User Manual for wEyes and pBrains Software

User Manual and Software By:

Kenneth Adam Barnett

ECOLYMER River Robot Plastic Cleanup

GA Tech Senior Culminating Design

wEyes Application for Windows

Requirements:

- Visual Studio Community (For Building/Debugging)
- Microsoft Kinect v2.0 SDK (For Kinect Drivers)
- Latest source files downloaded from: https://github.com/kbarnett256/RiverCleanupRobot

Building the Application:

Visual Studio Project files are included with the source files to assist with building the application. Open the provided project file with Visual Studio, then build, debug, or edit the code using the Visual Studio IDE. Alternatively, the latest Windows Binary for wEyes.exe is available as a release via the GitHub link, simply download the file and run (Kinect SDK is still required for the drivers.)

Starting the Application:

- *Without Networking:* To run the application with no networking, open the program normally. This allows the user to view the depth buffer and object detection results without sending any data to the coupled MOOS app, pBrains, which provides the collision avoidance behavior for the autonomous vehicle.
- *With Networking:* To run the application with networking enabled, the user must start the application via the command line. With the command prompt open, change the directory to the folder containing the wEyes.exe file, then use the command:

  ```
  wEyes.exe [Host Name or IP Address]
  ```

  The host name or IP address should be the address of the machine that is running the pBrains application, normally the vehicle or the machine hosting the vehicle simulation. All communications occur through port 9900. To close the application properly, use the following command, otherwise the program will remain running in the background:

  ```
  taskkill /F /im wEyes.exe
  ```

Using the Application:

- When the application starts it will remain unresponsive for about 30 seconds while the software determines a baseline for its object detection scheme. During this time, the Kinect device should be pointing to an operating area that is clear or obstacles because this step is needed to capture changes due to visual noise and movements of the vehicle caused by the water it is deployed in. If the window remains unresponsive for longer than 30 seconds, ensure the Kinect is properly connected and relaunch the program.

- After the baseline is determined, the application's user interface should appear inside the window:

1. **Depth Buffer:** Colorized visualization of the depth buffer from the Kinect.
2. **Anomaly Indicators:** The top bar reports the current column average via the intensity of the red-scale bar, darker values indicate a lower average value for that column. The bottom bar indicates whether the column average is within the threshold of the gathered baseline and is red whenever an anomaly is detected within the column.
3. **Screenshot Button:** Saves a picture of the current depth buffer to the user's default Pictures folder.

- When the autonomous vehicle is deployed, the application can be started via SSH connection to the host device. If the application begins to erroneously detect obstacles while deployed, it might be possible that the application needs to be recalibrated. To do so, simply use the command line arguments outlined in the "Starting the Application" to restart the program and recalibrate for the new operating area.

pBrains Application for Linux

Requirements:

- MOOS-ivp and MOOS-ivp-extend (For Building)
- Latest source files downloaded from: https://github.com/kbarnett256/RiverCleanupRobot

Building the Application:

Navigate to the directory `/moos-ivp-extend/src/` and place the pBrains folder containing the source files inside. Modify the CMakeLists.txt file inside the src directory (not the pBrains directory) to add the line highlighted in orange:

```
15 #============================================
16 # List the subdirectories to build...
17 #============================================
18 ADD_SUBDIRECTORY(lib_behaviors-test)
19 ADD_SUBDIRECTORY(pExampleApp)
20 ADD_SUBDIRECTORY(pXRelayTest)
21 ADD_SUBDIRECTORY(pBrains)
22
```

Navigate to the parent directory `/moos-ivp-extend/` and execute the build.sh script to build the MOOS app and output the results in the `/moos-ivp-extend/bin` directory. Finally, add the bin directory to your Linux Path variable so the system can locate the pBrains app by just using the file name as a command.

Starting the Application:

To use the application, a MOOS configuration block must be created to provide the app with details of the MOOS Database it is running on. A sample configuration block is provided with the source files in the folder “Sample Configuration Files” named plug_pBrains.moos. Please note that the AppTick value must be higher than the maximum frequency of the Kinect hardware, 30Hz. While in the folder containing the configuration block, use the following command to run the pBrains app by itself:

```
pBrains [Configuration Block Filename].moos
```

To start the pBrains app as part of a larger collection of MOOS Apps, such as with the simulator or vehicle apps, add the two lines highlighted in orange to the larger list apps and configuration block files in the meta vehicle configuration file:

```
 7 //------------------------------------------------------------
 8 Processconfig = ANTLER
 9 {
10   MSBetweenLaunches = 200
11
12   Run = MOOSDB              @ NewConsole = false
13   Run = uSimMarine          @ NewConsole = false
14   Run = pLogger             @ NewConsole = false
15   Run = pNodeReporter       @ NewConsole = false
16   Run = pMarinePID          @ NewConsole = false
17   Run = pHelmIvP            @ NewConsole = false
18   Run = pBasicContactMgr    @ NewConsole = false
19   Run = pHostInfo           @ NewConsole = true,XConfig=4
20   Run = uFldNodeBroker      @ NewConsole = false
21   Run = uFldMessageHandler  @ NewConsole = false
22   Run = pShare              @ NewConsole = false
23   Run = pBrains             @ NewConsole = true
24
25   4 = -geometry,80x10+20+950,-bg,purple,-fg,white
26 }
27
28 #include plug_pShare.moos
29 #include plug_uSimMarine.moos
30 #include plug_pLogger.moos
31 #include plug_pNodeReporter.moos
32 #include plug_pMarinePID.moos
33 #include plug_pHelmIvP.moos
34 #include plug_pBasicContactMgr.moos
35 #include plug_pHostInfo.moos
36 #include plug_uFldNodeBroker.moos
37 #include plug_uFldMessageHandler.moos
38 #include plug_pBrains.moos
```

Using the Application:

To use the application, the proper ConstantHeading and ConstantSpeed behaviors must be added to the behavior files for the vehicle. The correct configuration is included in the “Sample Configuration Files” folder as the meta_vehicle.bhv file. Please note that the variable names “AVOID” and “UPDATE_AVOIDH” are used by the pBrains app and cannot be changed without editing the source and rebuilding the files as well. The only requirements for adding the avoidance behavior are that both the ConstantHeading and ConstantSpeed behaviors are activated when the AVOID variable is “true”, and the ConstantHeading behavior receives updates through the variable UPDATE_AVOIDH. If the requirements are met, the pBrains app will begin reading the anomaly data sent from the wEyes app and set the craft to a new heading whenever an object is detected in the vehicle’s path.