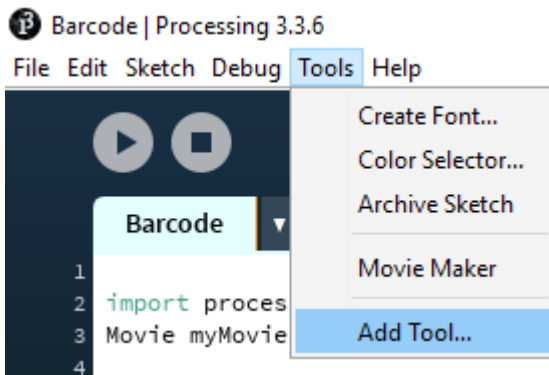


Tutorial For creating a Movie Barcodes via Processing 3.3.6

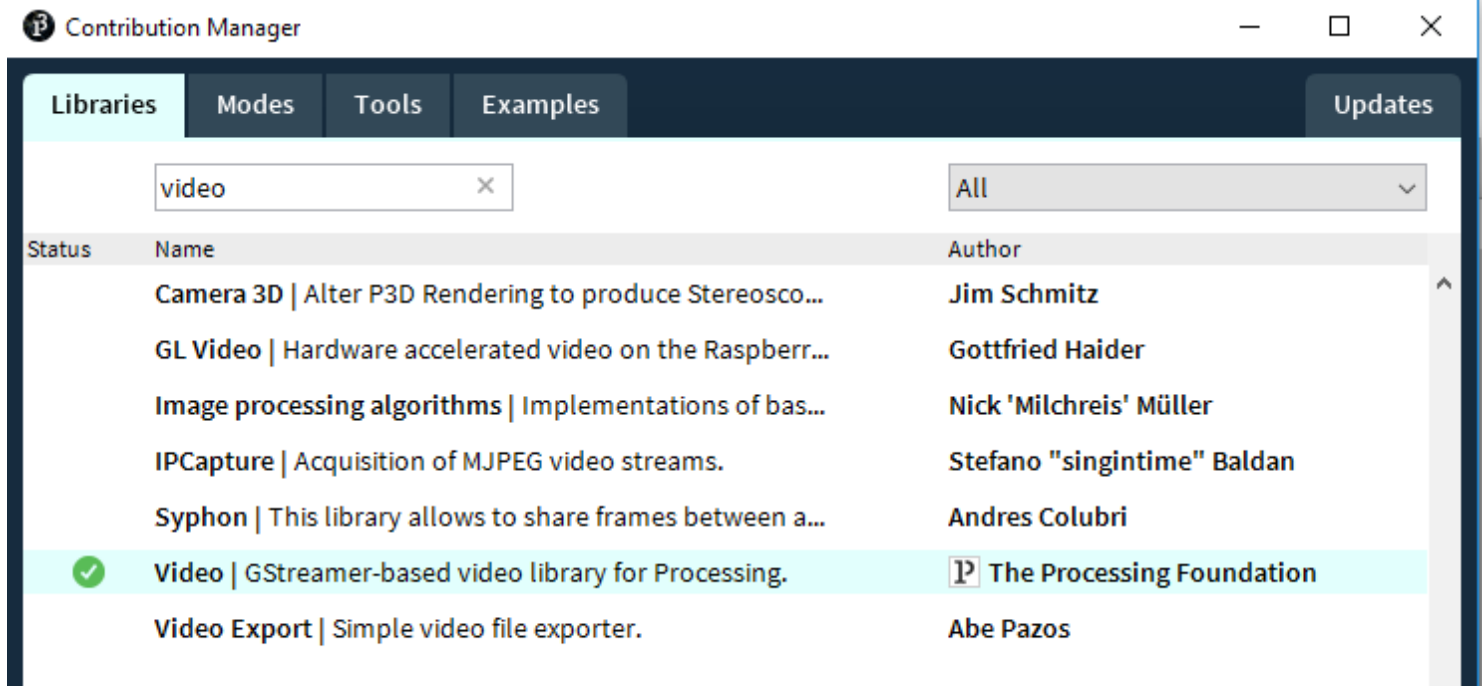
Since I noticed very few projects done by processing I decided to make this tutorial. The tutorial will mainly go over my code for generating movie barcodes and explain what each line does. However, before plunging into the exciting journey let us configure few things.

Setting up the library.

Make sure you have the video library. In other words, if processing doesn't recognize the movie class you probably don't. Then you need to go to the tools> Add Tools> Libraries>...

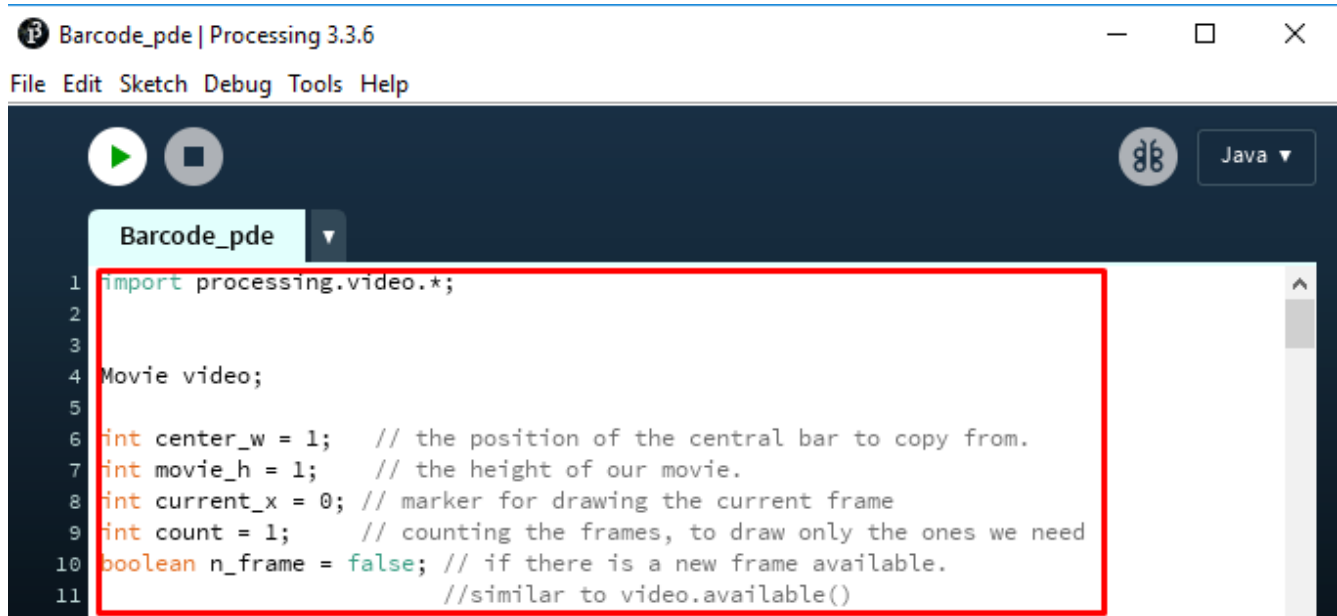


Search for the video and install the video library. Make sure it is the official one from the processing. After doing this you are set to go.



So let's begin the tutorial.

Import & Global vars.



The screenshot shows the Processing IDE window titled 'Barcode_pde | Processing 3.3.6'. The menu bar includes 'File', 'Edit', 'Sketch', 'Debug', 'Tools', and 'Help'. The toolbar has buttons for running (a green play button) and stopping (a square button). A dropdown menu shows 'Java'. The code editor displays the following code, with lines 1 through 11 highlighted by a red rectangle:

```
1 import processing.video.*;
2
3
4 Movie video;
5
6 int center_w = 1; // the position of the central bar to copy from.
7 int movie_h = 1; // the height of our movie.
8 int current_x = 0; // marker for drawing the current frame
9 int count = 1; // counting the frames, to draw only the ones we need
10 boolean n_frame = false; // if there is a new frame available.
11 //similar to video.available()
```

In the first line, we are just importing the library we will use for working with the video. If processing can't find the library probably you skipped the previous step and missed installing the library. I learn through my research that the library was in the list of main libraries in the processing; however, since it was heavy, now it should be separately installed.

The next line we are just initializing our video, which we will be using for all operations related to our file. You can find the reference to the operation through this link (I will still explain the ones I used): <https://processing.org/reference/libraries/video/Movie.html>

We create an int for our x position. Our height is going to be the same for all bars, however since we need to stick bars one after each other we will keep a variable which will start at 0 and end on the number of our bars. Also we will have two variables for determining the position of the bar we want to copy. We also want to keep track of on what frame we are so we don't copy every frame to not have an enormous image.

Intro To Processing.

Before proceeding further on let quickly understand the core function with which we will have to deal. The setup() will be called only once. It is used for initialization of components. The draw will be called every frame, it will be responsible for redrawing everything we see in the processing window. Even if we don't do animation, it will be called every frame. Finally, we have movieEvent since we are dealing with a movie. (There is also captureEvent used for direct capturing from the camera. Processing differentiates videos to movies, the one with which we will deal, and to camera captures.)

Now we can proceed.

The Setup function.

A screenshot of the Processing IDE window titled "Barcode_pde | Processing 3.3.6". The menu bar includes "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The toolbar shows a play button, a stop button, and a language dropdown set to "Java". The code editor displays the following code:

```
19
20
21 */
22 void setup() {
23   // size(1600,320);
24   size(7000, 320);
25   video = new Movie(this, "origins.avi");
26   video.play();
27   // video.frameRate(1);
28   video.frameRate(24);
29   video.speed(1920);
30   center_w = video.height/2;
31   movie_h = video.height;
32 }
```

First, we initialize our window. It's width and height. The command for it is `size(width, height)`. The best number for the width is the number of bars we want to have. However, since we always can crop the final image I put a larger number than needed. Similarly, we want to put the height of our video file. Here as well having a little bigger number won't hurt.

Next, we want to initiate our movie. We need to give a reference to our program for calling it ("this") and the name of the movie. ATTENTION: Movie must be located in the "data" folder, in the same place where our sketch(program) is located.

Then we want to make our program know that it has to play the movie. (`video.play()`). We can use `video.loop()` if we want it to start the movie again when it is done.

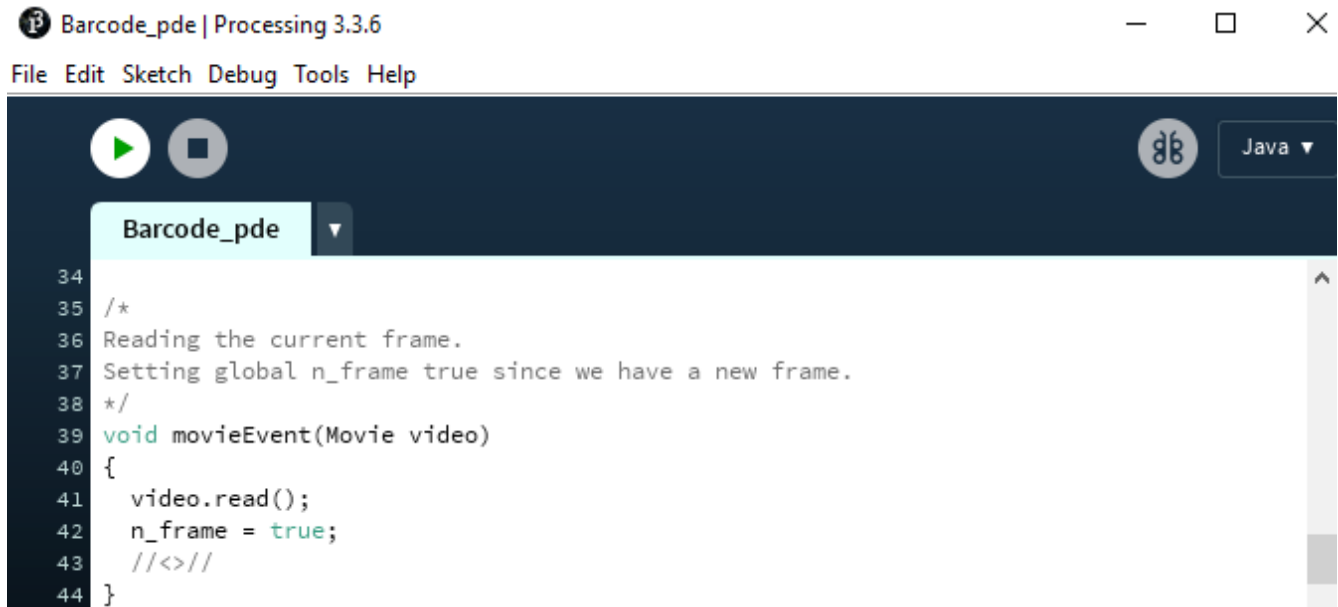
Then I used two extra commands.

One specified how many frames per second the movie will have. And the next is the speed.

Since for generating a barcode processing has to run the video, having a larger speed makes a program a little faster. Initially, I was experimenting with `video.jump(seconds)`; nevertheless, the results showed that it makes the program even slower.

MovieEvent.

Here we just tell the processing that it needs to read the movie file. We also set our global var `n_frame` to true so our draw knows it has a new frame. It could be done also via `video.available()` however the last one worked a little slower for me.

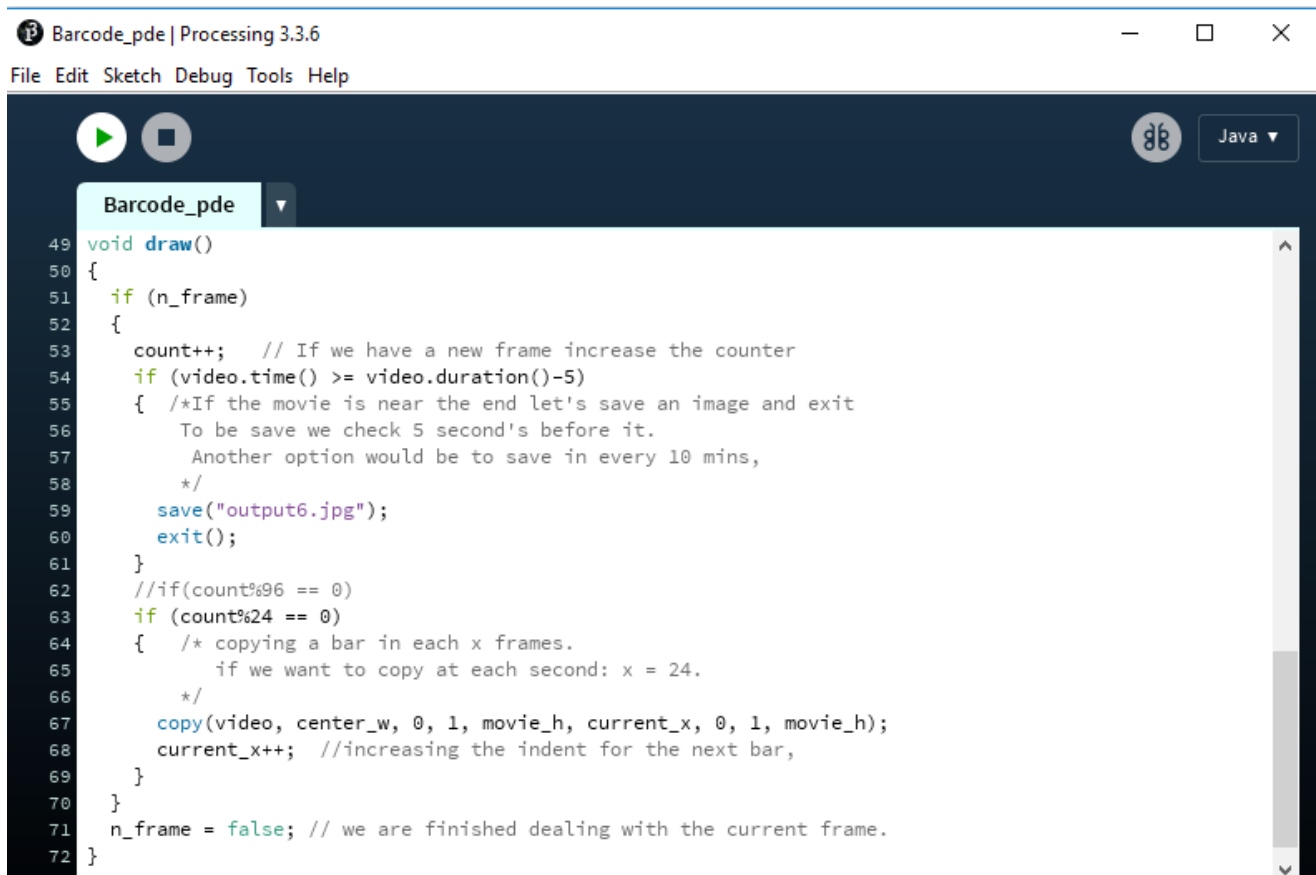


The screenshot shows the Processing IDE interface. At the top, the title bar reads "Barcode_pde | Processing 3.3.6". Below it is a menu bar with "File", "Edit", "Sketch", "Debug", "Tools", and "Help". The main workspace has a dark blue header bar with a play button, a stop button, a magnifying glass icon, and a language dropdown set to "Java". Below the header, a tab labeled "Barcode_pde" is active. The code editor shows the following code:

```
34
35 /*
36 Reading the current frame.
37 Setting global n_frame true since we have a new frame.
38 */
39 void movieEvent(Movie video)
40 {
41     video.read();
42     n_frame = true;
43     //<>
44 }
```

Now we reach to our final function.

Draw function.

A screenshot of the Processing IDE window titled 'Barcode_pde | Processing 3.3.6'. The menu bar includes 'File', 'Edit', 'Sketch', 'Debug', 'Tools', and 'Help'. The toolbar shows a play button, a stop button, and a language dropdown set to 'Java'. The code editor displays the following code:

```
49 void draw()
50 {
51   if (n_frame)
52   {
53     count++; // If we have a new frame increase the counter
54     if (video.time() >= video.duration()-5)
55     { /*If the movie is near the end let's save an image and exit
56        To be save we check 5 second's before it.
57        Another option would be to save in every 10 mins,
58        */
59       save("output6.jpg");
60       exit();
61     }
62     //if(count%96 == 0)
63     if (count%24 == 0)
64     { /* copying a bar in each x frames.
65        if we want to copy at each second: x = 24.
66        */
67       copy(video, center_w, 0, 1, movie_h, current_x, 0, 1, movie_h);
68       current_x++; //increasing the indent for the next bar,
69     }
70   }
71   n_frame = false; // we are finished dealing with the current frame.
72 }
```

First, we check if we have a new frame. If we do then we increase our counter responsible for keeping track of frames. .

We proceed by checking how far in the movie we are. If we are at the end (5 seconds away from the end) we save the image and exit.

Otherwise, we look at the counter. Initially, I was drawing a bar in every 4 seconds, that is in every 96 frames. However, you can tweak that number to see what fits better. So when we reach the desired frame we go on. (24 doesn't seem to work well).

Copy function copies a rectangle from a movie to our window. It takes nine parameters. First, the origin from which it has to copy. In our case, it is the video. Then it asks for the rectangle we want to copy, that is for the x,y, width, and height. Our x is fixed in the center of the movie. Basically, that is the most important number here, since it is responsible for the appearance of bars our final image will contain. You can experiment with it. For example, setting it to 1, works pretty good as well. Next, we set y to 0, since we want to copy the bar from the beginning. We set the width to 1 and the height to the height of the movie.

After this, we provide the rectangle in our final image when the bar should appear. Here all arguments are the same but the x-axis. For it, we provide our current_x value. That's it for the copy. Next, we increment our current_x so the next bar appears next to the current one.

We are done here, we can freely set n_frame to false and wait for the new frame.

That's it for the tutorial.

I hope it helped,

Aram Serobyanyan.