```
function varargout = PolyDigger(varargin)
% POLYDIGGER MATLAB code for PolyDigger.fig
%      POLYDIGGER, by itself, creates a new POLYDIGGER or raises the
 existing
%      singleton*.
%
%      H = POLYDIGGER returns the handle to a new POLYDIGGER or the
 handle to
%      the existing singleton*.
%
%      POLYDIGGER('CALLBACK',hObject,eventData,handles,...) calls the
 local
%      function named CALLBACK in POLYDIGGER.M with the given input
 arguments.
%
%      POLYDIGGER('Property','Value',...) creates a new POLYDIGGER or
 raises the
%      existing singleton*.  Starting from the left, property value
 pairs are
%      applied to the GUI before PolyDigger_OpeningFcn gets called.
 An
%      unrecognized property name or invalid value makes property
 application
%      stop.  All inputs are passed to PolyDigger_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 PolyDigger

% Last Modified by GUIDE v2.5 13-Feb-2016 17:15:33

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PolyDigger_OpeningFcn, ...
                   'gui_OutputFcn',  @PolyDigger_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
end

% --- Executes just before PolyDigger is made visible.
function PolyDigger_OpeningFcn(hObject, ~, 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 PolyDigger (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

set(handles.trap_bt,'Enable','off');
set(handles.mono_bt,'Enable','off');
set(handles.tria_bt,'Enable','off');
set(handles.cord_ck,'Enable','off');
set(handles.trap_ck,'Enable','off');
set(handles.diag_ck,'Enable','off');
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
POLY_BT = 0; TRAP_BT = 0; MONO_BT = 0; TRIA_BT = 0;
TRAP_CK = 0; DIAG_CK = 0;

zoom on
set (gcf, 'WindowButtonMotionFcn', @mouseMove);
warning off MATLAB:HandleGraphics:ObsoletedProperty:JavaFrame;
javaFrame = get(hObject,'JavaFrame');
javaFrame.setFigureIcon(javax.swing.ImageIcon('cg_mini.png'));
%javax.swing.UIManager.setLookAndFeel('com.sun.java.swing.plaf.windows.WindowsLook
%#ok<*DEFNU>
end

function mouseMove(~, ~)
C = get (gca, 'CurrentPoint');
x = C(1,1);
y = C(1,2);
xl = xlim;
yl = ylim;
if(x>xl(1) && x<xl(2) && y>yl(1) && y<yl(2))
     title(gca, ['\fontsize{7} \rm(X,Y) = (', num2str(x,3), ',
 ',num2str(y,3), ')']);
else
     title(gca, '');
end
end
```

```
% --- Outputs from this function are returned to the command line.
function varargout = PolyDigger_OutputFcn(~, ~, 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;
end

function vert_tx_Callback(~, ~, ~)
% hObject    handle to vert_tx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of vert_tx as text
%        str2double(get(hObject,'String')) returns contents of vert_tx
 as a double
end

% --- Executes during object creation, after setting all properties.
function vert_tx_CreateFcn(hObject, ~, ~)
% hObject    handle to vert_tx (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
 called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx RANDOM POLYGON
 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
% --- Executes on button press in poly_bt.
function poly_bt_Callback(hObject, ~, handles)
% hObject    handle to poly_bt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(TRAP_BT || MONO_BT || TRIA_BT || TRAP_CK || DIAG_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    POLY_BT = 0;
    return;
end;
POLY_BT = 1;

set(hObject,'Enable','off');
```

```
global gpoints;
global gn;
global gpoly_hl;
global gtrap_sz;
global gdiag_sz;

n = str2double(get(handles.vert_tx, 'String'));
if(floor(n)~=n)
    msgbox('Enter a valid number!', 'Error','error');
    set(hObject,'Enable','on');
    return;
elseif(n<3)
    msgbox('Enter a value greater than equal to 3!', 'Error','error');
    set(hObject,'Enable','on');
    return;
end
points = randi([-30*n 30*n], 2, n);
%x1 = sum(points(1,:))/n;
%y1 = sum(points(2,:))/n;
[y_min, y_min_idx] = min(points(2, :));
y1 = y_min;
x1 = points(1, y_min_idx);
angle = atan2d(points(2,:)-y1,points(1,:)-x1);
[~,perm] = sort(angle);
p_sorted = points;
p_sorted(1,:) = points(1,perm);
p_sorted(2,:) = points(2,perm);
points = p_sorted;

x_r = points(1,:);
y_r = points(2,:);
x = [x_r; circshift(x_r, 1, 2)];
y = [y_r; circshift(y_r, 1, 2)];
axes(handles.axes1)
%x = [0 1 1 0.5 0; 1 1 0.5 0 0];
%y = [0 0 1 2 1; 0 1 2 1 0];
poly_hl = plot(x, y, 'b', 'DisplayName','Polygon');
axis equal
legend('Polygon');
% text(x1, y1, 'c')
% hold on
% xl = xlim;
% plot([xl(1)-1, xl(2)+1], [points(2,n) points(2,n)], 'r');
% hold off
area = 0;
for j=1:n
    area = area + points(1,j)*points(2,mod(j,n)+1) -
 points(2,j)*points(1,mod(j,n)+1);
end
area = area / 2;
area_str = ['Area: ' num2str(area,'%.2f')];
set(handles.area_tx,'String',area_str);

gpoints = points;
```

```
gn = n;
gpoly_hl = poly_hl;
gtrap_sz = -1;
gdiag_sz = -1;

set(handles.trap_bt,'Enable','on');
set(handles.mono_bt,'Enable','on');
set(handles.tria_bt,'Enable','off');
set(handles.cord_ck,'Enable','on');
set(handles.trap_ck,'Enable','off');
set(handles.diag_ck,'Enable','off');
set(handles.cord_ck,'Value',0);
set(handles.trap_ck,'Value',0);
set(handles.diag_ck,'Value',0);
set(hObject,'Enable','on');
POLY_BT = 0;
end

function show_coordinates()
global gh;
global gpoints;
global gn;
points = gpoints;
n = gn;
h = zeros(1, n);
for j= 1:n
    r = points(:,j);
    h(j) = text(r(1),r(2),['(', num2str(r(1)), ', ',
 num2str(r(2)), ')'], 'Color', 'blue');
end
gh = h;
end

function hide_coordinates()
global gh;
global gn;
n = gn;
h = gh;
for j= 1:n
    delete(h(j));
end
gh = zeros(1, n);
end

% --- Executes on button press in cord_ck.
function cord_ck_Callback(hObject, ~, ~)
% hObject    handle to cord_ck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of cord_ck
val = get(hObject, 'Value');
if val == 1
    show_coordinates();
```

```
else
    hide_coordinates();
end
end

% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx TRAPEZOIDALIZE
 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
% --- Executes on button press in trap_bt.
function trap_bt_Callback(hObject, ~, handles)
% hObject    handle to trap_bt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(POLY_BT || MONO_BT || TRIA_BT || TRAP_CK || DIAG_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    TRAP_BT = 0;
    return;
end;
TRAP_BT = 1;

set(hObject,'Enable','off');
global gpoints;
global gn;
global gtrap;
global gtrap_sz;
global gcusp;
global gcusp_sz;
if(gtrap_sz ~= -1 && get(handles.trap_ck,'Value')==0)
    show_trapezoids();
    set(handles.trap_ck,'Value',1);
    set(hObject,'Enable','on');
    TRAP_BT = 0;
    return;
elseif(gtrap_sz ~=-1)
    set(hObject,'Enable','on');
    TRAP_BT = 0;
    return;
end
points = gpoints;
n = gn;
trap = zeros(2, 2, 1);
cusp = zeros(2, 1);
trap_sz = 0;
cusp_sz = 0;
x_min = min(points(1,:));
x_max = max(points(1,:));
points(1,n+1) = points(1,1);
points(2,n+1) = points(2,1);
for j= 1:n
    x1 = [x_min-1, x_max+1];
    y1 = [points(2,j), points(2,j)];
    [xi, ~] = polyxpoly(x1,y1,points(1,:),points(2,:));
    v1 = xi(xi>points(1,j));
    flag = 1;
    if(mod(numel(v1),2)==1)
        %plot([points(1,j) min(v1)],[y1(1) y1(1)], 'c');
        trap_sz = trap_sz+1;
        trap(:,1,trap_sz) = [points(1,j) min(v1)];
        trap(:,2,trap_sz) = [y1(1) y1(1)];
    else
        flag = 0;
    end
    v2 = xi(xi<points(1,j));
    if(mod(numel(v2),2)==1)
        %plot([points(1,j) max(v2)],[y1(1) y1(1)], 'c');
        trap_sz = trap_sz+1;
        trap(:,1,trap_sz) = [points(1,j) max(v2)];
        trap(:,2,trap_sz) = [y1(1) y1(1)];
    else
        flag = 0;
    end
    if(flag == 1)
        cusp_sz = cusp_sz+1;
        cusp(:,cusp_sz)= [points(1,j) points(2,j)];
    end
    %mapshow(xi,yi,'DisplayType','point','Marker','o');
end

gtrap = trap;
gtrap_sz = trap_sz;
gcusp = cusp;
gcusp_sz = cusp_sz;
show_trapezoids();

set(handles.trap_ck,'Enable','on');
set(handles.trap_ck,'Value',1)
set(hObject,'Enable','on');
TRAP_BT = 0;
end

function show_trapezoids()
global gtrap;
global gpoly_hl;
global gtrap_hl;
global gcusp_hl;
global gtrap_sz;
global gcusp;
global gcusp_sz;
poly_hl = gpoly_hl;
trap = gtrap;
trap_sz = gtrap_sz;
cusp = gcusp;
cusp_sz = gcusp_sz;
trap_hl = zeros(1, trap_sz);
cusp_hl = zeros(1, cusp_sz);
hold all;
for j=1:trap_sz
```

```
    trap_hl(j) = plot(trap(:,1,j), trap(:,2,j), 'y');
end
for j= 1:cusp_sz
    r = cusp(:,j);
    cusp_hl(j) = plot(r(1),r(2),'y*');
end
if(cusp_sz>0)
    legend([poly_hl(1) trap_hl(1)
 cusp_hl(1)],'Polygon', 'Trapezoids', 'Cusps');
else
    legend([poly_hl(1) trap_hl(1)],'Polygon', 'Trapezoids');
end
hold off
gtrap_hl = trap_hl;
gcusp_hl = cusp_hl;
end

function hide_trapezoids()
global gtrap_hl;
global gtrap_sz;
global gcusp_hl;
global gcusp_sz;
trap_sz = gtrap_sz;
trap_hl = gtrap_hl;
for j=1:trap_sz
    delete(trap_hl(j));
end
cusp_sz = gcusp_sz;
cusp_hl = gcusp_hl;
for j=1:cusp_sz
    delete(cusp_hl(j));
end
end

% --- Executes on button press in trap_ck.
function trap_ck_Callback(hObject, ~, ~)
% hObject    handle to trap_ck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of trap_ck
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(POLY_BT || TRAP_BT || MONO_BT || TRIA_BT || DIAG_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    TRAP_CK = 0;
    return;
end;
TRAP_CK = 1;

set(hObject,'Enable','off');
val = get(hObject, 'Value');
if val == 1
    show_trapezoids();
```

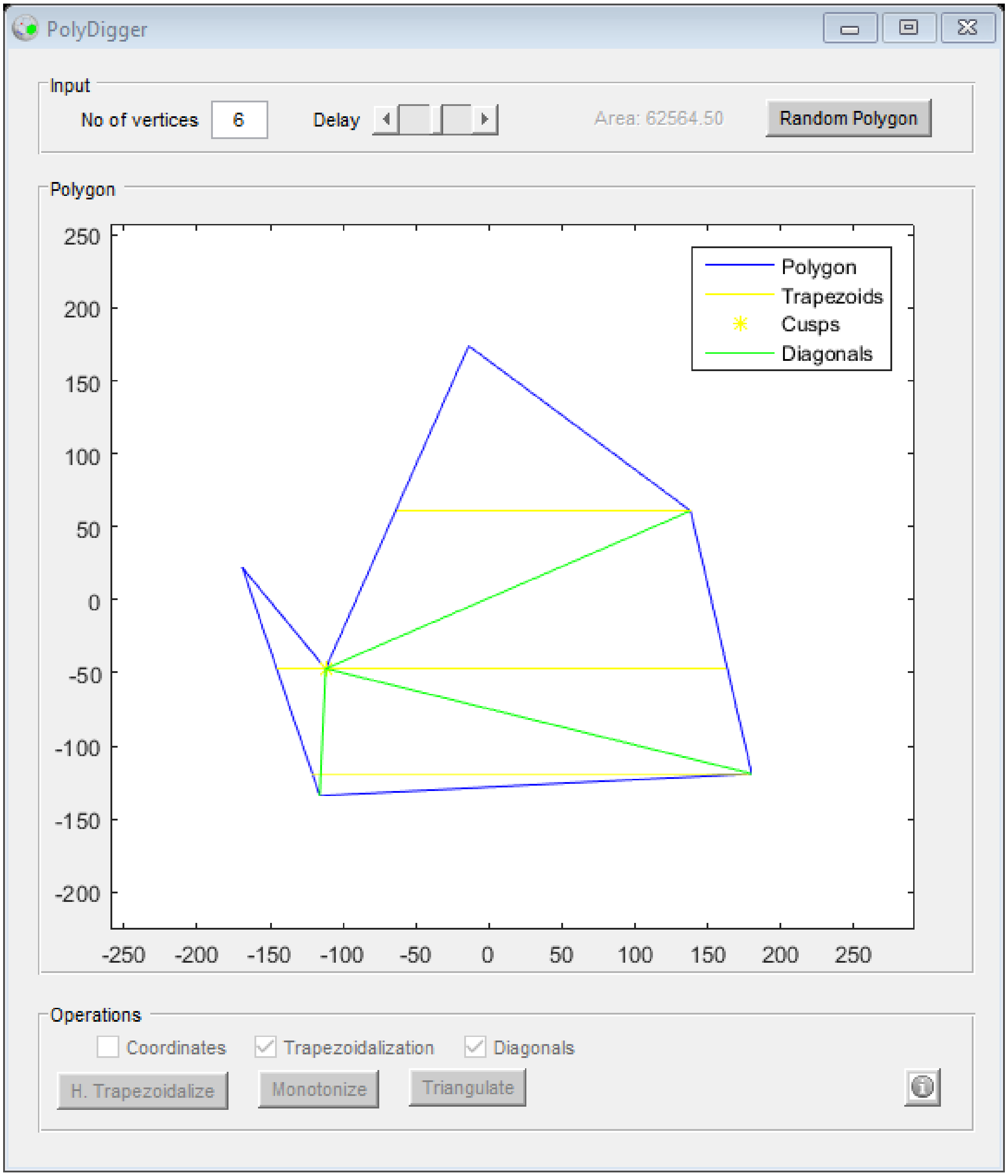
```
else
    hide_trapezoids();
end
set(hObject, 'Enable', 'on');

TRAP_CK = 0;
end

% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx MONOTONIZE
 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
% --- Executes on button press in mono_bt.
function mono_bt_Callback(hObject, ~, handles)
% hObject    handle to mono_bt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(POLY_BT || TRAP_BT || TRIA_BT || TRAP_CK || DIAG_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    MONO_BT = 0;
    return;
end;
MONO_BT = 1;

set(hObject,'Enable','off');
global gpoints;
global gn;
global gdiag;
global gdiag_sz;
global gdiag_hl;
global gpoly_hl;
global gdiag_hl_sz;
global TCALLD;
if(gdiag_sz ~= -1)
    hide_diagonals();
end
diag = zeros(1, 4);
diag_hl = zeros(1);
diag_sz = 0;
diag_hl_sz = 0;
points = gpoints;
o_points = points;
ot_points = transpose(points);
n = gn;
delay = get(handles.dlay_sl, 'Value');
[points(2,:), perm] = sort(points(2, :), 'descend');
points(1,:) = points(1,perm);
o_points(1,n+1) = o_points(1,1);
o_points(2,n+1) = o_points(2,1);
hold on
xl = xlim;
x_min = min(points(1,:));
x_max = max(points(1,:));
for j=1:n
```

```
        x1 = [x_min-1, x_max+1];
        y1 = [points(2,j), points(2,j)];
        [xi, yi] = polyxpoly(x1,y1,o_points(1,:),o_points(2,:));
        v1 = xi(xi>points(1,j));
        v2 = xi(xi<points(1,j));
        if(mod(numel(v1),2)==1 && mod(numel(v2),2)==1)
            %points(:,j);
            [~,idx]= ismember(transpose(points(:,j)), ot_points, 'rows');
            %o_points(:,idx-1);
            %o_points(:,idx+1);
            %here only downward cusp
            p1=-1;
            for k=idx+1:n
                if(o_points(2,k)<points(2,j))
                    p1=k;
                    break;
                end
            end
            p2=-1;
            for k=idx-1:-1:1
                if(o_points(2,k)<points(2,j))
                    p2=k;
                    break;
                end
            end
            if(p1~=-1 && p2~=-1 && o_points(2,p2)>o_points(2,p1))
                p1 = p2;
            elseif(p1==-1 && p2~=-1)
                p1 = p2;
            end
            o_points(:,p1);
            diag_hl_sz = diag_hl_sz+1;
            diag_hl(diag_hl_sz) = plot([points(1,j) o_points(1,p1)],
     [points(2,j) o_points(2,p1)], 'g');
            diag_sz = diag_sz+1;
            diag(diag_sz,:)=[points(1,j) points(2,j) o_points(1,p1)
     o_points(2,p1)];
        end
        if(delay>0)
            ph = plot([xl(1)-1, xl(2)+1], [points(2,j) points(2,j)], 'r');
            mh = mapshow(xi,yi,'DisplayType','point','Marker','o');
            pause(delay);
            delete(ph);
            delete(mh);
        end
    end
    if(diag_sz>0)
        legend([gpoly_hl(1) diag_hl(1)], 'Polygon', 'Diagonals');
    end
    hold off
    gdiag = diag;
    gdiag_sz = diag_sz;
    gdiag_hl = diag_hl;
    gdiag_hl_sz = diag_hl_sz;
```

```
TCALLD = 0;

set(handles.tria_bt,'Enable','on');
set(handles.diag_ck,'Enable','off');
set(handles.diag_ck,'Value',0);
set(hObject,'Enable','on');

MONO_BT = 0;
end

% --- Executes on slider movement.
function dlay_sl_Callback(hObject, ~, handles)
% hObject    handle to dlay_sl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range
 of slider
val = get(hObject,'Value');
if(val == 0)
    set(hObject,'TooltipString', 'No animations!');
    set(handles.mono_bt,'TooltipString', 'No animations!');
    set(handles.tria_bt,'TooltipString', 'No animations!');
elseif(val == 1)
    set(hObject,'TooltipString', 'Might take long time to
 triangulate!');
    set(handles.mono_bt,'TooltipString', 'Might take long time,
 consider reducing delay!');
    set(handles.tria_bt,'TooltipString', 'Might take long time,
 consider reducing delay!!');
else
    set(hObject,'TooltipString', '');
    set(handles.mono_bt,'TooltipString', '');
    set(handles.tria_bt,'TooltipString', '');
end
end

% --- Executes during object creation, after setting all properties.
function dlay_sl_CreateFcn(hObject, ~, ~)
% hObject    handle to dlay_sl (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
 called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
 get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
end
```

```
% xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx TRIANGULATE
 xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
% --- Executes on button press in tria_bt.
function tria_bt_Callback(hObject, ~, handles)
% hObject    handle to tria_bt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(POLY_BT || TRAP_BT || MONO_BT || TRAP_CK || DIAG_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    TRIA_BT = 0;
    return;
end;
TRIA_BT = 1;

set(hObject,'Enable','off');
global gpoints;
global gn;
global gdiag;
global gdiag_sz;
global gdiag_hl;
global gpoly_hl;
global gdiag_hl_sz;
global TCALLD;
if(TCALLD==1)
    show_diagonals();
    set(handles.mono_bt,'Enable','on');
    set(handles.trap_bt,'Enable','on');
    set(handles.trap_ck,'Enable','on');
    set(hObject,'Enable','on');
    TRIA_BT = 0;
    return;
end
delay = get(handles.dlay_sl, 'Value');
diag = gdiag;
diag_sz = gdiag_sz;
points = gpoints;
diag_hl = gdiag_hl;
diag_hl_sz = gdiag_hl_sz;
o_points = points;
ot_points = transpose(points);
n = gn;
[points(2,:), perm] = sort(points(2, :), 'descend');
points(1,:) = points(1,perm);
o_points(1,n+1) = o_points(1,1);
o_points(2,n+1) = o_points(2,1);
x_max = max(points(1,:));

hold on;
for j=2:n
    x1=points(1,j);
    y1=points(2,j);
    for k=1:j-1
```

```
        x2=points(1,k); y2=points(2,k);
        [~,idx] = ismember(transpose(points(:,j)), ot_points, 'rows');
        if(isequal([x2 y2],ot_points(mod(idx,n)+1,:)) ||
 (idx>1&&isequal([x2 y2],ot_points(idx-1,:))) || (idx==1&&isequal([x2
 y2],ot_points(n,:))))
            continue;
        end
        [xi, ~] = polyxpoly([x1 x2],[y1
 y2],o_points(1,:),o_points(2,:));
        if(numel(xi)==2 || (j==n&&numel(xi)==3))
            flag=1;
            for l=1:diag_sz
                x3=diag(l,1);y3=diag(l,2);x4=diag(l,3);y4=diag(l,4);
                if(isequal([x1 y1 x2 y2],diag(l,:))||isequal([x2 y2 x1
 y1],diag(l,:)))
                    flag=0; break;
                elseif(isequal([x1 y1],[x3 y3])||isequal([x1 y1],[x4
 y4])||isequal([x2 y2],[x3 y3])||isequal([x2 y2],[x4 y4]))
                    continue;
                end
                [xj, ~] = polyxpoly([x1 x2],[y1 y2],[x3 x4],[y3 y4]);
                if(numel(xj)>0)
                    flag=0; break;
                end
            end
            if(flag==1)
                xm =(x1+x2)/2; ym =(y1+y2)/2;
                [xk, ~] = polyxpoly([xm x_max+1],[ym
 ym],o_points(1,:),o_points(2,:));
                if(mod(numel(xk),2)==1)
                    diag_hl_sz = diag_hl_sz+1;
                    diag_hl(diag_hl_sz) = plot([x1 x2], [y1 y2], 'g');
                    diag_sz = diag_sz+1;
                    diag(diag_sz,:)=[x1 y1 x2 y2];
                    if(delay>0)
                        pause(delay);
                    end
                end
            end
        end
    end
end
if(diag_sz>0)
    legend([gpoly_hl(1) diag_hl(1)], 'Polygon', 'Diagonals');
end
hold off;
gdiag = diag;
gdiag_sz = diag_sz;
gdiag_hl = diag_hl;
gdiag_hl_sz = diag_hl_sz;
TCALLD = 1;

set(handles.diag_ck,'Enable','on');
set(handles.diag_ck,'Value',1);
set(hObject,'Enable','on');

TRIA_BT = 0;
end

function show_diagonals()
global gdiag;
global gdiag_sz;
global gdiag_hl;
global gdiag_hl_sz;
global gpoly_hl;
global gtrap_hl;
global gcusp_hl;
diag = gdiag;
diag_sz = gdiag_sz;
diag_hl = gdiag_hl;
diag_hl_sz = gdiag_hl_sz;
hold on
for j=1:diag_sz
    x1=diag(j,1); x2=diag(j,3); y1=diag(j,2); y2=diag(j,4);
    diag_hl_sz = diag_hl_sz+1;
    diag_hl(diag_hl_sz) = plot([x1 x2], [y1 y2], 'g');
end
if(diag_sz>0)
    legend([gpoly_hl(1) gtrap_hl(1) gcusp_hl(1)
 diag_hl(1)], 'Polygon', 'Trapezoids', 'Cusps', 'Diagonals');
end
hold off
gdiag_hl = diag_hl;
gdiag_hl_sz = diag_hl_sz;
end

function hide_diagonals()
global gdiag_hl;
global gdiag_hl_sz;
diag_hl = gdiag_hl;
diag_hl_sz = gdiag_hl_sz;
for j=1:diag_hl_sz
    delete(diag_hl(j));
end
gdiag_hl_sz = 0;
end

% --- Executes on button press in diag_ck.
function diag_ck_Callback(hObject, ~, ~)
% hObject    handle to diag_ck (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of diag_ck
global POLY_BT; global TRAP_BT; global MONO_BT; global TRIA_BT;
global TRAP_CK; global DIAG_CK;
if(POLY_BT || TRAP_BT || MONO_BT || TRIA_BT || TRAP_CK)
    waitfor(msgbox('Some operation is already in progress!'));
    DIAG_CK = 0;
    return;
end;
DIAG_CK = 1;

set(hObject,'Enable','off');
val = get(hObject, 'Value');
if val == 1
    show_diagonals();
else
    hide_diagonals();
end
set(hObject,'Enable','on');

DIAG_CK = 0;
end


% --- Executes on button press in info_bt.
function info_bt_Callback(hObject, eventdata, handles)
% hObject    handle to info_bt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
waitfor(msgbox({'This application is a part of the Coding Project
 submitted for the partial fulfillment of the course CSN-475
 (Computational Geometry) under the guidance of Dr Sudip Roy.';
'';
'Developed By:';
'(15535002) Abhishek Sharma';
'(15535029) Prakhar Dhama';
'Moodle Group# 19'}, 'Info', 'help'));
end
```



No of vertices 6

Area: 62564.50

☐ Coordinates ☑ Trapezoidalization ☑ Diagonals

Published with MATLAB® R2015b