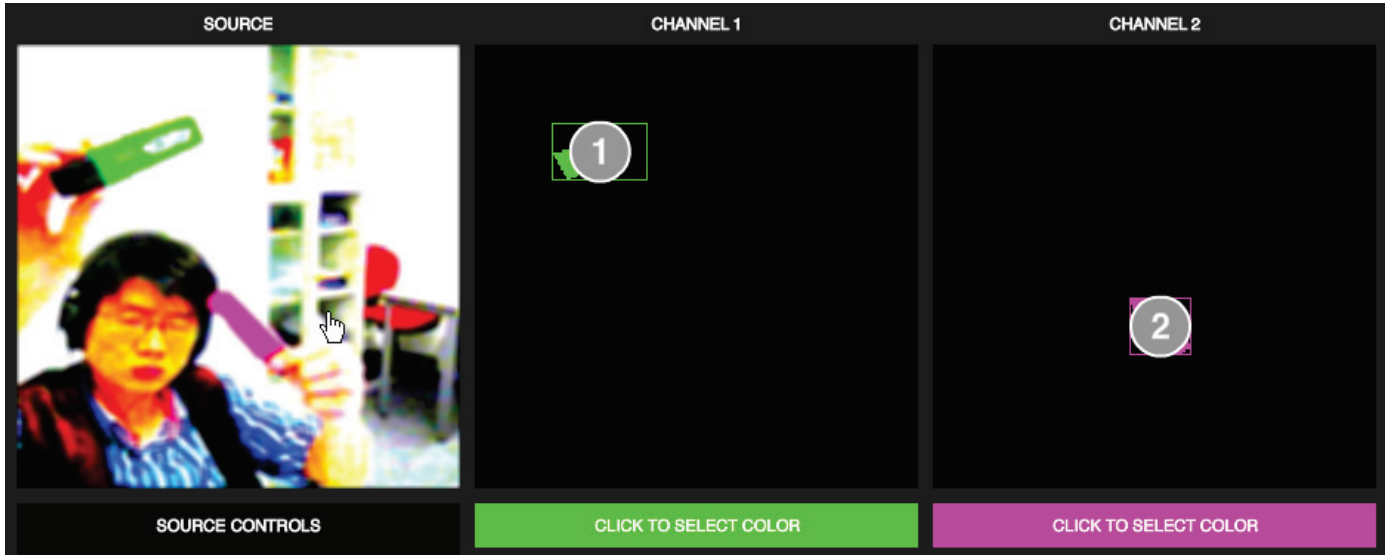
Topic 06:
Color Tracking

Hongbo Fu

Key Idea

- **Step 1**: **Set** some color as **tracking color**
- **Step 2**: **Drop all pixels that don't fall within color range** defined by the tracking color
  - **BitmapData.threshold** method
- **Step 3**: **Pinpoint the remaining pixels as tracked region**
  - **BitmapData.getColorBoundsRect** method



Online Demo

Color Tracking Library

- In **cityu.scm** package
  - **Webcam class**
    - An easy-to-use wrapper for webcam access
  - **ColorTracker class**
    - Main class for color tracking
    - Subclass of **Tracker** class
  - **ColorTrackerSettingUI class**
    - User interface for parameter settings

- **Library homepage**

Webcam Class

- A subclass of **Sprite**
- **Constructor**
  - `Webcam(w:Number=640, h:Number=480)`
    - **Scale up** original 320x240 video to satisfy the input size
- **Property**
  - `cam: Video; // webcam video`
- **Method**
  - **sendToBack**()
    - Avoid webcam occluding other display objects on stage

Webcam Class

- **Event**
  - **Webcam.READY**

```
import cityu.scm.Webcam; // import Webcam class

var webcam:Webcam = new Webcam(400, 300);

webcam.addEventListener(Webcam.READY, onCameraReady);

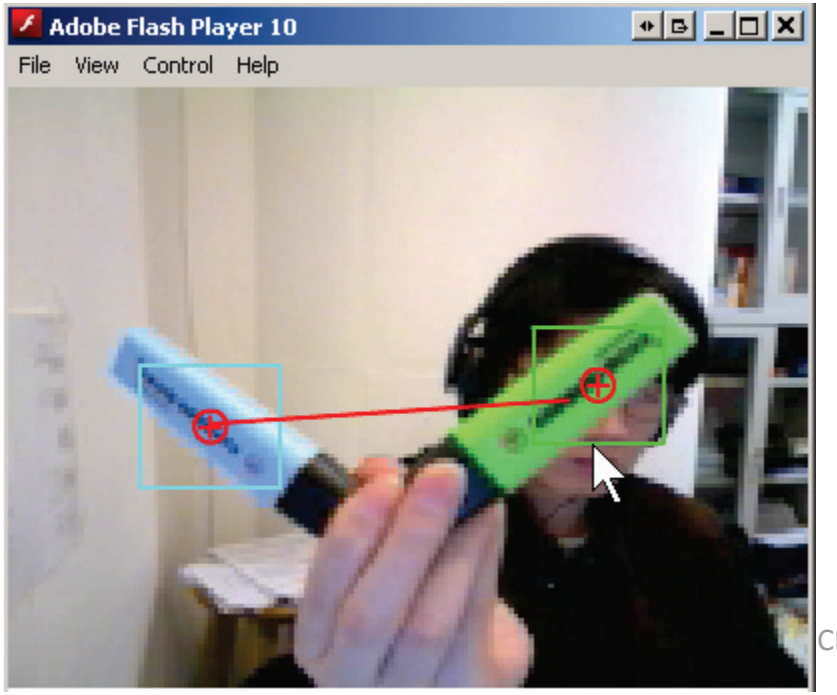
function onCameraReady(e:Event): void {
        addChild(webcam);
}
```

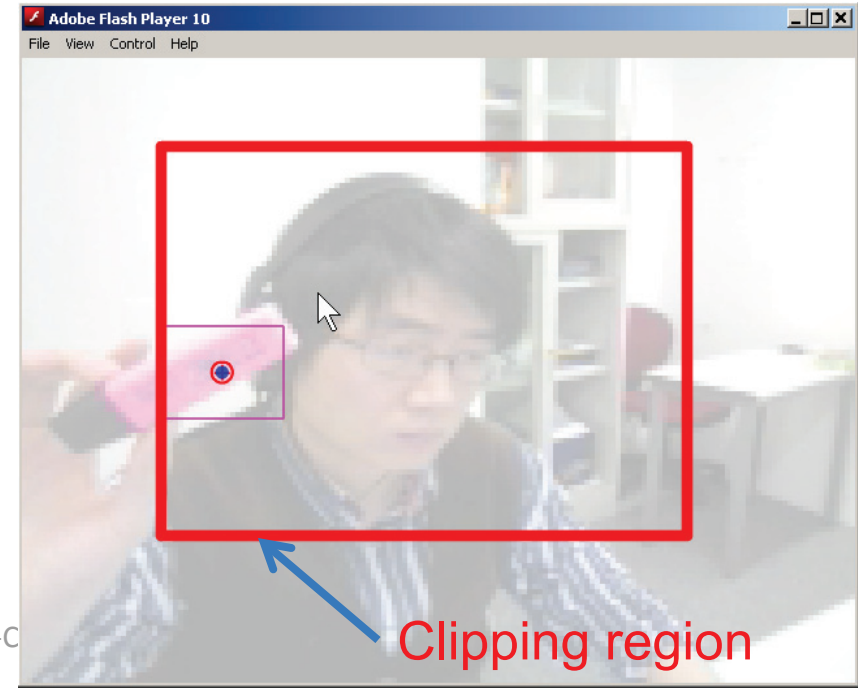
Webcam Class

- **Flipping as mirror effect** is done **internally**

ColorTracker Class

- **Default visualization** for tracked regions
- Able to **track multiple color objects**
- Each color object can have a **clipping region**
  - Check if the object is tracked **only within the clipping region**





ColorTracker Class

- Constructor
  - ```
    ColorTracker(theCam:Webcam,
    theL:Number=0, theT:Number=0, theR:Number=0,
    theB:Number=0, response:Number=2)
    ```
    - **theL, theT, theR, theB**
      - Left, top, right, bottom **positions of clipping region**
      - **With respect to top-left corner of theCam**
    - **response**
      - In **[1, 10]**; 1 is fast but jumpy, 10 is smooth but slow
    ```

ColorTracker Class

- Methods
  - `setColorFrom(x:Number, y:Number):void`
    - **Specify** the color at (x, y) of current frame as **tracking color**
  - `isTracked(): Boolean`
    - Tracked or not?
  - `hideTrackedArea, hideTrackedCenter`
    - Hide default visualization of tracked region
    - Call them if you want to have **custom visualization**

Simple Usage of ColorTracker

- **Step 1**: import ColorTracker class
  - I.e., `import cityu.scm.ColorTracker;`
- **Step 2**: initialize a ColorTracker object
  - E.g.,

```
var tracker:ColorTracker = new ColorTracker(webcam);
```

- **Step 3**: set tracking color
  - E.g., in some event listener for mouse click

```
tracker.setColorFrom(e.localX, e.localY);
```

Clipping Region

- If you specify a **clipping region**, then tracking is performed **only within this region**
  - Clipping region is specified when creating a ColorTracker object

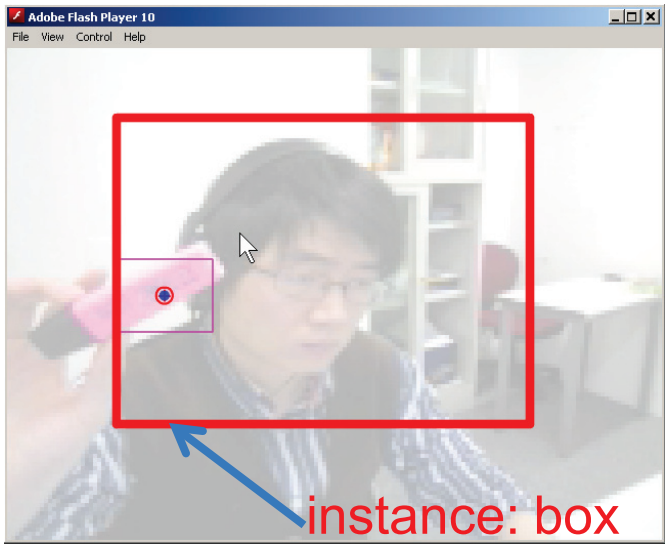
- Example

```
tracker = new ColorTracker(webcam,
                           100, 100, 400, 400);
```

Note: clipping region's position is with respect to top-left corner of webcam

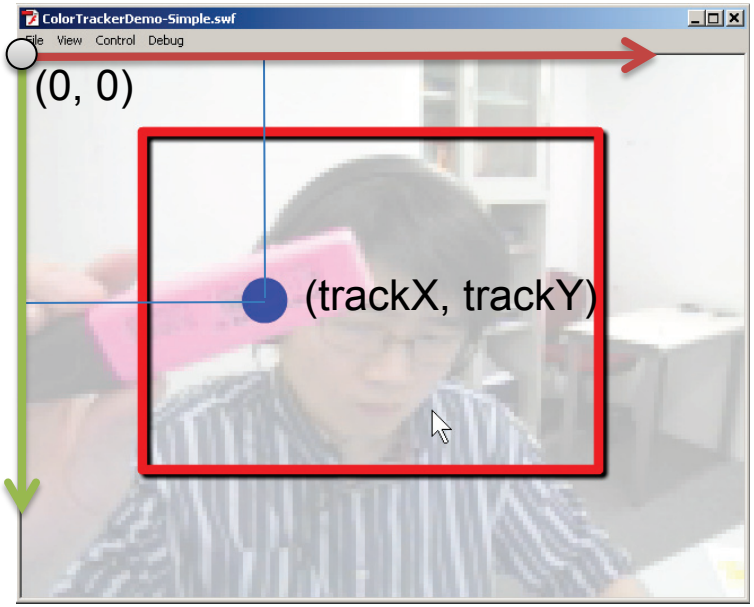
Class Exercise

- **Task 1**
  - Play with the **response** parameter of **ColorTracker constructor** (i.e., its last parameter)
    - In [1, 10]: **1 means fast; 10 means slow but smooth**
- **Task 2**
  - Add a **rectangular symbol instance (name: box)** onto stage
  - **Use this instance to define a clipping region**
    - To avoid occlusion of this instance by webcam, send camera to back by calling **webcam.sendToBack();**



Custom Visualization

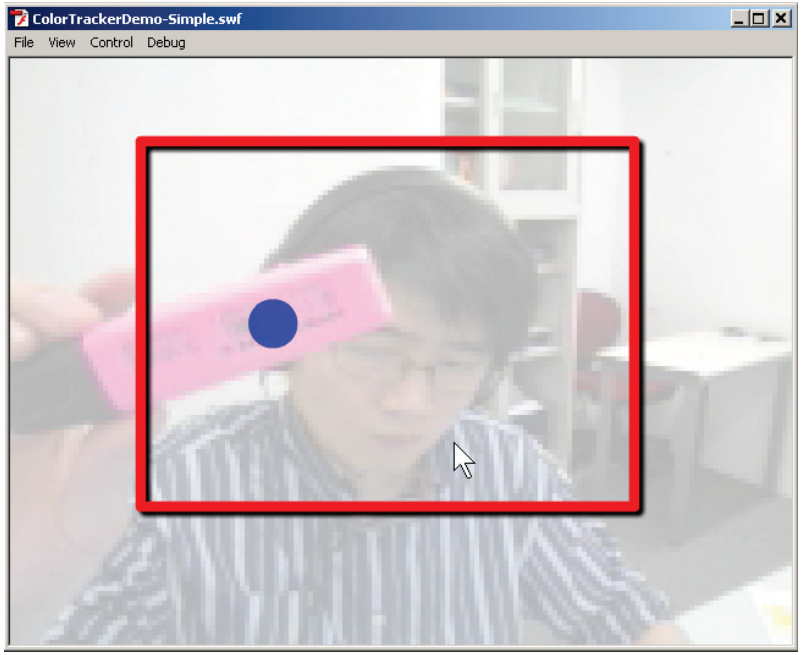
- Useful **properties** of ColorTracker class
  - `trackX: Number; trackY: Number`
    - Center of tracked region
    - **With respect to top-left corner of the webcam**
  - `trackRect: Rectangle`
    - Rectangle enclosing the tracked region
- Hide default visualization by
  - `hideTrackedArea`
  - `hideTrackedCenter`



Class Exercise

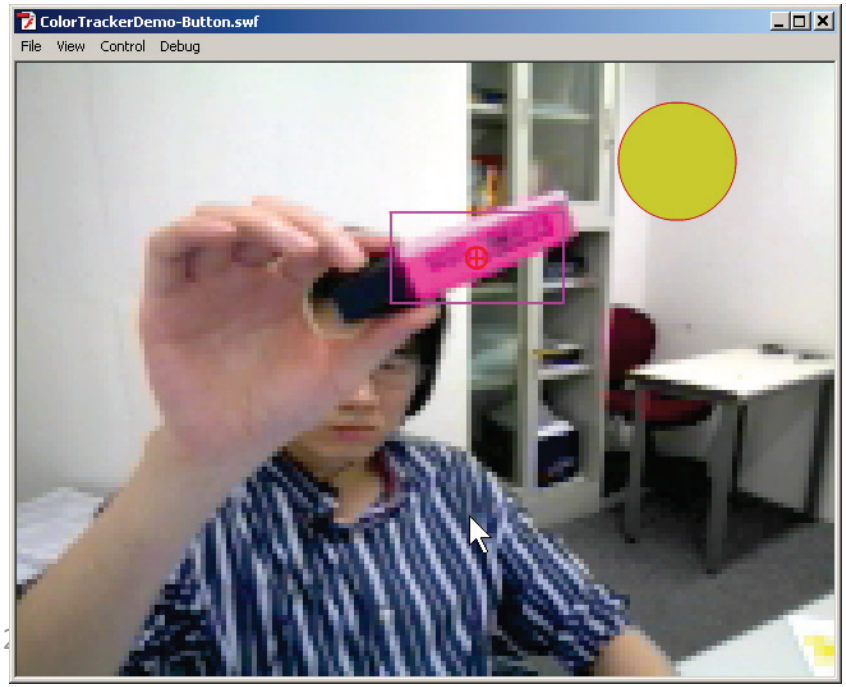
- **Task**: custom visualization
  - **Place a circle in blue at the tracked position**
    - (trackX, trackY)
  - **Hide the circle if no tracked object is found**

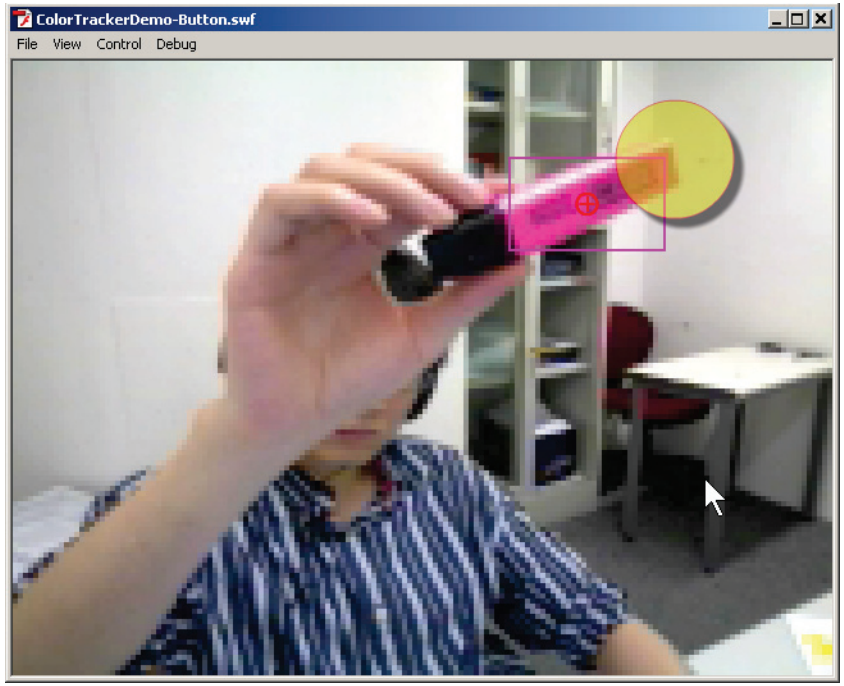
- **Hints**
  - Need **event listener** for ENTER_FRAME
  - Use **isTracked method** to check if tracked object is found



Class Exercise

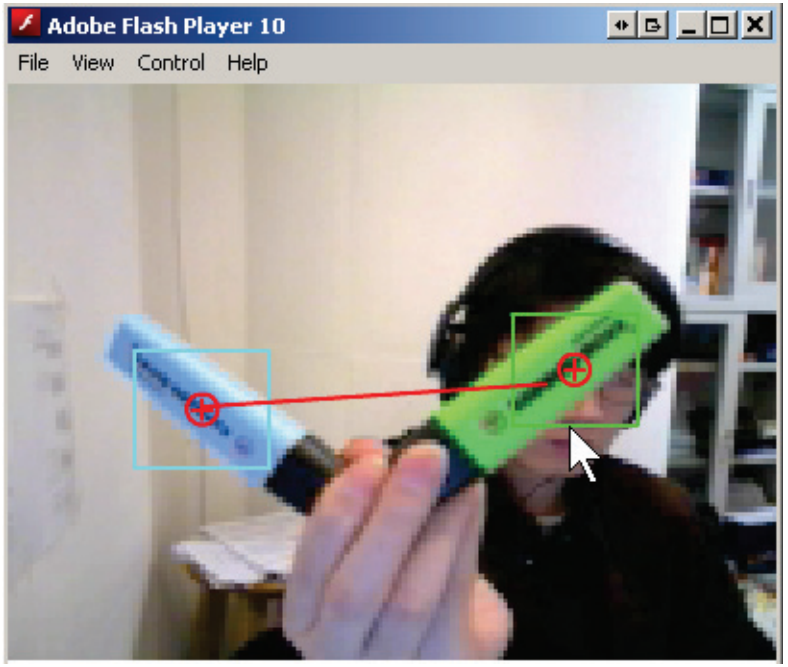
- **Button clicking**: make symbol instance have **shadow effect & semi-transparent when color object is over it**
- **Hints**: use **trackRect** property, **DropShadowFilter**, and **Rectangle**'s **intersects** method (ref)

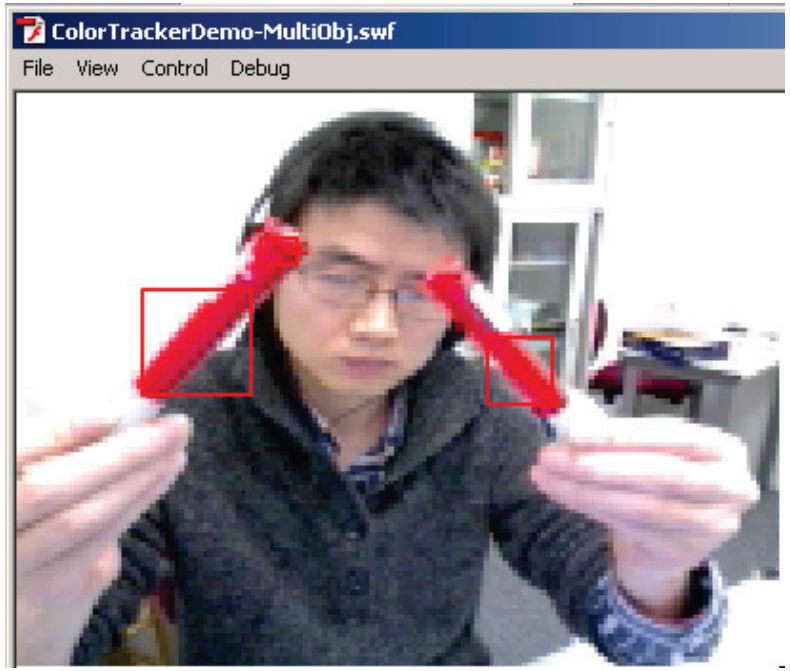




Tracking Multiple Objects

- Case 1: multiple object in **different colors**
- Case 2: multiple objects in **same color**





Case 1: Multiple Objects in Different Colors

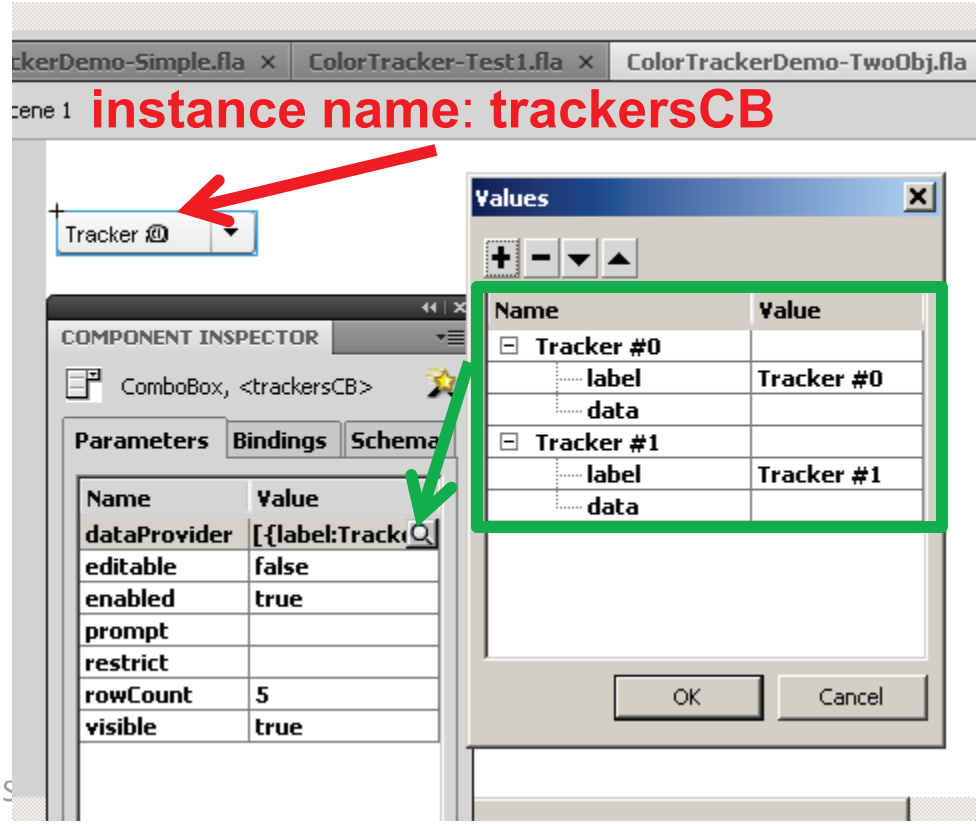
- **Idea**: add **a ColorTracker instance for EACH color object** to be tracked
  - E.g.,

```
var tracker1:ColorTracker = new ColorTracker(webcam);
var tracker2:ColorTracker = new ColorTracker(webcam);
```

- **Need set tracking color for each tracker**
  - **Different objects with same color are considered as a single object**

Example

- Tracking two objects with **different colors**
- Add a **ComboBox** instance to stage
  - Use **selected index** to check which tracker is being used
    - **trackersCB.selectedIndex**
    - For specifying tracking color



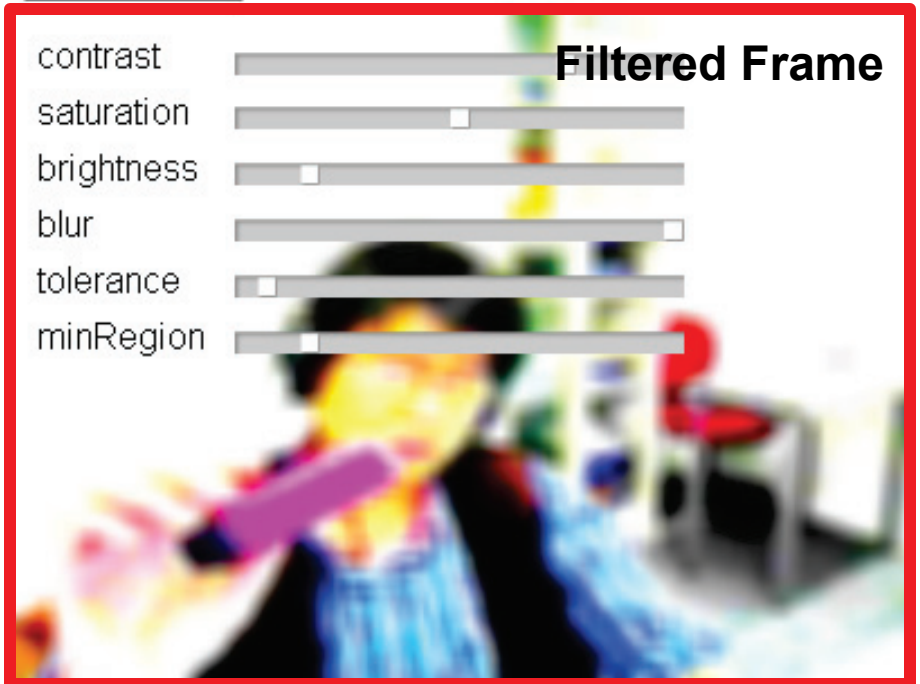
Parameter Settings

- Color tracking is sensitive to **both color being tracked and environment color/lighting**
  - **Filtering individual frames to make the color being tracked as visible as possible**
    - Done in **ColorTrackerPreprocessor.as**
      - You don't need bother this file
- **Different trackers require different parameters**
  - Use **ColorTrackerSettingUI** class
    - I did most of dirty work for you…

Tracker #0

contrast
saturation
brightness
blur
tolerance
minRegion

Filtered Frame

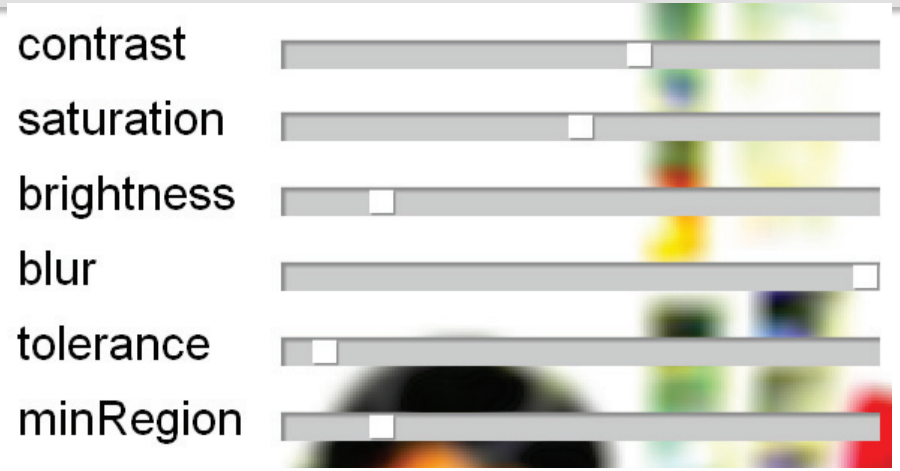


Final Processed Frame

ColorTrackerSettingUI Class

- A subclass of Sprite
- Constructor
  - `ColorTrackerSettingUI(showBmProcessed:Boolean = false, sideBySide = true)`
    - **Filtered frames are always visible (for parameter tuning)**
    - Use **showBmProcessed** to control if **final processed frames** should be shown or not
- Method
  - `setTracker(ct:ColorTracker) : void`
    - Set tracker for parameter tuning

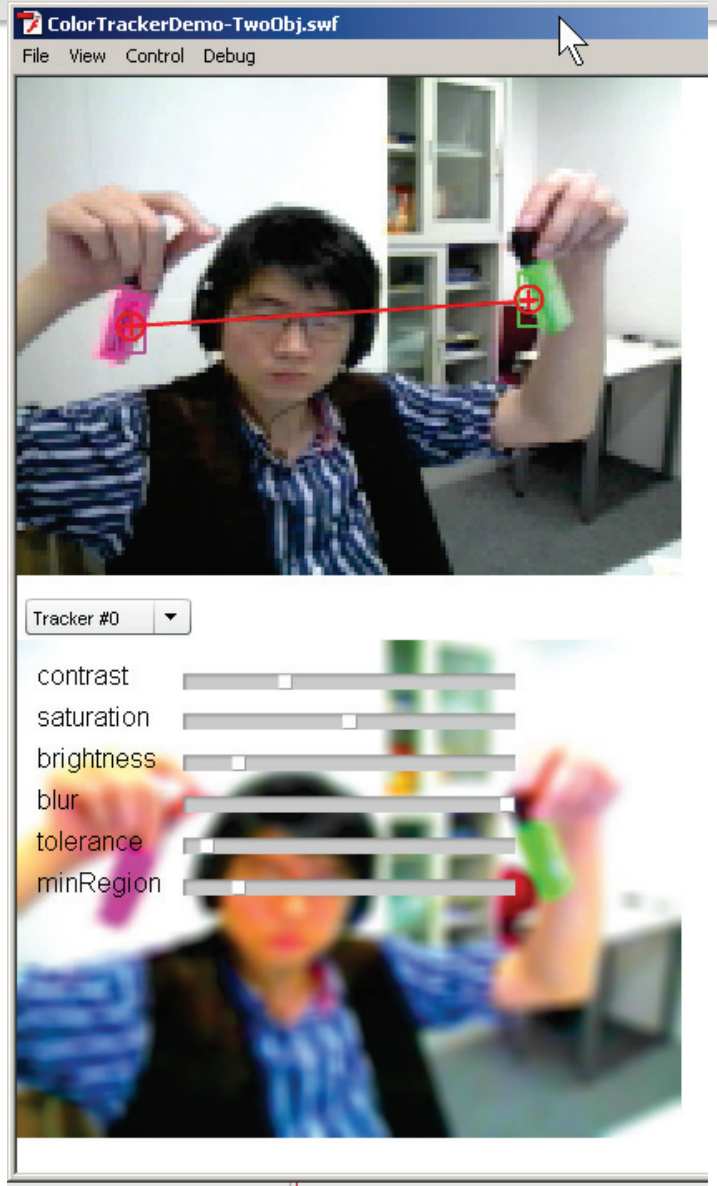
Available Parameters



- **First four parameters** are for **preprocessing filters**
  - I.e., BlurFilter and ColorMatrixFilter
  - Make **the color being tracked as distinguishable as possible**
- **tolerance**: **determine if a color is similar to tracking color**
  - Larger value → more colors considered as tracking color
- **minRegion**: **smallest size of a tracked region**

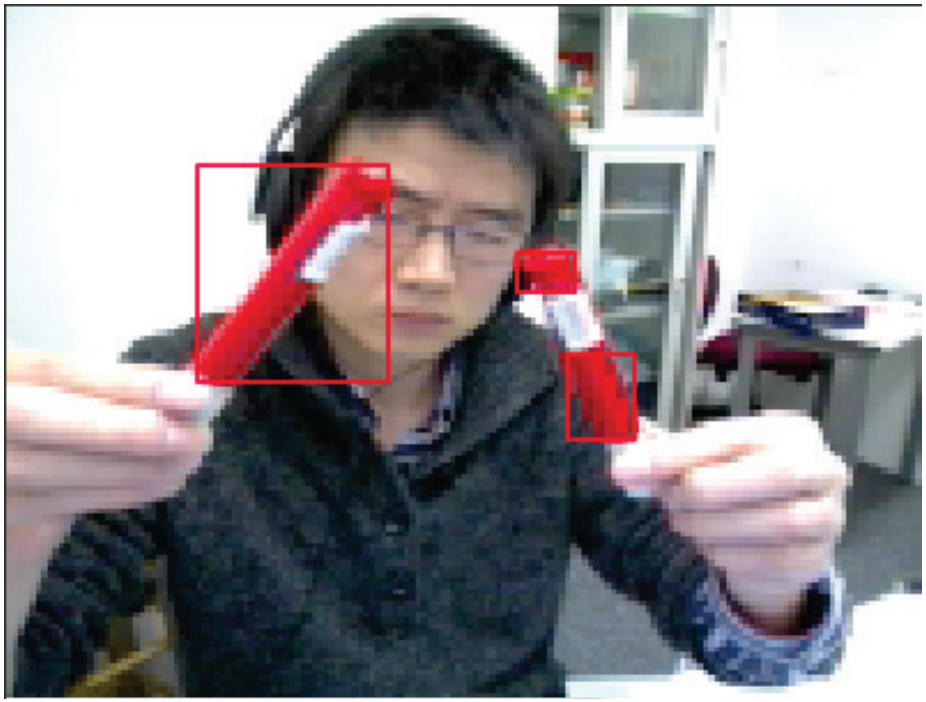
Class Exercise

- **Task 1**
  - **Connecting a line between centers of tracked regions**
    - Draw a line only if both objects are being tracked
- **Task 2**
  - **Specify clipping regions for individual trackers**



Case 2: Multiple Objects in Same Color

- All **disconnected** **regions with tracking color** are detected as **separate objects**
  - Implemented in **BlobDetector.as**



Case 2: Multiple Objects in Same Color

- To enable tracking multiple objects in same color
  - Set ColorTracker's property **`trackMultiObj`** to true

- To get the **array of rectangles** for enclosing tracked objects
  - Use **`ColorTracker`**'s **`trackRects`** property

Class Exercise

- **Task: gesture control by tracking a color object**
  - Need to get **paths of color object being tracked**
    - Simulate mouse movement
    - **Set low response to get smooth paths**

- **When to start/end recording movement path**?
  - Maybe use another color object to trigger gesture control start/end