Week 9

This Week:

Build Tool:

We are all developing on our computers, however, when we want to test our code, we would prefer to run it on our project laptop. There are two main reasons for this:

1. The OpenCV libraires are time consuming to compile and link, and James has gone through this process already on the project laptop.
2. The project laptop runs linux, which is important for our UartDecoder that reads and writes from a linux tty file.
    - This could be changed to work with windows, however I have not done this.

For these main reasons, and for consistency sake, we will for the majority of the time, compile and run our code on the project laptop. Only one person can use the laptop at a time, and would ideally like to compile and test their file with little to no resistance.

Therefore, to aid with testing multiple files, I upgraded our SConstruct file to support multiple builds:

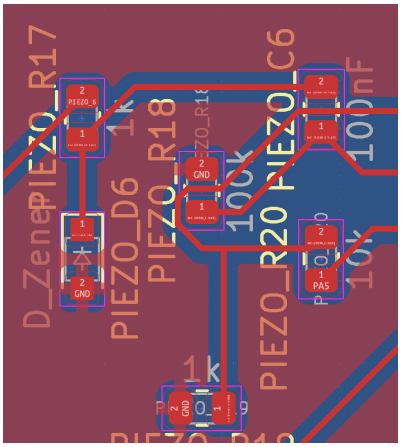
```
# EXAMPLE CALL:
# To build projector you would type
# scons --build=Projector

import os

# Set up build option parameter
AddOption('--build',
        dest='buildOption',
        type='string',
        nargs=1,
        action='store',
        help='Options: GameLoop, Uart, Camera, Projector')

# Defined directories
BUILDDIR='/home/rachel/git/spectator/build/'
SRCDIR='/home/rachel/git/spectator/src/'
VariantDir(BUILDDIR,SRCDIR)

# Set up enivronment
env = Environment(CCFLAGS='-fpermissive -I/home/rachel/git/spectator/inc/OpenNI2/Include -I/usr/local/include/opencv4',
        LIBS=['OpenNI2', 'opencv_core', 'opencv_features2d', 'opencv_imgcodecs', 'opencv_highgui', 'opencv_video', 'opencv_imgproc', 'openc
        LIBPATH=['/home/rachel/git/spectator', '/usr/local/lib'],ENV = os.environ)

# Add callback command to run and clean if build is successful
RUN = env.Command(target = 'runme', source=BUILDDIR+'runme', action=[BUILDDIR+'runme','rm -rf '+BUILDDIR,'mkdir '+BUILDDIR])
env.AlwaysBuild(RUN)

# Build program depending on build option parameter
BUILDOPTION = GetOption('buildOption')
if BUILDOPTION == None:
    print('Specify a build option using --build')
    print('Options: GameLoop, Uart, Camera, Projector')
    exit()
elif BUILDOPTION == 'Uart':
    # Add test flags that surround the main function you want to run
    env.Replace(CPPDEFINES = 'TESTUART')
    # Add all the cpp files needed. Keep the executable name 'runme' the same
    env.Program(BUILDDIR+'runme', [BUILDDIR+'UartDecoder.cpp'])
elif BUILDOPTION == 'GameLoop':
    env.Replace(CPPDEFINES = 'TESTGAMELOOP')
    env.Program(BUILDDIR+'runme', [BUILDDIR+'GameLoop.cpp',BUILDDIR+'UartDecoder.cpp'])
elif BUILDOPTION == 'Projector':
    env.Replace(CPPDEFINES = 'TESTPROJECTOR')
    env.Program(BUILDDIR+'runme', [BUILDDIR+'Projector.cpp',BUILDDIR+'UartDecoder.cpp'])
elif BUILDOPTION == 'Camera':
    env.Replace(CPPDEFINES = '')
    env.Program(BUILDDIR+'runme', [BUILDDIR+'cameraBlob.cpp',BUILDDIR+'CameraInterface.cpp'])
```

This new SConstruct will automatically run the executable after a successful build and clean after execution. Each build option specifies the cpp files and Cflags it needs. Then the main function for the build specified is surrounded in corresponding #ifdef statements. Additionally, the cpp files we edit are now located in a src directory, and the build files are stored in a build directory, so our repo is more organized.

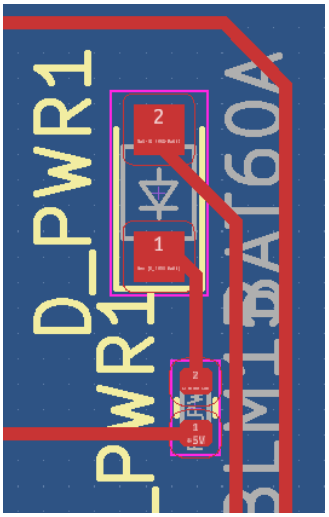
The main problem I had was running the projector class with this SConstruct file. I kept receiving an error saying that the GTK window could not be displayed. I eventually figured out that this was a problem with the environment. The fix was adding `ENV = os.environ` to the environment setup. This makes the environment that the binary builds and runs in the same as the environment that calls scons, giving access to things like the X11 window system.

PCB Design:

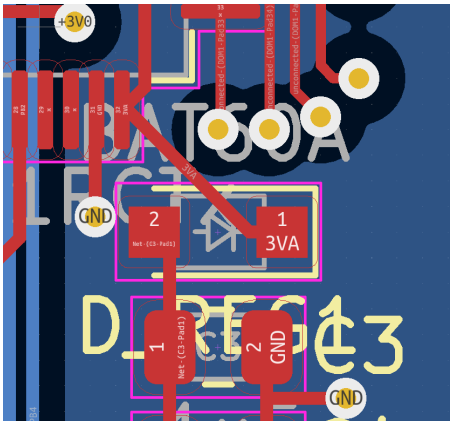
Based on feedback from instructors I modified the footprint of the 10 diodes on our PCB. The original footprint was SOD 323, and the new one is SOD 323 hand solder. Now the pads are larger and will be easier to assemble.



Piezo Filter Diode



USB C Power Diode



Voltage Regulator Diode

UI:

This week, I worked more on the wrapper functions for OpenCV that will allow us to display simple graphics to the screen. The specific functions that took the most work were `writeRotateText()` which allows us to easily display text to either side of the table with the correct orientation. The reason this function was a little tricky is because OpenCV only supports writing text horizontally. So an intermediate matrix was used and transformed and then combined with the display matrix to write the text. Below is the transformations preformed when rotating 90 degrees clockwise:

```
cv::putText(rotImage, text, cv::Point(y,proj.w - x),
      cv::FONT_HERSHEY_SIMPLEX, size, cv::Scalar(b,g,r), 20, CV_8UC3);
    cv::transpose(rotImage,rotImage);
    cv::flip(rotImage,rotImage,1);
```

After this function was written, I could create a higher level function `updateScore(int,int)` that allows for easy updating from the game loop. Here is how the score is currently displayed to the projector:

```
proj.drawCenterLine();
proj.updateScore(score_red,score_blue);
proj.refresh();
```

`refresh()` will erase display the current frame buffer to the window and clear it from the object. Similar to a traditional graphics pipeline.

Simple UI Displayed to Table

Next Week:

UI:

Next week I will use the base I have created this week and try to implement interactive UIs that detail the commands of each button on the keypad matrix. These UIs will primarily include the main menu, which will be it's own object that calls instances of gameloop.

Computer Vision:

This week James used RGB data from the camera to filter out the yellow ball atop our table. Next week, because I set up contour detection on the depth data, I will attempt to hook up the filtered image that James created, to a contour detection algorithm, that will save the x and y position of the detected blob.

Button Matrix:

I still need to order the button matrix PCB from JLCPCB

PCB:

We should expect to see our PCBs coming in soon. I expect to do a lot of surface mount soldering. I am quite a fan of it.