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

Week 3 Report

September 29 – October 5

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

- Decided on Cannon.js for the physics library because it is easy to use and the features it provides, such as: collision detection, gravity, inertia, and momentum.
    - Drone will react to collisions in mid-air as well as being struck while stationary.
    - Movement is affected by gravity, momentum, and inertia to reflect reality.
- Implemented a simple Three.js environment
    - Loaded OBJ and MTL files for a rudimentary drone using THREE.MTLLoader and THREE.OBJLoader
    - Added a THREE.Plane to represent our floors.
    - Added a THREE.AmbientLight for global lighting throughout the world.
    - Implement a THREE.OrbitControls for better viewing abilities.
    - Create an EventListener to handle key events to move the drone object up, down, left, and right.
- Integrated Cannon.js with our Three.js environment
    - Set up a CANNON.World and initialized the gravity to -2.4 m/s in the y direction to simulate real world gravity.
    - Added a CANNON.Body to sync up with the drone object and provide the physics effects. All physics are applied to the CANNON.Body object and every time the scene is rendered the position and rotation of the THREE.js drone object is synchronized with the CANNON.Body object to visually display the physics occurring.
    - Improved movement by applying velocity in the appropriate coordinate direction with respect to the desired movement action. For example, moving left applies a negative velocity to the x direction.
    - Implement rotation by applying angular velocity to rotate about the y axis.

- Added Jest to our project to begin Test Driven Development
    - Created tests for the drone component to test:
        - The Drone component successfully instantiates
        - The Drone's y position increases upon calling the ascend function.
        - The Drone's y position decreases upon calling the descend function.
        - The Drone's x position decreases upon calling strafeLeft.
        - The Drone's x position increases upon calling strafeRight.
    - Created tests for the scene component to test:
        - The component successfully renders. This required adding Enzyme in order to actually render the component.
    - Created tests for the app component to test:
        - The app renders without crashing using the ReactDOM.render function.
- A demonstration of the progress can be found here: http://sdmay19-35.sd.ece.iastate.edu/demos/prototype1/index.html
    - Click on 'Simulate'.
    - Move up and down with r and f keys.
    - Move left, right, forward, and back with a, d, w, and s respectively.
    - Rotate left and right with q and e.

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.
- The camera cannot move around when the drone start moving. It actually breaks when attempting to move it.
- If attempting to move horizontally while stationary on the ground, the drone will spin out of control.
- The model being used is very unappealing and should be improved.
- 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.
- Our group structure must be revisited as the recent changes in plans shift the project to a predominantly front-end oriented task.

Individual Contributions

Team Member	Contribution	Weekly Hours	Total Hours
Bansho	Researched ROS and how to integrate with web applications.	5	19
Ian	Implemented forward and backward movement and improved controls. Added the ability to rotate the drone.	8	22
Jawad	Researched ROS technologies and found other web applications that integrate with ROS.	6	21
Li	Researched ROS and how to integrate with web applications.	6	19
Mehul	Researched ROS technologies and found other web applications that integrate with ROS.	5	20
Sammy	Set up and populated the scene with THREE.js objects. Added Cannon.js and synchronized the THREE.js objects with the CANNON objects.	8	23

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.
    - Update the drone model so it looks more appealing to the user. Currently it is using a random free model but would be best to create our own.
    - Fix the camera controls so the user can look around without the camera breaking.
    - Add an additional Three.js camera that will only display the drones view of the world.
- Backend
    - Design and implement a better environment to be used in the simulation. The initial environments should be: forest and urban areas.
    - Set up ROS and start interfacing with the front-end.
    - Research and potentially implement ROS3DJS, a 3D modeling library built upon Three.js that is designed with the intention of integrating with ROS.