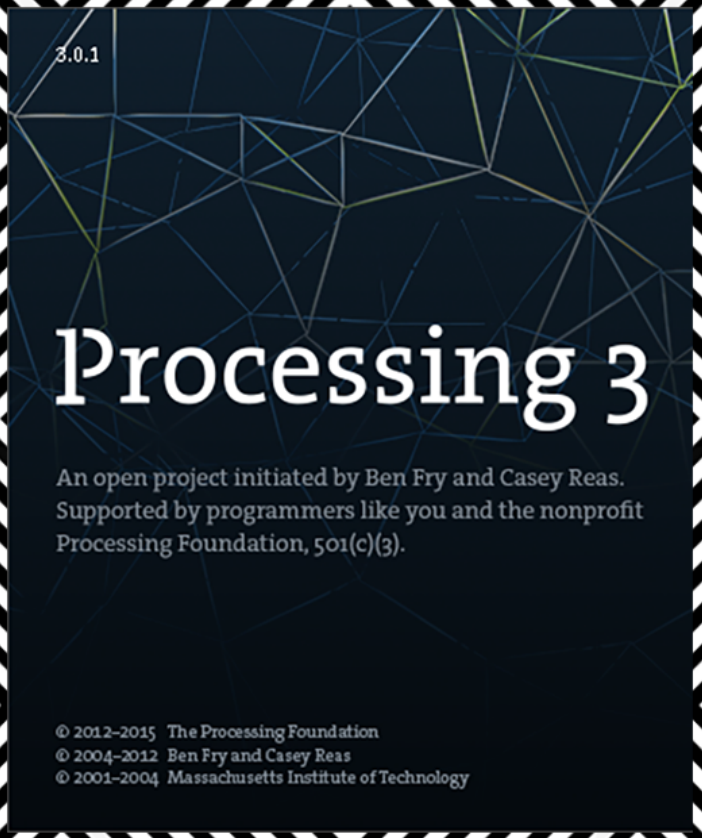
Wichita Women Who Code presents: Introduction to webcam graphics

MakeICT Workshop

January 30, 2015 (1pm-3pm)

Bio – Laura Scholl

- MakeICT / Women Who Code member
- Assistant Prof. Graphic Design and Digital Art Newman University – Art Department
- MSVS from MIT Media Lab
- Feature film career (lighting and compositing):
  - Fifth Element
  - Disney's Dinosaur
  - Harry Potter and the Sorcerer's Stone
  - Narnia – the Lion, the Witch and the Wardrobe


3.0.1
Processing 3
An open project initiated by Ben Fry and Casey Reas.
Supported by programmers like you and the nonprofit
Processing Foundation, 501(c)(3).
© 2012–2015 The Processing Foundation
© 2004–2012 Ben Fry and Casey Reas
© 2001–2004 Massachusetts Institute of Technology

Processing – Download

- https://processing.org/
- https://processing.org/download/

Processing.org Basics

- https://processing.org/reference/
- https://processing.org/tutorials/
- https://processing.org/reference/libraries/
- https://processing.org/reference/libraries/video/index.html

Processing – mini tutorial: functions

```
// setup runs once
void setup() {
}
// draw is an event loop that runs continuously unless you stop it
void draw() {
}
```

Setup and Draw functions

```
void setup() {
  // set the size of the display window
  size(1280, 720);
  // set the background to white note: white, gray and
  // black only require one number
  background(255);
}

void draw() {
  // draw a square at the center of the screen
  rect(width/2, height/2, 200, 200);
}
```

Drawing a rectangle

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
  // yes! Order matters
}
```

Drawing a centered black rectangle

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to black – this fills in any geometry that you draw
  fill(0);
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
}
```

Drawing a centered red rectangle

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to red – three parameters are required for color
  fill(255, 0, 0);
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
}
```

Drawing a centered red rectangle

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to red
  fill(255, 0, 0);
  // turn off the stroke – outline
  noStroke();
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
}
```

Drawing a transparent red rectangle

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to transparent red – add a 4th parameter for transparency
  fill(255, 0, 0, 127);
  // turn off the stroke – outline around the rectangle
  noStroke();
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
}
```

The event loop – noLoop()

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to transparent red
  fill(255, 0, 0, 127);
  // turn off the stroke – outline
  noStroke();
  // stop event loop
  noLoop();
}
```

The event loop – noLoop()

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to transparent red
  fill(255, 0, 0, 127);
  // turn off the stroke – outline
  noStroke();
  // stop event loop
  noLoop();
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  rect(width/2, height/2, 200, 200);
}
```

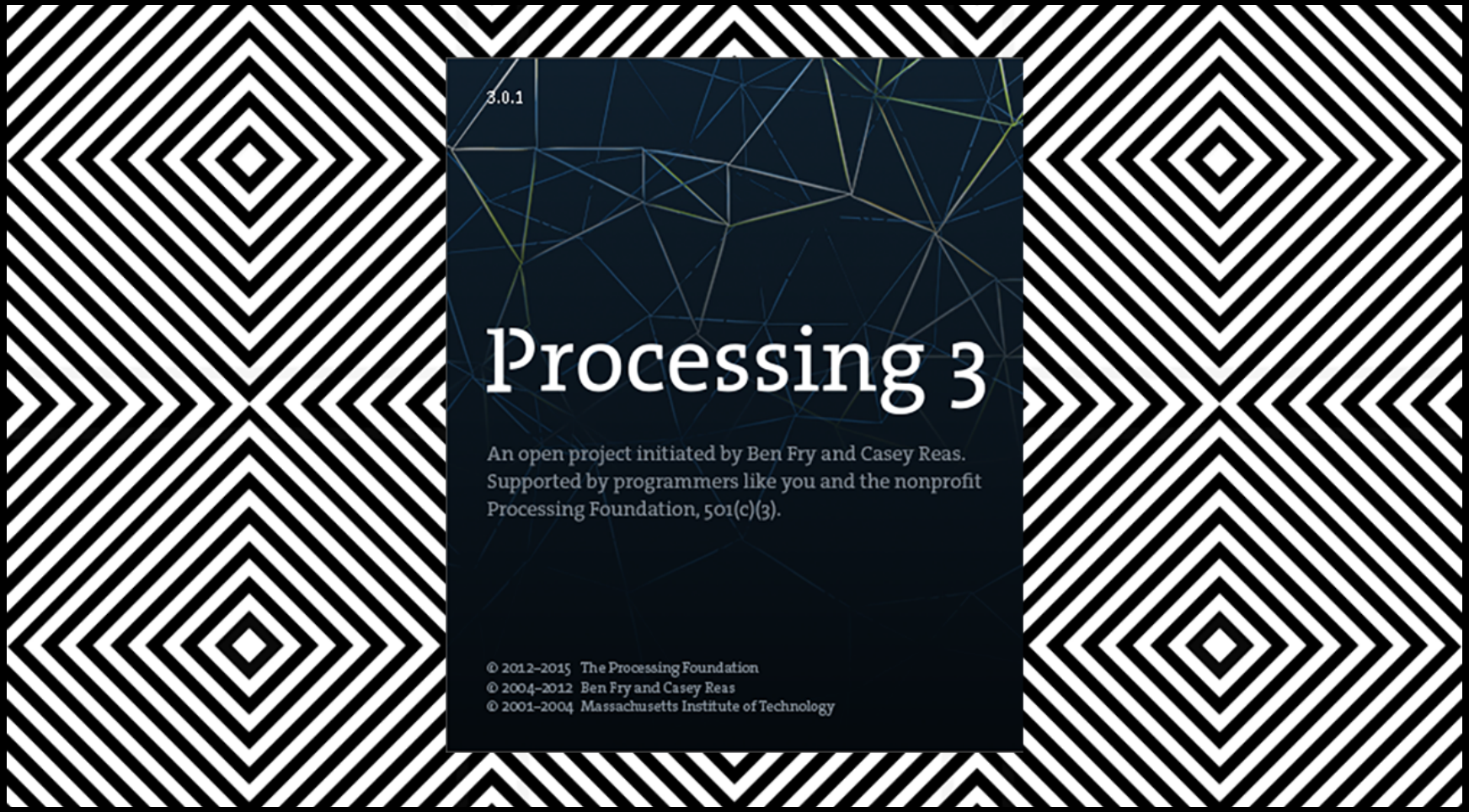
The event loop - frameCount

```
void draw() {
   // draw a centered rectangle
   rectMode(CENTER);
   if(frameCount <=1) {
      rect(width/2, height/2, 200, 200);
   }
   // this does not stop the loop – but it does only
   // draw the square once
}
```

The event loop - frameCount

```
void setup() {
  // set the size of the window
  size(1280, 720);
  // set the background to white
  background(255);
  // set the fill color to transparent red
  fill(255, 0, 0, 127);
  // turn off the stroke – outline
  noStroke();
  // stop event loop
}

void draw() {
  // draw a centered rectangle
  rectMode(CENTER);
  if(frameCount <=1) {
    rect(width/2, height/2, 200, 200);
  }
}
```


3.0.1
Processing 3
An open project initiated by Ben Fry and Casey Reas.
Supported by programmers like you and the nonprofit
Processing Foundation, 501(c)(3).
© 2012–2015 The Processing Foundation
© 2004–2012 Ben Fry and Casey Reas
© 2001–2004 Massachusetts Institute of Technology

Adding the webcam

In the Processing menu choose:

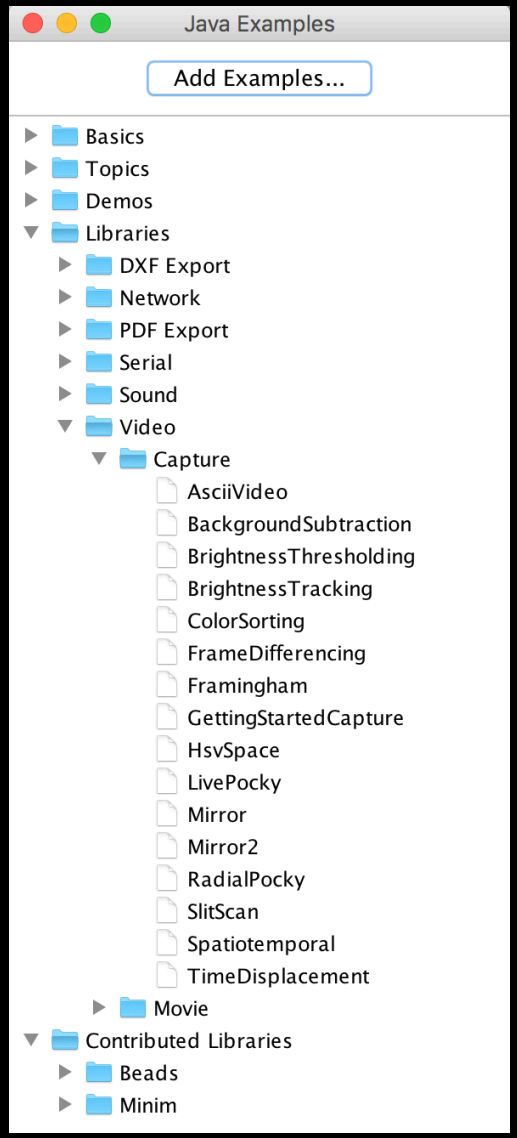
- Sketch/Import Library…
  - Video

  Adds the import statement to the top of your file

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

Looking at Examples

- File/Examples



Slit Scan

- https://headcrash.industries/research/slit-scan-renders-of-videos-in-javascript/

Slitscan code - commented

```
import processing.video.*;

Capture video;  // declare a global video camera named video

int videoSliceX; // declare a global integer called videoSliceX
int drawPositionX; // declare integer called drawPositionX

void setup() {
  size(600, 240); // create the display window

  // This the default video input, see the GettingStartedCapture
  // example if it creates an error
  video = new Capture(this,320, 240); // allocate memory for video

  // Start capturing the images from the camera
  video.start();

  videoSliceX = video.width / 2;  set the global videoSlixeX to ½ the video width – which was set to 320
  drawPositionX = width - 1;
  background(0);
}
```

Slitscan code - commented

```
void draw() {
  if (video.available()) { // check to see if the video camera is available and if it is begin reading
    video.read();
    video.loadPixels(); // load the video frame  pixel data into the vidoe.pixels[] array

    // Copy a column of pixels from the middle of the video
    // To a location moving slowly across the canvas.
    loadPixels(); // load the screen pixel data for the  pixels[] array
    for (int y = 0; y < video.height; y++){ // declare an integer y and initialize it to “0”;  while y is less than the height of the video do the
                                            // following, for each iteration of this loop increase y by “1” until it reaches the height
      int setPixelIndex = y*width + drawPositionX; // set the index for the screen pixels array
      int getPixelIndex = y*video.width  + videoSliceX; // get the index for the video frame pixels array
      pixels[setPixelIndex] = video.pixels[getPixelIndex]; // set the pixels array to the slice of video
    }
    updatePixels(); // update the display window

    drawPositionX--; // decrement the drawPositionX by “1”
    // Wrap the position back to the beginning if necessary.
    if (drawPositionX < 0) {
      drawPositionX = width - 1;
    }
  }
}
```

Let’s experiment!

Some useful links

- http://alternativeto.net/software/open-broadcaster-software
- https://kdenlive.org/downloading-and-installing-kdenlive
- http://windows.microsoft.com/en-us/windows/movie-maker#t1=overview

Thanks for coming to …

Introduction to webcam graphics @ MakeICT