

PIGEON 2.04
A Platform for Interactive Graphical
Experiments

by Nik Swoboda and James King

December 2002

Copyright © 2002 ATR and Queen Mary University of London

Contents

1	Overview and Introduction	5
1.1	The Software Components	5
1.1.1	The Whiteboard	6
1.1.2	Master Program	6
1.1.3	The Data Analysis and Presentation Tools	7
1.2	Controlling Subject Interaction	8
2	Setting Up an Experiment	9
2.1	Terminology	9
2.2	Installing Pigeon	10
2.2.1	Requirements to run Pigeon	10
2.2.2	Downloading and Unpacking	10
2.3	Preparing a new experiment	11
2.3.1	Creating a master configuration file	11
2.3.2	Setting up experimental machines	12
2.3.3	Running an experiment	12
2.3.4	Experiment progress	17
2.3.5	Analysis of results	17
3	Running an Experiment	19
3.1	Firing things up	19
3.2	Running the task	20
3.3	Cleaning up after a run	20
4	Available tasks	21
4.1	The generic task	21
4.1.1	Task Overview	21
4.1.2	Setting up the task	22

4.2	The music task	22
4.2.1	Task items	23
4.2.2	Task item selection	23
4.2.3	Setting up the task	24
4.3	The Pictionary task	25
4.3.1	Configuration	26
4.3.2	How the task works	27
5	User Configurable Options	31
6	The Playback/Markup Tool	41
6.1	Overview	41
6.2	Loading up the tool	41
6.3	Overview of the Pigeon Control Panel	42
6.4	Loading a Log	43
6.5	Controlling Real-time Playback	43
6.6	Jumping around the Log	44
6.7	Controlling who's drawing you see	45
6.8	Zooming the time-line	45
6.9	A closer look at the time line	46
6.10	Marks	47
6.11	Swapping Time-lines	48
7	Data Analysis Tools	49
7.1	Playback and Markup Tools	49
7.2	Log File Format	50
7.3	Drawing Statistics	52
7.3.1	Complexity measure	55
7.3.2	Screen Usage Statistics	56
7.3.3	Screen Usage Graphs	56
7.4	Converting Logs to Pictures	57
7.5	The task DATA file	57

Chapter 1

Overview and Introduction

This manual describes a tool for performing and analyzing experiments on graphical interaction between people. This software exploits the potential that electronic whiteboards offer for the direct capture and control of drawing activity. Currently, studies of graphical interaction have relied on direct observation, video analysis, or analysis of photographs and screen shots. These data collection techniques are labor intensive and susceptible to error. This tool was designed with three basic aims: to provide a fine-grained and reliable method of capturing drawing activities, to support experimental investigation through the manipulation of aspects of the graphical interaction, and to aid data processing by providing specialized tools for analyzing recorded graphical interactions. This tool has shown its usefulness in experiments conducted in Japan and the UK [1, 2].

1.1 The Software Components

The program consists of three basic components. Firstly, a simple shared whiteboard that allows pairs of subjects to communicate through drawing. The whiteboard runs across a network allowing users to be either co-present or in different locations. Users draw in freehand on the whiteboard selecting from a palette of colors in the editor bar. They see all their, and their partner's, drawing displayed in the whiteboard window. Secondly, a master program is used to configure, connect and monitor the individual whiteboard processes. These processes run separately on each subject's computer and record time stamped logs of both users' drawing, mouse or pen movements,

and button presses. Lastly, a suite of data analysis and presentation tools that is used to study the logs of data generated is provided.

1.1.1 The Whiteboard

The whiteboard window consists of three areas: the task bar, the editor bar, and the editing space. The task bar displays the buttons or other GUI components used by subjects in completing the given task e.g., buttons for response selection. The editor bar displays buttons to allow users to change color or to enter erase mode. The majority of the window consists of the editing space where the users can draw. A screen shot of the whiteboard window can be seen in Fig. 1.1.

Central to the whiteboard is the concept of a *subject event*. Subject events consist of particular subject interactions with the task bar or the editor. An example of an event is a line drawn by one of the subjects which has a starting and ending time stamp, a list of points, and a color. Subject events are sent between the pair of whiteboards and between the editor and the code that controls the experimental task being performed. Subject events are also written to a log file for subsequent data analysis. The logs contain a great deal of information including: each line or erase drawn as a time stamped list of points, the time of each button press in the task bar, and the time elapsed to complete the task. The logs also contain special task dependent information such as the result of the task and all experimental configuration parameters.

1.1.2 Master Program

Monitoring and control of the whiteboard processes is provided by the master program. The master reads a configuration file describing the task to perform. This configuration file is a human readable text file containing parameters such as the number of experimental trials and subject pairings. Once a configuration file is loaded, the master displays the current options and checks the status of subject's whiteboards. The master program is used to start, monitor, and repeat experimental trials. This program has been used in experimental settings to control and manage the connections of fourteen pairs of subjects simultaneously.

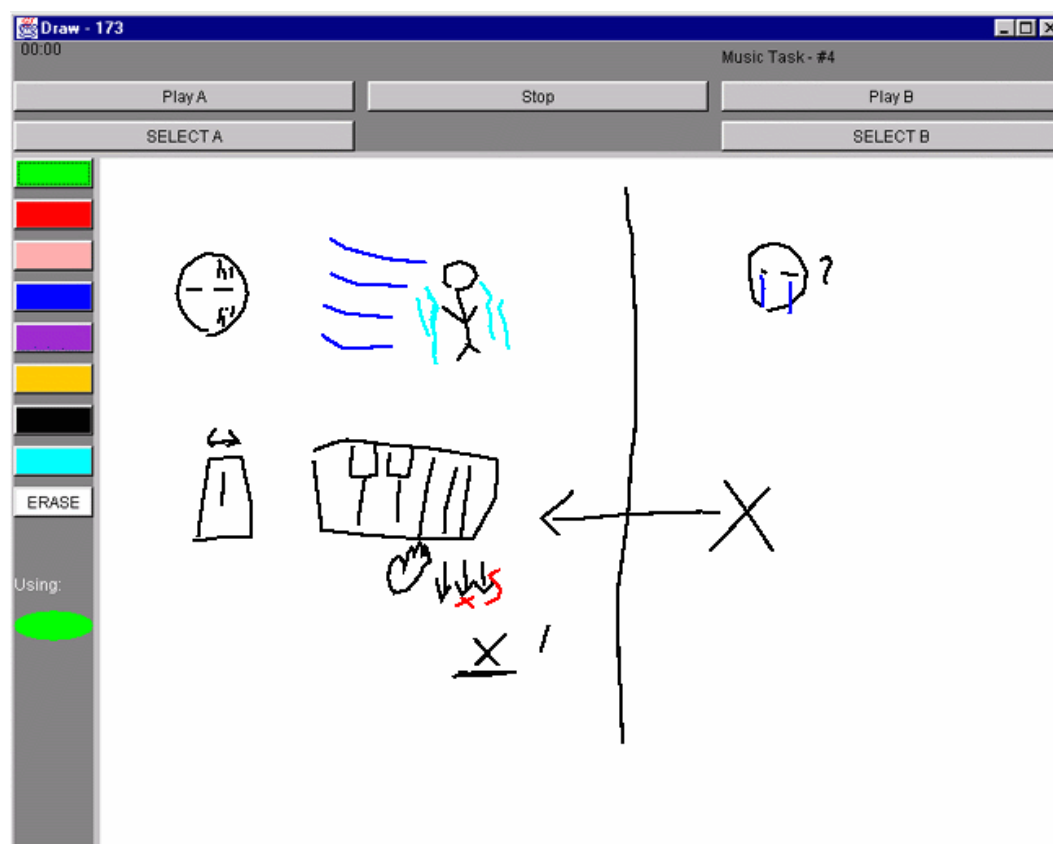


Figure 1.1: Example Whiteboard Screen

1.1.3 The Data Analysis and Presentation Tools

The comprehensive nature of the data logged by the whiteboard means that accurate, down to pixel level, recreations of the drawings at any point in the experiment can be produced in the form of gif, jpeg, PDF or postscript images. It can also provide separate images of the activities of each person individually and each pair.

The data can also be used to generate a variety of statistics for each phase of an experiment for individuals and pairs. Individual statistics include: the total number of lines drawn and erases made, the total line and erase lengths, the total time spent drawing and erasing, the number of different colors used, and screen usage. Per pair statistics include: the number of colors used by a

subject not used by their partner, and the amount of overlap in the drawing of the pair.

The data can also be used with a VCR-like playback tool. This tool allows the re-play, fast-forward, rewind, and stop of the drawing process at any point. It includes a time-line of the drawing activity of a pair which provides a visualization of patterns of, e.g., turn-taking and overlap in a graphical exchange.

1.2 Controlling Subject Interaction

This tool is designed to support the experimental investigation of graphical interaction. Thus, a significant part of its functionality has been developed to support the manipulation of the level of communicative interaction possible between experimental subjects. Currently, this is implemented in two main ways. The first is the manipulation of the spatial organization of the interaction, e.g, the division of the screen into separate drawing regions in which drawing can be selectively prevented. The second is the topological manipulation of the drawing in the form or transposition of regions of drawing from one part of a subject's screen onto another part of their partner's drawing space and the rotation of a subject's drawing before displaying it on their partner's drawing space.

Chapter 2

Setting Up an Experiment

This chapter contains information and guidelines for setting up and conducting an experiment using the Pigeon whiteboard tools.

2.1 Terminology

- Master machine - A single (not necessarily fast) machine that is used to control Pigeon experiments. The master application (which will run on the master machine) is used to start, control, and monitor the experiment.
- Subject machines - All of the provided tasks are pair based, so this is typically some even number of computers, one for each subject that will perform the task simultaneously. For many experiments this consists of a single pair, but larger group experiments can be run as well. For each pair running a task one of the machines of the pair will serve as the task controller and the other as the slave. This difference is completely transparent to the subjects.
 - Task Controller - Currently the whiteboard only allows pairs of people to work together. The task controller is one of the pair of machines that decides certain things like what music should be played (if a music task is chosen) and records the outcome of the task. The task summary file will be written to the disk of this machine at the end of each round.

- Slave - This is the other half of the pair of machines. It is subordinate to the task controller.

2.2 Installing Pigeon

Pigeon can be downloaded from the MAGIC research project homepage:

```
http://www.dcs.qmul.ac.uk/research/imc/magic.html
```

2.2.1 Requirements to run Pigeon

Pigeon requires that Java 1.1 or later is installed on all machines being used in association with the experiment. We have found that it is best if all the machines have the *same* version of Java installed on them.

2.2.2 Downloading and Unpacking

From the Pigeon web page you will find information on how to download an archive of all of the java programs and class files. The archive is available in two different forms, as a zipped archive and as a gzipped tar file. Various commercial and free utilities are available to uncompress these archives.

- Uncompressing a zip file. To uncompress the zip file you run:

```
unzip pigeon-2.04.zip
```

from the command line. If you are using PkUnzip or another unzip utility the program name might change. If successful this will produce a `pigeon-2.04` directory containing all the source and class files.

- Uncompressing a Tar.gz file

```
gunzip pigeon-2.04.tar.gz
```

This will produce a `pigeon-2.04.tar`, which is uncompressed with:

```
tar xvf pigeon-2.04.tar
```

This will then produce a `pigeon-2.04` directory containing all the source and class files.

2.3 Preparing a new experiment

The running of experiments with Pigeon can be broken down into several phases:

1. Creating a master configuration file
2. Setting up the subject machines that will run Pigeon
3. Conducting the experiment
4. Analyzing the results

2.3.1 Creating a master configuration file

The general setup of the experiment is controlled by a configuration file that is loaded by the master computer. In the top pigeon directory you will find a directory called `RUNS` containing various sample configuration files. The format of the files are fixed but the values can be changed. Only the master computer needs to have a local copy of the configuration file. When the experiment starts the master sends all the configuration parameters to the master and slave pairs that actually perform the experiment.

Below you will find a sample master configuration file.

```

2                // number of subjects required
#
testing-tmp     // file name that will be used
#
<Options key value pairs, see the Options chapter for more information>
#
20              // items used in this round
1              // number of groups in this round
2              // number in each group
1 2 1          // numbers of each of the partners and their group number

```

The first option in the file is the total number of subjects that will take part in the experiment. Two machines are one pair communicating together. 4 are two pairs etc.

Next is the name of the log file that will be used on all the machines to store the drawing information into. This can also be changed using the masters window before the experiment starts.

Then any number of options can be given. Some important options include the name of the task that will be used, and the duration of each item. More details on the available options and their allowed settings can be found in Chapter 5.

Following this is one section for each round of the experiment. First the number of items that will be used in that round is given, followed by the number of sub-groups and members of each sub-group. Lastly the pairings are listed along with their group number (which is given last). In the example above there is only one round with two subjects both in the number 1 group. Examples of configuration files for larger group experiments can be found in the `RUNS` directory.

2.3.2 Setting up experimental machines

The Pigeon class files need to be copied to all the machines that will take part in the experiment.

As mentioned earlier, Pigeon works best if all the machines run the same revision of Java. We have seen problems with one machine running Java 1.3 and another running 1.4. However, all the machines do not have to run the same operating system or have the same hardware configuration.

2.3.3 Running an experiment

The master and slave pairs get configuration information from the master machine. Therefore the master **MUST** be started before the pairs. To start the master application from the command line of the master machine enter the following:

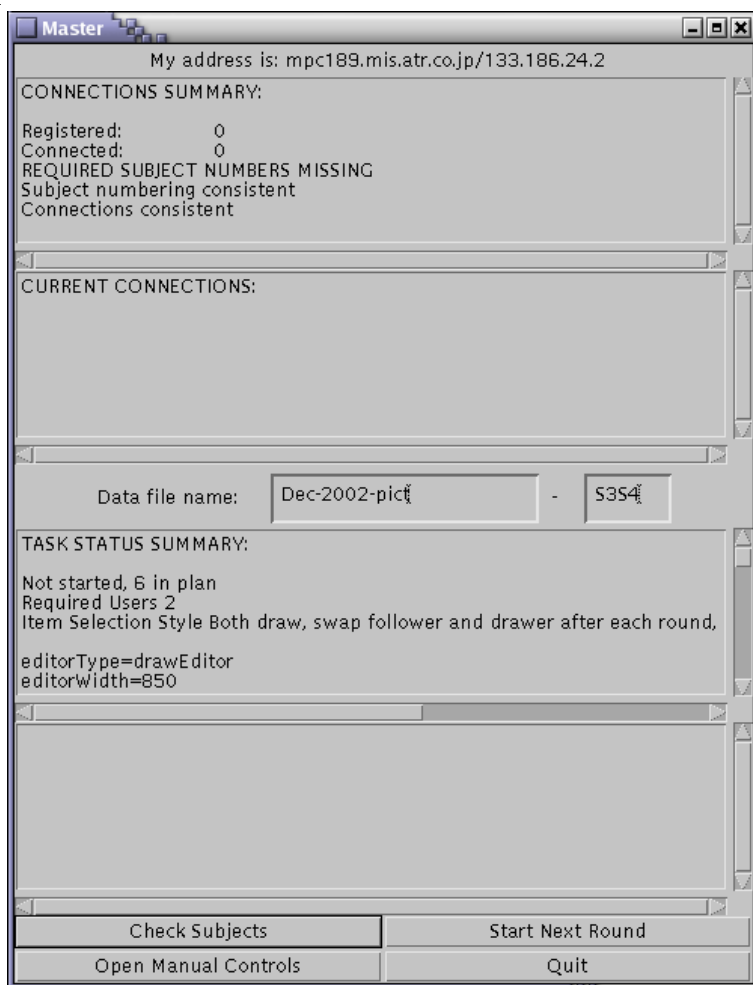
```
cd <root of Pigeon installation>  
java master.Master
```

Note: Case is important; the command is not `java master.master`.

The master should then display a file dialog box asking you to select an experimental configuration file. Select the experimental file you require and then Pigeon should then display the master application window. The configuration file can also be specified from the command line:

```
java master.Master -plan RUNS/test.dat
```

This will load the `RUNS/test.dat` file and begin the master application without opening a file dialog. Below you will find an image of the master application window.



At the top of the master window you will find the host-name and IP address of the machine running the master application. (This information can be useful when setting up the subject machines, as the the host-name or IP address of the master machine will need to be specified.) In the first text area you will see a brief summary of the current subject pairings, and in the second text area you will see a detailed list of the subject pairing connections information. In the middle of the master application window is a filename

area. This filename will be used on each of the subject machines for the stored data. (Additional information such as the round number, the subject machine IP address and the item number will be appended to this name.) The first part of the filename is read from the plan file and the second part can contain other more volatile information such as the subject numbers. In the third text area is a detailed summary of the current task and in the last text area errors are written.

NOTE. If you use a name which will result in existing logs being overwritten you are not warned about this. This is because the logs are written by the task controller and slave and not the master and the master does not check which logs already exist on the disks of the subject machines.

Trouble shooting: if Java says Exception in thread "main" java.lang.NoClassDefFoundError: master.Master check that you are in the Pigeon directory. If the error persists try

```
java --classpath . master.Master
```

If this works, please consult the documentation of the Java distribution that you have installed to modify the java *CLASSPATH* variable to include the current-working-directory, i.e., ".".

Trouble shooting: the master displays "no such element exception" – the configuration file is not in the correct format. Check carefully it matches the example files in format.

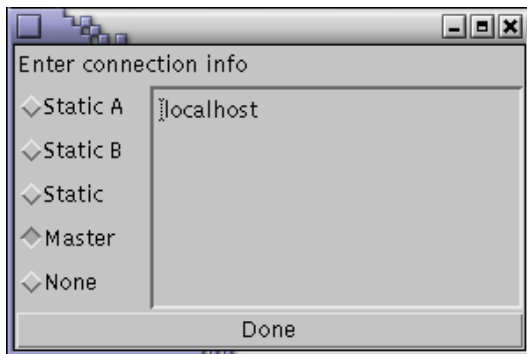
Now it is time to start the pairs of machines. At each machine that will have an experimental subject type:

```
cd <root of Pigeon installation>
java Experiment
```

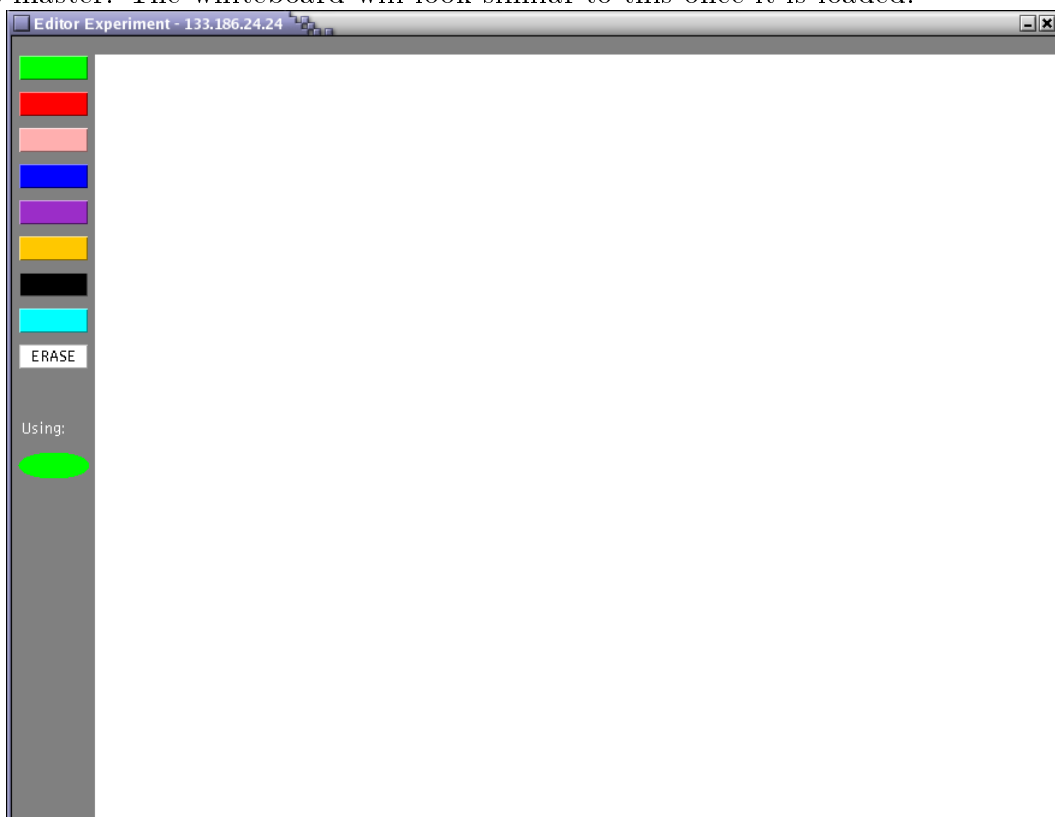
(If you are interested in testing a two editor setup on the same machine please read the note below.)

After starting properly, two windows should be displayed: the main Pigeon editor window and a second configuration dialog. The the exact size and location of the editor window is controlled by the settings given in the master configuration file. *Subjects can not move nor can they resize the editor windows.*

NOTE the configuration dialog may be behind the editor window. The dialog should look similar to the following screen-shot:



Raise the editor window if required and in the dialog replace the address displayed with the IP address of the master machine you just wrote down. Double check the address. Select the "master" radio button and then hit the OK button on the configuration dialog. The machine should now connect to the master. The whiteboard will look similar to this once it is loaded.



Alternatively you can pass command line arguments to the experiment to help to automate the starting of the subjects communication windows.

```
java Experiment <-master master_IP> <-code subject_code_number>
      <-port local_communication_port-number>
```

Trouble shooting: If the machines main Experimental vanishes after a minute or so, please check to see that:

- You got the IP address of the master machine correct.
- You did start the master application.

Trouble shooting: We have noticed some problems with how colored buttons are drawn by java in MacOS9 and MacOS10. They can appear as clear buttons on colored backgrounds in MacOS10 and as uncolored buttons in MacOS9. To solve this problem please set `hasMacOS9Bug` to be true in the master's option file.

For testing purposes, it is possible to run two editors on the same machine, but the second machine must be assigned a different communication port. By default the whiteboard uses port 9000, but a different port can be specified using the “-port” flag. For example:

```
java Experiment &
java Experiment -port 9001 &
```

Look at the master application's window and hit the check button. The window should show a machine has connected and is LOCKED continue this procedure for all the machines involved in the experiment. Then you are ready to start the experiment

To start the experiment hit the NEXT ROUND button ONCE on the master application. Each subject machine should now display an OK dialog. If you hit check on the master application the machines statuses should change to connected.

Once BOTH the subjects belonging to a master slave pair have hit OK the dialog boxes the editor window should change to display the task chosen and the round should begin.

2.3.4 Experiment progress

Important: At the end of the round it is important not to close the editors on the subject machines until they have finished writing out their log files. It is vital that all the “Round is over” dialogs OK buttons are pressed AND that the task bar with the timer has returned to a gray rectangle with no buttons or count down BEFORE you close the subject’s editor windows.

Subject’s editor windows can be closed by left clicking on the portion of the screen just below or around the place that shows the current drawing color with the control button pressed. A dialog comes up, in which you can press the quit button.

2.3.5 Analysis of results

The subject editor windows write a log file called:

```
<something>-<round number>-<IP address>:<port number>-<item number>
```

or

```
<something>-<round number>-<last part of IP address>-<item number>
```

Depending on the setting of the `useFullIPAddressName` option. Also a “DATA” file will be written at the end of each round on each of the task controller machines. Further information regarding the analysis of the results can be found in Chapter 7.

Chapter 3

Running an Experiment

This chapter contains a *brief* summary of what needs to be done while running an experiment. It is assumed that the software is correctly installed on the master and all the subject machines, and that the experimental materials and plan files have already been created. If this has not been done please refer to the instructions in Chapter 2.

3.1 Firing things up

To start the master application, on the master computer `cd` to the pigeon directory and then execute:

```
java master.Master
```

Next, on the master machine, you select the appropriate experiment plan file in the dialog.

Then you need to start the editor windows on each of the subject machines. On each machine `cd` to the pigeon directory and then execute:

```
java Experiment
```

In the dialog select the radio button beside the master label and then enter the master's host-name or IP address.

3.2 Running the task

On the master press the 'Check Subjects' button. Make sure that the appropriate number of subject computers are shown in the second text area and that they are all ALONE and LOCKED.

Add the subject numbers to the second text box in the file name section of the master application (the white text area in the middle of the master window), and check that the second text area from the bottom contains the correct settings. Make sure that the file name given is not the same as that used by any old data on any of the subject machines.

Press the 'Start Next Round' to begin the next round. The current round information is shown in the top of the second box from the bottom. At the end of each round make sure that all machines are ALONE and LOCKED before proceeding to the next round.

3.3 Cleaning up after a run

On each of the machines that were used to collect data make a folder for the newly collected data (preferably OUTSIDE the pigeon home directory). Copy the new data from the pigeon home directory into that new folder. It is also a good idea to copy the contents of those folders onto another computer as an emergency backup.

Chapter 4

Available tasks

4.1 The generic task

The least restrictive and most generally useful task provided by pigeon is the generic task. The generic task can be used in experiments where the experimenter provides all relevant task materials to the subjects outside of the pigeon application and does not require pigeon to collect any data outside of the shared editor. One example of such a task is a logic puzzle task that could be described for the subjects on a paper instruction sheet. While performing such a task the subjects could interact through pigeon's editor and provide an answer to the puzzle in the editor itself.

Tasks which require pigeon to manage the task materials or collect special task related information outside of pigeon's editor can not be conducted as generic tasks. Two such tasks, the music and the pictiionary task, are described in the next sections of this Chapter.

4.1.1 Task Overview

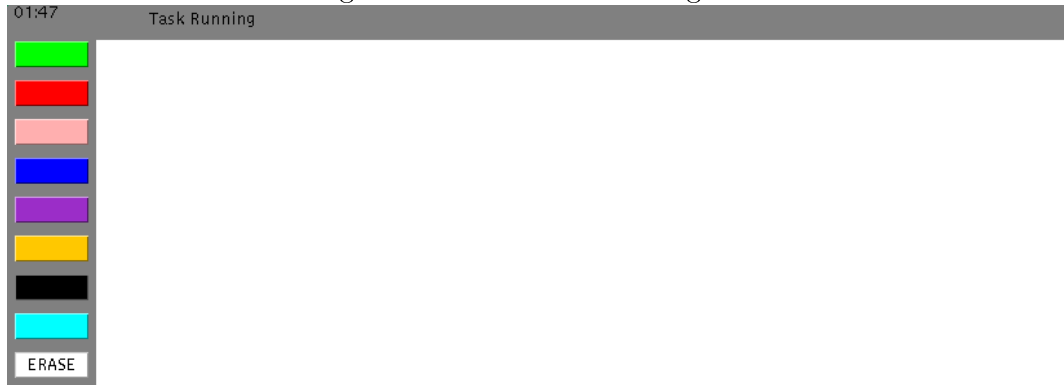
An individual item of a generic task consists of a timed interaction between a pair of subjects. At the start of the task both of the subjects' editors are locked (they do not respond to any mouse or key input). When ready, the experimenter can begin the task from the master computer. Once begun a dialog window opens on both computers asking the subjects to press an OK button when they are ready. Once both of these buttons are pressed a timer is displayed on pigeon's task bar and the editors are unlocked. The timer counts down the number of minutes and seconds remaining and when

it reaches 0 both editors are again locked and dialogs are opened telling the subjects that the task is complete. The data collected by the editor is automatically saved using the file name provided in the master application window.

4.1.2 Setting up the task

To use the generic task, `taskType=genericTask` must be specified in the master plan file. Though the generic task has no task specific options, any of the non-task specific configuration options can be used. The most important of these is the item duration (which is specified in seconds). For example, if `itemDuration=120` is given in the master plan file then subjects will be given two minutes for each item of the task.

A screen shot of the generic task's task bar is given below:



Once the master plan file is constructed the instructions in Chapter 2 and Chapter 3 can be followed for the actual running of the experiment.

4.2 The music task

The music task can be used to run experiments in which pairs of subjects are asked to communicate about pieces of music through pigeon's shared editor. The task can be presented in three different forms. In the first form, one subject is allowed to draw and is given one piece and the other is asked to decide which of two pieces is the same as the drawer's. In the second form, both the subjects are given one piece of music and they are asked to decide if they have the same piece or if they are different. In the last both subjects are presented with two pieces of music (one which their partner also has and

another different piece) and they are asked to select the piece that they have in common.

4.2.1 Task items

The music files used by the task are stored in the **TUNES** directory of the pigeon home directory. By default, in that directory the music is filed into 4 different sub-directories:

- **C-Maj** - Classical Major key pieces
- **J-Maj** - Jazz Major key pieces
- **C-Min** - Classical Minor key pieces
- **J-Min** - Jazz Minor key pieces

Additionally **slow**, **medium**, and **fast** can be included in file names to specify the pieces' tempo. If you wish to specify other materials directories, this can be done by changing the value of the **musicDirs** option.

Music can be stored in AU (Sun/NeXT audio data: 8-bit ISDN u-law, mono, 8000 Hz) or MP3 format. AU files can be played from inside java or by using an external native application, while mp3 files can only be played using an external native application. By default, the native application used to play au files is **play** and the native mp3 application is **mpg123** both of which are expected to be found in in the path. These settings can be changed by specifying the appropriate application with the **auPlayApp** and the **mpgPlayApp** options.

4.2.2 Task item selection

- Type 0: One subject has one piece (the drawer) and the other has two pieces (the follower). Only the drawer can use the editor (the follower's editor is locked). The follower must decide which of his/her two pieces is the same as the drawer's.
- Type 1: Each subject is presented with two pieces of music. One of the pieces is the same for both subjects while the others are different. The subjects must decide which of their pieces of music is the same.

- Type 2: Each subject is presented with one piece of music and they are each asked to decide if they have the same or different pieces of music. Some item selection balancing is performed, jazz vs. classical and major vs. minor.
- Type 3: Each subject is presented with one piece of music and they are each asked to decide if they have the same or different pieces of music. Task items are selected at random.
- Type 4: Each subject is presented with one piece of music and they are each asked to decide if they have the same or different pieces of music. Some item balancing is performed, jazz vs. classical, major vs. minor, and fast vs. medium vs. slow.
- Type 5: Each subject is presented with one piece of music and they are each asked to decide if they have the same or different pieces of music. Item order is not decided upon by software but rather read from external files. One text file (round0, . . . , roundN) for each round must be placed in the TUNES directory of the PIGEON directory. Each line of that text file must contain two white-space separated filenames relative to the TUNES directory. The first file is given to the task master and the second to the slave.

4.2.3 Setting up the task

Once you have appropriately named and organized the music files into the directories in the TUNES directory, you must specify `taskType=musicTask` and the appropriate item selection using the `pickType` option, e.g., `pickType=0`. You can also specify what file format and whether to use native or java music playing to present the music to the subjects. If you are running the task on an operating system which has a command line interface to the system's volume control you can also include a volume control on the editor auxiliary panel. This volume control executes a program called `volume` with an argument from 0-100 in the (or linked from the) `bin` directory of the pigeon home directory. See `task/VolumeScale.java` for details.

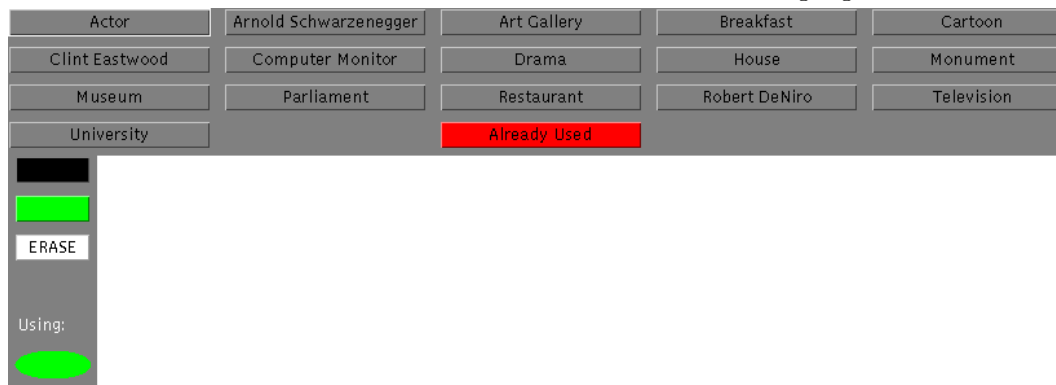
4.3 The Pictionary task

In the pictionary task, the basic goal is for one subject to communicate a term (like “house”) to the other subject who needs to select that term from a list. In the case of the drawing editor, this is done by drawing the term for the other subject (typically without the use of letters or numbers). Thus, at any point in the task each of the subjects plays either the role of the *drawer* or the *follower*.

The drawer is presented with a list of items, and needs to communicate, one by one, those items to their partner. The current item is highlighted, and then becomes gray once the drawer’s partner, the follower, selects an item from his or her list. The drawer’s task panel for the second item is shown in the following figure:



The follower is presented with a series of buttons each labeled with a different item. Once they understand which item their partner, the drawer, is trying to communicate they can select that item by pressing the appropriate button. The follower’s task panel is shown in the following figure:



4.3.1 Configuration

All the normal editor configuration options apply such as whether there is a shared-pointer etc, however for the task options there are a few “hardwired” options:

1. There is no time limit.
2. The number of items in a round is fixed to 16 (4 distractors and 12 targets).
3. The number of rounds is fixed to 6.

All of the item data for the pictorial task is stored in the PICT directory of the pigeon home directory. In that directory are item category directories each containing one line item files. Also in that directory are round files. The round files (named round0 . . . round 6) each contain a list of file names, one per line, of the items files that will be used in that round. The first 14 item files specified will be used as targets and the remaining four items will be used as distractors.

Item selection

The pictorial task comes in 8 different flavors described below. “Swap roles” means that at the end of each round the follower becomes the drawer and vice versa. Repeating means that the pair are presented with the items from the round0 file 6 times, and non-repeating means that in each round a different round file is loaded. Lastly in the case of “both draw”, both the drawer and the follower can use the editor, and when “follower can’t draw” is specified the follower can’t modify the contents of the editor.

- Type 0: Both draw, swap follower and drawer after each round, repeating- same targets and distractors each time.
- Type 1: Both draw, swap follower and drawer after each round, no-repeating-different targets and distractors each time.
- Type 2: Both draw, don’t swap follower and drawer, repeating- same targets and distractors each time.

- Type 3: Both draw, don't swap follower and drawer, no-repeating-different targets and distractors each time.
- Type 4: Follower can't draw, swap follower and drawer after each round, repeating-same targets and distractors each time.
- Type 5: Follower can't draw, swap follower and drawer after each round, no-repeating-different targets and distractors each time.
- Type 6: Follower can't draw, don't swap follower and drawer, repeating-same targets and distractors each time.
- Type 7: Follower can't draw, don't swap follower and drawer, no-repeating-different targets and distractors each time.

This is summarized in the following table:

type	swap roles (after round)	chooser blocked	repeating
0	yes	no	yes
1	yes	no	no
2	no	no	yes
3	no	no	no
4	yes	yes	yes
5	yes	yes	no
6	no	yes	yes
7	no	yes	no

The rotation of which round is first is NOT done automatically so for non-repeating experiments, if you don't want round0 to always be first round5 should be renamed to round0 and round1 to round2 etc.

4.3.2 How the task works

A brief description of how the pictonary task works will now be given.

1. The appropriate items are read from the item files for each of the 16 items (using the appropriate round file).
2. The last 4 items are chosen as distractors the remaining 12 are the targets.

3. If the `randomizeItems` option is set to true, the targets and distractors are appropriately randomized.
4. The drawer is presented the 12 target items in order labeled 1, 2, 3, ...12 with the first one highlighted. If the `drawerSeeAll` option is true the distractor items are also shown as items 13, 14, 15, and 16.
5. The chooser sees all 16 items. If the `sortFollowerItems` option is true, the items are sorted alphabetically and presented in order. If this option is false, the 16 items are presented in random order.
6. By setting the `canChangeColor` option to true and the `followerDefaultColor` option to some palette index other than 0, the drawer and follower can each be assigned a different drawing color which will be used throughout the experiment. For example, if the palette consists of black and green, and `canChangeColor` is true and `followerDefaultColor` is 1 then the drawer will always use black and the follower will always use green. If the drawer and follower roles swap after each round then the colors that each subject can draw in also change.
7. If the `oneUseButtons` option is true, once an item has been picked it cannot be picked again in this round. There is an extra "I have already used this" button which can be used multiple times. This button can be used when a subject recognizes that they mistakenly selected a previous item button. For example, imagine that the follower thinks that item 1 looks like a tea cup. The follower then selects the "tea cup" button which is then disabled (until the next round the follower cannot choose tea cup again). Now imagine that the second item drawn by the drawer really is a tea cup. In this case the follower would have to pick an item at random. But the follower don't want to select an item which may come up later in that round. So the "I have already used it button" can be used in this case. This button is always available. It makes a special entry in the task summary file so it can be differentiated from correct and wrong answers.
8. Once the 12 items have been gone through then for the repeating task round 0 is read again for the non repeating task round N+1 is read and we then go back to step 1.

During the round of 12 items NO randomization occurs; the choosers screen is only updated to disable choices they have already made, and the drawer's list is only updated to highlight the next item to draw and gray out the one just drawn. After the 12 items have been drawn then the next round file is read and the randomization is performed.

Chapter 5

User Configurable Options

Options are specified in the same way in both the master control files and the generated output data files. In both of these files options are given as key/value pairs. Options not explicitly given are assigned default values. For example, in a master configuration file's options section you could specify the following options:

```
itemLength=120 editorType=drawEditor taskType = generic-  
Task showSharedPtr =true rotateDrawing= 90 editorWidth=500  
editorHeight=1000 edgeColor=0,255,0  
drawPalette=green;red;pink;blue;purple;orange;black;cyan
```

The order in which options are specified does not matter, nor does white space, and spacing around '='s. All options are saved in the data files along with the drawing data (including the default values used).

For options which allow an *array* of data, the array can be specified as a semicolon (;) separated list of data of the appropriate format. In the above example, the option `drawPalette` was given as an array of colors, so each color was delimited with a colon.

Allowed option value types:

- **integer** – an integer value. Note that when being used to specify an on-screen location or size, such as window width, this number is taken to be the number of pixels. When being used to give a x location on the screen these numbers are taken to be the distance in pixels from the left edge of the screen and when being used to specify a y location the number is taken to be the number of pixels from the top edge of the

screen. The exact size of a pixel and the dimensions in pixels of your monitor's screen depends on the screen resolution you are using. OS dependent utilities can be used to determine these sizes and locations, but simple trial and error is often the simplest method.

- **string** – a series of any non-whitespace characters. Whitespace characters consist of the space, tab and return characters.
- **boolean** – ‘true’ or ‘false’ string (case insensitive).
- **color** – A valid color name or the RGB value of a color specified as three comma separated integers in the range of 0-255. Available color names are defined in the COLORNAMES array in the Options file. Currently the following colors are available:

```
green red pink blue purple orange black cyan dark-green
dark-red dark-blue gray dark-purple tan magenta dark-orange
```

For other colors the color can be specified as a RGB value. The RGB value of the color red would be given as ‘255,0,0’, and white as ‘255,255,255’ etc. There are many utilities for determining the RGB values of the color that you wish to use. For example, many image manipulation programs, like Photo-shop, have a color-editor tool that can be used to determine these numbers. On-line utilities can be found at: <http://www.mrcalculator.com/color.html> and <http://kresch.com/resources/javacolor.htm>.

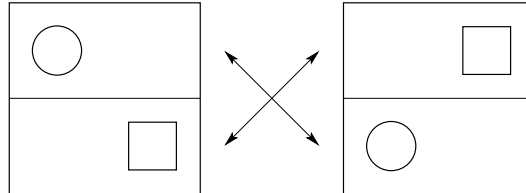
- **rectangle** – a series of four comma separated integers corresponding to the top-left x and y coordinates and the bottom-right x and y coordinates respectively.
- **special** – Values for these options are constrained to a limited set of values. For example the key ‘editorType’ can only be assigned the values ‘drawEditor’ and ‘textEditor’.

Option keys:

- Options for all editors:
 - **editorType special default:** drawEditor. Allowed values: drawEditor, textEditor.

- `editorWidth` `integer` default: 800.
- `editorHeight` `integer` default: 635.
- `mainWindowX` `integer` default: 50. The X coordinate in pixels of the top left corner of the editor's main window.
- `mainWindowY` `integer` default: 25. The Y coordinate in pixels of the top left corner of the editor's main window.
- `defaultMasterControl` `boolean` default: `true`. Whether by default the program starts on-line (under the control of the master) or not. This option is provided for network trouble-shooting purposes only.
- `showSharedPtr` `boolean` default: `false`. When this option is true, the mouse pointer of one editor is shown on the other editor of the pair. In the setting of the drawing editor, the mouse on one editor can be used by the subject to illustrate, without drawing, features on the other whiteboard.
- `edgeColor` `color` default: 128,128,128 (gray). The color used for the background of the task and auxiliary panels of the main window.
- `backgroundColor` `color` default: 255,255,255 (white). The background color of the editor.
- `hasMacOS9Bug` `boolean` default: `false`. We have noticed some problems with how colored buttons are drawn by java in MacOS9 and MacOS10. They can appear as clear buttons on colored backgrounds in MacOS10 and as uncolored buttons in MacOS9. When this option is true, special colored buttons are used to replace the default java buttons on the editor auxiliary bar (e.g., the palette for the draw editor).
- `partnerScreenUpdateType` `special` default: `simultaneous`. Allowed values: `simultaneous`, `oneLineAtTime`, `inChunks`.
- `divideScreenHorizontal` `boolean` default: `false`. Setting this to be true results in a normal full screen editor but:
 1. The screen is divided horizontally and subjects can not edit over the whiteboard's "equator".

2. If `otherAreaLocked` is true then a subject can only edit above or below the equator. The task pair's controlling member of the pair owns the top half and the task pair's slave the bottom half. (See `iControlTask` option below.) Note that the assignment of the controlling member of the pair is done transparently to users.
- `divideScreenVertical` **boolean** default: false. Like `divideScreenHorizontal`, except the editor is divided vertically into left and right halves.
 - `transposeRegions` **boolean** default: false. This option only has an effect when either `divideScreenHorizontal` or `divideScreenVertical` are true. When `divideScreenHorizontal` is true the screen is divided horizontally into two equal halves. What either subject edits in the top half of their screen appears in the bottom half of their partner's screen. What either edit in the bottom half appears in their partner's top half. If `otherAreaLocked=true` then the subjects can only edit the top halves of their windows (the bottom half is un-editable). This option has an analogous effect when `divideScreenVertical` is true except the window is divided vertically, and in this case when `otherAreaLocked` is true the subjects can only change the right side of the editor.

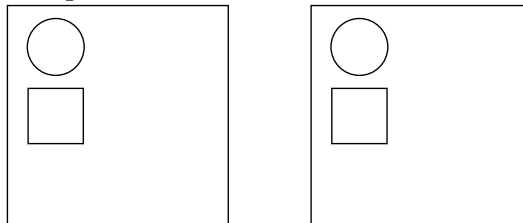


- `otherAreaLocked` **boolean** default: false. See the explanations with ‘`divideScreenHorizontal`’, and ‘`divideScreenVertical`’.
- `drawDivider` **boolean** default: false. Draw a divider line either horizontally or vertically on the screen to divide the screen into two halves. Whether horizontal or vertical is determined by the settings of ‘`divideScreenHorizontal`’, and ‘`divideScreenVertical`’.
- `dividerWidth` **integer** default: 10. The width of the dividing line specified with the ‘`drawDivider`’ option.
- `fullIPAddressName` **boolean** default: false. Whether the full IP address and port number are used in the file name.

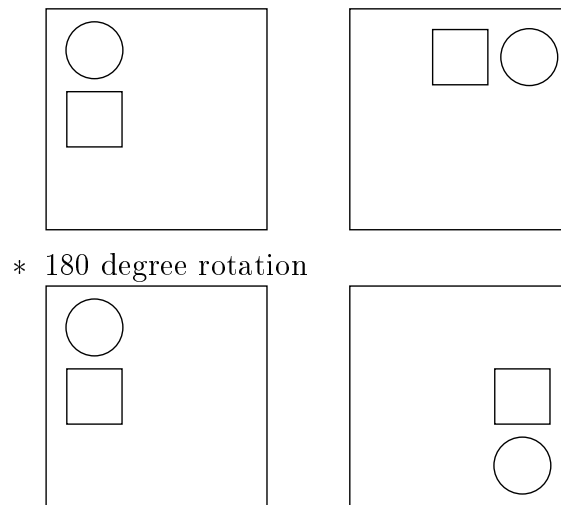
- Options for the Drawing editor:

- `mouseErase boolean` default: true. When true, an erase button is shown on the editor palette. If false it is assumed that a shift-drag is an erase.
- `backgroundImageName string` default: null. The name of the background image to display on the whiteboard. It is assumed that this will be a GIF file which is mostly transparent. The GIF will be drawn “on top of” any drawing on the whiteboard. If a non-transparent GIF is used the image will occlude the subject’s drawing.
- `backgroundImageX integer` default: 0. The x coordinate of the image’s top left corner. Note that when there is a background image, and `ROTATEDDRAWING` isn’t 0 this option is over-ridden to center the image on the editor.
- `backgroundImageY integer` default: 0. The y coordinate of the image’s top left corner. Note that when there is a background image, and `ROTATEDDRAWING` isn’t 0 this option is over-ridden to center the image on the editor.
- `rotateDrawing integer` default: 0, Whether drawing, when transmitted to the partner’s whiteboard, is rotated. You can rotate 0, 90 or 180 degrees only. 0 is specified when no rotation is desired, 90 when you wish to simulate two people sitting on adjacent sides of a small table, and 180 when simulating two people sitting facing one another. These rotation values are relative to the subject who is task pair’s controlling member. (See `iControlTask` option below.)

* 0 degree rotation



* 90 degree rotation



- `deltaSize integer` default: 1, The number of points contained in a `DrawDelta` message. This options is provided to reduce network traffic. If you are working over a slow internet connection you might want to change this value, however in most cases the default value works well. By default every recorded movement is immediately sent to the partner's computer. If higher than 1 is specified than a small cache of points is created and sent to the partner's computer when full. For example if you specify 3 then one packet of movement data is sent between the pair for every three recorded mouse movements. This will then reduce the network traffic to 30% of the default.
- `drawPalette Color array` default: green; red; pink; blue; purple; orange; black; cyan. The color buttons that are available on the subject's palettes.
- `canChangeColor boolean` default: true. Whether the color buttons on the palette are activated. If they are dis-activated then the subject can only draw in the default color.
- Options for the text editor:
 - None at the moment
- Options for all tasks:
 - `taskType string` default: `genericTask`. Allowed values: `genericTask`, `musicTask`, `pictionaryTask`.

- `restartAfterRound` `integer` default: 0. A rest value of 0 means no rest else a rest value of X means rest every X rounds
 - `halfWayWarning` `boolean` default: false. Should a dialog be opened half-way through a round. To close this special dialog you must control-click on the OK button. This option is useful when half-way through the task you need to change the experimental setting or materials. The control-click requirement will prevent the subjects from continuing the experiment until after you have changed the materials and control-clicked the button.
 - `itemDuration` `integer` default: 0. Time allowed for each item of the experiment.
 - `iControlTask` `boolean` default: false. This option is primarily for internal use only. *Unless you really understand the internals of the software*, it is suggested that you do not use/change this option. But if you do wish to do so, this determines whether that member of the pair is ‘controlling’ the task. This state is transparent to the users, but is used internally for various purposes.
 - `pickType` `integer` default: 0. The scheme used for the selection of experiment materials. The actual meaning of this number is determined by the selected task.
 - `showVolumeScale` `boolean` default: false. Whether a small volume scale should be drawn on the editor auxiliary panel. Note that this scale uses a native application to actually change the volume.
 - `giveTaskFeedback` `boolean` default:true. Whether at the end of each item feedback should be given to the subjects.
- Options for the Generic task:
 - None at the moment
 - Options for the Music task:
 - `useNativeMusic` `boolean` default: false. When using native music mode a separate process running native code is created to play the music. When not using native music, java 1.1 sun.audio classes are used to play the audio files. Note that at the moment sun.audio can only play AU files.

- `nativeMusicAppForwardSlash` `boolean` default: `true`. Whether the call to the native music application should use forward or backward slashes. When true `'/'` is used and when false `'\'` is used.
 - `useJavaSoundAPI` `boolean` default: `false`. Whether to use the java 1.3 sound API, or the unsupported sun audio in java 1.1.
 - `musicFileType` `special` default: `auFile`. Allowed values: `auFile`, `mp3File`.
 - `auPlayApp` `string` default: `play`. The default native application used to play AU files.
 - `mpgPlayApp` `string` default: `mpg123`. The default native application used to play MP3 files.
 - `musicDirs` `string` array default: `"J-Maj", "J-Min", "C-Maj", "C-Min"`. The default sub-directories of the TUNES directory in which the music files are stored.
- Options for the Pictionary task:
 - `itemDirectories` `string` array default: `"global", "local"`. The names of the directories under the top level PICT directory in which the materials files can be found.
 - `oneUseButtons` `boolean` default: `false`. Whether the follower can select an item using a labeled button more than once in each round.
 - `followerDefaultColor` `integer` default: `0`. When using the draw editor, by default the first color in the draw palette is the default selected color when the task begins. If you would like the follower of the task to start with a different default color than the sender, the button number of the desired default color can be specified. The first color in the palette is button 0, the second button 1 and so on. If the draw editor is not used this option had no effect.
 - `drawerSeeAll` `boolean` default: `false`. When false, the drawer is presented with just the targets and when true the drawer is presented with the distractors as well (though they will never be drawn).
 - `randomizeItems` `boolean` default: `true`. When true, randomize the order of the items given in round file for both the drawer and the

follower. When false, the items are presented in the exact order that they are given in the round file to the drawer, the follower gets a random order.

- `sortFollowerItems` `boolean` default: false. When true the follower is always presented items in alphabetic order. This option over-rides `randomizeItems`.

Chapter 6

The Playback/Markup Tool

6.1 Overview

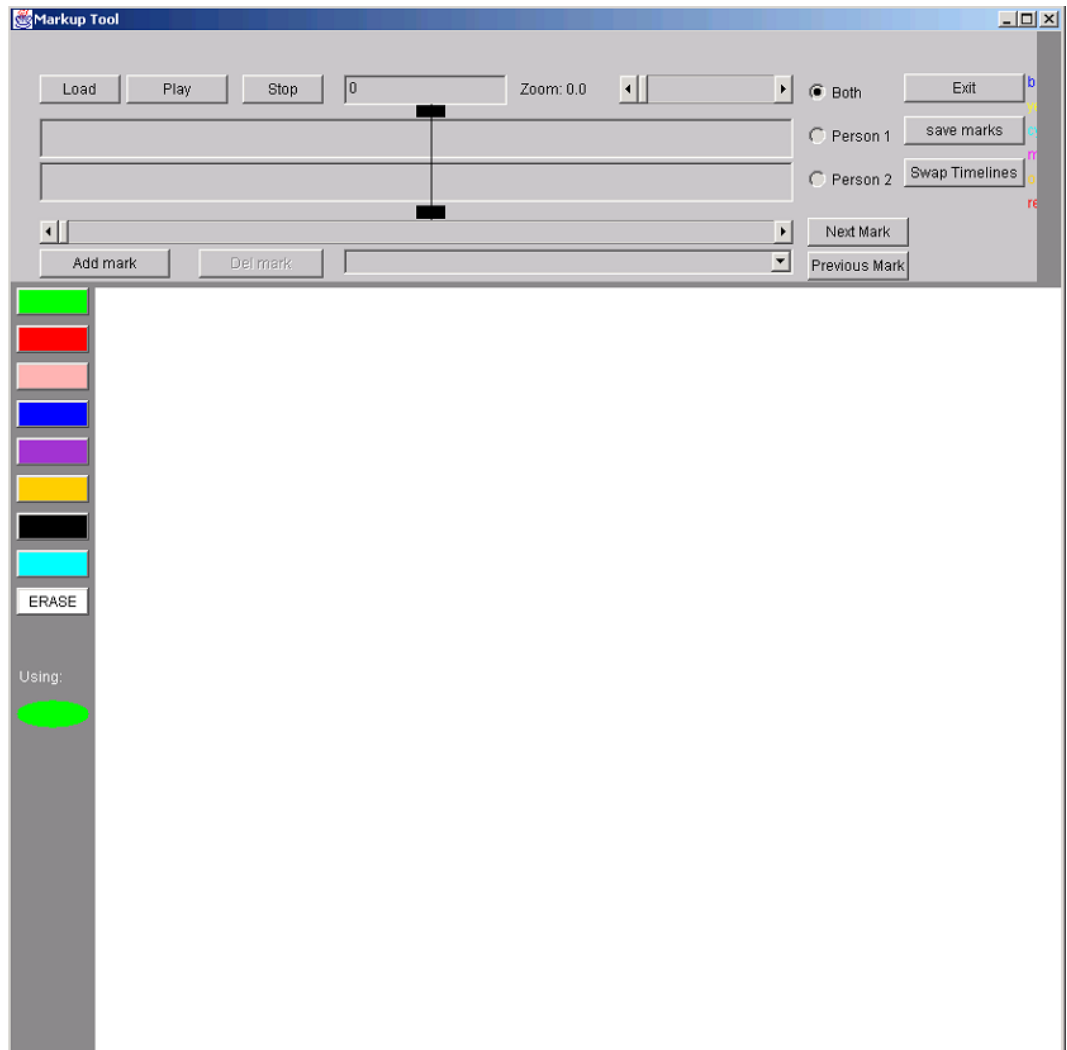
The markup tool is designed to playback logs captured by the whiteboard tool. It is useful when you want to do detailed content analysis of the log or to look at how the drawing and interaction progresses. You can navigate in various ways using the time line and can control some aspects of the playback. It is also possible to insert marks with an optional description and comments at any point in the log and navigate between marks.

6.2 Loading up the tool

In the Pigeon directory on the command line type:

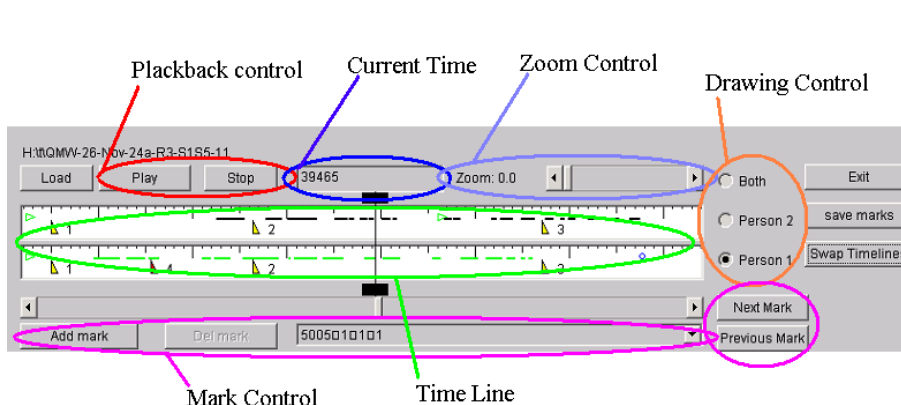
```
java editor.draw.Markup
```

The Pigeon playback and markup tool should then load. It has one main screen, The tool is divided into two parts, at the top control panel and time-line and underneath these the place where the log is displayed.



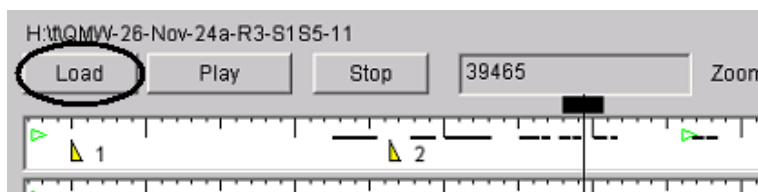
6.3 Overview of the Pigeon Control Panel

The Pigeon control panel is divided up into a number of logical parts, which will be referred to in this manual in the next sections.



6.4 Loading a Log

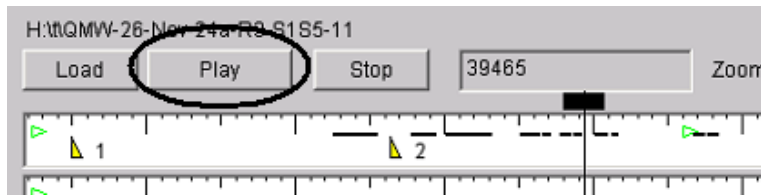
Little can be done until a log is loaded. You can load a log by pressing the "Load" button. Located near the top left of the tool, on the left of the playback controls.



You are then asked to select a log file to load. If you have no data on hand, you can test the markup tool with some of the sample data provided in the pigeon `sample-data` directory, e.g., `ticTacToe-test-R1-157-0`. The top part of the tool will update to tell you which log is loaded and the playback controls will update to show the contents of the log.

6.5 Controlling Real-time Playback

There are a variety of ways to control playback the simplest way is to play the log in real time. This can be done by pressing the "Play"/"Pause" button. It is located next to the "Load" button in the playback controls part of the control panel.

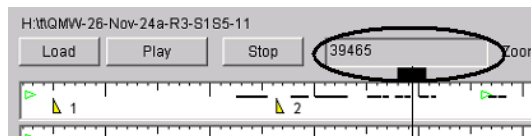


You will notice that this mode is analogous to the playback of a video recording. The "Play"/"Pause" button changes to pause allowing you to pause playback at any time during the replay. Conversely, if you pause the playback the "Play"/"Pause" will change to "Play". The "Stop" button is located next to the play/pause button in the playback controls and stops the playback. It has no effect if playback is not in progress (it is just there as an alternative to using the "Play"/"Pause" button). When the log is playing you will notice that the current time control increases from 0 and the time-line scrolls from right to left.

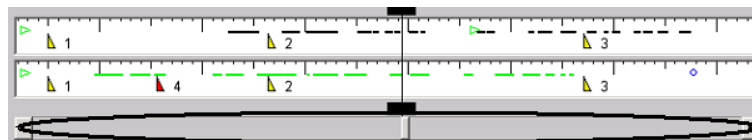
6.6 Jumping around the Log

An alternative to real time playback is to jump to a certain time point in the log. This can be done a variety of ways.

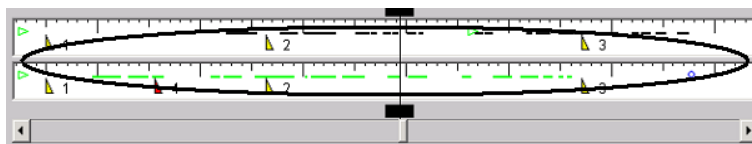
- Type a time in milli-seconds (1000 milli-seconds is 1 second) into the current time box and hit enter – this will jump you to the specified time and show you all the drawing that occurred up to that point in time. The current time box is situated to the left of the playback controls.



- Use the time line scroll-bar either by clicking the left mouse button inside it to jump or by pressing either of the arrows to smoothly move through the file. The scroll-bar is located at the bottom of the time line



- If you left click on the time line you will jump to that point in the log. For instance, clicking where the black arrow is will jump you just over two large tick marks i.e. 20 seconds forward.

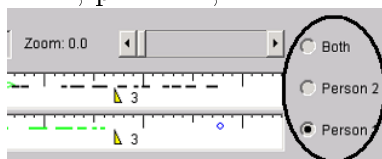


Note: if you are playing back and you use these jumping features playback will stop.

You can continue playback from the new position you jumped to by pressing the play button.

6.7 Controlling who's drawing you see

There may be situations where people draw over each other or other reasons why you only want to see one person's drawing. You can use the drawing control positioned on the right of the time-line to select only the drawing by person 1, person 2, or both.

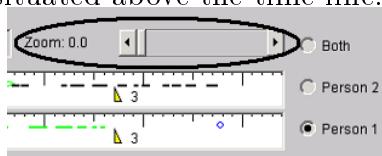


Note using this control during playback will stop the playback.

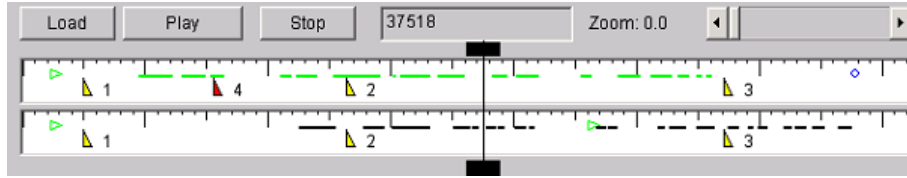
You don't have to re-position or start playback to use the control again you can change who you see as many times as you want in a row.

6.8 Zooming the time-line

Sometimes you may want to be able to jump to a precise position whilst other times you want to be able to move a long way forward or back in the log but you don't know the time. In this case the zoom control can help. It is situated above the time-line.



At minimum zoom 0.0 the program will fit the entire log on the time-line on the screen.

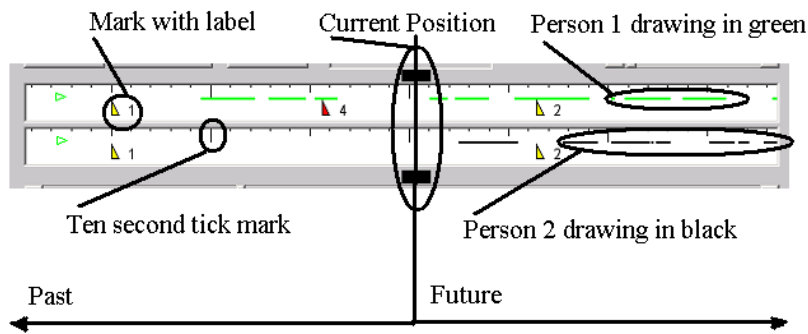


This allows you to move large distances but sacrifices precision. At maximum zoom 1.0 the time line is as long as the program can make it and this allows you to move more precisely forwards or backwards. Intermediate zoom values trade-off precision against the size of the overview that the time-line provides.



6.9 A closer look at the time line

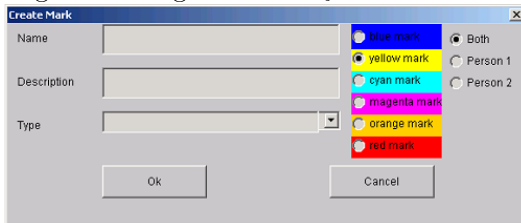
The time line is calibrated in seconds and like a ruler uses small tick marks for seconds, longer ones for every 5 seconds and the longest tick mark for each multiple of 10 seconds. The black bar in the center of the time line shows the currently displayed part of the log. Everything to the left of the mark is in the past (and therefore is also displayed in the whiteboard window below the time-line) everything to the right of the line is in the future (and therefore has not yet been displayed). The two sets of marks correspond to two subjects' drawing. The colors reflect the colors used in the drawing (so a blue line on the whiteboard corresponds to a blue mark on the time-line). Erasure is shown in the time-line as a double black line. In the illustration one subject has drawn in green and the second subject has started to draw later in black. Overlap indicates that subjects were drawing concurrently.



The triangles on the time-line are covered in the next section.

6.10 Marks

Sometimes it is useful to be able to add one or more markers to a log to indicate interesting events or for other reasons. Marks can be created using the “Add mark” button. This adds a mark to the current position in the log. A dialog box asks you for some information about the mark



The name is what will be displayed on the time line to help you identify the mark. The description is optional. The type is also optional. You can select a color for the mark or accept the default. You can choose to add a mark to only one person’s time-line or to both. In the example above the red mark labeled 4 has only been added to person 1 whilst the other marks have been added to both people. You can create multiple marks at the same place although you will only see the latest mark on the time-line. When you create a mark you will also see that it has been added to the list of marks displayed under the time-line. You can jump to a mark by selecting the mark in this list or by using the “Next Mark” and “Prev Mark” buttons which jump you to the next or previous mark.

Note: the mark list shows the marks in the order they were created.

The “Next Mark” and “Prev Mark” buttons take you to the next and pre-

vious marks in time respectively. Next and previous are sensitive to whether you have selected to see only one person's drawings. For instance if you have selected person 2 or both in the drawing control the next mark from mark 1 is mark 4. However, if you selected to only see person 2's drawings the next mark from mark 1 is mark 2

You can delete a mark by selecting it in the mark list and hitting the "Del mark" button

Marks can be saved at any time by hitting the "Save marks" button. The program will also ask you if you want to save your marks before you exit or if you try to load another log.

In order to avoid modifying the original log, marks are saved into a separate file with the same name as the original log but with .m on the end. If the log file is subsequently re-loaded the mark log file will also be re-loaded.

6.11 Swapping Time-lines

Under some situations you may not be happy with who is on the top time line and who is on the bottom, for instance consistency between multiple logs. In this case you can swap the time lines by using the "Swap Time-lines" button. The labels for controlling who's drawings are displayed are also swapped so that person 1 always refers to the same drawing and person 2 always refers to the same drawing. If you have selected to only see persons 1 or 2 drawings then when the labels swap your selection will be maintained.



Chapter 7

Data Analysis Tools

The log files produced by subjects drawing on the whiteboard are saved, on each machine used, as

```
<your file name>-R<round number>-<ip address>:<port>-<global item number>
```

or

```
<your file name>-R<round number>-<last part of ip address>-<global item number>
```

depending on the value of the `useFullIPAddressName` option. The actual file name is determined in each case by the name entered in the Pigeon master window at the start of the experimental run. The fact that both machines store a log file of the interaction provides some redundancy and protection against data loss. Note that network delays and lack of synchronization between different machine clocks means that the timing information in the two files will not be identical (although they should be very close).

The log files can be analyzed and manipulated in a variety of ways. The tools provided fit into three categories. Tools to playback the logs of the drawing, tools to analyze the logs and tools to convert the logs into different formats.

7.1 Playback and Markup Tools

The playback and markup tool is also invoked by running

```
java editor.draw.Markup
```

A window appears that is similar to the main Pigeon whiteboard window except the task bar is replaced by a time-line which provides an overview of the drawing activity in a log file and allows you to navigate around it. Pressing the load button allows you to load a log file. Please note that that there may be a delay while the system parses the log file.

Once the log file is loaded, the time-line is updated and will show zero or more colored lines. The divisions of the time-line represent 1, 5, and 10 second intervals and the horizontal colored lines show the color and duration of drawing activity by subjects. If the whiteboard was used by a pair of subjects there will be two colored lines one above the other. The lower part represents one subject's activity and the upper the other subject's activity.

There are several ways to navigate the log files. You can jump directly to a point in the log by entering the desired elapsed time (in milliseconds) into the edit box and pressing enter. The screen will update to show the state of the drawing (including erase activity) at the time entered. Alternatively, pressing start plays back the log in real time from the beginning. Pressing stop stops the playback. When the real time playback is stopped it is also possible to click on another portion of the time line and the screen updates to show the drawing at the time indicated by the black line.

A more detailed description of the Mark-up tool and its use to code events in the log files can be found in Chapter 6.

7.2 Log File Format

The log file format is as follows:

First comes the version of the pigeon editor module used:

```
# draw 200
```

After this comes the time and date the experiment started:

```
# 998988641689 Tue Aug 28 08:50:41 GMT 2001
```

Next is the name of the machine generating this log

```
# JAMES-PC/138.37.88.106
```

Then there is also an optional list of key value pairs of user configurable options. These options are used for two purposes. Firstly to help document how an experiment was set up and secondly to provide the play back tools with enough information to display experiment specific variations such as screen split bars. When a log is read the `readRawPicture` routine in `Editor` parses this line and sets the options in `main.Options` accordingly. Further information regarding the available options can be found in Chapter 5.

```
# OPTIONS:
```

Now come the data generated by subjects' mouse actions, drawing activities, and task bar use. The first part of each of these lines is the name of the data feature that will be described and the second is the version number of the described feature. Next is the start and end times of the features given in milliseconds from the computers clock. Then whether the feature was created locally or remotely on the other subject's machine is given. Following this is feature specific data. Some example lines of data will now be given.

A line giving the start time of the current item of the task:

```
Started 200 998988625005 998988625005 local 138.37.88.106
```

A line describing one line that was drawn:

```
FreeLine 200 998988627058 998988628190 local green 150,217
150,217 151,215 154,210 157,204 163,188 181,174 199,168
221,164 245,162 267,162 283,163 288,164 290,165 291,166
292,168 295,173 297,179 299,195 299,211 299,229 299,247
298,263 298,281 298,295 298,301 300,315 304,333 318,353
338,369 360,373 384,375 406,374 430,368 458,352 488,334
522,316 554,311 582,310 606,311 622,315 636,321 642,335
647,340 650,345 652,350 654,364 654,380 654,386 652,391
650,394 649,397 649,398
```

A line describing a subject's button press:

```
MouseStuff 200 998988620469 998988620469 138.37.88.114
select-tune flat/5
```

The extra fields at the end of the file are specific to the task being performed. The following examples come from the extra data written by the pictinary task. First the name of the task:

```
#EXTRA: Pictionary Task
```

The next line is task specific setup such as the number of pieces of music being played or target items and distractors etc.

```
#EXTRA: number of targets 12 number of distractors 4
```

The next line is also task specific and contains information about e.g., the round or item number in an experiment.

```
#EXTRA: block 0 item 1
```

Information about what is displayed on the original task bar for both participants is displayed. This is used by the play back task to simulate views of the task bars and so that any data analysis tools know what choices the users had. For a new task a new parser should be written and hooked into the simulateDrawing tool.

```
#EXTRA: drawers list: green/11::flat/6::box/4::box/2::
green/2::green/5::box/9::box/10::flat/7::flat/5::green/1::
flat/3::animals/5::animals/10::animals/4::animals/7::
```

```
#EXTRA: target order: green/2::flat/5::box/4::green/11::
box/10::animals/7::animals/4::flat/3::green/1::box/2::
flat/7::green/5::animals/5::flat/6::animals/10::box/9::
```

7.3 Drawing Statistics

Most of the drawing statistics are created by running `Editor.draw.MakeDrawingData` on the data files.

```
java editor.draw.MakeDrawingData <list of files>
```

There are some switches at the top of the `MakeDrawingData` java file that control what statistics are produced (by default all available statistics are generated). Three files are created. These files can be read into a spreadsheet for editing and further analysis.

`drawing-data.txt` contains a comma separated set of statistics, one line for each log file. The following is the contents of that file for the sample data file `ticTacToe-test-R1-157-0` which can be found in the `sample-data` directory of the pigeon home directory.

```

ticTacToe-test-R1-157-0, Real Total Time, 60427, Time
from start to first draw, 1342, Local Count:, Draw, 6,
Erase=, 0, Local Time:, draw, 8351, erase, 0, Buddy Count:,
draw, 12, erase, 2, Buddy Time:, draw, 6812, erase=, 4607,
General Overlap =, 922, DD Overlap =, 922, DE Overlap =,
0, EE Overlap =, 0, General Overlap Count =, 1, DD Overlap
Count =, 1, DE Overlap Count =, 0, EE Overlap Count =,
0, General Overlap Avg Distance =, 102.1077861869505, DD
Overlap Avg Distance =, 102.1077861869505, DE Overlap Avg
Distance =, 0.0, EE Overlap Avg Distance =, 0.0, General
Overlap Dynam Avg Distance =, 154.38476190310035, DD Overlap
Dynam Avg Distance =, 154.38476190310035, DE Overlap Dynam
Avg Distance =, 0.0, EE Overlap Dynam Avg Distance =, 0.0,
Local Line Length =, 1857, Local Erase Length =, 0, Remote
Line Length =, 1643, Remote Erase Length =, 854, Local
General Overlap Dynam Length =, 318.10922437758745, Local
DD overlap Dynam Length =, 318.10922437758745, Local DE
overlap Dynam Length =, 0.0, Local EE overlap Dynam Length
=, 0.0, Remote General Overlap Dynam Length =, 363.0178557195236,
Remote DD overlap Dynam Length =, 363.0178557195236, Remote
DE overlap Dynam Length =, 0.0, Remote EE overlap Dynam
Length =, 0.0, Local Number of Colors =, 1, Remote Number
of Colors =, 3, Total Different Colors used =, 2, complexity=,
9872.646994535518, totalPixels, 6405, regular, m, 114.0,
barX, 12.56140350877193, barY, 9.192982456140351, varX,
11.983071714373652, varY, 12.06802092951677, covar, -0.7311480455524778,
overlap, m, 13.0, barX, 11.384615384615385, barY, 7.6923076923076925,
varX, 4.390532544378699, varY, 4.059171597633137, covar,
-1.2662721893491122

```

`ticTacToe-test-R1-157-0` is the name of the log (data) file from which the statistics have been generated.

`Real Total Time,60427` is the total time taken to complete the item or trial in milliseconds

`Time from start to first draw, 1342` is the time elapsed from the start of the task until one of the subjects used the editor.

`Local count:`, indicates the start of the statistics for machine (and the person using it) that made this log file

Draw, 6 is the number is lines drawn

Erase=, 0 is the number of white (background color) lines drawn. Because erasing is achieved by drawing in white over the top of existing lines this does not correspond to the number of original lines erased (i.e., over-painted)

Local Time: indicates the start of the timing information for the local machine

draw, 8351 indicates the time in milliseconds spent drawing by the local person

erase, 0 indicates the time spent in milliseconds erasing - i.e. drawing background (usually white) color lines

The same information is then listed for the partner machine (and subject) in the same format.

General Overlap =, 922 shows the total amount of overlap for the subject's drawing activities. This is basically the amount of time during the experiment in which both pointing devices were pressed.

DD Overlap =, 922 shows the amount of drawing overlap accounted for by concurrent drawing of (non-background color) lines.

DE Overlap =, 0 shows the amount of drawing overlap accounted for by concurrent drawing and erasing.

EE Overlap =, 0 shows the amount of overlap accounted for by concurrent erasing.

General Overlap Count =,1,DD Overlap Count =,1, DE Overlap Count =,0, EE Overlap Count =,0, gives the number of times overlapping drawing occurred. Each time a subject's pointing device was pressed while that subject's partner is already drawing is counted as one occurrence of overlap. The last three counts give the breakdown of the number of times that this overlap occurred with them both drawing, one drawing and the other erasing, or both erasing.

General Overlap Avg Distance =, 102.1077861869505, DD Overlap Avg Distance =, 102.1077861869505, DE Overlap Avg Distance =, 0.0, EE Overlap Avg Distance =, 0.0, gives the average distance between the centers of gravity of the pairs of overlapping lines. The center of gravity of a line is computed as the average of all the points of the line. The average distances is then broken down between the various kinds of overlap.

General Overlap Dynam Avg Distance =, 163.1992264233166, DD Overlap Dynam Avg Distance =, 163.1992264233166, DE Overlap Dynam Avg Distance =, 0.0, EE Overlap Dynam Avg Distance =, 0.0, gives the approximate

average distance between the pens during simultaneous drawing and or erasing (sampled five times per second).

Local Line Length =, 1857 records the total length in pixels or 'ink' used in drawing by the subject on the 'local' machine. This value is calculated mathematically using the coordinates of the points (*not* using the actual pixels shown on the screen). Erasure has no effect on this measure. Local Erase Length =, 0 records the total length in pixels of 'erase' or background color pixels used in drawing by the subject on the 'local' machine. Remote Line Length =, 1643 and Remote Erase Length =, 854 record this same data for the other subject.

Local General Overlap Dynam Length =, 318.10922437758745, records the approximate total length of all the local drawing activity that occurred during simultaneous drawing, drawing/erasing and erasing/erasing. The next three measures Local DD overlap Dynam Length =, 318.10922437758745,, Local DE overlap Dynam Length =, 0.0,, Local EE overlap Dynam Length =, 0.0,, divide this length into draw-draw, draw-erase and erase-erase categories. Remote General Overlap Dynam Length, Remote DD overlap Dynam Length, Remote DE overlap Dynam Length, and Remote EE overlap Dynam Length record this same data for the other subject.

Local Number of Colors =, 1 this is the number of distinct colors used by the local drawer

Remote Number of Colors =, 3 is the number of distinct colors used by the remote drawer

Total Different Colors used =, 2 this indicates the total number of distinct colors used by both people (machines).

7.3.1 Complexity measure

complexity=, 9872.646994535518

The complexity measure calculates the visual complexity of an image following the algorithm described in the paper "Identifying Letters" by Pelli, Burns, Farell and Moore (to appear in Vision research). This measure (perimeter squared over 'ink' area) provides an index of how visually complex a drawing is. Note that this measure does not take account of color.

7.3.2 Screen Usage Statistics

The data files then contain a number of statistics that summarize the shape of the screen use distribution.

`totalPixels`, 6405 gives the total number of non-white pixels in the final drawing. Note that the lines drawn are two pixels wide so this is roughly double the total line length. However, the line length calculations do not take into consideration erasure and the pixelation of lines as they are rendered on the screen which are considered in the `totalPixels` measure.

`regular` Now, using 32 by 32 pixel regions, statistics regarding the location of non-white pixels in the final image are given.

`m`, 114.0 gives the total number of 32 by 32 pixel regions in a grid covering the drawing area that contained at least one line in the file.

`barX`, 12.56140350877193 gives the average X co-ordinate of the lines in the log file.

`barY`, 9.192982456140351 gives the average Y co-ordinate of the lines in the log file.

`varX`, 11.983071714373652 gives the variance of the X co-ordinates in the log file.

`varY`, 12.06802092951677 gives the variance of the Y co-ordinates in the log file.

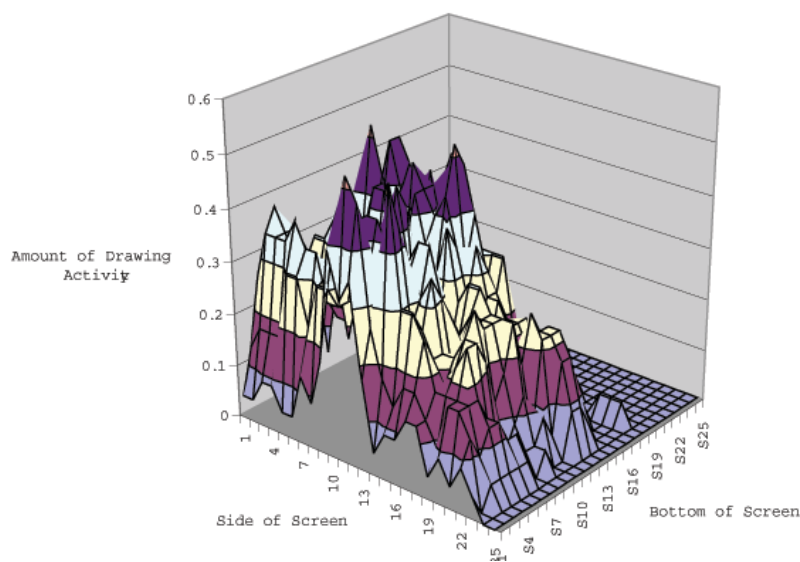
`covar`, -0.7311480455524778 gives the co-variance of the x and y co-ordinates in the log file.

`overlap` Finally, using the same 32 by 32 pixel regions, statistics regarding the overlapping of drawing are given. One 32 by 32 pixel region is marked as having overlap if both of the subjects drew in that area in the final image. The data is presented in the same manner as the `regular` screen usage statistics.

7.3.3 Screen Usage Graphs

`distrib-plot.txt` provides a frequency distribution of the amount of drawing activity in each area of the the whiteboard screen. `distrib-overlap-plot.txt` provides a frequency distribution of the overlapping drawing activity in each area of the the whiteboard screen. Both of these are presented as x,y,z values for each (arbitrary) 32 by 32 pixel region on the whiteboard. This data can be used to generate 3-dimensional plots of the drawing activity and overlap

drawing activity in which higher peaks indicate more drawing activity in that area of the screen.



7.4 Converting Logs to Pictures

Log files can be saved into FIG format by executing

```
java editor.draw.MakeFig <list of log files>
```

Fig files can be imported into a variety of UNIX and Windows tools and converted into most other formats. FIG does not use lossful compression and so provides pixel accurate reproductions of the actual drawings.

Further information on the FIG file format and programs which produce and use FIG files can be found at www.xfig.org.

7.5 The task DATA file

At the end of each round of the task, the task controller machine writes a summary file describing the performance of the subjects in performing the task. The exact format of these files is governed by the tasks themselves.

Acknowledgments

We gratefully acknowledge the support of the ERSC/EPSRC PACCIT initiative through the grant MAGIC: Multimodality and Graphics in Interactive Communication (L328253003). This work was also supported in part by ATR Media Integration and Communications Research Laboratories, Japan.

Bibliography

- [1] HEALEY, P., SWOBODA, N., UMATA, I., AND KATAGIRI, Y. Representational form and communicative use. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society* (2001), J. Moore and K. Stenning, Eds., pp. 411–416.
- [2] HEALEY, P., SWOBODA, N., UMATA, I., AND KATAGIRI, Y. Graphical representation in graphical dialogue. *International Journal of Human-Computer Studies* 57, 4 (2002). Special issue on interactive graphical communication.