Week 12

This Week:

Ball Tracking:

This week I built on the work I did last week and added positioning of another point to define the edges of the table. Currently, this detection method assumes that the table is at a perfect 90 degrees to the table. This is not completely unnecessary to assume as the projector and camera are mounted to the same piece of plywood. This means that if the table ever goes skew then it can be readjusted by lining it up with the projector. Now that we have the points of the table defined within the space of the camera, it is easy to translate coordinates from camera space to table space. This is what we now do in the Table class:

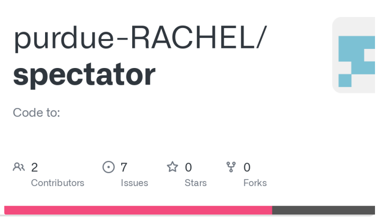
```
cv::Point2f retPos;
retPos.x = (currPos.x - table.top_left.x) / (table.bottom_right.x - table.top_left.x);
retPos.y = 1- (currPos.y - table.top_left.y) / (table.bottom_right.y - table.top_left.y);
```

The rest of the class can be found here:

spectator/Table.cpp at working · purdue-RACHEL/spectator

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode

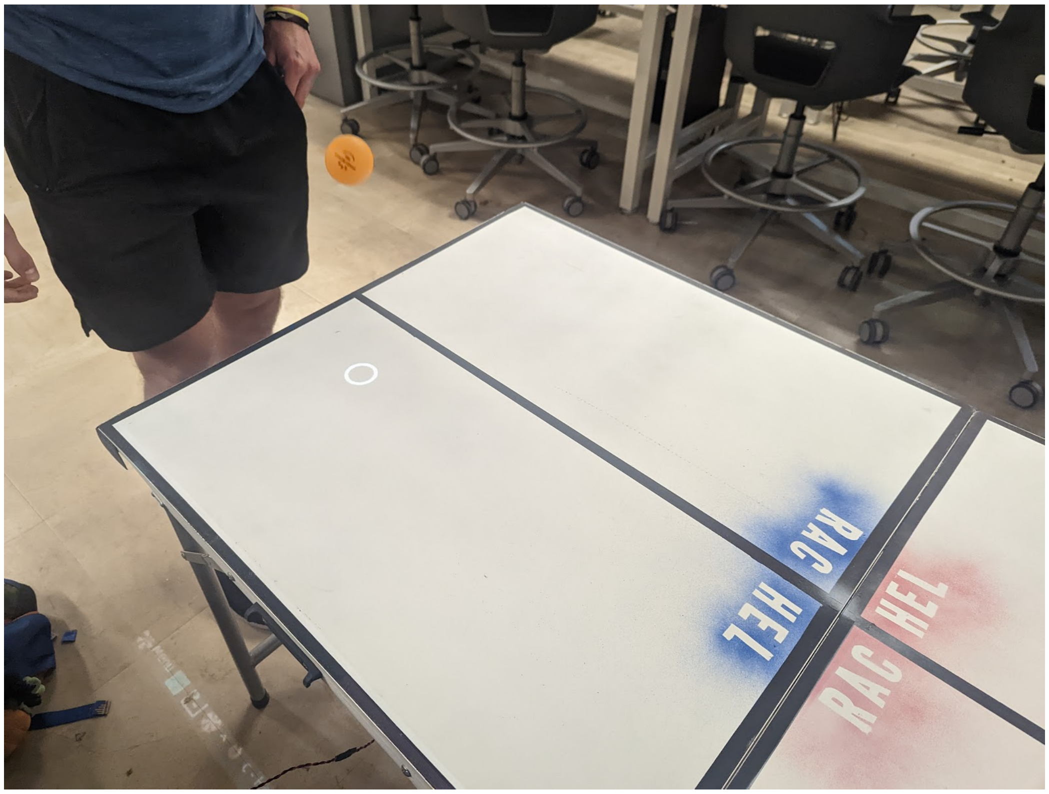
https://github.com/purdue-RACHEL/spectator/blob/working/src/Table.cpp


purdue-RACHEL/
spectator

Code to:

2 Contributors · 7 Issues · 0 Stars · 0 Forks

We can now with the table coordinates of the ball, display a circle to the table around where our camera and system thinks the ball is. This is the demo I set up.



The position of the ball drawn to the table

This demo works decently well, however there is a noticeable delay of about .25-.5 seconds. There is also more of an offset from the ball as the ball approaches the edges of the table. All in all though, this demo accomplishes the goals of our stretch goal.

Hardware:

This week I drag soldered the STM32 to our board and tested UART with James. We discovered that the baud rate of our assembled PCB differed from our test hardware. Bartosz and Micah analyzed the UART signal and discovered that the clock was oscillating at 9Mhz instead of 48Mhz. We tried to change this configuration within our code but to no luck. Eventually we just adjusted for the clock manually within our UART code but, Micah’s triangulation plans may be affected by this.

Additionally with the button matrix that was ordered an assembled, we have had a few problems with two buttons being almost coupled. When one is pressed it registers the other one as well. We spent a bit of time debugging this issue to no avail. I believe this strange behavior is due to the very close nature of the traces, as they run parallel along the button matrix. Either way, we solved this by setting a specific sampling rate, this then ignores the small incorrect signal that gets sent.

Finally we received our plastic cut housing pieces today, so we assembled these to the board, and the result is pretty satisfying.

Next Week:

Ball Detection:

Now that we can gather relatively accurate data about the position of the ball, we can start to work our way toward the development of drop shot the minigame. This requires us to know where the ball bounced. After incorporating UART messages into the table test demo, we discovered that if we read the position of the ball immediately after receiving a UART message specifying a bounce, we end up with a coordinate that differs sizably from the real world bounce. This is caused by the delay between the actual bounce, and when the C++ on the laptop finally read the ball position. To account for this I plan to put the ball detection code in it’s own thread and have it collect a buffer of ball position data at a specified sampling rate. The ball position data within the buffer will contain a time value of the position that is calculated using the system clock. From here I will attempt to calculate the delay between a bounce, and a UART message, calculate the real bounce time from this data, and then use this time to look up position data within the buffer.

Minigame:

Bartosz spent some time writing the initial code for drop shot, so I will attempt to incorporate the ball detection into the main game loop.

Projector:

Dropshot also requires the ability to render regions to the table. This means that the projector will have to be equipped with a function such as render hexagon to display the cells of the game.