Unit testing:

Cutting point-After button task reads from the fifo:

1. Test that a fifo read from an empty fifo returns false.-Pass
2. Test that a fifo read from a fifo returns the correct value.-Pass
3. Test that the rail gun charge gets calculated correctly.-Pass

Cutting point-After physics task updates values periodically.

1. If capsense force is not equal to 0, check if the horizontal position of the platform is correct.-Pass
2. For satchels, check that the proper number of satchels are in the air at all times.-Pass
3. Check that the vertical position of the satchels was updated correctly.-Pass
4. If something collides with a wall, check that its velocity changed sign.-Pass
5. If the shield was activated during the physics update, check if any satchels in range were properly destroyed.-Pass
6. When a satchel reaches the ground, check that the satchel’s x-position is on target. Pass

Cutting point-after the display/LED task updates periodically

1. Check that the left LED turns on and off at the correct duty cycle.-Pass
2. After the castle evacuation time expires, check the left LED is constantly on.-Pass
3. Check that the Pulse width for the right LED is relatively equal to the current force magnitude.-Pass

Functional Tests:

1. Press the left/right side of the CAPSENSE slider and check if the platform moves to the left/right. Pass
2. The outer quarters should cause the right LED to stay constantly lit while the inner quarters should have the LED blink on and off evenly. -Pass
3. Pressing two sections of the CAPSENSE slider on opposite sides combines their effective force, potentially canceling each other out. -Pass
4. If BTN1 is released, a force field should appear on the display and destroy any satchels in range.
5. Pressing and holding BTN0 then releasing should fire a projectile with speed increasing as you hold the button, maxing out after 5 seconds. -Pass
6. If the generator does not have enough power for the shield or the railgun nothing should happen. -Pass
7. Hitting the Castle foundation (2 pix wide) should reduce the castles health by 1, while hitting the Castle wall (1 pix wide) should reduce the castles health by 2 - Pass
8. After the castle has less than half health (rounding down), the left LED should start blinking. -Pass
9. After depleting the castle’s health completely or after the evacuation sequence ends and the left LED is completely on, the game should take the user to a victory screen. -Pass
10. The platform hitting a satchel or railgun shot, a wall too fast, or running out of energy while the castle is not in an escape function should take the user to a game over screen. Pass

Project Summary:

This week, I implemented max\_inAir satchel throwing as an extension and finalized my game parameters for my demo

Summary Effort/Estimate:

I have 100% of my current work (45 estimated hours out of 45 total) in 84.4% of the budgeted time (38 hours spent out of 45 hour estimate) For the work that has been completed, I took 0.844x (38 actual hours/45 estimated hours) as much time as I estimated.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task | Estimated Effort | %Estimate | Actual Effort | Status |
| Task Diagram | 3 hours | 6.66 | 2.75 hours | Complete |
| Unit Testing | 8 hours | 17.77 | 4 hours | Complete |
| Platform Phys | 5 hours | 11.11 | 5 hours | Complete |
| Satchel Phys | 5 hours | 11.11 | 4.75 hours | Complete |
| Railgun Phys | 5 hours | 11.11 | 6 hours | Complete |
| Display Task | 12 hours | 26.66 | 12 hours | Complete |
| Button Task | 5 hours | 11.11 | 2.5 hours | Complete |
| CAPSENSE Task | 2 hours | 4.44 | 1 hour | Complete |
| Total Complete | 45 hours | 100 | 38 hours | 45 hours budgeted |

Reflection:

* Task priority
  + The highest priority task for my code is the force task for the Capsense slider, since if it gets interrupted for too long it starts to function incorrectly. The next highest priority task is my physics task, since the physics task has the most important deadline for my code. The display task has the lowest priority out of my periodic tasks, since it has the longest execution time and its deadlines are slightly less strict. My button task shares the same priority as the display task, but is also only ran whenever a button is pressed and remains blocked otherwise, so having a lower priority is not too much of an issue.

A screenshot of a computer

Description automatically generated with medium confidence

* Code Space
  + Overall, my main app.c is about 1200 lines long. I think some of my code definitely could be optimized a bit better to make this shorter, or have certain parts of my code be broken down into separate files and functions to make the code as a whole more readable.
* Evaluation of physics update
  + The first thing my physics update does is check if button events happened and waits for 1 ms before moving on. If a railgun event was called and there is currently no railgun shot fired, a new one is initialized. If a shield event was called, the shield also gets activated. The game updates the platform first. Since the platform is the main physics item that is influenced by the user, having it update before any other items allows for fairer collision checks. For all physics items, the edge case order typically goes checking for wall/castle hits first, since those are relatively simple and don’t require checking for any specifics in most cases, then checking for ground collisions since it also requires checking if the item hit the platform. After the platform is updated the railgun shot is updated, followed by the satchels. The main limitation I had to deal with was the pending time needed for the OSFlagPend to get the button events to work. Initially I wanted to have this be a non-blocking call, but I was unable to get this to work, so I just reduced my task delay by one tick to adjust for the time spend blocking at the start of the task.
* Scaling of variables
  + I think my game was playable with just about any settings given to it. The main limiter on my games scale was the canyon size variable, since a lot of other variables like the castles height and max bounce speed of the platform were dependent on how large the canyon walls and floor were relative to it. I found the generator power variable to be unnecessary since its use was never really explained well in the document and I never really found a good place to use it while coding my project.
* Future steps
  + If I were to continue working on this project, I would add a menu setup to my project so the game be more customizable while playing rather than having to reupload new values each time and continue to work on how the multi-satchel mode functions since it can occasionally cause issues with the game.