sdmay19-35: Implementing a Web Portal System for Drone Simulation and Control

Week 4 Report

October 6 – October 12

Client: Ali Jannesari

Faculty Advisor: Ali Jannesari

Team Members

Bansho — *Test Engineer. Back-end Developer.*

Ian — *Scrum Master. Front-end Developer.*

Li — *Test Engineer. Back-end Developer.*

Jawad — *Meeting Manager. Back-end Developer.*

Mehul — *Project Lead. Back-end Developer.*

Sammy — *Report Manager. Front-end Developer.*

Summary of Progress this Report

- Fix bugs with camera when attempting to change the view after the drone has taken flight.
    - In an attempt to provide the most effective behavior with minimal bugs, the camera no longer moves with the drone.
    - The camera now must be fully controlled by the user using Three.js's Orbit Controls. (Mouse and keyboard controls to move the camera around).
- Improved some aspects of movement
    - Cannon.js provides a function called applyLocalForce that allows us to apply a certain force in a direction relative to the object we are targeting, rather than the world.
    - Implemented applyLocalForce in our movement routine and reduced the amount of force being applied.
    - Less force being applied prevents the drone from moving sporadically when moving horizontally while resting on a flat surface.
- Added a separate window to display the drone's camera view
    - Gave the drone a Three.js Perspective camera and positioned it beneath and slightly in front of the drone with a minor downward rotation.
    - Added a new Three.js Renderer to the scene to render the drone's perspective.
    - Positioned the drone's view canvas using CSS. Absolutely positioned in the top left with a fixed width and height of 200 pixels.
- Improved drone model
    - Found a nice model here. However, the file is huge and cannot be loaded quick enough (roughly 150 mb).
    - Loaded model into Blender and decimated the geometry to reduce the size down to approximately 11mb (could probably be reduced further).

    - Placed into the scene and created 2 Cannon physics bodies to be associated with it. The drone body is oddly shaped, so a single box would not result in believable physics since collisions would seemingly occur for no reason.
- Added individually controllable propellers to the drone
    - The drone new model had propellers, but there would be no way to make them rotate.
    - Separated them from the model and saved 1 copy of the propeller to be loaded as 4 individually controllable objects.
    - Loaded the propeller object file, made 4 copies, and positioned them in their proper locations on the drone.
    - Created Cannon physics bodies for each propeller. Doing so gives us the ability to apply cannon functions to each propeller, making the flight more believable. In addition, the drone can no longer flip over so easily.

Pending Issues

- Drone movement is not very believable and should be worked on more. It might be best to have each propeller as its own object with its own physics.
- Still need to get machines set up in order to use ROS, which is necessary to integrate with our simulator and is preventing the backend from accomplishing much.

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Bansho	Continued research on ROS and how to integrate with web applications.	6	25
Ian	Improved movement of drone by utilizing the Cannon.js function mentioned above.	7	29
Jawad	Continued research on ROS and how to integrate with web applications.	6	27
Li	Continued research on ROS and how to integrate with web applications.	6	25
Mehul	Continued research on ROS and how to integrate with web applications.	7	27
Sammy	Found, modified, and loaded new drone model and propellers. Fixed camera. Added drone view.	8	31

Plans for Upcoming Reporting Period

- Frontend
    - Add a control panel that will allow the user to enter in a command (like up(10) left(4), etc) and interact with the drone accordingly.
    - Add an on-screen joystick that allows the user to move the drone.
    - Improve the movement and tweak the physics to be more believable.
    - Make each propeller rotate and adjust their speed individually depending on the movement.
- Backend
    - Set up ROS and start interfacing with the front-end.
    - Set up Raspberry PI and camera and start researching computer vision techniques.
    - Establish communication between ROS and the Raspberry PI to mock the communication with the drone.
    - Begin designing and setting up server.