Image Acquisition on PixInsight
Do we really need another tool for controlling cameras?

David Raphael

I think it depends on whether or not you are happy and content with what you use today. Personally, I love a number of the tools that are available commercially. In fact, if it wasn't for Emanuele (a PixInsighter) – I may have never thought of working on this. But not long after it was suggested on the PixInsight forums – I realized that PixInsight would make a fantasic image acquisition platform!

PixInsight allows me to extract every ounce of data possible without destroying the data in the process. I realized that this is exactly what I wanted for image acquisition as well! I believe that it accomplishes this with several key features:

- Object Oriented Design
- Component Based Architecture
- Scriptability
- Programmability
- Community

These features allow PixInsight to be very fluid. It moves with the desires and philosophy of the users. This may seem like a peculiar way to describe a piece of software, but it reminds me very much of the text editor EMACS. *For those of you unaware of what EMACS is – it is basically one of 2 standard text editors that hackers / programmers / sysadmins have been using for over 30 years – vi being the other.*

Just like EMACS became more powerful as users contributed functionality – PixInsight becomes more powerful as users contribute! This is a very organic growth model.

Visual Studio is to MaximDL what EMACS is to PixInsight – at least in my crazy way of looking at the astronomical software landscape. Of course most C++ developers use Visual Studio – but the really good hackers use something like EMACS, vi or even something else...they want to get closer to the code...just like we want to get closer to the pixels!

Along with the awesome extensibility that is afforded by this approach – PixInsight runs on every Operating System I would ever run. In fact, it even runs on FreeBSD! I don't particularly like being tied to one platform for astronomical work.

So now that I've ranted – let's talk about how I am implementing ImageAcquisition on top of PixInsight!

First, I've divided up the effort into several building blocks:

- Image Acquisition Settings
- Expose Image
- Imaging Session
- Auto-guide

- Focus Control
- HFD/FWHM
- Auto-Focus
- Plate-Solve
- Slew7
- Dome Control

This is not a comprehensive list – I simply made a list of all the items that I felt belonged in the image acquisition bucket.

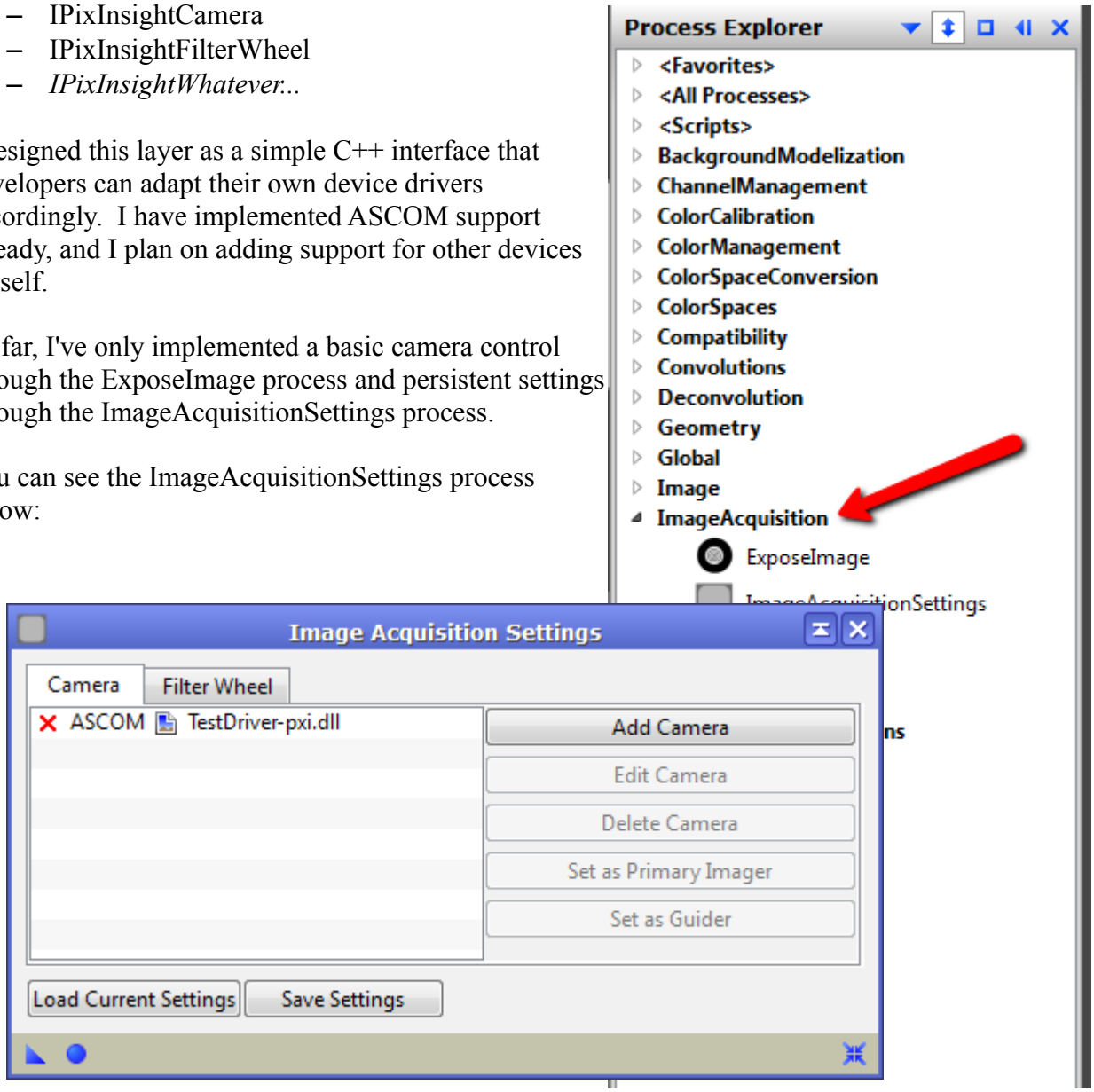
In addition to these items, I also had to design a device driver interface. I considered a number of options, but ultimately I decided that it would be best to have a PixInsight device layer:

- IPixInsightDevice
- IPixInsightCamera
- IPixInsightFilterWheel
- *IPixInsightWhatever...*

I designed this layer as a simple C++ interface that developers can adapt their own device drivers accordingly. I have implemented ASCOM support already, and I plan on adding support for other devices myself.

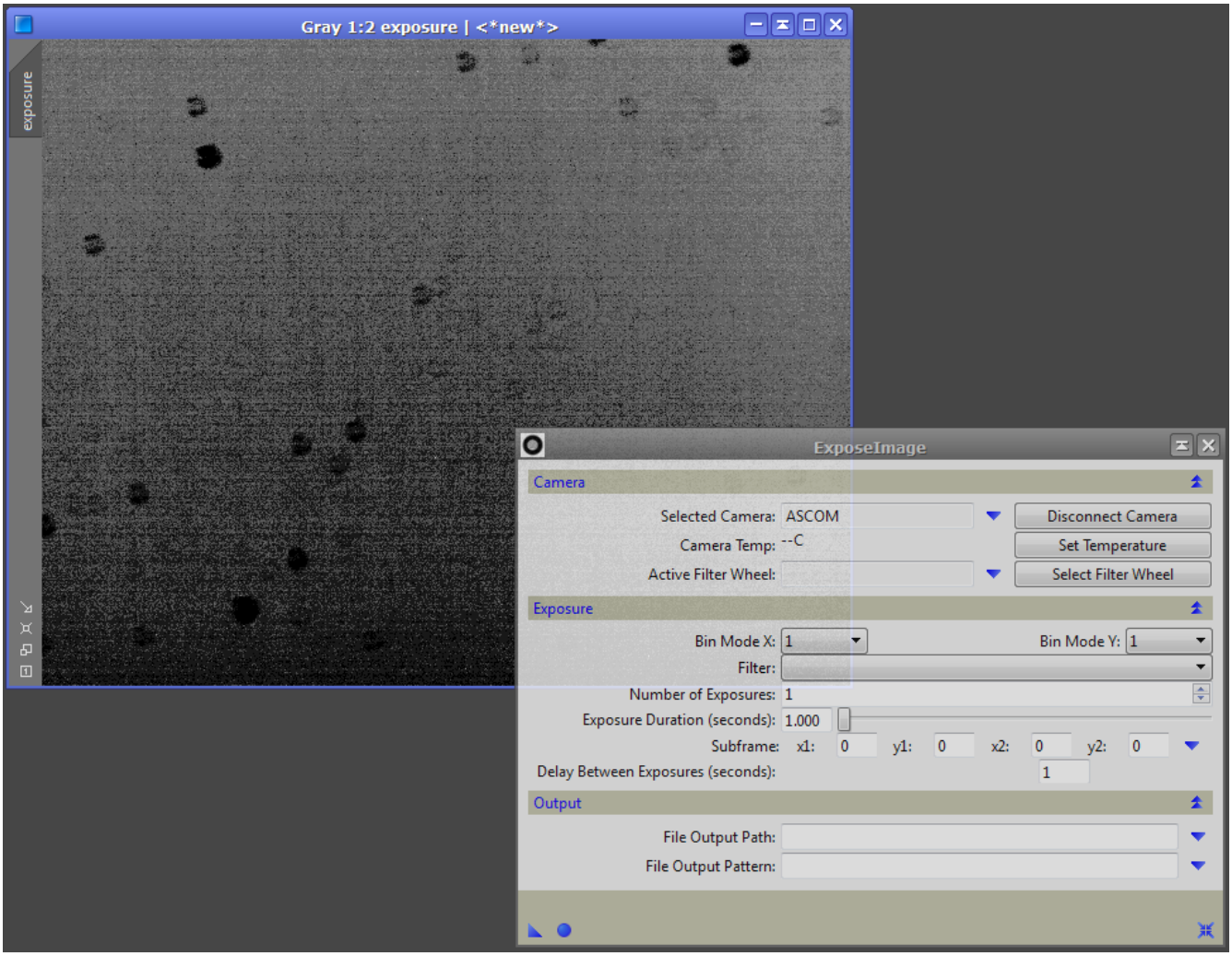
So far, I've only implemented a basic camera control through the ExposeImage process and persistent settings through the ImageAcquisitionSettings process.

You can see the ImageAcquisitionSettings process below:



This allows you to configure as many different cameras and guiders as you want. You can save these settings and reload them when you open and close PixInsight. Additionally, you can save your settings through the standard PixInsight mechanism via a process icon.

Below is the ExposeImage process with a sample image that was taken. Look at all of that dark nebula!



If this were just another application with a basic UI for controlling a camera – I would say that this is not very impressive. However, what really gets me excited – is the fact that you can leverage the PixInsight platform's genius right away! For example – this screenshot demonstrates the power of an object oriented platform:


ProcessContainer
Proc Id Mask Start UTC | Time
0 <Root>
1 ExposeImage
2 ExposeImage
3 ExposeImage
var p = new ProcessContainer;
with (p)
{
var p_0 = new ExposeImage;
with (p_0)
{
exposureDuration = 1;
exposureCount = 1;
cameraName = "cam_name";
filterWheelName = "";
setTemperature = -1.5e+001;
filter = "NONE";
binModeX = 1;
binModeY = 1;
subFrameX1 = 0;
subFrameY1 = 0;
subFrameX2 = 0;
subFrameY2 = 0;
delayBetweenExposures = 1.0e+000;
fileOutputPath = "";
fileOutputPattern = "";
}
add(p_0);
var p_1 = new ExposeImage;
with (p_1)
{
exposureDuration = 2;
exposureCount = 1;
cameraName = "cam_name";
filterWheelName = "";
setTemperature = -1.5e+001;
filter = "NONE";
binModeX = 1;
binModeY = 1;
subFrameX1 = 0;
subFrameY1 = 0;
subFrameX2 = 0;
subFrameY2 = 0;
delayBetweenExposures = 1.0e+000;
fileOutputPath = "";
fileOutputPattern = "";
}
add(p_1);
var p_2 = new ExposeImage;
with (p_2)
{
exposureDuration = 3;
exposureCount = 1;
cameraName = "cam_name";
filterWheelName = "";
setTemperature = -1.5e+001;
filter = "NONE";
binModeX = 1;
binModeY = 1;
subFrameX1 = 0;
subFrameY1 = 0;
subFrameX2 = 0;
subFrameY2 = 0;
delayBetweenExposures = 1.0e+000;
fileOutputPath = "";
fileOutputPattern = "";
}
add(p_2);
}
JavaScript
Copy Code
Edit Info

For those of you unfamiliar with the power of PixInsight's processing container – you need to check them out. You can drag the little triangle of any process over to a process container and it will allow you to chain together multiple processes that can be executed with a single click. Here I am simply demonstrating 3 different exposure lengths.

Since this project is still in the early stages, the example is not very exciting. However, it already allows you to acquire a series of images with different length and different filters! Not too bad...

Vicent Peris, the editor of this magazine, asked me to add a feature to my list – auto-flats. I told him that this would be something easily scripted! No need to hard code this into the system. At least not just yet...here is a simple script that exposes images until a target mean pixel value is reached. This is not a great example – but I spent about 5 minutes writing this -

I don't know about you...but this gets me really excited!

```
var theDuration = .1;
var running = true;
while( running && theDuration < 2 )
{
   var p_0 = new ExposeImage;
   with ( p_0 )
   {
      exposureDuration = theDuration;
      exposureCount = 1;
      binModeX = 1;
      binModeY = 1;
      subFrameX1 = 0;
      subFrameY1 = 0;
      subFrameX2 = 0;
      subFrameY2 = 0;
      fileOutputPath = "c:\\tmp";
      fileOutputPattern = "test";
   }

   p_0.executeGlobal();

   var window = ImageWindow.activeWindow;
   console.show();
   console.writeln( "<end><cbr><br><b>" + window.currentView.fullId + "</b>" );
   console.writeln( "exposure..." + theDuration );
   console.writeln( "Calculating mean pixel value..." );
   var img = window.currentView.image;
   var statistics = new ImageStatistics();
   with(statistics)
   {
     meanEnabled = true;
   }
   statistics.generate(img);
   console.writeln( "mean pixel value:" + statistics.mean );
   console.flush();
   if( statistics.mean > .0006 )
   {
      console.writeln( "target exposure value:" + theDuration );
      break;
   }
   theDuration += .1;
}
```

There is still a lot of work to do in order to get ImageAcquisition ready for the field. I have open sourced all of the code – it is available here:

https://github.com/ceterumnet/ImageAcquisition

I hope that you all take a look at this module once we release it to the public. If you are feeling adventurous and want to contribute to the project – feel free to PM me on the PixInsight forums!