Week 10

This Week:

Ball Detection:

This week, as I mentioned in my lab report, I wanted to hook up James' threshold code into the contour detection from the depth camera work. This was easy enough to do, however, we ran into the problem where there could be multiple contours on screen just by chance. To combat this I created a simple loop that calculates the area of the contour based on it's moments and counts the largest contour as the ball. This function exists in a ContourTracker class that contains all the logic for contour/ball detection:

```
cv::Point ContourTracker::findBallCenter(){
  double largestArea = 0;
  int ballx = -1;
  int bally = -1;
  for (std::vector<cv::Point> cont : contours){
    // Calculate Area
    double currArea = cv::contourArea(cont);
    if(currArea < largestArea){
      continue;
    }
    largestArea = currArea;

    // Calculate Center if Area is Largest
    cv::Moments currM = cv::moments(cont);
        if(currM.m00 != 0){
            ballx = (int)(currM.m10/currM.m00);
            bally = (int)(currM.m01/currM.m00);
    }
  }
  return cv::Point(ballx,bally);
}
```

Graphics:

This week, after creating simple graphics functions in the previous week. Our team decided it would be beneficial to add the functionality of loading in images. In my computer graphics course we accomplished a similar task when doing texture mapping.

So, I used and adapted code from a graphics homework assignment, and integrated it with OpenCV mats. This function allows you to specify an X and Y coordinate to draw the top left corner of the TIFF. TIFF is the file format we chose to use because it can be easily exported from GIMP, and does not have any compression (by choice) so we don't loose any quality when loading an image, and we can easily write pixel data to a buffer. This long function is included in Projector and listed below:

```
void Projector::renderTiff(std::string& fname, int xOff, int yOff){
  // Get and easy reference to the projector
  Projector &proj = *this;
  // Convert to Cstring for TIFF reading
  char * cstr = new char[fname.length() + 1];
  strcpy_s(cstr, fname.length()+1, fname.c_str());
  // Attempt to open TIFF
  TIFF* in = TIFFOpen(cstr, "r");
  // If TIFF cannot be opened
  if (in == NULL) {
  std::cout << fname << " could not be opened" << std::endl;
    return;
  }
  // TIFF opened
  if (in) {
    // Find out TIFF dimensions
    uint32 imageLength;
    TIFFGetField(in, TIFFTAG_IMAGELENGTH, &imageLength);
    int imageW = (int) TIFFScanlineSize(in) / (sizeof(unsigned char) * 3);
    int imageH = imageLength;
    std::cerr << "TIFF WIDTH: " << imageW << std::endl;
    std::cerr << "TIFF HEIGHT: " << imageH << std::endl;

    // Read the TIFF image
    size_t nPixels;
    uint32* raster // Image buffer
    nPixels = imageW * imageH;
    raster = (uint32*)_TIFFmalloc(nPixels * sizeof(uint32));
    if (raster != NULL) {
      if (TIFFReadRGBAImage(in, imageW, imageH, raster, 0)) {
        for (int i = 0; i < imageH; i++) {
          for (int j = 0; j < imageW; j++) {
            // Check the bounds of the projector
            if(xOff + j >= proj.w)
              break;
            if(yOff + i >= proj.h)
              break;
            if(xOff + j < 0)
              continue;
            if(yOff + i < 0)
              continue;
            // Get RGB values from unsigned int
```

```
              uint32 currCol = raster[i *imageW + j];
              // This could be wrong, need to test
              char r = (currCol >> 16) & 0xFF;
              char g = (currCol >> 8) & 0xFF;
              char b = (currCol) & 0xFF;
              // Set the current pixel from the raster
              proj.display.at<cv::Vec3b>(j + xOff, i + yOff) = cv::Vec3b(b,g,r);
            }
          }
        }
        _TIFFfree(raster);
      }
      TIFFClose(in);
    }
  }
```

Because our project laptop is a shared resource, this code has not yet been tested on the RACHEL laptop and may contain bugs.

Game Logic:

The last thing I worked on this week was adding a time between point scored. This means that as soon as a double bounce triggers a point, there will be a bit of “down time” where a double bounce can’t trigger another point. This will help to eliminate accidental points from multiple bounces on one side. To do such a thing, I added a new state to the game logic code, and made a conditional that starts a new thread if the score was updated from a bounce. This thread changes the game state, and when it is in the IDLE state, it will not attempt to process any bounce data:

```
StatusChange bounceChange = NO_CHANGE;
        if(gameStatus != IDLE){
            bounceChange = handleBounce(bounce);
        }
        StatusChange buttonChange = handleButton(button);
        if((bounceChange == SCORE_CHANGE) || (buttonChange == SCORE_CHANGE)) {
            if(bounceChange == SCORE_CHANGE){
                // If change was because of bounce, start the timer
                std::thread timer (timeOut);
            }
...

// Thread for timeout between bounce scores
void timeOut(){
    gameStatus = IDLE;
    std::this_thread::sleep_for (std::chrono::seconds(1));
    gameStatus = ACTIVE;
}
```

However, on second glance, it seems Micah has already made an implementation for this, so this code will probably be scrapped alas.

Next Week:

Ball Detection:

There needs to be a better way to filter out the contours. I originally had an idea to sample the depth camera at the center of each contour and use that depth in conjunction with the contour area, to determine the physical objects actual size, then find the range that the ping pong ball lies in. However, after discussing with our TA. It seems that just finding the range that the contour falls in, might be enough to correctly find the ball, and will not require sampling of multiple sensors.

Graphics:

The load tiff function still needs to be tested. Once it is, I will implement it into the Pause Menu and Start Menu systems to display information about what the button matrix does at each menu. I will also help to implement the logic that displays the pause menu and halts the game.

Game Logic:

For game logic, I still need to find out why the score was incrementing extremely fast and further develop the Main menu class that instantiates the game loop.e