function snake\_ladder\_game

% Create the figure window

f = figure('Position', [200, 200, 800, 600], 'Name', 'Snake and Ladder Game');

% Load and display the game board image

board\_img = imread('snk.jfif'); % Load your board image here

ax = axes(f, 'Position', [0.05, 0.25, 0.7, 0.7]); % Axes for board image

imshow(board\_img, 'Parent', ax);

hold on;

% Snakes and Ladders mapping (Key: start position, Value: end position)

snakes = containers.Map([17, 54, 62, 64, 87, 93, 95, 98], [6, 34, 19, 60, 24, 73, 75, 78]);

ladders = containers.Map([1, 4, 9, 21, 28, 36, 51, 71, 80], [38, 14, 31, 42, 84, 44, 67, 91, 100]);

% Player positions and colors

player\_positions = [1, 1]; % Start positions for two players

player\_colors = {'r', 'b'}; % Red and Blue tokens for players

% Create dice roll button and text display

dice\_roll\_button = uicontrol(f, 'Style', 'pushbutton', 'String', 'Roll Dice', ...

'Position', [600, 400, 100, 50], 'FontSize', 14, 'Callback', @roll\_dice);

dice\_result = uicontrol(f, 'Style', 'text', 'String', 'Dice: 0', ...

'Position', [600, 350, 100, 30], 'FontSize', 14);

player\_turn\_text = uicontrol(f, 'Style', 'text', 'String', 'Player 1 Turn', ...

'Position', [600, 450, 150, 30], 'FontSize', 14);

% Function to handle dice roll and player movement

function roll\_dice(~, ~)

current\_player = mod(sum(player\_positions > 1), 2) + 1; % Determine current player

dice = randi(6); % Roll a dice (1 to 6)

dice\_result.String = ['Dice: ', num2str(dice)];

% Move the current player

new\_position = player\_positions(current\_player) + dice;

if new\_position > 100

new\_position = 100;

end

% Check for snakes or ladders

if isKey(snakes, new\_position)

new\_position = snakes(new\_position); % Slide down snake

elseif isKey(ladders, new\_position)

new\_position = ladders(new\_position); % Climb ladder

end

% Update the player's position

player\_positions(current\_player) = new\_position;

update\_board();

% Check for win

if new\_position == 100

msgbox(['Player ', num2str(current\_player), ' Wins!'], 'Game Over');

reset\_game();

return;

end

% Change player turn

player\_turn\_text.String = ['Player ', num2str(mod(current\_player, 2) + 1), ' Turn'];

end

% Function to update player positions on the board

function update\_board

% Clear previous tokens

cla(ax);

imshow(board\_img, 'Parent', ax);

hold on;

% Plot the players' tokens at the new positions

for i = 1:2

[x, y] = get\_coordinates(player\_positions(i));

scatter(x, y, 200, player\_colors{i}, 'filled', 'MarkerEdgeColor', 'k', 'LineWidth', 2);

% Adjust '200' for token size if necessary

end

end

% Function to reset the game

function reset\_game

player\_positions = [1, 1];

update\_board();

end

% Function to calculate the x, y coordinates on the board image

function [x, y] = get\_coordinates(position)

% This logic assumes a 10x10 board, modify based on your board layout

board\_size = 10; % 10x10 board

row = ceil(position / board\_size); % Determine row number

col = mod(position - 1, board\_size) + 1; % Determine column number

% Reverse column direction every other row for snake and ladder board

if mod(row, 2) == 0

col = board\_size + 1 - col;

end

% Scale the x and y coordinates based on the board image size

board\_image\_width = size(board\_img, 2); % Get width of the image

board\_image\_height = size(board\_img, 1); % Get height of the image

x\_spacing = board\_image\_width / board\_size;

y\_spacing = board\_image\_height / board\_size;

% Calculate the center coordinates of the square (x, y)

x = (col - 0.5) \* x\_spacing;

y = board\_image\_height - (row - 0.5) \* y\_spacing; % Reverse y for image coordinates

end

% Initialize the board

update\_board();

end