Plugin Architecture for simplecv-js

The main goals of implementing this plug-in structure in simplecv-js are :

- ➔ To make the code more managable
- ➔ To allow other developers build sophisticated applications using simplecv-js
- ➔ To provide a structure which facilitates future extension of the tools
- ➔ To make development of each module independent.

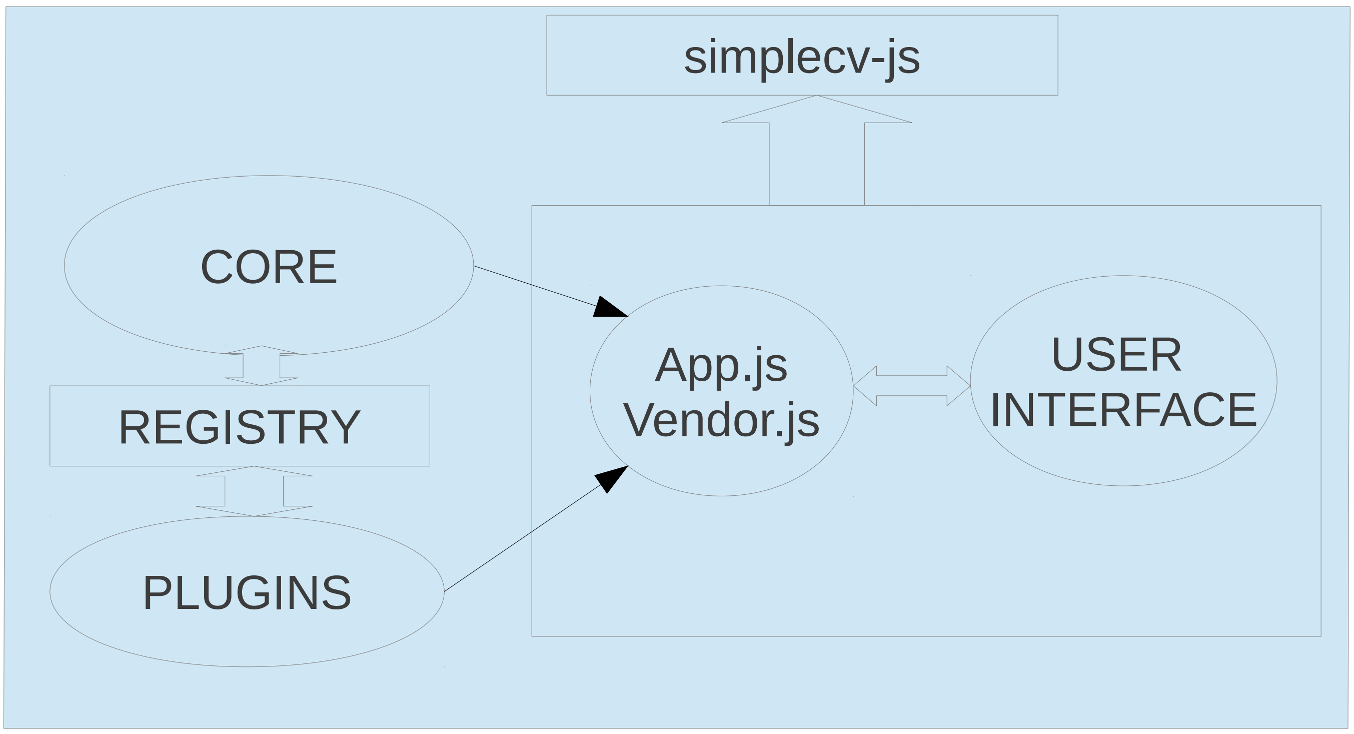
I'm going to explain a plugin architecture which i think would be better for simplecv-js. The whole codebase consists of three parts.
They are :

- ➔ Core
- ➔ plugins
- ➔ User Interface

core : The core of the simplecv-js consists of basic elements of the toolset for which separation makes no sense. They provide core functionality of loading images and rendering and some basic methods. It contains registry which stores plugins to provide extra functionality.

Plugins : These contains of the methods which extend core functionalities and implements different interfaces to obtain a tool for its specific use. These interact with the core using the registry.

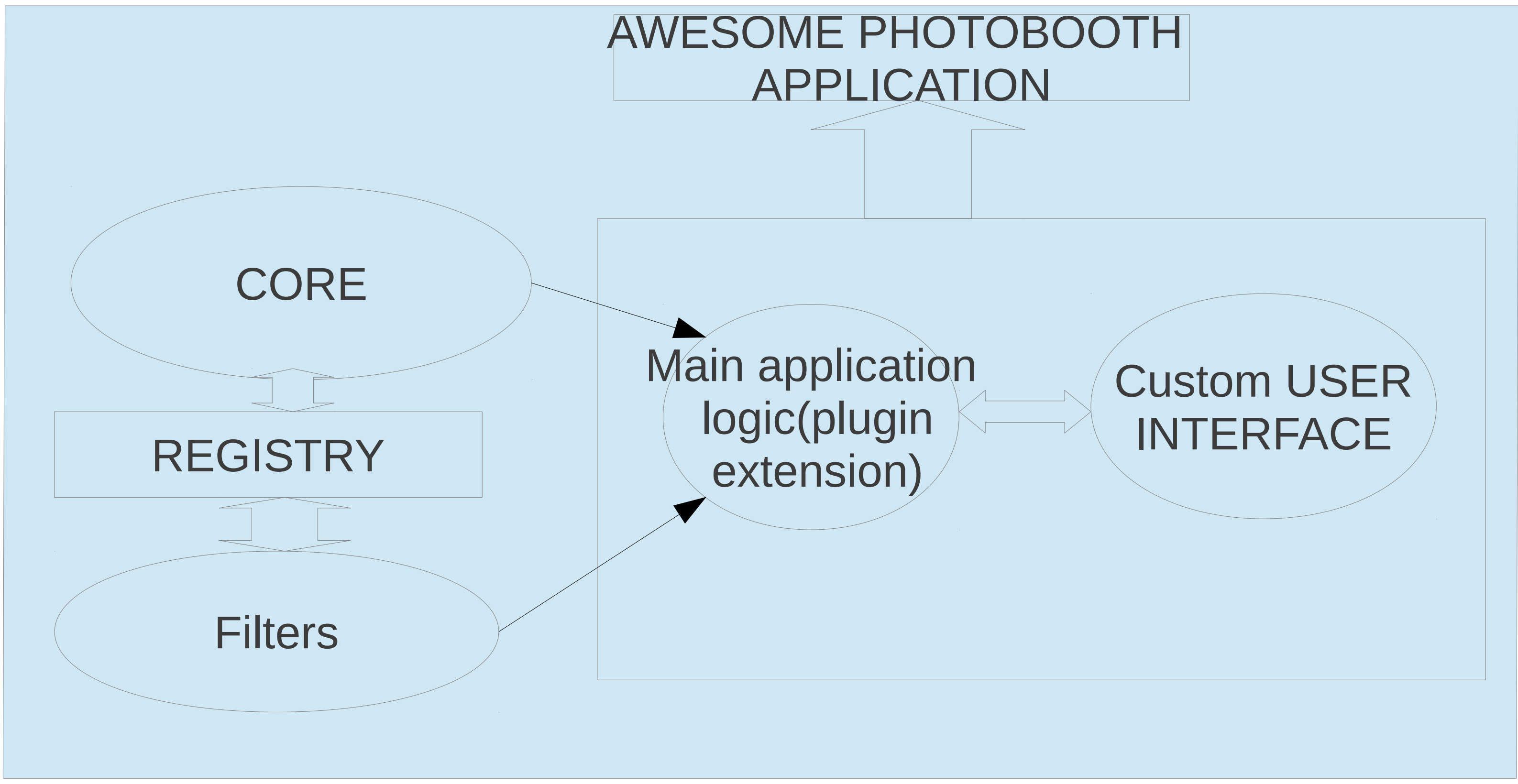
User Interface : This contains basic design of the interface and interacts with core and plugins . This is not mostly dependent on the plugin architecture.



I'm going to state a few examples of how developers can use simplecv-js and how the plugin system comes in hand.

A PHOTO-BOOTH APPLICATION :

People who want to build a photobooth application can use simplecv-js' core and plugins like filters and extend it with the application logic. A figure that shows the block diagram is below :


AWESOME PHOTOBOOTH APPLICATION
CORE
REGISTRY
Filters
Main application logic(plugin extension)
Custom USER INTERFACE

In the same way one can build many apps like a custom hangout tool with effects (simplecv-js is built with webRTC functionality) , Object tracking , Face authentication tools , QR and barcode scanner , build 3D games (kinect support) , and many more sophisticated browser based computer vision applications.

An example syntax of the implementation looks like this :

```
class Plugin
  @plugins = {}
  @register: (name, plugin) -> @plugins[name] = plugin
  @execute: (context, name, args) -> @plugins[name].apply context, args

Image.Plugin = Plugin
```

And an example plugin will be registered and defined like this

```
Image.Plugin.register "compoundBlur", (radiusData, radius,increaseFactor,
blurLevels) ->
```

P.S : If i have enough time i would like to develop an application to demonstrate how simplecv-js can be used with plugins .