

Supporting High-Complexity Coding

Amanda Shehu, Computer Science



CS 485: Autonomous Robotics

My main goal has been to give students assignments that do not feel like toy ones, but ones that indeed they can point to as accomplishments in their vitas when applying in industry or graduate school. However, such assignments demand time of students. To allow for complexity and yet feasibility, I now provide students with support code. Effectively, I implement the low-level behaviors, so that students can focus on implementing the high-level behaviors and see the concepts taught in class in action. The picture below shows how I provide snippets of the support code, highlighting where students are asked to add their implementations. This has allowed more complex assignments. Such assignments also give students a real and immediate purpose for their learning and focus their energies very effectively.

A screenshot of an Emacs editor window titled 'emacs@abetare'. The code is for a C++ class named 'BugAlgorithms'. It includes a header file 'BugAlgorithms.hpp'. The class has a constructor that initializes 'm_simulator' and sets various mode and hit/leave variables. It also has a destructor that prevents deletion of 'm_simulator'. Three methods are shown: 'Bug0', 'Bug1', and 'Bug2', each returning a 'Move' object. The code is color-coded, and red comments indicate where students should add their implementations.

```
#include "BugAlgorithms.hpp"

BugAlgorithms::BugAlgorithms(Simulator * const simulator)
{
    m_simulator = simulator;
    //add your initialization of other variables
    //that you might have declared

    m_mode = STRAIGHT;
    m_hit[0] = m_hit[1] = HUGE_VAL;
    m_leave[0] = m_leave[1] = 0;
    m_distLeaveToGoal = HUGE_VAL;
}

BugAlgorithms::~BugAlgorithms(void)
{
    //do not delete m_simulator
}

Move BugAlgorithms::Bug0(Sensor sensor)
{
    //add your implementation
    Move move = {0.01,0.01};

    return move;
}

Move BugAlgorithms::Bug1(Sensor sensor)
{
    //add your implementation
    Move move = {0,0};

    return move;
}

Move BugAlgorithms::Bug2(Sensor sensor)
{
    //add your implementation
    Move move = {0,0};

    return move;
}
```



These materials are presented under a Creative Commons *Attribution-NonCommercial-ShareAlike* license: you may use, adapt, and build on these ideas, for non-commercial purposes, as long as you credit the author.