```matlab
%   Academic Integrity Statement:
%   We have not used source code obtained from
%   any other unauthorized source, either modified
%   or unmodified.  Neither have we provided access
%   to our code to other teams. The project we are
%   submitting is our own original work.

function varargout = gameGUI(varargin)
global playerCount;

global frontDie;
global backDie;
global scoresArray;
global leftSum;
global centerSum;
global rightSum;
global y;
global Fs;

scoresArray = [];
leftSum = 0;
centerSum = 0;
rightSum = 0;

% Read in sound file for dice rolling sound
[y,Fs] = audioread('dicesound.wav');

% Create two instances of the DiceClass with 6 sides -- User defined
 OOP
die1 = DiceClass(6);
die2 = DiceClass(6);

% Read in two images that the DiceClass produced in its constructor
frontDie = imread('dieLetters.png');
backDie = imread('dieDots.png');

% Determine number of players selected from input arguments
if nargin > 0
    playerCount = varargin{1};
end

% GAMEGUI MATLAB code for gameGUI.fig
%      GAMEGUI, by itself, creates a new GAMEGUI or raises the
 existing
%      singleton*.
%
%      H = GAMEGUI returns the handle to a new GAMEGUI or the handle
 to
%      the existing singleton*.
%
%      GAMEGUI('CALLBACK',hObject,eventData,handles,...) calls the
 local
```

```
%      function named CALLBACK in GAMEGUI.M with the given input
 arguments.
%
%      GAMEGUI('Property','Value',...) creates a new GAMEGUI or raises
 the
%      existing singleton*.  Starting from the left, property value
 pairs are
%      applied to the GUI before gameGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
 application
%      stop.  All inputs are passed to gameGUI_OpeningFcn via
 varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
 only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gameGUI

% Last Modified by GUIDE v2.5 04-Dec-2017 17:37:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gameGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @gameGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gameGUI is made visible.
function gameGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gameGUI (see VARARGIN)


% Choose default command line output for gameGUI
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);



global playerCount;
global playernames2;
global playernames3;
global playernames4;

% Set playernames array based off of number of players
% Set extra scores/names to invisible in GUI
switch(playerCount)
    case 2
        % Set all extra names/scores invisible
        set(handles.playerthreename,'visible','off');
        set(handles.playerfourname,'visible','off');
        set(handles.playerthreescore,'visible','off');
        set(handles.playerfourscore,'visible','off');

        % Set all player names
        set(handles.playeronename, 'string', playernames2.player1);
        set(handles.playertwoname, 'string', playernames2.player2);

        % Set all player scores to 3 initially
        set(handles.playeronescore, 'string', '3');
        set(handles.playertwoscore, 'string', '3');

        set(handles.playertoroll,'String',strcat('Waiting for -> ',
 playernames2.player1));

    case 3
        % Set all extra names/scores invisible
        set(handles.playerfourname,'visible','off')
        set(handles.playerfourscore,'visible','off')

        % Set all player names
        set(handles.playeronename, 'string', playernames3.player1);
        set(handles.playertwoname, 'string', playernames3.player2);
        set(handles.playerthreename, 'string', playernames3.player3);

        % Set all player scores to 3 initially
        set(handles.playeronescore, 'string', '3');
        set(handles.playertwoscore, 'string', '3');
        set(handles.playerthreescore, 'string', '3');

        set(handles.playertoroll,'String',strcat('Waiting for -> ',
 playernames3.player1));


    case 4
        % Set all player names
        set(handles.playeronename, 'string', playernames4.player1);
        set(handles.playertwoname, 'string', playernames4.player2);
        set(handles.playerthreename, 'string', playernames4.player3);
        set(handles.playerfourname, 'string', playernames4.player4);

        % Set all player scores to 3 initially
        set(handles.playeronescore, 'string', '3');
        set(handles.playertwoscore, 'string', '3');
        set(handles.playerthreescore, 'string', '3');
        set(handles.playerfourscore, 'string', '3');

        set(handles.playertoroll,'String',strcat('Waiting for -> ',
 playernames4.player1));

end




% UIWAIT makes gameGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gameGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in rollbutton.
function rollbutton_Callback(hObject, eventdata, handles)
% hObject    handle to rollbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global count;
global frontDie;
global backDie;
global playernames2;
global playernames3;
global playernames4;
global playerCount;
global y;
global Fs;


namesArray = {};

% Read in dice faces for L, R, C, and Dot
diceFaceL = imread('diceFaceL.png');
diceFaceC = imread('diceFaceC.png');
```

```
diceFaceR = imread('diceFaceR.png');
diceFace1 = imread('diceFace1.png');


numPlayers = 1;

% Redetermine number of players and fill in namesArray with names of
% players
if strcmpi(handles.playerfourscore.Visible, 'on')
    namesArray = {playernames4.player1, playernames4.player2,
 playernames4.player3, playernames4.player4};
    numPlayers = 4;
elseif strcmpi(handles.playerthreescore.Visible,'on')
    namesArray = {playernames3.player1, playernames3.player2,
 playernames3.player3};
    numPlayers = 3;
else
    namesArray = {playernames2.player1, playernames2.player2};
    numPlayers = 2;
end


global scoresArray;
global leftSum;
global centerSum;
global rightSum;
global winnerName;
global winnerScore;


% Run game logic for different number of players
switch numPlayers
```

For two players

```
    case 2

        % Logic to determine which player won the game given 2 players
        if ( (numPlayers*3 == eval(get(handles.potcount, 'String'))
 + eval(get(handles.playeronescore, 'String'))) ||
 (numPlayers*3 == eval(get(handles.potcount, 'String'))
 + eval(get(handles.playertwoscore, 'String')))
 || ( eval(get(handles.potcount, 'String')) +
 eval(get(handles.playeronescore, 'String')) +
 eval(get(handles.playertwoscore, 'String')) > numPlayers * 3))

            if (numPlayers*3 == eval(get(handles.potcount, 'String'))
 + eval(get(handles.playeronescore, 'String')))
                winnerName = namesArray(1);
                winnerScore =
 eval(get(handles.playeronescore, 'String'));
            else
                winnerName = namesArray(2);
```

```
               winnerScore =
eval(get(handles.playertwoscore, 'String'));
           end

           % Display winner screen GUI
           close(gameGUI);
           run('WinnerScreen')
       else

           % Determine which player is the current player
           if mod(count,numPlayers) == 0
              roundCounter = 1;
           else
               roundCounter = 2;
           end

           % Set the scoresArray to scores of the players
           scoresArray = [eval(get(handles.playeronescore,'String')),
eval(get(handles.playertwoscore,'String'))];
           set(handles.currentplayer, 'string',
namesArray(roundCounter));

           % Let user know which player the program is waiting on
           if roundCounter + 1 <= numPlayers
               set(handles.playertoroll,'String',strcat('Waiting for
-> ', namesArray(roundCounter + 1)));
           else
               set(handles.playertoroll,'String',strcat('Waiting for
-> ', namesArray(1)));
           end

           % Play dice roll sound for every roll
           sound(y,Fs);

           % Randomizer to have random number of times the dice rolls
for
           % each turn
           max = ceil((7*rand()+11));

           % Code to show animated dice on bottom right of screen
           for i = 1:max
               if mod(i,2) == 0
                   frontDie =
imrotate(frontDie,-90,'bilinear','crop');
                   imshow(frontDie, 'Parent',
handles.axes6,'Border', 'tight','XData', [10,60],'YData', [10,60]);
                   imshow(backDie, 'Parent',
handles.axes7,'Border', 'tight','XData', [10,60],'YData', [10,60]);

imshow(imrotate(frontDie,90,'bilinear','crop'), 'Parent',
handles.axes8,'Border', 'tight','XData', [10,60],'YData', [10,60]);

               else
                   backDie = imrotate(backDie,-90,'bilinear','crop');
```

```
                    imshow(backDie, 'Parent',
handles.axes6,'Border', 'tight','XData', [10,100],'YData', [10,100]);
                    imshow(frontDie, 'Parent',
handles.axes7,'Border', 'tight','XData', [10,100],'YData', [10,100]);

imshow(imrotate(backDie,90,'bilinear','crop'), 'Parent',
handles.axes8,'Border', 'tight','XData', [10,100],'YData', [10,100]);
               end

               % Code to show animated dice faces for Die1, Die2, and
Die3
               switch (mod(i,6))
                   case 1
                       imshow(diceFaceL, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes3,'Border', 'tight');
                   case 2
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes3,'Border', 'tight');
                   case 3
                       imshow(diceFaceR, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceC, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes3,'Border', 'tight');
                   case 4
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes3,'Border', 'tight');
                   case 5
                       imshow(diceFaceC, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes3,'Border', 'tight');
                   case 0
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes3,'Border', 'tight');
                       imshow(diceFaceC, 'Parent',
handles.axes3,'Border', 'tight');
```

```
            end

            % Increasing pausing time to allow for more realistic
            % animation of dice roll
            pause(.01*i/3);
        end

        % Logic to determine the values of the dice faces the
player
        % rolled (L, R, C, or Dot)
        switch mod(max,6)
            case 1
                leftSum = leftSum + 1;
                if strcmpi(handles.rollpanel2.Visible, 'on')
                    rightSum = rightSum + 1;
                end
                if strcmpi(handles.rollpanel3.Visible, 'on')
                    centerSum = centerSum + 0;
                end
            case 2
                if strcmpi(handles.rollpanel2.Visible, 'on')
                    rightSum = rightSum + 1;
                end

            case 3
                rightSum = rightSum + 1;
                if strcmpi(handles.rollpanel2.Visible, 'on')
                    centerSum = centerSum + 1;
                end
            case 4
                if strcmpi(handles.rollpanel3.Visible, 'on')
                    leftSum = leftSum + 1;
                end

            case 5
                leftSum = leftSum + 1;
                rightSum = rightSum + 1;
                centerSum = centerSum + 1;
            case 0
                if strcmpi(handles.rollpanel2.Visible, 'on')

                    leftSum = leftSum + 1;
                end
                if strcmpi(handles.rollpanel3.Visible, 'on')

                    centerSum = centerSum + 1;
                end
        end

        % Logic to assign new score to each player in scoresArray
based
        % on what the player rolled for the 3 dice
        if roundCounter - 1 == 0 && leftSum > 0
```

```
                   scoresArray(end) = scoresArray(end) + leftSum;
                   scoresArray(roundCounter) = scoresArray(roundCounter)
- leftSum;
               elseif leftSum > 0

                   scoresArray(roundCounter - 1) =
scoresArray(roundCounter - 1) + leftSum;
                   scoresArray(roundCounter) = scoresArray(roundCounter)
- leftSum;
               end
               if roundCounter + 1 > numPlayers && rightSum > 0

                   scoresArray(1) = scoresArray(1) + rightSum;
                   scoresArray(roundCounter) = scoresArray(roundCounter)
- rightSum;
               elseif rightSum > 0

                   scoresArray(roundCounter + 1) =
scoresArray(roundCounter +1) + rightSum;
                   scoresArray(roundCounter) = scoresArray(roundCounter)
- rightSum;
               end

               % Logic to set the pot count and update the Pot Chips
section
               scoresArray(roundCounter) = scoresArray(roundCounter) -
centerSum;
               set(handles.potcount, 'string',
string(eval(get(handles.potcount, 'String')) + centerSum));

               % Error handling in case player scores drop to negatives
               if scoresArray(1) < 0
                   set(handles.playeronescore,'string','0');
               else
                   set(handles.playeronescore,'string',scoresArray(1));
               end
               if scoresArray(2) < 0
                   set(handles.playertwoscore,'string','0');
               else
                   set(handles.playertwoscore,'string',scoresArray(2));
               end

               count = count + 1

               % Logic to determine if any players have won the game
               if ( (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String'))) ||
(numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playertwoscore, 'String')))
|| ( eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String')) +
eval(get(handles.playertwoscore, 'String')) > numPlayers * 3))
```

```
               if (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String')))
                   winnerName = namesArray(1);
                   winnerScore =
eval(get(handles.playeronescore, 'String'));
               else
                   winnerName = namesArray(2);
                   winnerScore =
eval(get(handles.playertwoscore, 'String'));
               end

               % Open up the winner screen GUI
               close(gameGUI);
               run('WinnerScreen');
           end

       end
```

For three players

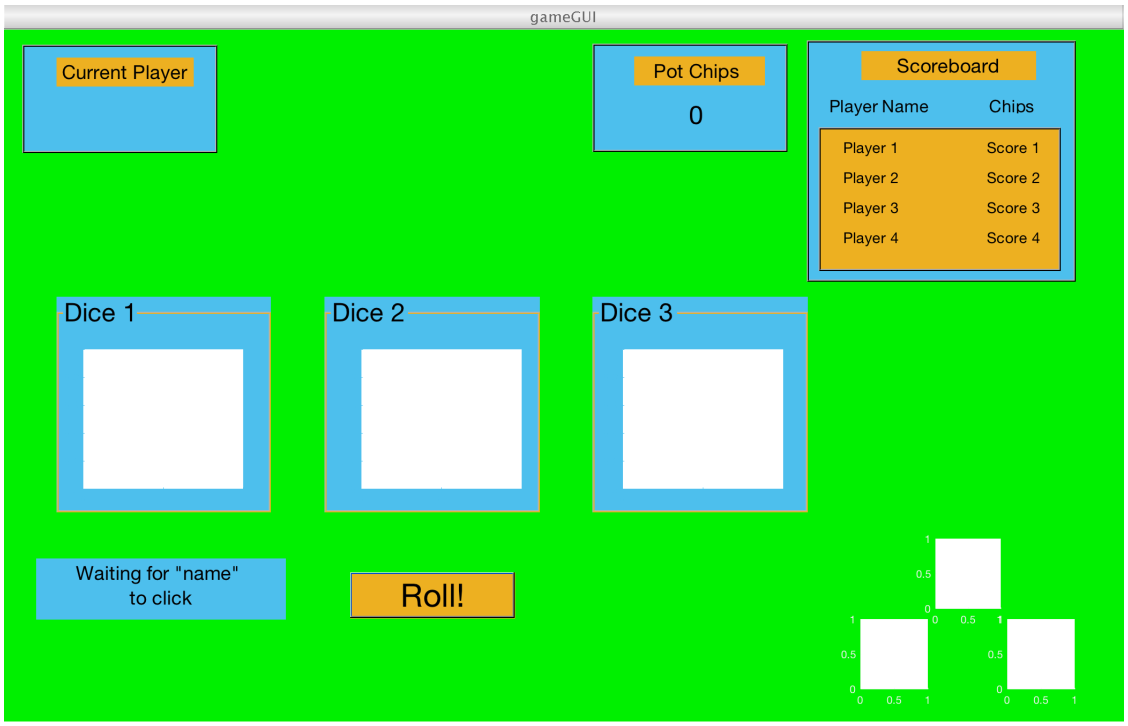
```
   case 3

       % Code to determine if any players have won the game
       if ( (numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playeronescore, 'String'))) ||
(numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playertwoscore, 'String'))) ||
(numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playerthreescore, 'String')))
||( eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String')) +
eval(get(handles.playertwoscore, 'String')) +
eval(get(handles.playerthreescore, 'String')) > numPlayers * 3))

           if (numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playeronescore, 'String')))
               winnerName = namesArray(1);
               winnerScore =
eval(get(handles.playeronescore, 'String'));
           elseif (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playertwoscore, 'String')))
               winnerName = namesArray(2);
               winnerScore =
eval(get(handles.playertwoscore, 'String'));
           else
               winnerName = namesArray(3);
               winnerScore =
eval(get(handles.playerthreescore, 'String'));
           end

           % Open up winner screen GUI
```

```
            close(gameGUI);
            run('WinnerScreen')

        else

            % Determine which players turn it is currently
            if mod(count,numPlayers) == 0
               roundCounter = 1;
            elseif mod(count,numPlayers) == 1
                roundCounter = 2;
            else
               roundCounter = 3;
            end

            % Set scoresArray to values of players scores
            scoresArray = [eval(get(handles.playeronescore,'String')),
eval(get(handles.playertwoscore,'String')),
eval(get(handles.playerthreescore,'String'))];
            set(handles.currentplayer, 'string',
namesArray(roundCounter));

            % Let user know which player the program is waiting on
            if roundCounter + 1 <= numPlayers
                set(handles.playertoroll,'String',strcat('Waiting for
-> ', namesArray(roundCounter + 1)));
            else
                set(handles.playertoroll,'String',strcat('Waiting for
-> ', namesArray(1)));
            end

            % Play dice rolling sound during each roll
            sound(y,Fs);

            % Randomizer to have random number of times the dice rolls
for
            % each turn
            max = ceil((7*rand()+11));

            % Code to show three dice roll at the bottom right of the
screen
            for i = 1:max
                if mod(i,2) == 0
                    frontDie =
imrotate(frontDie,-90,'bilinear','crop');
                    imshow(frontDie, 'Parent',
handles.axes6,'Border', 'tight','XData', [10,60],'YData', [10,60]);
                    imshow(backDie, 'Parent',
handles.axes7,'Border', 'tight','XData', [10,60],'YData', [10,60]);

imshow(imrotate(frontDie,90,'bilinear','crop'), 'Parent',
handles.axes8,'Border', 'tight','XData', [10,60],'YData', [10,60]);

                else
                    backDie = imrotate(backDie,-90,'bilinear','crop');
                    imshow(backDie, 'Parent',
handles.axes6,'Border', 'tight','XData', [10,100],'YData', [10,100]);
                    imshow(frontDie, 'Parent',
handles.axes7,'Border', 'tight','XData', [10,100],'YData', [10,100]);

imshow(imrotate(backDie,90,'bilinear','crop'), 'Parent',
handles.axes8,'Border', 'tight','XData', [10,100],'YData', [10,100]);
               end

               % Code to show different dice faces (L,R,C, and Dot)
               switch (mod(i,6))
                   case 1
                       imshow(diceFaceL, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes3,'Border', 'tight');
                   case 2
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes3,'Border', 'tight');
                   case 3
                       imshow(diceFaceR, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceC, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes3,'Border', 'tight');
                   case 4
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFace1, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes3,'Border', 'tight');
                   case 5
                       imshow(diceFaceC, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes2,'Border', 'tight');
                       imshow(diceFaceR, 'Parent',
handles.axes3,'Border', 'tight');
                   case 0
                       imshow(diceFace1, 'Parent',
handles.axes1,'Border', 'tight');
                       imshow(diceFaceL, 'Parent',
handles.axes3,'Border', 'tight');
                       imshow(diceFaceC, 'Parent',
handles.axes3,'Border', 'tight');
               end
```

```
                    % Increasing pausing time to allow for more realistic
                    % animation of dice roll
                    pause(.01*i/3);
                end

                % Logic to determine the values of the dice faces the
player
                % rolled (L, R, C, or Dot)
                switch mod(max,6)
                    case 1
                        leftSum = leftSum + 1;
                        if strcmpi(handles.rollpanel2.Visible, 'on')
                            rightSum = rightSum + 1;
                        end
                        if strcmpi(handles.rollpanel3.Visible, 'on')
                            centerSum = centerSum + 0;
                        end
                    case 2
                        if strcmpi(handles.rollpanel2.Visible, 'on')
                            rightSum = rightSum + 1;
                        end

                    case 3
                        rightSum = rightSum + 1;
                        if strcmpi(handles.rollpanel2.Visible, 'on')
                            centerSum = centerSum + 1;
                        end
                    case 4
                        if strcmpi(handles.rollpanel3.Visible, 'on')
                            leftSum = leftSum + 1;
                        end

                    case 5
                        leftSum = leftSum + 1;
                        rightSum = rightSum + 1;
                        centerSum = centerSum + 1;
                    case 0
                        if strcmpi(handles.rollpanel2.Visible, 'on')

                            leftSum = leftSum + 1;
                        end
                        if strcmpi(handles.rollpanel3.Visible, 'on')

                            centerSum = centerSum + 1;
                        end
                end

                % Logic to handle score wrapping in case the chip needs to
be
                % passed left or right and the index is out of the
dimensions
                % of scoresArray
                if roundCounter - 1 == 0 && leftSum > 0
                    scoresArray(end) = scoresArray(end) + leftSum;
                    scoresArray(roundCounter) = scoresArray(roundCounter)
- leftSum;
                elseif leftSum > 0

                    scoresArray(roundCounter - 1) =
scoresArray(roundCounter - 1) + leftSum;
                    scoresArray(roundCounter) = scoresArray(roundCounter)
- leftSum;
                end
                if roundCounter + 1 > numPlayers && rightSum > 0

                    scoresArray(1) = scoresArray(1) + rightSum;
                    scoresArray(roundCounter) = scoresArray(roundCounter)
- rightSum;
                elseif rightSum > 0

                    scoresArray(roundCounter + 1) =
scoresArray(roundCounter +1) + rightSum;
                    scoresArray(roundCounter) = scoresArray(roundCounter)
- rightSum;
                end

                % Logic to set the pot count and update the Pot Chips
section
                scoresArray(roundCounter) = scoresArray(roundCounter) -
centerSum;
                set(handles.potcount, 'string',
string(eval(get(handles.potcount, 'String')) + centerSum));

                % Error handling in case players drop to negatives
                if scoresArray(1) < 0
                    set(handles.playeronescore,'string','0');
                else
                    set(handles.playeronescore,'string',scoresArray(1));
                end
                if scoresArray(2) < 0
                    set(handles.playertwoscore,'string','0');
                else
                    set(handles.playertwoscore,'string',scoresArray(2));
                end
                if scoresArray(3) < 0
                    set(handles.playerthreescore,'string','0');
                else
                    set(handles.playerthreescore,'string',scoresArray(3));
                end

                count = count + 1

                % Set player scores into scoresArray
                scoresArray(1) =
eval(get(handles.playeronescore, 'String'));
```

```
            scoresArray(2) =
eval(get(handles.playertwoscore, 'String'));
            scoresArray(3) =
eval(get(handles.playerthreescore, 'String'));

            % Logic to determine if any players have won the game
            if ( (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String'))) ||
(numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playertwoscore, 'String'))) ||
(numPlayers*3 == eval(get(handles.potcount, 'String'))
+ eval(get(handles.playerthreescore, 'String')))
||( eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String')) +
eval(get(handles.playertwoscore, 'String')) +
eval(get(handles.playerthreescore, 'String')) > numPlayers * 3))

                if (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playeronescore, 'String')))
                    winnerName = namesArray(1);
                    winnerScore =
eval(get(handles.playeronescore, 'String'));
                elseif (numPlayers*3 ==
eval(get(handles.potcount, 'String')) +
eval(get(handles.playertwoscore, 'String')))
                    winnerName = namesArray(2);
                    winnerScore =
eval(get(handles.playertwoscore, 'String'));
                else
                    winnerName = namesArray(3);
                    winnerScore =
eval(get(handles.playerthreescore, 'String'));
                end

                % Open up the winner screen GUI
                close(gameGUI);
                run('WinnerScreen')

            end

        end

end


die =

  DiceClass with properties:

          numSides: 6
    faceValueArray: 'LRC111'
```

```
die = 

  DiceClass with properties:

          numSides: 6
    faceValueArray: 'LRC111'
```



Published with MATLAB® R2017a