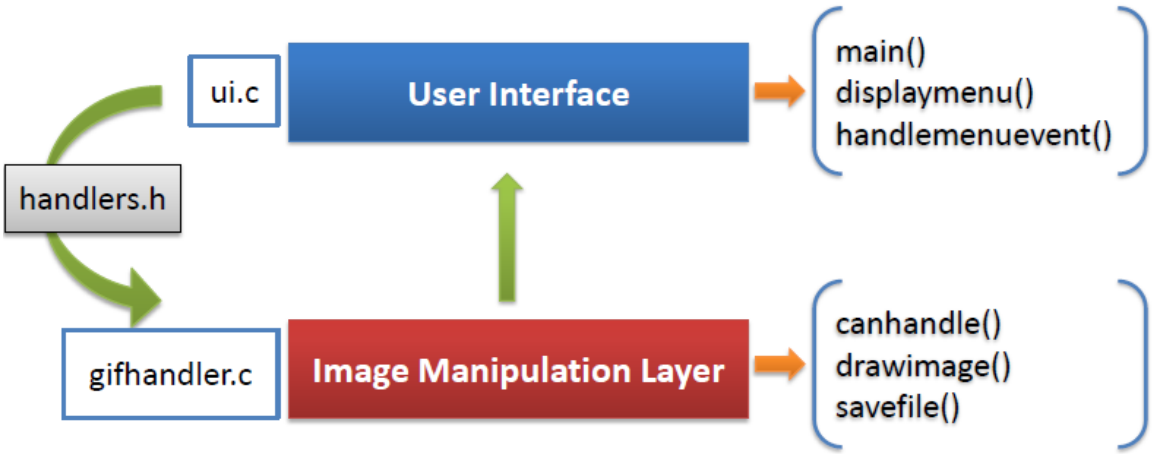
Case Study on Application Designs – Class Notes

- Let's design an application that can take an image file as input and display its contents. It should also provide image conversion facility.

Approach 1:

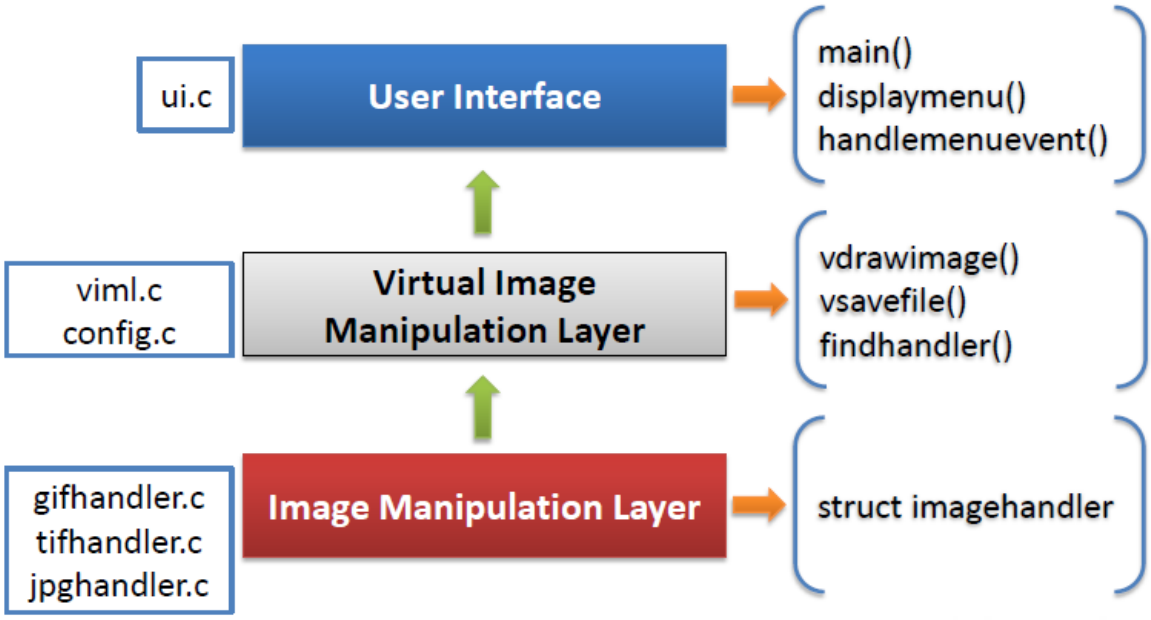


- The above approach displays gif format files. It provides a user interface which provides a display menu and it takes the input from the user and handles it.

The limitation of the above approach is it supports only gif format files; it cannot handle any other file formats.

- Redesign the above application to meet the following objectives.
  - Add support for at least 3 different image formats
  - Proposed new implementation should allow future extensions with minimal or no changes to existing code
  - Achieve objectives 1 & 2 with minimal changes to current implementation.

Approach 2:



Design changes:

- Image manipulation layer will be modified to accommodate three separate image handlers
- A new module called Virtual Image Manipulation layer (VIML) is introduced to abstract user interface from handlers in IML (Image Manipulation Layer).

VIML should provide following:

- A common interface (file format independent) for UI (User Interface) layer to request image processing and conversion.
- An interface for the image handlers to register with VIML.
- When UI submits a request, request processing logic should be able to locate and invoke appropriate handler.

Code changes:

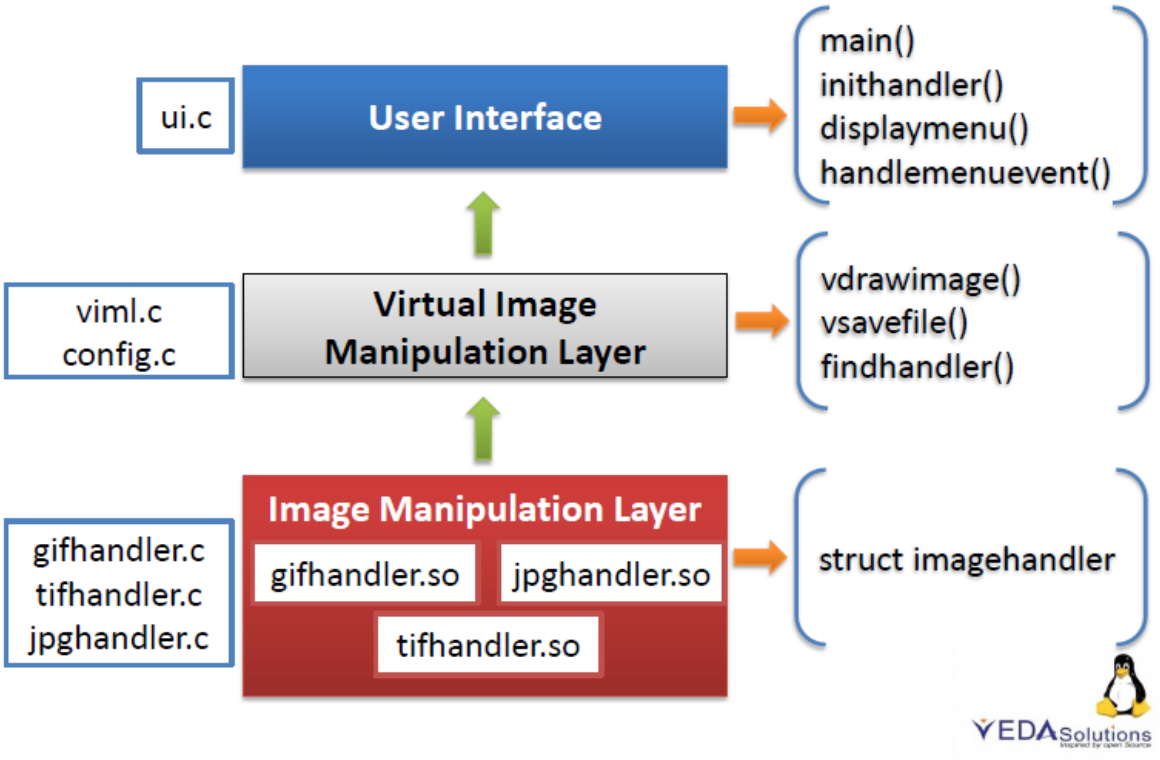
- Modify **ui.c** to invoke functions of VIML.
- Implement VIML source code as per the design needs.
- Each handler should implement its image processing routines and store their addresses in an object of type image handler
- Image handler objects of each handler will be stored in a data structure maintained by VIML.

Limitations of approach 2:

- The current design is extensible but only when plug-in developers have access to source code
- Each extension requires a rebuilt.

Redesign the approach 2 to add support for binary extensibility

Approach 3:



Design changes in approach 3:

- All handlers in IML layer will be built into independent shared objects.
- Application executable image will comprise of user interface and VIML layers.

Code changes:

- Main function in the user interface module shall call a routine init_handler(), that loads all handlers (**.so** s) before invoking display menu.
- VIML handler provides two new functions at **config.c** that can add an image handler object to VIML vector and remove the object when it is no longer needed. They are :

  reg_handler and unreg_handler
- Each of the handlers of the IML layer would provide their own implementations of **_init** and **_fini** (over writing default run time routines).