Developer Manual

Project subject: Improving an OCR program

Project instructo: Hananel Hazan

Project performed by: Alex Heifetz & Vadim Maizlin

Short Description: The program HOCR was originally written by Niv Maman and Maxim Drabkin under the guidance of Hananel Hazan. The program uses a neural network to recognize text by transforming the image of a certain character to a binary array (1s and 0s) with values representing pixel grey values and, using a neural network mapping it to a certain character. The program could learn a few fonts and recognize Hebrew texts typed in those fonts. Our project goal was to enable adding new fonts, recognize English as well as Hebrew, recognize handwriting, add spell checking corrections, improve general recognition success, and add the ability to synchronize with a server.

Note: This manual is partly written in Hebrew.

Changes to the original program (Excluding the major new functions described later)

Encog update: The program used v2.5.2 of Encog. It now uses v3.1.0

GUI changes: New buttons and menu options were added, and some new forms were added to perform new functions

Training: Resilient propagation is used instead of back propagation.

Neural network change: The program now uses 3 neural networks working synchronously instead of 1. The reason for this was that the results depend on the training of the network. 2 different training sessions even with the same error can recognize the same character differently, once correctly and once incorrectly. In addition too low an error results in over fitting and thus lower recognition success. But higher error must necessarily include more deliberate mistaken recognition. By training 3 networks separately each gets different recognition patterns and synchronizing their recognition of the same character and choosing the best average result improves the handwriting recognition rate by 10-20%.

Using the AForge API: most graphical functions were replace by AForge imaging filters and new ones all used AForge.

Noise handling was improved: The program ignores very small noise, though more noise still creates problems.

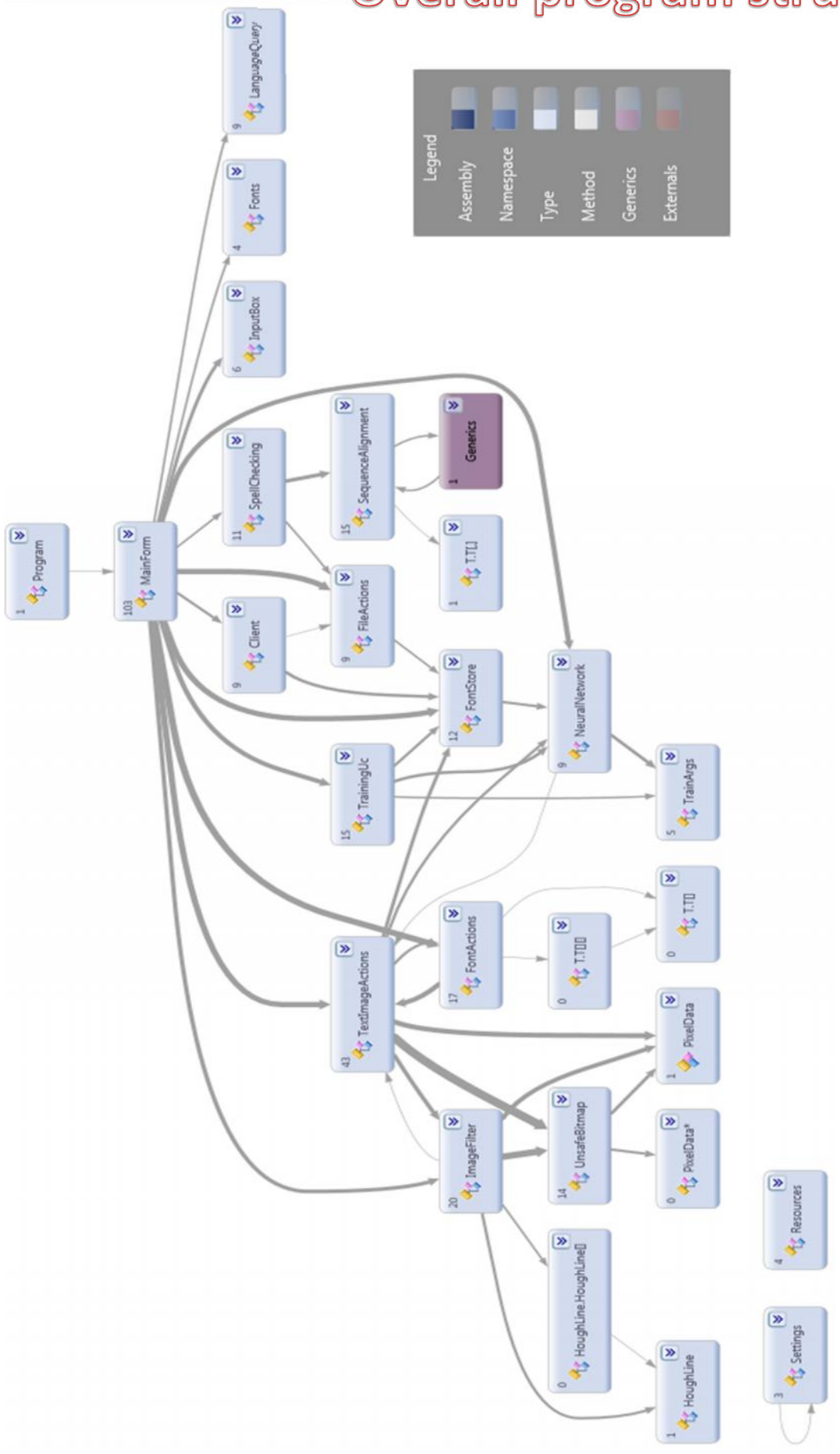
Rotation: The rotation function was improved to minimize memory use and so enable rotation of larger images, and better processing.

English: Recognition of English character was added. The user can choose whether his text is English, Hebrew or both.

Better character imaging: smaller characters are not zoomed in to resemble larger ones. This improves the recognition of characters such as “ו”.

Many other improvements in letter recognition, separation, and other areas.

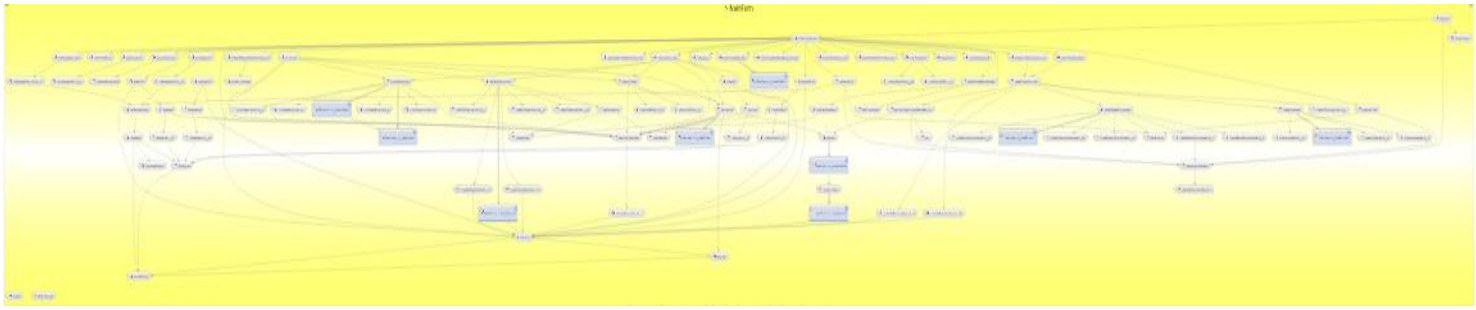
Overall program structure



The UI Classes

Mainform

The program's main form.



Properties:

```
private const int MaxWidth = 500; // Max width size for the image. larger images will be
                                  // resized to fit
private const int MaxHeight = 800;// Max height size for the image. larger images will be
                                  // resized to fit

//GUI
private Form _trainForm; //training dialog
private int _currentPage; //saves the current page of image
private int? _markedLine; //number of marked line
private int? _markedLetter; //number of marked letter
private OpenFileDialog fileDialog; // Used to select multiple handwriting learning pages
private PrintDocument printer = new PrintDocument(); // used to pring handwriting learning pages
private int Pages = 0; // shows the handwriting learning pages numbers
private BackgroundWorker worker = new BackgroundWorker(); // used to show a status bar
public int _progress=0; // shows the status bar progress
private int _languageselector; // selects the language mode to use


//Image
private Bitmap _bitmap; //bitmap of current image
private Bitmap _picture; //bitmap of current shown image
private Bitmap[][] _letters; //letters of current image
private int[][][] _letterBounds; //letters bounds of current image
private string _currentImagePath; //path of current image
private Bitmap[] _HandScans = new Bitmap[6]; // contains the handwriting learning pages

//Neural Network
private NeuralNetwork _fontNetwork; //pixels neural network of current font
private FontStore _fontstore; // contains the font information
private string _currentFontPath; //path of current font pixels
```

```
private int _inputLayer; //size of input layer pixels
private readonly int _middleLayers = 50; //size of middle layer
private readonly int _outputLayer = FontActions.LettersInNetwork.Length; //size of output
                                                                          // layer

//Script
private bool _scriptMode; //indicates if we use script mode
```

Methods:

AdjustWindow: Adjust window to size of content
AutoDetectFontTask: read image, recognize which font using in it and load that font. (it reads only the first line to determine)

CalculateLetter: Get letter bitmap, calculate which letter is it, and return it's index.

CenterizeClick: Event of clicking on Centerize button that call for centeize task and center the image

CenterizeTask: Center the image and save it into current bitmap. It also show the result on screen

CleanClick: Event of clicking on Clean button, that call for Clean task

CleanTask: Task that take the current image bitmap and clean it using the median filter

CreateAllFontsScriptClick: Event of clicking on Create All Fonts Script that call for the appropriate task that create the font images and then create the networks from it

CreateAllFontsScriptTask: Task that create the font images and then create the networks from it

CreateFontClick: Event of clicking Create Font button, that loads font image from using dialog and create font network from it.

CreateFontImages: Create font images of all known fonts of windows

CreateFontTask: Create font action that take the current image and make font network from it. return message in case of error, otherwise returns null.

CreateNetworksFromImages: Create all networks from images of fonts that are now known to the application

EmptyFontClick: Event of clicking on Empty Font button, that create an empty font network

FontsComboBoxSelectedIndexChanged: Event of changing Font ComboBox selection, that loads the selected network.

GetMultiPageTextTask: Task of recognizing text of a multi pages image or pdf

GetPageTextTask: Task of recognizing text of one page out of multi pages image. It also returns parameter that indicates whether operation succeed

GetTextCallBack: Called back after auto detect font, and call for get text task

GetTextClick: Event of clicking Get Text button, that use the network to retrieve the text.

GetTextTask: Task that been called when click on Get Text button

InAction: Get boolean that indicates if application in action, and enabled or disable the buttons depending on state.

InitializeComponent: Required method for Designer support - do not modify the contents of this method with the code editor.

InitializeFolders: Check existence of fonts and results folders and create them if needed

LoadImage: Load the image to picturebox and return bitmap. returns null in case of failure loading image.

MarkCurrentLetter: Mark the current letter on source picturebox, using the letters bounds

PageNextClick: Event of clicking on Next Page button that showing the next image

PagePrevClick: Event of clicking on Prev Page button that showing the prev image

RepairNetworkClick: Event of clicking Repair Network, that get letter bitmap and correct letter in text and repair the network for current letter.

RestoreWindowState: Restoring window to be enabled again after finishing loading text result

RotateClick: Event of clicking on Rotate button, that call for Rotate task

RotateTask: Task that take the current image bitmap and rotate it till straight if needed

SelectImageClick: Event of Clicking Select Image Button, that open file dialog and asks you for image to use.

SelectImageDialog: Open selection image dialog, set image and return boolean value indicates if it succeed

ShowLetterResult: Get image bitmap and show it at the result picture box

ShowPagesButtons: Get boolean indicates whether to show next\prev buttons. if true, set their location.

ShowResultLetterText: Get line and letter numbers and show that letter on screen at the repair section

ShowSource: Get image bitmap and show it at the source picture box

ShowText: Show Text Result

StopProcessClick: Event of clicking on Stop Process button that stop any process by setting his own button visibility to false. It is only posible if the process checkes the visiblity of the button every cycle.

TextResultSelectionChanged: Event of changing selection text in Text Result, that calc selected letter and show it on repair section

ThresholdClick: Event of clicking on Threshold button, that call for Threshold task

ThresholdTask: Task that take the current image bitmap and fix it using the threshold filter

TrainAllScriptClick: Event of clicking on Train All Script button that opens the training user control and start training all fonts.

TrainNetworkClick: Event of clicking Train Network button, that train the network and show test results.

UpdateFontsComboBox: Enter fonts items into fonts combobox

CXMOpened: Handles right click in text window

learnFromPagesToolStripMenuItem_Click: Learns font from handwriting learning pages.

printInputPagesToolStripMenuItem_Click: Prints handwriting learning pages.

printDoc_PrintPage: Creates handwriting learning pages for printing.

Sort: Sorts handwriting learning pages from first to last.

addWordToolStripMenuItem_Click: Handles clicking on add word menu item.

toolStripMenuItem1_Click: Handles clicking on “Deploy dataset” menu item.

fromListToolStripMenuItem_Click: Handles creating new fonts from selection.

worker_DoWork: Used by background worker. Used for status bar updating.

Statusbar: Creates background worker. Used for status bar updating.

updateFontsToolStripMenuItem_Click: Creates Update Form object and makes it visible

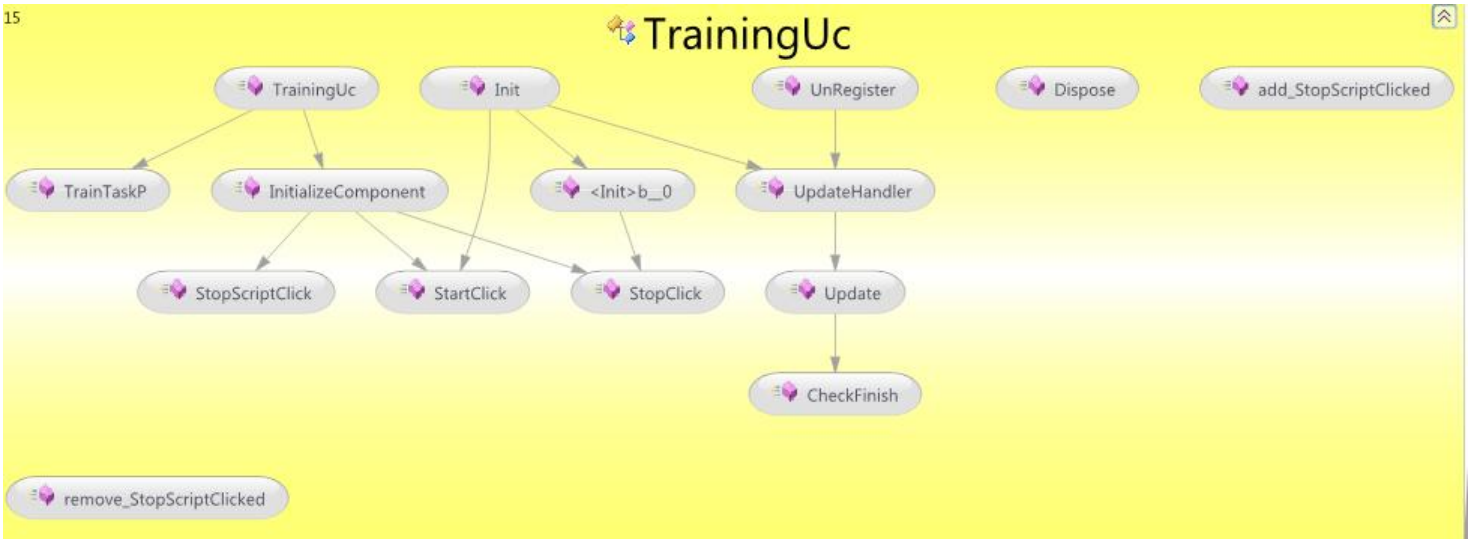
openUpdatesLogFileToolStripMenuItem_Click: Opens `serverlog.txt log file what exist in Client.ServerFolderPath`

showClientNetworksToolStripMenuItem_Click: Creates NetShowForm (listbox window for print output existing networks). Checks which fonts networks exists in Client Folder and prints this list with properties of this fonts.

settingsToolStripMenuItem_Click: Creates new Server Settings Form and make it visible.

TrainingUC

Used for neural net training.



Properties:

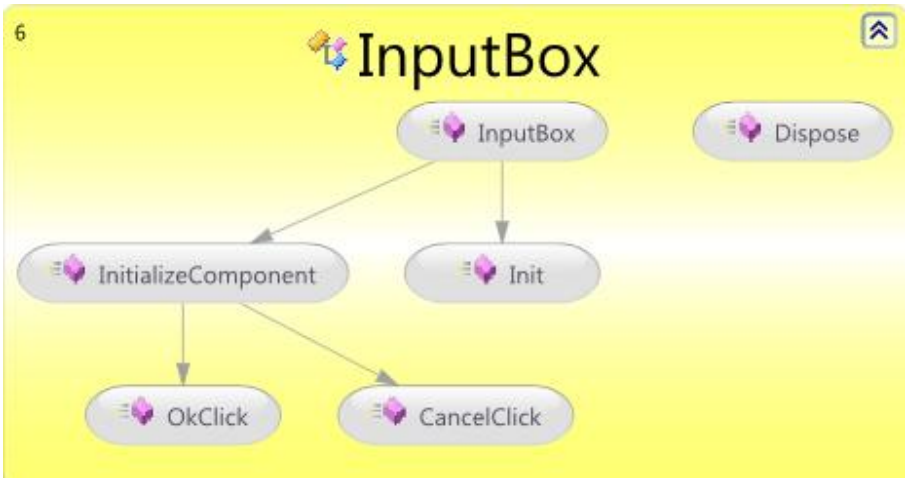
```
        private int _repeats; //number of repeats
        private double _error;// training parameters
        private NeuralNetwork _fontNetwork; //pixels neural network of current font
        private FontStore _fontstore;

        private readonly Action<int, double> _trainActionP; //action used for train task P
        private Form _parentForm; //parent form, used for closing
        public event Action StopScriptClicked; //event of stoping script

        private bool _activeTrain; //indicates if train is active
        private bool _scriptMode; //indicates if we use script
```

Methods:

TrainingUc: Constructor method.
Init: Initializes training UI.
StartClick: Handles clicking “Start training” button.
StopClick: Handles clicking “Stop training” button.
TrainTaskP: Starts training.
UpdateHandler: Runs update.
Update: Method run after every training cycle to update the UI.
CheckFinish: Check if both trainig finished and update user interface if needed.
StopScriptClick: Event of clicking on Stop Script button that stops script invoking event and close window.
UnRegister: Unregister all events. You should call this method from outside class after closing dialog.

InputBox

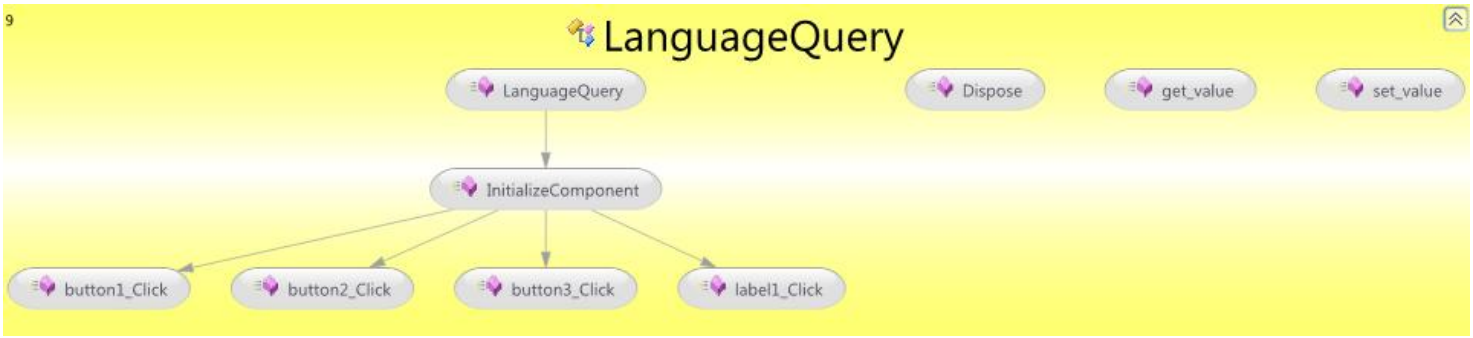
Shows input box for entering new font name.



Properties:
`Result: The name entered`

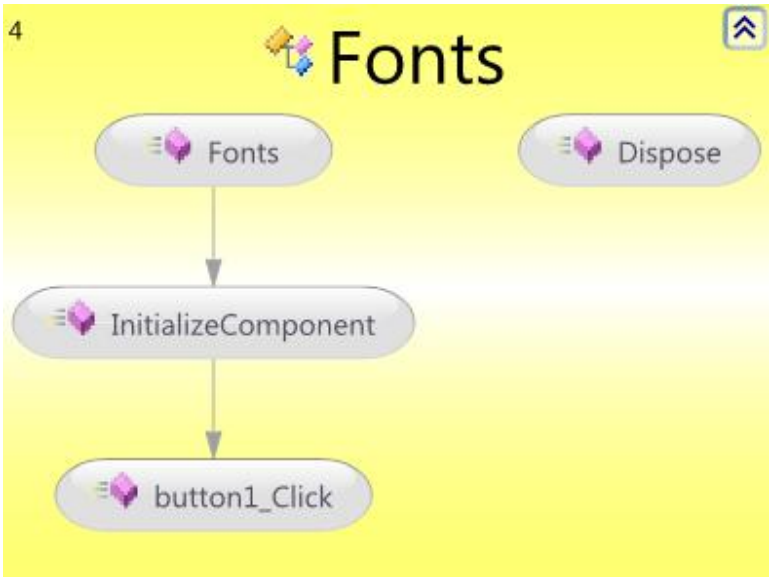
LanguageQuery

Used for choosing what language the text is in.



Fonts

Used for choosing new fonts for the program to learn



FontStore

This class contains the font network and other information on the font. It is serializable and is saved in a file.

Properties:

Network: The neural network

Version & error: version and error of the network

Handwriting: Boolean shows whether the font is a handwriting font

Methods:

getHW: returns whether the font is a handwriting font

getNetwork/setNetwork :returns/saves the network.

getError/setError :Returns/saves the error rate

getSize :returns dataset size.

getVersion :returns network version .

StoreFont :Saves the font with the neural net and other properties..

NeuralNetwork

This class contains and works with the neural net.

Properties :

_network 1:An Encog basic neural network.
_network 2:An Encog basic neural network.
_network 3:An Encog basic neural network.
_dataSet :The net's dataset. Contains the input in the form of binary arrays representing pixels in the letter image and arrays indicating the letter's index number in the LettersInNetwork
_isActive ו **IterationChanged** : Used in the network's training. **IterationChanged** is launched after every training cycle. **_isActive** allows stopping the training.

Methods:

Train :Uses resilient propagation in training the network
StopTrain :Stops the training.
Compute :Computes the likely outputs.
Test :Used for testing and debugging. Not used in normal operation..
AddToDataSet :Gets a letter in the form of a binary array (1's and 0's) and adds to the network's dataset.
Reversedataset :Turn the characters contained in the dataset back to images and saves them in the ims folder.
NeuralNetwork :The constructor method creating the network and dataset..

FileActions

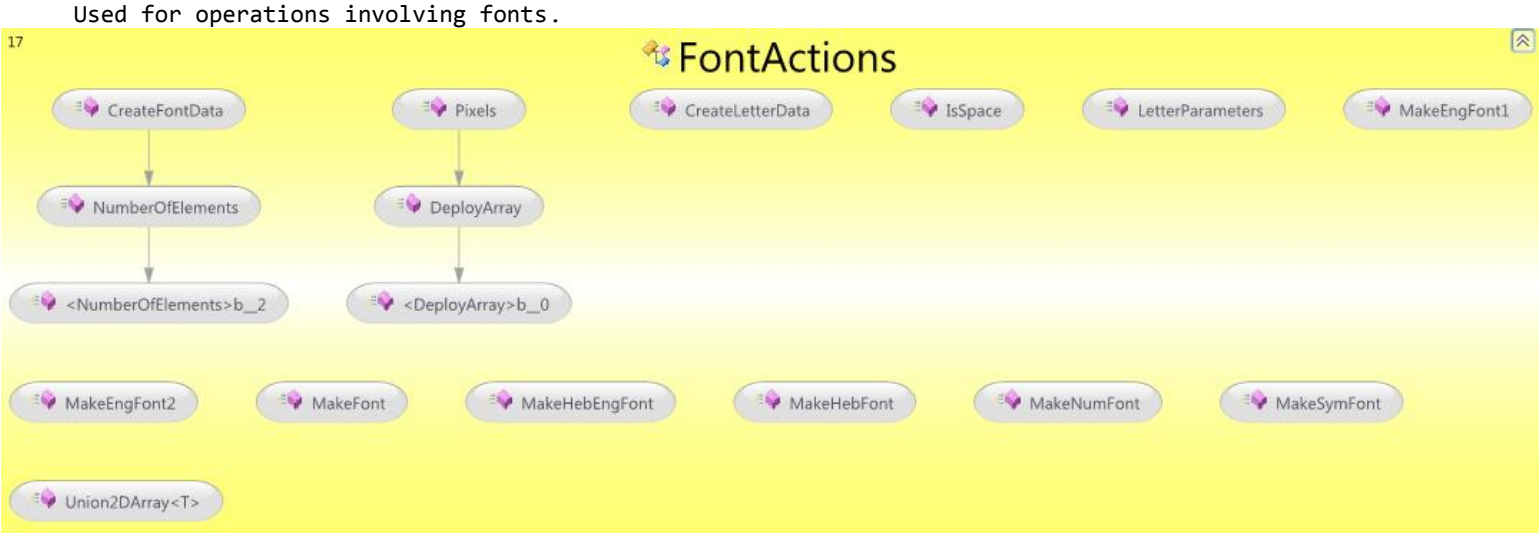
Used for operations on files.

Methods:

LoadNetwork :Loads a .net file containing a fontstore
SaveNetwork :gets a font store and saves it in a .net file
SaveTextIntoFile :saves a text in a text file. In case of adding word to the spell checking dictionary, appends the word to an existing file.
LoadTextFromFile :Loads a text file
ConvertPdfToTiff :converts a PDF file to a TIFF file
IsFontNameAvailable :Checks if a font name is available

FontActions

```
Used for operations involving fonts.
```



Properties:

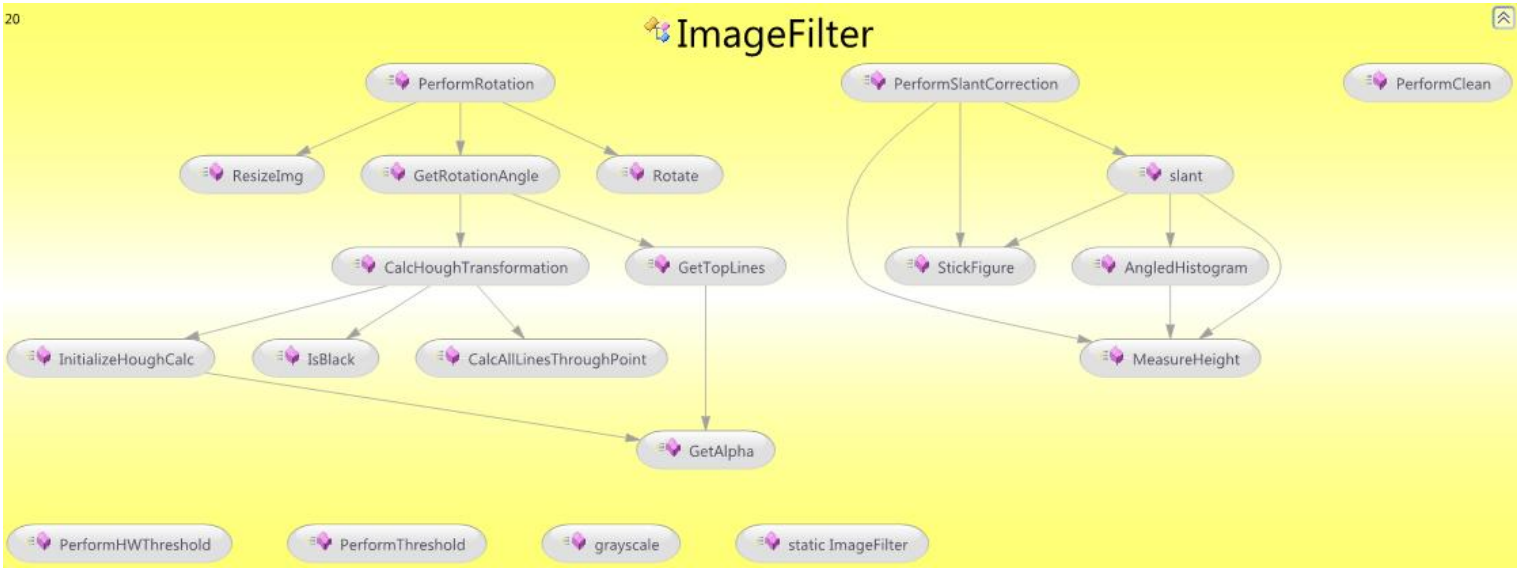
LetterSize :Size of printed letters
HWLetterWidth & **HWLetterHeight** :Size of handwritten letters.
LettersOfInitial & **LettersInNetwork** :contain all the characters the program is trained to recognize.
LettersPrint? :Used to print the letters on a page for the program to learn.
FontsFolderPath :The address of the folder containing the font files

Methods:

CreateFontData :Creates a dataset from the pictures of letters
Union2DArray :Unites two arrays into 1. Used for creating the dataset..
CreateLetterData :Adds a letter to an existing dataset.
DeployArray :Flattens an array from an array of arrays to a single array.
Pixels :Used to transform an image of a character to a binary array.
IsSpace :Checks whether an image of a character is a space or a character.
MakeFont :Creates a font image from which the program will learn a font.
LetterParameters :Not used.
Make*Font :Creates the printed pages from which a handwritten font is learned.

ImageFilter

מכיל פילטרים שונים לעיבוד תמונה.



Properties:

ThresholdLoose & **ThresholdTight** :These determine the border between black and white. Used for thresholding .

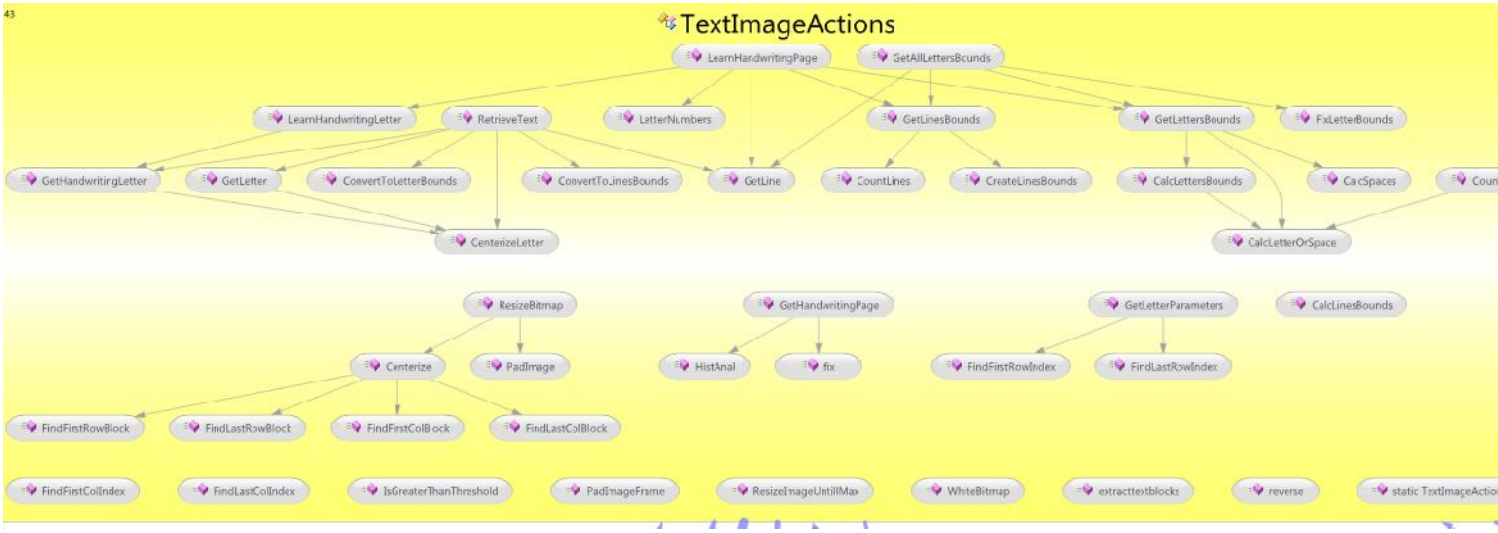
Methods:

PerformThreshold & **PerformHWThreshold** :Used for turning a grayscale image into a black and white one.
ResizeImg :Gets an image and a new width and height and resizes it..
PerformClean: Cleans noise using Median filter.
PerformRotation : Rotates a text image. Identifies angle using Hough transform and rotates by that angle.
GetRotationAngle :Finds the rotation angl.
Rotate :Gets an image and a rotation angle and rotates the image.
StickFigure :A form of thresholding used in handwritten characters..
Grayscale :Changes color images to grayscale. That is needed for some Aforge filters..
PerformSlantCorrection :Used for slant correction in handwritten characters.
Slant: Turns a handwritten character to an optimal angle.
MeasureHeight :Measures the Maximum height of a histogram. Used for line straightening and slant correction.
AngledHistogram :Used to find in which direction a character should be rotated for slant correction..

- A more detailed explanation of slant correction is available later on.

TextImageActions

Used for working on text images.



Properties:

MekademLetterSpace: A constant used for identifying whether a space is big enough to separate words.
_averageW & **_averageH**: Average height and width of the letters in the line

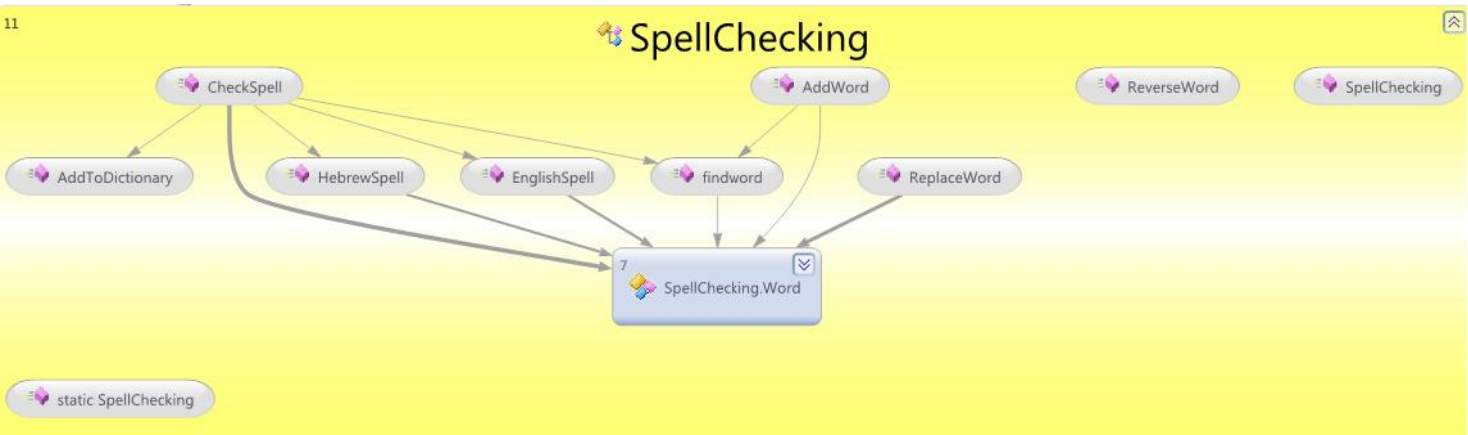
Methods:

IsGreaterThanThreshold :Checks whether the image contains pixels dark enough to be part of a character.
RetrieveText: Returns an array of character images to be used as inputs to the neural networks.
Centerize: Puts the text in the center of the image and cuts away the empty space around it.
CalcLetterOrSpace: Checks whether a column contains pixels dark enough to be part of a character.
CalcLinesBound: Returns the upper and lower bounds of a line.
CalcSpaces : Adds spaces between words to the bounds list**.**
ConvertToLetterBounds :Gets the letter bounds for all the lines and returns the bounds for a given line.
ConvertToLinesBounds: Gets the letter bounds for all the lines and returns the line bounds for a given line.
CountLines: Counts the number of lines in an image.

CountLetters: Counts the number of letters in a line.
CreateLinesBounds: Returns an array of ints showing whether this column belongs to a character.
FixLetterBounds: Not used anymore.
GetAllLettersBounds: Get bitmap and return 3D int array of letter bounds.
GetLetter: Gets the letter bounds and a bitmap line and returns an image of the letter in a size suitable for the neural net.
GetLettersBounds: Gets a line bitmap and returns the letter bounds for that line.
GetLine: Gets a line's bounds and a bitmap of the text and returns an image of the line.
GetLinesBounds: Gets a bitmap and returns the bounds of the lines in it.
ResizeImageUntillMax: Resizes a letter image for display in the corrections section.
PadImage: Get Bitmap and return new Bitmap, padded with zeros
PadImageFrame: Get Bitmap and return it with white frame. It used to avoid problems in recognition where a letter extend to the edge of frame.
WhiteBitmap: Get width and height and return white Bitmap in that size.
LearnHandwritingPage: Learns a scanned handwriting page and enters it into the dataset.
LearnHandwritingLetter: Used for cutting away the black margins around the letter on the handwriting page.
GetHandwritingLetter: Gets the bounds of a letter and the line image and returns an image of that letter in size suitable for the neural net.
LetterNumbers: Returns the expected letter indices in each handwriting page.
CenterizeLetter: Returns the bounds of the letter itself, or if it's a short character the bounds of a suitable position of the letter.
GetHandwritingPage: Gets an image of the scanned handwritten text, performs line straightening and returns an image with straightened lines.
Fix: Fixes distortions created by the line straightening.
HistAnal: Returns the bounds of the handwritten lines.
Reverse: Gets a binary array, expected character size and returns an image of that character.

SpellChecking

Used for correcting recognition result



Properties:

Word class contains: word string, number of word in the text, before spellchecking word, English reversed word, word start and end indexes in the text block, markers for word split between 2 lines.

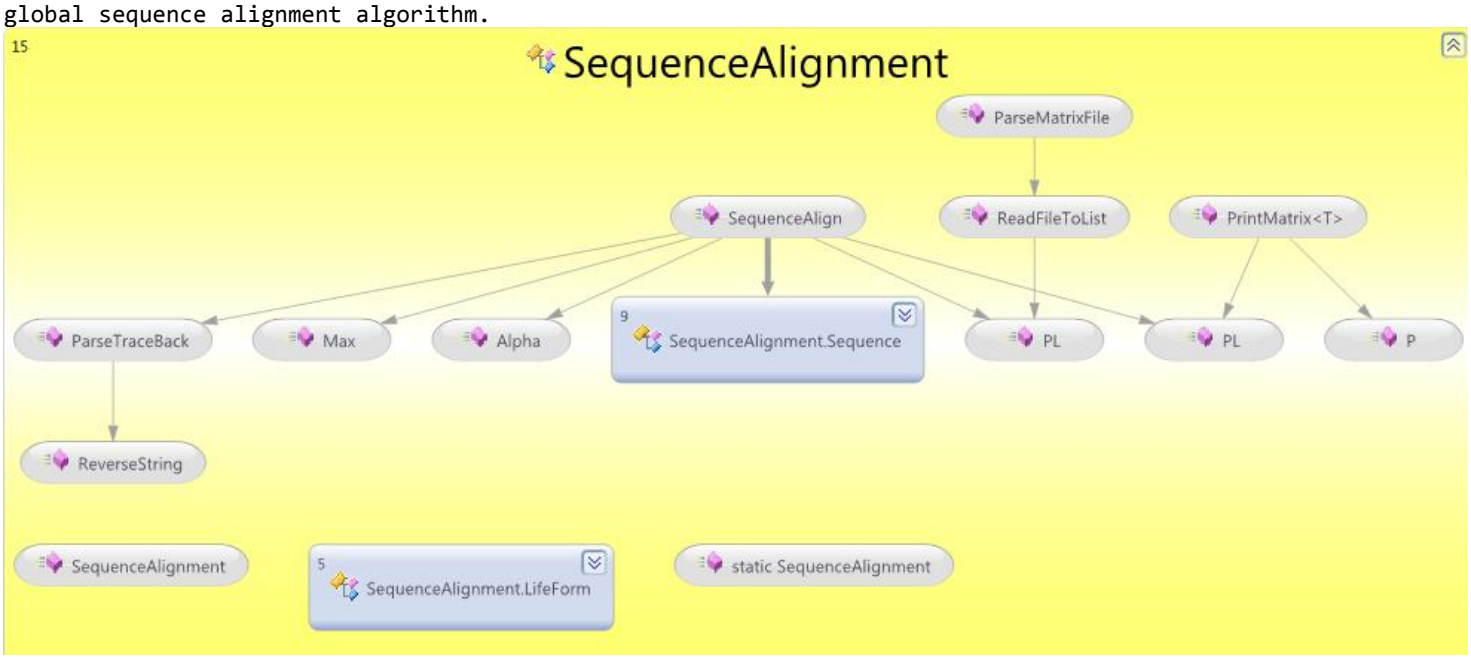
Methods:

CheckSpell: The main method in charge of spellchecking.
Findword: Finds whole words in the text block and stores them in an arrylist of Word class objects.
ReverseWord: Reverses English words. Needed when the program works in right-to-left mode.
HebrewSpell: Check word spelling using Hebrew dictionary. Gets the word, produces and grades possible suggestions and returns a corrected word.
EnglishSpell: Check word spelling using English dictionary. Gets the word, produces and grades possible suggestions and returns a corrected word.
AddToDictionary: Add words to dictionaries from added words files.
AddWord: Read a word from the text window and store it in added words file or future use by dictionaries.
ReplaceWord: Replace word in the text block with the corrected version.

- This class uses the Hunspell spell checker for giving suggestions and the sequence alignment class for grading word suggestions.

SequenceAlignment

This class was written by originally Kunuk Nykjaer and adapted for use in our project. It uses the Needleman-Wunsch global sequence alignment algorithm.



Properties:

Size: The size of the weights matrix. In this case weights are the degree to which 2 characters are visually similar.
linesMatrix: A list into which the visual similarity matrix is read from a file.
Matrix: a 2-d array into which a matrix is transferred from the linesMatrix.
Lookup: Matrix values
NL: A new line marker.
DONE, **DIAG**, **UP**, **LEFT**, **GAP**: used for traceback. Not used in our project.
Sequence class: This is what is returned by the algorithm. It contains the alignment score, and 2 strings showing the optimal alignment.

Methods:

ParseMatrixFile: Reads the file containing the visual similarity matrix and stores the matrix in a 2-d array usable by the algorithm.
PrintMatrix: Prints the matrix on screen.
SequenceAlign: performs the Needleman-Wunsch global sequence alignment algorithm.
ParseTraceBack: Performs traceback for producing a string showing the sequence alignment.
ReadFileToList: Used to read the visual similarity matrix txt file.
Max: Compare values and get the biggest one.

Alpha: Get value from matrix

UnsafeBitmap

Used for creating bitmap images. Contains unsafe code to work faster, than when working with normal bitmaps.

Properties:

_bitmap: The bitmap.
_bitmapData: The bitmap attributes.
_pBase: Contains the bitmap data in unsafe mode.
Width & **Height** : Contain the bitmap's width & height.
PixelData struct: contains the pixel's RGB color values.

Methods:

UnsafeBitmap: Constructor methods.
Dispose: Destructor method.
GetPixel & **SetPixel**: Gets/sets a pixel's color.
LockBitmap & **UnlockBitmap**: Makes the bitmap safe/unsafe for changing pixel color values

Detailed explanations

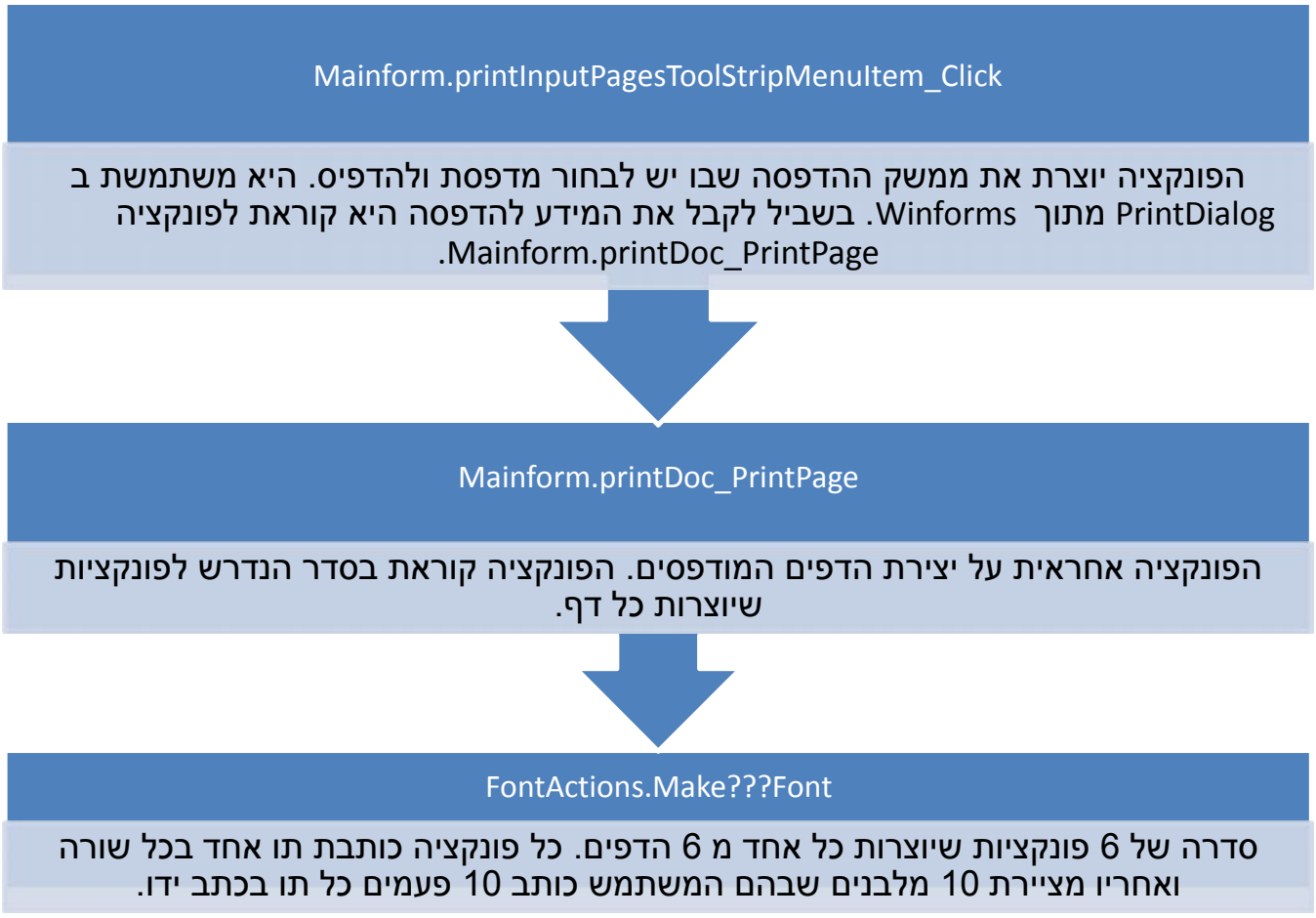
זיהוי כתב יד

תקציר: האתגרים בזיהוי כתב יד שונים במידה רבה מזיהוי כתב דפוס. כתב הדפוס תמיד ישר, וכל אות מכל מחשב מגופן מסויים תמיד זהה לאותה אות מכל מחשב אחר. לעומת זאת אין 2 אנשים להם אותו כתב יד ובנוסף אותו אדם יכול לכתוב אותה אות בצורה שונה, בזווית שונה, בגודל שונה אפילו באותה מילה. לכן המטרה היתה להביא למצב שבו אות מסוימת תהתיה בזהות מקסימלית מבחינת גודל, זוית, מיקום עם אותה אות במסד הנתונים של רשת הנוירונים בשביל להגדיל עד כמה שאפשר את האפשרות לזיהוי נכון. בנוסף המשתמש שרוצה ללמד את כתב היד שלו לתוכנה צריך לכתוב כל תו 10 פעמים כדי שלתוכנה יהיה מגוון גדול יותר של הופעות התו להשוואה.

חלק ראשון

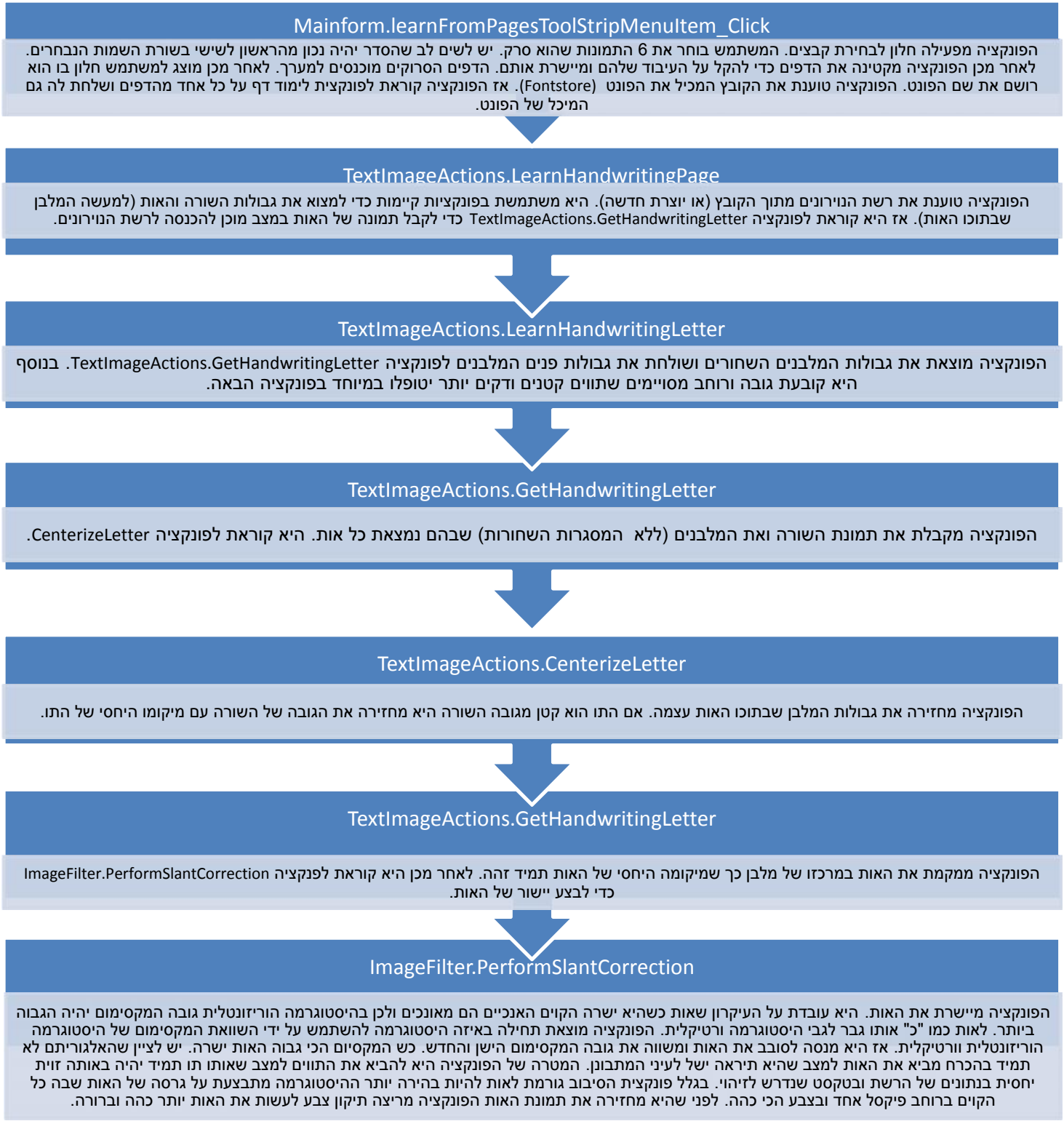
לימוד כתב היד

שלב א' הדפסת דפי קלט: בתפריט Learning->Create Handwriting Font->Print input pages.



שלב ב': לימוד הדפים: בתפריט Learning->Create Handwriting Font->Learn from pages

לפני ההפעלה יש לסרוק את הדפים הכתובים ולשמור אותם כתמונות.



TextImageActions.GetHandwritingLetter

לאחר שהופנקציה מקבלת את הגרסה המיושרת של האות היא משנה את הגודל. אם האותיות\תוים דקים הם מוחזרים מוכנסים בעובי טבעי לתוך המלבן אחרת האות מוגדלת עד לגבולות המלבן, לגודל קבוע לכל האותיות כדי שיתאים למס' הנוירונים של הרשת. לבסוף מוחזר ה"שלד" של האות כדי שהיא תוכנס נכון ל DATATSET.

TextImageActions.LearnHandwritingPage

הפונקציה שולחת את תמונת האות לפנק' הקיימת FontActions.CreateLetterData כדי לקבל את האות במצב מתאים להכנסה לרשת ואז מכניסה אותה לרשת הנוירונים. אחרי שהיא מסיימת לקלט דף שלם הפונקציה שומרת את הרשת.

שלב ג' קריאת דפים: בחר בפונט של כתב היד שלך ברשימת הפונטים, בחר סריקה של הדף שאתה רוצה לזהות והקש "Get Text".

Mainform.GetTextTask

קודם כל הפונקציה שואלת את המשתמש באיזה שפה הטקסט שהוא מבקש לזהות. זה נועד לשפר את הזיהוי ע"י הקטנת מס' התווים האפשריים. לאחר מכן היא בודקת האם הפונט הנבחר הוא של כתב יד. אם לא היא מבצעת זיהוי של כתב דפוס. זיהי כתב דפוס מתבצע ברובו בדרך שהיתה בתוכנה המקורית. הכנסנו כמה שינויים, עליהם בהמשך. אם הפונט הוא של כתב יד היא קוראת תחילה לפונקציה TextImageActions.GetHandwritingPage

TextImageActions.GetHandwritingPage

הפונקציה הזו אחראית ליישור שורות כדי להקל על זיהוי והפרדת השורות. היישור מתבצע באמצעות אנליזה של היסטוגרמות. תחילה נקראת הפונקציה TextImageActions.HistAnal

TextImageActions.HistAnal

הפונקציה יוצרת היסטוגרמה אנכית של כל הדף. בהיסטוגרמה אנכית כל איבר מכיל את סכום הפיקסלים בשורה שלמה. בצורה כזו האיברים בהיסטוגרמה המקבילים לשורות פיקסלים שמכילים שורות טקסט בדף יהיו הכי גדולים והאיברים ביניהם שלא מקבילים לשורות טקסט יהיו קטנים הרבה יותר. כלומר ה"הרים" בהיסטוגרמה הם השורות בדף. הפונקציה מתוכנתת להתעלם משורות דקות מאד ביחס לאחרות התוך ההנחה שהן תוצאה של רעש. הפונקציה מחזירה מערך שמכיל את גבולות השורות.

TextImageActions.GetHandwritingPage

הפונקציה תחילה חותכת כל שורה מתוך תמונת הדף לפי גבולות השורה שהיא קיבלה. לאחר מכן היא מוצאת באמצעות היסטוגרמה אופקית את תחילת וסוף השורה, זה על מנת לחסוך בזמן העיבוד, ודם כדי להתעלם משורות קצרות במיוחד, מתוך הנחה שגם הן תוצאה של רעש. לאחר מכן לכל עמודה נחתך קטע לגובה השורה של האזור מסביב לעמודה. זה בעצם חלון ברוחב קבוע שנע לאורך השורה. לכל עמודה מחושבת ההיסטוגרמה האנכית של החלון שמסביב לה. ההיסטוגרמה נסרקת מלמעלה למטה. בהיסטוגרמה יש הרים שמייצגים את השורה, ובעצם ה"הר" מורכב מ 2 מדרגות מכל צד. בעוד ששרוב האותיות הן בגובה דומה, חלק בולטות מעל ומתחת לשורה. כך המדרגה הראשונה היא אותם החלקים של האות שבולטים מעל ומתחת לגוף האמצעי של השורה. הגבולות האלה נשמרים בשביל לדעת איפה מתחילה ונגמרת האות. ונמדרגה המרכזית היא הגוף העיקרי של האותיות שהוא בגובהנ השורה והמרכז שלו הוא בעצם מרכז השורה. המרכז הזה שהוא באמצע הגוף המרכזי נשמר גם הוא. התוצאה היא שנוצר קו שעוקב אחרי האמצע המדויק של כל השורה. הקו הזה עובר עיבוד כגי לשטח ולעדן אותו ולמנוע קפיצות ומדרגות שיכולות לעוות את האותיות. לאחר מכן השורה ממופה על קו אמצע חדש, ישר, כך שכל עמודה בעצם מונחת על הקו החדש בצורה שנקודת האמלע של העמודה נמצאת על הקו החדש. כך נוצרת מחדש כל שורה מיושרת והן מודברות אחת אחרי השניה ליצור דף חדש עם שורות מיושרטת. התהליך הזה גורם לעיוותים והפרעות בצורת חלק מהתווים לכן התוצאה עוברת בפונקציית TextImageActions.fix כדי לתקן אותן הפרעות.

Mainform.GetTextTask

הדף חוזר לפונקציה זו עם השורות מיושרות. אז הדף נשלח לפנקציהImageFilter.PerformHWThreshold

ImageFilter.PerformThreshold

הפונקציה הופכת את הדף לשחור.לבן בלבד. בגלל שבסריקה של כתב יד האותיות יותר בהירות ממה שיוצא מסריקה של טקסט מודפס יש לכתב יד threshold נפרדת משלה.

Mainform.GetTextTask

הדף נשלח בשלב הבא לפונקציה TextImageActions.GetAllLettersBounds שיוצרת מערך של הגבולות של כל האותיות והשורות. הפונקציה הזו פועלת אותו דבר גם על כתב דפוס וגם על כתב יד.

TextImageActions.RetrieveText

הפונקציה הזו יוצרת את התמונות של כל האותיות שאחרי זה ישלחו לזיהוי. קודם כל מחושב ממוצע הגובה והרוחב של האותיות שישמש לתווים דקים במיוחד, ולתווים קטנים מגובה השורה. לאחר מכן הגבולות של כל אות נשלחים לפונקציה שיוצרת את התמונה של אותה אות מתוך השורה. כאן שוב כתב יד ודפוס נפרדים. כתב יד נשלח לפונקציה TextImageActions.GetHandwritingLetter.

TextImageActions.GetHandwritingLetter

דרך פעולתה של פונקציה זו כבר הוסברה בחלק על לימוד הפונט. ההבדל היחיד שהפעם היא משתמשת בממוצעים שחושבו בפונקציה הקודמת ליצירת תמונות טובות של תווים דקים\קטנים במיוחד.

מגבלות התוכנה

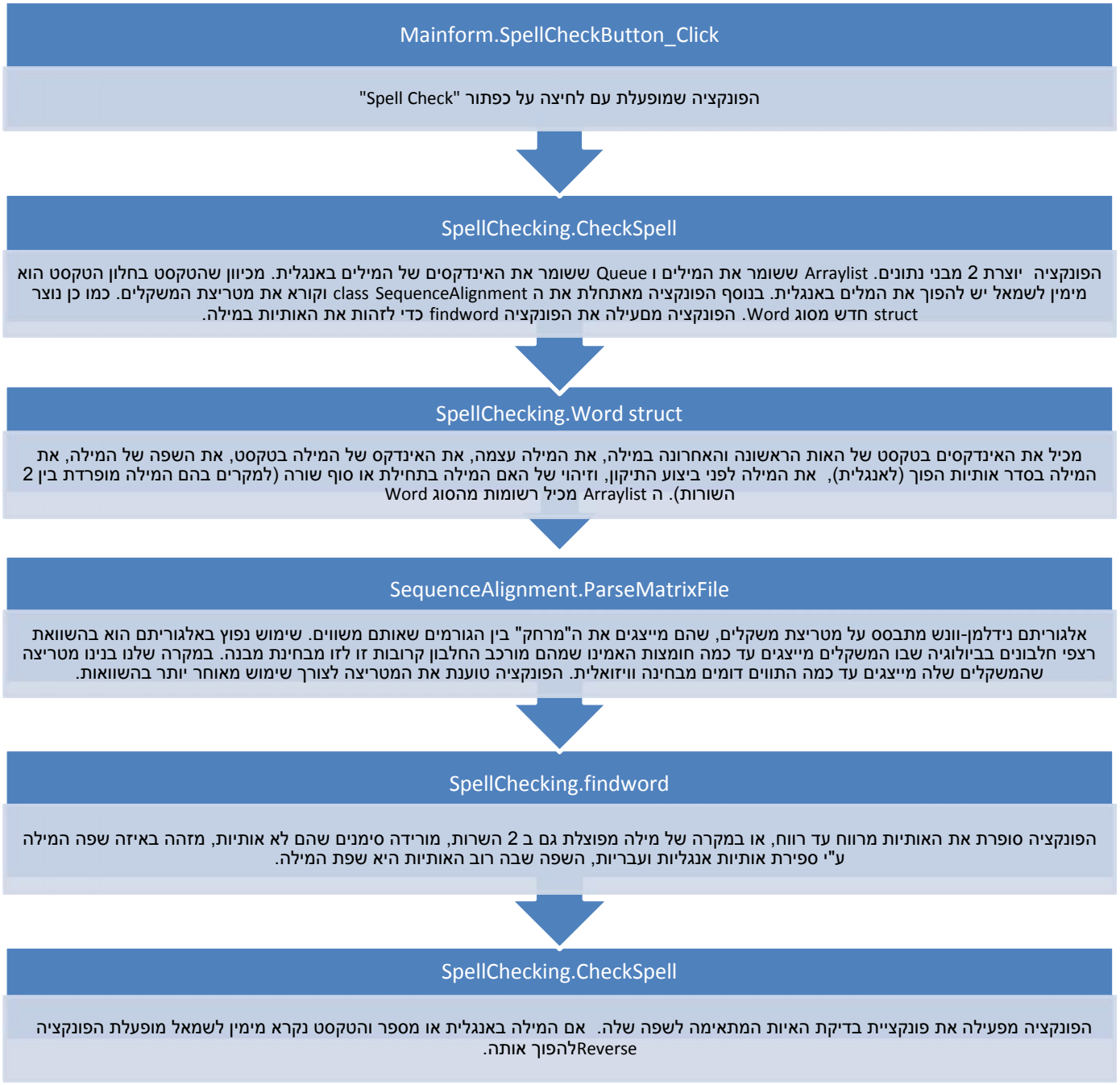
בשביל זיהוי אופטימאלי ממולץ למשתמש לפעול ע"פ כללים מסויימים.

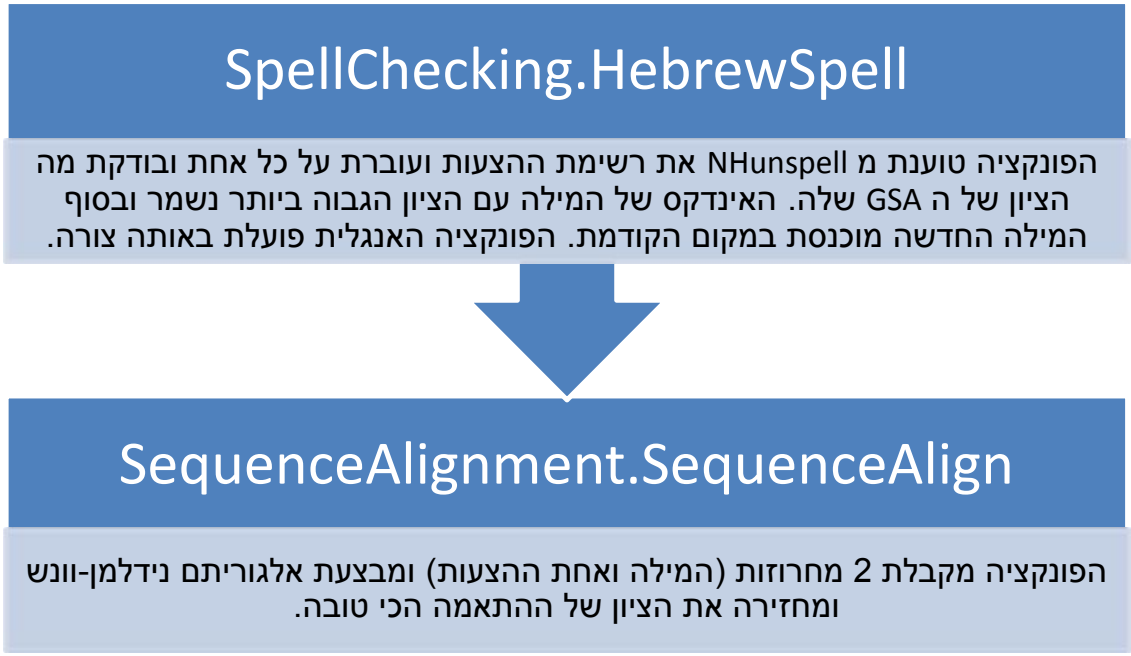
- הדף שמקבלת התוכנה צריך להיות ישר במידה סבירה. התוכנה יודעת לסובב את הדף לזוית של 0°, אבל דף שמסובב ב 90° או או 180° היא לא תוכל לזהות נכון.
- בכתיבה בכתב יד יש לדאוג שהשורות יהיו ישרות ומופרדות במידה סבירה. התוכנה אמורה ליישר שורות שלא ישרות לגמרי אך שורות שנכתבו בזויות גדולות או כשהשורה בברור מזו שמעליה התוכנה תתקשה.
- האותיות בכתב יד חייבות להיות מופרדות לחלוטין. בין אות לאות חייב להיות רווח ברור אפילו אם רק בעובי פיקסל אחד, אחרת התוכנה לא תוכל להפריד ביניהן.

בדיקת איות

הסבר: כדי לשפר את דיוק זיהוי הטקסט ניתן לבצע בדיקת איות לאחר הזיהוי. בשביל לבצע בדיקת איות יש ללחוץ על הכפתור " Spell Check" שמופיע לאחר השלמת הזיהוי. בדיקת האיות מתבססת על מילון מותאם ל net. שנקרא NHunspell (אוצר המילים העברי מבוסס על HSpell). המילון מציע כמה הצעות למלים מתאימות. מציאת המילה המתאימה ביותר מתבצעת באמצעות אלגוריתם נידלמן-וונש (Global sequence alignment). ה Class שאחראי על ביצוע האלגוריתם מבוסס על הקוד של `Kunuk Nykjaer`.

פירוט דרך הפעולה:





הקובץ שמכיל את מטריצת המשקלים מצורף לפרויקט בתיקיית Nhunspell. אפשר לערוך אותו ולשנות את המשקלים מה שיגרום לשינוי בבחירת המלים של התוכנה. בשביל להמיר אותו לקובץ טקסט מתאים יש לשמור אותו בצורת Unicode text ובשם VisualSimilarity.txt באותה תיקייה.

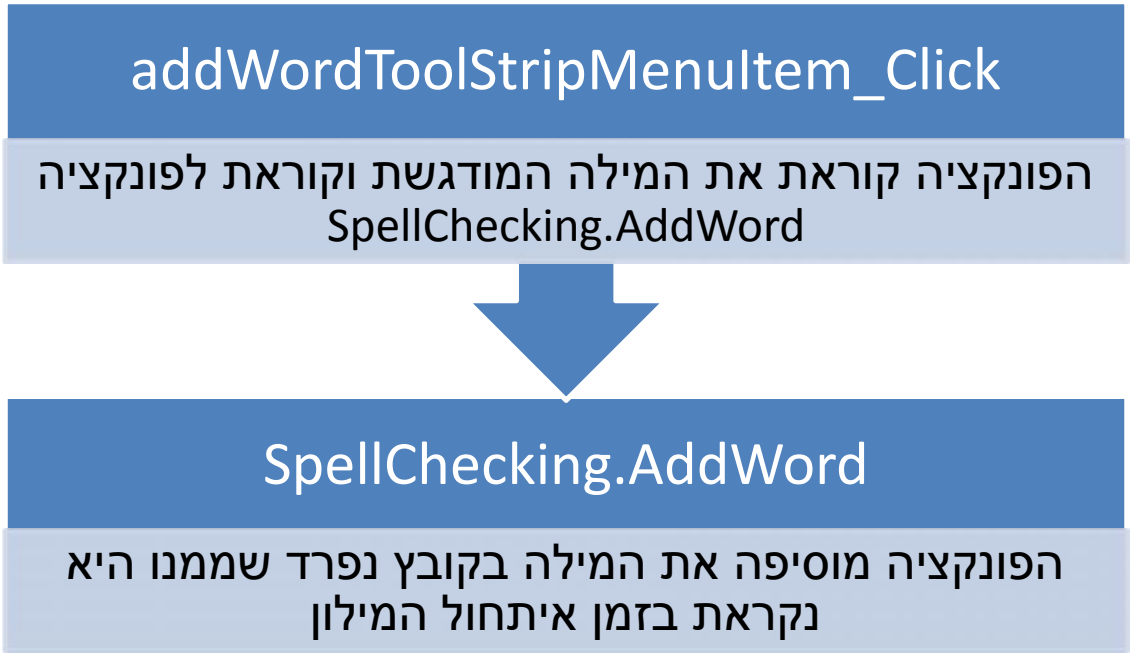
כמו כן מצורפים קבצי המילון וקובץ manual של Hunspell בו יש הסברים לשינוי קבצי המילון.

הערה: התוכנה יכולה לתקן רק מלים עם אחוז זיהוי גבוה. אם אחוז הזיהוי של הטקסט נמוך עדיף לא להשתמש במילון אלא קודם לתקן טעויות ידנית ולאמן את הרשת.

פרטים נוספים:

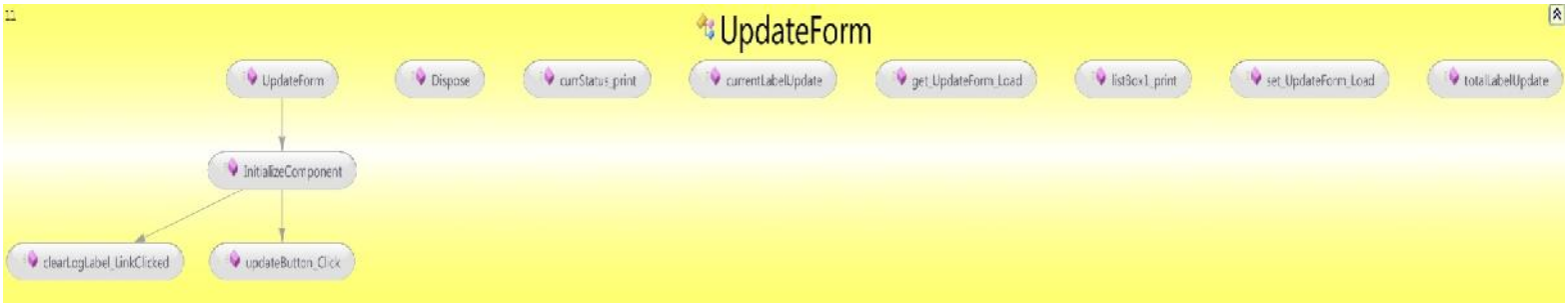
- http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm
- http://kunuk.wordpress.com/2010/10/17/dynamic-programming-example-with-c-using-needleman-wunsch-/algorithm
- http://nhunspell.sourceforge.net
- http://www.codeproject.com/Articles/33658/NHunspell-Hunspell-for-the-NET-platform

הוספת מילים למילון: אין אפשרות פשוטה להוסיף מילים למילונים הקיימים. במקום זה המילים שהמשתמש מוסיף נשמרות בקובץ נפרד ונטענות לתוך המילון בזמן איתחול המילונים. בשביל לשמור מילה לתוך מילון יש לסמן את המילה בחלון הטקסט, ללחוץ על כפתור עכבר ימני ולבחור "Add to Dictionary".



UpdateForm

Used for update session of program.



Methods:

UpdateForm: Initialize components

listBox1_print: Receive string as a parameter and print it in listBox1 and in the serverlog.txt file with adding date and time stamp.

clearLogLabel_LinkClicked: clear all elements in listBox1.

updateButton_Click: Start update session. Creates new Client object and calls ClientUpdate function.

currentLabelUpdate: Receive string as a parameter and print it in current_lab(change label).

totalLabelUpdate: Receive string as a parameter and print it in total_lab (change label).

ServerSettingsForm

Used for change update settings.

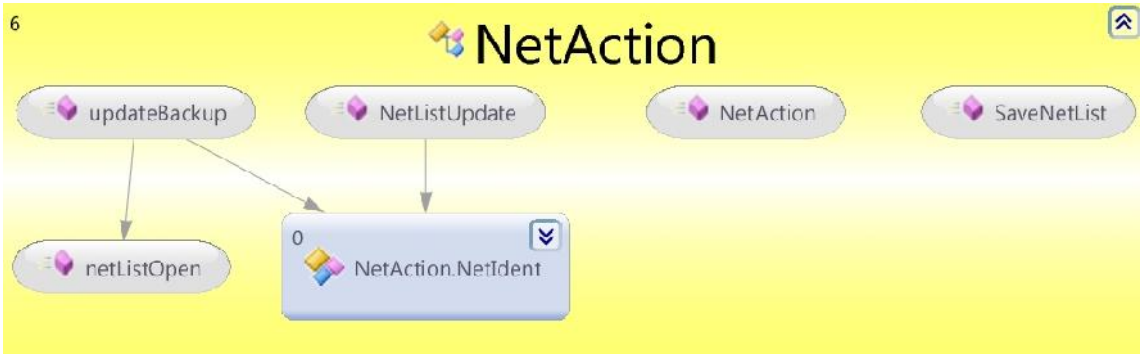
Methods:

ServerSettingsForm(): Constructor, initialize components, calls Client.openServerSettings to open saved server settings file if exist, and print it in appropriate textboxs.

button1_Click: checks if entered information is correct, if not opens Message Box, returns to ServerSettingsForm. If correct calls Client.updateServerSettings to save this settings in serversettings.txt file.

NetAction

Used for make networks lists and actions on it.



`public struct NetIdent`: Used as element of networks list, that contains basic information about font network, (version, NetName, size or data size).

Methods:

NetListUpdate: Receive string as a parameter. This is path which contains fonts networks file. Opens all file in this folder with extension .net and create `List<NetIdent>` of all this files. Return this list.

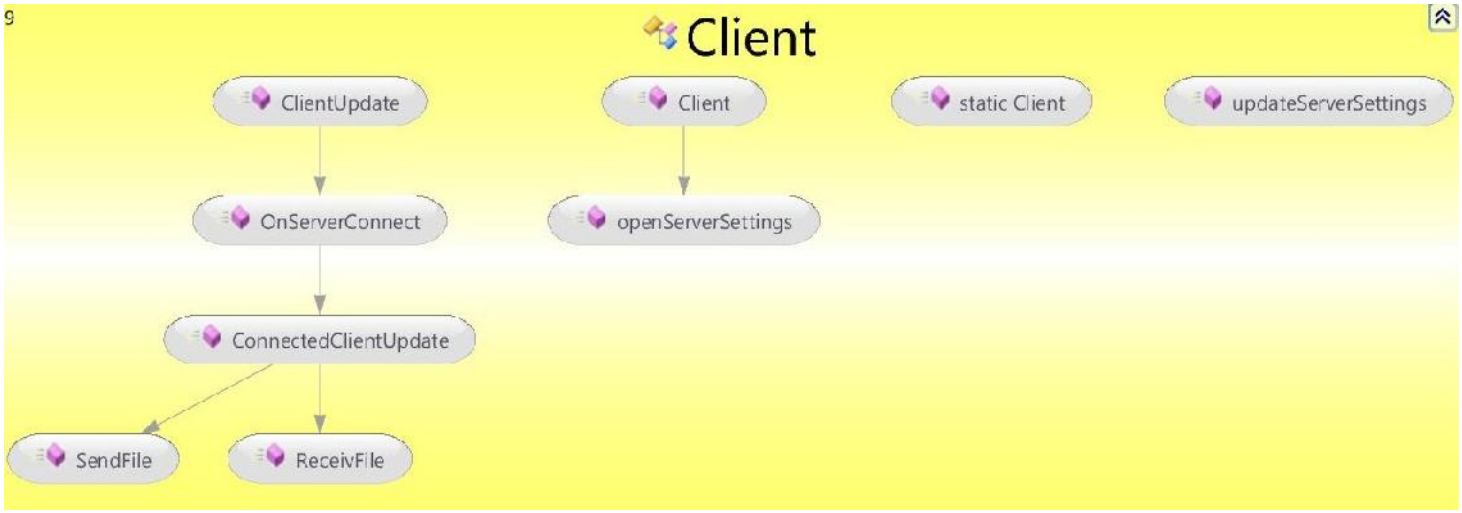
SaveNetList: Receive as a parameters strings: filename (name of file for save this List), saveTo (path to save this file in) and `List<NetIdent>` the network list to save. Save this network list as XML.

netListOpen: Receive string as a parameter the full path of network list file to open and return this list as `List<NetIdent>` object. If file does not exist, create empty networks list and return it.

updateBackup: Receive as a parameters string fullfileName (the networks list that need to be backuped) and string sourcePath (Folder with font networks files).
If not exist creates `Updates_Backup` the folder for backup files. Copy all files of network which exist in received networks list.

Client

Main class for update session.



Properties:

`public static string serverHost;` - host to make update with

`public static int port;` - port on this host

`Socket clientSock;` - socket on which client runs update session

`int ReceiveTimeout = 10000;` - Set the timeout for synchronous receive methods, to 1 second (10 milliseconds.)

`int SendTimeout = 10000;` - Set the timeout for synchronous sent methods, to 1 second (10 milliseconds.)

`public const string ServerFolderPath = @"Server\";` - folder to save service files

`public const string ClientFolder = FontActions.FontsFolderPath;` - folder which contain fonts networks files.

`public static byte[] successful = BitConverter.GetBytes(1);` - variable to compare report of sent routine in byte[] format

`public static byte[] unsuccessful = BitConverter.GetBytes(0);` - variable to compare report of sent routine in byte[] format

`public const int maxfilelentgh = 10240000;` - max file length (default 10 megabyte)

`public UpdateForm callForm;` - UpdateForm object which created this Client object. To make available to print output in this form (use it`s print function).

Methods:

Client: Receive UpdateForm as a parameter, save it in object parameter. Open server settings from file, and save this parameters in corresponding variables.

ClientUpdate: - Creates IPEndPoint and Socket from client variables, updates socket parameters and start Asynchronous connection on this socket.

OnServerConnect: - This is the call back function, which will be invoked when a server is connected. Call ConnectedClientUpdate routine.

ConnectedClientUpdate: Receive opened socket as parameter. Creates fonts networks lists to work with it in this update session. Calls NetListUpdate function for update current networks list. Sent it to server with SendFile routine. Receive from server lists of networks to be updated in this session. Sent and receive the files. Counts this files and prints report.

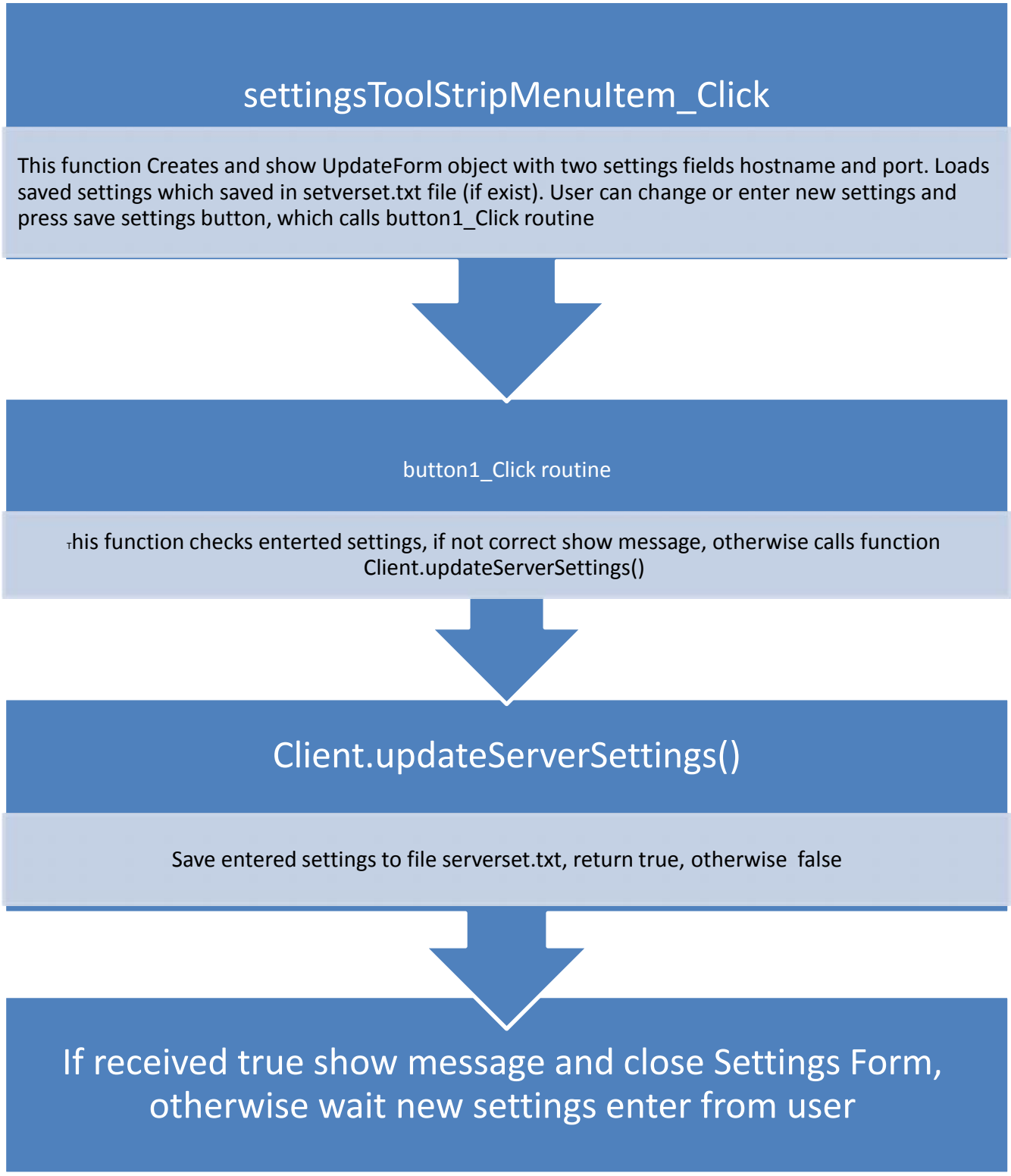
SendFile: Receives as parameter fullfileName (full path of file to sant), clientSock (opened socket), CallForm (form to print outputs in it).

ReceivFile: clientSock (opened socket to receive file from), saveTo (path to save file in it), CallForm (form to print outputs in it).

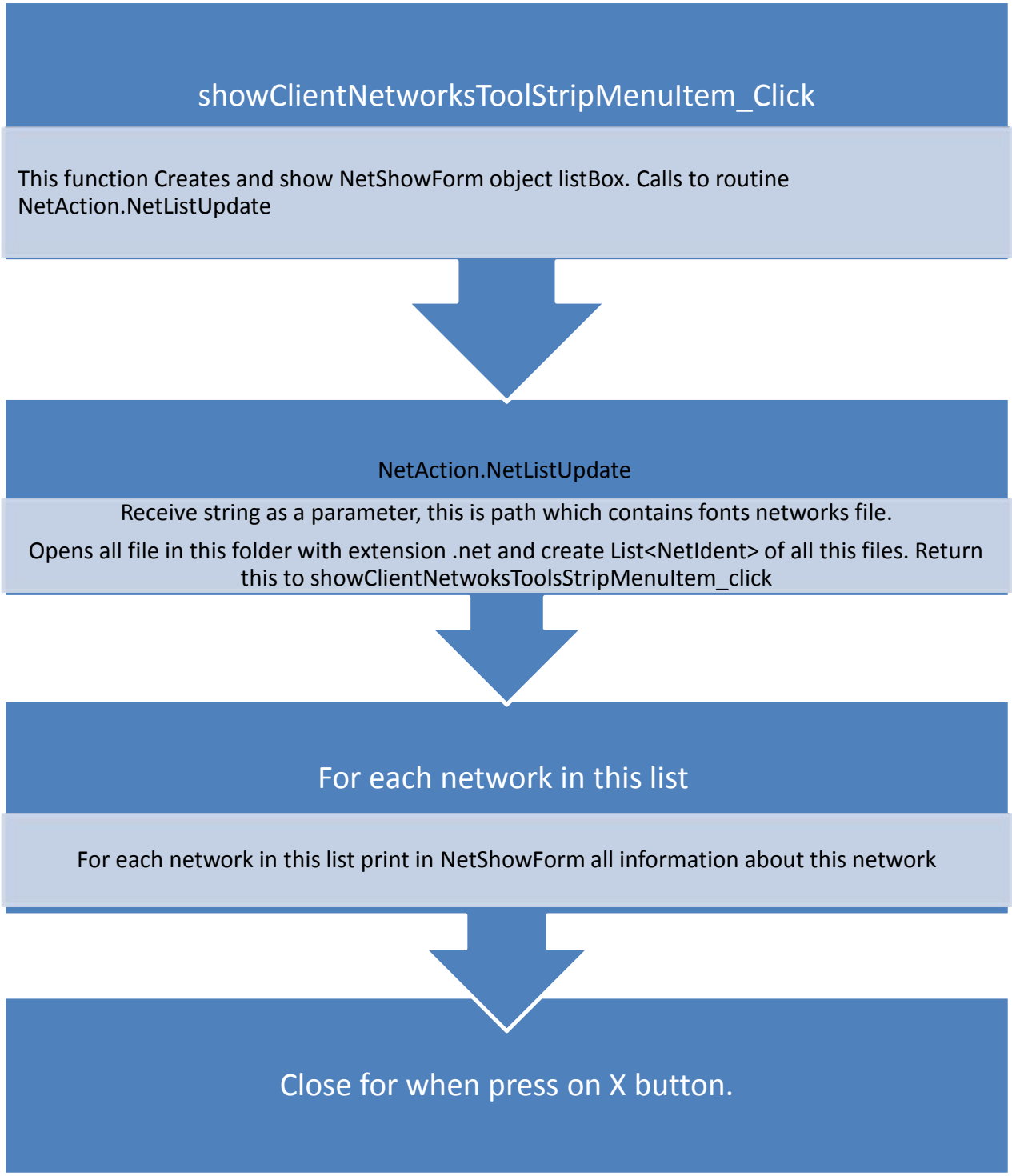
updateServerSettings: calls from ServerSettingsForm to save server settings.

openServerSettings: Opens server setting (server host and port) from serverset.txt

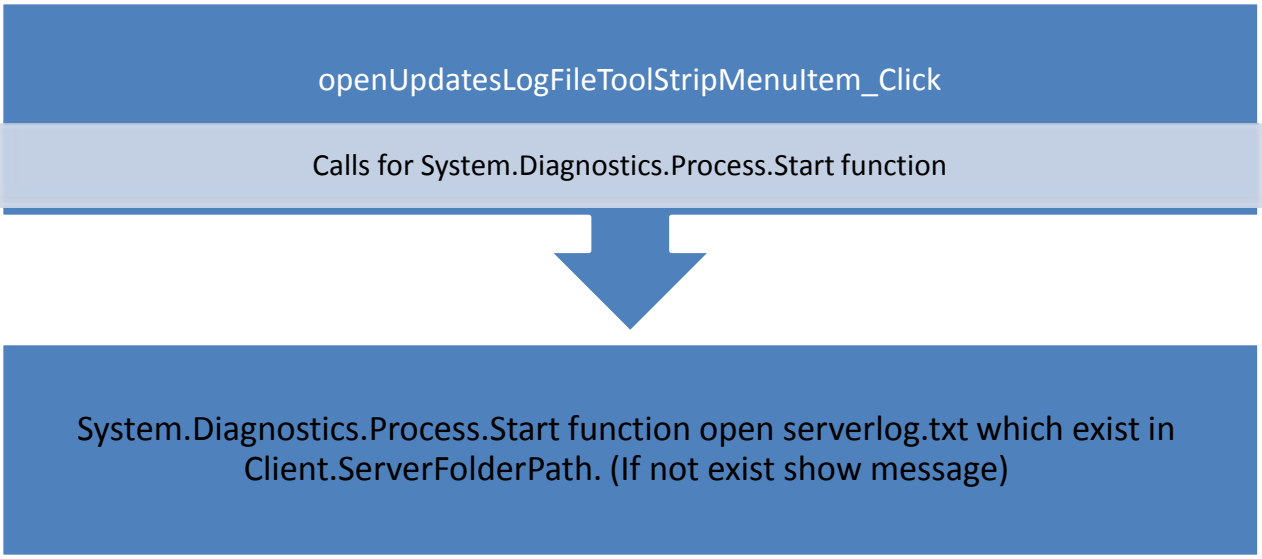
Set settings for update process: from menu Server->Settings



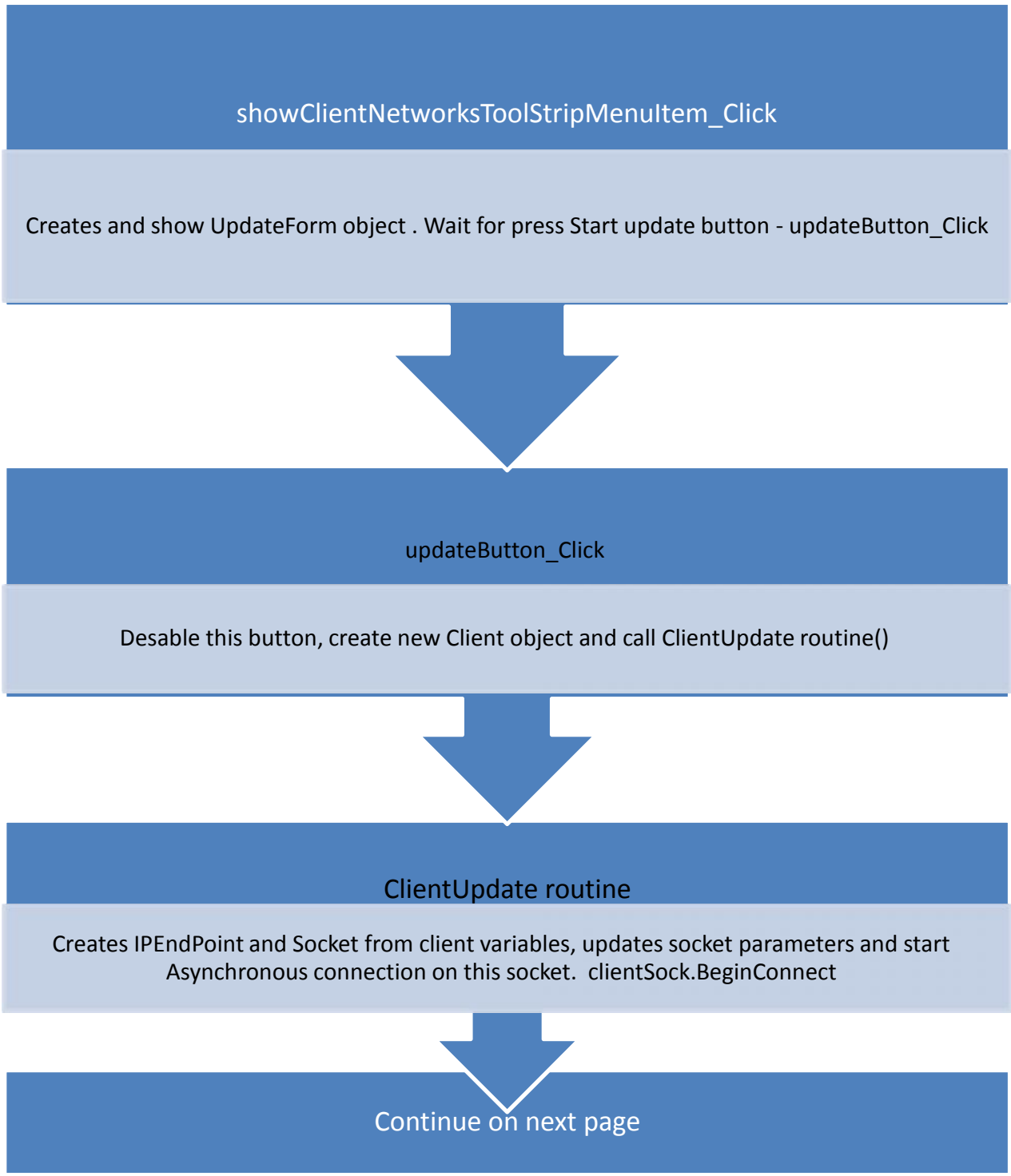
Check current client`s networks: from menu Server-> Show client networks

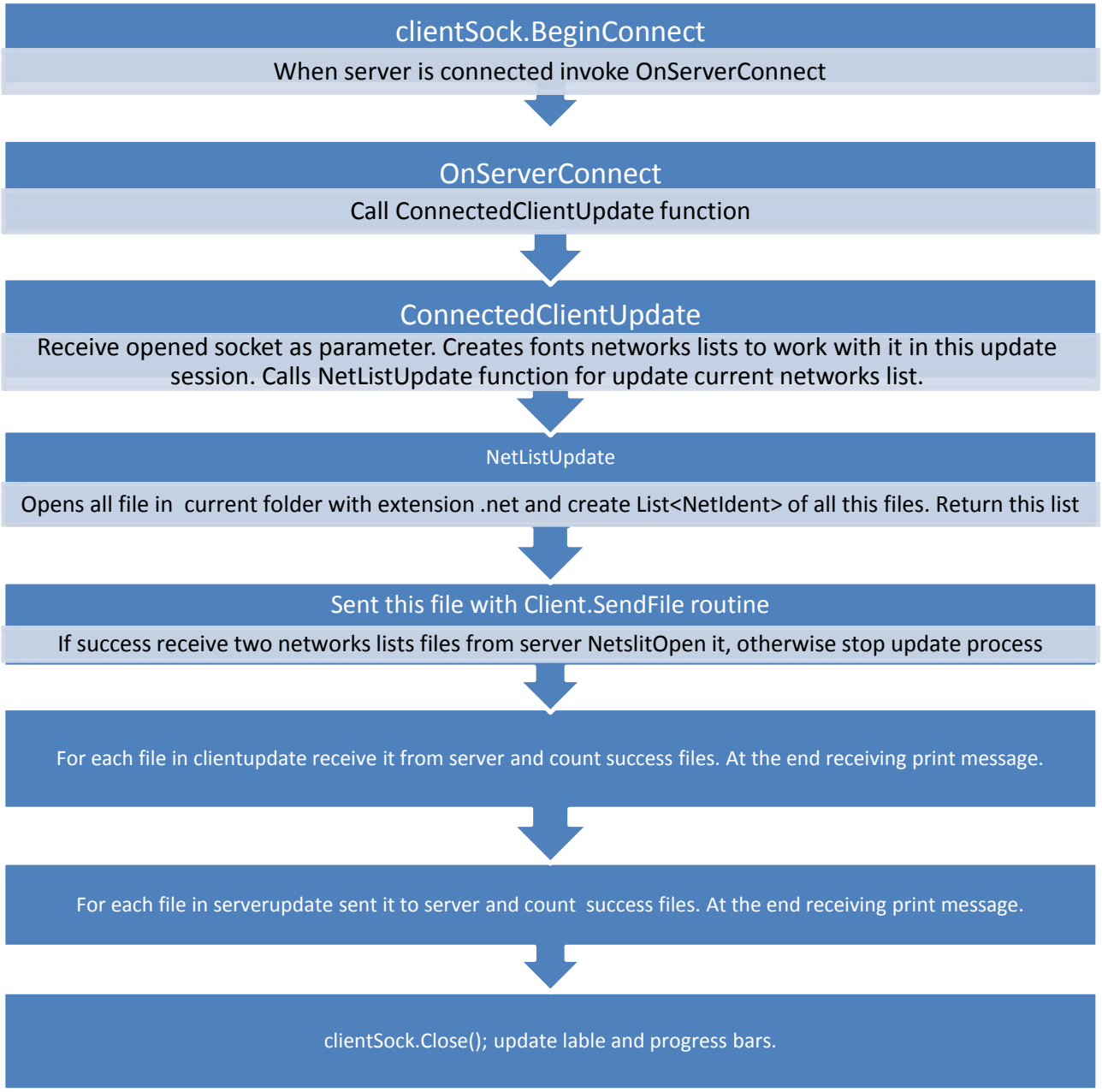


Open updates log file in default program for txt files: from menu Server->Open updates log file



Start update session: from main menu Update Fonts

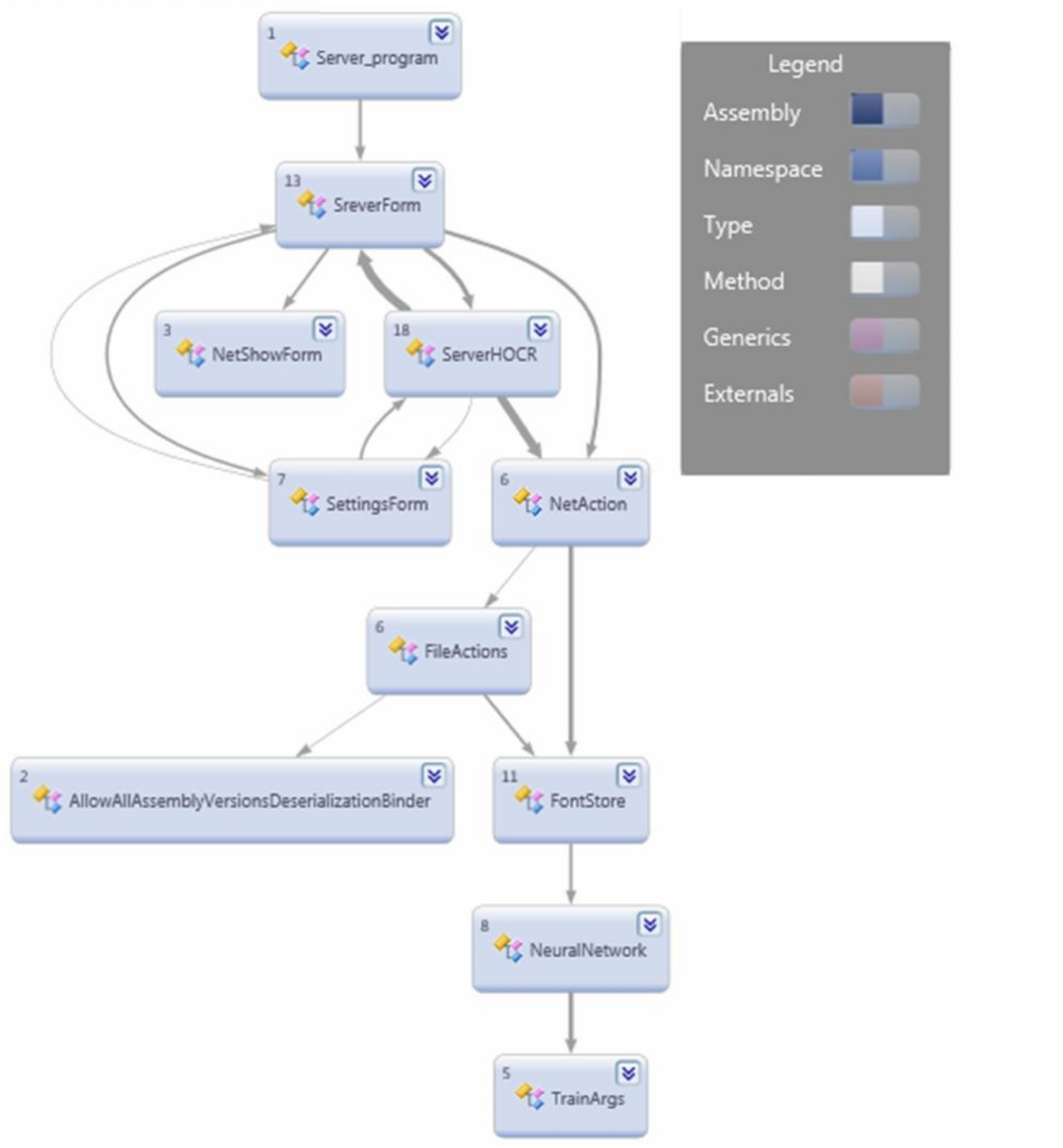



clientSock.BeginConnect
When server is connected invoke OnServerConnect
OnServerConnect
Call ConnectedClientUpdate function
ConnectedClientUpdate
Receive opened socket as parameter. Creates fonts networks lists to work with it in this update session. Calls NetListUpdate function for update current networks list.
NetListUpdate
Opens all file in current folder with extension .net and create List<NetIdent> of all this files. Return this list
Sent this file with Client.SendFile routine
If success receive two networks lists files from server NetslitOpen it, otherwise stop update process
For each file in clientupdate receive it from server and count success files. At the end receiving print message.
For each file in serverupdate sent it to server and count success files. At the end receiving print message.
clientSock.Close(); update lable and progress bars.

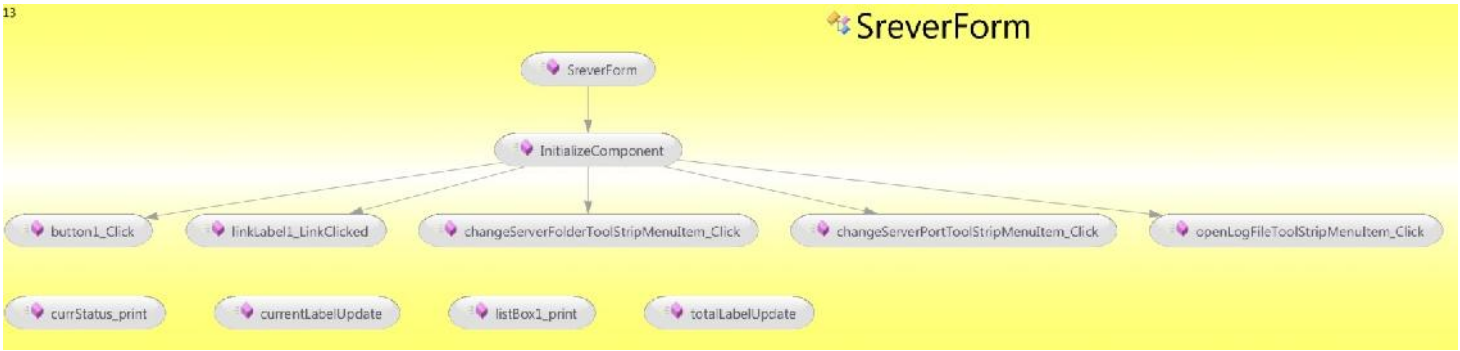
Server Program

Server is support program for HOCR program, which allow updating fonts network files.

מבנה כללי של התוכנה



ServerForm



Properties:

`public ServerHOCR formsServer; - ServerHOCR object which will be created in constructor of this form`

Methods:

listBox1_print: Receive string as a parameter and print it in listBox1 and in the serverlog.txt file with adding date and time stamp.

currStatus_print: Receive string as a parameter and print it in currStatus textbox (use fot show current status).

currentLabelUpdate: Receive string as a parameter and print it in current_lab(change label).

totalLabelUpdate: Receive string as a parameter and print it in total_lab (change label).

button1_Click: Change name of button to “Stop server), calls to StartServer routine of formsServer object (to start server).

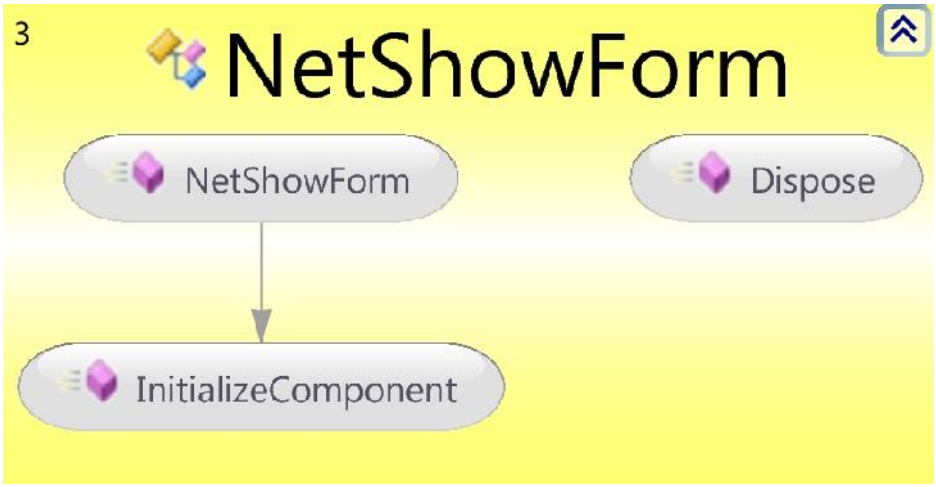
linkLabel1_LinkClicked: Clear all text in listBox1.

changeServerFolderToolStripMenuItem_Click: Creates NetShowForm (listbox window for print output existing networks). Checks which fonts networks exists in Client Folder and prints this list with properties of this fonts.

openLogFileToolStripMenuItem_Click: Opens serverlog.txt log file what exist in ServerHOCR.ServerDefaultFolder with default program for txt files.

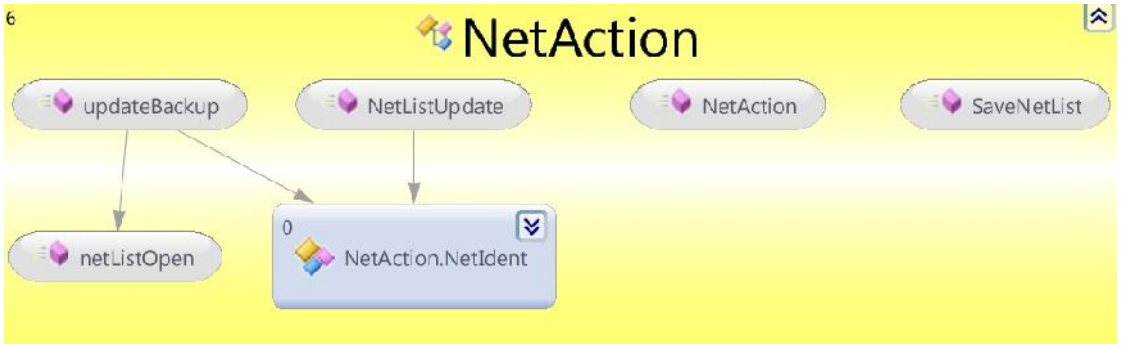
changeServerPortToolStripMenuItem_Click: Creates new form SettingsForm and show it.

NetShowForm



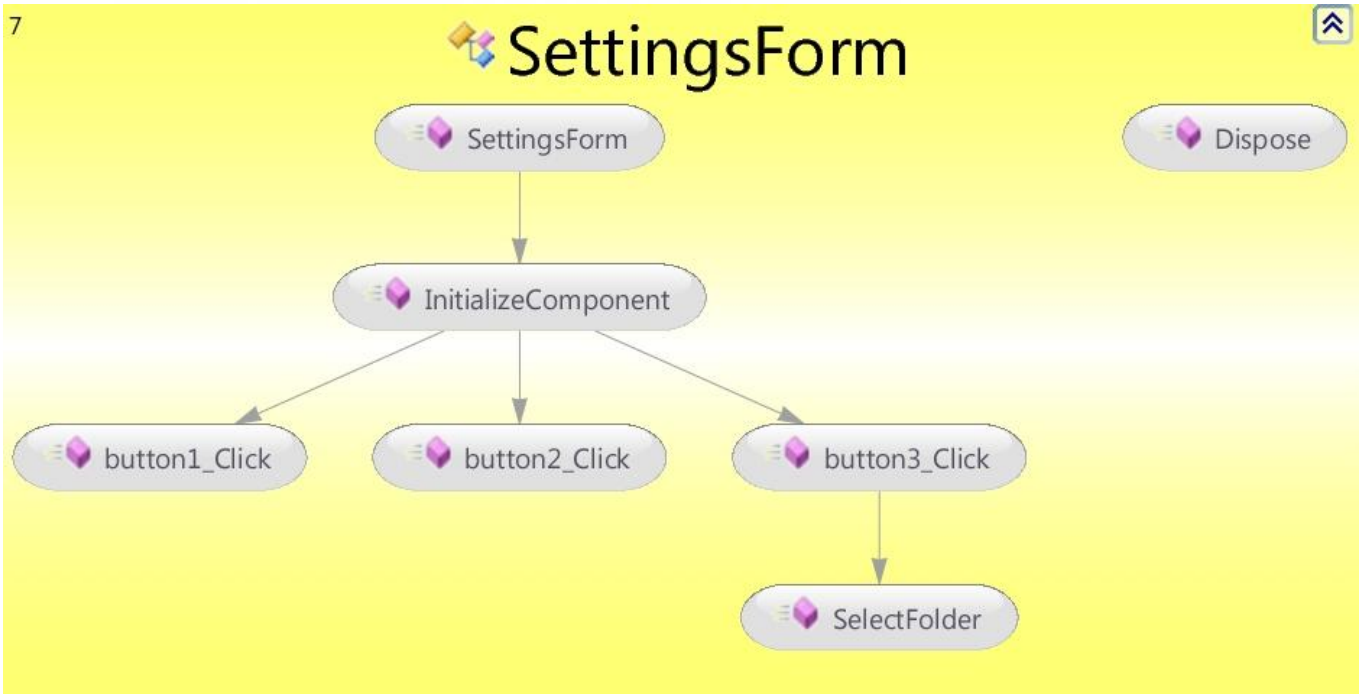
This for has only one listBox to show output information about existing networks on server.

NetAction



Identical to NetAction in HOCR program.

SettingsForm



Properties:

public SreverForm callForm; - SreverForm object which created this Client object. To make available to print output in this form (use it`s print function).

Methods:

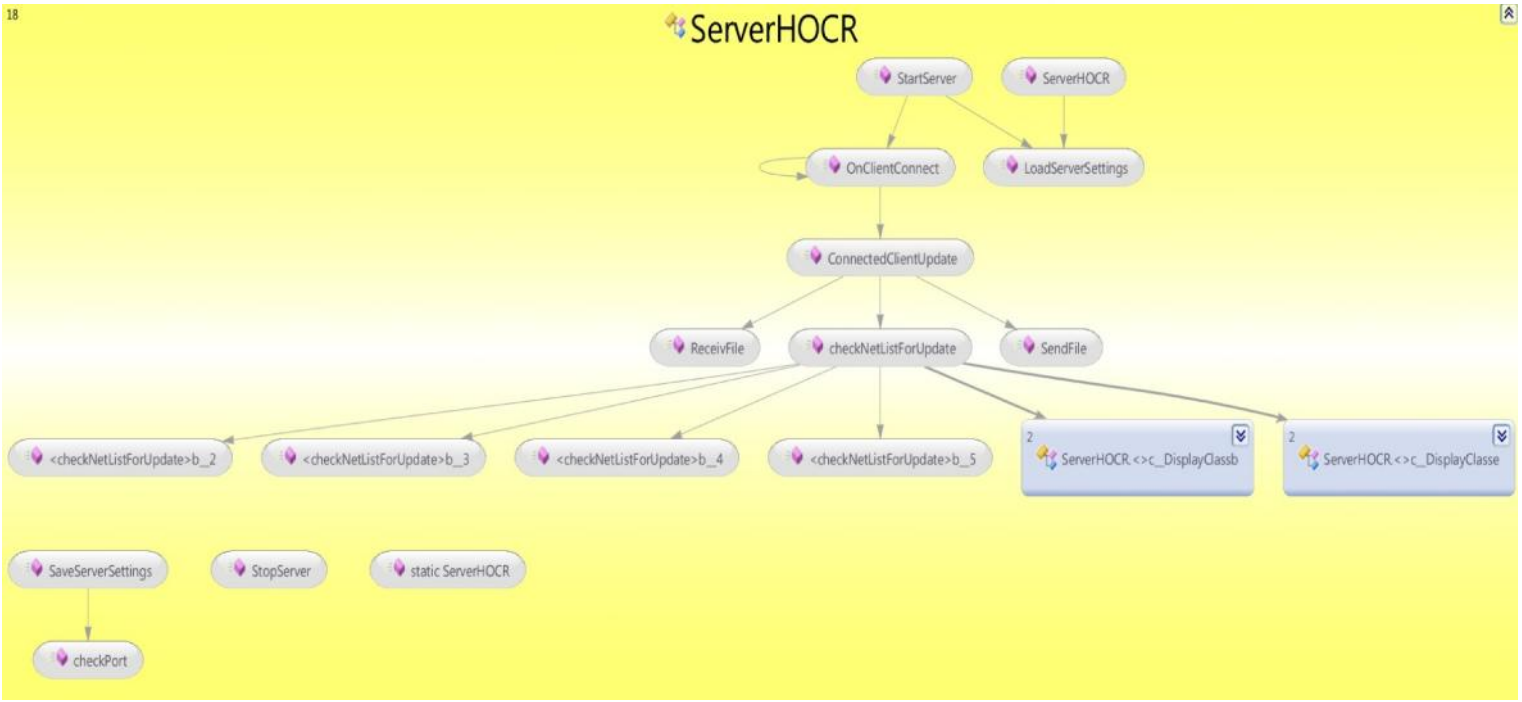
SettingsForm: Constructor of this form. Initialize components. Try to open server_settings.txt from ServerHOCR.ServerDefaultFoldel and prints it`s information to textboxes (server port; server folder; max file size). If file does not exist or there is a problem to open it, load default settings from ServerHOCR class.

button1_Click: Calls for ServerHOCR.SaveServerSettings routine which saves these settings.

button2_Click: Load defaults button click which load default settings from ServerHOCR class.

button3_Click and SelectFolder: Open Browse standard window to choose folder, and after it show this result in textBoxt2 (server folder).

ServerHOCR



Properties:

```
public const string ServerDefaultFolder = @"Server_files\"; - Default server folder
public static string ServerFolder = ServerDefaultFolder; - Folder of all servers files (except serversettings.txt file)

IPEndPoint ipEnd; - object`s variable.

public const int defaultPort = 5656; - Default port
public static int port = defaultPort; - object`s  port variable
Socket sock; - object socket variable
public int ReceiveTimeout = 10000; - Set the timeout for synchronous receive methods, to 1 second (10 milliseconds.)

public int SendTimeout = 10000; - Set the timeout for synchronous sent methods, to 1 second (10 milliseconds.)

int clientCount = 0; - count of connected clients
public static byte[] successful = BitConverter.GetBytes(1); - variable to compare report of sent routine in byte[] format
public static byte[] unsuccessful = BitConverter.GetBytes(0); - variable to compare report of sent routine in byte[] format

public const int defaultmaxfilelentgh = 10240000;  - default max file length 10 megabyte
public static int maxfilelentgh = defaultmaxfilelentgh; - object`s max file length variable.
public HOCR.SreverForm CallForm: - SreverForm object which created this Client object. To make available to print output in this form (use it`s print function).
```

Methods:

ServerHOCR: Receive SreverForm as a parameter, save it in object parameter. Calls to LoadServerSettings routine.

LoadServerSettings: Open server settings from file, and save this parameters in corresponding variables.

SaveServerSettings: checks if entered information is correct, if not opens Message Box, returns to ServerSettingsForm. If correct, save this settings in serversettings.txt file, and print this information in log windows.

checkPort: - Receive integer port value as a parameter, check if this port is already in used by another program or process.

StartServer: - Creat new Socket object, updates variables and parameters, start listening for connection on opened socket.

OnClientConnect: - This is the call back function, which will be invoked when a client is connected. Calls ConnectedClientUpdate routine. After return from this function, call sock.BeginAccept, for wait for other connections.

ConnectedClientUpdate: - Print information about connected client. And start update process.

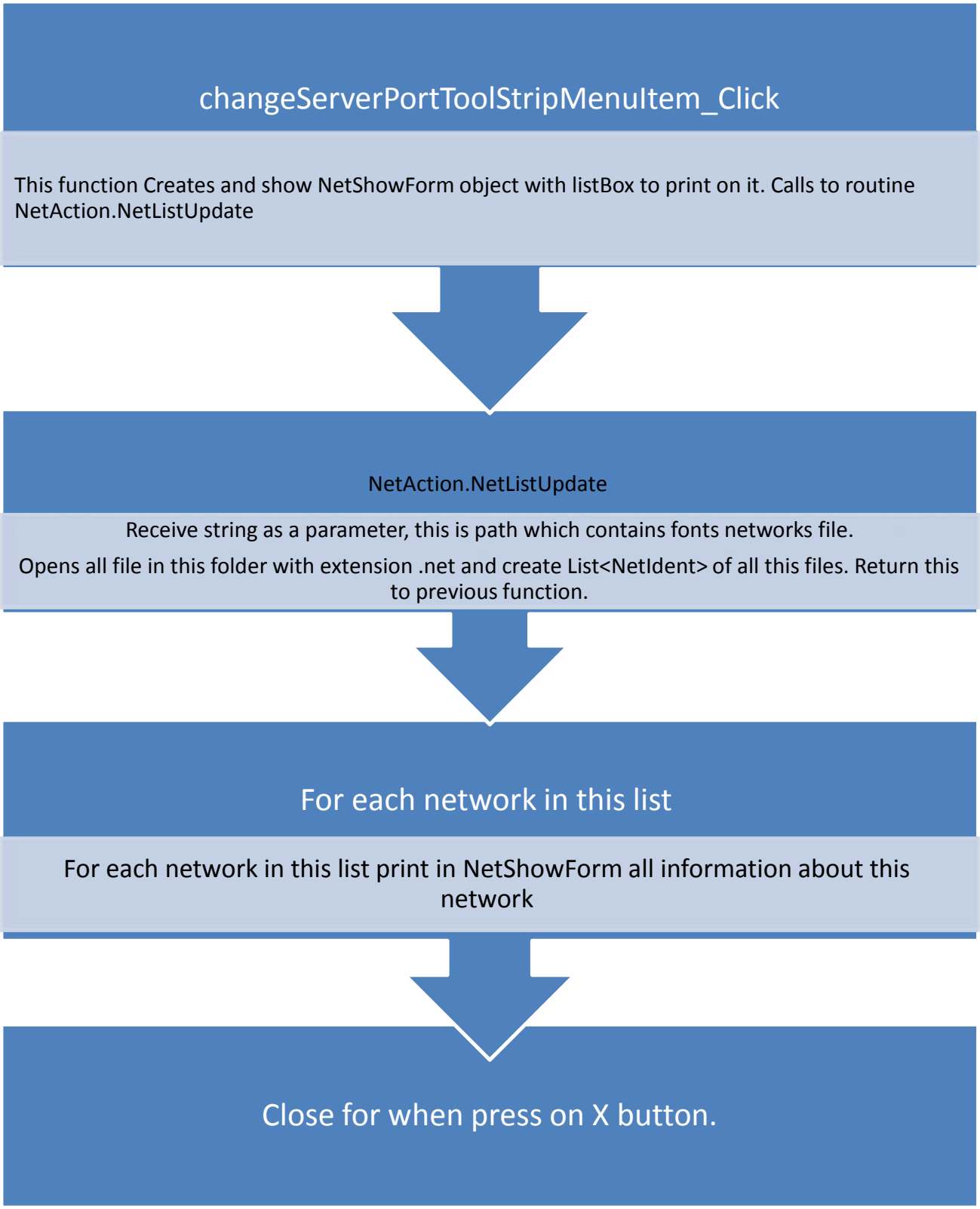
SendFile: - identical to SendFile routine in HOCR program. (copy of this function)

ReceivFile: identical to ReceivFile routine in HOCR program. (copy of this function)

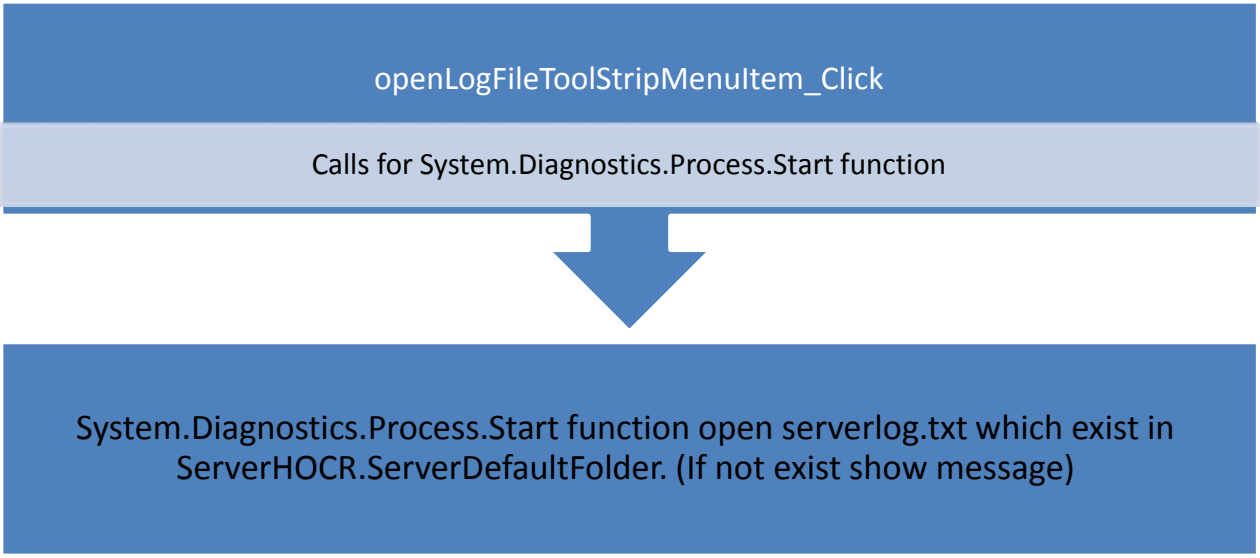
checkNetListForUpdate: - open netlist, which was received from client. Update servernetlist files by NetAction.NetListUpdate function. Checks which networks in netlist does not exist in server, and enter them to serverupdate list, and same process with networks in servernetlist if not exist add to clientupdate list. After it checks if there are new versions of networks or there is bigger data size of network, and enter them to corresponding list (serverupdate or clientupdate). Save this list in files by calling NetAction.SaveNetList routine.

StopServer: - Close server socket, print messages about it.

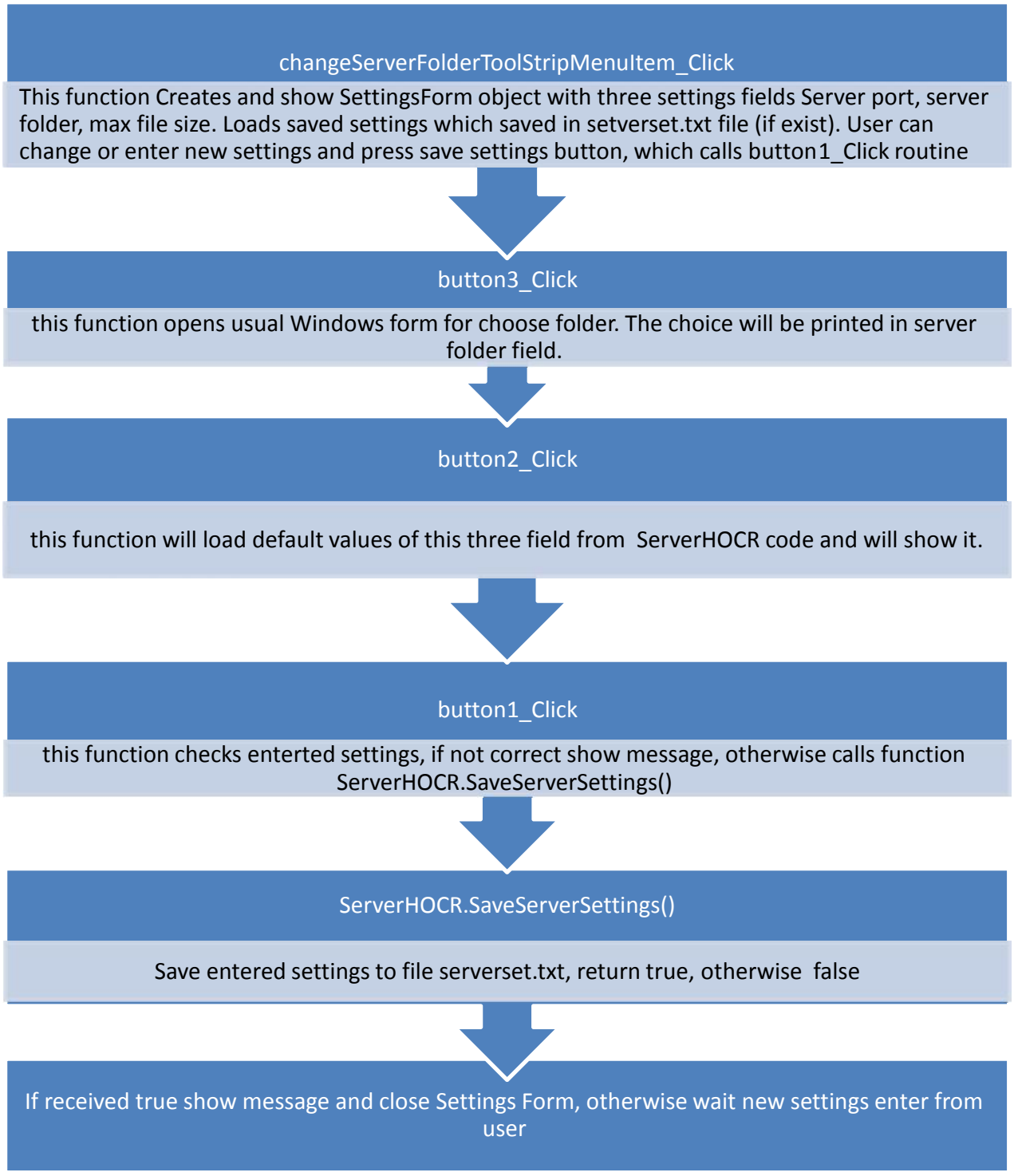
Check current server`s networks: from menu Tools-> Show client networks



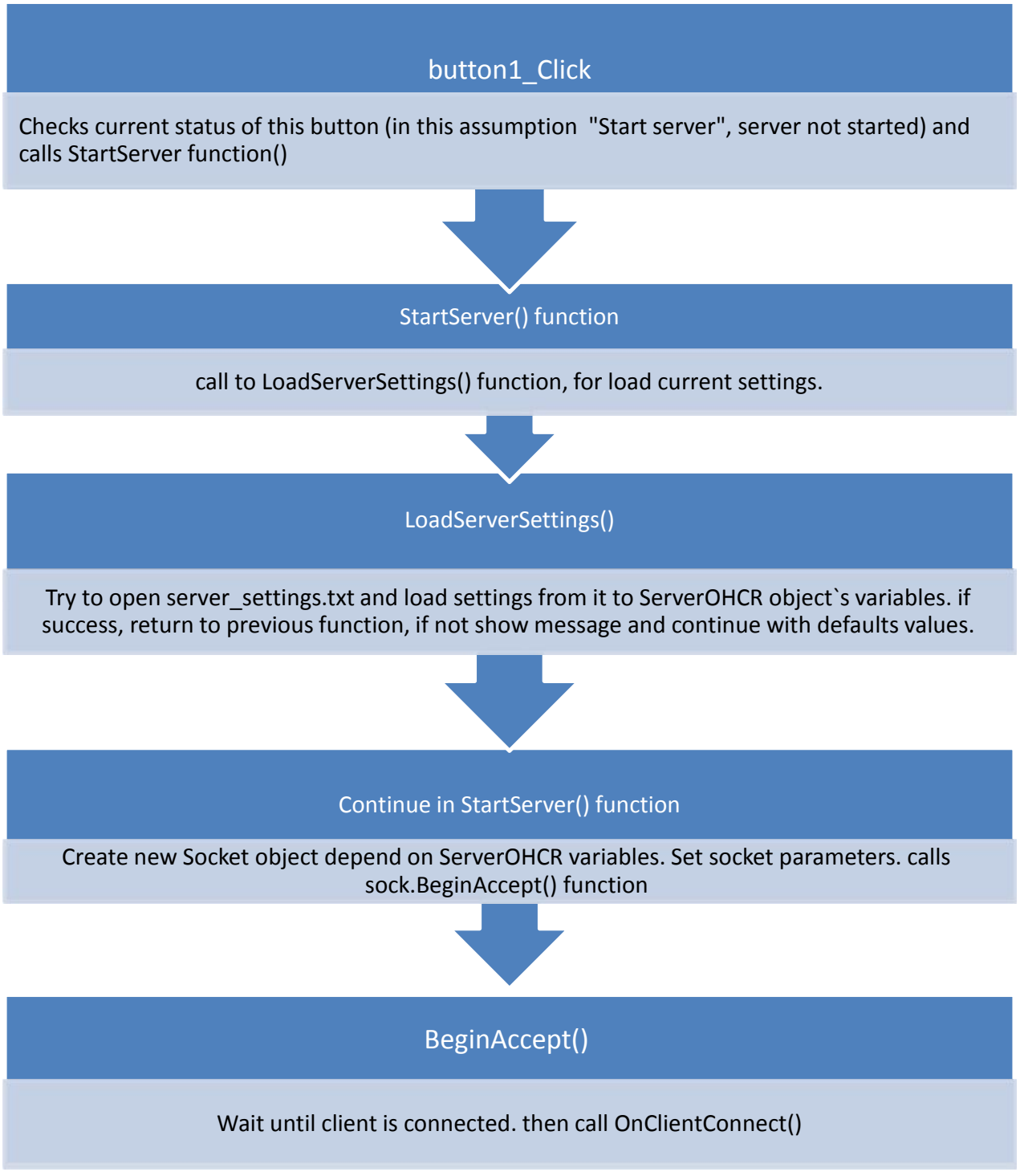
Open server`s log file in default program for txt files in Windows: from menu -> Open updates log file



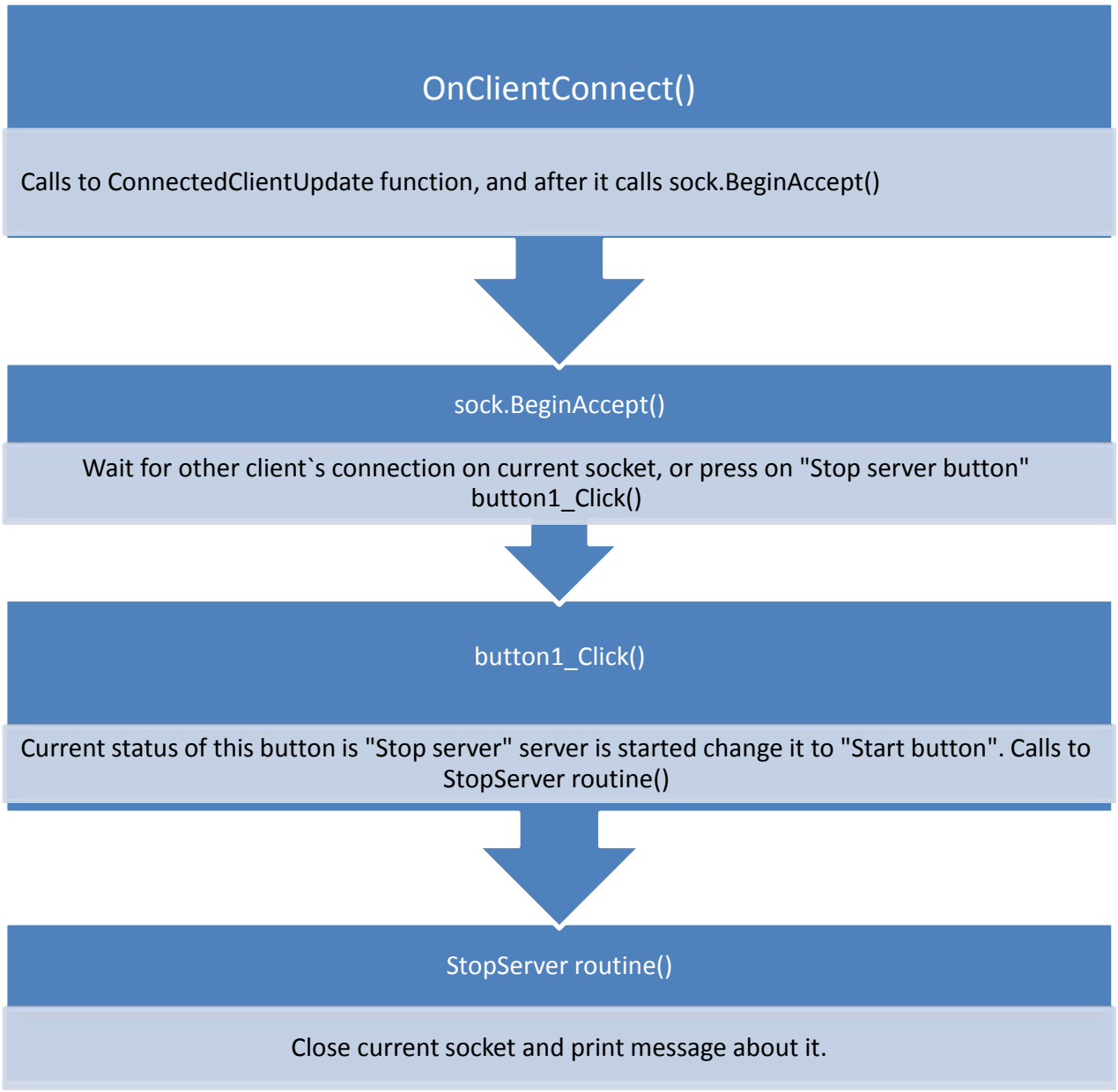
Set or change server settings: from menu ->Settings



Start server: press on Start server button



This is the call back function, which will be invoked when a client is connected:



ConnectedClientUpdate routine call on from OnClientConnect function which will be invoked when a client is connected