Mancala Practical

mjr 17.3.15

# Introduction

This practical is to introduce you to the game of mancala. First we are going to play the game, to make sure that you know what it is about. You will find the game in the file "mancala.zip" on Moodle. Alternatively, you will find it on the web here:

<http://www.lutanho.net/play/mancala.html>

Next we will practice modifying the appearance and the performance of the game.

You are already at a level where you could write a game like this from scratch. Unfortunately you do not have time, so you will need to learn how to modify an existing piece of code.

# Playing the game

Download the file mancala.zip from Moodle and unzip it. It will not run properly unless you unzip it. Put it on your desktop, H:drive or pen-drive.

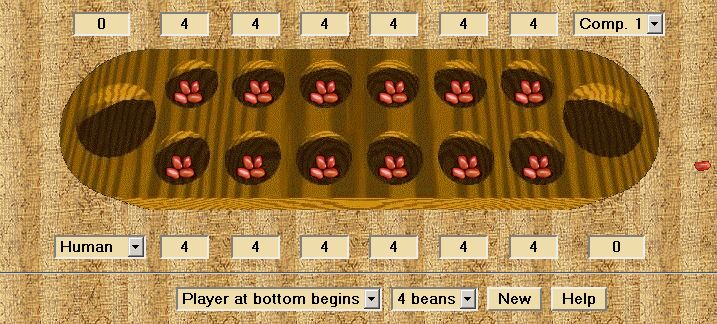
Right-click it

Select "Extract All..."

If the target destination looks sensible, click "Extract".

You can now see the extracted folder in Explorer.

Double-click the HTML file to run the game.

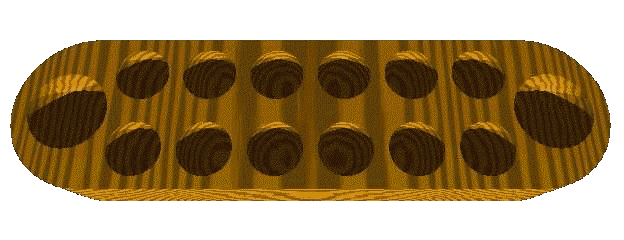


Quoting from the game's own help file:

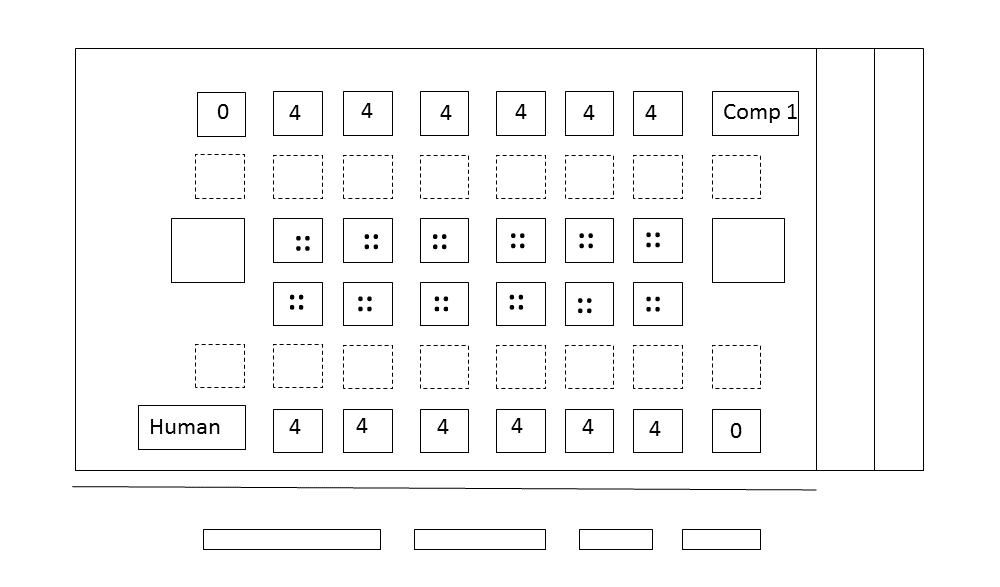
|  |
| --- |
| Mancala is an ancient African game. There exist many variations with slightly different rules.  Each player owns the mancala (the bigger bowl at the end of the row) on his right side and the six small bowls closest to him.   1. A player starts by taking all the beans from one of his small bowls and drops one bean into the next bowl on the right, continuing around the board anticlockwise until he has no more beans to drop. 2. If a player reaches his own mancala, he drops a bean into it, but he never drops a bean into the opponents' mancala. 3. The player with the most beans in his mancala wins.   The following are the so-called Egyptian rules:   * If a player drops a bean into his mancala, and that is the last bean from that bowl, he gets to move again. * If a player drops the last bean into one of the empty bowls on his side, he takes that bean, and all the beans in the opponent's bowl directly across and puts them in his mancala.   The game ends when one player no longer has beans in his small bowls. The other player takes all remaining beans into his own mancala. The winner is the player with the most beans. |

Play the game a few times to get the idea of how it works.

# The Structure of the Screen



If we are going to modify the game, we need to see how it is built and especially how the screen is presented. You might find an HTML editor useful. You can use the code editor of Expression Web from the "All Programs" list, or from the Application Window there are UltraEdit or Notepad++. You might consider downloading one or more of these for your own use. All give code line numbers, which you will need. As far as I know, only Firefox actually prints line numbers.



1. The body of the page (line 262) sets the background of the entire screen to what I would call amber. Then, in the same line, it replaces the even amber background with an amber texture, bg\_papyrus.gif.
2. The little information windows have their colour set to pale yellow by the two stylesheet elements in lines 10 & 11.
3. The whole of the playing area is centred in the screen by the "div" in line 263.
4. The playing area itself is contained in a table containing only one cell (line 270). The board is then laid out as a table (line 270 again) within that table cell. That outer table with its one cell allows the board (mancala\_bg.gif) to be used as a background to the whole game. Almost all graphics used in this game are items like boards and beans, set against a transparent background.
5. The top line of the game itself contains information on the numbers of beans in the top cells (line 273)
6. Then there is a blank row of cells (line 290)
7. Then there is a row of cells of images of beans (line 293). The larger "mancala" cells are at either end of this row. All cells are initialised to contain "mancala0.gif", which is a transparent image with no beans. These will be changed when the JavaScript initialises.
8. Then comes the bottom row of bean images (line 303)
9. Then a row of empty cells (line 311)
10. Then a row of bean numbers (line 314)
11. This ends the inner table. The outer table, which positions the board, then has two more cells on the right hand side. These are used for single spare beans, used in the game. That closes the outer table
12. At this point we are no longer in any table. We draw a horizontal rule (line 336)
13. Finally, a single-row table to contain the game controls such as "Player at bottom begins".
14. That's it.

# How it works

I could go on for hours about how the game works. Here are a few essential points:

1. The stylesheet elements control the background colours of the numerical windows (lines 9... 12).
2. The "intelligence" of the game lies in the JavaScript (lines 13... 260).
3. The HTML (lines 262... 352) describes the layout of the screen.
4. at the end of the HTML, there is a line which initialises the JavaScript once the code has loaded in the browser (line 354) and then sets the timer to 700 milliseconds (line 355). That ¾ second timer slows down the machine's moves so that a human can follow them.

Here is how the game flows:

1. When the player clicks on a cell, the details of the cell are sent to a JavaScript routine. For example, look at line 297, "onMouseDown=MouseDown(0,3)".
2. The JavaScript then works out how to change the game state.
3. Then the bean images in the cells are changed. There is an array of image names, created in lines 24 - 27. You can use a set of images with different names by changing the word "mancala" in line 27 to something else. Then the cell's image is taken from the array and put into the appropriate cell (lines 213 - 220).
4. There is one "gotcha" in the writing of the images to the screen. The code line:  
   "document.images[n].scr = ....." replaces the nth image in the code listing. At the moment it works correctly, but if you put your own image above it in the code, the images will all go into the wrong places. A solution is to write new images at the end and use CSS to put them where you really want them. When I write games I use "GetElementById" methods, but I did not write this one so we have to hack what we've got.
5. A lot of the decisions are random (lines 148 & 163). This makes the programme look wise and decisive, and gives the player the impression that his opponent is intelligent. Some human players seem to make random moves too.

# What you do next

We're going to change the appearance of the screen by changing the images used by the game. We can change:

The game's background colour. (line 262)

The game's background texture. (line 262)

The background colours of the boxes, or even the colour of the text in them. (lines 10 & 11)

The appearance of the board

The appearance of the bean images

The timing of the machine's moves. (line 355)

With a little thought you could do a lot more, but these are the low-hanging fruit.

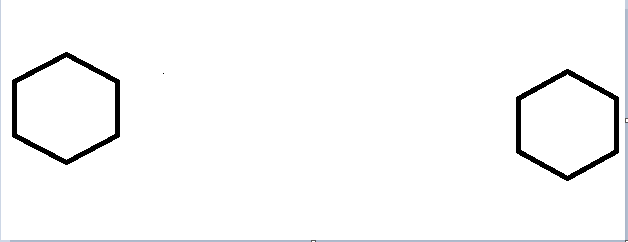
The board is 624 pixels wide x 240 pixels high.

The bean images are 60 pixels wide x 72 pixels high.

If you need a transparent background, you will find it here:

<http://www.soc.napier.ac.uk/~cs104/games/transparent.gif>

## How to do it

1. Go to Google images and type in "web textures". Find a nice one and download it. put it in the same folder as mancala.html. Change line 262 of mancala.html to the name of your texture.
2. Go to <http://html-color-codes.com/> and pick a nice background colour for the information windows on lines 10 & 11. At the moment they are set to #eeddaa; My preference is for #FFCCCC; If you're curious about how the letters link to the colours, go to W3Schools and find out. Experiment - you aren't going to break anything. If you get lost, you can revert to the original unwrecked file.
3. In Paint, go File-new. Resize the canvas to 624 wide x 240 high. "Control-a" selects the whole canvas. Then in the Select menu, go to the bottom to make the canvas transparent. Add a shape at either end as a mancala box and save it as mancala\_bg.gif. 

There are at least four ways of visually defining containers of the bean rows:

* You can leave them blank, to be defined by the bean images,
* You can add them into the board image, which is tricky,
* You can add them to the beans (see below)
* You can give their table cells borders:

|  |
| --- |
| td.bordered {      border-style: solid;      border-width: 2px; } |

1. In paint, design a new set of bean images. Call them mancala0.gif... mancala11.gif. Use them to replace the existing bean images. The image size needs to be 60 pixels wide x 72 pixels high.

|  |  |
| --- | --- |
|  | C:\Users\cs104\Desktop\mancala\pissab2\mancala4.gif |