Behavior Research Methods, Instruments, & Computers
1996, 28 (4), 607-610

Cinematica: A system for calibrated, Macintosh-driven displays from within Mathematica

JOSHUA A. SOLOMON
Institute of Ophthalmology, London, England

and

ANDREW B. WATSON
NASA Ames Research Center, Moffet Field, California

Cinematica is a minimal system for producing calibrated grayscale movies on an Apple Macintosh computer from within the Mathematica programming environment. It makes use of the ISR Video Attenuator and the VideoToolbox software library developed by Denis Pelli. By design, Cinematica provides a very low-level interface to the display routine. Display instructions take the form of a list of pairs (image index, colormap index). The philosophy is that programming is much easier in Mathematica than in C, so we reserve the complexity for Mathematica. A few simple examples are provided.

Precise measurement of visual function requires precise control of contrast and display timing. Recent technological advances, in both hardware and software, have made it possible for vision scientists to make such measurements with the use of ordinary computer equipment.

Exploiting an idea by Watson et al., (1986), Pelli and Zhang (1991) have designed a device called the Video Attenuator, which provides precise contrast control by enabling ordinary color video cards (with 8-bit DACs) to produce monochrome displays with 12-bit accuracy. Pelli has also assembled a collection of low-level C routines (the VideoToolbox) that, when used in conjunction with a Video Attenuator, are capable of producing displays of grayscale movies from Macintosh computers with the precision that vision science demands.

C is a low-level computer language. As such, it can often take considerable effort to write a working program that will produce the desired image displays, even with the VideoToolbox. We wanted a high-level, yet versatile, language with which both experts and novices could quickly assemble vision experiments. Mathematica (Wolfram, 1991) is a versatile, high-level computer language, but, by itself, it is not capable of producing calibrated visual displays. Thus we have created Cinematica, a system integrating the precision of the Video Attenuator and VideoToolbox with the accessibility of Mathematica.

The distribution of Cinematica contains three important items. The first item, Cinematica.m, is a small package of Mathematica routines which provides the interface (via MathLink) to the second item, an application which, by utilizing select features of the VideoToolbox, directly controls image displays. The third item, Cinematica.ma, contains the Cinematica Tutorial, excerpts from which appear below. Cinematica requires a minimum of 4 Mb unused RAM. One additional byte is required for every pixel.

The tutorial is written as a Mathematica notebook in which Mathematica commands are indented, in bold typeface. The tutorial requires a computer with two monitors. On one monitor, the user follows the tutorial, executing each command with a keystroke. The displays appear on the other monitor. These examples demonstrate the ease with which one can now control the display of research-quality visual stimuli from within Mathematica.

Correspondence should be addressed to J. A. Solomon, Institute of Ophthalmology, Bath Street, London EC1V 9EL, England.

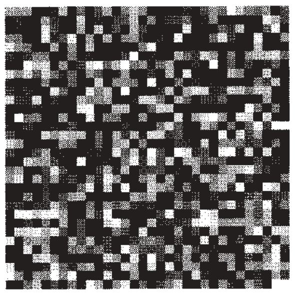
Tutorial

Display a single image. The simplest possible use of Cinematica consists of three steps: (1) load an image, (2) load a colormap, and (3) display the image with the colormap. We illustrate these steps.

We first create in Mathematica an image consisting of a single frame of noise, and display it on the console for verification.

```
noise =
   Table[Random[Integer, {2, 254}], {32}, {32}];

ListDensityPlot[noise,
   Frame->False, Mesh->False];
```



Next we load the image into a type of Macintosh memory called a GWorld. Each image is associated with an arbitrary integer index.

```
MovieToGWorld[1, noise];
```

Next we create one colormap. (A colormap is a table mapping pixel gray level to pixel luminance.) This also gets an arbitrary integer index. We use the defaults for the other parameters. This will get us a linear, unit contrast colormap.

```
ColorMap[1];
```

Next we display the image using the colormap. This is done by calling **GWorldToScreen** with a single argument consisting of a display list that consists of {image index, colormap index} pairs. In this case, the list contains only one pair.

```
GWorldToScreen[{{1, 1}}];
```

One should see a 32 × 32 pixel noise image on an otherwise dark background.

Changing the background luminance. GWorldToScreen copies images to an area of video memory that we will call the screen. Initially the entire screen is set to the gray level zero. To create a uniform background of a luminance other than zero, we create a new colormap. This call to ColorMap is a little more complicated. We specify that the colormap index is 1 (note that we may freely reuse colormap and image indices), the contrast is 1, and the image gray levels will range between 2 and 254; and we assign the remaining 3 gray levels to specific relative luminances that correspond to the mean, minimum, and maximum. (*The gray level values 1 and 255 will not be used in this excerpt from the tutorial*.) Thus our uniform background of 0s will map to a relative luminance of 0.5 (the mean luminance). Note also that symmetric placement of minimum and maximum gray levels means that 128 will also map to the mean luminance.

```
ColorMap[
   1, 1, 2, 254, {{0, .5}, {1, 0.}, {255, 1.}}];
```

Now we display the noise image with the new colormap.

```
GWorldToScreen[{{1, 1}}];
```

Display a single image for a specified duration. Here we load an image of zeroes.

```
MovieToGWorld[2, Table[0, {32}, {32}]];
```

Here we load a zero-contrast colormap.

```
ColorMap[
   2, 0, 2, 254, {{0, 0.5}, {1, 0.}, {255, 1.}}];
```

Here we display the blank image with the zero-contrast colormap

```
GWorldToScreen[{{2, 2}}];
```

It is now a simple matter to show the noise image for a fixed duration of 32 frames.

The duration of this display will depend on the refresh rate of the display monitor. For example, if the refresh rate is 67 Hz, this 32-frame display will be 0.48 sec long.

```
GWorldToScreen[
   Join[{{2, 2}}, Table[{1, 1}, {32}], {{2, 2}}]];
```

Note that not all video drivers are capable of redrawing the entire screen at every available refresh rate. TimeVideo (an application within the VideoToolbox) can determine the display rate accuracy of any Macintosh-driven display.

A moving image. Cinematica contains no special provisions for making images move. Instead, moving images are created in advance as a sequence of frames. In this example, we create the frames by scrolling the noise image created above. Note that the first frame is assigned index 3, and subsequent frames increment by 1.

```
MovieToGWorld[
   3, Table[RotateRight[noise, k], {k, 32}]];
```

Now we can show something. We create a display list that consists of {image index, colormap index} pairs. This list is given to **GWorldToScreen**.

```
GWorldToScreen[Join[{{2, 2}},
   Table[{k, 1}, {k, 3, 34}], {{2, 2}}]];
```

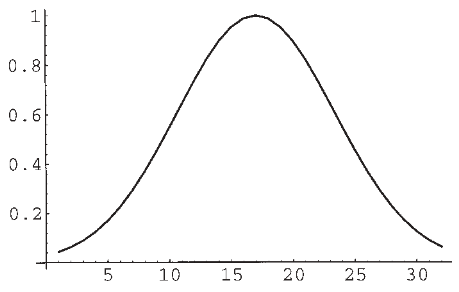
One should see a 128 × 128 gray rectangle in which is embedded a square of noise that scrolls vertically for a duration of 32 frames.

Varying contrast over time. Now we create a set of 32 colormaps that describe a Gaussian variation in contrast. First we create a sequence of contrast values (the time list). This Gaussian has a time scale of 16 frames and is centered on the 17th frame.

```
gauss =
   Table[N[Exp[-Pi*(((t - 17)/16)^2)]], {t, 32}];
```

We plot the waveform for verification.

```
ListPlot[gauss, PlotJoined->True];
```



We use the time list to load a set of 32 colormaps, one for each contrast. We start at index 3, to leave our unit- and zero-contrast colormaps in place.

```
ColorMap[3,
   gauss, 2, 254, {{0, 0.5}, {1, 0.}, {255, 1.}}];
```

We create a display list that associates our static noise repeatedly with the elements of the list of colormaps. As usual, we return to the uniform field at the end of the display.

```
dlist = Join[{{2, 2}},
   Table[{1, n}, {n, 3, 34}], {{2, 2}}];

GWorldToScreen[dlist];
```

Loading image files. Above we have emphasized loading of images and movies directly from Mathematica expressions. Occasionally one may wish to load an image file that was created previously by Mathematica or some other source. To do this we use the function FileToGWorld. For illustration, we first write an image to a file.

```
WriteBinary["Cinematica.test",
   Flatten[noise], ByteConversion->Identity];
Close["Cinematica.test"];
```

Then we write the file to the GWorld, specifying the image dimensions.

```
FileToGWorld["Cinematica.test", 4, 32, 32];
```

Then we verify that it displays correctly.

```
GWorldToScreen[{{4,1}}];
```

Availability

Cinematica and the VideoToolbox are free on the World Wide Web. The URL is http://vision.arc.nasa.gov/mathematica/Cinematica/Cinematica.html. The ISR Video Attenuator is available from Syracuse University. Consult the VideoToolbox for ordering instructions. Mathematica is available from Wolfram Research Institute, Champaign, IL. The authors have no proprietary interest in any of these programs.

Conclusion

The preceding should convince readers that Cinematica provides a simple means of displaying grayscale stimuli for vision experiments. Readers should also appreciate the ease with which programs can be assembled in Mathematica. Since each line of a Mathematica program can be executed and verified as soon as it is typed, we believe that learning to program in Mathematica should be easier than learning to program in C. Vision experiments in our laboratories are driven by Macintosh programs written entirely in Mathematica. Readers wishing to learn about procedural aspects in addition to image display are encouraged to contact the authors. A summary of Cinematica functions is given in Appendix A. Code for an example experiment is provided in Appendix B.

REFERENCES

Pelli, D. G., & Zhang, L. (1991). Accurate control of contrast on microcomputer displays. *Vision Research*, **31**, 1337-1350.

Watson, A. B., Nielsen, K. R. K., Poirson, A., Fitzhugh, A., Bilson, A., Nguyen, K., & Ahumada, A. J., Jr. (1986). Use of a raster framebuffer in vision research. *Behavior Research Methods, Instruments, & Computers*, **18**, 587-594.

Wolfram, S. (1991). *Mathematica: A system for doing mathematics by computer* (2nd ed.). New York: Addison-Wesley.

APPENDIX A
Summary of Cinematica Functions

CinematicaOpen[]: Must be called prior to any other Cinematica Functions.

MovieToGWorld[imageIndex_,movie_]: Load a grayscale image or movie into GWorld(s). The user must ensure that the pixels are integers between 0 and 255. Each frame of the movie has its own GWorld. The index of the first frame's GWorld is imageIndex. The index of the n-th frame's GWorld imageIndex+n, and so on.

ColorMap[cMapIndex_,contrast_:1.,grayMin_:0,grayMax_:255,cMapList_:{{0,.5}},gamma_:1.]: Calculate a lookup table with index cMapIndex. contrast must be a real number between -1. and 1., corresponding to the desired proportion of obtainable contrast (when the minimum luminance is close to zero, this will be equal to luminance contrast). grayMin and grayMax refer to the range of gray levels to which the to-be-displayed image has been scaled. Any leftover color map entries can be assigned a luminance manually by setting cMapList={{gray_1, luminance1},{gray_2, luminance2}, . . .}, where luminancei is a real number between 0. and 1. on a linear scale of the available luminances. For technical reasons, cMapList cannot be empty. However, manual luminance assignments to gray levels between grayMin and grayMax are ignored. gamma is the desired exponent of the mapping between gray levels and luminance; if a linear relation is desired, this should be the default value of 1.

GWorldToScreen[displayList_]: Display a movie. displayList should be a list of pairs, {{imageIndex1,cMapIndex},{image Index2,cMapIndex},. . .}, one pair for each refresh of the display. Note that the writing of an image to the screen and the implementation of a colormap cannot be truly simultaneous. Cinematica implements the colormap first. To prevent the brief display of an image before the proper colormap is implemented, the user may wish to begin displayList with a blank image coupled to a zero-contrast colormap.

CinematicaClose[]: Should be called prior to termination of any Mathematica session involving any Cinematica Functions.

FileToGWorld[file_,imageIndex_,cols_,rows_:1,frames_:1]: Load a raw, byte-format, grayscale movie from a file to GWorlds. Each frame of the movie has its own GWorld. The index of the first frame's GWorld is imageIndex, the index of the n-th frame's GWorld imageIndex+1 and so on.

APPENDIX B
Example Experiment

Here is an example of an experiment in which the method of adjustment is used to collect a contrast threshold for a square decrement stimulus of variable size and duration. After each presentation, the observer responds "decrement visible," "decrement invisible," or "decrement exactly at threshold." The contrast of the decrement is adjusted accordingly.

```
<< Cinematica.m

Experiment[size_:64, frames_:16] := (
   CinematicaOpen[];
   ColorMap[1, 0., 1];
   MovieToGWorld[
      blank = 1, Table[0, {size}, {size}]];
   MovieToGWorld[
      stimulus = 2, Table[1, {size}, {size}]];
   GWorldToScreen[{{blank, 1}}];
   displayList = Join[
    Table[{stimulus, 2}, {frames}],
    {{blank, 1}}];
contrast = 1.;
Catch[While[True, (
        ColorMap[2, contrast, 1];
        GWorldToScreen[displayList];
        contrast =
            N[Min[1., contrast*Sqrt[2]^Switch[
                Input["1 (visible)\n\
                    0 (invisible)\n\
                    2 (at threshold)"],
                    0, 1,
                    1, -1,
                    2, (CinematicaClose[];
                        Throw[contrast])]]])]])
```

Experiment[ ]

0.0078125

(Manuscript received August 10, 1995;
revision accepted for publication November 16, 1995.)