Human Computer Interaction Coursework 1: Report

s0949775 and s0935850

Introduction:

This report describes the process of designing and implementing a user-interface for an image annotation application.

Task Analysis and Design Process:

We started by analysing the task for which we were designing an interface. In brief, the task is to create, edit and view labelled regions on an image, and to be able to do this for a large collection of images. We wanted to create a user interface which was simple and intuitive, while offering powerful functionality. The interface had to be designed with the user in mind, so we took the position of a user and tried out some existing systems which tackled a similar task. Whenever we encountered a bug or difficulty in doing the task with that interface, we stopped to consider how it could be done better. We also performed a rudimentary Hierarchical Task Analysis which is shown below:

Hierarchical Task Analysis:

- Draw a new shape
    - Enter draw mode by clicking toolbar button.
    - Click on object boundary in image to make first point.
    - Click along boundary to draw shape.
    - Complete shape by clicking on first point.
- Edit an existing shape
    - Enter edit mode by clicking toolbar button.
    - Click and drag corners of shape to reposition them.
- Delete a shape
    - Right click on shape in list of shapes.
    - Select ‘Delete’.
- Rename a shape
    - Right click on shape in list of shapes.
    - Select ‘Rename’.
    - Modify name in presented dialogue box.
    - Select ‘OK’ to confirm or ‘Cancel’ to keep old name.
- Load a new image
    - Click on ‘Open’ button in toolbar or in file menu.
    - Browse to image.
    - View preview to check it is the correct image.
    - Click open in file browser.
- Save a label set

- Click on the ‘Save’ button in the toolbar or in file menu.

- Access help files and documentation
  - Either choose ‘Help’ from Help menu, or press F1 on keyboard.
  - Read HTML document that appears.
  - Click X in top right corner when finished with help, or reposition window to work with app while checking help.
- Quit application
  - Click on ‘Quit’ button in toolbar or file menu.
  - If there are unsaved changes, choose to quit anyway or cancel in dialogue box.

We made the hierarchical deconstruction as flat as possible to ensure that tasks were kept as simple as possible. We coupled with the hierarchical task analysis with the user centered approach, and integrated the result with HCI design rules to create the feature list shown below: (Also included are notes on relevant usability guidelines and HCI principles).

Features:

- Draw edges of polygon with a series of mouse clicks.
  - This is very simple to understand: high learnability.
- Adjust dimensions of shape by clicking and dragging on corners. e.g ----------> 
  - This is an intuitive way to modify shapes, as well as offering useful functionality.
- Ability to undo and redo previous actions.
  - Allows users to quickly and easily correct their mistakes. An alternative solution is to allow users to delete points and redraw them, but we think our solution is more intuitive and familiar to users.
- Shapes are highlighted when selected.
  - This makes it easier to understand which polygon is which for crowded scenes, and makes for an intuitive WYSIWYG editing interface.
- Liberal use of right click context menus for extra options.
  - While context menus do hide features away slightly, they can offer extra options, or just an alternative location to find features (eg having an Undo icon *as well as* right click > undo). It is playing on the HCI paradigm of familiarity: most users will have seen right click context menus before and will expect to see them when they right click.
- Ability to rename and delete shapes
  - We used right click context menus as these should be familiar to most users of computer software. The name of the object is right clicked in order to perform tasks relating to it, which we think is intuitive. While renaming, the existing name is presented to allow users to easily make small edits to the name, eg correcting a spelling error, without having to recopy the whole name.
- Availability of keyboard shortcuts for simple tasks, eg undo, redo.

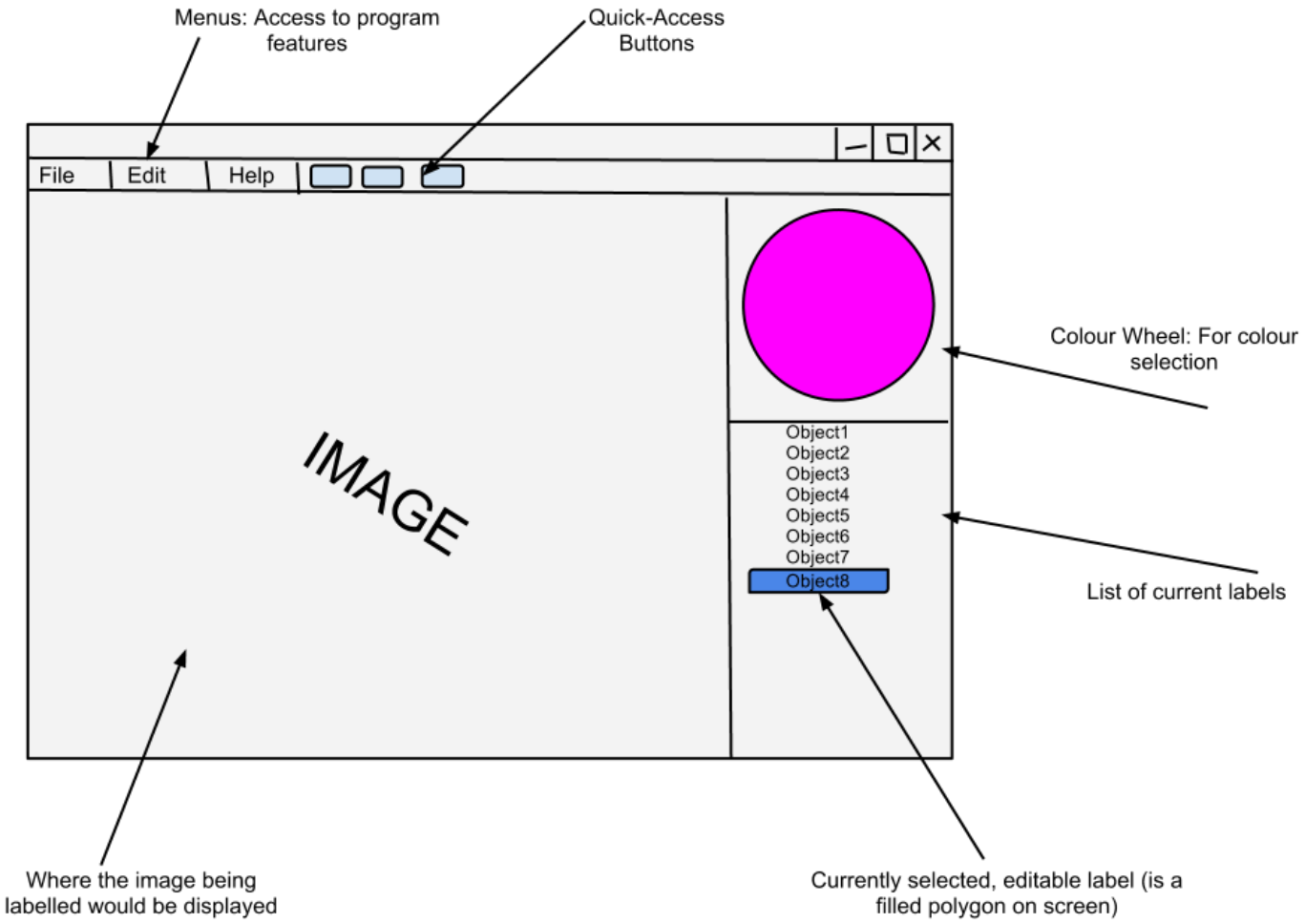
    - This improves flexibility, making the program more useful for frequent users who want to make tasks faster (Shneiderman's 2nd Golden Rule). It doesn't affect learnability as simple icon based tools will still be available, it is a feature for frequent users. Shortcuts will be chosen to be familiar to users, eg Ctrl + Z for undo.
- File menu with options to save and open images, redo and undo actions, and offer access to help.
    - Familiarity again here: most programs have menus of the style File, Edit, Help etc, so we are meeting that user expectation.
- Toolbar with icons for common tasks
    - Provides a simple and visual way to achieve a task. Later, frequent users can transfer to using keyboard shortcuts instead. Icons were carefully chosen to provide a visual metaphor for the tasks being performed, for example the icon for draw mode is a pencil.
- Cursor changes depending on the task being performed. Eg, if edit mode is selected, it turns into an hand to grab points with.
    - This reduces the short term memory load on the user, and makes clear what will happen upon a mouse click. This is also an example of using a visual metaphor in design: when in edit mode the user can 'grab' points and move them with a cursor that looks like a grabbing hand. We use a crosshair for draw mode, when the aim is to precisely draw points.
- Provision of help files and other user guidance.
    - We use tooltips on icons, labels on the interface that suggest functionality, and a more detailed help file available through a menu.
- Toolbar icons are greyed out to show whether they are available or not.
    - For example, if there are no unsaved changes, the save button is greyed out. This provides a very visual way for a user to understand whether a task is possible or not; another example of visual metaphor.
- Use of previews in file browser when loading new images.
    - This allows the user to see what image they are about to open without having to load it and go through the whole process again if they picked the wrong one. This reduces users' short term memory load (and frustration).
- Save icon on toolbar saves label set
    - Save button stores the label set in a way that is transparent to the user and does not modify the original image. Loading is also transparent: load an image and a label set, if present, is loaded with it. (Otherwise a label set is automatically and transparently created.)
- Overlapping shapes can be selected by repeated clicking which cycles through the shapes, highlighting each one until the desired shape is selected.
    - This ensures shapes are never inaccessible because they are obscured by another.

We followed the Interaction Design Cycle: analysing the task, designing features and prototyping, and cycling through the process until we were ready to implement. Our prototyping

consisted of sketches outlining the user interface as a whole, and the specific details of parts inside it, such as how the polygons would look. When we came to implement these features, we did so iteratively, according to agile development, focusing on always having a working system and adding new features according to priority. We did not use the sample code but instead started from scratch.

Prototyping:

We prototyped our designs by mocking them up in a simple graphics application, one such example can be seen below:



A few differences can be seen between this prototype and our final implementation. Most notable is that the colour wheel in the prototype is not present in the final design. We explored this alternative design choice, where the user would be able to select the colour of a new shape, but we eventually decided in favour of having a random colour arbitrarily chosen for each shape. We chose the random colour option to reduce complexity for the user. The colour wheel would offer useful functionality to some, as it would allow for colour coding of shapes, but we felt that

this was at a cost to the simplicity of the system and ease of use, and this directed our design decision.

User Testing

We recruited some users who were not domain experts to try out our design and report any problems they encountered, or possible extensions we could make. One thing that came out of this process was that the file browser was not easy to use, and could be simplified. At that time it was opening in the root directory of the computer and so required significant navigation to the desired file. As a result of the user feedback, we changed the browser so it opens in the current working directory, we added a filter so that only images could be seen or opened, and we added icons on the images to make it clear what they were. Finally, we added a preview pane to the file browser to ensure that the user has successfully navigated to the file they thought they had.

Comparative analysis of LabelMe

The LabelMe tool (http://labelme.csail.mit.edu/tool.html) is an application which also enables image annotation. It has some similarities to our design, but also some differences. The core functionality of how polygons are drawn is similar, in that points are drawn one by one until the polygon is closed. In our design, polygons can be edited easily by clicking on the 'Edit Mode' button. LabelMe supports label editing but the feature is less visible: you must first click on the polygon to select it, then choose 'Adjust Polygon' to see the editing handles.

LabelMe has a zoom functionality, which we do not have, although it is temperamental: zooming very far in, adding a polygon, then clicking the button to return to the full picture broke the program.

For controls, LabelMe uses a toolbar of labelled icons, which is a good way to make very clear what features are available. Our application also uses a toolbar of icons. LabelMe provides help icon on the toolbar, which links to an HTML page of help. We also have an HTML help file, though it is linked to through a menu.

As we do, LabelMe provides a list of polygons which can be clicked on to select shapes for editing. When moused over, the relevant polygon in the image is highlighted, which makes it clear that the two are linked.

A final note is that LabelMe allows the user to download both the image file, and an XML file of the labelset. Our application is compatible with these XML files and can load them in without modification (so long as the file names match, e.g. image.jpg must have image.jpg.xml).

Strengths, weaknesses, and possible extensions of the design

- Simple to use, yet with powerful and flexible features.

- Complementary ways to achieve the same goal, eg opening an image via the file menu (File > Open) or opening via an icon on the toolbar. The rationale is that users will find one of the ways fairly quickly, and can form their own preferences.
- Once an image has been loaded, the tools for working on it are good. However we could implement a more fully featured way to select images which abstracted the user away from dealing with the file system directly.
- Shapes can be edited in flexible ways, although they cannot be moved from one place to another without moving all the points one by one. An alternative solution would be to have a handle in the shape which can move it, however, our task analysis led us to believe that this is not a common task, as users are drawing polygons traced over an image in a specific case, so we decided it was not worth complicating the design for this rare task.
- We followed usability guidelines when possible, such as allowing frequent users to use keyboard shortcuts.
- Lack of zoom/resize, this makes it harder to annotate very small or very detailed things. Adding a zoom feature would be a good possible extension of the design.
- Drawing of points can be undone and redone, but this does not generalise for all actions. This would be another possible extension of the design, but would require significant refactoring of the code to store all actions and make them reversible.

Conclusion

This report has detailed the process of design for an image labelling application, from task analysis through prototyping and implementation to evaluation and analysis.

See the README file at the top level of the source code zip file for instructions on running the program.