A HUMAN-ROBOT INTERACTION JAVASCRIPT LIBRARY: ROS WEB COMPONENTS

INTRODUCTION

The problem of developing GUIs (Graphical User Interfaces) for robots is one that typically requires knowledge of robotics programming and web or desktop GUI development to communicate the user's inputs to a robot, and to communicate a robot's state and outputs to the user. This project introduces a FOSS (Free and Open-Source Software) JS (JavaScript) library: 'RWC' (ROS Web Components) (Elliott, 2019a), which utilises the also FOSS 'ROS' (Robot Operating System) (Quigley et al., 2019). RWC simplifies the development of web-based GUIs for HRI (Human-Robot Interaction), by exposing common robot behaviours and data through one-line JS function calls, and defining custom HTML (Hypertext Markup Language) elements which can call these functions to display data or send user input to a robot. RWC requires only beginner level knowledge of HTML or JS, and ROS, to enable rapid and easy development of web-based UIs for HRI. RWC is configurable, with more experienced users able to define their own ROS topics and action servers for the library to interface.

AIMS AND OBJECTIVES

This research aimed to create a JS library to simplify the creation of robust GUIs for HRI, providing reusable components which interface with the robot to give a visual representation of its current state, send commands, and receive and display data. Though ROS runs on desktop OSs, RWC UIs should run remotely on PCs, phones, and tablets, with the state of the interface available to a connected robot.

METHODOLOGY

The RWC library abstracts roslibjs (Toris et al., 2019), the standard ROS JS library. roslibjs connects to ROS via rosbridge (Mace, 2019), which provides a JSON and WebSocket interface to ROS. RWC's functions were divided into two categories: actions, and listeners. Actions are preemptable actionlib (Marder-Eppstein et al., 2019) goal commands with callbacks for their completion. Listeners return robot data from a ROS topic (Quigley et al., 2019, 3). The GUI of the tour-guide robot Lindsey was redeveloped using RWC, lines and bytes of code were counted, for this new interface and the original, and comparative statistics were calculated.

SCAN ME

Lindsey at The Collection museum
Credit: University of Lincoln

USE CASES

- Redeveloping the touchscreen web interface of tour-guide robot Lindsey at Lincoln's The Collection museum. Expanding on its features and robustness.
- A backend for a web-based visual robot programming tool by Onis Brown, for public engagement with robot programming.

The Web-based Interface of Lindsey at the Collection Museum
Credit: University of Lincoln

RESULTS & CONCLUSION

Comparisons of Lindsey's UI redeveloped with RWC and the original UI illustrated a reduced amount of work required from the developer, and a smaller codebase, hence more maintainable code: ~3 times less lines and bytes of code were written for the redeveloped UI, than for the original UI. When pre-built code, e.g. libraries, is taken into account, the RWC UI is comprised of ~11 times less lines of code, and ~9 times less bytes of code. See the '.ods' spreadsheets in the RWC UI's repository for the full results (Elliott, 2019b).

REFELECTIVE ANALYSIS

The case study of Lindsey's UI redevelopment provides evidence of RWC's effectiveness at reducing the workload of developing a HRI UI, but it would be safer to generalise from a usability study involving PPs (Participants) with varying levels of robotics and web programming experience. PPs could receive basic instruction on programming with roslibjs or RWC, and be asked to complete a simple HRI UI programming task. Afterwards the difficulty of this task for each PP could be recorded via interview or questionnaire. With RWC exposing robot functions to a web-client, issues of security may arise which are not addressed in this project, though integration of a more secure version of rosbridge, with topic and IP whitelisting could combat this.

REFERENCES

Elliott, L. (2019a) *roswebcomponents* [software]. Lincoln: University of Lincoln. Available from github.com/laurencejbelliott/roswebcomponents [accessed 5 September 2019].

Elliott, L. (2019b) *lindimp_ui_rwc* [software]. Lincoln: University of Lincoln. Available from *github.com/laurencejbelliott/lindimp_ui_rwc* [accessed 5 September 2019].

Mace, J. (2019) *rosbridge_suite* [software]. Robot Web Tools. Available from *github.com/RobotWebTools/rosbridge_suite* [accessed 5 September 2019].

Marder-Eppstein, E., Pradeep, V. and Arguedas, M. (2019) *actionlib* [software]. ROS core stacks. Available from *github.com/ros/actionlib* [accessed 5 September 2019].

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R. and Ng, A. (2019) ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, 3(3.2) 6.

Toris, R., Lee, J., Alexander, B., Gossow, D. and Pitzer, B. (2019) *roslibjs* [software]. Robot Web Tools. Available from *github.com/RobotWebTools/roslibjs* [accessed 5 September 2019].

Student: Laurence Elliott

Supervisors: Marc Hanheide & Francesco Del Duchetto

UNIVERSITY OF LINCOLN
Lincoln Academy of Learning and Teaching

Undergraduate Research Opportunities Scheme

DISCOVER_
LNCN.EU/UROS
@U0L_LALT
#UROS2019