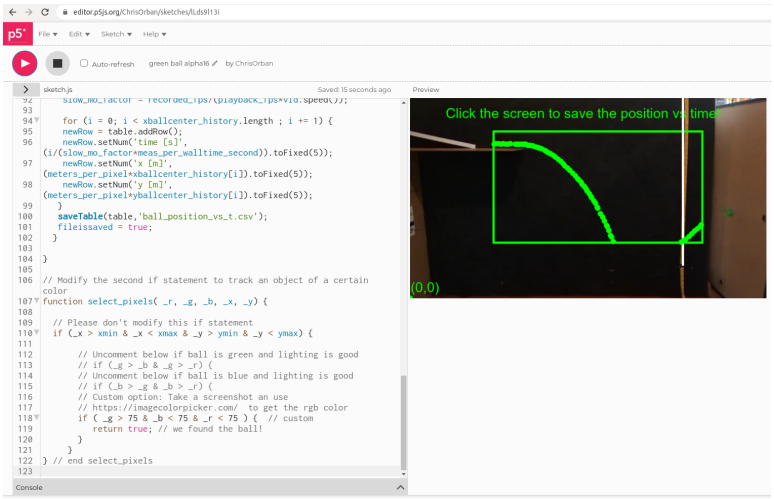
Track an object with specific colors!

By Chris Orban and Jennifer Boughton
(STEMcoding Project)



Recording Compatibility: Android and iPhone / iPad (in compatibility mode)
Analysis Compatibility: Chromebook / Windows / Mac / iPad / Linux
Browser: Tested with Chrome, Firefox, and Safari
(recommend updating browser if there are any issues or errors)

Your smartphones and tablets can record at 240 frames per second. This is faster than your eyes can see and it is about 10x more frames per second that most TV shows! Wow! We are going to use this technology to do physics experiments at home. This example will highlight a ball rolling off a table but you can (and should!) do whatever you want with this.

There are great programs out there like Pivot Interactives, Tracker Video and http://jst.lucademian.com which help you go frame by frame and click on an object to get its position and see how it moves over time. Then you can analyze that data to get velocities over time. This often involves a lot of clicking which can be effective but it can take a long time, which is no fun.

If your object is bright green, or blue or red then you can track objects using the same method they use in the movies to add computer generated effects to a movie set – specifically they put the actors in front of a giant blue or giant green screen. Here we are going to use a bright green or blue or red object in front of a black background (like a black posterboard). Instead of removing the green or red or blue objects and replacing them with computer graphics like they do in the movies, we are going to use the bright colors to track the positions of these objects as they move. The program is configured to find all the pixels of a certain color and record the middle of all those pixels.

Here is a demo of a code that does this with a green ball (works best with chrome or safari):
https://editor.p5js.org/ChrisOrban/sketches/lLds9l13i

Press play there to run the code. If you want to see what it looks like without the white and green dots tracking the object position you can just comment out or delete where it says displaySelectedPixels(); Just make sure you can put it back where it was later.

Click the screen when you are ready to download the data file (ball_position_vs_t.csv).Once you download the data you can analyze it in a google sheet like this one: https://docs.google.com/spreadsheets/d/160zDCb4V-sWaOXE0hr71B40U71G2di528cnUKXZgZvQ/edit?usp=sharing

To set up your own experiment, find a brightly colored object (preferably a green or blue ball) and put it in front of a dark background (preferably black). Make sure there is some kind of ruler or meter stick or yard stick in the frame. You may also find it helpful to put a stop watch in the frame.

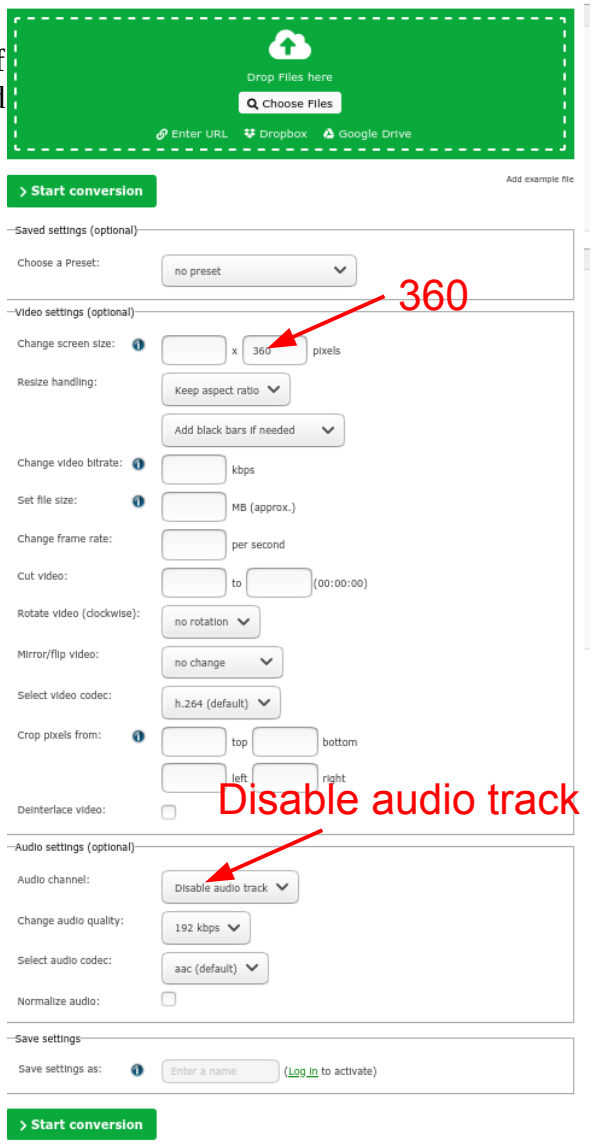
If you are using an apple device, go into photo settings and select “compatibility mode” and select 720p resolution with 240 fps.

On an android device just use the highest fps that you can do even if it is not the highest resolution. Frame rate matters more than resolution for this experiment.

Record the video of your experiment making sure your device is as stable as it can be. Drop an object straight down or slide it off a table or bounce it off a wall or do whatever you want. On iPhone and iPads, press record and wait a few seconds before starting your experiment. On apple devices, there is typically a few seconds of normal recording speed at the beginning of each “slow motion” video capture. We want to try to avoid those few seconds.

Upload your recording to this site: https://video.online-convert.com/convert-to-mp4

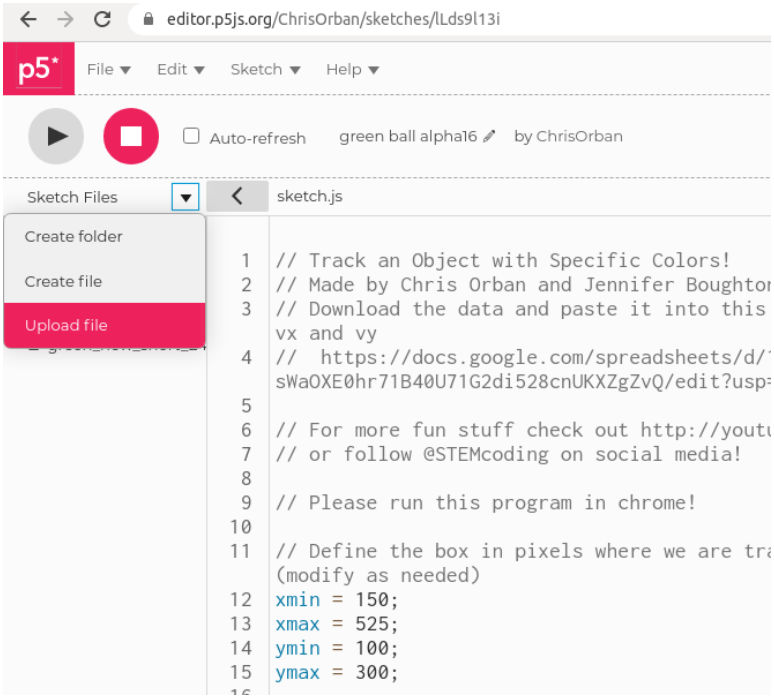
Before you press convert you need to change two settings! Change the vertical pixels to 360 and set the audio channel to “Disable audio track”. This is the best way to get the size of the mp4 file to be less than 5 MB.



Click “Start conversion” After it uploads the file it should only take 1-2 minutes for it to give you a much smaller file that you can download.

Open up this link in chrome or safari browser (same link as before)
https://editor.p5js.org/ChrisOrban/sketches/lLds9l13i
Note: to replace the video with a new file you will need to click “Sign Up” on the top left and create a free account (no spam, don’t worry). Then click File → Duplicate in order to modify this program and change out the video.

Click < on the top left to open up a new menu on the left then click the down arrow to upload the file like this:



You can feel free to erase the old file (green_new_short_24fps.mp4) if you want.

Find the line that looks like this:

vid = createVideo(“green_new_short_24fps.mp4”);

And change the file name to the name of your file.

If you recorded your video at some other rate than 240 fps you will need to change the value of recorded_fps to whatever it was. Some smartphones may only go up to 60 fps which is totally fine or maybe you recorded at 24 or 30 fps which is totally fine too.

Next we need to figure out the playback rate. This is usually the same or less than the recorded_fps. To figure it out your file in http://jst.lucademian.com and notice what it says for frame rate. Put that number as playback_fps

If you recorded with 16:9 aspect ratio you will not need to change horizontalpixels so you can leave it at 640. You only need to change this number if the video looks stretched. If you recorded at 4:3 aspect ratio, then change vertical pixels from 640 to 480. If you recorded 4:3 aspect ratio but vertical, then

change horizontalpixels to 270. If you recorded 16:9 aspect ratio but vertical, then change horizontalpixels to 203. If you are totally unsure what horizontalpixels is all about just try 640, 480, 270 and 203 and one of them will probably look right (not stretched). Go with that one.

Now change this part of the code until the white line overlaps with your ruler or meter stick or yard stick or whatever you used:

```
// Modify as needed
xruler1 = 490;
xruler2 = 490;
yruler1 = 60;
yruler2 = 360;
physical_length_meters = 1.0; // metric length of ruler or yardstick or meterstick
```

Then change physical_length_meters to whatever the metric length of your ruler is. If you are using a 12 inch ruler then set physical_length_meters = 0.3048; for example.

The program will break down each frame into colors and each pixel will have a color scale with red going from 0 (not red at all) to 255 (as red as it gets), and the same for green and blue. Pure black is red = 0, blue = 0, green = 0 and pure white is red = 255, green = 255, blue = 255. So if it is a little dark where you are recording you may find, for example, that green is larger than red or blue but green might not be much more than 50, which is the case with the example video.

The program is set up to follow green objects because select_pixels() has an if statement that is only true for green objects. There is other code there that you can use if the object you have is blue or red.

```
// Modify the second if statement to track an object of a certain color
function select_pixels( _r, _g, _b, _x, _y) {

  // Please don't modify this if statement
  if (_x > xmin & _x < xmax & _y > ymin & _y < ymax) {

      // Uncomment below if ball is green and lighting is good
      // if (_g > _b & _g > _r) {
      // Uncomment below if ball is blue and lighting is good
      // if (_b > _g & _b > _r) {
      // Custom option: Take a screenshot an use
      // https://imagecolorpicker.com/  to get the rgb color
      if ( _g > 75 & _b < 75 & _r < 75 ) {  // custom
         return true; // we found the ball!
      }
    }
} // end select_pixels
```

You can even do a custom color (like purple) by taking a screenshot of your video and loading the image into http://imagecolorpicker.com and getting the rgb color of the object you are interested to track. The code above is looking for objects “greener” than 75 and less blue than 75 and less red than 75. In javascript the “and” operator is &

You may notice that there is also an if statement there:

```
// Please don't modify this if statement
if (_x > xmin & _x < xmax & _y > ymin & _y < ymax) {
```

This is so that we only look for the object in a particular part of the screen. This is a helpful thing to do if you are tracking a green object, for example, and have something greenish in the corner of the screen that you don't really want to track. Notice how the ball in the example is only tracked when it enters the green rectangle and as it falls below the green rectangle we stop tracking it.

You can modify this rectangular area by changing the code at the beginning of the program:

```
// Define the box in pixels where we are tracking the ball position (modify as needed)
xmin = 150;
xmax = 525;
ymin = 100;
ymax = 300;
```

As you change these values the green box on the screen will change too. On the screen (0,0) is marked in the bottom left so the min and max values are relative to those numbers. Move the green box to encompass the part of the video that we want to analyze the motion. Make sure not to use ymax > 360 or xmax > 640 because that would definitely be off the screen.

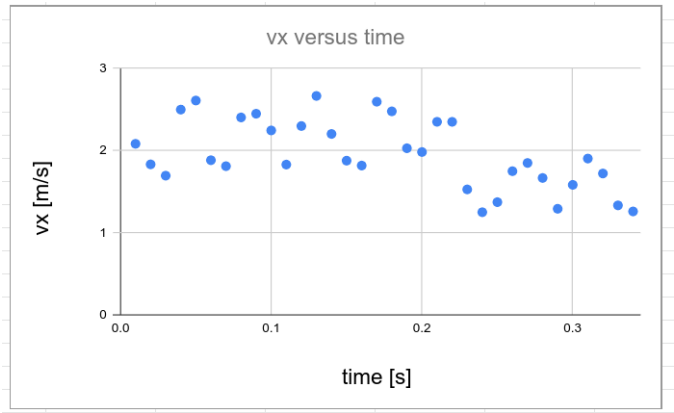
Last thing:

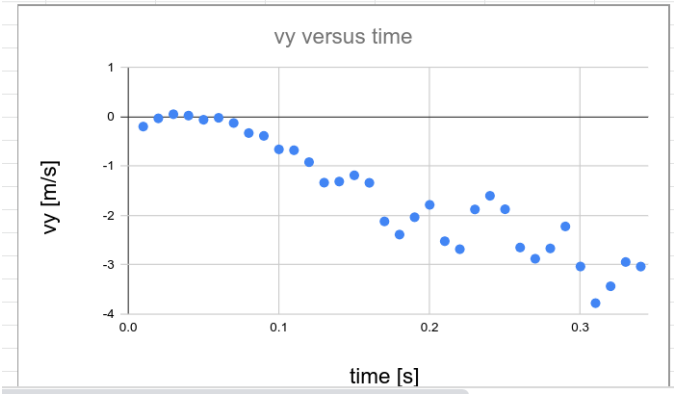
Once you have everything in place how you want it, you will need to uncomment `frameRate(1);` in `setup()` and set `vid.speed(0.1);` in `preload();` This will make the program run much more slowly but you will get more data points and the program will do a better job of making sure that there are equal intervals of time between measurements.

Now run the code and when the experiment in the video is about finished you can click the screen to save your data to the file `ball_position_vs_t.csv` As you wait for the program to finish you can think about all the time you are saving by not needing to go frame by frame clicking on the position of the ball.

Open up `ball_position_vs_t.csv` in Google sheets and copy the data from the first three columns to the clipboard by highlighting and pressing Control + C. Then open up the google sheet link from earlier: https://docs.google.com/spreadsheets/d/160zDCb4V-sWaOXE0hr71B40U71G2di528cnUKXZgZvQ/edit?usp=sharing
and erase all the data in the first three columns. Then press Control + V (or on an iPad cmd + C) to paste your data into those columns. If you are in calculus-based physics, take a moment to appreciate that the spreadsheet is set up to calculate the velocities using a central difference approximation

Below is some data from our example video (green_new_short_24fps.mp4) that shows that the horizontal velocity of the ball is close to constant but the vertical velocity is increasing at a constant rate. You can analyze it to estimate gravitational acceleration if you want to!





Summary:

It takes some extra work, but if you use a brightly colored object and a dark background and take the time to figure out the approximate rgb values of your brightly colored object, then you can track the object's motion without all the tedious (but important!) work of advancing the frame and clicking on the object's new location like you would with other programs (for example http://jst.lucademian.com). Even though this is a nice time saver, the clicking and advancing is still important because it is always helpful to have one or more independent ways of measuring what you want to measure, like the earth's gravity or the final speed, etc. This is science after all, so it's good to be scientific about it!

If you liked this code and are interested to play around with more fun stuff check out the STEMcoding Youtube Channel http://youtube.com/c/STEMcoding

Send Chris Orban an e-mail at orban@physics.osu.edu if you have suggestions to improve this document.