A Change of Scene

Andrew Glassner

Films and television share a unique language for visual storytelling. One feature of this language lets us move through space and time in ways that are impossible in real life. For example, in a stage play, there needs to be some continuity of time and place: if a character is in the kitchen at one moment, he or she can't be behind the wheel of a speeding car in the next moment. But this kind of abrupt transition happens all the time in films.

The simplest and most familiar way to get from one scene to another is via the *cut*. To create a cut, you can literally take two pieces of film, cut them at frame boundaries, and tape them together. In a film projector, a new frame is displayed every 1/24 of a second, so it takes no longer than that to get from the kitchen to the car.

The cut belongs to a general class of *transitions*—visual effects that form the boundary between different scenes. Let's say that we're watching scene A and we want to go to scene B. If the transition takes many frames, then at each step along the transition we conceptually take the appropriate frames from A and B and combine them to create a new image.

The cut is the simplest transition. It takes zero frames to execute and simply stops showing A and starts showing B. Perhaps the next most familiar transition is the *dissolve* (also called the *cross-dissolve*), where we smoothly blend from A to B. To dissolve over several frames, simply scale the appropriate frames from A and B and adds them together. The scale factor on A goes from 1 to 0 while the scale on B goes from 0 to 1. Typically the dissolve eases along, so that it starts slowly, runs faster, then finishes up slowly again.

A popular variation on the dissolve is the *fade to black*, which is just a dissolve where the B scene is a black frame.

Many video-editing programs ship with a collection of ready-to-use transitions. In this column I'll discuss the field and then present some transitions that I cooked up.

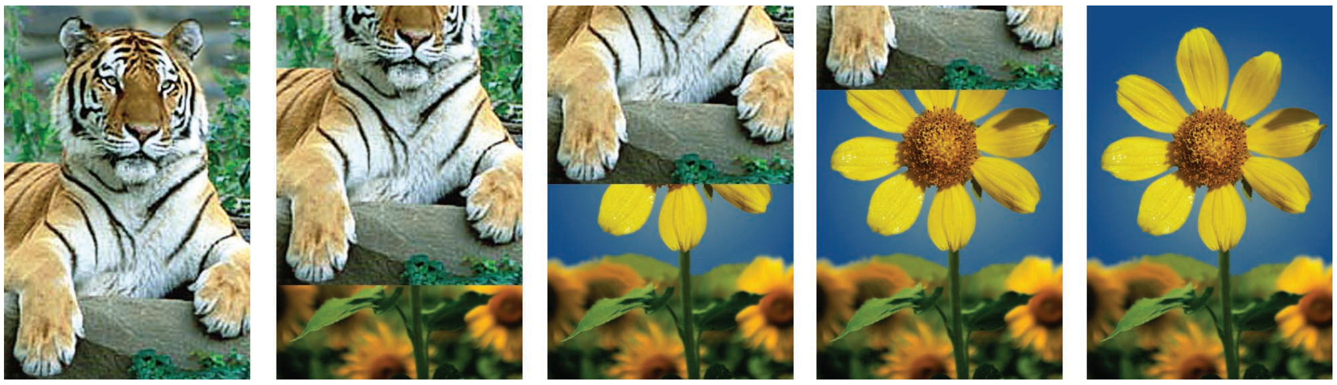
Wipes

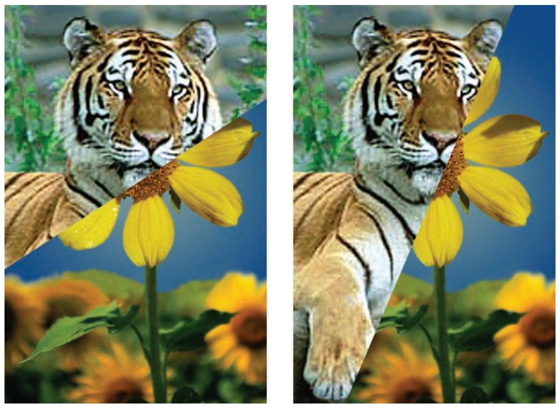
A general class of transitions is the *wipe*. A wipe is a recipe for gradually revealing parts of B over time. In many systems, when the wipe calls for a pixel of B to replace the corresponding pixel of A, it doesn't replace it immediately, because a snap from one color to a another would be distracting. Instead, the wipe triggers a multiframe dissolve for that pixel.

Referring to the different frames to be used at each moment in a transition makes for busy notation and language. To keep things simple in the rest of this column, I'll use constant images for A and B: this would be equivalent to a transition between still frames. In practice, when creating a transition between motion sequences, you'd pull the corresponding frames from A and B at each moment and apply the changing transition to them.

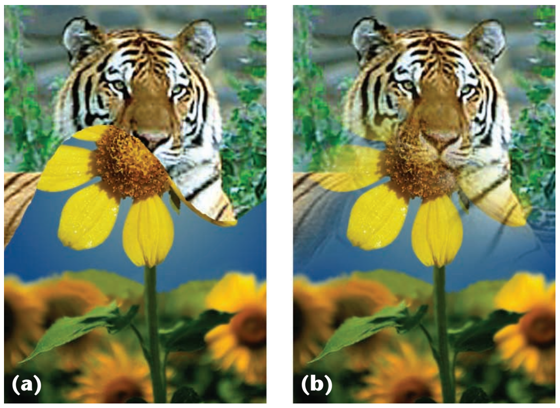
The most common wipes are probably the simple geometrics. Let's look at a simple vertical wipe, as in Figure 1. The effect is as though frame A was simply pulled up, revealing B underneath. An easy way to think about this class of wipes is to imagine that a simple black-and-white *mask* controls them.

Such a mask could be represented by a card painted white and black. For each white pixel in the mask, we'll

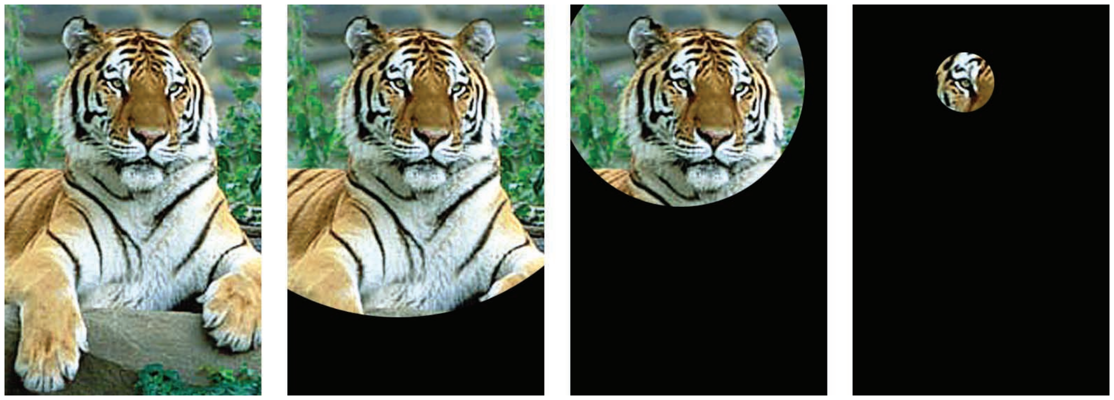

1 A simple vertical wipe.



2 A wipe at an angle.



3 (a) A wipe with an edge that isn't straight. (b) The edge need not be crisp.



4 The iris-out transition.

use the corresponding pixel from frame A, while black pixels direct us to pixels from frame B. To make a simple vertical wipe, you paint the card white on the top and black on the bottom and start with the top half occupying the entire frame. Then pull the card up, so that the boundary moves upward, eventually leaving you with a black image.

You can rotate the card into other orientations, so you can wipe in any direction, as in Figure 2. You can also change the boundary's shape from a straight edge to something curved, or diffuse it from such a sharp boundary to a smoother one, as in Figure 3. In this case, shades of gray represent interpolations of the corresponding pixels from A to B, just like the texture masks used in 3D modeling. In this approach, you might not need to trigger a transition at each pixel, since the gray value itself provides the smooth change from one image to the next.

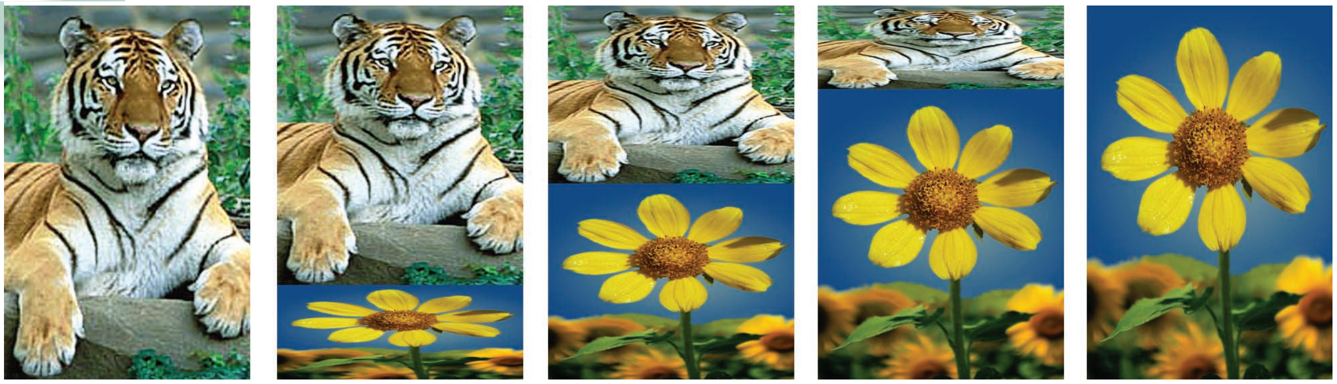
Another fun trick is to change the scale of the mask over time. This creates a class of effects called *iris* transitions. A famous iris transition is a circle that shrinks smaller and smaller around the lead character, revealing a black B scene, as in Figure 4. This is also called an *iris out* effect. People sometimes use the iris' shape to indicate something about the scene's emotional content. For example, it's now a cliché to iris-out at the end of a romantic movie with a heart shape around a pair of newly joined lovers.

These masks can change color over time. Typically this occurs by making the mask pixels strictly darker as the transition proceeds. You could also lighten the pixels during the transition, so bits of A fade away and then reappear. In this way, we can make A appear to "twinkle." But generally, the transitions are strictly one way, each pixel going from A to B. Any geometric pattern that you can program can be used for a transition. Common examples include Venetian blind effects, pinwheels, diamonds, checkerboards, page turns, spinning and growing iris shapes, and even irises that morph in shape from the beginning of the transition to the end.

Distorted wipes

The next more general class of wipes involves distorting images A and B as well as revealing them. For example, let's return to the vertical wipe in Figure 1. During the transition, we're basically splitting the screen into two rectangles, with A on the top and B on the bottom. We could vertically scale each image so that it fits into its allotted rectangle. Figure 5 (next page) shows how this would look. Of course, we can make the warps fit into any shape, not just rectangles. For example, image B can appear to bubble up from the bottom of the frame, it can drop and expand like a raindrop, and so on.

Once we're willing to distort the images, the genie is out of the bottle and all kinds of things become possi-



5 A simple wipe where I warped each image to fit the available region.

ble. Most of them, unfortunately, aren't useful in practice. Remember that the purpose of the transition is to carry the viewer's attention from one scene in the story to the next. Except in special circumstances, if the viewer is actively aware of the transition itself, then that transition at that time must be considered a failure. The transition advances the story. There's nothing worse for a storyteller than to have the audience remove themselves from the narrative flow and start marveling at the technique, because then you've lost their emotional connection to the work. Cool transitions have their place, but like any special effect, they must be carefully chosen and used so that they match and support the story, and never overwhelm it.

With that in mind, it's worth noting that audiences are remarkably adaptive. Consider the cut itself. There's nothing resembling a cut in daily experience. But we've come to accept it, and modern audiences aren't thrown in the slightest when a film cuts from one scene to another. Wipes are just as familiar and unobtrusive, but a little less bizarre: a vertical wipe is like a relative of a rising curtain, and a horizontal wipe is reminiscent of an opening door revealing a room beyond. I can't think of anything outside a house of mirrors that physically corresponds to the transition in Figure 5, but that doesn't mean it should be always avoided. For example, this transition could be used to reinforce the story's emotional content if the characters in scene A are feeling trapped and pressured and those in B are feeling a growing sense of possibility and power. Then the visual images are harmonious with the story itself. Of course, transitions can (and should) be used in much more subtle ways. But the point is always to ensure that the effect never interferes with the story. If the effect can resonate with the story or emphasize the emotional mood, so much the better.

One interesting version of distorted wipes is the class of 3D effects. Suppose that we painted scene A on one side of a box and scene B on an adjoining face. Then we could rotate the box to create the transition. This is something like the zoom-to-fit of Figure 5, but if there's perspective the effect can look unmistakably like a rotating box rather than two images being scaled. You could implement this transition by texture mapping a cube and spinning it, and some video editors do it just that way. Others fake the effect using a variety of tricks, including prerendered pixel-mapping tables.

Now that we're in 3D, we can treat scene A as though it was on a sheet of paper and cause it to flutter away, revealing B beneath it. Or A and B could be on opposite sides of a rotating cylinder. Anything you can do with two texture maps in a 3D scene—so that you start with A and end with B—is fair game for a transition.

Special dissolves

A special form of distorted dissolve is the *morph*. The idea is to create a time-varying warp that takes frame A into frame B. Typically people move the geometry of the pixels of frame A around so that features in one picture (such as eyes or arms) move to similar features in frame B. While the pixels move from one place to the other, they also change color, flowing from the color in image A at the starting location to the color in image B in the ending location.

Another special dissolve is the *no-change dissolve*. Suppose that scene A ends with the hero hanging up her telephone in her kitchen and scene B begins with the villian picking up her own phone in her living room. In film production, we might want to emphasize the role the phone plays in both character's lives by tying together the two scenes visually: we could make that last frame of A match the first frame of B as carefully as possible. That means we use a close-up of the same phone, the same lighting, the same camera position, and so on. Of course, the two phones are in different physical locations and so the match is imperfect, creating a slight wobble in the image as we dissolve from one phone to the other. This is the no-change dissolve. One might think that computer animation can improve on this transition by actually matching the two shots perfectly.

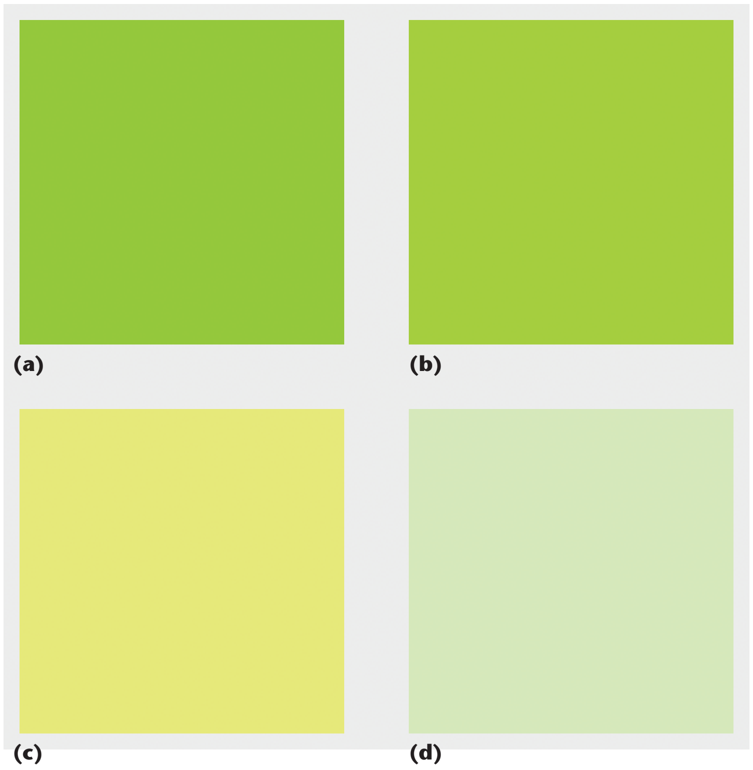
But wait! In fact, that slight wobble between scenes is very important: it's what alerts the audience that a transition has, in fact, occurred. Remember, the whole idea is to make a story point about the characters and their relationship to the telephones and not to trick the audience. If the two shots are really perfectly matched, then the audience will certainly be confused. Think of the sequence: a shot of the hero sadly placing the phone in its cradle, a zoom into the phone, and then a zoom out of the phone to reveal the villian's home. What happened? People will figure it out, but that moment of disorientation is rarely to the story's advantage. The slight change in the phone's visual appearance that's inevitable in film is actually vital to keeping the audience engaged. In good animation, no-change dissolves have a small amount of change in them—just enough to cue the audience that a transition has occurred but that the two scenes are very similar.

Color-driven transitions

I thought it would be interesting to look at some transitions that are based on the colors in the images themselves rather than on some additional geometry that was laid over them or used for time-varying distortions.

These transitions all use color information to compute the image and modulate the colors to blend from one image to the next. Because we're adjusting the colors, it's important to work in an appropriate color space.

Most of the time in graphics we save our images in an RGB format. Let's assume that these are each floating-point numbers between 0 and 255. For specificity in the discussion, let's pick a bright green color C, given by RGB (127, 255, 64), shown in Figure 6a. Now suppose that we want to slowly move C to black. We could do so by scaling each of the RGB values over time. If α is a number that runs from 1 to 0, we can just compute C′ = (α 127, α 255, α 64).



6 **(a) Our initial green color (127, 255, 64). (b) The clamped version of the brightened color (152, 255, 77). (c) The original color scaled and clamped to get the desired luminance: (236, 255, 119). (d) The original color adjusted in the hue, saturation, brightness (HSB) space, with RGB = (211,255,189). This has the same luminance as (c) but hasn't shifted in hue. The background is gray with our desired luminance: (234, 234, 234).**

Now suppose we want to make the color brighter. A common way to estimate a color's brightness is to take its luminance. Constants for computing the luminance vary a little bit depending on your specific hardware, but a common formula is L(RGB) = .3R + .59G + .11B. The luminance of C by this formula is about 195. Suppose now we want to raise the luminance to 234. The obvious way is to scale the components as before. The value that does the trick is about α = 1.2, giving us C′ = (152, 306, 77). Uh-oh. Since our monitor can't show any green value above 255, we have to clamp the green component from 306 to 255. Figure 6b shows the result. But the luminance of the clamped color (152, 255, 77) is about 204, not 234. Our pixel is only about 5 percent brighter, not the 20 percent we wanted. If we care about how bright and dark our pixels are (and in a transition, that's very important) then we have to address this problem.

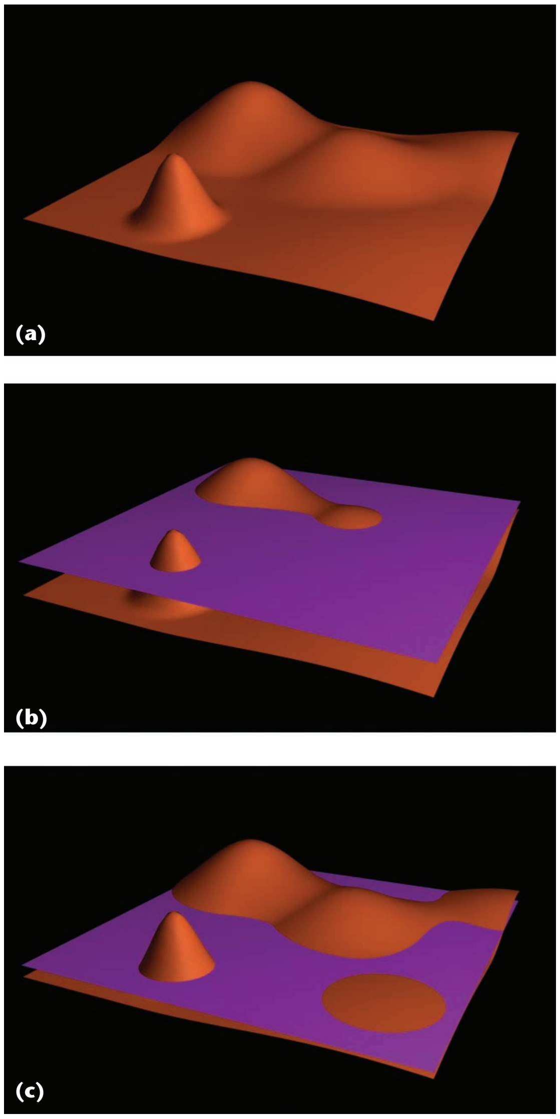
The obvious answer is to hold the green value fixed at 255 and raise red and blue by equal percentages until we get the desired luminance. In other words, we crank up the three values, clamping the overflow, until L = 234. The result is (236, 255, 119), shown in Figure 6c.

But this repair has actually made things worse. As Figure 6a shows, our clamped and scaled color has moved from green to yellow. Because the green component is less dominant in the clamped version, the new color looks yellow. This color shifting is very noticeable. Unless you're going for a weird, color-distorted look, such color shifts are unacceptable during a transition.

The cure is to work in a different color space. We have a number of choices from which to choose. One nice compromise in terms of efficiency and performance is the hue, saturation, brightness (HSB) space. The relatively simple technique for converting between RGB and HSB is available in almost every graphics textbook, and you can even get implementations freely on the Web.

In HSB, our original color C = HSB(100, 75, 100). To make our color brighter, we only need to decrease the saturation; this has the effect of pushing the color toward white while retaining its hue. Note that for simplicity, I cooked the example so that the brightness is already at a maximum. In practice, you'd want to address both brightness and saturation while adjusting the color. To get to a luminance of 255, we decrease the saturation to 26. To get the equivalent RGB values, we run the conversion the other way: HSB(100, 26, 100) = RGB(211, 255, 189). This color also has the desired luminance of 234, but as Figure 6d shows, it's a very light green rather than yellow. This is a much better result.

So if we'd just scaled up our original RGB color to make it brighter, it would have indeed become brighter, but it would have changed hue to a yellow. Working in HSB



7 (a) A height plot for the luminance of image A. (b) A slicing plane moving from the highest peaks to the valleys. All pixels above the plane are replaced with B pixels. (c) The plane a bit further down.

solved the problem and kept the color green while making it lighter. We could have achieved similar results using other popular color spaces such as hue saturation value (HSV) or hue lightness saturation (HLS). The perceptually uniform L*a*b* space is probably the best for this kind of work, but it's much more expensive to compute.

Luminance planes

Let's look at three transitions based on the brightness of the two images. The first one is pretty simple. Think of the luminance of image A as a height field, as shown in Figure 7a . Create a plane parallel to the base and set it just above the highest point of A. Now push the plane downwards, as in Figure 7b. Each time a pixel of A gets sliced off by the plane, start its transition to the color in B. Figure 8 shows images from the transition that results if you just replace each cut-off pixel in A by its counterpart in B. In practice, I transform these colors smoothly by kicking off a linear blend when the pixel is first sliced. I also pass the blend through the ease curve that I presented in the last issue (March/April) to make the transition look nice and smooth.

This technique has lots of variations. One is to take inspiration from the iris effects and use a shape other than a plane to cut the height field. A hemisphere, a star, a cone, and other shapes would each have their own look. If you use one of these other shapes, you can try rotating and scaling it or even changing its shape entirely as it moves downward.

Black bounce

In a typical fade-to-black transition, we darken all the A pixels until they become black and then crank them all back up again until they reach their B values. In other words, the whole image goes uniformly dark and then bright again.

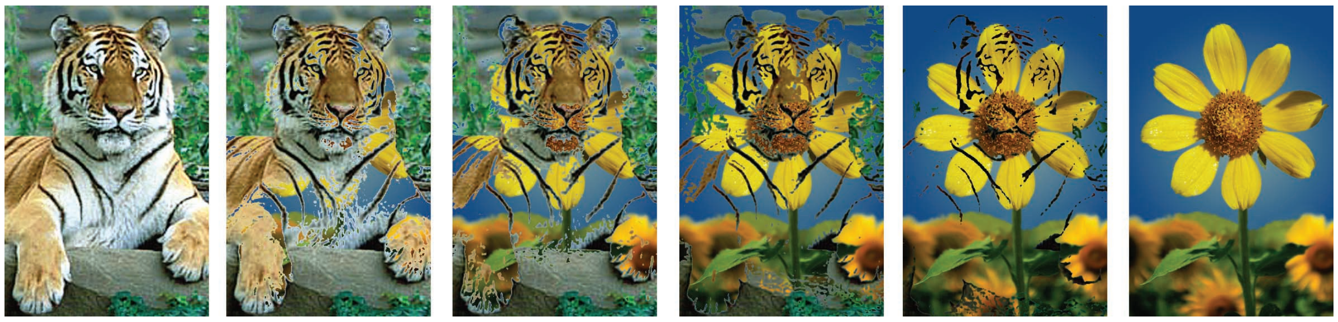
Notice that in general, each pixel in the image is fading down (and then up) at the same relative rate but at different absolute rates. For example, if a transition takes five frames to go from A to black, then five frames to get to B, then each pixel will lose (or gain) 10 percent of the total distance it has to travel at each step. For simplicity in this discussion, let's suppose our pixels are gray (so R = G = B). Now consider a gray pixel that begins with a value of 50 at A and ends with a value of 150 at B, for a total of 200 units (50 to 0, and then 0 to 150). The pixel will change by 200/10 = 20 units at each step. But a brighter gray pixel with starting and ending values of 200 will cover 400 units, so it will change by 400/10 = 40 at each step. So they have the same relative speeds (10 percent per step) but different absolute speeds.

What if we turn this around and give each pixel the same absolute speed? Consider some particular pixel P, with starting and ending gray values P_A and P_B. We need to get from P_A to 0 and then up to P_B over the course of the transition. Suppose that the transition takes a total of 10 frames. This pixel will need to change by $(P_B + P_A)/10$ at each step.

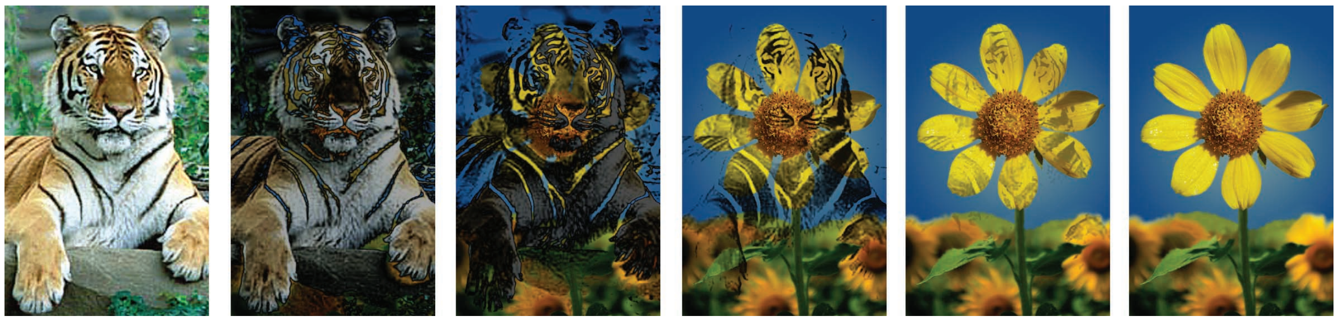
Each pixel will have a different range $P_B + P_A$. Let's search the images and find the pixel Q with the biggest total distance to travel. This distance tells us to adjust the gray value of Q by $Q\Delta = (Q_B + Q_A)/10$ at each step (first by subtracting that amount from Q_A, until we get to black, and then adding it back in, until we get back up to Q_B). Now let's apply that same transition rate to every pixel, regardless of where it starts. So for some general pixel P, we start at P_A, subtract $Q\Delta$ at every step until we get to black, and then add in $Q\Delta$ until we've made it up to P_B. Because Q has the largest distance to travel, all the other pixels will reach their goals sooner (or certainly no later) than Q.

Figure 9 shows this transition. I used the HSB space to adjust the apparent intensity of each pixel so that it starts out at its initial color, fades to black, and then comes back up to the final color.

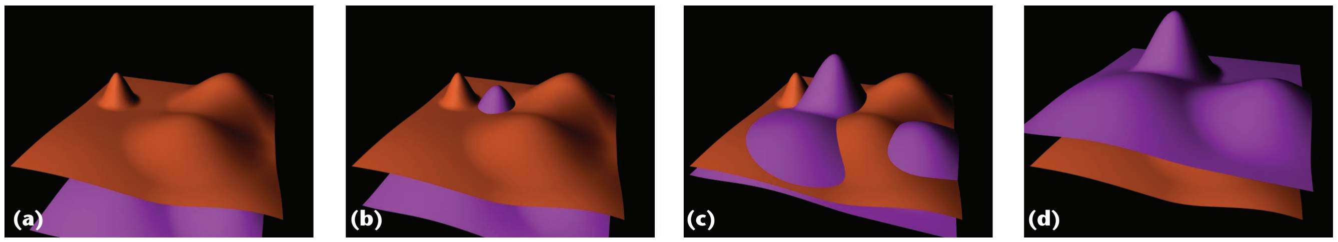
This transition has a silky, smooth look to it. Sometimes it almost looks as though the image is being inverted, like a piece of color negative film. I think that's because of the different rates involved in changing bright and dark regions. For example, regions that are dark in A but bright in B become bright very quickly, while pixels that are bright in both need time to catch up—this delay is part of what gives the effect its look.



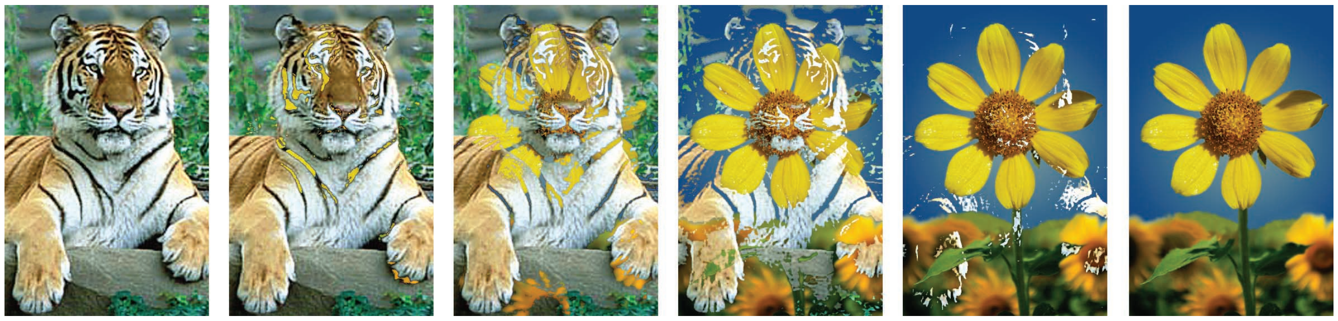
8 A transition using the technique of Figure 7.



9 A transition using constant speed for each pixel change.



10 (a) The start of the transition, showing the height plot for A (orange) above that for B (purple). (b) B moves up. Anywhere that B rises above A the corresponding pixels are turned to B. (c) B moving farther up. (d) The end of the transition, with B above A.



11 A transition using the technique of Figure 10.

Moving mountains

Returning to our interpretation of pixel intensities as height fields, what if we include the height fields for both A and B in our transition?

Figure 10 shows the basic idea. We place B under A, so that the highest point of B lies just below the lowest point of A. Now we start moving B upward, as in Figure 10b. Anytime a pixel of B pokes itself above that of A, we start that pixel on its transition from A to B.

Figure 11 shows a transition created by this approach. It has a nice organic feel to it. I think this is because the technique seems to thrive on contrast: when B is bright and A is dark, B will poke through early. Often, the bright parts of an image are the important ones, so we get to see the most important parts of B showing through the least important parts of A and then gradually B develops from there.

Web Extras

To view animations of Figures 8, 9, and 11, visit CG&A's Web site at http://computer.org/cga/cg2001/g3toc.htm and click on the following links:

- Animation of Figure 8: A transition that results if you replace each cut-off pixel in scene A by its counterpart in scene B. See http://computer.org/cga/cg2001/extras/g3086x1.avi.
- Animation of Figure 9: A transition using constant speed for each pixel change. See http://computer.org/cga/cg2001/extras/g3086x2.avi.
- Animation of Figure 11: With scene B underneath scene A, this transition moves the luminance of scene B upward. See http://computer.org/cga/cg2001/extras/g3086x3.avi.

Once this is up and running, we can explore many variations. As in the previous examples, we can translate, rotate, and scale the height field for B as it moves. In fact, it could translate in from the side, rising up from below while wiping. This creates an interesting ripple effect in A. Note that any motion of the height field for B need not have any correlation to any motion of image B itself with respect to A.

Summing up

There are a million interesting transitions for us to discover. I found that it's a lot of fun to cook up transitions and implement them. I tried a couple of dozen over the course of writing this column. The last three I explained here are the ones that actually passed muster—the others were visually interesting, but I didn't think that anyone would ever use them in practice.

Of course, the images that illustrate these transitions can only suggest the feeling. Because most of the effect depends on the motion and the nature in which the pictures change, a real feel for these transitions can only come after seeing them in action. (See the "Web Extras" sidebar for information on where you can view short motion clips of these three transitions.)

Inventing transitions is like inventing filters for image-processing programs. It's easy to dream up possibilities, but often it's a case of a solution without a problem. As I mentioned earlier, if a transition is too flashy or too cool, then it threatens to interrupt the audience's attention from the story. Unless the transitions themselves are part of the story, oddball transitions should be used sparingly, like unusual spices in traditional dishes. The story is the reason for making the film—the job of the transition is to help tell it. ■

Readers may contact Glassner by email at andrew_glassner@yahoo.com.

Coming Soon

International Workshop on Volume Graphics
Stony Brook, New York, USA
21-22 June 2001
The program committee of the workshop is seeking research papers and proposals for panel discussions concering all aspects of volume graphics. Visit http://www.cs.sunysb.edu/~vg01 or write to vg01@cs.sunysb.edu for more information.

Graphics Hardware
Los Angeles, California, USA
12-13 August 2001
This workshop is an inclusive forum for the entire graphics hardware community and brings together researchers, engineers, and architects. This year's workshop will be held in conjunction with Siggraph 2001. Visit http://www.graphicshardware.org or email info@graphicshardware.org for more information.