1 Sound Effects

1.1 Delay

```
function result = delay(data, fs, delta)
% Delays the sound stored in data by an amount delta given in seconds.
% Each data point is seperated by 1/fs seconds.

dn = floor(delta * fs);
result = zeros(size(data));
result(dn+1:end,:) = data(1:end-dn,:);
```

1.2 Echo

```
function result = echo_effect(data, fs, num_echos, delta, damping)
% Creates one or multiple echoes seperated by a time difference
% delta which is given in seconds. Each echo has a lower sound
% level than its source. The ratio of the levels is given by the
% variable damping.

result = data;
for n = 1:num_echos
   delayed = (damping^n) * delay(data, fs, n*delta);
   result = result + delayed;
end
```

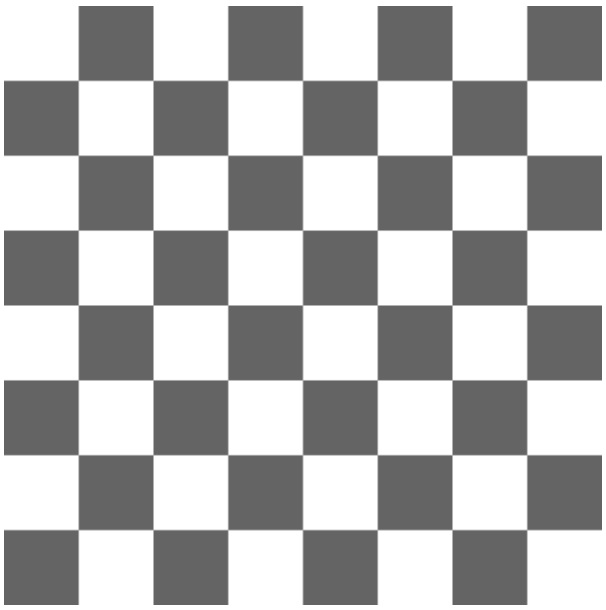
1.3 Fade-out

```
function result = fade_out(data, fs, delta, damping)
% Creates a fade-out effect after delta seconds. The damping parameter is
% used in the multiplier function, e.g. exp(-damping * someOtherStuff).

result = data;
dn = length(data) - floor(delta * fs);
result(end-dn:end,:) = result(end-dn:end,:) .* exp(-damping*(0:dn)'/fs);
```

2 Chess

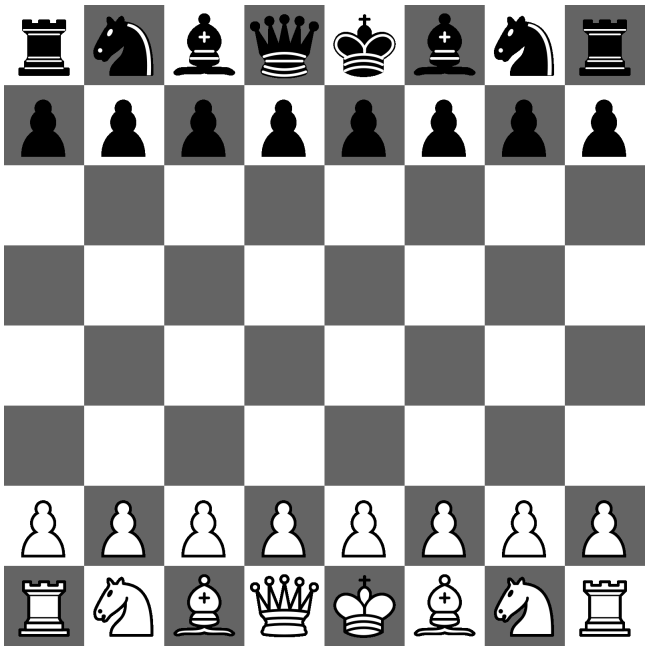
2.1 Draw Board

```
function imboard = draw_board(filename, side)
% Returns an image data for a chessboard. Stores the result in a file.
% The side length of a square is given in pixels.

% There are various way you can start with an empty board.
% Here we 'paint' a gray color everywhere, and later we will
% update the color of white squares. The reason is that black
% pieces provided don't have a border and when they are shown
% on black squares, they become invisible.
% Also notice that we can directly create 'ones' of 'uint8' type.
imboard = 100*ones(side*8,side*8,3,'uint8');

for r = 1:8
   for c = 1:8
      if rem(r + c,2) == 0
         imboard(side*(r-1)+1:side*r,side*(c-1)+1:side*c,:) = 255;
      end
   end
end

imwrite(imboard, filename,'png');
```



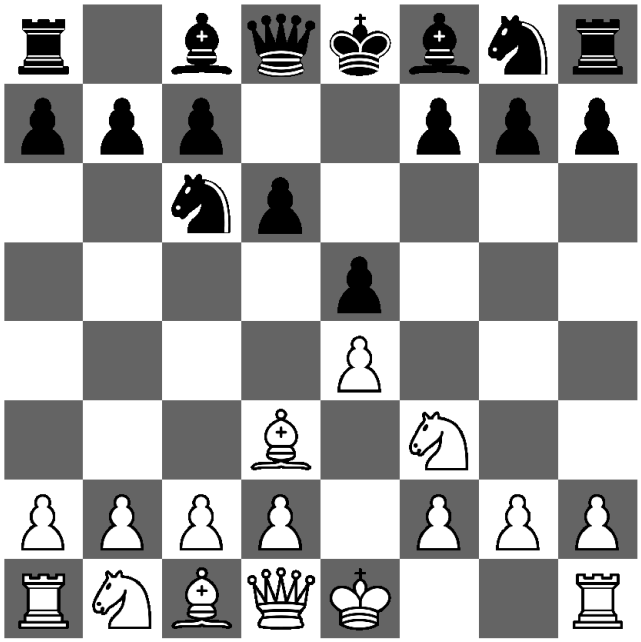
2.2 Show Pieces

```
function result = show_pieces(board)
% Shows the current state of the board as an image with pieces.
% The output array, result, is an RGB image data.

[wp,t,wpm] = imread('w_pawn.png'  ,'png');
[wr,t,wrm] = imread('w_rook.png'  ,'png');
[wn,t,wnm] = imread('w_knight.png','png');
[wb,t,wbm] = imread('w_bishop.png','png');
[wq,t,wqm] = imread('w_queen.png' ,'png');
[wk,t,wkm] = imread('w_king.png'  ,'png');

[bp,t,bpm] = imread('b_pawn.png'  ,'png');
[br,t,brm] = imread('b_rook.png'  ,'png');
[bn,t,bnm] = imread('b_knight.png','png');
[bb,t,bbm] = imread('b_bishop.png','png');
[bq,t,bqm] = imread('b_queen.png' ,'png');
[bk,t,bkm] = imread('b_king.png'  ,'png');

sides = size(wp);
h = sides(1); % height
w = sides(2); % width
result = draw_board('board.png',w);

for r = 1:8
   for c = 1:8
      switch board(r,c)
         case 1,   piece = wp; mask = wpm;
         case 2,   piece = wr; mask = wrm;
         case 3,   piece = wn; mask = wnm;
         case 4,   piece = wb; mask = wbm;
         case 5,   piece = wq; mask = wqm;
         case 6,   piece = wk; mask = wkm;
         case 7,   piece = bp; mask = bpm;
         case 8,   piece = br; mask = brm;
         case 9,   piece = bn; mask = bnm;
         case 10,  piece = bb; mask = bbm;
         case 11,  piece = bq; mask = bqm;
         case 12,  piece = bk; mask = bkm;
      end

      if board(r,c)
         for i = 1:h
           for j = 1:w
              if mask(i,j) ~= 0,
                result(h*(r-1)+i,w*(c-1)+j,:) = piece(i,j,:);
              end
           end
         end
      end
   end
end

imshow(result)
```



2.3 Display Game

```
function display_game(filename, delay)
% Reads moves in coordinate notation from a textfile. Shows them on the
% chessboard. Waits delay seconds in between the moves.

f = fopen(filename,'r');

board = initialize();
show_pieces(board);

moves = fscanf(f, '%c%c%c%c%c\n', [5 inf])'
for i = 1:size(moves,1)
   pause(delay);
   board = move_piece(board, moves(i,:));
   show_pieces(board);
end

fclose(f);
```



2.4 Animate Game

```
function animate_game(movie_fname, game_fname, delay)
% Creates an animation of a chess game and stores it as a movie

f = fopen(game_fname,'r');

board = initialize();
show_pieces(board);

moves = fscanf(f, '%c%c%c%c%c\n', [5 inf])'

M(1) = getframe;
for i = 1:size(moves,1)
   pause(delay);
   board = move_piece(board, moves(i,:));
   show_pieces(board);
   M(i+1) = getframe;
end
fclose(f);

movie2avi(M,movie_fname,'FPS',1/delay);
```