Video

Uses Video library

- Must add library to processing
  - Sketch->Import Library->Add Library
  - Select "Video"
  - Click "Install"
- At the top of your code:

```
import processing.video.*;
```

Image Concepts Transfer

- Video is basically a series of images
- Many of the commands used with images transfer directly
  - translate(); rotate();
  - tint();

Live Video

- Must setup and connect camera
  - See book for instructions – varies on machine
- Capture is main video class
  - Capture(this, <width>, <height>, <fps>);
  - *this* is a reference to the current object

```
Capture video;
void setup()
{
    video = new Capture(this, 320, 240, 30);
}
```

Live Video

- Must start the Capture object
  - video.start()
  - Do in setup

Must Read Each Frame of Video

- Option 1:

```
void draw()
{
    if ( video.available() )   //checks if there is a
                               // new frame
    {
        video.read();          //reads the frame
    }
}
```

Must Read Each Frame of Video

- Option 2:

```
//callback triggered when frame is ready
void captureEvent(Capture video)
{
    video.read();    //reads the frame
}
```

Display Frame

- Same as an image:

```
image(video, 0, 0);
```

- Again, all the image commands apply
- Run example

Prerecorded Video (movie files)

- The main object is Movie
- Loading:

```
Movie movie;
movie = new Movie(this, "test.mov");
```

- Playing (activating movie as input source):

```
movie.play();  //plays once
movie.loop();  //loops continuously
movie.stop();  //stops
movie.pause(); //pauses
```

Must Still Read Each Frame

- Option 1:

```
void draw()
{
    if ( movie.available() )  //checks if there is a
                              // new frame
    {
        movie.read();    //reads the frame
    }
}
```

Must Read Each Frame of Movie

- Option 2:

```
//callback triggered when frame is ready
void movieEvent(Movie movie)
{
    movie.read();    //reads the frame
}
```

Movie Functions

- Display (same):

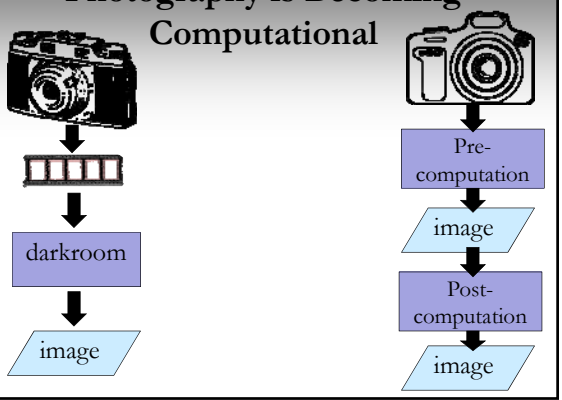
```
image(movie, 0, 0);
```

- Other functions:

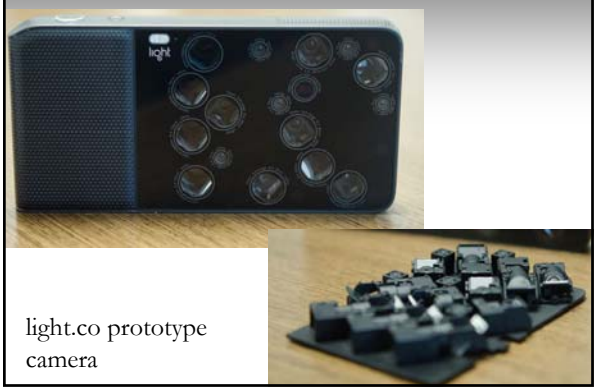
```
movie.duration();
movie.jump(<time>);
```

- Demo

Photography is Becoming Computational



New Cameras



light.co prototype camera

Video Mirrors

- Insert computation into process
  - Display something other than captured pixels
    - Invert color
    - Grayscale
    - Rectangles proportional to the image brightness
- Display image is based on pixel data, but transformed
- BrightnessMirror

Class Problem

- Modify code to create a mirror where:
  - All squares are the same size (80% of block size)
  - The squares are a shade of red corresponding to how bright the original pixel is (black to bright red)

How are cameras used?

- Selfies
- Photography
- Reading QR codes
- Surveillance/security cameras
- Remote monitoring
- Driver assistance
- Motion capture
- Computer input
  - Kinect, Leap, etc.

Video as Input

- Can view camera as an input device!
  - Don't need to display video at all to analyze it
  - Track user's movements or other items in scene
    - Bright spot, flashlight, particular color

Tracking Algorithm

- Idea: Check every pixel to find the pixel that is closest to the color you are trying to track
- Algorithm:
  - Set match color (e.g. red)
  - Set distToMatch large (used to see how close a pixel is)
  - Visit every pixel
    - If pixel is closer to match color
      - Save that location
      - Update distToMatch

Video as Input

- Show demos
  - Basic drawing
  - Drawing a trail
  - Put it all together: combine drawing with video mirror

Background Subtraction

- Separate foreground from background

```
Take a static image of the background at the start
For each new image
    For each pixel
        If dist(background, newImage) > threshold
            Pixel is foreground
```

Code

```
PImage background = createImage(video.width, video.height, RGB);
background.copy(video, …);
```

Shiffman, Ex. 16-12

Codecs

Codecs

- Compression required for video
  - Broadcast video requires more than 100 Mbits/s
  - HDTV requires over 1 Gb/s
- A codec is a particular encoding scheme for storing video
  - Similar to image formats
  - Compression based on statistical structure of data and psychophysical redundancy
  - Can take advantage of temporal nature of data

Codecs

- MPEG-1 (1991)
  - 1.5 Mbits/s
  - Designed for CD ROMs
- MPEG-2, AKA H.262 (1994)
  - 2-15 Mbits/s for DVDs, 19.2 Mbits/s
- MPEG-4 Part 2 (1999)
  - Multimedia content

Codecs

- H.261 (1990)
  - Target rate 64 – 1920 kbit/s
  - Designed for teleconferencing
- H.263 (1996 and on)
  - Very low bit rate, < 64 bits/s
- MPEG-4 Part 10, AKA H.264/AVC (2005)
  - Higher coding efficiency (double MPEG-2)
  - Suitable for many applications

Codecs

- VC-1 (2005)
  - Developed by Microsoft and implemented as WMV 9
  - Performance close to H.264/AVC