1 Introduction

As discussed in my proposal, I have implemented text inputs, a data output interface, and a game deployment interface.

2 Text Inputs

2.1 Designer

Text inputs were fairly straightforward. The only real design decision was in how to show the widget’s meta-information to the user (e.g., its name). The design that seemed to be the best fit was the one in which the widget’s meta-information appears when the widget is selected (see fig. 1), and hidden otherwise (fig 2).

Since text inputs are variables in their own right, users may rename them, just as they can rename screens. See fig. 3 for an illustration.

And, as text inputs are variables, they must appear in the Data Builder accordingly; see fig. 4.

2.2 GameEngine

Supporting text inputs in the GameEngine was relatively straightforward. When a button on a screen is clicked, triggering the request to fetch the next screen, the data the user entered into all text inputs on the screen is collected and sent with the request. The system then logs the results of the text inputs identically to any other variable. See figures 5, 6 for a demonstration.

Validation is handled by a custom Flex validator attached to each text input on screen creation. The validator continually checks its input, in order to give the user immediate feedback. If any text input on the screen is not in a valid state, none of the buttons on the screen are enabled.
Figure 1: Text input selected

Figure 2: Text input deselected
Figure 3: Renaming a text input

Figure 4: Text input results treated as variables
Figure 5: Text input in a game

Figure 6: Text input data entered, received, and displayed to the other player
3 Deploy

The Deploy interface involved several challenges. We decided to implement a form of versioning; whenever a game is deployed to Mechanical Turk, a snapshot of the game is saved, along with a nickname for this “run”, the number of HITs to post, and the amount the designer would like to pay the workers per HIT. This allows designers to tweak experimental parameters (game variables). See fig. 8.

4 Data Output

The Data Output interface exists to give experimenters the opportunity to view the data from a particular run of their game. Essentially, it shows them a version of the backend’s log. Of course, it translates internal variable names into human-readable forms, and parses it into a simple CSV format. This put a much friendlier interface between the designers and the SQL backend.
Figure 8: Deploy interface

Figure 9: Data output interface