```
class KinectTracker {
  // Depth data
  int[] depth;
  PImage videoImage;

  void track() {
    // Get the raw depth as array of integers
    depth = kinect.getRawDepth();
    // Being overly cautious here
    if (depth ==null)  {
      println("WHHHOOOOOA ARRAY LENGTH MISMATCH");
      return;
    }
    // re-init presence measurements just to make sure something else isn't changing them
    presenceFront = 0;
    presenceMiddle = 0;
    presenceBack = 0;
     videoImage = kinect.getVideoImage();// Read a new video frame
    videoImage.loadPixels();//display the frame
    for (int x = 0; x < kw; x++) {
      for (int y = 0; y < kh; y++) {
        // Mirroring the image
        int offset = kw-x-1+y*kw;
        int i = offset;
        //
        int currColor = videoImage.pixels[i];
        int bkgdColor = averagePixels[i];//turning the pixel average into an array that I

        //setting what things look like now to be referenced next frame
        int currR = (currColor >> 16) & 0xFF;
        int currG = (currColor >> 8) & 0xFF;
        int currB = currColor & 0xFF;

        //setting the currColors last frame to the bkgdColor this frame
        int bkgdR = (bkgdColor >> 16) & 0xFF;
        int bkgdG = (bkgdColor >> 8) & 0xFF;
        int bkgdB = bkgdColor & 0xFF;

        //resets the background to not see anything
        if (resetBackground) {
          averagePixels[i] = videoImage.pixels[i];
          ;
         }

        //subtracted background values
        int  bgSubR  =abs(currR - bkgdR);
        int  bgSubG  =abs(currG - bkgdG );
        int  bgSubB  =abs(currB - bkgdB);

        // background subtracted image
        bgSubAndThreshPixels[i] = 0xFF000000 | (bgSubR << 16) | (bgSubG << 8) | bgSubB;
        // threshold it!
```

```
      bgSubAndThreshPixels[i] =brightness(bgSubAndThreshPixels[i]) > thresh ? 1 : 0 ;//


    // Grabbing the raw depth
    int rawDepth = depth[offset];
    // eliminate any "background" depth measurements
    int correctedDepth = rawDepth * bgSubAndThreshPixels[offset];

     bgSubAndThreshPixels[i] =color(map(correctedDepth, 0, 2047, 0, 255));

    // start counting pixels
    if (correctedDepth > minDepth/ 2) {//map(mouseX, 0, width, 0, 2047)) {
      if (correctedDepth < minDepth) {
        presenceFront++;
       }
      else if (correctedDepth < maxDepth) {
        presenceMiddle++;
       }
      else {
        presenceBack++;
      }
     }
    else {
    }
  }
}

if (resetBackground) {
   resetBackground =false;// turn the background reset back off
}

int  []list = {
  presenceFront, presenceMiddle, presenceBack
};

int currentPos = -1;
int maxPresence  =max(list);

// HERE WE NAME OUR CURRENT POSITION BASED ON PRESENCE VALUES
// LOTS OF LOGIC COMING UP!!!!!
if (maxPresence < minimumPresence) {
  currentPos = NOTHING;
 }
else if (maxPresence == presenceFront) {
   currentPos =ENTER;
 }
else if (maxPresence == presenceMiddle) {
  currentPos = RECORDING;
 }
else if (maxPresence == presenceBack) {
  currentPos = SOUND_PLAYBACK;
}
```

```
if (currentPos == whereIsViewer) {
  // we haven't moved, so do nothing
  return;
}
// then do checks cause something changed
if (whereIsViewer == NOTHING) {
  if (currentPos  ==ENTER) {
     whereIsViewer =ENTER;
    // nothing special happens here
   }
  else if (currentPos == RECORDING) {
    whereIsViewer = RECORDING;
    enteredRecordingArea();
   }
  else if ( currentPos == SOUND_PLAYBACK) {
    // this is an unusual situation .. it probably means that there was a tracking err
    enteredInvalidState();
  }
 }
else if (whereIsViewer  ==ENTER) {
  if (currentPos == NOTHING) {
    whereIsViewer = NOTHING;
    exited();
    // nothing special happens here
   }
  else if (currentPos == RECORDING) {
    whereIsViewer = RECORDING;
    enteredRecordingArea();
   }
  else if ( currentPos == SOUND_PLAYBACK) {
    // this is an unusual situation .. it probably means that there was a tracking err
    enteredInvalidState();
  }
 }
else if (whereIsViewer == RECORDING) {
  if (currentPos == NOTHING) {
    whereIsViewer = NOTHING;
    exited();
    // nothing special happens here
   }
  else if (currentPos  ==ENTER) {
     whereIsViewer =ENTER;
    /// ???????
   }
  else if ( currentPos == SOUND_PLAYBACK) {
    whereIsViewer = SOUND_PLAYBACK;
    enteredSoundPlayback();
  }
 }
else if (whereIsViewer == SOUND_PLAYBACK) {
  if (currentPos == NOTHING) {
```

```
      whereIsViewer = NOTHING;
      exited();
      // nothing special happens here
     }
    else if ( currentPos ==ENTER) {
      // we are leaving so ignore
      // it must have skipped recording.
       whereIsViewer =ENTER;
     }
    else if (currentPos == RECORDING) {
      whereIsViewer = RECORDING;
      // we are leaving so ignore
    }
  }
}

void enteredRecordingArea() {
  println("ENTERED RECORDING AREA!~!!!!");
}

void enteredInvalidState() {
  //println("ENTERED INVALID STATE!!"); //though I could have it print that, it wouldn't
}

void exited() {
  println("EXITED!!");
   isRecording =false;
  currentPos = -1;
  whereIsViewer = NOTHING;
  lightsOn();
}

void enteredSoundPlayback() {
  lightsOff();
  println("ENTERED SOUND PLAYBACK AREA!~!!!");
  setup();
}


void lightsOff() {//starts all of the things that we need to happen when the viewer ente
  port.write(0);//lights off
  println("LIGHTS OFF!!!!");
   OscMessage myMessage =new OscMessage("/play");//sound on
  myMessage.add(1); // add an int to the osc message
  oscP5.send(myMessage, myRemoteLocation);
  println("SENT");
}

void lightsOn() {//does all of the "reseting" things that happen when the viewer leaves
   OscMessage myMessage =new OscMessage("/play");//sound off
  myMessage.add(0); /* add an int to the osc message */
  oscP5.send(myMessage, myRemoteLocation);
```

```
    println("NO MORE SOUND");
    port.write(255);//lights on
    println("LIGHTS ON!!!!!!!");
     resetBackground =true;//reset the background again, now that things have change and h
  }

  /* incoming osc message are forwarded to the oscEvent method. */
  //just in case max wants to send us a message
  void oscEvent(OscMessage theOscMessage) {
    /* print the address pattern and the typetag of the received OscMessage */
    print("### received an osc message.");
    print(" addrpattern: "+theOscMessage.addrPattern());
    println(" typetag: "+theOscMessage.typetag());
  }


  void quit() {
    kinect.quit();
  }
}
```