Do you want to design a computer game?  
Do you want to learn about tools for designing simple games?  
Do you want to know the approaches to implementing simple games?  Let do that in this lab.

Goals: Learn to solve problem and design solutions. In particular learn about

- Functional decomposition,
- Use of data structures such as arrays
- Use pushMatrix and popMatrix of Processing to layer your sketch.
- Simple game logic.

Objectives:

Learn to solve problems by designing algorithmic solutions. Use array data structures, functions and complex logical expressions and arithmetic operations such as modulus. Use switch statement to implement the state machine-defined logic of a game. Use pushMatrix and popMatrix of Processing to layer your sketch.

Problem context:

The traveler in the first two labs has gotten bored of travelling and wants to rest for a while. You have been assigned the task of building a simple game for this world traveler. Only request from the traveler is to make use of all the images the person (you) have collected during the world travel around the world. We have decided to design the famous “Memory” game. See a sample version of the game at this location: http://www.mathsisfun.com/games/memory/

Problem Statement:

You are required to build a version of the memory game that has nXn tiles where n is even. The tiles have two faces, one is closed and the other has an image. Exactly two tiles of the nXn will have the same image. Initially all the tiles are covered. You goal is to open a pair of tiles one at a time with the aim of finding a matching pair. When you select a tile using the mouse, it flips to its image. After selecting two tiles, if the tiles do not have matching images they will revert back to the closed state. The game ends when all the tiles have been matches and there are no more tiles to flip open. We need some means for scoring the game. We will assume the number of tries to open up all the tiles as the overall score. Lower the score, the better is your “memory”. Your game board will also have START and NEXT buttons and a place to display the score of the player. There are a lot of other possible enhancements, but due to time limitations we will restrict it to a simple version. See figure 1 for a sample layout.

The tile of the game can be at the bottom of the sketch. While n can be large, we will limit the maximum size to 6 X 6. Start by implementing a 2X2 game as a proof of concept; then move on to 4X4 and then to
a 6X6 board. Images you have will have to be scaled to fit the tile size. You are required to provide a general solution using variables and parameterized functions wherever appropriate.

![Image of a 6X6 board with buttons: Start, Next, Score](image.png)

**Figure 1 A Simple layout of the memory game**

Implementation Details:

- Play the game and understand the game.
- Identify the problems to be solved: Here are some the problems to be solved:
  - Setting up the layout (Eg. 400X 400 for the tiles plus the bottom strip; title can be at the bottom)
  - Flipping a tile by pressing the mouse.
  - Remembering whether it is the first tile flip or second tile;
  - When it is second tile you need to determine if there is a match or not;
  - You will have to keep track of first selection of tile, and the second selection to determine if there is a match.
  - Keeping track of the tiles (eg. in arrays) to allow for matching.
  - A challenging task is that mapping of tile position to the index of the array(s).
  - Keeping track of score.
  - Keeping track of the progress of the game and display the score when it ends.
- It is very important you develop the solution in incremental steps.
- Your program should be modular with coherent functions and parameters to customize the functions.
- You can use println to debug your program.
- Reuse the images you collected in the lab1 and lab2 and any others if you prefer. Or you can use a brand new set of pictures. Go around the campus and snap some spring pictures: flowers, brids, etc.
You can use mouse operations such as mouse pressed or mouse released operations.

Suggested program Layout based on the state diagram given above:

//Name box

// constants for the states above

// other variables: many array representing the data for the game board

// you will need a lot of arrays and for/while loops to process them

void setup()
{
  // square set for the board + the panel for start, next and score
}

void draw()
{
  //probably empty? But it keeps listening to events and loops to redraw
}

void mousePressed()
{
  // very important logic, “switch” or multi-way selection
  ....
}

void initializeBoard(int n)
{
  // one single function for initializing the board irrespective of the size
  // because it is parameterized on “n”
}

void initializeTurn()
void selectTile1()
{
}

void selectTile2()
{
}

int mapSelectionToIndex()
{
}

void match()
{
  
  // it matches the tiles selected
  // if matched, it scores
  // It sets MA or NM state

}

void closeTiles()
{
}

void displayScore()
{
}

Submission: Submit it on ublearns as soon as it ready to submit. You need to demo it to your TA to get a grade.: 10/24/2014