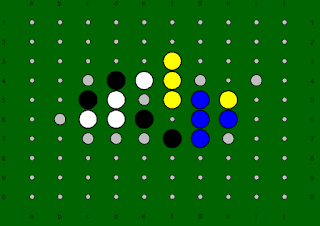
In this post, I finish the Reversi / Othello game in R by improving the graphics, adding the ability to save and load boards, and fixing bugs. Also, many more boards have been added and tested, including those with unusual shapes, three or more players, and walls that can make the board into unusual shapes or even break it in half.

This newer version can be [downloaded with this link](https://drive.google.com/open?id=1BEVMGqfxgrg5_UWA3K6vvicX8x6yB3E9). It can be run by setting the appropriate working directory and typing source(“Reversi 2019-08-07.R”).

In this version, stones are represented by coloured circles like they are in most implementations of Reversi. Spaces were stones can be played are represented by small grey circles, which become larger when there are legal moves for the current player.

[](https://i0.wp.com/1.bp.blogspot.com/-6pjETGbsxeY/XUxlqEXFqSI/AAAAAAAAArc/P4JtEC0yNTUrQk1OeiG_asYcbP83rspfACEwYBhgL/s1600/RevDemo_2.PNG?ssl=1)

This is all done with the plot.stones() function, which has taken over part of the plot.board() function’s work.

plot.stones = function(board, legal\_board){

…

First, define the colours of each stone (or blank space)

Up to six players can be on the board at once, representing by WURBG and Y. Player turns are done alphabetically based on the single character representations of the colour, which is (B)lack, then (G)reen, then (R)ed, then bl(U)e, then (W)hite, and finally (Y)ellow.

draw\_col = rep(“”,length(chars))

draw\_col[which(chars == “.”)] = “gray”

draw\_col[which(chars == “W”)] = “white” #…

draw\_col[which(chars == “U”)] = “blue”

draw\_col[which(chars == “Y”)] = “yellow”

Next define the radii of each circle to be drawn. Recall that each space on the board is drawn at an integer coordinate, so a radius of 0.35 means that stones should take 70% of the space between cardinally adjacent spaces. Realistically, it also depends a bit on the plot window.

draw\_rad = rep(0.1,length(chars))

draw\_rad[legal] = 0.2

draw\_rad[!(chars %in% c(“.”,” “,”#”))] = 0.35

For each of the spaces on the board, draw a circle for each stone or empty space. We do this using the draw.circle() function from the plotrix package, which takes in the x and y coordinates of the circle center, the radius, border colour, and fill colour. The fill only appears because we specified a colour, otherwise the circle would be transparent.

for(k in 1:length(chars)){

if(draw\_col[k] != “”){

draw.circle(x[k],y[k],radius=draw\_rad[k],

border=”black”,col=draw\_col[k])  }

For each wall, draw a square filling the 1×1 space with the base R polygon() function.

We draw nothing for voids. Mechanically voids and walls are identical, but they are treated differently graphically.

if(chars[k] == “#”){

size = 0.5

corners\_x = c(x[k]-size, x[k]+size, x[k]+size, x[k]-size)

corners\_y = c(y[k]-size, y[k]-size, y[k]+size, y[k]+size)

polygon(corners\_x,corners\_y,col=”gray”)  }   }

A few notable updates to the main function play.game(). At the start we load the required plotrix function, set the background of any plots to darkgreen, and remove the inner margins so that we can use the entire plot window. The par() function sets a variety of basic graphical parameters (in the computing science sense, not the statistical one). You can see what the settings available are, and their current value, by simply typing par().

require(plotrix)

par(bg = “darkgreen”)

par(mar = c(0,0,0,0))

The while loop that checks for a valid click also now checks that the plot window is still open. If it is closed, the loop and the game end. This is done with dev.cur(), which returns the names of any graphical devices, such as plot windows, that are actively. Rstudio and vanilla R have different names for their plotting devices, but both implementations return “null device” if no plot window is open.

while(  … & any(names(dev.cur()) != “null device”))

The import.board() function takes a raw .txt file and converts in into a matrix of single characters, which are then used as a board in the game.

The function readLines() takes a text file and saves a vector of string variables, one for each line. The strsplit function splits these into individual characters, but organizes them into a list of vectors, one vector per original string. For these reason, we also need unlist(). The rest is arithmetic.

import.board = function(filename, echo=TRUE)

{

     rawboard = readLines(filename)

     rawchars = unlist(strsplit(rawboard,””))

     Ny = length(rawboard)

     Nx = length(rawchars) / Ny

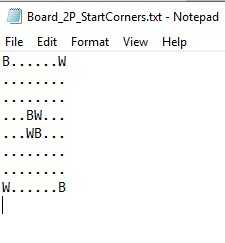
     board = matrix(rawchars,byrow=TRUE,nrow=Ny,ncol=Nx)

     if(echo){print(board)}

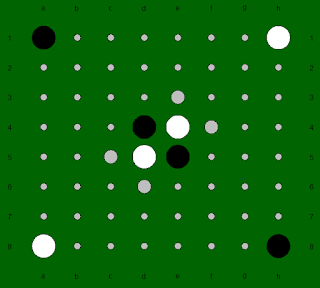
     return(board)

}

 The function import.board() turns this…

[[](https://i1.wp.com/1.bp.blogspot.com/-ubatN0JraG4/XUxlsEp6eoI/AAAAAAAAArM/zhV5AhWHNVgxQeAMsOZQb3tRtwhDhWnaACLcBGAs/s1600/RevDemo_File.PNG?ssl=1)](https://i1.wp.com/1.bp.blogspot.com/-ubatN0JraG4/XUxlsEp6eoI/AAAAAAAAArM/zhV5AhWHNVgxQeAMsOZQb3tRtwhDhWnaACLcBGAs/s1600/RevDemo_File.PNG?ssl=1)

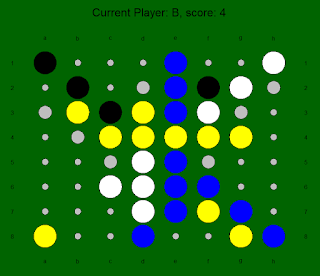
…into this…

[](https://i0.wp.com/1.bp.blogspot.com/-zS6oYdc3zNQ/XUxmK9RmBFI/AAAAAAAAArg/IngTMoTbCwEaY-RJxprwKl6fMhL7uLDOACLcBGAs/s1600/RevDemo_7.PNG?ssl=1)

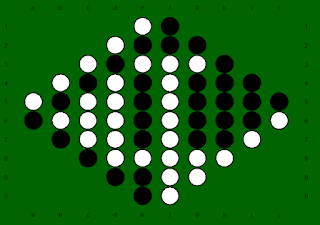
 …a board with each of two players already holding two corners.

 Among the boards included, and tested, in the download are also:

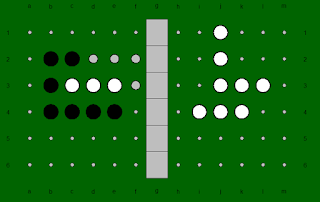
 A corner-filled version for four players. Shown here is a game in progress for this board.

[](https://i0.wp.com/1.bp.blogspot.com/-4KAy7CqXdqw/XUxlqopDRhI/AAAAAAAAArY/Aea1_APeBWkIbAFArMQf4Y_vAGVOpfK6ACEwYBhgL/s1600/RevDemo_4.PNG?ssl=1)

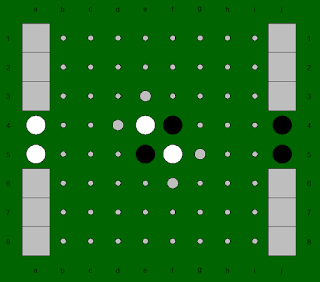
A diamond-shaped board with double-size corners. Here a completed game is shown.

[](https://i1.wp.com/1.bp.blogspot.com/-vpAjPjwvvvQ/XUxlqLde5XI/AAAAAAAAArY/-hU94P4EEdQ7NafUxsMhbifQqBRu0QLvgCEwYBhgL/s1600/RevDemo_1.PNG?ssl=1)

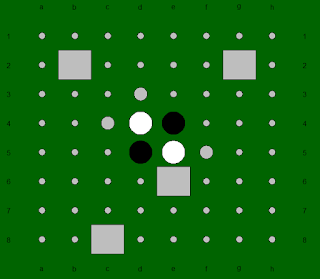
Two 6×6 boards separated by a wall. If a player is eliminated from one side, as black has been on the right, then neither player can make a move on that side again, so you really have balance priorities.

[](https://i0.wp.com/1.bp.blogspot.com/-tU6enY1dkb8/XUxlqGxMeWI/AAAAAAAAArQ/AaOH1AWCyA4gzWsQUtje3sm73B0qyg80wCEwYBhgL/s1600/RevDemo_3.PNG?ssl=1)

A setup where each player has a fortress that extends beyond the usual 8×8 board.

[](https://i2.wp.com/1.bp.blogspot.com/-uazs0l57P_o/XUxlrPHB3-I/AAAAAAAAArU/hvQXDEXnWHwy34wo5WpImT16B1wu53soACEwYBhgL/s1600/RevDemo_5.PNG?ssl=1)

A board on which a few obstacles have been scattered about.

[](https://i1.wp.com/1.bp.blogspot.com/-tJCZLp9fN6k/XUxlrh3F6WI/AAAAAAAAArY/UGWHlmDBJJM3aIvouCgtlDy6xo-9E-AMgCEwYBhgL/s1600/RevDemo_6.PNG?ssl=1)