COMPONENTS AND INTERFACES OF TORNREPAIR

Tan Wang

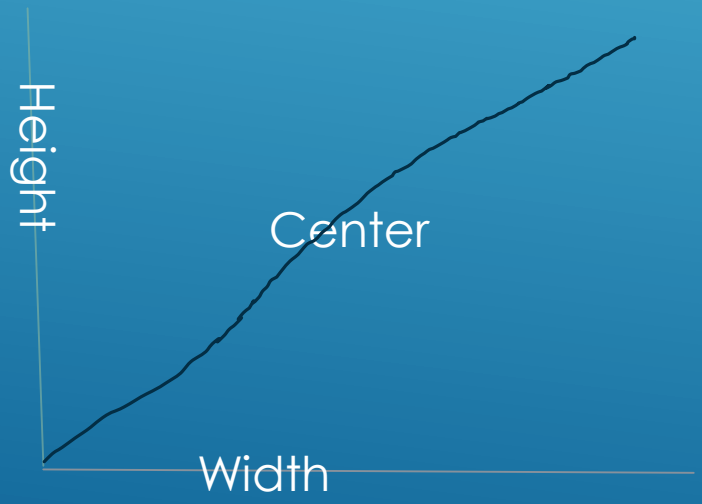
DATA STRUCTURES OVERVIEW

- ColorfulContourMap : ContourMap for the edge map
- ColorfulPoint for the point representation
- Match for the matching parameters
- Phi for the extracted edge features, including turning angles and color on the edge, a list of this forms the edge feature data structure for a single image fragment
- ReturnImg for returned image
- Image<T,byte>, provided by Emgu CV, for temporary image storage
- When matching, there is a queue data structure for all the intermediate fragments
- All of the data structures are session specific. If one session ends all the data contained will be deleted

CONTOURMAP

- new public List<ColorfulPoint> _points; // all the points on the contour
- new public List<ColorfulPoint> _polyPoints; // the points that form the approximated polygon of the contour map
- public int Width { get; internal set; } // see picture below

  public int Height { get; internal set; } // see picture below
- public Point Center { get; internal set; } // see picture below



MATCH

- public int t11, t12;//starting and ending arc length parameter for piece1
- public int t21, t22;//starting and ending arc length parameter for piece2
- public int x11, y11, x12, y12; //starting and ending coordinates for piece1
- public int x21, y21, x22, y22; //starting and ending coordinates for piece2
- public double confidence; // confidence level, the metrics used to get the best match

PHI

- public double x; // x position on the image
- public double y; // y position on the image
- public double theta; // turning angle
- public int l; // arc length
- public Bgr color; // edge color

RETURNIMAGE

- public Image<Bgr, byte> img; // the returned image
- public Image<Bgr, byte> img_mask; // not displayed to user, useful for further matches
- public Image<Bgr, Byte> source1; // source image
- public Image<Bgr, Byte> source2; // source image
- public bool returnbool; // true if it is a successful match

EXTERNAL APIS

- Emgu CV: Image processing, also used for camera input and OCR
- ColorMine: Color analysis
- NetOffice: Writing text output into MS word
- Facebook SDK: Uploading photo output into Facebook
- IronPDF: Generate PDF file from HTML

MAJOR ALGORITHMS

- Three major algorithms for finding matching parameters:
  - Edge feature : used on images with a large number of polygons
  - Color feature : used on images with a small number of polygons
  - Character feature with the help from OCR : used on text extraction
- Transformation algorithm : transform the image fragments according to the parameters of the best match
- Extraction of edge color : extract the color on the edge according to the color near the edge
- Filling colors on gaps between image fragments

PSEUDOCODE DESIGN

INPUT MANAGER

- Prompt user to input an image file
- If the input is not an image file, prompt user for another file
- Scan the input
- While the input does not meet requirements, prompt user again
- Get the input and send to input file storage

CONTOUR MAP FETCHER

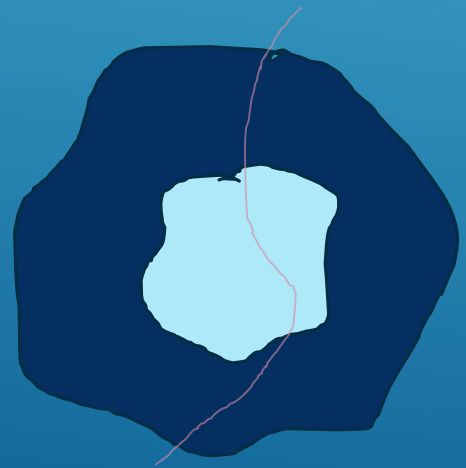
- Fetch all of the contours of the input image that has perimeter over 10px
- Check if the contour is inside a larger contour, if yes, discard that
- Check if the contour share pixels with the edge of the image, if yes, discard that
- For all valid contours, extract their edge feature, including the turning angles and color on the edge
- Send the contours and edge feature into the matching queue

CORE MATCHING ALGORITHM

- Check how many polygon points in the edge feature
- For all other pieces in the queue
  - If the algorithm found out that the color of the edge are completely the same for two pieces, it will discard that piece
  - If there are more than 20 (need to test) polygons, the algorithm will find the most possible matching edge by differences of turning angles
  - Else, if there are more than ¼ of the edge pixels have matching edge color with respect to the other piece, use color matching algorithm
- Try to match this piece to the best possible other pieces, if none is found, push this piece into the bottom of the queue and start the second one
- If there is a match, the algorithm will do the transformations, including the image and contour maps
- If one of the matching algorithm success, the algorithm will push the matched image into the top of the queue, and fetch the third piece.
- Do the previous things until there are no matchings for all the pieces, including intermediate pieces.
- After all matching is done, push each final piece into next stage. Each piece is considered one image file for output
- If this is document repair, each final piece will be OCRed for text outputs. Each piece will generate one page of text output

GAP PIXEL FILLING

- If the distance of the matching edges of two pieces are closer than 10px, it will just draw the color of the edge.
- If the distance is greater than 10px but less than 50px, it will do linear interpolation to fill the colors.
- If the distance is greater than 50px, it will not do anything because the original image might be separated at that point. See the picture below, the white place is hollowed out



OUTPUT MANAGER

- If user choose to output image files, it will prompt user to enter the directory and name of output file. User can also set the configuration of output too (similar to export as JPG in Photoshop)
- If user choose to output a PDF or HTML, it will form a list of outputs and put the album cover at the first item. It will send all the pictures into an HTML file. After the HTML file is generated, it will send the HTML file into IronPDF for the output PDF file.
- If user choose to output to MS Word, all the process in HTML/PDF generation will be done, and it will send all the items in the list to NetOffice
- If user choose to output to Facebook, all the files will be send to Facebook API.
- The manager will wait for response for above actions, and prompt user whether it is successful or not

UI PREVIEW

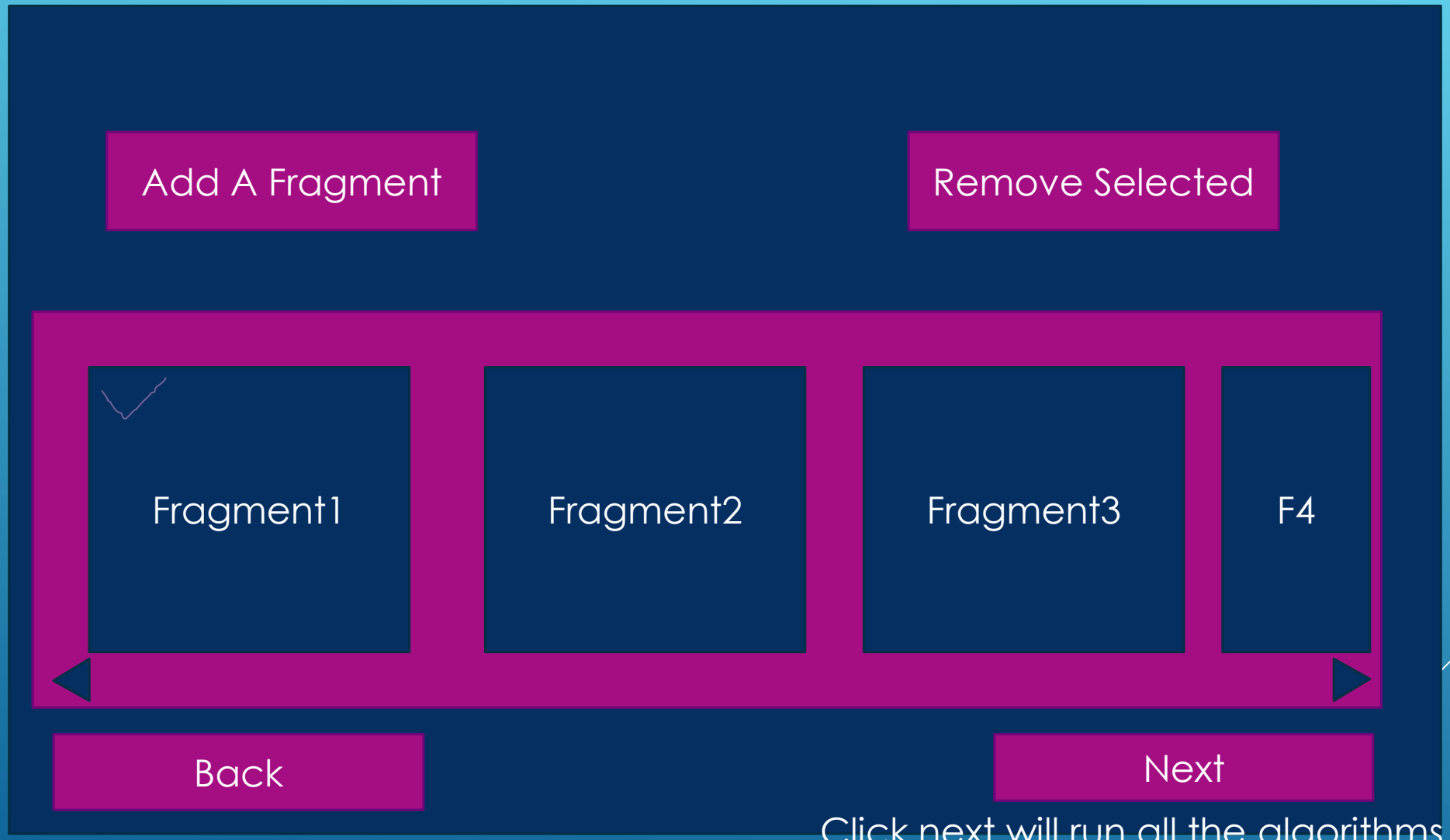
- Only contains the UI I will develop, some UI are provided by either external APIs or C# library
- The UI here is not drawn to scale, and the size here is not the actual size in the software
- The color used in here is not the color used in the software

TornRepair

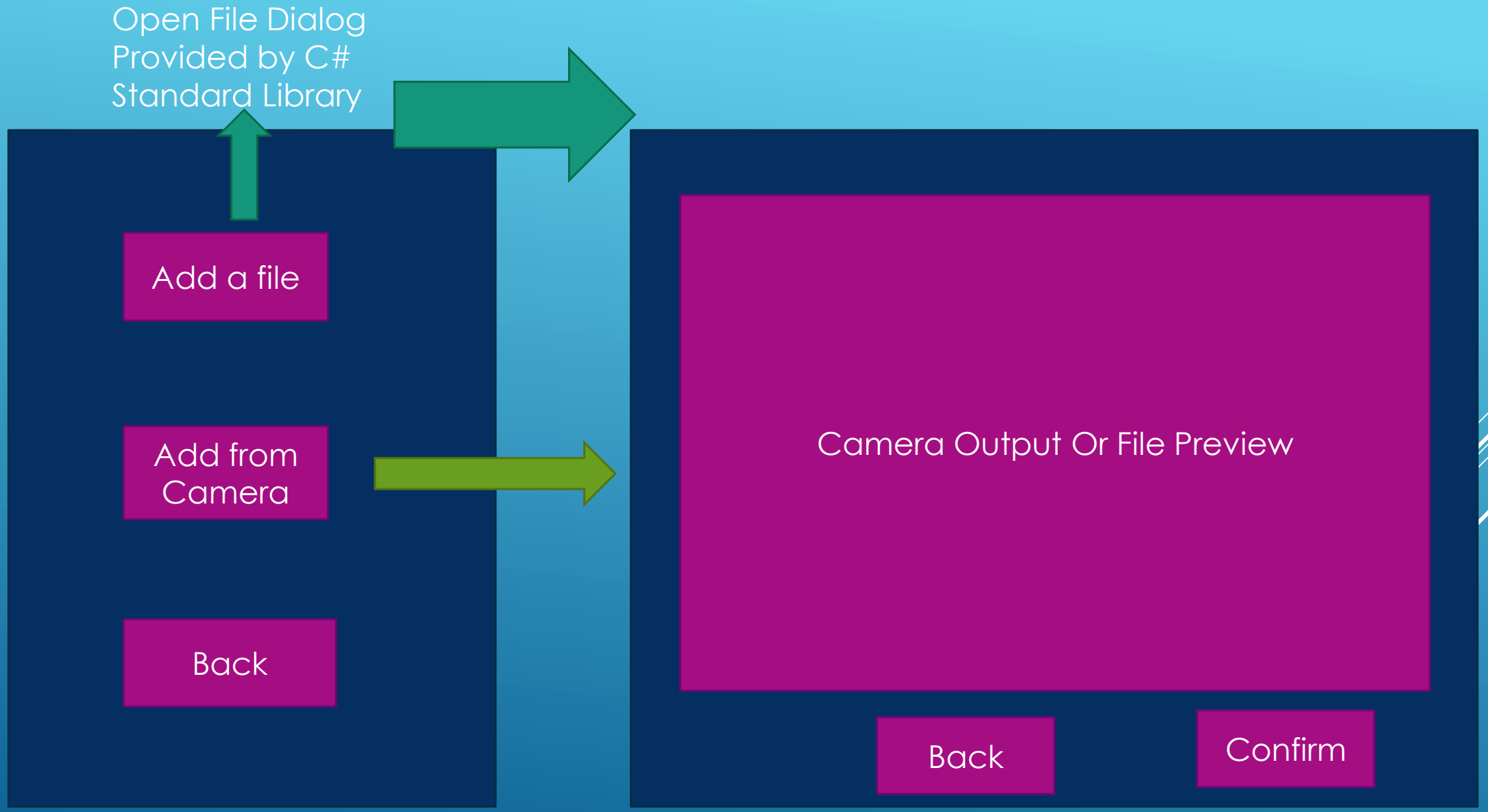
Repair Torn Photo

Repair Torn Document

Closing this window will end the current session

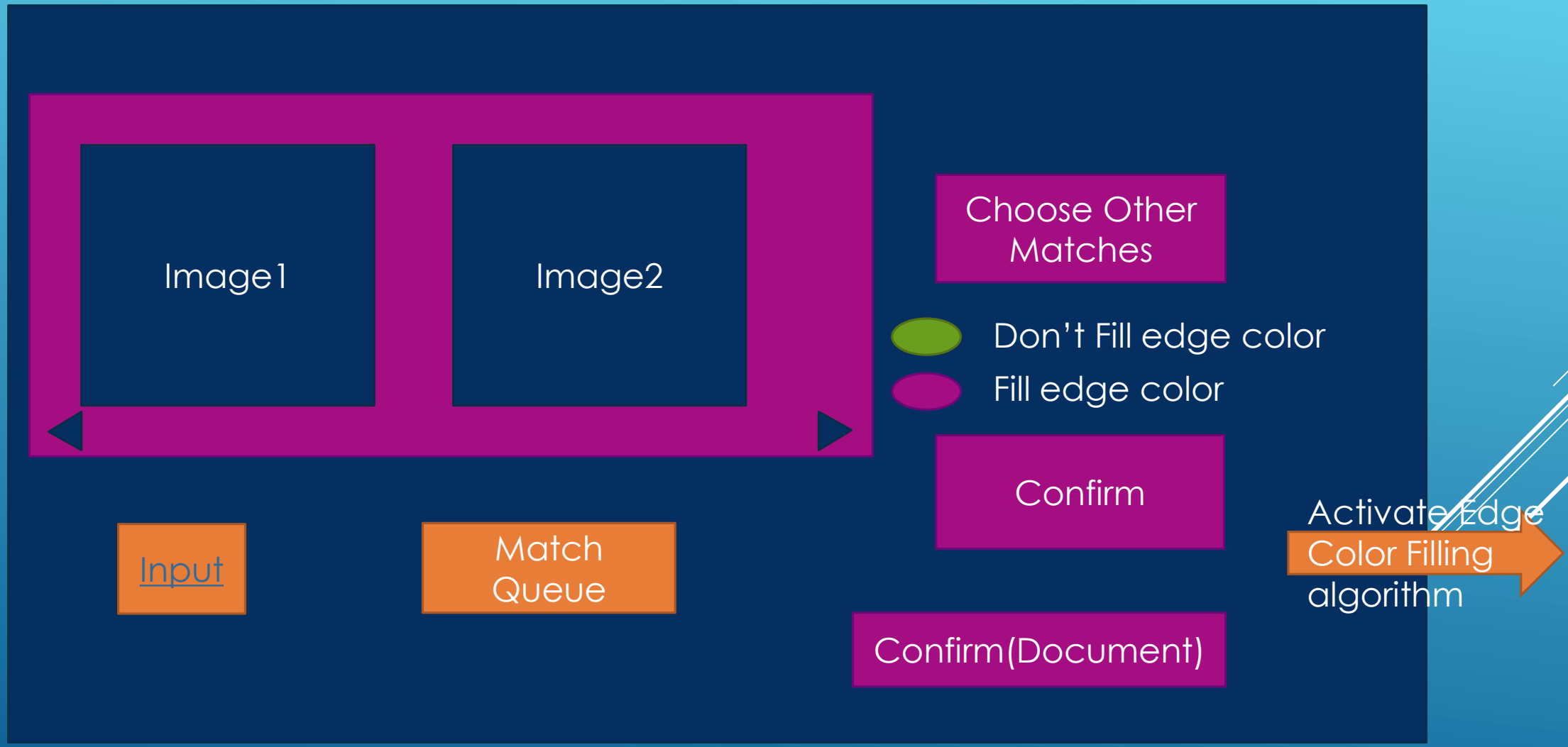


Click next will run all the algorithms required for matching pieces. This includes all the listed algorithms except the last one


Open File Dialog
Provided by C#
Standard Library
Add a file
Add from
Camera
Back
Camera Output Or File Preview
Back
Confirm

Orange buttons are used for debugging only, will not appear in release version

Closing this window will end the current session



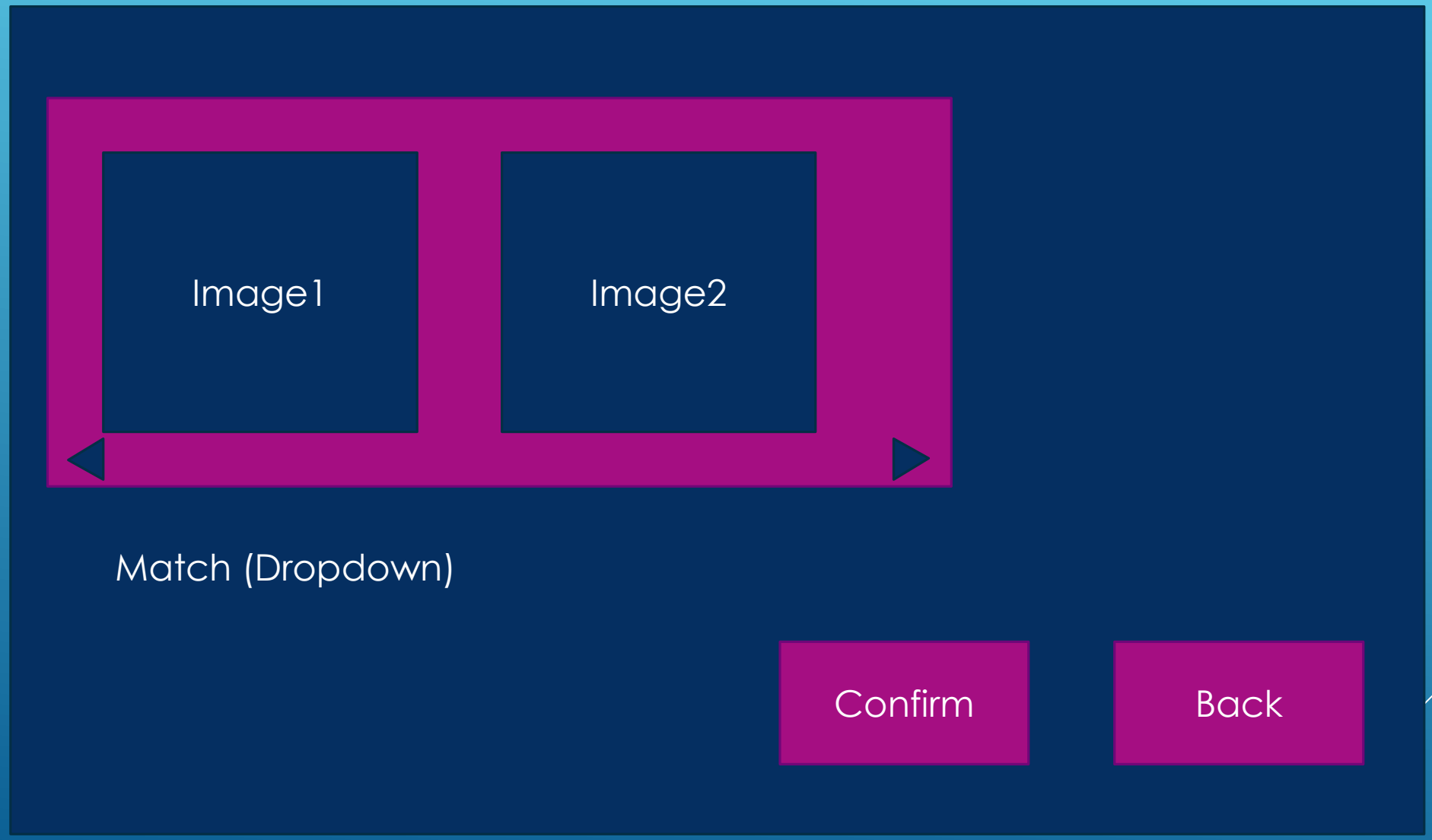
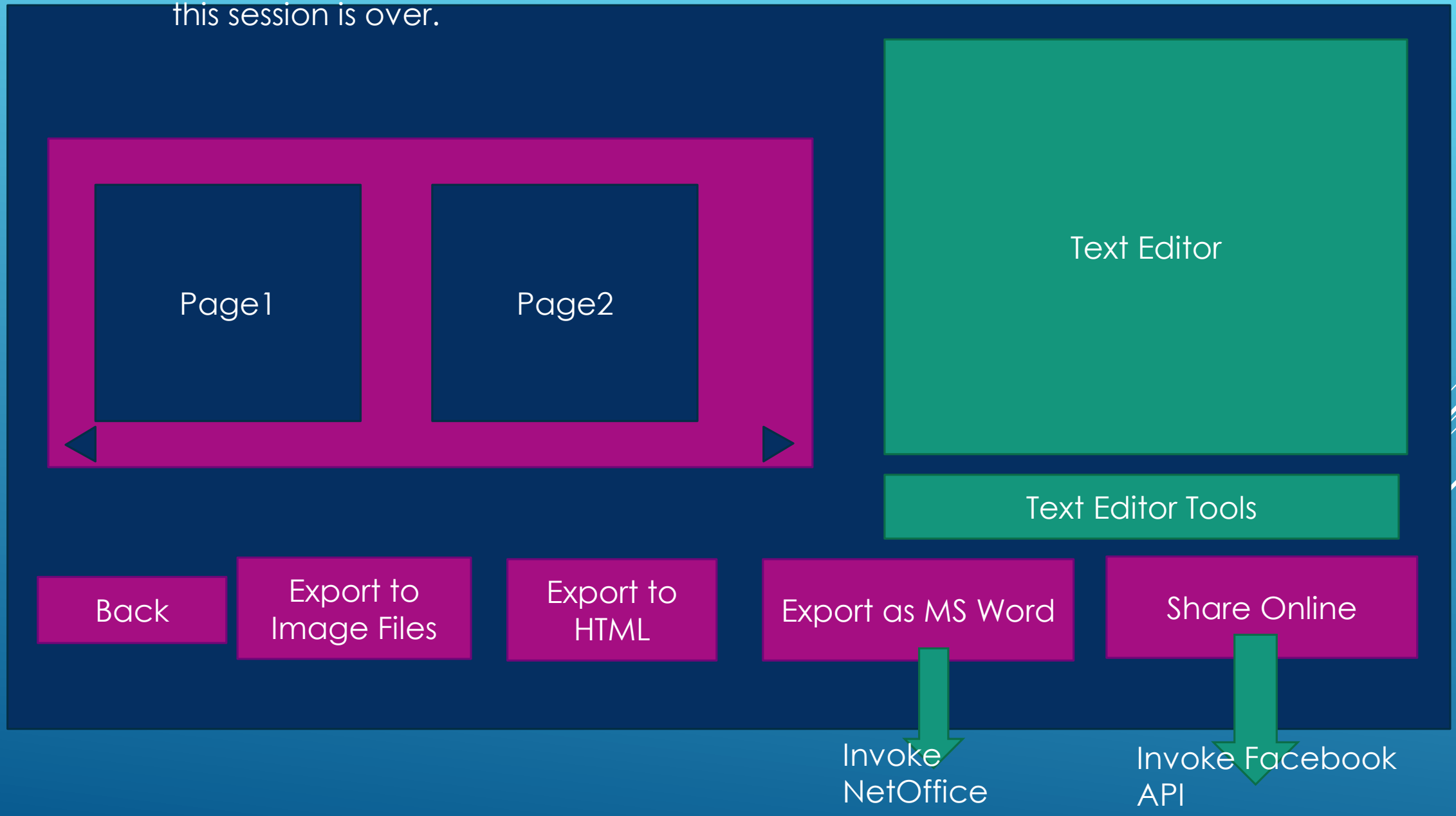
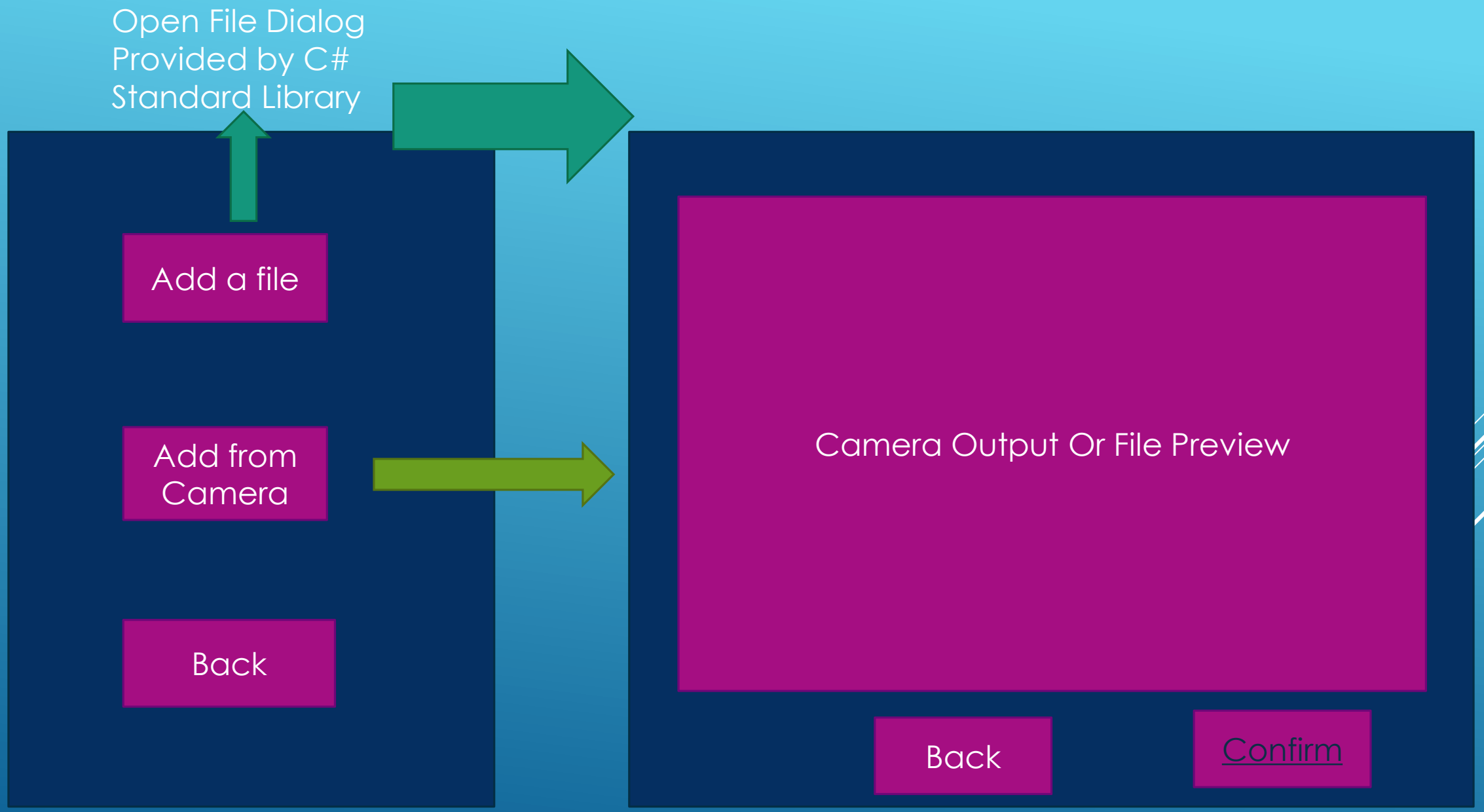


Image1
Image2
Match (Dropdown)
Confirm
Back

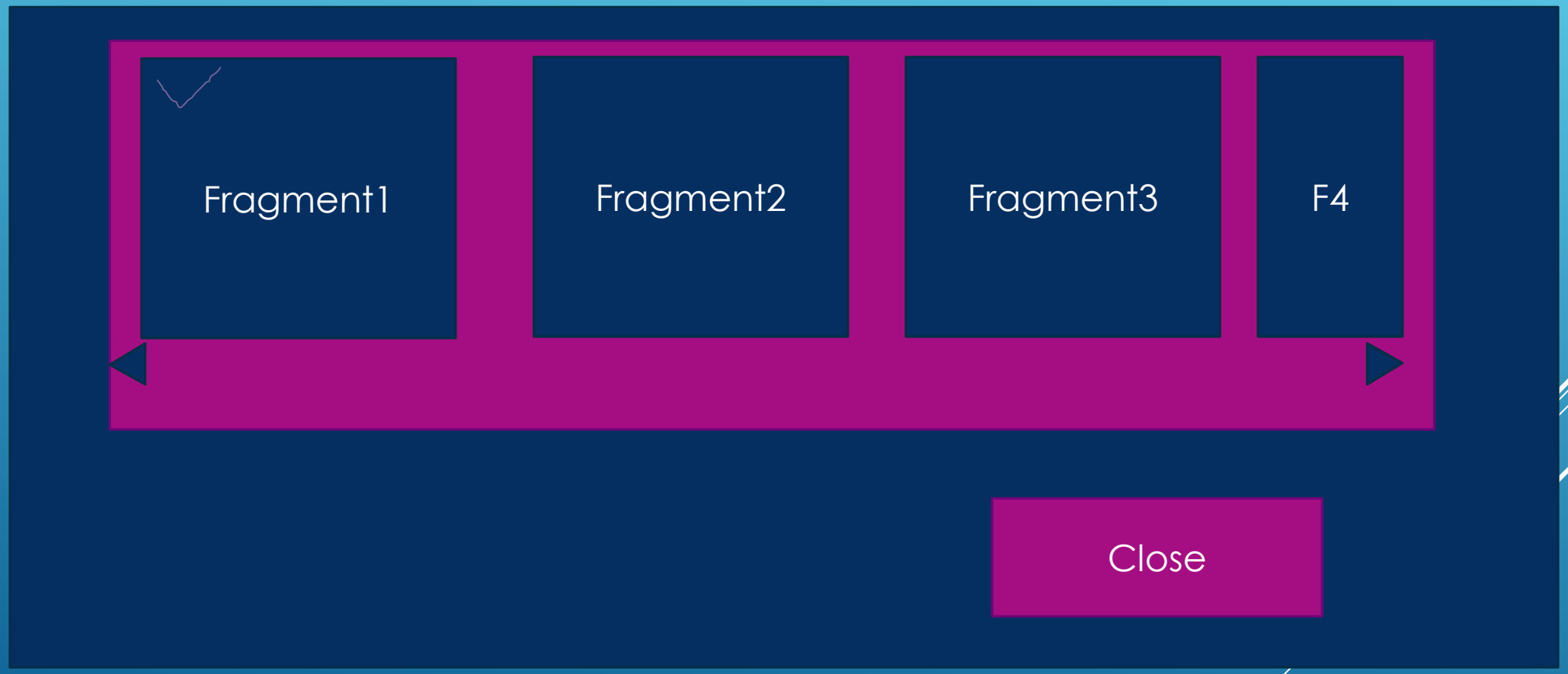
The output will be first converted into HTML, then transfer to PDF using IronPDF. Users will get both outputs. If user close this window, all the temporary storage will be deleted and this session is over.

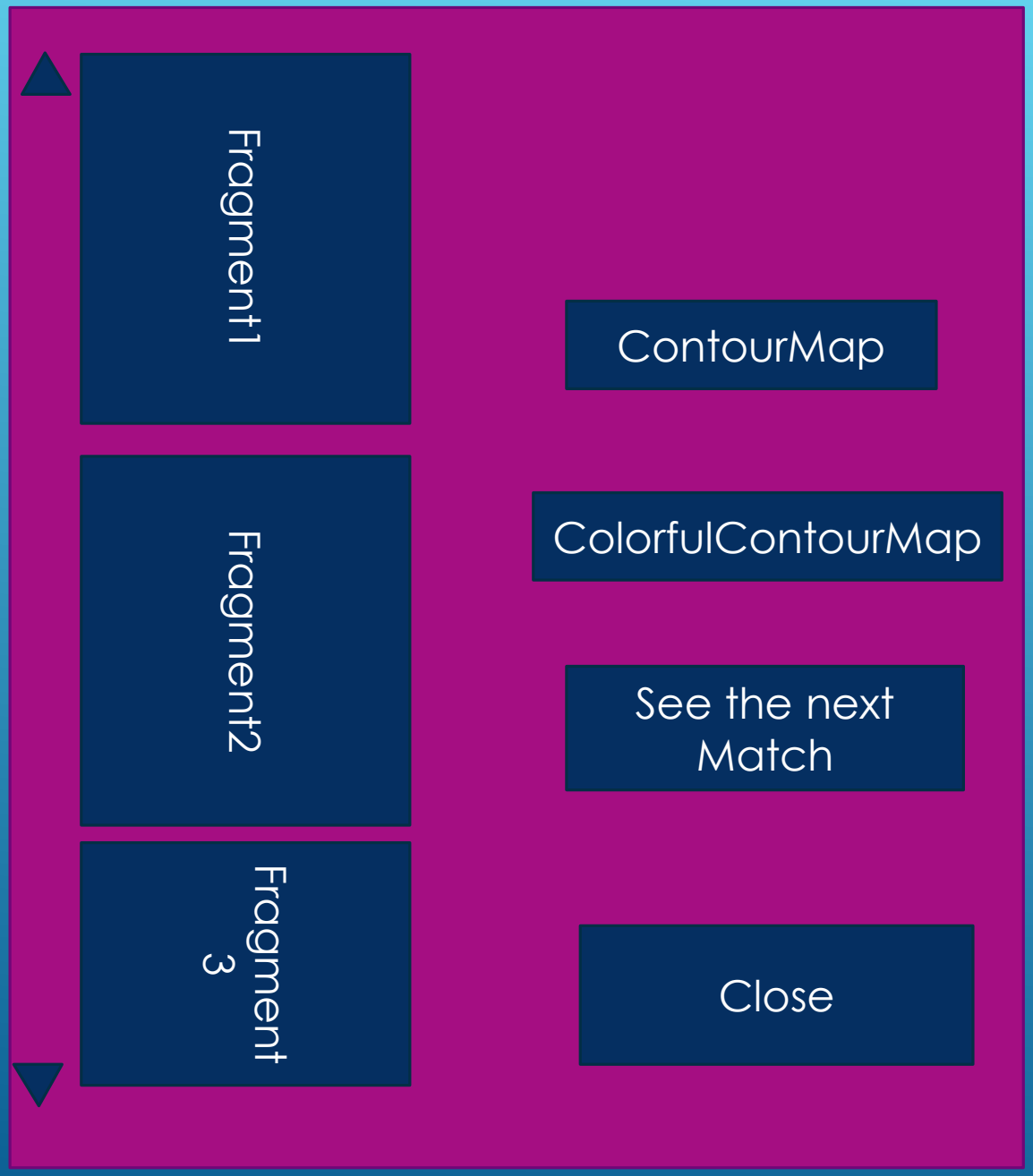
The output is text file, but if user wants image file, they can do so too. If user close this window, all the temporary storage will be deleted and this session is over.




Open File Dialog
Provided by C#
Standard Library
Add a file
Add from
Camera
Back
Camera Output Or File Preview
Back
Confirm

Debugging screen: Input





Debugging screen: Matching Queue

A visual representation of the List<ContourMap>

Used to show the queue data structure when matching the pieces

Debugging screen: Analyze the 2-piece matching