WEEK 10:

Hours: 8

This week I continued work on color tracking, albeit indirectly. One of the major problems the team has been battlling when testing color-based tracking of the table tennis balls is the automatic exposure and white balance control present in our camera module. These features, while helpful for extracting the best possible picture out of the camera sensor in dynamic lighting conditions, make It difficult to rely on the color and luminosity values of objects in the playing area. To solve this problem, I have been writing code to extend my `CameraInterface` class so that it can change the settings within the camera. OpenNI2 provides part of the interface, but I wanted to be able to refer to properties by their names instead of by number, and automatically handle filling the repetitive and verbose parameters of the OpenNI2 functions. The next step on this part of the project will be to extend last week's ColorPicker program to also control the exposure, sensor gain, white balance, and other parameters. Eventually, I think the color picker program will evolve into a calibration tool for setting up the table in new environments, incorporating the microphones and projector.

Also this week, I spent about 3 hours restructuring the build script that the team uses for all of our submodules. Before my changes, the different build targets and therir respective parameters were stored in a long if-else tree, which was also highly repetitive. I decided to changed this to a structure where the build targets and parameters are stored in an associative container with the mdoule names as keys. This means that the code for actually building an executable, which was identical amongst all of the build targets, was written only once instead of being repeated. This change should reduce the frequency of typos when adding or modifying build targets, as well as making It easier to read the code where the parameters are specified, by removing all the noise of repeated function names and arguments.

```
45  # Build program depending on build option parameter
46  BUILDOPTION = GetOption('buildOption')
47  if BUILDOPTION == None:
48      print('Specify a build option using --build')
49      print('Options: GameLoop, Uart, Camera, Projector, ColorPicker')
50      exit()
51  elif BUILDOPTION == 'Uart':
52      # Add test flags that surround the main function you want to run
53      env.Replace(CPPDEFINES = 'TESTUART')
54      # Add all the cpp files needed. Keep the executable name 'runme' the same
55      env.Program(BUILDDIR+'runme', [BUILDDIR+'UartDecoder.cpp'])
56  elif BUILDOPTION == 'GameLoop':
57      env.Replace(CPPDEFINES = ['TESTGAMELOOP', 'DISABLEOPENCV'])
58      env.Program(BUILDDIR+'runme', [BUILDDIR+'GameLoop.cpp',BUILDDIR+'UartDecoder.cpp',BUILDDIR+'Projector.cpp'])
59  elif BUILDOPTION == 'Projector':
60      env.Replace(CPPDEFINES = 'TESTPROJECTOR')
61      env.Program(BUILDDIR+'runme', [BUILDDIR+'Projector.cpp',BUILDDIR+'UartDecoder.cpp'])
62  elif BUILDOPTION == 'Camera':
63      env.Replace(CPPDEFINES = '')
64      env.Program(BUILDDIR+'runme', [BUILDDIR+'cameraBlob.cpp',BUILDDIR+'CameraInterface.cpp', BUILDDIR+'ColorTracker.cpp', BUILDDIR+'ContourTracker.cpp'])
65  elif BUILDOPTION == 'ColorPicker':
66      env.Replace(CPPDEFINES = '')
67      env.Program(BUILDDIR+'runme', [BUILDDIR+'colorPicker.cpp',BUILDDIR+'CameraInterface.cpp', BUILDDIR+'Projector.cpp', BUILDDIR+'UartDecoder.cpp'])
```

Shown on this page and the previoius page are before and after views of the build script showing how the execution concerns have been separated from the business logic, and the names/purposes of parameters have been made more explicit.

```
16  # Define build options
17  BUILDOPTS = ['GameLoop', 'Uart', 'Camera', 'Projector', 'ColorPicker', 'BuildTest']
18
19  # create empty dictionary to store buildflags and arguments for env.Program()
20  BUILDDICT = {opt: {'defs': list(), 'source': list()} for opt in BUILDOPTS}
21
22  BUILDDICT['GameLoop'] = {
23      'defs': ['TESTGAMELOOP', 'DISABLEOPENCV'],
24      'source': [BUILDDIR+'GameLoop.cpp',BUILDDIR+'UartDecoder.cpp',BUILDDIR+'Projector.cpp']
25      }
26  BUILDDICT['Uart'] = {
27      'defs': ['TESTUART'],
28      'source': [BUILDDIR+'UartDecoder.cpp']
29      }
30  BUILDDICT['Projector'] = {
31      'defs': ['TESTPROJECTOR'],

76  # Build program depending on build option parameter
77  CLEAN = GetOption('clean')
78  BUILDOPTION = GetOption('buildOption')
79  if CLEAN:
80      env.Command(source=None, target=None, action='rm -rf '+BUILDDIR+'/*')
81  elif BUILDOPTION == None:
82      print('Specify a build option using --build')
83      print('Options: GameLoop, Uart, Camera, Projector, ColorPicker')
84      exit()
85  else:
86      env.Replace(CPPDEFINES = BUILDDICT[BUILDOPTION]['defs'])
87      env.Program(
88          target = BUILDDIR+'runme',
89          source = BUILDDICT[BUILDOPTION]['source']
90      )
```