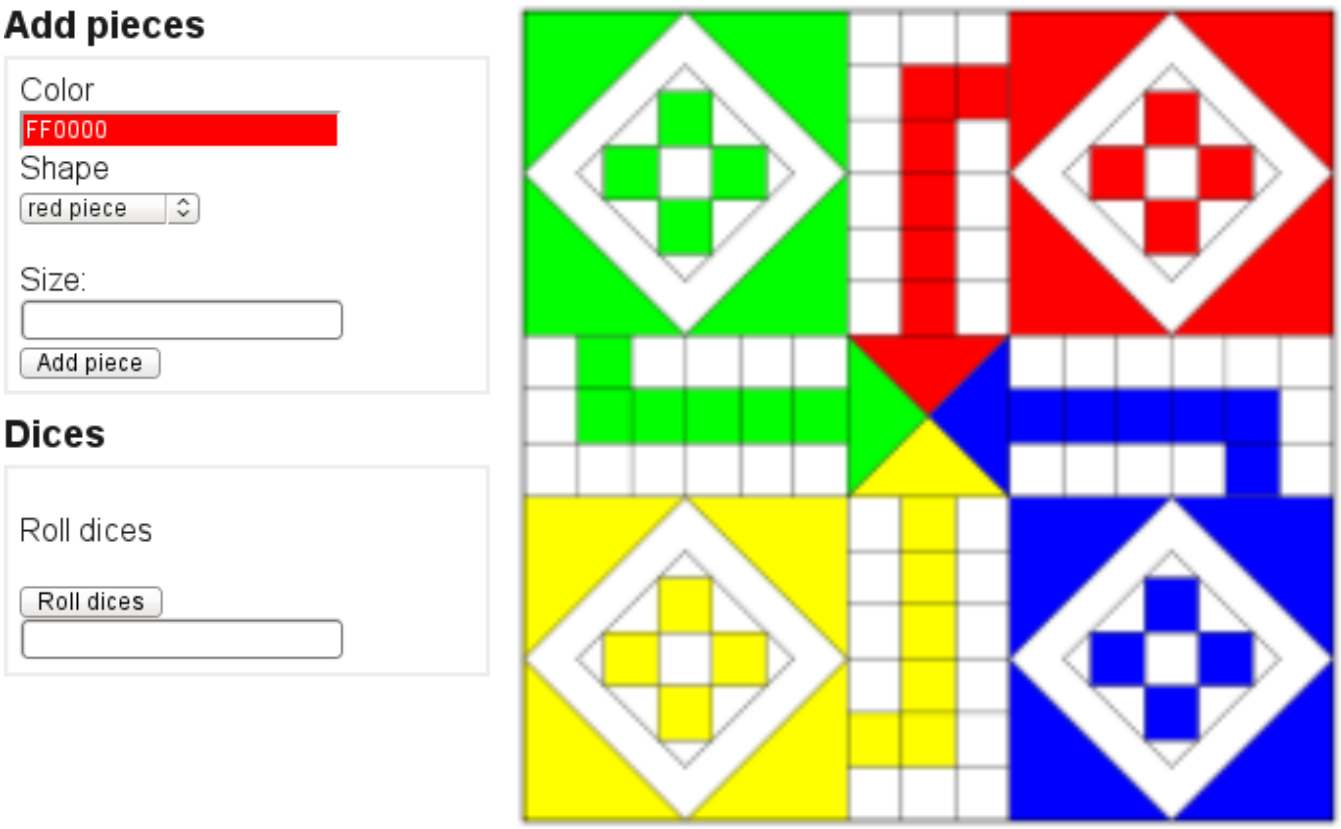
First of all the game definition file is defined using the programming language json. It is recommended but not mandatory that the user is familiarized with it: http://json.org/

As you can see, there are some properties defined as gameName, boardImage, boardSize and pieces. boardGame indicates what this game is called ("ludo"), and boardImage indicated the image file that will be used as game’s board ("ludoboard1.png").

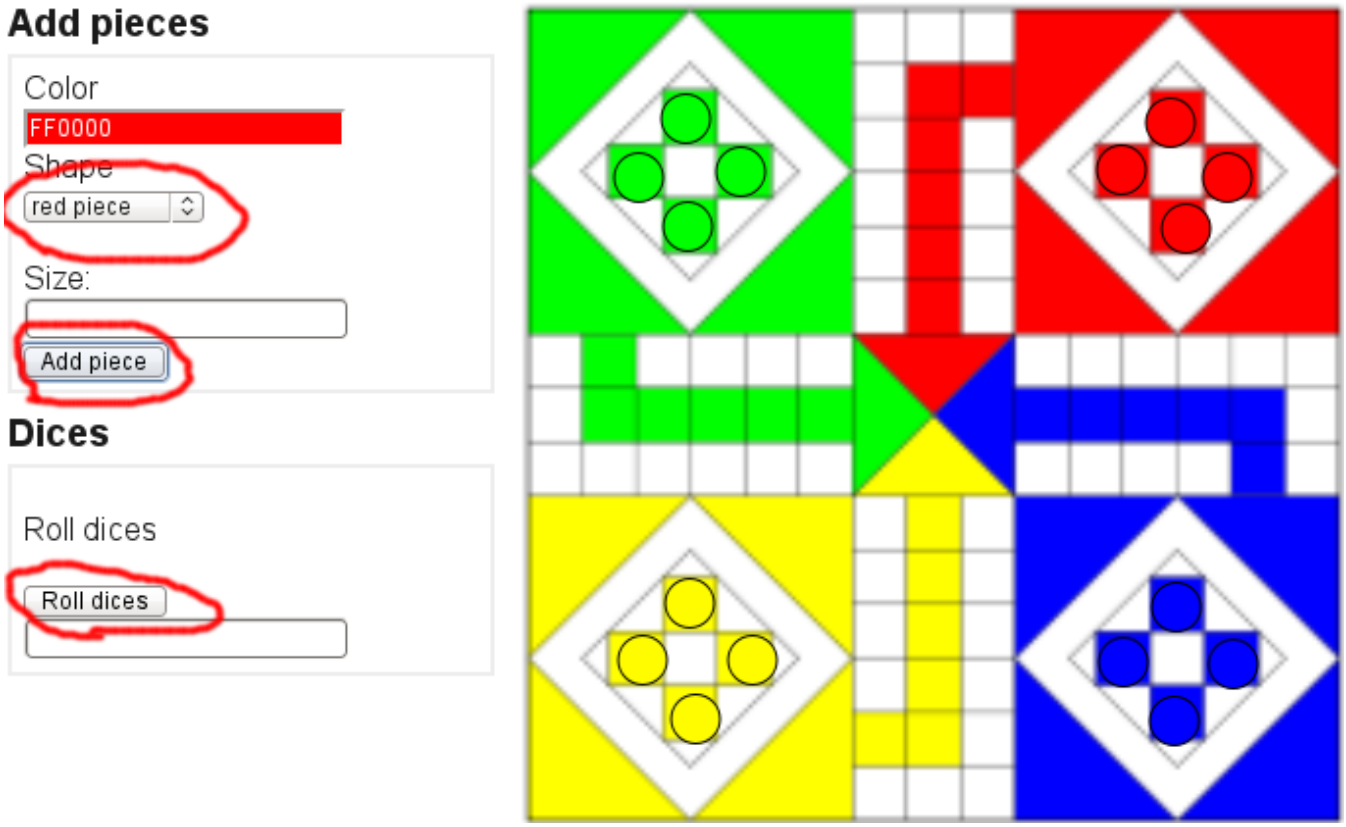
The more complex one is pieces. Remember we said the game ludo consist in 4 pieces of color red, green, blue and yellow. So the "pieces" property is telling just that. The following code defines a single game piece (circular piece of size 20 x 20 pixel and color red-#ff0000):

```
{name: "red piece", code: "circle(10, 10, 10, 10)",
    size: "20,20", color: "#ff0000"},
```

- **Ludo is ready** If there are no errors in the data.js file we are ready. Simply open the file index.html with your web browser and the game will start. and show something like this:



- **Now let’s play** In our ludo game, before playing the players must create the necesary pieces. Asuming will be 4 players, we create 4 pieces of each colors red, blue, green and yellow and move them in the board to their initial positions. For moving the pieces, just add them and then drag them with the mouse to desired board position. We perform this using the "Add Piece" tool at the left top corner:



step 3) If the data is valid, your trivia is ready!! Simply open generatrivia.html in your browser for start playing.

4 Built in game tools

The game is structered in the board, the pieces, and tools. Tools can be of several types:

- a game related activity tool like the dices tool, or the question and answer tool.
- a game developemnt heloper tool, like the piece position tool
- any user defined tool...

This section will teach you 1) give detailed information about using each built-in tool and 2) teach you how to program your own tools using the javascript framework's api.

4.1 Pieces tool

4.2 Dices tool

4.3 Questions and answers tool

4.4 Pieces position tool

This tool is usefull when defining games that consists in several pieces with defined positions in the board, like chess and checkers. With this tool, you can add all the games pieces manually using the pieces tool, each piece with the correct size, color and position, and then clicking the button. When you are ready, press the tool's button "get pieces porcentual info", and it will return all the "addPieces" code for data.js, so you only have to copy and paste it, for your game to show the initial pieces in the board.

5 Building custum tools

In this section we will show you how to build your own custom tools with an example. It is required to know the basis of javascript and html languages.

A tool consists on the following elements: 1) some user interface like input widgets like buttons, text entries, etc. that the player uses for controlling the tool. 2) some logic that do something related to the game. 3) some configuration in data.js for the specific tool

For example, the dices tool, consists on a button and text entry to controlling the tool (roll dices button and a text for showing the dice result). It has no logic related to the game. It can be configured using the "dices" property in data.js

We will define an abstract tool

```
/* a cutom tool (plugin) example : mouseCoords */
gt.tools.mouseCoords = {

name: "mouseCoords",

description: "a tool that shows absolute and percentual position "+
  "of a point in the board on mouse click",

/**this method will be called by the frameowork for loading the tool,
 * In this method the tools should create the markup in the
 * parent, register event listeners, etc.
 * the tool shoud *append* a new element to the parrent.
 *
 * @param parentEl - is a dom object or an id or jquery selector for this
 * tool dom parent */
init : function(parentEl) {
   //create an check box that will be our only tool control.
  $(parentEl).append("<input type=\"checkbox\" id=\"mouse-coords-tool-button\">get click  ↩
      board coords</input>");
  //add a mouse click listener in the paper
  $("#trivia-paper").click(gt.tools.mouseCoords._mouseCoordsMouseHandler);
},


/** this function indicates if this tool shoud be showed according to a game's data.
 * This method will be called at least one, so this way the tool can define custom
 * game's configuration for indicating if tool should be enabled, or passing other data. */
isEnabled : function(data) {
  return data.mouseCoordsToolEnabled;
},
/** internal mouse click handler */
_mouseCoordsMouseHandler : function(e) {
  //if the checkbox is checked, then show click's  coordinates
  if($("#mouse-coords-tool-button:checked").size()>0) {
    var pos = gt.getPositionFromEvent(e),
      posPercent = gt.getPointPercent(pos);
    var s = "Click position in the board:\n"+
      "absolute: "+pos[0]+","+pos[1]+"\n"+
      "percentual: "+posPercent[0]+"%,"+posPercent[1]+"%";
    alert(s);
  }
}

};
```

Generatrivia is made using html, javascript, css and these main javascript libraries: 1) jquery.com for general javascript toolkit, 2) jqueryui.com for general gui, 3) raphaeljs.com for drawing and graphics.

So when developing your own tools, you can use any of these toolkits for accomplish your purposes.

Note
Generatrivia framework api will be detailed in the following section.

6 Generatrivia framework api

When writing your own generatrivia tools, it is useful to be familiarized with the api of the framework.

6.1 gt.initGame()

The main method for creating a game. It must be called from the html document for starting the game. If you are writing your own html, make sure all necesary game scripts and css are loaded before starting the game, using jquery ready() function like this:

```
<script>
$(document).ready(function(){
  gt.initGame();
});
</script>
```

gt.initGame() can be called with a configuration object parameter. This object can override game definition configuration in data.js. In fact an entire game definition object can be passed as parameter. It also accepts the property "gameParentElement" that can contains a reference to an html element that will contains the game inside.

7 Interesting usage scenarios

7.1 Teaching

Trivia game can be used by (high)school teachers for knowledge review, for example the class before an exam. For example, a philosofy teacher, can perform a trivia with questions of different studied topics like epistemology, logic, ethics. Optionally the teacher want's its trivia to contain a nice famous painting as board background. Generatrivia is an excelent tool for this because the teacher only has to focus on its questions and answers and generatrivia will do the game work. It is up to the teacher to spend time decorating the board with its own image. Also the rules of the game are defined in the classroom.

7.2 Defining and making your own board games

...