# Train Module Test Plan Results

Shane Lester

### Communication with Train Model

Inputs-

* Command originating from Train Controller
* Value used in command (power)

Outputs-

* Log window displaying a command being sent
* Train info panel reflects the new command

For this test, a command regarding the power of the active train will be sent to the train model. This will test successful integration of the two modules. The command will originate from the train controller GUI, using the Give Power button in conjunction with the Power (W) input field. If the test is successful, the power displayed in the train info will be the same, and the speed will change as well as the distance traveled.

***Actual Result: Interactive-*** Train Controller successfully sends power to the train model

***Tester:*** James Vento

***Date:*** 12/9/2012

***Notes:*** The power is successfully passed to the active train, and the info is reflected in the GUI. The train begins accelerating appropriately, while obeying present speed limits.

***Acceptance:*** Acceptable

### Correct Distance Calculation

Inputs-

* Power
* Acceleration and Deceleration limits

Outputs-

* Distance Moved

This will test whether the GUI will correctly calculate and display distance for the active train. It will also test whether the updating of the GUI is appropriate handled.

***Actual Result: Interactive-*** The GUI correctly calculates the distance, and it updates in a timely manner.

***Tester:*** Shane Lester

***Date:*** 12/10/2012

***Notes:*** It originally updated too fast, giving the appearance that the train was moving at impossible speed.

***Acceptance:*** Acceptable

### Handle switching between active trains

Inputs-

* Selected train in the GUI dropdown

Outputs-

* Updated train info in the GUI

This will test if the GUI is able to switch between different trains and display the appropriate information.

***Actual Result: Interactive-*** Originally, the GUI was not showing the correct information, and instead showed shared information from each train. It was fixed to work as intended.

***Tester:*** James Vento

***Date:*** 12/10/2012

***Notes:*** The information was showing skewed information, and power information from one train would bleed into the other. After the test was ran, changes were made to the code to make less of the variables static, and it allowed the context switch to occur seamlessly.

***Acceptance:*** Acceptable, after defect correction

**James Vento**

# **Train Controller Installation guide**

The train controller module, as with the entire Albion package, requires recent versions of Java Virtual Machine and Java Runtime Environment, both of which can be downloaded from Oracle. The train controller can be ran independently by compiling and running TrainController.java, but for real functionality it must be used in conjunction with the other Albion modules. All source files must be compiled together. Running these commands from the command line within the directory of your source files will run the train controller:

javac \*.java

java TrainController

# **Train Controller User Manual**

The Albion Train Control System can be ran without directly accessing the train controller, but the train controller can also be used independently to model train movement. Upon running the module, you will see an interface split into two halves. The right half is a log containing system messages regarding commands being issued, such as when a train is created, when power is given, etc. The left half contains actual command buttons, and train-specific real-time information. There is also a menu bar at the top of the window, with two options. The File menu contains the “Add Train” operation, as well as the Exit button. The Help menu contains About information. Clicking on Add Train will bring up the Add Train menu, which contains radio buttons to select either the Red or Green line, as well as input fields for the Train ID and the number of cars. Be sure not to create two trains with the same ID numbers, as these must be unique across both train lines (i.e. G2 and R2 will cause an error).

After adding a train, there will be now be selection options in the dropdown menu located underneath the menu bar. This menu will contain all of the present trains, and selecting one of them will display the associated train information in the fields underneath. This information will update in real-time as the train moves. Underneath the information, there are 4 buttons and an input field. Give Power is used to give a specific power to the train. It must be used in conjunction with the text field directly to the right of it, where you input a value in Watts. To the right of that input is a Stop Power button, which will stop giving power to the train and allow the train to “coast” at the current speed. Next to that is the Emergency Brake button, which will stop the train as soon as possible. Next to that is the Call CTC Office button, which will call the CTC office.

You can switch between trains using the dropdown menu while they are running, and the info will reflect the active train. Any commands issued through the GUI will be sent to the active train, and will not affect any trains that are in the background.

# **Train Module Configuration Management**

For future versions of the train controller, it must retain its current functionality with all of the other modules. Improvements could be made regarding navigation, and pulling information from the track model. It should be possible to add these functionalities without changing any of the other modules, but some changes may require new functionality in the track controller. If that is the case, these functionalities must be documented and added as a joint update.