Compositing Theory and Practice Continued

Now that we've looked at the basic theory and math behind image manipulation and compositing theory, let's look at some different approaches in software.

Quick Review

Recall that basic image manipulation is usually performed on one image, with an input image, an operation, and the output image that is produced. We looked at things like RGB multiply, Add, Gamma Correction, Invert, Contrast; Filters that are convolved to produce Blurring, Sharpening, and other effects, and also basic geometric Transformations. Moving on to Multisource operators, we looked at Over, Mix, Subtract, In, Out, and Atop. Over is the one we'll use most in compositing. Things to be careful -- remembering if an image is pre-multiplied, including it's own alpha channel matte.

Creating mattes can be done in a variety of ways, the Color Difference Method (still based on Petro Vlahos' optical work in the 1950s) will be the most common. For every pixel, if the blue is greater than the green color, then in the new image the new blue will be the green value. Otherwise, the new blue will be the old blue value. Remember -- if it's a blue screen area, the blue value will be high (close to 1) and the green should be low (close to 0). That'll make that area 0 or black in the new image. If it's a fringe area, the blue will still be pretty high, but in the new image it will be replaced by a low number (old green value), resulting in a translucent edge. If it's a normal foreground area, the green should be higher than the blue because the blue should be close to zero.

There are a few ideas from Chapters 8, 9, and 11 that we had just begun to talk about. We'll continue them a bit here.

Overview of Chapter 11: Quality and Efficiency

The main point of this material is to be concerned with issues like compression of images, numeric representation, numeric accuracy, and consolidating operators.

Lossy compression will hurt us in compositing, particularly if it's done several times -- such as the several layers in a complex composited image. We'll continue to lose a little bit each time, and overall quality will be degraded.

Numeric representation is important because we need to know how much range (much like film latitude) that can be represented by our format. For instance, 24-bit color can represent 16.8 million colors, but since only 8-bits are used for R, G, and B respectively, we only have 256 values of each of those -- not a huge range of brightness when you think about it. Other formats might use 10-bits where 1024 values can be represented instead on each color channel. (It's a bit like going from reversal film to negative film that has a wider latitude or range of darks and lights that can be represented).

Numeric accuracy is important, as discussed in class, because if we start rounding, truncating, flooring, or ceiling before calculations are completed, then we may affect calculations and their accuracy. Operations such as rounding should be help until the end.

For this same reason, consolidating operators is an important concept in compositing. If a variety of image filters, effects, etcetera are applied one after the other, we may lose quality. One may clip values such as darks or lights as discussed in class. Others may multiply or perform other numeric calculations. The order in which they are performed may be important, and hopefully a program seeks to maintain numeric accuracy through the compositing operators, but sometimes we have to realize that it will not, and we have to create our composite differently to preserve quality.

Overview of Chapter 8 & 9: The Interface, Image Viewing, and Analysis

Near the end of class last Thursday, we began to discuss different ways to implement a composite operation or operations. There are three main approaches: scripting/expressions, a timeline with layers, or a node/graph-based approach. Under each of these, effects and operators are indicated in different ways, but the operations are still the same -- things like Over and Atop.

Here are some examples:

Script

Output = Over(Image A, Image B)

or

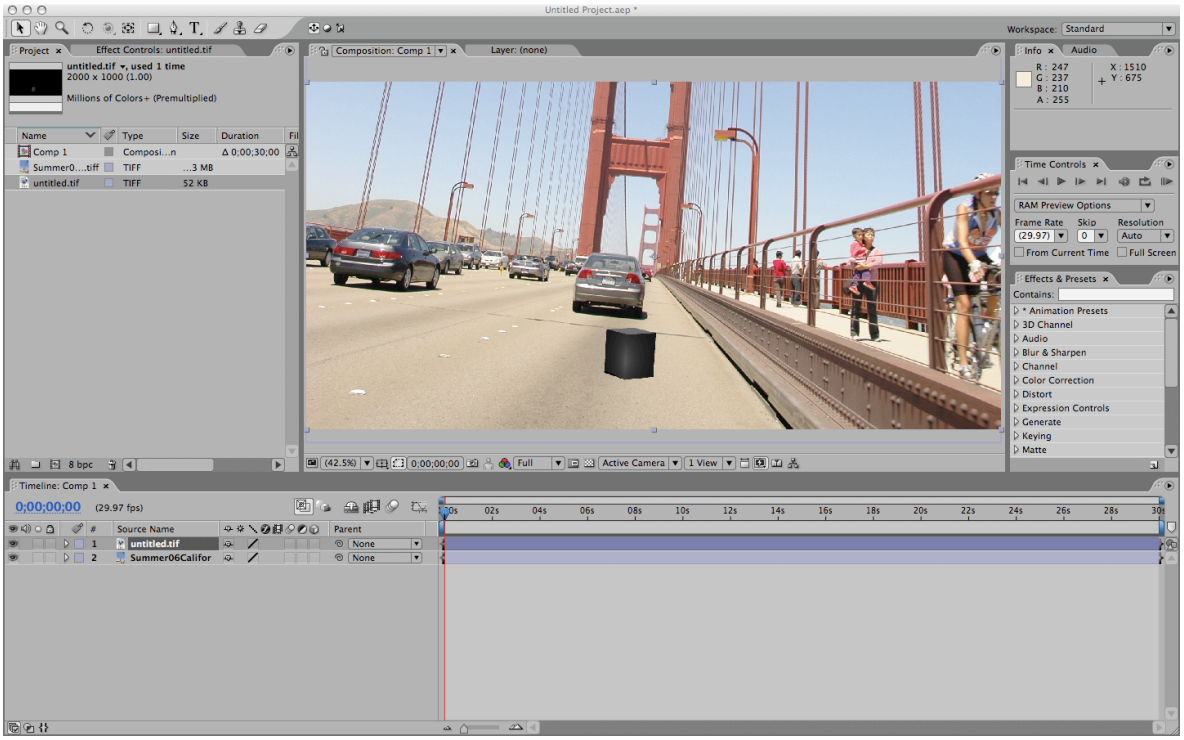
Output = Over(Brightness(Foreground, 1.2), Brightness(Blur(Background, 4.0), 0.8)))

Try to picture what these scripts might look like in a timeline or graph interface as you view the next examples.

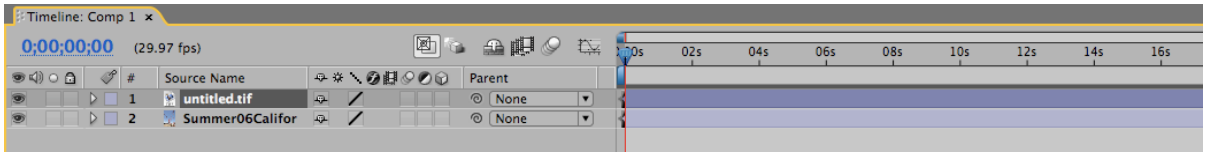
Timeline

After Effects takes a timeline approach in its main working space. (As with many programs, there are ways to switch approaches to varying degrees within a program.)

Here's an example of the interface with a simple *Over* operation:

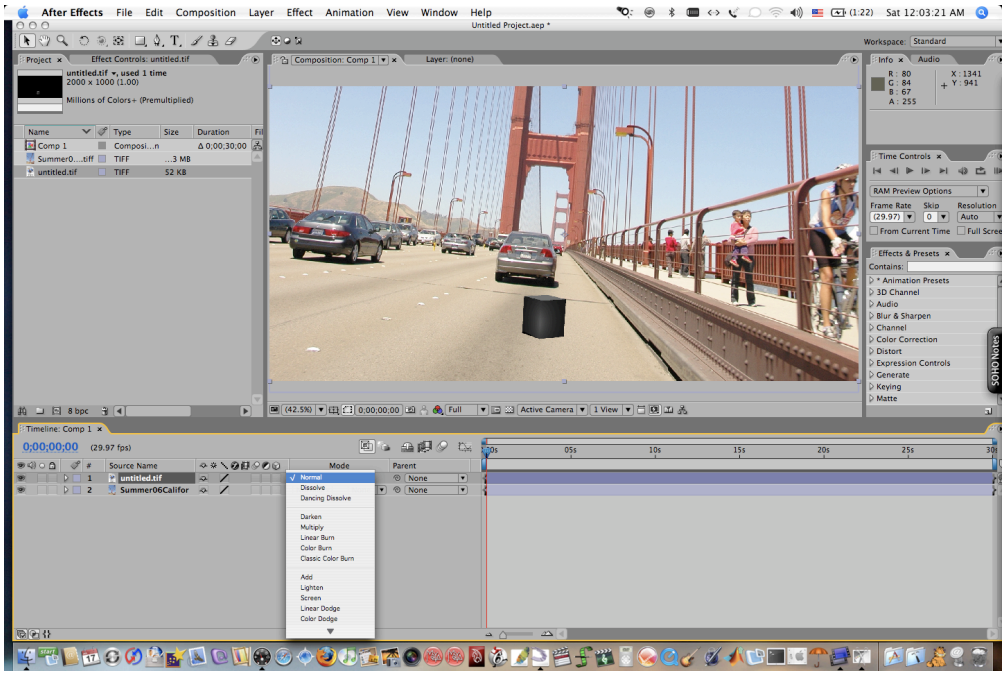


Notice the various parts of the working space. In the upper left corner is the project window where media clips are imported and referenced. In the middle is the viewing area, showing the current output of the location of the timeline playhead. In the right are a few controls and menus. At the bottom, is the main working area: the timeline. Notice that there are various layers for each part of the composition.

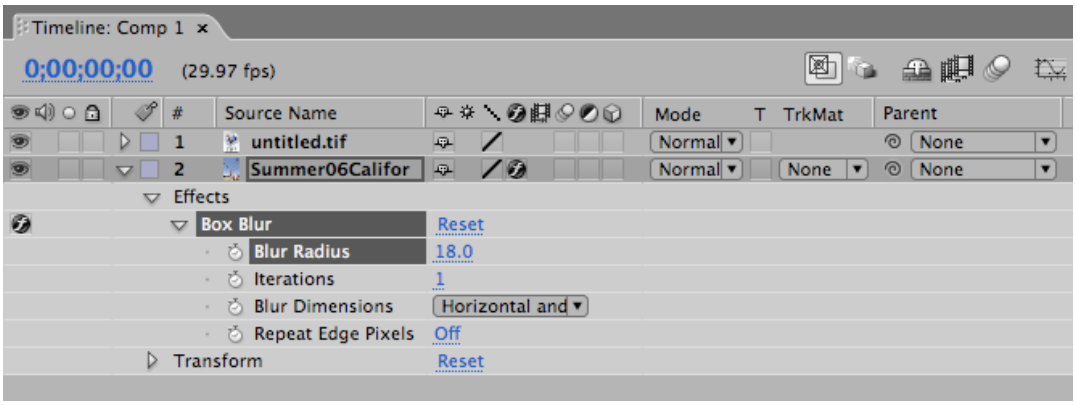


These layers are stacked vertically, showing elements that may be layered with others, using operations such as *Over* (the default). Different blending modes are available, though.

Right-click (or Ctrl-click) on the Timeline tab if you're in After Effects, and choose Columns-->Modes. This gives a new column in the timeline workspace as shown above, labeled "Mode". It says "Normal" in the image above. You can pull down on this tab, though, to select other blend modes such as the other Multisource compositing operators that we've discussed. (Hint for homework...)



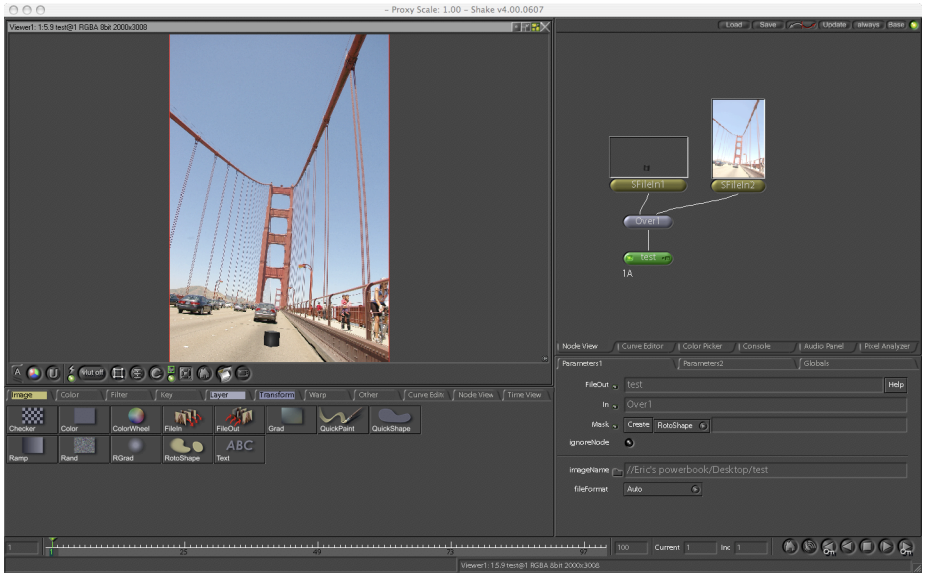
Also, in After Effects, effects such as filters and color correction may be applied by selecting a layer and choosing the Effects menu or by right-clicking on a layer and choosing from the Effects menu. When an effect is applied, it appears under the expanding/collapsing triangles below a layer where the options for that effect may be adjusted.



That's the gist of a timeline interface. Stack layers and stack and stack... the nice thing is that it makes it easier or at least somewhat more intuitive to apply images, layers, etc. to a certain range of time in the composite (or composition as After Effects labels it).

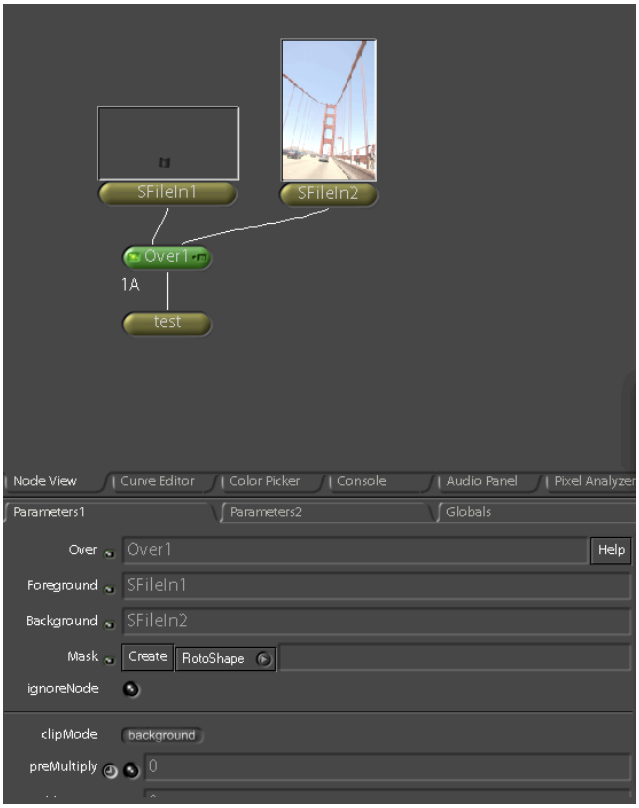
Node/Graph-based Approach

Shake has a predominately graph-based interface, although scripts may be written in Shake Script and there is a timeline view, too.



Notice the graph and nodes in the upper-right. This is the main working area. The left contains the current view of which node is selected. The bottom left contains selection tabs for operations, layers, image transforms, etc. The bottom right contains the parameters for whichever node is selected.

In this example, it's a very simple "Over" being performed. The background plate (imported with a File-In node) is connected to an "Over" node chosen from the Layer tab on the left. The foreground element (also imported with a File-In node) is also connected to the "Over" node. When the "Over" node is chosen, its properties show below in the bottom right. Notice the "premultiply" option selection.



Here's the help entry in Shake for "Over".

Over

Function
The **Over** function is the main compositing node of Shake. This function places one image over another, according to the matte of the foreground image. Images are assumed to be premultiplied. You can use a **MMult** on an input node if it is not premultiplied, or you can toggle the preMultiply parameter to 1.

Parameters	Type	Defaults	Function
clipMode	int	1	Toggles between foreground (0) and background resolution (1)
preMultiply	int	0	When this is on (1), the foreground image is multiplied by its alpha mask. If it is off (0), it is assumed the foreground image is pre-multiplied.
addMattes	int	0	When this is toggled on (1), the mattes are added together for the composite.

Synopsis

```
image Over(
   image Foreground,
   image Background,
   int clipMode,
   int preMultiply,
   int addMattes

);
```

Script

```
image = Over(
  Foreground,
  Background,
  clipMode,
  preMultiply,
  addMattes
);
```

Command Line

shake -over image clipMode preMult

See Also
Under, Screen, Inside, Atop

Notice that the help entry lists all of the parameters, describes their function, and also lists the script command for the same operation.

Here are some of the other Layer nodes that may be selected:

They should look pretty familiar, showing many of the operations that we've discussed so far. Over will still be used the most, though!

Now that we've covered image manipulation and compositing theory and checked out how these things are performed, try opening After Effects (or Shake if it's installed, yet) and performing one of these composites. You can use your material from your first homework assignment, or you can use the Maya object you made in homework #3. (In Maya, you can render your alpha channel by checking the tab under render settings. If a .tiff is rendered, the alpha channel is included -- a good 'ol premultiplied image ready for importing to AE or Shake.)