Projekts:

Valsts pētījumu programmas SOPHIS Projekts Nr.4. „Tehnoloģijas drošai un uzticamai gudrajai pilsētai”

Plūdu situāciju lokālas simulēšanas programma “AdazuPludi”

Lietošanas apraksts

Autori:

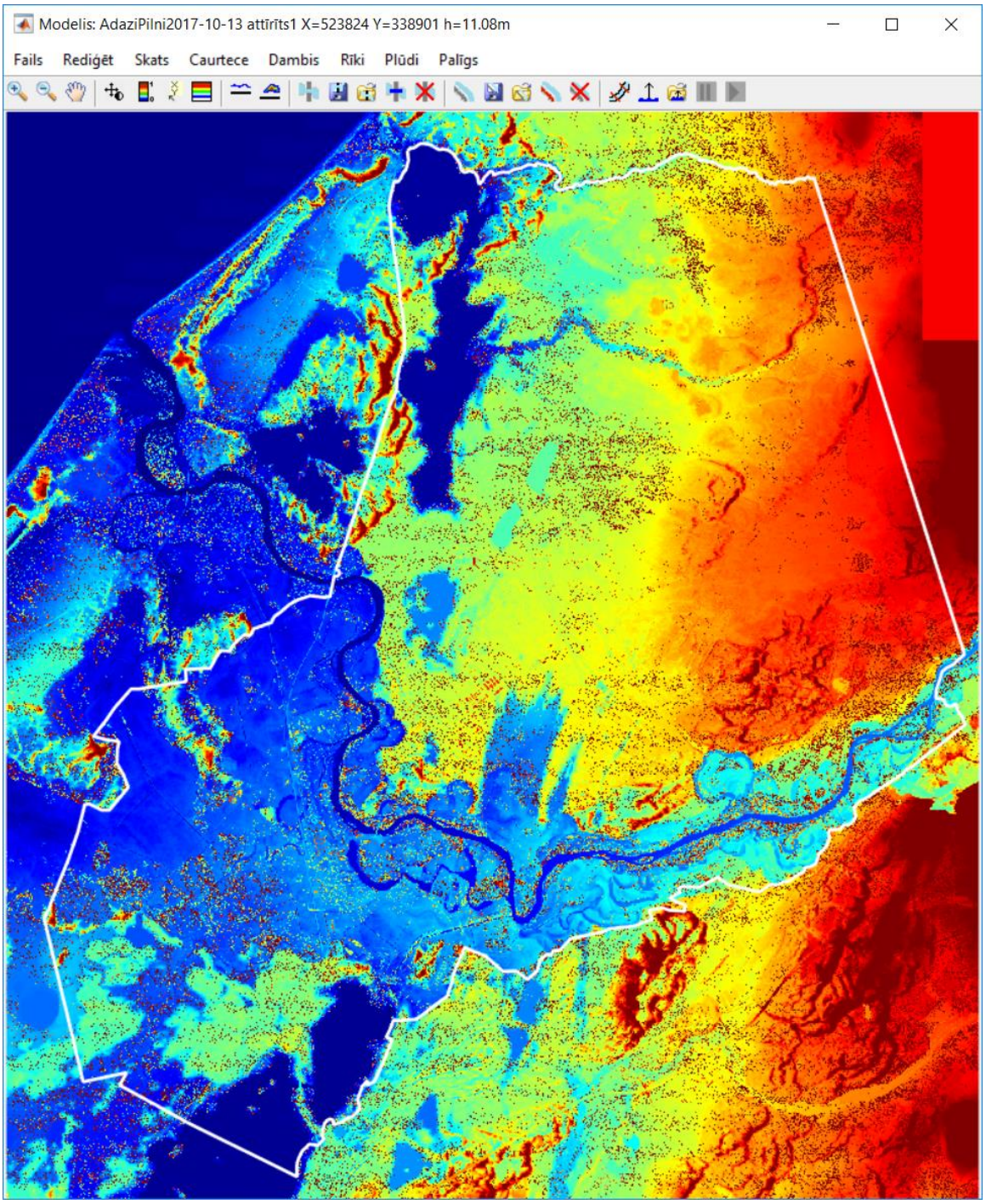
Ints Mednieks

2017.g.

Saturs

Ievads

AdazuPludi ir programma, kas domāta lokālai plūdu imitēšanai, balstoties uz zemes reljefa augstuma v.j.l. modeli, kas veidots no lāzerskenēšanas datiem. Programma neizmanto hidroloģisko informāciju, tādēļ ar tās palīdzību var imitēt plūdu izplatību ūdens līmeņa celšanās dēļ, bet nevar iegūt informāciju par plūdu dinamiku. Programmas logs ar Ādažu novada modeli redzams zemāk.



Zemes augstuma modelis veidots no Latvijas Ģeotelpiskās informācijas aģentūras piegādātajiem LAS formāta datiem, kuri satur lāzera stara atstarošanās informāciju ar vidējo blīvumu 4 atstarojumi/ m2. No šiem datiem veidots modelis (augstuma attēls) ar telpisko izšķirtspēju 1m, no katra lāzera impulsa izmantojot tikai augstumu v.j.l., kas iegūts no pēdējā atstarojuma. Tā kā lāzera stars neatstarojas no ūdens virsmas, programma satur līdzekļus ūdens līmeņa augstuma v.j.l. novērtēšanai ūdenstilpnēs un modeļa papildināšanai ar šiem datiem. Ezeru u.c. noslēgtu ūdentilpņu augstums tiek v.j.l. novērtēts, izmantojot krasta pikseļu augstumu (pēc šī paša principa piepildīti arī citi "caurumi" datos). Ūdens līmenis upēs arī novērtēts pēc krasta pikseļu augstuma, bet ierobežotā attālumā, lai ņemtu vērā upes augstuma profilu. Tā kā bieza meža rajonos atstarojumi no zemes virsmas saņemti reti, programmā paredzēts rīks meža filtrēšanai, ar kura palīdzību tiek paplašināta atstarojuma no zemes ietekme, tādejādi panākot, ka arī meža rajonu pikseļos modelis satur augstuma datus no zemes virsmas un ir iespējama ūdens izplatīšanās t.i. plūdu imitācija tajos.

Programma pārbaudīta darbā ar Ādažu novada un apkārtnes augstuma modeli; citu apvidu datu apstrādei, iespējams, nepieciešamas modifikācijas. Lai tuvinātu augstuma modeli reālajiem apstākļiem, tajā iebūvēti līdzekļu caurteku (zem tiltiem un ceļiem) un jaunveidojamu dambju informācijas ievadīšanai.

1. Galvenie principi

Startējot programmu, tā atver noklusēto Ādažu novada daļas augstuma modeli. Sagatavoti arī citu apgabalu modeļi un kopējais novada modelis. Atkarībā no pieejamajiem datora resursiem, nepieciešams programmu darbināt ar piemērota izmēra modeli, lai datoru nepārslogotu.

Augstuma modeļi glabājas failos ar paplašinājumu "*.hedi", kuri satur augstuma un koordinātu informāciju. Modelis tiek nolasīts no faila un attēlots logā, augstuma informācija tiek vizualizēta, izmantojot pikseļa krāsu. Krāsu skala un atbilstošie augstumi tiek attēloti, startējot programmu; vēlāk to var paslēpt, izmantojot rīku joslas funkciju **Ieslēgt / izslēgt krāsu skalu**.

Pārvietojoties pa vizualizētā modeļa attēlu ar peles kursoru, augstuma informācija v.j.l. tiek dinamiski attēlota programmas loga nosaukumā kopā ar attiecīgā punkta koordinātēm LKS-92 sistēmā. Šo informāciju var arī attēlot blakus peles kursoram.

Programma ļauj modeli mainīt, pa vienam iezīmējot tajā caurteces un dambjus. To definīcijas var saglabāt failos kopā ar ģeogrāfiskajā koordinātēm. Tādejādi saglabātos objektus var lietot dažādos attēlos.

2. Sistēmas prasības

Datora konfigurācija, kurā var izmantot programmu:

- Procesors: Intel Core i5/ 2GHz vai jaudīgāks
- Operatīvā atmiņa: vismaz 8GB
- Monitors ar vismaz HD izšķirtspēju (1920 x 1080 pikseļi)
- Diska vieta: vismaz 20GB (atkarīgs no saglabājamo modeļu skaita)
- Operāciju sistēma: Windows 7 vai vēlāka

3. Instalēšana

Instalēšanai nepieciešams piestartēt izplatīto failu "AdaziInstaller_web.exe" un sekot instrukcijām. Programma realizēta MATLAB izstrādes vidē (skat. https://www.mathworks.com/products/matlab.html). Instalēšanai nepieciešams interneta pieslēgums (jāatkopē MATLAB izpildes laika bibliotēka ar izmēru apm.500MB.

4. Funkcijas

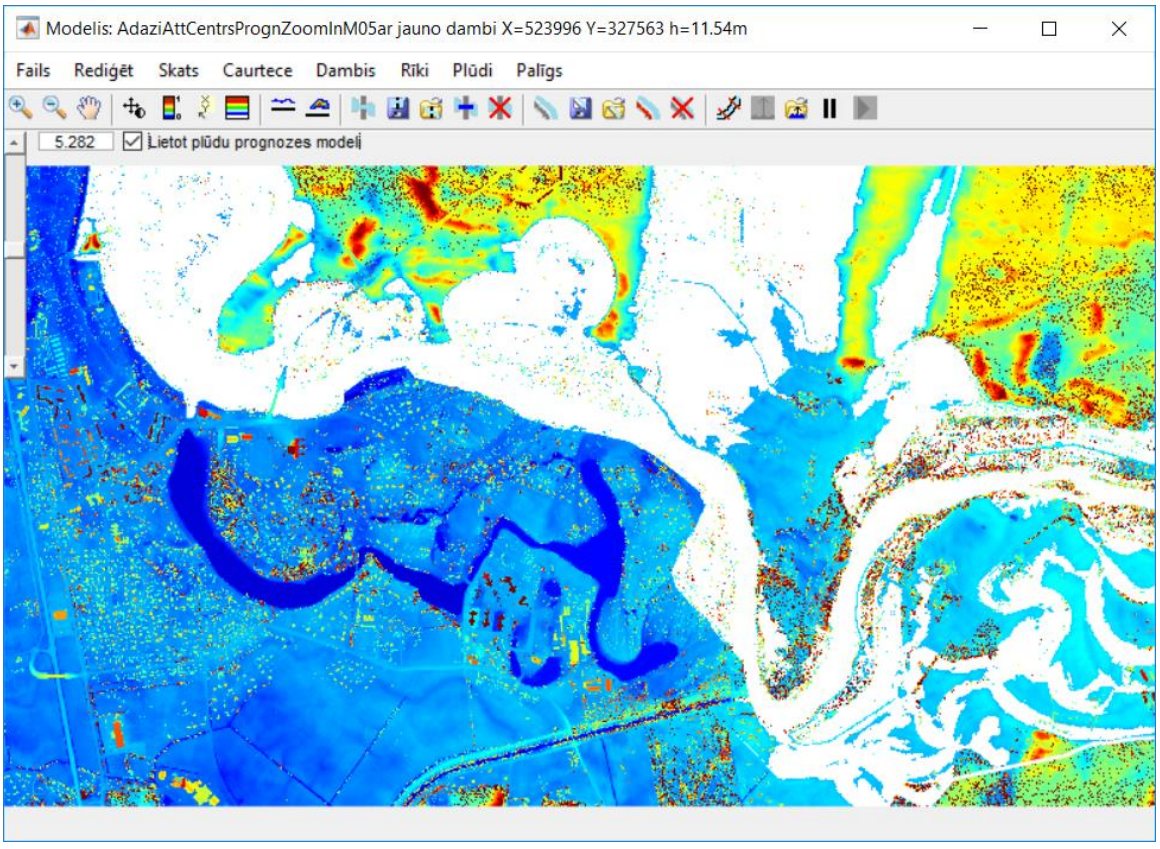
Programmas izvēlne un rīku josla satur līdzekļus šādu funkciju veikšanai:

- **Fails / Atvērt augstuma modeli** (izvēlnes funkcija). Šī funkcija ļauj izvēlēties failu ar paplašinājumu "*.hedi", kurā ir augstuma un koordinašu informācija. Modeļa dati tiek nolasīti no faila un modelis tiek attēlots logā, augstuma informāciju kodējot ar pikseļa krāsu.
- **Fails / Saglabāt jaunu modeļa versiju** (izvēlnes funkcija). Šī funkcija ļauj izvēlēties failu ar paplašinājumu "*.hedi", kurā saglabāt jaunu modeļa versiju ar pašreiz ieviestajām izmaiņām (caurtecēm, dambjiem, filtrētiem mežiem utt.)
- **Fails / Saglabāt modeļa fragmentu** (izvēlnes funkcija). Šī funkcija ļauj iezīmēt attēla fragmentu, velkot pār to ar peli, un saglabāt modeļa fragmentu izvēlētajā "*.hedi" failā.
- **Rediģēt / Kopēt attēlu** (izvēlnes funkcija). Šī funkcija kopē pašreiz redzamo attēlu uz starpliktuvi, lai to izmantotu citā (piem. dokumentu gatavošanas) programmā.
- **Rediģēt / Piepildīt ezeru ar augstuma datiem** (izvēlnes funkcija). Šī funkcija ļauj attēlā izvēlēties ezeru, noklikšķinot uz to ar peli. Pēc tam programma nosaka un iezīmē ezera robežas un pārvaicā, vai tas izdarīts pareizi. Ja lietotājs to apstiprina, tiek aprēķināts ezera augstums v.j.l. (vienāds visam ezeram) un ezera objekts modelī tiek aizpildīts ar šo lielumu.
- **Rediģēt / Atzīmēt upes/ezera robežu** (izvēlnes funkcija). Ja ezers saistīts ar upi, būs nepareizi šiem abiem objektiem piešķirt vienu un to pašu augstuma līmeni. Šī funkcija ļauj attēlā atdalīt ezeru no upes, lai tos varētu apstrādāt atsevišķi. Lai to veiktu, vietā, kur upe ietek ezerā vai iztek no ezera, funkcija piedāvā ar peli iezīmēt atdalošo līniju. Līnijas pikseļiem tiks piešķirtas augstuma vērtības, interpolējot starp sākuma un beigu punktu augstumiem.
- **Rediģēt / Piepildīt upi ar augstuma datiem no krastiem** (izvēlnes funkcija). Lai piepildītu upes objektu ar augstuma datiem, kas seko upes augstuma profilam, šī funkcija piedāvā atzīmēt upes punktus virzienā no lejteces uz augšteci. Tiks izveidots upes augstuma profils, atrodot katra atzīmētā punkta apkārtnē krasta zemākos pikseļus. Upes pikseļiem ūdens līmeņa augstums tiks piešķirts vienāds ar tuvākā augstuma profila pikseļa vērtību.
- **Rediģēt / Piepildīt visas atlikušās vietas, kurām nav datu (h=0)** (izvēlnes funkcija). Šī funkcija ļauj piepildīt ar augstuma datiem visus objektus, kuri līdz šim palikuši bez tiem. Tas tiks darīts, katram objektam izmantojot tādu pašu pieeju, kā ezera piepildīšanai ar augstuma datiem.
- **Rediģēt / Atcelt iepriekšējo piepildīšanas operāciju** (izvēlnes funkcija). Šī funkcija ļauj atcelt iepriekšējo ezera, upes vai citu objektu piepildīšanas operāciju ar augstuma datiem.
- **Krāsu diapazona izvēle** (izvēlnes "Skats" un rīku joslas funkcija). Šī funkcija atver MATLAB iebūvētā kontrasta uzlabošanas rīka logu, kurā var izvēlēties attēlojamo datu zemāko (Window-Minimum) un augstāko (Window-Maximum) vērtību.
- **Attēlot zemu augstumu baltā krāsā** (izvēlnes "Skats" un rīku joslas funkcija). Šī funkcija ieslēdz/izslēdz zemu vietu (augstums vienāds vai zemāks par krāsu diapazona zemāko vērtību) attēlošanu ar baltu krāsu.
- **Skats / Rādīt novada robežas** (izvēlnes funkcija). Šī funkcija ļauj uz attēla fona atzīmēt Ādažu novada robežas.
- **Atzīmēt caurteci** (izvēlnes "Caurtece" un rīku joslas funkcija). Šī funkcija ļauj uz attēla atzīmēt vienu caurteces viduslīniju. Tā paliek attēlā, un to var rediģēt, velkot ar peli galapunktus.
- **Saglabāt caurteci failā** (izvēlnes "Caurtece" un rīku joslas funkcija). Šī funkcija ļauj saglabāt caurteces definīciju failā. Tā paliek attēlā, un to var rediģēt, velkot ar peli galapunktus.

- **Nolasīt caurteci** (izvēlnes “Caurtece” un rīku joslas funkcija). Šī funkcija ļauj nolasīt caurteces definīciju, kas iepriekš ierakstīta failā. Tā tiek atzīmēta attēlā, un to var rediģēt, velkot ar peli galapunktus.
- **Ieviest caurteci modelī** (izvēlnes “Caurtece” un rīku joslas funkcija). Šī funkcija izpildāma tad, kad caurtece ir pareizi atzīmēta un nepieciešams to ieviest modelī. Caurtece tiek ieviesta modelī, interpolējot augstumus starp atzīmētajiem galapunktiem.
- **Dzēst caurteci** (izvēlnes “Caurtece” un rīku joslas funkcija). Šī funkcija domāta caurteces definīcijas dzēšanai bez ieviešanas modelī.
- **Atzīmēt dambja līniju** (izvēlnes “Dambis” un rīku joslas funkcija). Šī funkcija ļauj uz attēla atzīmēt lauztu imitēta dambja viduslīniju, klikšķinot uz tās virsotnēm. Pēdējo punktu jāatzīmē ar dubultklikšķi. Dambja līnija paliek attēlā, un to var rediģēt, velkot ar peli virsotnes. Virsotnes ir numurētas, un katras plānoto augstumu v.j.l. jāatzīmē, klikšķinot uz numura.
- **Saglabāt dambi failā** (izvēlnes “Dambis” un rīku joslas funkcija). Šī funkcija ļauj saglabāt dambja definīciju (virsotņu koordinātes un augstumus) failā.
- **Nolasīt dambi** (izvēlnes “Dambis” un rīku joslas funkcija). Šī funkcija ļauj nolasīt dambja definīciju, kas iepriekš ierakstīta failā. Tā tiek atzīmēta attēlā, un to var rediģēt, velkot ar peli virsotnes, kā arī atzīmējot to augstumus.
- **Ieviest dambi modelī** (izvēlnes “Dambis” un rīku joslas funkcija). Šī funkcija izpildāma tad, kad dambis un virsotņu augstumi ir pareizi atzīmēti un nepieciešams to ieviest modelī. Dambis tiek ieviests modelī, interpolējot augstumus starp atzīmētajām virsotnēm.
- **Dzēst dambi** (izvēlnes “Dambis” un rīku joslas funkcija). Šī funkcija domāta dambja definīcijas dzēšanai bez ieviešanas modelī.
- **Augstuma profils līnijai** (izvēlnes “Rīki” un rīku joslas funkcija). Šī funkcija domāta augstuma profila nolasīšanai no modeļa un attēlošanai grafikā atsevišķā logā. Funkcija piedāvā ievadīt lauztu līniju, ar peles klikšķiem atzīmējot tās virsotnes (pēdējo ar dubultklikšķi). Pēc līnijas ievades atzīmētās līnijas augstuma profils tiks attēlots grafikā. Līnijas virsotnes attēlā var rediģēt, šajā gadījumā augstuma profils tiks izmainīts. Attēlā var atzīmēt vairākas līnijas profilu attēlošanai.
- **Augstuma modelis apgabalam** (izvēlnes “Rīki” un rīku joslas funkcija). Šī funkcija domāta 3D objekta attēlošanai, kura pamats attēlā tiek atzīmēts ar daudzstūri (pēdējā virsotne jāsavieto ar pirmo. 3D objekts tiks attēlots atsevišķā logā un to var grozīt, izmantojot funkciju **Rotate 3D** šajā logā.
- **Rīki / Filtrēt mežu** ir funkcija, kas ļauj nofiltrēt koku vainagus mežā, lai plūdu simulācija varētu darboties arī rajonos, kur koku lapotne nelaida cauri lāzera staru un atstarojumi saņemti pamatā tikai no vainagiem.
- **Rīki / Atcelt pēdējo filtrēšanas rezultātu** ir funkcija, kas ļauj atcelt iepriekšējās filtrācijas rezultātu un atjaunot modeļa versiju pirms tās.
- **Palīgs / Lietotāja rokasgrāmata** ir funkcija, kas ļauj atvērt šo tekstu.
- **Palīgs / Par** funkcija atver logu ar informāciju par šo programmu.
- **Pietuvināt, Attālināt, Bīdīt** ir parastie attēla novērošanas rīki, kas maina mērogu un novēroto attēla apgabalu.
- **Ieslēgt / izslēgt krāsu skalu** ir funkcija, kas ļauj apskatīties krāsu gammu un atsevišķām krāsām atbilstošo augstumu v.j.l.
- **Rādīt koordinātes un augstumu pie kursora** (rīku joslas funkcija) ir domāta, lai ērti novērotu informāciju par pikseli, uz kuru norāda peles kursors. Koordinātes un augstums šajā gadījumā tiek attēloti ne tikai loga virsrakstā, bet arī blakus kursoram speciālā tekstlodziņā.

- **Atzīmēt upes fragmentu plūdu imitēšanai** (rīku joslas funkcija) ir domāta, lai izvēlētos upes daļu, kuras ūdens līmeņa celšanās tiks imitēta. Plūdu simulācija notiks tikai šī rajona ietvaros.
- **Imitēt ūdens celšanos ūdenstilpnē** (rīku joslas funkcija) ir domāta, lai uzsāktu lokālu plūdu imitēšanu. Kad tā izvēlēta, programma ļauj izvēlēties upes punktu, kura līmeņa celšanos imitēsim. Upes vai tās fragmenta pikseļi tiek attēloti ar baltu krāsu. Attēla kreisajā augšējā pusē parādās ritjosla un ievadāma skaitļa lauciņš, kuri sākumā rāda atzīmētā upes punkta augstumu v.j.l. Ja šis augstums tiek mainīts (to var tikai palielināt), programma imitē visa izvēlētā upes rajona ūdens līmeņa celšanos par tādu pašu lielumu, pārbauda upes krasta pikseļu augstumu un, ja tas ir zemāks par blakus pikseļa augstumu upē, pieskaita pikseli pie upes pikseļiem un attēlo ar baltu krāsu. Tas tiek veikts iteratīvi, katrā solī pārbaudot upes fragmenta sānu blakus pikseļus, kamēr kāds no tiem ir pieskaitāms upei.
- **Apturēt ūdens zonas izplešanās imitāciju** (rīku joslas funkcija). Ja lieto šo funkciju, iteratīvā ūdens izplatīšanās tiek apturēta.
- **Turpināt ūdens zonas izplešanās imitāciju** (rīku joslas funkcija). Ja lieto šo funkciju, iteratīvā ūdens izplatīšanās tiek atjaunota.

Zemāk attēlots palielināts modeļa attēla piemērs Ādažu centra apkārtnē, kurā novērotas teritorijas, kuras potenciāli varētu applūst (iezīmētas ar baltu krāsu), ja Gaujas līmenis pie tilta uz Kadagu ilgstoši saglabājas 5,28m augstumā virs jūras līmeņa. Simulācija darbināta modelim, kurā iestrādāts jaunceļamais dambis augšup gar Gaujas kreiso krastu no tilta uz Kadagu līdz Gaujas- Baltezera kanālam.



Pielikums: programmu kodi

Adazi.m

```
iptsetpref('ImshowBorder','tight');
set(0,'defaulttextinterpreter','none');
clear all

if ismac
    % Code to run on Mac plaform
    javax.swing.UIManager.setLookAndFeel('com.apple.laf.AquaLookAndFeel')
elseif isunix
    % Code to run on Linux plaform
javax.swing.UIManager.setLookAndFeel('com.jgoodies.looks.plastic.Plastic3DLookAndFeel')
elseif ispc
    % Code to run on Windows platform
javax.swing.UIManager.setLookAndFeel('com.sun.java.swing.plaf.windows.WindowsLookAndFeel'
)
else
    errordlg('Platform not supported :(','Error');
end

floodTool
```

floodTool.m

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

% Last Modified by GUIDE v2.5 13-Nov-2017 13:30:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @floodTool_OpeningFcn, ...
                   'gui_OutputFcn',  @floodTool_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 floodTool is made visible.
function floodTool_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 floodTool (see VARARGIN)

% Choose default command line output for floodTool
handles.output = hObject;
handles.mainFigure = gcf;

load (which('AdaziD.hedi'), '-mat');
handles.modelName = 'AdaziD';
cmap = single(colormap(jet(1024)));
cmap(1,:)=1.;
% him=imshow(modelM, [0 20],'colormap', cmap);
% set(gcf,'renderer','painters');
him=imshow(model, [0 20],'colormap', cmap);
colorbar;
set(handles.Colorbar, 'State', 'on');
set(gcf, 'UserData', him);
%colorbar %('northoutside')
hContrast = []; %imcontrast(gca);
handles.hBorder = [];
try
    load (which('AdaziNovads.mat'));
    hold on
    hBorder = plot((Sn.X-X1)/dX+1, (Y1-Sn.Y)/dY+1, 'w-', 'LineWidth', 2);
    handles.hBorder = hBorder;
catch
end

hold on
str = {[' X=' num2str(X1)],[' Y=' num2str(Y1)], [' h=' num2str(model(1,1))]};
handles.posText = text(0,0,str,'Position',[0
0],'VerticalAlignment','bottom','BackgroundColor',[1 1 223/255], 'Margin',0.1,
'Visible','off','FontSize',9);

handles.hImage = him;
handles.colormap = cmap;
handles.hContrast = hContrast;
handles.X1 = X1;
handles.dX = dX;
handles.Y1 = Y1;
handles.dY = dY;

handles.model = model;
BW = model>0;
clear model;
handles.BW0 = BW;   % caurumi pašreizējā variantā (0-ļļu piepildīšanai)
handles.BW1 = BW;   % caurumi iepriekšējā variantā (priekš Undo)
handles.modelOld = [];

handles.walkPos = 1;
handles.walkSpeed=3;
handles.walkWindowSize=1024;
posSteps = [];
handles.posSteps = posSteps;
handles.walkActive = 0;
handles.walkWindow = [];
handles.walkWindowInMain = [];
handles.WaterLevel = [];
```

```
handles.hLinesPoly = [];
handles.hRegionPoly = [];
handles.handlesLines = [];
handles.handlesRegions = [];
handles.floodActive = 0;

handles.hTextDambis = [];
handles.hLineDambis = [];
handles.hCaurtece = [];

handles.hTextFloodForecast = [];
handles.hLineFloodForecast = [];

if ~exist('Upes','var'), Upes = struct('Name',{},'Pixels',{}); end
handles.Upes = Upes;
if ~exist('Prognozes','var'), Prognozes =
struct('Name',{},'UpeName',{},'Pixels',{},'Heights',{}); end
handles.Prognozes = Prognozes;
handles.selectedPrognoze = 0;
% Update handles structure
guidata(hObject, handles);
set(gcf,'CloseRequestFcn',@floodTool_ClosingFcn);

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


% --- Outputs from this function are returned to the command line.
function varargout = floodTool_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;

function floodTool_ClosingFcn(hObject, eventdata, handles)
%
handles = guidata(hObject);
delete(handles.handlesLines);
delete(hObject);

% --- Executes on mouse motion over figure - except title and menu.
function figure1_WindowButtonMotionFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


hMain = handles.mainFigure;
hAxis = get(hMain, 'CurrentAxes');
C = get (hAxis, 'CurrentPoint');
if isfield(handles, 'hImage')
    him=get(hMain, 'UserData');
    if ~ishandle(him), return; end
%    if ~isfield(him, 'CDATA'), return; end
    data = get(him,'CDATA');
    [r,c] = size(data);
    indX = round(C(1,1));
    indY = round(C(1,2));
    if indX>0 && indX<=c && indY>0 && indY<=r
%         xText = num2str(round((C(1,1)+handles.X1-1)*10)/10);
%         yText = num2str(round((handles.Y1-C(1,2)+1)*10)/10);
        xText = num2str(round(C(1,1)+handles.X1-1));
        yText = num2str(round(handles.Y1-C(1,2)+1));
        hText = num2str(round(data(indY,indX)*100)/100);
        if strcmp(get(handles.UuLiimenis, 'Visible'),'on')
```

```
            hText = num2str(round(handles.modelPluudu(indY,indX)*100)/100);
        end
        set(handles.mainFigure, 'Name', ['Modelis: ' handles.modelName ' X=' xText ' Y='
yText ' h=' hText 'm'])
        str = {[' X=' xText ' '],[' Y=' yText ' '], [' h=' hText 'm ']};
        set(handles.posText, 'String', str);
%         ext = get(handles.posText, 'Extent');
%         ext(1) = ext(1)+0.5;
%         ext(2) = ext(2)-0.5;
        set(handles.posText, 'Position', C(1,1:2));
%         set(handles.posText, 'Extent', ext);
    end
end


% --- Executes on button press in pushbutton1. Staigāt
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.walkActive = 1;
guidata(hObject, handles);
currPos = handles.walkPos;
[lastPos,c]=size(handles.posSteps);
while handles.walkActive
    currPos = currPos+handles.walkSpeed;
    if currPos>lastPos, currPos=1; end
    xi=posSteps(currPos,1);
    yi=posSteps(currPos,2);
    sizeXY = handles.walkWindowSize/2;
    set(gca,'xlim',[xi-sizeXY xi+sizeXY-1])
    set(gca,'ylim',[yi-sizeXY yi+sizeXY-1])
    handles=guidata(hObject);
end
handles.walkPos = currPos;
guidata(hObject, handles);


% --- Executes on button press in pushbutton2. Apstāties
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.walkActive = 0;
guidata(hObject, handles);

% Walk logs - Staigāt-------------------------------------------------------------------
function uipushtool1_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.walkActive = 1;
hMain=hObject;
guidata(hObject, handles);
currPos = handles.walkPos;
[lastPos,c]=size(handles.posSteps);

if isempty(handles.walkWindow) || ~any(ishandle(handles.walkWindow))
%     figure('doublebuffer','on');

    currX=round(handles.posSteps(currPos,1));
    currY=round(handles.posSteps(currPos,2));
    sizeXY = handles.walkWindowSize/2;
    minX = currX-sizeXY;
    maxX = currX+sizeXY-1;
    minY = currY-sizeXY;
    maxY = currY+sizeXY-1;
    pos = [minX,maxY; maxX,maxY; maxX,minY; minX,minY];
```

```
    if isempty(handles.walkWindowInMain) || ~any(ishandle(handles.walkWindowInMain))
        handles.walkWindowInMain = impoly(handles.hImage.Parent, pos);
    end
    hsf=figure('numbertitle', 'off', 'menubar','none','toolbar','none','name','Ādaži-
Gauja', 'CloseRequestFcn',{@my_closereq,handles});

    handles.walkWindow = imshow(handles.model(currY-sizeXY:currY+sizeXY-1,currX-
sizeXY:currX+sizeXY-1),[-5 20], 'colormap', colormap(jet));
end
set(handles.uipushtool1, 'Enable', 'off')
set(handles.uipushtool2, 'Enable', 'on')
guidata(hMain, handles);
while any(ishandle(handles.walkWindow)) && handles.walkActive
    currPos = currPos+handles.walkSpeed;
    if currPos>lastPos, currPos=1; end
    currX=round(handles.posSteps(currPos,1));
    currY=round(handles.posSteps(currPos,2));

    sizeXY = handles.walkWindowSize/2;
    minX = currX-sizeXY;
    maxX = currX+sizeXY-1;
    minY = currY-sizeXY;
    maxY = currY+sizeXY-1;
    pos = [minX,maxY; maxX,maxY; maxX,minY; minX,minY];
    setPosition(handles.walkWindowInMain, pos);
    set(handles.walkWindow.Parent,'clim',get(handles.hImage.Parent, 'clim'));
    set(handles.walkWindow, 'CDATA', handles.model(currY-sizeXY:currY+sizeXY-1,currX-
sizeXY:currX+sizeXY-1));
    drawnow
    handles=guidata(hMain);
end
handles.walkPos = currPos;

guidata(hMain, handles);

% Walk logs - Apstāties-------------------------------------------------------------
--
function uipushtool2_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to uipushtool2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.walkActive = 0;

set(handles.uipushtool1, 'Enable', 'on')
set(handles.uipushtool2, 'Enable', 'off')

guidata(hObject, handles);

%Aizvērt Walk un nodzēst impoly---------------------------------------
function my_closereq(src,callbackdata, handles)
%function my_closereq(src,callbackdata, hPoly)
% Close request function
% to display a question dialog box
try
delete(handles.walkWindowInMain);
%delete(hPoly)
catch
end
delete(gcf)

set(handles.uipushtool1, 'Enable', 'on')
set(handles.uipushtool2, 'Enable', 'off')

guidata(handles.output, handles);

% ---------------------------------------------------------------------
function Contrast_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Contrast (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

handles2 = handles;
if isempty(handles.hContrast)
    handles2.hContrast = imcontrast(gca);
%     handles2.contrastTool = handles.Contrast;
    set(handles2.hContrast, 'CloseRequestFcn',{@closeContrast,handles});
else
    delete(handles.hContrast);
    handles2.hContrast = [];
end
guidata(hObject, handles2);

%Aizvērt Contrast -----------------------------------
function closeContrast(src, callbackdata, handles)
% Close request function
% to display a question dialog box
set(handles.Contrast, 'state', 'off');
handles.hContrast = [];
delete(gcf)
guidata(handles.output, handles);


% --------------------------------------------------------------------
function Ezers_uipushtool5_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Ezers_uipushtool5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structuEzersMenure with handles and user data (see GUIDATA)

hh=helpdlg('Atzīmēt punktu ezerā, kuram nav augstuma datu (h=0)', 'Ezera vidus');
uiwait(hh);
hStartPoint = impoint(get(handles.hImage, 'parent'));
pos = round(getPosition(hStartPoint));
heightStart = handles.model(pos(2),pos(1));
if ~heightStart, % neko nedara, ja atzīmētā punkta augstums nav =0
    BW1 = imfill(handles.BW0, [pos(2) pos(1)], 4);
    Ezers = BW1 & ~handles.BW0;
    model2 = handles.model + single(Ezers)*max(handles.model(:));
    set(handles.hImage, 'CDATA' , model2);
    drawnow
    button = questdlg('Vai ezers izdalīts pareizi?','Ezera robežas','Jā', 'Nē', 'Nē');

    if strcmp(button, 'Jā')
        expandedEzers = imdilate(Ezers, strel([0 1 0; 1 1 1; 0 1 0]));
        malas = expandedEzers & ~Ezers;
        maluAugstumi = handles.model(find(malas));

        %hh = single(mode(maluAugstumi(:)));
        hh = single(percentile(maluAugstumi, 5));

        handles.model = handles.model + single(Ezers)*hh;
        handles.BW1 = handles.BW0;
        handles.BW0 = BW1;
        guidata(hObject, handles);
        set(handles.hImage, 'CDATA' , handles.model);
        drawnow
    else
        set(handles.hImage, 'CDATA' , handles.model);
        drawnow
    end
else
    errordlg('Atzīmētā punkta augstums nav =0', 'Ezers', 'modal');

end
delete(hStartPoint);
```

```
% --------------------------------------------------------------------
function UErobeza_uipushtool6_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to UErobeza_uipushtool6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

hh=helpdlg('Zīmējiet (velkot ar peli) līniju, pa kuru interpolēt augstuma datus', 'Upes
robeža');
uiwait(hh);
hLine = imline(get(handles.hImage, 'parent'));
if isempty(hLine), return; end
pos = round(getPosition(hLine));
heightStart = handles.model(pos(1,2),pos(1,1));
heightEnd = handles.model(pos(2,2),pos(2,1));
model2 = handles.model;
BW2 = handles.BW0;
if heightStart && heightEnd, % neko nedara, ja kāda atzīmētā punkta augstums =0
    x1 = pos(1,1); x2 = pos(2,1); dx = x2-x1;
    y1 = pos(1,2); y2 = pos(2,2); dy = y2-y1;
    steps = sqrt(dx*dx + dy*dy);
    ddx = dx/steps;
    ddy = dy/steps;
    ddh = (heightEnd - heightStart)/steps;
    for istep = 1:steps
        x = round(x1+istep*ddx);
        y = round(y1+istep*ddy);
        if model2(y,x)>0, continue; end
        h = heightStart + istep*ddh;
        model2(y,x) = h;
        BW2(y,x) = true;
    end
    delete(hLine);
    set(handles.hImage, 'CDATA' , model2);
    drawnow
    button = questdlg('Vai līnija iezīmēta pareizi?','Upes robeža','Jā', 'Nē', 'Nē');

    if strcmp(button, 'Jā')
        handles.model = model2;
        handles.BW0 = BW2;
        guidata(hObject, handles);
    else
        set(handles.hImage, 'CDATA' , handles.model);
        drawnow
    end
else
    errordlg('Sākuma vai beigu punkta augstums ==0', 'Upes robeža', 'modal');

end
delete(hLine);


% --------------------------------------------------------------------
function Upe_uipushtool7_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Upe_uipushtool7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


hh=helpdlg({'Atzīmēt upes ~vidus punktus pret straumi,',...
    'kuriem nav augstuma datu (h=0).' ,...
    'Pēdējo punktu atzīmēt ar dubultklikšķi.'}, 'Upes atzīmēšana');
uiwait(hh);
hPoints = impoly(get(handles.hImage, 'parent'), 'Closed', false);
if isempty(hPoints), return; end
pos = round(getPosition(hPoints));

% oldpointer = get(handles.mainFigure, 'pointer');
% set(handles.mainFigure, 'pointer', 'watch')
```

```
% drawnow;

try
    handles2 = fillUpe2(handles, pos);
    if ~isempty(handles2),
        set(handles2.hImage, 'CDATA' , handles2.model);
        guidata(hObject, handles2);
        drawnow
    end
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% set(handles.mainFigure, 'pointer', oldpointer)
drawnow;
delete(hPoints);

% --------------------------------------------------------------------
function UpeFragment_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to UpeFragment (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if ~isfield(handles, 'Upes') || length(handles.Upes)==0,
    errordlg('Ūdenstilpnes nav definētas!', 'Kļūda');
    return;
end

hh=helpdlg('Atzīmēt punktu upē, kuras fragmentu gribat izvēlēties', 'Upes fragmenta
izvēle');
uiwait(hh);
hStartPoint = impoint(get(handles.hImage, 'parent'));
pos = round(getPosition(hStartPoint));

iUpes = findUpe(handles, pos); % atrast upes struktūru no punkta
if isempty(iUpes),
    errordlg('Punkts nepieder ūdenstilpnei!', 'Kļūda');
    delete(hStartPoint);
    return;
end
delete(hStartPoint);
Upe = handles.Upes(iUpes(1));
set(handles.hImage, 'CDATA' , handles.model.*~Upe.Pixels);
drawnow

hh=helpdlg({'Uzzīmēt daudzstūri, kura iekšienē ir vēlamais upes fragments.',...
    'Pēdējo punktu atzīmēt ar dubultklikšķi, to ar sākumu savienos automātiski.'},...
    'Upes fragmenta izvēle');
uiwait(hh);
hPoly = impoly(get(handles.hImage, 'parent'));
posPoly = round(getPosition(hPoly));
[rows, cols] = size(Upe.Pixels);
BW = poly2mask(posPoly(:,1), posPoly(:,2), rows, cols);
UpesFragments = Upe.Pixels & BW;

set(handles.hImage, 'CDATA' , handles.model.*~UpesFragments);
drawnow

options.Interpreter = 'tex';
answer = inputdlg('Upes fragment nosaukums:                                    \color{white}
.','Upe',1, {'Gauja n'}, options);
if ~isempty(answer),
    Upes = handles.Upes;
    nUpes = length(Upes);
    Upes(nUpes+1).Name = answer{1};
    Upes(nUpes+1).Pixels = UpesFragments;
    handles.Upes = Upes;
    handles.FragmentMask = BW;
```

```
    guidata(hObject, handles);
end
delete(hPoly)
set(handles.hImage, 'CDATA' , handles.model);
drawnow


% --------------------------------------------------------------------
function Caurumi_uipushtool8_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Caurumi_uipushtool8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

BW = ~handles.BW0; %caurumi
if length(find(BW==true))==0, return; end
BWc = false(size(BW));
CC = bwconncomp(BW,4);
model1 = handles.model;
strel4 = strel([0 1 0; 1 1 1; 0 1 0]);
[rows, cols] = size(model1);
npixels = rows*cols;
nZeroFragm = length(CC.PixelIdxList);

hw = waitbar(0,'Lūdzu gaidiet...','Name','Piepildām "caurumus"...');
for ic=1:nZeroFragm
    %ic
    try
        if ~ishandle(hw),
            hw = waitbar(0,'Lūdzu gaidiet...','Name','Piepildām "caurumus"...');
        end
        waitbar(ic/nZeroFragm, hw);
        indices = CC.PixelIdxList{ic};
        rowInd = mod(indices-1, rows)+1;
        colInd = floor((indices-1)/rows)+1;
        row1 = max(1, min(rowInd)-1);
        rowN = min(rows, max(rowInd)+1);
        col1 = max(1, min(colInd)-1);
        colN = min(cols, max(colInd)+1);
        BWfragm = false(rowN-row1+1, colN-col1+1);
        [rowsF, colsF] = size(BWfragm);
        indicesF = (colInd-col1)*rowsF + rowInd-row1+1;
        BWfragm(indicesF) = true;
        %BWc(indices) = true;
        %BWe = imdilate(BWc, strel4);
        BWe = imdilate(BWfragm, strel4);
        malas = BWe & ~BWfragm;
        [malasRowIndF, malasColIndF] = find(malas);
        malasInd = (malasColIndF+col1-2)*rows + malasRowIndF+row1-1;
        malasInd(malasInd>npixels)=[];
        maluAugstumi = model1(malasInd);
%        hh = single(mode(maluAugstumi(:)));
        hh = single(percentile(maluAugstumi, 5));
        model1(row1:rowN, col1:colN) = model1(row1:rowN, col1:colN) + single(BWfragm)*hh;
        %BWc(indices) = false;
    catch ex
        errordlg(getExceptionStr(ex), 'Kļūda');
        logException(exception);
    end
end

close(hw)

%model2 = imfill(handles.model);
handles.model = handles.model + single(BW).*model1;
handles.BW1 = handles.BW0;
handles.BW0 = true(size(handles.BW0));
guidata(hObject, handles);
set(handles.hImage, 'CDATA' , handles.model);
```

```
drawnow


% --------------------------------------------------------------------
function Undo_uipushtool9_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to Undo_uipushtool9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try

    handles.model = handles.model .* single(handles.BW1);
    handles.BW0 = handles.BW1;
    guidata(hObject, handles);
    set(handles.hImage, 'CDATA' , handles.model);
    drawnow

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end
% --------------------------------------------------------------------
function Colormap_OffCallback(hObject, eventdata, handles)
% hObject    handle to Colormap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

try
    cmap = single(colormap(jet(1024)));
    handles.colormap = cmap;
    colormap(gcf,cmap);
    guidata(hObject, handles);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end
% --------------------------------------------------------------------
function Colormap_OnCallback(hObject, eventdata, handles)
% hObject    handle to Colormap (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

try
    cmap = single(colormap(jet(1024)));
    cmap(1,:)=1.;
    handles.colormap = cmap;
    colormap(gcf,cmap);
    guidata(hObject, handles);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

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

try
    %ļauj uzzīmēt taisnstūri - detalizētās apskates attēla robežas
    hfrag = imrect;
    rect = getPosition(hfrag);
    if rect(3)<3 || rect(4)<3, delete(hfrag); return; end

    %neļauj sākumā uzzīmēt rajonu ārpus attēla
    xlim = get(gca,'XLim') + [1 -1];
    ylim = get(gca,'YLim') + [1 -1];
    rect(1) = max([rect(1) xlim(1)]);
```

```
    rect(3) = min([rect(3) xlim(2)-rect(1)]);
    rect(2) = max([rect(2) ylim(1)]);
    rect(4) = min([rect(4) ylim(2)-rect(2)]);

    %nosaka fragmenta indeksus
    x1 = round(rect(1));
    y1 = round(rect(2));
    xn = x1+round(rect(3))-1;
    yn = y1 + round(rect(4))-1;

    [fileName, pathName] = uiputfile('*.hedi','Izvēlieties saglabājamo projekta failu');
    if fileName==0, delete(hfrag); return; end

    fileF = [pathName fileName];
    if isempty(strfind(fileName, '.hedi')), fileF = [fileF '.hedi']; end

    delete(hfrag);

    hf = gcf;
    h = waitbar(0.5,'Lūdzu, pagaidiet...','Name','Saglabājam projektu...');

    model = handles.model;
    % hAxis = get(handles.hImage, 'Parent');
    % XLim = get(hAxis, 'XLim');
    % YLim = get(hAxis, 'YLim');
    % x1 = ceil(XLim(1)); xn = floor(XLim(2));
    % y1 = ceil(YLim(1)); yn = floor(YLim(2));
    model = model(y1:yn,x1:xn);
    X1 = handles.X1 + handles.dX * (x1-1);
    Y1 = handles.Y1 - handles.dY * (y1-1);
    dX = handles.dX;
    dY = handles.dY;
    Upes = handles.Upes;
    for iUpe = length(Upes):-1:1
        pix = Upes(iUpe).Pixels;
        pix = pix(y1:yn,x1:xn);
        if any(pix(:))
            Upes(iUpe).Pixels = pix;
        else
            Upes(iUpe) = [];
        end
    end
    Prognozes = handles.Prognozes;
    for iPrognoze = length(Prognozes):-1:1
        pix = Prognozes(iPrognoze).Pixels;
        pix = pix(y1:yn,x1:xn);
        if any(pix(:))
            Prognozes(iPrognoze).Pixels = pix;
            heights = Prognozes(iPrognoze).Heights;
            heights = heights(y1:yn,x1:xn);
            Prognozes(iPrognoze).Heights = heights;
        else
            Prognozes(iPrognoze) = [];
        end
    end


    save(fileF, 'model', 'X1', 'Y1', 'dX', 'dY', 'Upes', 'Prognozes',  '-v7.3');

    close(h)

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

%%
function UuLiimenisStr_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to UuLiimenisStr (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 UuLiimenisStr as text
%        str2double(get(hObject,'String')) returns contents of UuLiimenisStr as a double
hh = str2double(get(hObject,'String'));
hSlider = get(handles.UuLiimenis, 'Value');
if hh<hSlider,
    set(hObject,'Value',hSlider);
    return;
end
set(handles.UuLiimenis, 'Value', hh);
%handles.WaterLevel = hh;
%guidata(hObject, handles);
UuLiimenis_Callback(handles.UuLiimenis, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function UuLiimenisStr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to UuLiimenisStr (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

% --- Executes on slider movement.
function UuLiimenis_Callback(hObject, eventdata, handles)
% hObject    handle to UuLiimenis (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
height = get(hObject,'Value');
if (height<handles.WaterLevel),
    set(hObject,'Value',handles.WaterLevel);
    return;
end
delta = height - handles.WaterLevel;
set(handles.UuLiimenisStr, 'String', num2str(height));

if isfield(handles, 'modelPluudu'),
    modelPluudu = handles.modelPluudu;
else
    modelPluudu = handles.model;
end
if handles.selectedPrognoze>0
    measuredPos = handles.FloodPos;
    Prognoze = handles.Prognozes(handles.selectedPrognoze);
    PrognozeHeights = Prognoze.Heights;
    PrognozePixels = Prognoze.Pixels;
    PrognozeHI = PrognozeHeights(measuredPos(2),measuredPos(1));
    modelPluuduHeight = modelPluudu(measuredPos(2),measuredPos(1));
    propDelta = (height - handles.WaterLevel)/(PrognozeHI-handles.WaterLevel);
    modelPluuduD = modelPluudu + delta;
    modelPluuduD = modelPluuduD.*single(1-PrognozePixels) +
(modelPluudu+propDelta*(PrognozeHeights-modelPluudu)).*single(PrognozePixels);
else
    modelPluuduD = modelPluudu + delta;
end
```

```
UpePaplasinas = true;
%attēlo bildē, kā ūdens pacēlies
UpePluudu = handles.UpePluudu;
UpePluuduPix = UpePluudu.Pixels;
modelPluudu = modelPluudu .* single(~UpePluuduPix) + modelPluuduD .*
single(UpePluuduPix);

handles.floodActive = 1;
guidata(hObject, handles);

set(handles.StopFlooding, 'Enable', 'on')
set(handles.ResumeFlooding, 'Enable', 'off')
set(handles.UdensCelsanas3, 'Enable', 'off')

h = waitbar(0.5,'Lūdzu, vērojiet...','Name','Imitējam ūdens līmeņa celšanos...');
v = VideoWriter('Gauja.avi');
% [rows,cols] = size(UpePluuduPix);
% v.Height = rows;
% v.Width = cols;
open(v);
hImage = handles.hImage;
if isfield(handles,'FragmentMask')
    fragmMask = handles.FragmentMask;
else
    fragmMask = true(size(UpePluuduPix));
end
sqstrel3 = strel('square',3);
% parfor nevar izmantot, jo nākošais rezultāts atkarīgs no iepriekšējā

i=0;
while UpePaplasinas
        UpePluuduPix = UpePluuduPix & fragmMask;
        expUpe = imdilate(UpePluuduPix,sqstrel3 );
        malas = expUpe & ~UpePluuduPix;
        hMalas = modelPluudu .* single(malas);
        hUpe = modelPluudu .* single(UpePluuduPix);
        hMalasNoUpes = imdilate(hUpe, sqstrel3) .* single(malas);
        changes = hMalasNoUpes > hMalas;
        hChanges = hMalasNoUpes .*single(changes);
        if any(changes(:)),
            UpePluuduPix = UpePluuduPix | changes;
            hUpeNew = hUpe + hChanges;
            modelPluudu = modelPluudu.*single(~UpePluuduPix) + hUpeNew;

            if mod(i,5)==0, set(hImage, 'CDATA' , modelPluudu.*single(~UpePluuduPix));
end
            drawnow

           frame = getframe(handles.mainFigure);
           writeVideo(v,frame);

            handles2 = guidata(hObject);
            if ~handles2.floodActive, break; end
            if ~ishandle(h),
                h = waitbar(0.5,'Lūdzu, vērojiet...','Name','Imitējam ūdens līmeņa
celšanos...');
            end
            set(h, 'Name', ['Applūst: ' num2str(length(find(changes==true))) ' jauni
ūdens pikseļi'])
            i=i+1;
        else
            set(hImage, 'CDATA' , modelPluudu.*single(~UpePluuduPix));
            UpePaplasinas = false;
        end
end
```

```
close(v)
delete(h)
disp 'Video saglabāts failā "Gauja.avi"'
handles.modelPluudu = modelPluudu;
UpePluudu.Pixels = UpePluuduPix;
handles.UpePluudu = UpePluudu;
handles.floodActive = 0;
handles.WaterLevel = height;
guidata(hObject, handles);

% ---
function [r1,rn,c1,cn] = indNonZero(A)
[ru,cu]= size(A);

colsums = sum(A);
rowsums = sum(A,2);
r1 = max(1,find(rowsums>0, 1)-5);
rn = min(ru,find(rowsums>0, 1, 'last' )+5);
c1 = max(1,find(colsums>0, 1)-5);
cn = min(ru,find(colsums>0, 1, 'last' )+5);

% --- Executes during object creation, after setting all properties.
function UuLiimenis_CreateFcn(hObject, eventdata, handles)
% hObject    handle to UuLiimenis (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

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


% --------------------------------------------------------------------
function UdensCelsanas3_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to UdensCelsanas3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function UdensCelsanas3_OffCallback(hObject, eventdata, handles)
% hObject    handle to UdensCelsanas3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

set(handles.UuLiimenis, 'Visible', 'off');
set(handles.UuLiimenisStr, 'Visible', 'off');
set(handles.UseForecastModel, 'Visible', 'off');
set(handles.UseForecastModel, 'Value', 0);
handles2 = handles;
handles2.selectedPrognoze = 0;

handles.modelPluudu = handles.model;
set(handles.hImage, 'CDATA' , handles.model);
drawnow
set(handles.StopFlooding, 'Enable', 'off')
set(handles.ResumeFlooding, 'Enable', 'off')
guidata(hObject, handles2);

% --------------------------------------------------------------------
function UdensCelsanas3_OnCallback(hObject, eventdata, handles)
% hObject    handle to UdensCelsanas3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
if ~isfield(handles, 'Upes') || length(handles.Upes)==0,
    errordlg('Ūdenstilpnes nav definētas!', 'Kļūda');
    set(handles.UdensCelsanas3, 'State', 'off');
    return;
end

hh=helpdlg('Atzīmēt punktu ūdenstilpnē, kuras līmeņa celšanos imitēt', 'Ūdens līmeņa
celšanās vieta');
uiwait(hh);
hStartPoint = impoint(get(handles.hImage, 'parent'));
pos = round(getPosition(hStartPoint));

iUpes = findUpe(handles, pos); % atrast upes struktūru no punkta
if isempty(iUpes),
    errordlg('Punkts nepieder ūdenstilpnei!', 'Kļūda');
    delete(hStartPoint);
    return;
end
delete(hStartPoint);
Upe = handles.Upes(iUpes(1));
nUpes = length(iUpes);
if nUpes>1
    UpeNames = cell(1,nUpes);
    for iUpe = 1:nUpes
        UpeNames{iUpe} = handles.Upes(iUpes(iUpe)).Name;
    end
    [Selection, ok] = listdlg('Name', 'Upes izvēle', 'PromptString', 'Izvēlēties upi vai
fragmentu, kura līmeņa celšanos imitēt',...
            'ListString', UpeNames, 'ListSize', [300 200],...
            'OKString', 'Labi', 'CancelString', 'Atcelt', 'SelectionMode', 'single');
    if ok==0, return; end
    Upe = handles.Upes(iUpes(Selection(1)));
    %disp (Upe.Name)
end
set(handles.hImage, 'CDATA' , handles.model.*~Upe.Pixels);
set(handles.WhiteLowHeight,'Checked','on');
cmap = single(colormap(jet(1024)));
cmap(1,:)=1.;
handles.colormap = cmap;
colormap(gcf,cmap);
drawnow

heightStart = handles.model(pos(2),pos(1));

% atvērt slider-i u.c.
set(handles.UuLiimenis, 'Value', heightStart);
set(handles.UuLiimenisStr, 'String', num2str(heightStart));
set(handles.UuLiimenis, 'Visible', 'on');
set(handles.UuLiimenisStr, 'Visible', 'on');
set(handles.UseForecastModel, 'Visible', 'on');

handles.WaterLevel = heightStart;
handles.FloodPos = pos;
handles.UpePluudu = Upe;
handles.modelPluudu = handles.model;
guidata(hObject, handles);
%set(handles.StopFlooding, 'Enable', 'on')



% --------------------------------------------------------------------
function ResumeFlooding_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to ResumeFlooding (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if ~isfield(handles, 'Upes') || length(handles.Upes)==0,
    errordlg('Ūdenstilpnes nav definētas!', 'Kļūda');
```

```
    set(handles.UdensCelsanas3, 'State', 'off');
    return;
end
set(handles.StopFlooding, 'Enable', 'on')
set(handles.ResumeFlooding, 'Enable', 'off')
set(handles.UdensCelsanas3, 'Enable', 'off')
UuLiimenis_Callback(handles.UuLiimenis, eventdata, handles)

% --------------------------------------------------------------------
function StopFlooding_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to StopFlooding (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if ~isfield(handles, 'Upes') || length(handles.Upes)==0,
    errordlg('Ūdenstilpnes nav definētas!', 'Kļūda');
    set(handles.UdensCelsanas3, 'State', 'off');
    return;
end

handles.floodActive = 0;
guidata(hObject, handles);
set(handles.ResumeFlooding, 'Enable', 'on')
set(handles.StopFlooding, 'Enable', 'off')
set(handles.UdensCelsanas3, 'Enable', 'on')


% --------------------------------------------------------------------
function AugstumaProfils_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to AugstumaProfils (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
hh=helpdlg({'Atzīmēt lauztu līniju, kuras augstuma profilu gribat analizēt.',...
    'Pēdējo punktu atzīmēt ar dubultklikšķi.'}, 'Augstuma profils');
uiwait(hh);
hPoly = impoly('Closed', false);
%get(handles.hImage, 'parent'), 'Closed', false); ????

if isempty(hPoly), return; end
pos = round(getPosition(hPoly));

try
    hLinesPoly = [handles.hLinesPoly hPoly];
    handles.hLinesPoly = hLinesPoly;
    handles2 = drawProfile(handles, pos, hPoly);
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% zīmē augstuma profilu līnijai atsevišķā logā
function handlesOut = drawProfile(handlesIn, pos, hPoly)
%drawProfile Līnijas punktu augstumu aprēķins un vizualizācija
% Ieejā:
%   handlesIn.model - augstuma modelis ieejā
%   pos - atzīmētie lauztas līnijas punkti
% Izejā:
%   handlesOut.model - modelis ar ????

handlesOut = handlesIn;
model = handlesIn.model;

[heights, distances] = getProfile(pos, model);

handlesLines = handlesIn.handlesLines;
nLine = 0;
if ~isempty(handlesLines),
    for il = 1:length(handlesLines)
```

```
        nLine = max(nLine,get(handlesLines(il), 'UserData'));
    end
    %nLine = get(handlesLines(length(handlesLines)), 'UserData') + 1;
end
hL = figure('NumberTitle','off', 'Name', ['Augstuma profils ' num2str(nLine+1)],
'UserData', nLine+1, 'Interruptible', 'off');
handlesLines = [handlesLines hL];
handlesOut.handlesLines = handlesLines;
set(hL, 'CloseRequestFcn',  {@AugstumaProfils_ClosingFcn, handlesOut.mainFigure});
addNewPositionCallback(hPoly, @(varargin)Poly_NewPositionCallbackFcn(hPoly, hL));

plot(distances, heights);
set(gca,'Ylim',[0 max(model(:))]);
ylabel ('augstums, m')
xlabel ('attālums no sākuma, m')
grid on

%
function [heights, distances] = getProfile(pos, model, pixSize)
if nargin<3, pixSize=1; end     % noklusētais pikseļa izmērs ir 1m
gabali = size(pos,1)-1;
x = round(pos(1,1));
y = round(pos(1,2));
heights = model(y,x);
distances = 0;
for ig = 1:gabali
    x1 = pos(ig,1); x2 = pos(ig+1,1); dx = x2-x1;
    y1 = pos(ig,2); y2 = pos(ig+1,2); dy = y2-y1;
    distPix = sqrt(dx*dx + dy*dy);
    steps = round (distPix);
    distM = distPix * pixSize;
    ddx = dx/steps;
    ddy = dy/steps;
    dist1 = distances(length(distances));
    distances = [distances dist1+(1:steps)/steps*distM * pixSize];
    for istep = 1:steps
        x = round(x1+istep*ddx);
        y = round(y1+istep*ddy);
        heights = [heights model(y,x)];
    end
end
%heights = medfilt2(heights, [1 9]);


% ja aizver augstuma profila attēlu, dzēš arī attiecīgo līniju pamatattēlā
function AugstumaProfils_ClosingFcn(hObject, eventdata, handleMain, varargin)

handles = guidata(handleMain);
hLine = gcf;
nLine = get(hLine, 'UserData');
handlesLines = handles.handlesLines;

for ih=1:length(handlesLines)
    nLineI = get(handlesLines(ih), 'UserData');
    if nLineI==nLine;
        handlesLines(ih) = [];
        delete(hLine);
        delete(handles.hLinesPoly(ih));
        handles.hLinesPoly(ih) = [];
        break;
    end
end

handles.handlesLines = handlesLines;
guidata(handleMain, handles);

% ja maina augstuma līniju, izmaina attiecīgo augstuma profila attēlu
function Poly_NewPositionCallbackFcn(hPoly, hLineFigure)
```

```
pos = round(getPosition(hPoly));
hFigMain = gcf;

handles = guidata(gcf);

[heights, distances] = getProfile(pos, handles.model);

figure(hLineFigure)
ylim = get(gca,'Ylim');
plot(distances, heights)
set(gca,'Ylim',ylim);
ylabel ('augstums, m')
xlabel ('attālums no sākuma, m')
grid on
drawnow

figure(hFigMain)

%


% --------------------------------------------------------------------
function AugstumaModelis_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to AugstumaModelis (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
hh=helpdlg({'Atzīmēt apgabalu ar lauztu līniju, kuras augstuma modeli gribat
analizēt.',...
    'Pēdējo punktu atzīmēt ar dubultklikšķi.'}, 'Augstuma modelis');
uiwait(hh);
hPoly = impoly;

if isempty(hPoly), return; end
pos = round(getPosition(hPoly));

try
    hRegionPoly = [handles.hRegionPoly hPoly];
    handles.hRegionPoly = hRegionPoly;
    handles2 = drawPolygonHeight(handles, pos, hPoly);
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% zīmē augstuma profilu līnijai atsevišķā logā
function handlesOut = drawPolygonHeight(handlesIn, pos, hPoly)
%drawProfile Līnijas punktu augstumu aprēķins un vizualizācija
% Ieejā:
%   handlesIn.model - augstuma modelis ieejā
%   pos - atzīmētie lauztas līnijas punkti
% Izejā:
%   handlesOut.model - modelis ar ????

handlesOut = handlesIn;
model = handlesIn.model;

[heights] = getHeightModel(pos, model, hPoly);

handlesRegions = handlesIn.handlesRegions;
nReg = 0;
if ~isempty(handlesRegions),
    for il = 1:length(handlesRegions)
        nReg = max(nReg,get(handlesRegions(il), 'UserData'));
    end
end
```

```
hR = figure('NumberTitle','off', 'Name', ['Augstuma modelis ' num2str(nReg+1)],
'UserData', nReg+1, 'Interruptible', 'off');
handlesRegions = [handlesRegions hR];
handlesOut.handlesRegions = handlesRegions;
set(hR, 'CloseRequestFcn',  {@AugstumaModelis_ClosingFcn, handlesOut.mainFigure});
addNewPositionCallback(hPoly, @(varargin)Reg_NewPositionCallbackFcn(hPoly, hR));

[r,c] = size(heights);
%heights = flipud(heights);
mesh ((1:c)*handlesIn.dX, (1:r)*handlesIn.dY,  heights);
%set(gca,'Zlim',[0 max(model(:))]);
zlabel ('augstums, m')
xlabel ('attālums uz A, m')
ylabel ('attālums uz Z, m')

grid on

%
function [heights] = getHeightModel(pos, model, hPoly)
if nargin<3, pixSize=1; end     % noklusētais pikseļa izmērs ir 1m

xx= pos(:,1); yy = pos(:,2);
xmin = min(xx); xmax = max(xx);
ymin = min(yy); ymax = max(yy);
mask = createMask(hPoly);
heights = model(ymin:ymax, xmin:xmax).*mask(ymin:ymax, xmin:xmax);
heights = flipud(heights);
heights(heights==0)=NaN;


%heights = medfilt2(heights, [1 9]);


% ja aizver augstuma profila attēlu, dzēš arī attiecīgo līniju pamatattēlā
function AugstumaModelis_ClosingFcn(hObject, eventdata, handleMain, varargin)

handles = guidata(handleMain);
hLine = gcf;
nLine = get(hLine, 'UserData');
handlesRegions = handles.handlesRegions;

for ih=1:length(handlesRegions)
    nLineI = get(handlesRegions(ih), 'UserData');
    if nLineI==nLine;
        handlesRegions(ih) = [];
        delete(hLine);
        delete(handles.hRegionPoly(ih));
        handles.hRegionPoly(ih) = [];
        break;
    end
end

handles.handlesRegions = handlesRegions;
guidata(handleMain, handles);

% ja maina augstuma līniju, izmaina attiecīgo augstuma profila attēlu
function Reg_NewPositionCallbackFcn(hPoly, hLineFigure)

pos = round(getPosition(hPoly));
hFigMain = gcf;

handles = guidata(gcf);

[heights] = getHeightModel(pos, handles.model, hPoly);

figure(hLineFigure)
[r,c] = size(heights);
%heights = flipud(heights);
```

```
mesh ((1:c)*handles.dX, (1:r)*handles.dY,  heights);
%set(gca,'Zlim',[0 max(model(:))]);
zlabel ('augstums, m')
xlabel ('attālums uz A, m')
ylabel ('attālums uz Z, m')

grid on

figure(hFigMain)


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

try
    if any(handles.model==0)
        errordlg('Caurteces zīmē tikai pēc tam, kad visos modeļa punktos ir augstuma dati
<>0', 'Caurtece', 'modal');
        return
    end

    handles2 = handles;
    if ~isempty(handles.hCaurtece)
        atbilde = questdlg('Caurtece jau atzīmēta, dzēst?', 'Caurtece', 'Atcelt', 'Labi',
'Labi');
        if strcmp(atbilde, 'Labi')
            delete(handles.hCaurtece);
            handles2.hCaurtece = [];
        else
            return
        end
    end
    hh=helpdlg('Atzīmējiet (velkot ar peli) caurteces līniju', 'Caurtece');
    uiwait(hh);
    hLine = imline(get(handles.hImage, 'parent'));
    if isempty(hLine), return; end
    %pos = round(getPosition(hLine));
    handles2.hCaurtece = hLine;

    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end



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

try

    if any(handles.model==0)
        errordlg('Dambi zīmē tikai pēc tam, kad visos modeļa punktos ir augstuma dati
<>0', 'Dambis', 'modal');
        return
    end
    handles2 = handles;
    if ~isempty(handles.hLineDambis)
        atbilde = questdlg('Dambis jau atzīmēts, dzēst?', 'Dambis', 'Atcelt', 'Labi',
'Labi');
```

```
        if strcmp(atbilde, 'Labi')
            delete(handles.hLineDambis);
            delete(handles.hTextDambis);
            handles2.hLineDambis = [];
            handles2.hTextDambis = [];
        else
            return
        end
    end

    hh=helpdlg({'Atzīmējiet lauztu līniju, kura imitē dambja vietu attēlā.',...
        'Pēdējo punktu atzīmēt ar dubultklikšķi'}, 'Dambis');
    uiwait(hh);
    hPoly = impoly('Closed', false);
    %get(handles.hImage, 'parent'), 'Closed', false); ????

    if isempty(hPoly), return; end
    pos = round(getPosition(hPoly));
    virsotnes = size(pos,1);
    hold on

    handles2.hLineDambis = hPoly;
%    addNewPositionCallback(hPoly, @(varargin)Dambis_NewPositionCallbackFcn(hPoly));
    htext = [];
    for iv = 1:virsotnes
        hv = text(pos(iv,1), pos(iv,2)+4, num2str(iv), 'Color','white', 'UserData',
handles.model(pos(iv,2),pos(iv,1)), 'ButtonDownFcn', {@pointButtonDownFcn,iv});
        htext = [htext hv];
    end
    handles2.hTextDambis = htext;
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% ja maina augstuma līniju, izmaina attiecīgo augstuma profila attēlu
function Dambis_NewPositionCallbackFcn(hPoly, hLineFigure)

pos = round(getPosition(hPoly));
hFigMain = gcf;
handles = guidata(gcf);

[heights, distances] = getProfile(pos, handles.model);

figure(hLineFigure)
ylim = get(gca,'Ylim');
plot(distances, heights)
set(gca,'Ylim',ylim);
ylabel ('augstums, m')
xlabel ('attālums no sākuma, m')
grid on
drawnow

figure(hFigMain)


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

try
    if isempty(handles.hLineDambis), return; end

    [fileName, pathName] = uiputfile('*.matd','Izvēlieties dambja datu failu');
    if fileName==0, return; end
```

```
    fileD = fullfile (pathName, fileName);

    handles2 = handles;
    hLineDambis = handles.hLineDambis;
    pos = getPosition(hLineDambis);
    coords = pos;
    coords(:,1) = (coords(:,1)-1)* handles.dX + handles.X1;
    coords(:,2) = handles.Y1 - (coords(:,2)-1)* handles.dY;

    hTextDambis = handles.hTextDambis;
    augstumi = get(hTextDambis, 'UserData');

    save(fileD, 'augstumi', 'coords');

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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

try
    [fileName, pathName, FilterIndex] = uigetfile('*.matd','Izvēlieties dambja datu
failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    load(fileD, '-mat');
    if ~exist('augstumi','var') || ~exist('coords','var')
        errordlg('Fails nesatur datus par dambi!', 'Kļūda', 'modal');
        return
    end
    virsotnes = numel(augstumi);
    pos = zeros(virsotnes,2);
    pos(:,1) = (coords(:,1) - handles.X1)/ handles.dX + 1;
    pos(:,2) = (handles.Y1 - coords(:,2))/ handles.dY + 1;
    hLineDambis = impoly(gca, pos, 'Closed', false);

    handles2 = handles;
    handles2.hLineDambis = hLineDambis;
    htext = zeros(1,virsotnes);
    for iv = 1:virsotnes
        htext(iv) = text(pos(iv,1), pos(iv,2)+4, num2str(iv), 'Color','white',
'UserData', augstumi{iv}, 'ButtonDownFcn', {@pointButtonDownFcn,iv});
    end
    handles2.hTextDambis = htext;
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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

try
    if isempty(handles.hLineDambis), return; end
```

```
    handles2 = handles;
    hLineDambis = handles.hLineDambis;
    pos = getPosition(hLineDambis);
    coords = pos;
    coords(:,1) = (coords(:,1)-1)* handles.dX + handles.X1;
    coords(:,1) = handles.Y1 - (coords(:,2)-1)* handles.dY;

    hTextDambis = handles.hTextDambis;
    augstumi = get(hTextDambis, 'UserData');
    model2 = handles.model;

    virsotnes = numel(augstumi);
    augstumiM = cell2mat(augstumi);

    % TODO interpolēt augstumu starp galapunktiem
    if virsotnes>2 && augstumi{1}>0 && augstumi{virsotnes}>0 &&
numel(find(augstumiM(2:virsotnes-1)>0))==0
        heightStart = augstumi{1};
        heightEnd = augstumi{virsotnes};
        dh = heightEnd - heightStart;
        garums = 0;
        for jj=2:virsotnes
            garums = garums + CalcDistance(pos(jj,:), pos(jj-1,:));
        end
        d = 0;
        for jj=2:virsotnes
            d = d + CalcDistance(pos(jj,:), pos(jj-1,:));
            h = heightStart + d/garums * dh;
            augstumi{jj} = h;
        end
    end

    for iv = 2:virsotnes
        heightStart = augstumi{iv-1};
        heightEnd = augstumi{iv};
        x1 = pos(iv-1,1); y1 = pos(iv-1,2);
        x2 = pos(iv,1); y2 = pos(iv,2);

        dx = x2-x1; dy = y2-y1;
        steps = sqrt(dx*dx + dy*dy);
        ddx = dx/steps;
        ddy = dy/steps;
        ddh = (heightEnd - heightStart)/steps;
        for istep = 0:steps
            x = round(x1+istep*ddx);
            y = round(y1+istep*ddy);
            h = heightStart + istep*ddh;
            if dx>dy % Ja vairāk horizontāli
                model2(y-1:y+1,x) = h;
            else    % Ja vairāk vertikāli
                model2(y,x-1:x+1) = h;
            end
        end
    end
    set(handles.hImage, 'CDATA' , model2);
    handles2.model = model2;
    delete(handles.hTextDambis);
    delete(handles.hLineDambis);
    handles2.hTextDambis = [];
    handles2.hLineDambis = [];
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


% --------------------------------------------------------------------
```

```
function Exit_Callback(hObject, eventdata, handles)
% hObject    handle to Exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close all hidden


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


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


% --------------------------------------------------------------------
function CopyToClipboard_Callback(hObject, eventdata, handles)
% hObject    handle to CopyToClipboard (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    print(gcf, '-dmeta');
catch exception
    errordlg(getExceptionStr(exception), 'Error');
    logException(exception);
end

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

cmap = single(colormap(jet(1024)));
if strcmp(get(handles.WhiteLowHeight, 'Checked'), 'on')
    set(handles.WhiteLowHeight, 'Checked', 'off');
    handles.colormap = cmap;
    colormap(gcf,cmap);
    guidata(hObject, handles);
else
    set(handles.WhiteLowHeight, 'Checked', 'on');
    cmap(1,:)=1.;
    handles.colormap = cmap;
    colormap(gcf,cmap);
    guidata(hObject, handles);
end


% --------------------------------------------------------------------
function White0_ClickedCallback(hObject, eventdata, handles)
% hObject    handle to White0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

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


% --------------------------------------------------------------------
```

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

try
    handles2 = handles;
    delete(handles.hLineDambis);
    handles2.hLineDambis = [];
    delete(handles.hTextDambis);
    handles2.hTextDambis = [];
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


% --------------------------------------------------------------------
function UserGuide_Callback(hObject, eventdata, handles)
% hObject    handle to UserGuide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
system(which('AdazuPludi100.htm'));

% --------------------------------------------------------------------
function About_Callback(hObject, eventdata, handles)
% hObject    handle to About (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    A = imread(which('Adazi100about.png'));
    h = figure('NumberTitle','off','Name','Par Ādažu
Plūdiem','WindowStyle','modal');%'MenuBar','None','Toolbar','None','Resize','off');
    hh = imshow(A,'InitialMagnification',100);
    set(hh,'ButtonDownFcn','close(gcf);');
catch exception
    errordlg(getExceptionStr(exception), 'Error');
    logException(exception);
end


% --------------------------------------------------------------------
function uitoggletool4_OffCallback(hObject, eventdata, handles)
% hObject    handle to uitoggletool4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%colorbar('off')

% --------------------------------------------------------------------
function uitoggletool4_OnCallback(hObject, eventdata, handles)
% hObject    handle to uitoggletool4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%colorbar('eastoutside')


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


% --------------------------------------------------------------------
function LoadModelMenu_Callback(hObject, eventdata, handles)
% hObject    handle to LoadModelMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

[fileName, pathName] = uigetfile('*.hedi','Izvēlieties atveramā projekta failu');
if fileName==0, return; end

h = [];
try
    fileF = fullfile(pathName, fileName);

    S = whos('-file',fileF);
    nvarS = length(S);
    irX = false; irY = false; irModel = false;
    if nvarS>2
        for iv=1:nvarS
            name = S(iv).name;
            if strcmp(name,'model'), irModel = true; end
            if strcmp(name,'X1'), irX = true; end
            if strcmp(name,'Y1'), irY = true; end
        end
    end

    if nvarS<3 || ~irModel || ~irX || ~irY,
        errordlg('Tas nav FloodEDI projekta fails!', 'Projekta atvēršanas kļūda');
        uiwait
        return
    end

    h = waitbar(0.5,'Lūdzu, pagaidiet...','Name','Atveram projektu...');
    handles.model = [];
    load (fileF,'-mat')
    cd (pathName)

    hContrast = []; %imcontrast(gca);

    handles.hContrast = hContrast;
    handles.X1 = X1;
    handles.Y1 = Y1;
    if exist('dX', 'var')==1,
        handles.dX = dX;
    else
        handles.dX = 1.;
    end
    if exist('dY', 'var')==1,
        handles.dY = dY;
    else
        handles.dY = 1.;
    end
    handles.model = model;
    BW = model>0;
    clear model;
    handles.BW0 = BW;    % caurumi pašreizējā variantā (0-ļļu piepildīšanai)
    handles.BW1 = BW;    % caurumi iepriekšējā variantā (priekš Undo)

    if exist('Upes','var'),
        handles.Upes = Upes;
    else
        handles.Upes = struct('Name',{},'Pixels',{});
    end
    if exist('Prognozes','var'),
        handles.Prognozes = Prognozes;
    else
        handles.Prognozes = struct('Name',{},'UpeName',{},'Pixels',{},'Heights',{});
    end

    hfig = handles.mainFigure;
    set(handles.Caurtece, 'HandleVisibility', 'off');
    figure(hfig);
    %clf
```

```
    %cmap = colormap(hfig);
    cmap = single(colormap(jet(1024)));
    if any(handles.model==0)
        cmap(1,:)=1.;
        set(handles.WhiteLowHeight, 'Checked', 'on');
    else
        set(handles.WhiteLowHeight, 'Checked', 'off');
    end
    handles.colormap = cmap;

        hmenus = findall(hfig, 'Type', 'uimenu');
        set(hmenus, 'HandleVisibility', 'off'); % menus invisible to retain them
        %set(handles.main, 'HandleVisibility', 'off'); % toolbar invisible to retain it
    colorbar('off');
    set(handles.Colorbar, 'State', 'off');
    cla
    hold off
    him = imshow(handles.model, [0 20], 'colormap', cmap);
    %set(handles.uitoolbar, 'Visible', 'on');
    hold on
    str = {[' X=' num2str(X1)],[' Y=' num2str(Y1)], [' h=' num2str(handles.model(1,1))]};
    handles.posText = text(0,0,str,'Position',[0
0],'VerticalAlignment','bottom','BackgroundColor',[1 1 223/255], 'Margin',0.1,
'Visible','off','FontSize',9);
    set(handles.DataTip, 'State', 'off');
    try
        load (which('AdaziNovads.mat'));
        hBorder = plot((Sn.X-X1)/dX+1, (Y1-Sn.Y)/dY+1, 'w-', 'LineWidth', 2);
        handles.hBorder = hBorder;
    catch
    end

    try
        %set(data.main, 'HandleVisibility', 'on');
        set(hmenus, 'HandleVisibility', 'on');
    catch
    end

    [pathstr,name,ext] = fileparts(fileF);
    set(gcf, 'UserData', him, 'Name', ['Modelis: ' name]);
    handles.hImage = him;
    handles.modelName = name;

    % set(handles.uipushtool1, 'Enable', 'off')
    % set(handles.uipushtool2, 'Enable', 'off')

    guidata(hfig, handles);
    close(h) % close waitbar

catch exception
    if ~isempty(h); close(h); end
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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


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

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

[fileName, pathName] = uiputfile('*.hedi','Izvēlieties saglabājamo projekta failu');
if fileName==0, return; end

try
    fileF = fullfile (pathName, fileName);
    if isempty(strfind(fileName, '.hedi')), fileF = [fileF '.hedi']; end

    hf = gcf;
    h = waitbar(0.5,'Lūdzu, pagaidiet...','Name','Saglabājam projektu...');

    model = handles.model;
    X1 = handles.X1;
    Y1 = handles.Y1;
    dX = handles.dX;
    dY = handles.dY;

    Upes = handles.Upes;
    Prognozes = handles.Prognozes;
    save(fileF, 'model', 'X1', 'Y1', 'dX', 'dY', 'Upes', 'Prognozes', '-v7.3');

    close(h)

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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


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


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


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


% --------------------------------------------------------------------
function InsertDamMenu_Callback(hObject, eventdata, handles)
% hObject    handle to InsertDamMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)


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


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

try
    if isempty(handles.hCaurtece), return; end

    [fileName, pathName] = uiputfile('*.matc','Izvēlieties caurteces datu failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    hCaurtece = handles.hCaurtece;
    pos = getPosition(hCaurtece);
    coords = pos;
    coords(:,1) = (coords(:,1)-1)* handles.dX + handles.X1;
    coords(:,2) = handles.Y1 - (coords(:,2)-1)* handles.dY;
    coordsCaurtece = coords;

    save(fileD, 'coordsCaurtece');

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% --------------------------------------------------------------------
function LoadCaurteceMenu_Callback(hObject, eventdata, handles)
% hObject    handle to LoadCaurteceMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    if any(handles.model==0)
        errordlg('Caurteces zīmē tikai pēc tam, kad visos modeļa punktos ir augstuma dati
<>0', 'Caurtece', 'modal');
        return
    end

    handles2 = handles;
    if ~isempty(handles.hCaurtece)
        atbilde = questdlg('Caurtece jau atzīmēta, dzēst?', 'Caurtece', 'Atcelt', 'Labi',
'Labi');
        if strcmp(atbilde, 'Labi')
            delete(handles.hCaurtece);
            handles2.hCaurtece = [];
        else
            return
        end
    end

    [fileName, pathName, FilterIndex] = uigetfile('*.matc','Izvēlieties caurteces datu
failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    load(fileD, '-mat');
    if ~exist('coordsCaurtece','var')
```

```
        errordlg('Fails nesatur datus par caurteci!', 'Kļūda', 'modal');
        return
    end
    coords = coordsCaurtece;
    pos = zeros(2);
    pos(:,1) = (coords(:,1) - handles.X1)/ handles.dX + 1;
    pos(:,2) = (handles.Y1 - coords(:,2))/ handles.dY + 1;
    hCaurtece = impoly(gca, pos, 'Closed', false);

    handles2 = handles;
    handles2.hCaurtece = hCaurtece;
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

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

try
    if isempty(handles.hCaurtece)
        return
    end

    handles2 = handles;
    pos = round(getPosition(handles.hCaurtece));
    heightStart = handles.model(pos(1,2),pos(1,1));
    heightEnd = handles.model(pos(2,2),pos(2,1));
    model2 = handles.model;

    x1 = pos(1,1); x2 = pos(2,1); dx = x2-x1;
    y1 = pos(1,2); y2 = pos(2,2); dy = y2-y1;
    steps = sqrt(dx*dx + dy*dy);
    ddx = dx/steps;
    ddy = dy/steps;
    ddh = (heightEnd - heightStart)/steps;
    for istep = 1:steps
        x = round(x1+istep*ddx);
        y = round(y1+istep*ddy);
        h = heightStart + istep*ddh;
        if dx>dy % Ja vairāk horizontāli
            model2(y-2:y+2,x) = h;
        else    % Ja vairāk vertikāli
            model2(y,x-2:x+2) = h;
        end
    end
    set(handles2.hImage, 'CDATA' , model2);
%     drawnow

    delete(handles.hCaurtece);
    handles2.hCaurtece = [];
    handles2.model = model2;
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end
% --------------------------------------------------------------------
function DeleteCaurteceMenu_Callback(hObject, eventdata, handles)
% hObject    handle to DeleteCaurteceMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
try
    handles2 = handles;
    delete(handles.hCaurtece);
    handles2.hCaurtece = [];
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

% --------------------------------------------------------------------
function FilterForest_Callback(hObject, eventdata, handles)
% hObject    handle to FilterForest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
hh=helpdlg({'Ar lauztu līniju atzīmēt meža apgabalu, kura koku vainagus centīsimies
atfiltrēt.',...
    'Pēdējo punktu atzīmēt ar dubultklikšķi.'}, 'Meža filtrs');
uiwait(hh);
hPoly = impoly;

if isempty(hPoly), return; end
pos = round(getPosition(hPoly));

try
    handles2 = handles;
    model = handles.model;
    [r,c] = size(model);
    xx= pos(:,1); yy = pos(:,2);
    xmin = max(1,min(xx))-2;
    xmax = min(c,max(xx)+2);
    ymin = max(1,min(yy)-2);
    ymax = min(r,max(yy)+2);
    modelF = model(ymin:ymax,xmin:xmax);
    modelFf = imerode(modelF, strel('disk',2,4));
    modelE = model*0;
    modelE(ymin:ymax,xmin:xmax) = modelFf;

    mask = createMask(hPoly);
    model2 = model.*single((1-mask)) + modelE.*single(mask);
    handles2.model = model2;
    handles2.modelOld = model;
    set(handles.hImage, 'CDATA' , model2);

    guidata(hObject, handles2);
    delete(hPoly);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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

try
    handles2 = handles;
    handles2.model = handles.modelOld;
    guidata(hObject, handles2);
    set(handles.hImage, 'CDATA' , handles2.model);
    drawnow
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
```

```
end


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


% --------------------------------------------------------------------
function Untitled_2_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


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


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

if strcmp(get(hObject, 'Checked'), 'on')
    set(hObject, 'Checked', 'off');
    if ~isempty(handles.hBorder),
        set(handles.hBorder, 'Visible', 'off');
    end
else
    set(hObject, 'Checked', 'on');
    if ~isempty(handles.hBorder),
        set(handles.hBorder, 'Visible', 'on');
    end
end


% --------------------------------------------------------------------
function Colorbar_OffCallback(hObject, eventdata, handles)
% hObject    handle to Colorbar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
colorbar('off');


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


% --------------------------------------------------------------------
function DataTip_OffCallback(hObject, eventdata, handles)
% hObject    handle to DataTip (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.posText, 'Visible','off')
% --------------------------------------------------------------------
function DataTip_OnCallback(hObject, eventdata, handles)
% hObject    handle to DataTip (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.posText, 'Visible','on')


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

try
    [fileName, pathName, FilterIndex] = uigetfile('*.matpl','Izvēlieties plūdu augstuma
datu failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    load(fileD, '-mat');
    if ~exist('augstumi','var') || ~exist('coords','var')
        errordlg('Fails nesatur datus par ūdens līmeņiem plūdos!', 'Kļūda', 'modal');
        return
    end
    virsotnes = numel(augstumi);
    pos = zeros(virsotnes,2);
    pos(:,1) = (coords(:,1) - handles.X1)/ handles.dX + 1;
    pos(:,2) = (handles.Y1 - coords(:,2))/ handles.dY + 1;

    if isfield(handles, 'modelPluudu'),
        modelPluudu = handles.modelPluudu;
    else
        modelPluudu = handles.model;
    end

    %modelPluuduD = modelPluudu + delta;
    % TODO jāaprēķina plūdu līmeņi, interpolējot/ekstrapolējot no importētajiem punktiem

    UpePluudu = handles.UpePluudu;
    UpePluuduPix = UpePluudu.Pixels;
    modelPluudu = modelPluudu .* single(~UpePluuduPix) + modelPluuduD .*
single(UpePluuduPix);

   % c




    handles2 = handles;

    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


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

try
    if any(handles.model==0)
        errordlg('Plūdu prognozes līniju ievada tikai pēc tam, kad visos modeļa punktos
ir augstuma dati <>0', 'Plūdi', 'modal');
        return
    end
```

```
    handles2 = handles;
    if ~isempty(handles.hLineFloodForecast)
        atbilde = questdlg('Plūdu prognozes līnija jau atzīmēta, dzēst?', 'Plūdi',
'Atcelt', 'Labi', 'Labi');
        if strcmp(atbilde, 'Labi')
            delete(handles.hLineDambis);
            delete(handles.hTextDambis);
            handles2.hLineDambis = [];
            handles2.hTextDambis = [];
        else
            return
        end
    end

    hh=helpdlg({'Atzīmējiet lauztu līniju, kura imitē plūdu prognozes vietu uz upes.',...
        'Pēdējo punktu atzīmēt ar dubultklikšķi'}, 'Plūdi');
    uiwait(hh);
    hPoly = impoly('Closed', false);
    %get(handles.hImage, 'parent'), 'Closed', false); ????

    if isempty(hPoly), return; end
    pos = round(getPosition(hPoly));
    virsotnes = size(pos,1);
    hold on

    handles2.hLineFloodForecast = hPoly;
    htext = [];
    for iv = 1:virsotnes
        hv = text(pos(iv,1), pos(iv,2)+4, num2str(iv), 'Color','white', 'UserData', 0,
'ButtonDownFcn', {@pointFloodButtonDownFcn,iv});
        htext = [htext hv];
    end
    handles2.hTextFloodForecast = htext;
    guidata(hObject, handles2);
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

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

try
    if isempty(handles.hLineFloodForecast), return; end

    [fileName, pathName] = uiputfile('*.matpl','Izvēlieties plūdu prognozes vietas datu
failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    handles2 = handles;
    hLineFloodForecast = handles.hLineFloodForecast;
    pos = getPosition(hLineFloodForecast);
    coords = pos;
    coords(:,1) = (coords(:,1)-1)* handles.dX + handles.X1;
    coords(:,2) = handles.Y1 - (coords(:,2)-1)* handles.dY;

    hTextFloodForecast = handles.hTextFloodForecast;
    augstumi = get(hTextFloodForecast, 'UserData');

    save(fileD, 'augstumi', 'coords');
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end
```

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

try
    [fileName, pathName, FilterIndex] = uigetfile('*.matpl','Izvēlieties plūdu prognozes
vietas datu failu');
    if fileName==0, return; end
    fileD = fullfile (pathName, fileName);

    load(fileD, '-mat');
    if ~exist('augstumi','var') || ~exist('coords','var')
        errordlg('Fails nesatur datus par plūdu prognozes vietu!', 'Kļūda', 'modal');
        return
    end
    virsotnes = numel(augstumi);
    pos = zeros(virsotnes,2);
    pos(:,1) = (coords(:,1) - handles.X1)/ handles.dX + 1;
    pos(:,2) = (handles.Y1 - coords(:,2))/ handles.dY + 1;
    hLineFloodForecast = impoly(gca, pos, 'Closed', false);

    handles2 = handles;
    handles2.hLineFloodForecast = hLineFloodForecast;
    htext = zeros(1,virsotnes);
    for iv = 1:virsotnes
        htext(iv) = text(pos(iv,1), pos(iv,2)+4, num2str(iv), 'Color','white',
'UserData', augstumi{iv}, 'ButtonDownFcn', {@pointFloodButtonDownFcn,iv});
    end
    handles2.hTextFloodForecast = htext;
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

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

try
    handles2 = handles;
    pos = getPosition(handles.hLineFloodForecast);
    augstumi = get(handles.hTextFloodForecast, 'UserData');
    virsotnes = numel(augstumi);
    augstumi = cell2mat(augstumi);
    virsotnesArAugstumiem = find(augstumi>0);

    distances = zeros(virsotnes);
    for ii=1:virsotnes-1
        d = 0;
        for jj=ii+1:virsotnes
            d = d+CalcDistance(pos(jj,:), pos(jj-1,:));
            distances(ii,jj) = d;
            distances(jj,ii) = d;
        end
    end

    % ekstrapolē no 1. zināmā atpakaļ
    if virsotnesArAugstumiem(1)>1,
        iv1 = virsotnesArAugstumiem(1);
        iv2 = virsotnesArAugstumiem(2);
```

```
        h1 = augstumi(iv1);
        h2 = augstumi(iv2);
        dh = h2-h1;
        ddist = distances(iv1,iv2);
        for iiv = 1:iv1-1
            h = h1 - distances(iiv,iv1)/ddist * dh;
            augstumi(iiv) = h;
        end
    end

    % interpolē starp zināmiem
    for iv=1:numel(virsotnesArAugstumiem)-1
        iv1 = virsotnesArAugstumiem(iv);
        iv2 = virsotnesArAugstumiem(iv+1);
        if (iv2-iv1)<2, continue; end
        h1 = augstumi(iv1);
        h2 = augstumi(iv2);
        dh = h2-h1;
        ddist = distances(iv1,iv2);;
        for iiv = iv1+1:iv2-1
            h = h1 + distances(iiv,iv1)/ddist * dh;
            augstumi(iiv) = h;
        end
    end

    % ekstrapolē no pēdējā zināmā uz priekšu
    if virsotnesArAugstumiem(numel(virsotnesArAugstumiem))<virsotnes,
        iv1 = virsotnesArAugstumiem(numel(virsotnesArAugstumiem)-1);
        iv2 = virsotnesArAugstumiem(numel(virsotnesArAugstumiem));
        h1 = augstumi(iv1);
        h2 = augstumi(iv2);
        dh = h2-h1;
        ddist = distances(iv1,iv2);
        for iiv = iv2+1:virsotnes
            h = h2 + distances(iiv,iv2)/ddist * dh;
            augstumi(iiv) = h;
        end
    end

    % izveidot pilnu viduslīniju ar augstumiem
    [ posSteps, posHeightsFull ] = getEqualStepHeights( pos, augstumi,  10);

    % izvēlēties upi, kurai izveidot plūdu modeli
    iUpes = findUpe(handles, pos(1,:)); % atrast upes struktūru no punkta
    if isempty(iUpes),
        errordlg('Punkti nepieder nevienai upei!', 'Kļūda');
        return;
    end
    Upe = handles.Upes(iUpes(1));
    nUpes = length(iUpes);
    if nUpes>1
        UpeNames = cell(1,nUpes);
        for iUpe = 1:nUpes
            UpeNames{iUpe} = handles.Upes(iUpes(iUpe)).Name;
        end
        [Selection, ok] = listdlg('Name', 'Upes izvēle', 'PromptString', 'Izvēlēties upi
vai fragmentu, kura plūdu prognozi izveidot',...
                'ListString', UpeNames, 'ListSize', [300 200],...
                'OKString', 'Labi', 'CancelString', 'Atcelt', 'SelectionMode', 'single');
        if ok==0, return; end
        Upe = handles.Upes(iUpes(Selection(1)));
        %disp (Upe.Name)
    end

    % iezīmē plūdu prognozes rajonu ap viduslīniju
    [rowsU, colsU] = size(Upe.Pixels);
    prognozesRajons = zeros (rowsU, colsU,'uint8');
    for ip = 1: numel(posHeightsFull)
```

```
        posP = round(posSteps(ip,:)); x=posP(1); y=posP(2);
        x1 = max(1,x-100); xn = min(x+100, colsU);
        y1 = max(1,y-100); yn = min(y+100,rowsU);
        prognozesRajons (y1:yn,x1:xn) = 1;
    end
    UpesPixProgn = Upe.Pixels & prognozesRajons;

    % atrod prognozēto plūdu pikseļu augstumus
    % atrast upes punktu augstumus kā tuvākā posHeightsFull pikseļa augstumu
    model = handles.model;

    [yu, xu] = find(UpesPixProgn); % prognozes pikseļu indeksi
    npix = length(yu);
    npixinc = round(npix/100); %rādīsim katru 100-to
    hw = waitbar(0,'Lūdzu gaidiet...','Name','Veidojam plūdu prognozes modeli...');

    for ipix = 1:npix
        if mod(ipix, npixinc) == 0,
            waitbar(ipix/npix, hw);
        end
        yui=yu(ipix); xui=xu(ipix);
        dist = (posSteps(:,2) - yui).^2 + (posSteps(:,1) - xui).^2;
        [dmin,ind]=min(dist);
        model(yui,xui) = posHeightsFull(ind);
    end
    close(hw)
    options.Interpreter = 'tex';
    answer = inputdlg('Prognozes modeļa nosaukums:
\color{white} .','prognoze',1, {'Gauja100G_Prognoze'}, options);
    if isempty(answer),
        return;
    else
        Prognozes = handles2.Prognozes;
        nPrognozes = length(Prognozes);
        Prognozes(nPrognozes+1).Name = answer{1};
        Prognozes(nPrognozes+1).UpeName = Upe.Name;
        Prognozes(nPrognozes+1).Pixels = UpesPixProgn;
        Prognozes(nPrognozes+1).Heights = model .* single(UpesPixProgn);
        handles2.Prognozes = Prognozes;
    end
    delete(handles.hLineFloodForecast);
    delete(handles.hTextFloodForecast);
    handles2.hLineFloodForecast = [];
    handles2.hTextFloodForecast = [];
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end

function euclideanDistance = CalcDistance(pos1, pos2)
euclideanDistance = sqrt((pos2(1)-pos1(1))^2 + (pos2(2)-pos1(2))^2);


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

try
    handles2 = handles;
    delete(handles.hLineFloodForecast);
    delete(handles.hTextFloodForecast);
    handles2.hLineFloodForecast = [];
    handles2.hTextFloodForecast = [];
    guidata(hObject, handles2);
```

```
catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end


% --- Executes on button press in UseForecastModel.
function UseForecastModel_Callback(hObject, eventdata, handles)
% hObject    handle to UseForecastModel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
try
    status = get(hObject,'Value');
    handles2 = handles;
    if status==0
        handles2.selectedPrognoze = 0;
    else
        Prognozes = handles.Prognozes;
        nPrognozes = length(Prognozes);
        if nPrognozes==0,
            errordlg('Nav neviena plūdu prognozes modeļa!', 'Kļūda');
            set(hObject,'Value',0);
            return;
        end
        if nPrognozes>1
            PrognozeNames = cell(1,nPrognozes);
            for iPr = 1:nPrognozes
                PrognozeNames{iPr} = [handles.Prognozes(iPr).Name ' : '
handles.Prognozes(iPr).UpeName];
            end
            [Selection, ok] = listdlg('Name', 'Prognozes izvēle', 'PromptString',
'Izvēlēties izmantojamo plūdu prognozes modeli',...
                    'ListString', PrognozeNames, 'ListSize', [300 200],...
                    'OKString', 'Labi', 'CancelString', 'Atcelt', 'SelectionMode',
'single');
            if ok==0, return; end
            handles2.selectedPrognoze = Selection(1);
        else
            handles2.selectedPrognoze = 1;
        end
    end
    guidata(hObject, handles2);

catch exception
    errordlg(getExceptionStr(exception), 'Kļūda');
    logException(exception);
end
```

fillUpe2.m

```
function handlesOut = fillUpe2(handlesIn, pos);
%fillUpe Upes punktu augstumu aprēķins no atzīmētas viduslīnijas
% Ieejā:
%   handlesIn.model - augstuma modelis ieejā
%   pos - atzīmētie upes punkti pret straumi
% Izejā:
%   handlesOut.model - modelis ar piepildītiem upes punktu augstumiem

handlesOut = handlesIn;

[ posHeights, maxRadius, hhc ] = findPosHeights(handlesIn.model, pos, 50);
%maxRadius
nPointsMarked = length(posHeights);
if posHeights(nPointsMarked)<posHeights(1),
   posHeights = ones(1,nPointsMarked) * posHeights(1);
else
   if nPointsMarked<3,
       posHeights = linspace(posHeights(1), posHeights(nPointsMarked), nPointsMarked);
   else
```

```
        % median filter
        hh0 = [posHeights(1) posHeights posHeights(nPointsMarked)];
        hh1 = medfilt2(hh0,[1,3]);
        posHeights = hh1(2:nPointsMarked+1);
        % ensure that heights are ascending
        hmax = posHeights(1);
        for ip=2:nPointsMarked
            posHeights(ip) = max(posHeights(ip-1:ip));
        end
    end
end

% izveidot pilnu viduslīniju ar augstumiem
[ posSteps, posHeightsFull ] = getEqualStepHeights( pos, posHeights,  10);

hw = waitbar(0,'Lūdzu gaidiet...','Name','Piepildām upi...');

% izdalīt upes pikseļus
BW1 = handlesIn.BW0; % tukšumi (punkti ar h=0)
%BW2 = imfill(BW1, fliplr(pos), 4);
BW2 = imfill(BW1, round(fliplr(posSteps)), 4);
Upe = BW2 & ~BW1;
Upe = imfill(Upe,'holes');

% atrast upes punktu augstumus kā tuvākā posHeightsFull pikseļa augstumu
model = handlesIn.model;

[yu, xu] = find(Upe); % upes pikseļu indeksi
npix = length(yu);
npixinc = round(npix/100); %rādīsim katru 100-to
for ipix = 1:npix
    if mod(ipix, npixinc) == 0,
        waitbar(ipix/npix, hw);
    end
    yui=yu(ipix); xui=xu(ipix);
    dist = (posSteps(:,2) - yui).^2 + (posSteps(:,1) - xui).^2;
    [dmin,ind]=min(dist);
    if dmin>maxRadius^2,
        Upe(yui,xui) = false;
        continue;
    end % ja attālums >>, nepiepilda
    model(yui,xui) = posHeightsFull(ind);
    BW1(yui,xui) = true;
end
close(hw)
delete(hhc)
options.Interpreter = 'tex';
answer = inputdlg('Upes nosaukums:
\color{white} .','Upe',1, {'Gauja'}, options);
if isempty(answer),
    handlesOut = [];
else
    Upes = handlesOut.Upes;
    nUpes = length(Upes);
    Upes(nUpes+1).Name = answer{1};
    Upes(nUpes+1).Pixels = Upe;
    handlesOut.Upes = Upes;
    handlesOut.BW1 = handlesIn.BW0;
    handlesOut.BW0 = BW1;
    handlesOut.model = model;
end

end
```

findPosHeights.m

```
function [ heights, maxRadius, hhc ] = findPosHeights(model, pos, dist )
%findPosHeights Katrai koordinātei atrod augstumu no tuvākajiem pilnajiem
```

```
%                augstuma modeļa punktiem
% Ieejā:
%   model - augstuma modelis
%   pos - punktu koordinātes x,y (skaits,2), kuriem jāatrod augstums
%   distance - sākuma rādiuss pikseļos, kurā meklē tuvākos punktus ar atzīmētiem
augstumiem
% Izejā:
%   heights - punktu augstumu vektors (1,skaits)
%   maxRadius - maksimālais attālums, kurā meklēti augstuma punkti
%   hhc - uzzīmēto apļu handles

npoints = size(pos,1);
heights = zeros(1,npoints);
SE = strel('disk', dist);
NHOOD = getnhood(SE);
radius0 = floor(size(NHOOD, 1)/2);
maxRadius = radius0;
[rows, cols] = size(model);
hhc = [];
for ip = 1:npoints
    x = round(pos(ip,1));
    y = round(pos(ip,2));
    heightFound = false;
    nhood = NHOOD;
    radius = radius0;
    while ~heightFound
        [rowsn, colsn] = size(nhood);
        x1n = 1; xnn = colsn;
        y1n = 1; ynn = rowsn;
        x1 = x-radius; if x1<1, x1n=2-x1; x1=1; end
        xn = x+radius; if xn>cols, xnn=colsn-(xn-cols); xn=cols; end
%        y1 = max(1,y-radius); yn = min(rows, y+radius);
        y1 = y-radius; if y1<1, y1n=2-y1; y1=1; end
        yn = y+radius; if yn>rows, ynn=rowsn-(yn-rows); yn=rows; end
        pixelsAround = model(y1:yn,x1:xn).*single(nhood(y1n:ynn, x1n:xnn));
        pixelsAround = pixelsAround(:);
        pixelsAround(pixelsAround==0)=[];
        %minH = min(pixelsAround);
        minH = 0;
        if length(pixelsAround)>length(nhood(:))/4,
            heightFound=true;
            minH = single(percentile(pixelsAround, 0.1));
        else
            radius = round(radius*1.2);
            SE = strel('disk', radius, 0);
            nhood = getnhood(SE);
            radius = floor(size(nhood, 1)/2);
            maxRadius = max(maxRadius, radius);
        end
    end
    heights(ip) = minH;
    hc=rectangle('Position',[x-radius y-radius radius*2+1 radius*2+1],'Curvature',[1 1]);
    drawnow;
    pause(0.1);
    hhc=[hhc hc];
end

end
```

findUpe.m

```
function iUpes = findUpe( handles, pos )
%findUpe Atrod upi, kam pieder pozīcija 'pos'
%   Ja neatrod, atgriež 0. Ja nav upju, atgriež []
% pos- x,y koordinātes

iUpes = [];
```

```
if ~isfield(handles, 'Upes'), return; end
Upes = handles.Upes;
nUpes = length(Upes);
if nUpes==0, return; end

x = round(pos(1));
y = round(pos(2));

for iUpe = 1:nUpes
    Upe = Upes(iUpe);
    if Upe.Pixels(y,x), iUpes = [iUpes iUpe]; end
end
```

getEqualStepHeights.m

```
function [ posSteps, posHeightsFull ] = getEqualStepHeights( pos, posHeights, delta )
%getEqualStepHeights No ieejām uzdotām virsotņu koordinātēm un augstumiem izveido lauztu
% līniju ar vienādiem attālumiem (~delta) starp punktiem un visiem punktiem atrod
augstumus.
% Ieejā:
%   pos - līnijas virsotņu koordināšu x,y pāri (skaits,2)
%   posHeights - virsotņu augstumi
%   delta - vēlamais attālums starp punktiem (m)
% Izejā:
%   posSteps - līnijas punkti ar ~vienādiem attālumiem delta
%   posHeightsFull - posSteps punktu augstumi

nPoints=size(pos,1);
posSteps=pos(1,:);
jj=1;
posHeightsFull = posHeights(1);
for istep=1:nPoints-1
    dx=pos(istep+1,1)-pos(istep,1);
    dy=pos(istep+1,2)-pos(istep,2);
    dh=posHeights(istep+1)- posHeights(istep);
    dist=sqrt(dx*dx+dy*dy);
    nrMinorSteps = round(dist/delta);
    nrMinorSteps = max(nrMinorSteps,1);
    dminor = dist/nrMinorSteps;
    dxminor = dx/nrMinorSteps;
    dyminor = dy/nrMinorSteps;
    dhminor = dh/nrMinorSteps;
    for iminorstep=1:nrMinorSteps

        nextX=posSteps(jj,1)+dxminor;
        nextY=posSteps(jj,2)+dyminor;
        nextH=posHeightsFull(jj)+dhminor;
        posSteps = [posSteps; [nextX nextY]];
        posHeightsFull = [posHeightsFull nextH];
        jj=jj+1;
    end
end


end
```

getExceptionStr.m

```
function strException = getExceptionStr(exception)
strException = [datestr(now) ' Exception ' sprintf('%s : %s\n',exception.identifier,
exception.message) getStackTrace(exception)];
```

getStackTrace.m

```
function strStackTrace = getStackTrace(exception)
stack = exception.stack;
len = length(stack);
strStackTrace = sprintf('%s\n', 'Stack trace:');
```

```
for line = 1:len
    strStackTrace = [strStackTrace sprintf('    %s at %i\n',exception.stack(line).name,
exception.stack(line).line)];
end
```

logException.m

```
function logException(exception)
disp(getExceptionStr(exception))
```

percentile.m

```
function perc_le = percentile(data, percent)
%percentile Aprēķina vērtību, par kuru mazākas ir 'percent' procenti
% vērtības no masīva 'data

d = sort(data(:));
n = length(d);
ind = ceil(n*percent/100);

perc_le = d(ind);

end
```

pointButtonDownFcn.m

```
function pointButtonDownFcn(hObject, evt, id)
    data = guidata(hObject);
    currentHeight = get(data.hTextDambis(id), 'UserData');
    %msgbox({num2str(currentHeight)},'Lūdzu ievadiet punkta augstumu v.j.l.!');
    answer = inputdlg({'Lūdzu ievadiet punkta augstumu virs jūras līmeņa:
'},'Augstums',1,{num2str(currentHeight)});
    if isempty(answer),
        return;
    end
    try
        augst = str2double(answer{1});
        set(data.hTextDambis(id), 'UserData', augst);
    catch ex
        errordlg('Ievades kļūda!', 'Kļūda', 'modal');
    end
    guidata(hObject, data);

end
```

pointFloodButtonDownFcn.m

```
function pointFloodButtonDownFcn(hObject, evt, id)
    data = guidata(hObject);
    currentHeight = get(data.hTextFloodForecast(id), 'UserData');
    %msgbox({num2str(currentHeight)},'Lūdzu ievadiet punkta augstumu v.j.l.!');
    answer = inputdlg({'Lūdzu ievadiet punkta augstumu virs jūras līmeņa:
'},'Augstums',1,{num2str(currentHeight)});
    if isempty(answer),
        return;
    end
    try
        augst = str2double(answer{1});
        set(data.hTextFloodForecast(id), 'UserData', augst);
    catch ex
        errordlg('Ievades kļūda!', 'Kļūda', 'modal');
    end
    guidata(hObject, data);

end
```