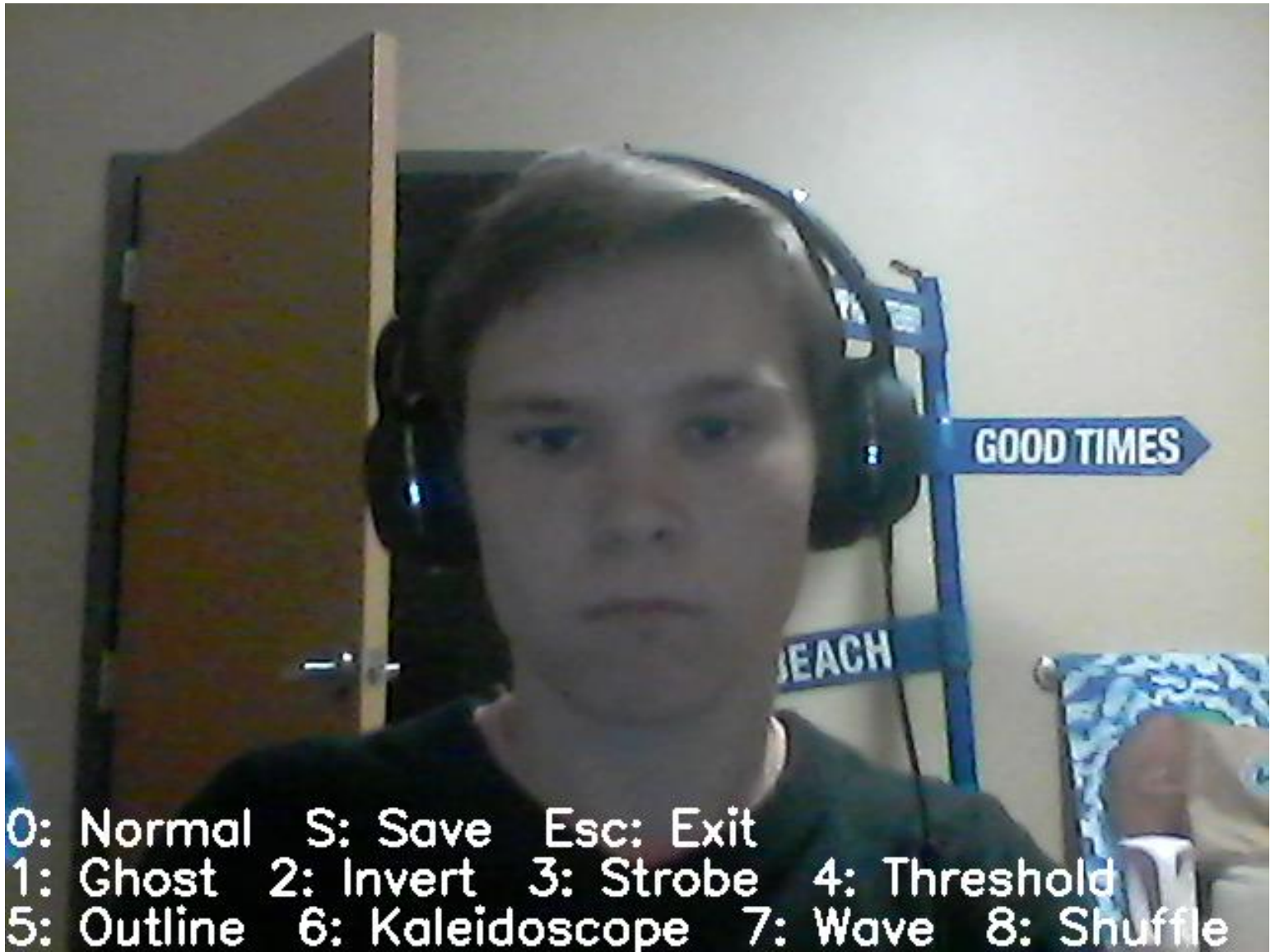
Assignment 06: Photobooth

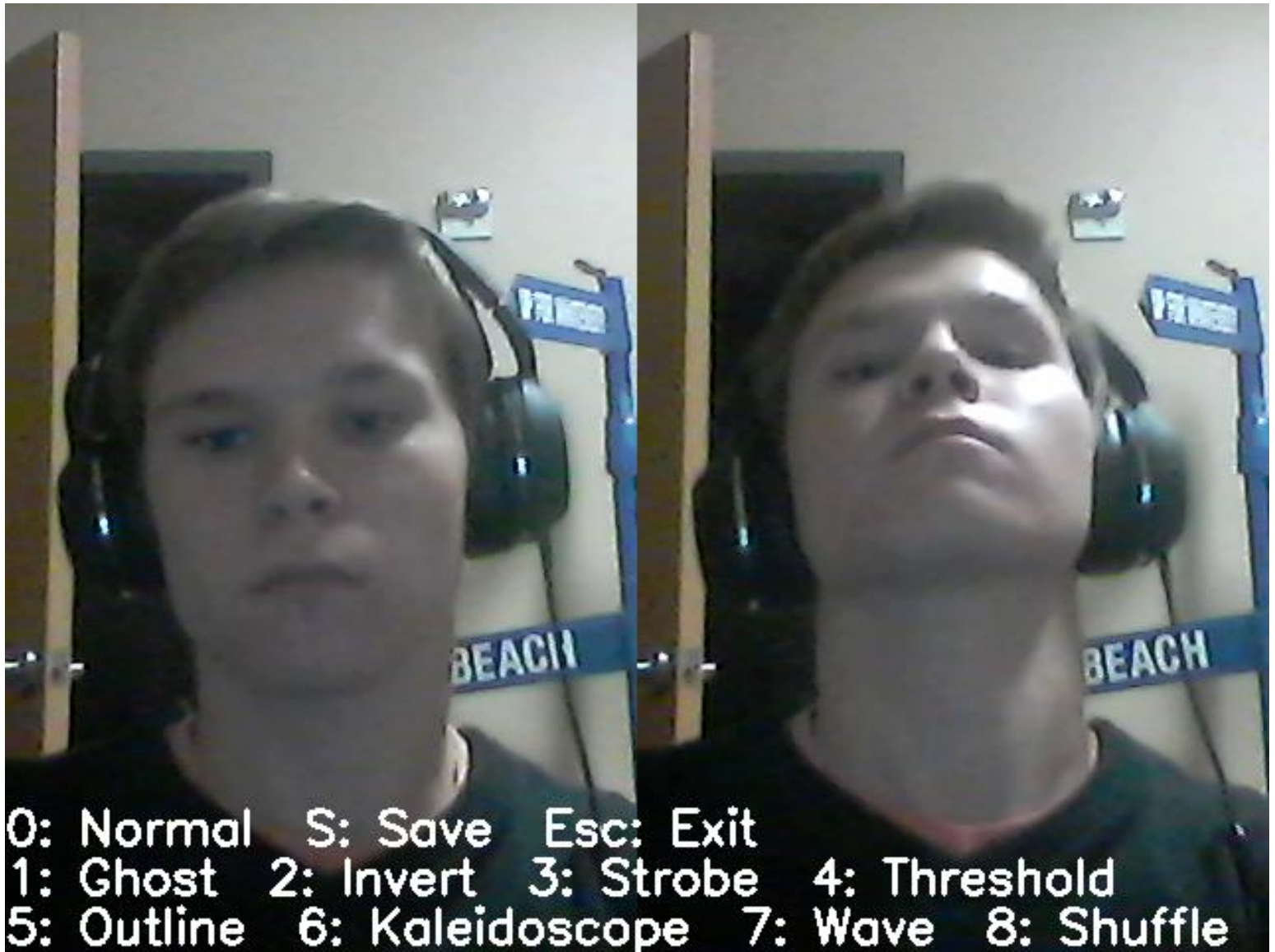
Brock Dvais

0: Normal


GOOD TIMES
0: Normal S: Save Esc: Exit
1: Ghost 2: Invert 3: Strobe 4: Threshold
5: Outline 6: Kaleidoscope 7: Wave 8: Shuffle

1: Ghost

- Shows the unedited picture from 30 frames ago



1: Ghost

```
last30.append(frame)
if len(last30)>30:
    last30.pop(0)

elif key==ord("1"):
    current=frame[:,w/4:3*w/4,:]
    ghost=last30[0][:,w/4:3*w/4,:]
    out=np.hstack((current,ghost))
```

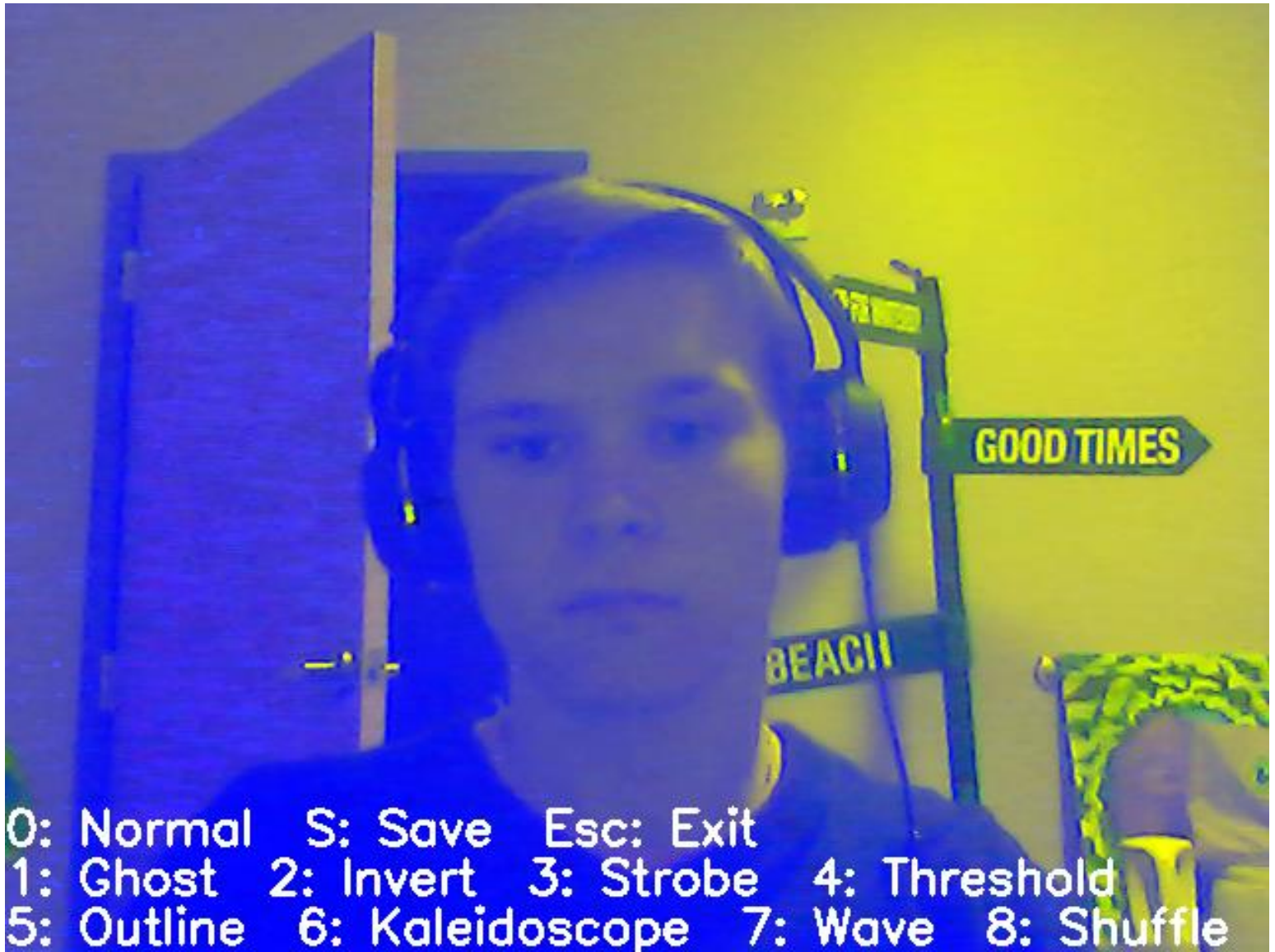
2: Invert

- Takes the inverse image of the read image

`out=255-frame`

3: Strobe

- Makes the screen strobe red, green, and blue



3: Strobe

```
if (i%6)/2==0:
    out=np.dstack((255-frame[:,:,0],
                   frame[:,:,1],
                   frame[:,:,2]))
elif (i%6)/2==1:
    out=np.dstack((frame[:,:,0],
                   255-frame[:,:,1],
                   frame[:,:,2]))
else:
    out=np.dstack((frame[:,:,0],
                   frame[:,:,1],
                   255-frame[:,:,2]))
```

4: Threshold

- Thresholds each color of the image. The threshold value varies back and forth based on the frame

4: Threshold

```
threshold=abs(256-(i%511))
out=frame.copy()
out[out<=threshold]=0
out[out>threshold]=255
```

5: Outline

- Shows the colored energy map of the image

5: Outline

```
frame=np.float32(cv2.GaussianBlur(frame,(5,5),0))
derivative_kernel=np.float32([[-1,0,1],[-2,0,2],[-1,0,1]])
Ix=cv2.filter2D(frame,-1,derivative_kernel)
Iy=cv2.filter2D(frame,-1,derivative_kernel.T)
out=(Ix**2+Iy**2)**.5
out-=np.min(out)
out/=np.max(out)
out*=255.999999
out=np.uint8(out)
```

6: Kaleidoscope

- This is an 8 way kaleidoscope based on the top left corner of the image

6: Kaleidoscope

```
polar=np.zeros((h,w,2),dtype=np.float32)
```

- This creates an array the same size of the frame that saves the equivalent polar coordinates with the origin as the center of the array

```
x_dist=np.abs(x-np.ones((h,w))*w/2)
y_dist=np.abs(y-np.ones((h,w))*h/2)
polar[:,:,0]=(x_dist**2+y_dist**2)**.5
```

- This gives the distance from the origin

6: Kaleidoscope

```
polar[:,:,1]=np.minimum(np.arctan2(y_dist,x_dist),
                        np.arctan2(x_dist,y_dist))
```

- This gives the angle of the point in polar coordinates, but only in the first 45 degree range, because x_dist and y_dist are all positive values, and it takes the minimum of two supplementary angles.

```
new_rect=np.ones((h,w))
x_coord=polar[:,:,0]*np.cos(polar[:,:,1])
y_coord=polar[:,:,0]*np.sin(polar[:,:,1])
```

- This turns the polar coordinates back into Cartesian coordinates to use for the remap function

7: Wave

- This creates a sine wave with the coordinates of the picture

7: Wave

```
x_wave=(x+3*np.sin(2*np.pi*(x)/(w/8)))
y_wave=(y+5*np.sin(2*np.pi*(x)/(h/4)))
```

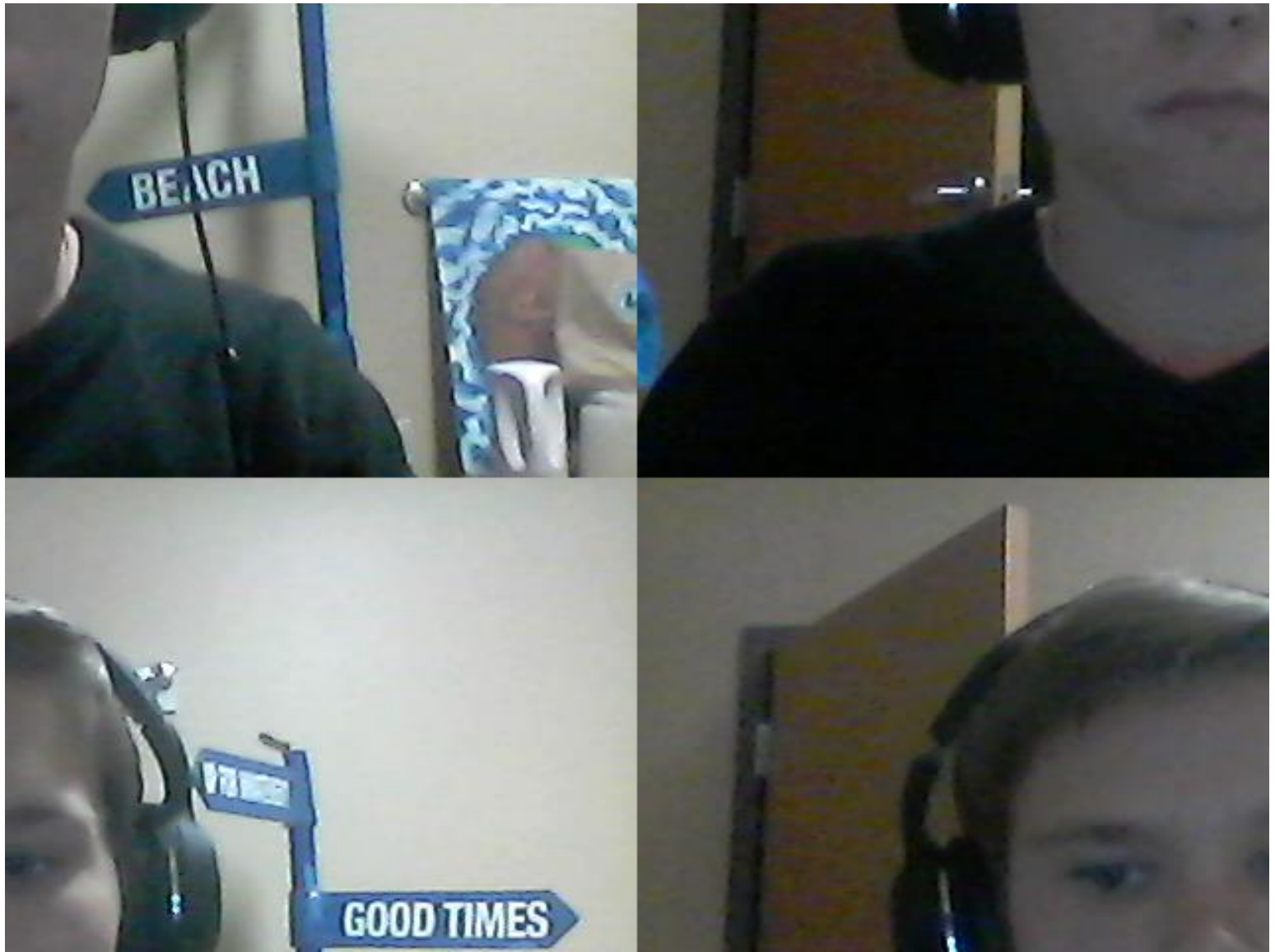
8: Shuffle

- This shuffles the four quadrants of the picture

A	B
C	D

⇨

D	C
B	A



8: Shuffle

```
x_shuff[:h/2,:]=(x_shuff[:h/2,:]+(w/2))%w
y_shuff[:,:w/2]=(y_shuff[:,:w/2]+(h/2))%h
x_shuff[h/2:,:]=(x_shuff[h/2:,:]-(w/2))%w
y_shuff[:,w/2:]=(y_shuff[:,w/2:]-(h/2))%h
```