Joel Clark

ECEN 3753 Real-Time Operating Systems

7 April 2022

Week 4 Report: Unit Testing Continued + Function Tests

Week 4 Tasks & Time Estimates:

* Rework unit testing plan with more specific tests and goals: 5 hours
* Code unit tests and functional tests: 5 hours
* Create a playable game using CapSense (minus the button functions): 25 hours

1. ­**Unit Testing**

This week’s work includes concrete unit testing along with functional testing as one can see in the test.c file in the code on GitHub.

1. **Test Plan and Results: 3 Cutting Points for Testing**

After hours of project clarification, I’ve decided to completely reassess my cutting points, now that I know more about what makes a good cutting point.

* 1. The first cutting point for testing would be the platform function. This function will receive data from the CapSense slider after being triggered by a mutex in the physics task. This function is responsible for calculating the force and speed­ of the platform shield and giving providing that data to the physics task which provides it to the display task.
  2. The second cutting point would be the HM physics function. This function takes in several input parameters, so testing is vital. These include but are not limited to: gravity, canyon size, initial conditions (x/y velocity, x position), user defined parameter, and display diameter.
  3. The third cutting point will be the HM platform collision function. This function will take data from the physics task and calculate whether the HM is set to collide into the platform shield or not and calculate how much force is needed to apply to the platform in order to do so. It will then take this necessary force value and convert into a PWM to power the LED.

Each of the cutting points will have at least four varying data sets to ensure the functionality of them. This will end up equaling at least 10 testing points.

Week 4’s additional testing includes the following functional tests:

* 1. Press either side of the CapSense slider and the platform on the screen should start to move in that direction; if anything but that happens, this test has failed.
  2. Pressing the inside sectors of the four sectors of the CapSense slider should apply less force to the platform than pressing the outside sectors of the slider. If this is not visually apparent, the test has failed.
  3. Pressing no sectors of the CapSense slider should not move the platform at all, assuming the platform is currently at rest.
  4. Releasing the slider should no longer apply force to the platform and the velocity should no longer change until a force is applied again.
  5. The platform should not bounce off of the walls of the canyon, even when force is continually applied in the maxed-out direction; if anything but this has happens the test has failed.
  6. The user needs to visually know when they’ve lost; display something on the LCD if the enemy goes past the platform.
  7. The LEDs should glow according which side of the CapSense slider is pressed; This way the user knows if there are false positives from the CapSense hardware.

1. **Statement of Project Current Standing**
   1. This week I was able to get a functional game working. The enemy now has X and Y direction physics, and the Gecko is able to detect CapSense touch and move the platform according to the CapSense data. Most of the project is now complete.
   2. What still needs to be done, functionality wise, the buttons need to be programmed to boost the enemy out of the playing area or call in laser air support.
   3. I have also coded the unit tests and the function tests to assess functionality of not only physics but user input.
   4. I’ve completed 100+% of this week’s scoped work and so far, am ahead of the total scoped progress.
      1. One caveat is that I need to code in the Micrium OS features and make the game utilize the OS features to optimize and protect data.
   5. There will be no necessary change to scope considering all tasks have been accomplished for this week.
2. **In-Scope Work Items**
   1. ***COMPLETE*** Complete flowchart: 1.5 hrs
   2. ***COMPLETE*** Configure GitHub desktop for project folder: 0.5 hrs
   3. ***COMPLETE*** Risk Register: 1.5 hrs
   4. ***COMPLETE*** Reassess unit test plan: 3.5 hrs
   5. ***COMPLETE*** Complete redo of task diagram: 4 hrs
   6. ***COMPLETE*** Layout code for unit test implementation: 2 hrs
   7. ***COMPLETE*** Layout structure for physics functions: 2 hrs
   8. Configure inputs (code): 1.5 hrs
      1. Buttons with interrupts
         1. Physics data mutex.
      2. Slider on an OS Timer
         1. Laser blast semaphore.
   9. Create tasks (code): 46 hrs
      1. Physics: 35 hrs
         1. ***COMPLETE*** Make gravity for y direction.
         2. ***COMPLETE*** X direction physics.
         3. ***COMPLETE*** Configure boost from button input.
         4. ***COMPLETE*** Make physics modular for custom inputs.
         5. Configure timer updates.
         6. Physics data mutex.
         7. Laser blast semaphore.
         8. If speed is too slow, enemy goes through platform shield.
         9. Make canyon and platform size modular and easily configurable.
      2. Display (code): 10 hrs
         1. Make custom UI
            1. Remaining enemies & laser blasts, boost cool-down.
         2. Configure timer updates.
         3. Laser blast semaphore.
         4. Make cool animations
      3. Platform(code): 12 hrs
         1. ***COMPLETE*** Calculate force from slider input
         2. Calculate trajectory somehow to light up LED to signal force needed from platform.
         3. ***COMPLETE*** Determine if enemy made it past the platform shield
      4. LED Out (code): 4 hrs
         1. Calculate PWM from force percentages to control brightness’s.