MultiDraw:
A Cooperatively-Controlled Drawing Game

by

Kyle C. McClellan

A senior thesis submitted in partial fulfillment of

the requirements for the degree of

Bachelor of Science
With College Honors

Computer Science & Engineering

University of Washington

June 2004

Presentation of work given on May 5th, 2004

Thesis and presentation approved by _____
Steven L. Tanimoto, Advisor

Date _____

Contents

Chapter 1

Introduction

My hope with this paper is to acquaint the reader with the program MultiDraw. I will briefly explain the motivation behind the creation of the program and the system it has been integrated into before I discuss the features of the program in detail. I will also provide instructions for running MultiDraw on a variety of platforms; discussing various problems I have encountered, and conclude with some ideas of how I foresee this project continuing in the future.

1.1 Background

MultiDraw was created in the context of the OLE (Online Learning Environments) Research Group at the University of Washington. The OLE group is a cross-discipline collaboration working to develop educational tools and improve the feasibility of using computer-based learning environments in the classroom. Two of the major goals behind our work are promoting interactive student learning and providing means for unobtrusive assessment of student knowledge. Keeping these goals in mind, we have developed the INFACT (Interactive Networked Facet-based Assessment Capture Tool) Online Learning Environment. The INFACT system provides an integrated suite of tools that can be used by instructors to foster interactive learning. INFACT also facilitates the collection of student data within the system's database that can be used for unobtrusive assessment and improving the system as a whole.

MultiDraw is a tool loosely integrated with the INFACT system. To use the program an individual must have an account within the system. This allows the program to record all student activity on the INFACT database making it possible for an interested party to evaluate the record at a later date.

1.2 Overview

MultiDraw is one of a few collaborative learning games integrated into the INFACT system. It is designed to be used as a tool for both educational purposes and research. Collaborative learning, in a general sense, is learning where students work together to solve a problem or complete a task. It aims to involve students in the learning process in a manner that is as engaging as it is educational. In some collaborative settings, students can discuss class content, share their thoughts and ideas, and brainstorm possible

solutions to the problems they are faced with. In other settings, both competitive and non-competitive, they may be working in groups to accomplish team-related goals. In most settings, collaborative learning encourages communication and hands-on interaction with classroom material.

Cooperatively-controlled games such as MultiDraw are a form of collaborative learning. In these games, individual users are only given a portion of the overall control so they must work in groups to accomplish the goal of the game. As mentioned above, this setup promotes communication between group members. Also, the common goal inherent in playing a game encourages teamwork. In educational cooperatively-controlled games, learning to work together is just as important for success as learning the subject matter. Cooperatively-controlled objects were first defined and studied by Lauren Bricker [Bricker, 1998].

MultiDraw is a cooperatively-controlled drawing game with educational intent. The basic functionality of the game allows users to draw images by working in teams. Each mode in MultiDraw is set up to use the same drawing method and control separation. Beyond simply indulging their artistic leanings, users interact with geometric shapes in a coordinate plane. MultiDraw exposes students to coordinate systems, shapes, geometric formulas, and helps students get an idea of how different shapes are formed. The capabilities of the program make it more useful as a tool to augment the understanding of students who are studying geometry than as an instructional tool. MultiDraw can be a useful to students with varying levels of knowledge. Modes in the game range from requiring a very basic understanding of tracing to requiring that the user has a working knowledge of the algebraic functions that describe different geometries.

To make MultiDraw easy to use, the program operates on a variety of platforms. Students can access it through their internet browsers, play it as an application on their PCs, or use it on a handheld iPAQs. This type of flexibility requires that the program not exceed the capabilities of any platform it may run on. Both the size and user interface are implemented to work within the confines of a small display such as the screen on the iPAQ.

1.3 The INFACT System As It Relates To MultiDraw

There are two specific ways that MultiDraw has been integrated into INFACT. First, it is impossible to run the program without having a user account in at least one of the INFACT forums. This limitation ensures that the user has the account credentials necessary for the program to successfully connect to the database and record information. This leads into the second way that MultiDraw has been integrated into the system which is through event logging. Once a user begins the game, all their actions are recorded in a log kept on the database. The database sorts all the information the game sends according to session (the period of time the game is running), user ID, and time. The purpose of this is to keep a detailed enough record of events that a log can be evaluated to gain a greater understanding of student learning and understanding. Logs can be analyzed for evidence and trends that are illustrative of a student's understanding in a

particular area. They can also be review to evaluate the effectiveness of the program and gain a better understanding of how the program is being used by students.

Chapter 2

MultiDraw

2.1 API for the Development of Collaborative Games

MultiDraw is built on top of an API created by Eric Bessette in 2002 that simplifies the development of collaboratively-controlled games. Eric's Multi-Control API has limitations and the current version of MultiDraw uses a library based closely on the original API that has been augmented in a few ways. The API handles user login and group creation (and/or dissolution), before handing control over to the program. It ensures that a user has an account in the INFACT system and then sets up threads in the background that allow the program to interact with the database. The WriteEventsThread in the API makes it a simple task to log game events. As a result of the API, MultiDraw is distributed between a main server program and many clients. The setup uses the RMI package that is a part of the Java language. Graphics and control are handled locally by each client while all the data is stored and manipulated on the server. This ensures synchronization and takes some of the computational load off the client-side application which is important when running on the iPAQ.

A few of the major adjustments made to the API include the addition of a replay screen to allow a user to easily replay the game and the addition of a wait screen to help synchronize the display. The replay screen appears after the game has concluded and gives the user the option of playing again, changing groups, or quitting. The selected choice is applied to all group members. The wait screen is displayed to some users until all group members are ready to start the game. This allows all users in the group to move to the play screen at the same time which minimizes synchronization and control problems.

2.2 General Setup

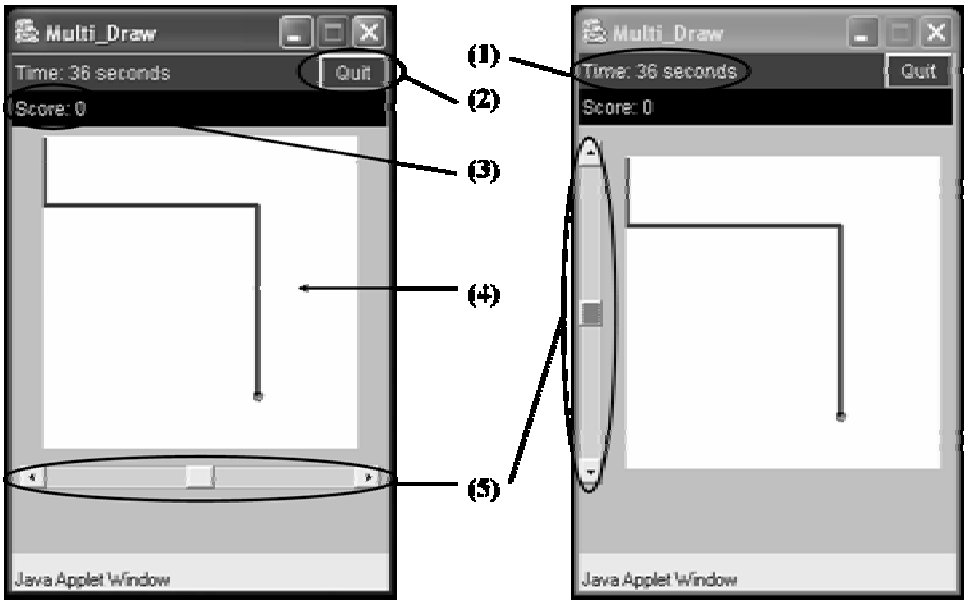
MultiDraw is a cooperatively-controlled game. It must be played in groups as users only have a portion of the control. MultiDraw is a two-dimensional drawing game and its current control division requires that it be played in teams of two. Each user controls drawing on a single axis. In conventional terms, one user controls drawing on the x-axis and the other controls drawing on the y-axis. To support play on iPAQs, MultiDraw has been written using Java 1.1.8. This constrains the visual side to be implemented with Java's awt libraries instead of the more sophisticated swing libraries.

To facilitate group play, the API allows for the formation of groups. An individual has to 'host' (or start) a new game. Other users are then allowed to see the game and may attempt to join it. Once the group reached capacity, it can start a game. The individual who elected to host the game is always considered the owner of the game. There are a number of option menus that are available to the owner while the other group members are left waiting. Granting the owner sole control of game setup enforces the synchronization of game options. Ideally, group members would be able to discuss the options they would like selected and could work with the owner to set the game up.

Control in MultiDraw is in form of scrollbars (sliders). The scrollbars are displayed either horizontally or vertically depending on the axis that the user controls to provide a visual representation of the control. For example, the user that controls movement in the horizontal direction will see a horizontal scrollbar. The users adjust the scrollbars to control velocity of the drawing. The best way to envision this is to imagine that the user is driving a point around and leaving an image of the path. The scrollbars allow for a range of magnitudes in either direction.

All drawings originate in the upper-left corner of the screen. There is no method for choosing which portions of the path (drawing) are displayed, so every picture will have a line that connects it to the upper-left corner.

The elapsed time (from the start of the game) is displayed in a label at the top of the screen (1). The game is timed as more of a formality than anything else. Most of the time limits exceed the time needed to draw an image, but they are in place to prevent the program from running indefinitely. A quit button appears directly to the right of the time (2). If this button is pushed, it will end the drawing session for the group and take the user to the replay screen. Even if the time limit expires, the quit button must be pushed to proceed. Below the time label is the score (3). The score is only used in some of the game modes. A button may be displayed to the right of the score labeled '?' (Not shown). This button will be described more in section 2.3.3. Below the score label are the canvas that contains the drawing (4) and the scrollbar used for control (5).



Game Screens in MultiDraw in Free Draw Mode

2.3 Modes of Play

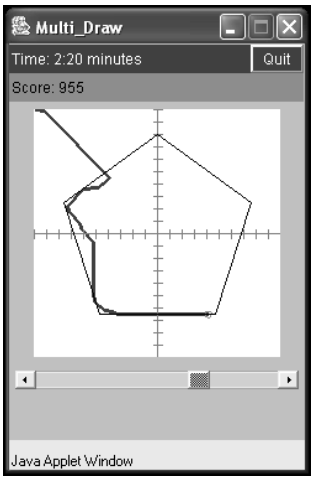
There are 3 different modes of play in MultiDraw. Each has its own purpose and feature set. However, all modes are two-player such that one player controls all movement along the x-axis and the other controls movement along the y-axis. An important aspect of the game regardless of the mode of play is learning how to share control in order to draw complex shapes and pictures. All modes also support the general features of the program and database logging.

2.3.1 Free Draw Mode

Free draw mode is the most flexible mode of use in MultiDraw as there is no explicitly defined goal. It provides complete game functionality in an environment devoid of requirements in a manner that encourages creativity. Students can draw anything they feel inspired to using MultiDraw in this mode. One of the useful aspects of having access to game functionality is that event logging on the INFACT database is enabled. For example, a teacher interested in what students know about a particular shape not supported by the other game modes could require the students to draw the shape in free draw mode. The teacher could then observe the logs produced by the program and get a good idea of what the students understand about that particular shape. This kind flexibility is useful in overcoming the limitations that exist in the other game modes.

2.3.2 Copy Mode

Copy mode is essentially an implementation of tracing. It requires users to trace a shape and scores them on how skillfully they perform. In setting up a game, the owner chooses from a library of shapes that have been implemented within the program. The shape types available to choose from are circle, ellipse, rectangle, and polygon. If the owner decides to trace a polygon, the number of sides the polygon has and whether it should be a regular polygon become available options. Once a shape type has been chosen, the program dynamically generates an instance of that shape, and the game can begin. In copy mode a pair of axes is displayed and all shapes are constrained to them. This ensures that all shapes can be easily defined by an algebraic formula. This constraint is considerably more important in plot mode and will be discussed again in section 2.3.3. In copy mode, both the shape and the pair of axes appear on the canvas. Once all group members have been sent to the game screen, a group can begin to trace the shape. The game then gives them a score based on how well they do. Scoring will be examined in greater detail in section 2.4.



Both the shape and the axes are displayed in copy mode

2.3.3 Plot Mode

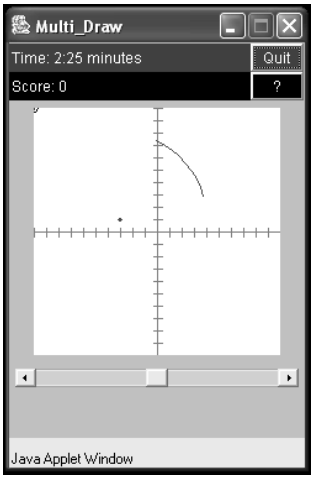
In plot mode, a group is given a mathematical description of a shape and asked to draw it. The descriptions are dynamically generated, but will always contain enough information to define the form and location of the shape. Plot mode is similar to copy mode in many ways. First, plot mode utilizes the built-in library of shapes which a group must choose from before they can play. Second, scoring is done in the same way and is based on how well a group draws the shape. Third, axes are displayed on the drawing canvas and all generated shapes are still constrained to them. As mentioned previously, this constraint is of greater importance in plot mode. It forces all shape-defining quantities to be whole numbers with respect to the coordinate system which makes the shape descriptions decipherable.

The major difference between plot mode and copy mode is that the shape is no longer displayed in plot mode. Instead, the group draws the shape based simply on description. This requires a higher level of mathematical knowledge and visualization ability. At the option screen, descriptions are generated dynamically upon request until the group confirms they have found one they would like to draw. This allows users to view multiple descriptions until they find one they feel comfortable drawing. Another difference between plot mode and copy mode is that plot mode does not support irregular polygons. This is because there is no particularly elegant way to describe a polygon mathematically (especially an irregular one). To avoid creating a list of points in the shape description, only a limited scope of polygons are supported.

Once the group enters the game screen, there are a few noticeable differences. As stated earlier, the shape is not displayed. All the user will see in the background of the drawing canvas is a pair of axes. Also, there will be a button to the right of the score labeled with a question mark. Once this button is clicked, the canvas will be hidden and a text area containing the description will be displayed. This allows any user to view the description as many times as necessary. Below the text area, a check box is displayed

that allows the user to request a hint. The '?' button can be pressed again at any time to return to the drawing canvas.

Hints are generated dynamically in plot mode. Once a user requests one, a small portion of the shape will be displayed until the user decides to hide it again. For instance, a single corner point of a rectangle gets displayed each time a hint is requested. Since hints are generated dynamically, a different hint can appear on each request (though no particular actions are taken to prevent generating hints that have been seen before). Also, different hints may be displayed to different users in the group. This encourages group members to share hints and interact with each other as they try to draw the shape.



A hint is displayed in plot mode

2.4 Scoring

Score is calculated incrementally. Each time the cursor is moved, a small score addition is calculated and added to the total score. The score that gets displayed on the game screen, therefore, is the sum of all score additions up to the current point in the game. The final score is the sum of all the score additions throughout the entire drawing or tracing process. Scoring is done by group, so every member in the same group will receive the same score

2.4.1 Motivation

The reason for adding a scoring feature to MultiDraw was to increase the somewhat nebulous quality of 'replay value'. Skill-based scoring adds another dimension to MultiDraw. Users can work not only towards tracing/drawing perfectly, but also towards achieving the highest score. While the two goals overlap quite a bit, score reflects skill and speed as well as accuracy.

2.4.2 Calculation

Three different factors are considered when calculating score; accuracy, speed, and time. Accuracy is used to calculate a base score. The base score can then be multiplied by speed. Whether this occurs or not is based the time that has elapsed since the game began. The result of this multiplication is the score addition at any moment in the game. The formula for calculating a score addition is:

if (current time < estimated time for completion) then
*score addition = base score * (x-magnitude + y-magnitude)*
else
*score addition = base score * .5*

The quantity 'accuracy' is based on the number of pixels distance between the cursor and the edge of the shape. If the distance is zero, accuracy is considered to be perfect, and if the distance equals or exceeds nine pixels, accuracy is considered to be poor. Between zero and nine pixels of deviation, there is an incremental scoring scale. The current scoring metric for calculating base score is:

Deviation	Base Score
0	10
1 or 2	7
3,4, or 5	4
6,7, or 8	2
9+	0

Accuracy is not based on a time component. In other words, accuracy isn't calculated using a parametric function. A user can draw the shape at any speed they choose. It is also perfectly legal to trace a shape in either direction. In fact, the user could trace a shape one way, reverse direction, trace over the shape in the other direction, and still get a good score. For each shape, the exact way that accuracy calculations are implemented slightly differs.

Speed is a calculated as the sum of x and y magnitudes as indicated in the score addition formula above. To simplify calculations, x and y magnitudes are just added together to get the total speed. This gives a slight bonus to diagonal movement that calculating the actual speed using Pythagorean's theorem would not.

The estimated time required to finish drawing or tracing a shape is based on the perimeter of the shape. In the calculation, a constant is added to the perimeter to counteract the time it takes the program to load and the group to move the cursor to the shape. The formula for calculating the estimated time component is:

estimated time = perimeter + 14

The motivation for creating an estimated time component is to prevent users from inflating their scores. Because scoring is not done in a parametric fashion, a user could

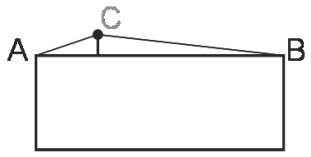
simply trace a small portion of the shape numerous times to increase their score. With the estimated time component in place, however, it becomes difficult to increase the score in a noticeable manner once the estimated time has elapsed. It is still possible for a user to pad their score; it would just be a tedious process.

As a small twist to this whole setup, there are small quantities that can be added or subtracted from the score as a result of prolonged periods of good or bad performance. For instance, perfect tracing will, after a short period of time, trigger a 'Precision Bonus' where the calculation of the score addition is done with a base score of fifteen instead of ten. Extended periods of poor tracing will set the base score to negative five and begin to decrement the score in a 'Deviation Penalty'. When either state occurs, a suitable indicator is displayed adjacent to the score in the score label.

2.4.3 Implementation specifics

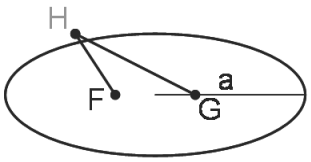
There are three different scorers implemented for MultiDraw. Every scorer has functionality for pre- and post-scoring periods in addition to regular scoring functionality. The reason for adding this functionality is to orient and initialize the scorer before it begins to operate and to decide when to stop scoring and what to do afterward. In a general sense; scoring doesn't begin until the cursor nears the shape and ends only after the cursor has returned to the point where scoring began after tracing or drawing the shape.

The point-to-point scorer can be used to score all shapes composed strictly of straight edges. To determine accuracy, the scorer calculates the altitude of the triangle created by the cursor (C) and the two vertices close to it (A and B). Scoring can conclude once the cursor nears the point at which scoring began after closely passing over half the vertices in the shape.



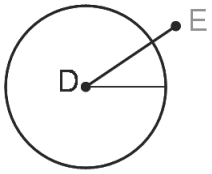
The altitude of triangle ABC is calculated to get the accuracy

The ellipse scorer is used to score ellipses. It calculates accuracy by subtracting the distance from one focus (F) to the cursor (H) to the other focus (G) from twice the length of the major axis (a). Scoring stops after all four quadrants (relative to the center of the ellipse) have been visited and the cursor is near to the point at which scoring began.



Accuracy is calculated using foci F and G

Instead of using the ellipse scorer to score circles, the circle scorer is used (a circle is a special case of an ellipse with both foci located at the center). The motivation behind creating a circle scorer was to simplify calculations. Accuracy is calculated based on the difference between the center (D) of the circle and the cursor (E) and the length of the radius. Similarly to the ellipse scorer, scoring can conclude after all four quadrants have been visited and the cursor approaches the angle where the tracing or drawing started.



Accuracy is calculated using the radius

2.4.4 Tips on Improving a Score

To get a high(er) score, trace quickly and accurately. The most important aspect to getting a good score can be a high speed. However, if a user loses accuracy when speed is increased, the benefit may be little to none. Once a user can regularly attain a given level of accuracy, they can increase their score by up to 500% by tracing a shape at a higher speed. Also, to improve a score, the user needs to complete the shape within the estimated time of completion. This doesn't tend to be a problem, however, when the shape is traced quickly. Something else to note is that getting 'Precision Bonuses' essentially multiplies everything by 1.5 which can be more beneficial than increasing the speed by one.

2.5 Feedback

Feedback is another feature of MultiDraw. It provides the user with immediate indication as to how well they are drawing or tracing a shape. The aim of providing feedback is to help users develop their skills by prompting them when they begin to do well. The feedback takes two forms; numeric feedback and color feedback.

The score that groups receive in copy and plot modes acts as numeric feedback. The incremental nature of how the score is calculated provides users with a rough idea of how well they are performing. However, since most users aren't familiar with the quantities used in scoring, it's difficult for them to interpret, and subsequently react to, the numeric feedback they receive during the game. Instead, the final score at the end of the game gives the user a rough idea of how skillfully they performed over the entire shape. If the score is high, they know they've traced or drawn the shape well. Even so, this type of scoring does not help the user improve during the course of the game.

The alternate form of feedback built into MultiDraw is color feedback. Both the score bar and cursor change colors according to how well the user is tracing or drawing at

a given moment. The color is directly related to the base score (calculated from the accuracy) used in the scoring algorithm. A user drawing perfectly will see the score bar and cursor turn green while poor drawing will cause them to turn black. As with the accuracy metric, between the extremes of green and black, a color scale is used to reflect differing degrees of accuracy. This type of feedback gives the user a clear idea of how they are performing at a specific point in time and can cue them to make adjustments with how they are drawing.

The desire in using color feedback is to provide the user with a visual assessment of how they well they are performing that they can react to. When the color begins to slide away from green, the user can try to correct their error. The color scheme includes (in order) green, yellow, orange, red, and black. The use of a progression similar to what can be found on a traffic light was intentional with the hope that the majority of users who come into contact with MultiDraw will already have specific concepts associated with the colors used and will react on a more intuitive level.

Color feedback can be enabled in plot mode or copy mode by checking the 'Feedback Enabled' check box in the options menu.

2.6 Event Logging

MultiDraw logs all game events that occur in the course of its execution using either the INFACT database or a local file. The idea of logging everything that happens in the course of a game is to record it in enough detail that the entire game could be reconstructed or 'replayed' at a later point in time. Therefore, not only is the image recorded, but so is every system and game mode variable that effects how a game is being played. Database logging can either be enabled or disabled. If it is disabled, a log will be written to a file local to the root folder of the program with the file name 'MultiDraw Session_<username>_log.txt'. Here is an excerpt from a log file:

Wed May 26 14:11:47 PDT 2004 -- View Changed -- Switch To – 8 ***(1)***
Wed May 26 14:12:04 PDT 2004 -- View Changed -- Switch To -- 6
Wed May 26 14:12:04 PDT 2004 -- Group Members -- Username -- kcm2
Wed May 26 14:12:04 PDT 2004 -- Group Members -- Username -- kcm3
Wed May 26 14:12:04 PDT 2004 -- Axis Controlled -- Axis – x

In a log file, every event is time stamped so the ordering and recreation of a game session is possible. For the rest of the excerpt, timestamps will be omitted for readability:

Game Mode -- Mode – 2 ***(2)***
Shape Type -- Class -- multi_draw.client.base.geometry.MCircle
Shape Values -- Values -- Center:(60,140) | Radius:50 ***(3)***
Shape Description Generated -- Description -- Draw a circle described by the equation (x+4)^2 + (y+4)^2 = 25. ***(4)***
Color Feedback State -- Enabled? -- true
Time Limit -- (in minutes) -- 5
Play Started -- --
Description View -- -- ***(5)***
Hint Values -- Values -- HintNum:206 ***(6)***
Hint State Changed -- Enabled? -- true

Returned To Game View -- --
Description View -- --
Hint Values -- Values -- HintNum:111
Hint State Changed -- Enabled? -- false
Returned To Game View -- --
.
Game Values -- Values -- Position:(1,0) | Direction:(1,0) | Score:0 | Feedback:0 ***(7)***
.
Game Values -- Values -- Position:(107,126) | Direction:(0,3) | Score:68 | Feedback:10
Game Values -- Values -- Position:(108,135) | Direction:(1,3) | Score:79 | Feedback:7
Game Values -- Values -- Position:(109,138) | Direction:(1,3) | Score:82 | Feedback:7
Game Values -- Values -- Position:(111,141) | Direction:(2,3) | Score:87 | Feedback:10
Game Values -- Values -- Position:(112,144) | Direction:(1,3) | Score:90 | Feedback:7
Game Values -- Values -- Position:(112,147) | Direction:(0,3) | Score:93 | Feedback:7
.
Game Values -- Values -- Position:(14,102) | Direction:(4,-2) | Score:147 | Feedback:0
Game Values -- Values -- Position:(39,91) | Direction:(5,-1) | Score:155 | Feedback:4
Game Values -- Values -- Position:(59,88) | Direction:(5,0) | Score:157 | Feedback:20
Game Values -- Values -- Position:(68,88) | Direction:(4,0) | Score:157 | Feedback:20
.
Game Values -- Values -- Position:(131,95) | Direction:(4,-5) | Score:157 | Feedback:20
Wed May 26 14:14:51 PDT 2004 -- Play Ended -- --

There are some events in the log that are difficult to interpret. Those events have been set in bold type and numbered so they can be referred to easily.

- Event 1 is a view changed event. The text of the event indicates that a view switch has been made to frame 8. As with many of the numerals that seem similarly out of place in the log, 8 corresponds to a constant in MultiDraw. Specifically, these constants can be found in MainFrame and MdrawFrame. Using a program constant in the log file, however, makes creating a log replay tool easier as values are in the form used by the computer. The log makes replay easier at a slight cost to readability.

- Event 2 records a constant in the same manner as event 1 did. Constant for game mode can be found in OptionPane.

- Event 3 is a record of the important values of the shape being displayed. The line directly above it, indicates that the shape is a circle centered at (60,140) with a radius of 50. The numbers in event 3 don't reflect the way the shape overlays the axes, but rather the position it is in on the display. For example, on a set of axes, the aforementioned circle appears to have a radius of 5.

- Event 4 is the algebraic description of the shape that corresponds to the values in event 3. If multiple descriptions are generated in plot mode (as described in section 2.3.3), each value-description pair will be recorded in the log and the last one on the list will be the one that is actually used.

- Event 5 indicates that the '?' button has been pushed in plot mode and the user has switched to the description view. The event 'Returned To Game View' indicates that the user has switched back to the game.

- Event 6 is a record of the hint that is generated. Every hint that is generated has a hintNum that defines it.

- Event 7 is a Game Value event that records the variable game values every time the direction changes. If straight lines are drawn between the points recorded in the Position field of each consecutive event, the drawing will be recreated. The value in the Direction field records (x-magnitude, y-magnitude). Score is the total score, and Feedback is the number that the color feedback keys on. The Feedback numbers are constants that can be found in Scorer. When the number in the Feedback field is 20, scoring has ended.

Chapter 3

Running MultiDraw

MultiDraw is written in Java and can be used on different platforms. The platforms the program has been tested on are Windows 2000, Windows XP, UNIX, and the iPAQ 3800 series with Windows CE 3.0. It is not a requirement that users be running on the same platform to play a game together. The first step to running MultiDraw on any platform is to start the server-side application on the server kirsch.cs.washington.edu. You must have access to the kirsch server to do this. If you do, the application can be easily started from the root of the MultiDraw directory using a shell with the command: *sh bin/ServerApp.sh* and stopped with the command: *sh bin/ServerApp.sh stop*. Once the server has created a RMI registry (it will indicate successful creation), client programs can be started on any platform.

3.1 Running MultiDraw on a Desktop

MultiDraw can easily be run from a desktop using a Windows batch file, internet explorer, or a shell on UNIX. To use the program, you also need to be able to log into the INFACT database. If you do not already have a user ID in one of the forums, the following are two user IDs that have accounts in the Kyle Development Forum that can be used with the program.

UserID: kcm3 – Password: kcm3
UserID: kyle – Password: kyle

3.1.1 Running MultiDraw as a Java Applet

To run MultiDraw as an Applet, you need to have a Java applet viewer installed and a program to access the internet. As with anything java-related, a viewer can be downloaded at java.sun.com. If you're uncertain as to whether you have one installed, go to http://kirsch.cs.washington.edu/cgi-bin/nickb/pda/infact-forum/pda.cgi and complete the logon. If the application appears to have problems after you select MultiDraw, it is likely that you need to download the viewer. To exit the program, close the internet window that appears behind the program or press the back button.

3.1.2 Running MultiDraw as a Batch File

To run MultiDraw with a batch file on a Windows machine, you must have a JRE installed. As in the case of an applet, double click on the batch file (located in the root MultiDraw directory) to attempt to run the program. If nothing appears to happen, it is likely you need to download a JRE from Sun. Otherwise, the batch file should take you directly to the program.

3.1.3 Running MultiDraw as a Shell

To run MultiDraw with a shell on a UNIX machine, you must have an installed JRE. If you don't, get one before trying to run the shell. Also, you need to have Xwindow capability and tunnel connections (typically I enable X11 tunneling. I don't know if other options are available). The shell can be executed from the root MultiDraw directory using the command: *sh bin/MdrawApp.sh*. Multiple instances of the program can be run from a single command using: *sh bin/MdrawApp.sh X* where *X* is the number of instances you want to run.

3.1.4 Running MultiDraw from the Command Line

Running MultiDraw from the command line is slightly more difficult than using the other methods and I won't describe the format in detail here. If you are interested in running it from the command line, I suggest that you open either the .bat or .sh file, copy the contents, and add the --help flag after StartApplication to view the range of options that are available from the command line.

3.2 Running MultiDraw on an iPAQ

Running MultiDraw on the iPAQ tends to be more complicated than on a Desktop. Since iPAQs that are left unplugged discharge and lose their memory, I'll start near the beginning. Make sure the iPAQ has a JRE installed. Jeode EVM Version 1.9 comes on a CD packaged with the iPAQ 3800. Also figure out a way to connect the iPAQ to the internet. In all my tests, I found that wireless connections were not reliable enough to run the program, but my hardware was outdated, so it is very possible that the program can run wirelessly. Otherwise, establish a link via the cradle with a computer that has internet access installing Microsoft ActiveSynch as necessary. Once you've taken care of the internet connection, copy the iPAQMultiDraw folder to the iPAQ. In it there should be a shortcut to the Jeode virtual machine that can be used to start the program.

There is also the possibility of running MultiDraw as an applet on the iPAQ, my settings never allowed for it, but it would be worth navigating to the link in 3.1.1 and attempting to run the program as an applet. If it worked, it would be the easiest solution.

3.2.1 Security and the .policy File

It is important that you enable the use of .policy files on the command line for the VM on the iPAQ. The virtual machine on my iPAQ originally came with them disabled. It's possible that you can still have permission violations even if you enable .policy files (which will prevent the application from running). If that's the case, you can either edit the java.policy (located in \Windows\lib\security on my iPAQ) file to look like this:

```
grant {
  permission java.security.AllPermission;
};
```

or use the shortcut labeled UNSECUREMultiDraw.exe. It is poor practice to grant programs this kind of permission, but in this case, it will eliminate all permission related errors and allow MultiDraw to run.

Chapter 4

Future Work

The current version of MultiDraw is still a basic tool with drawbacks that limit the practicality of using it for educational purposes. There are a lot of different actions that can be taken to increase its capabilities and to make it a better instrument for both education and research.

4.1 Expand to More Dimensions

MultiDraw is currently a two-dimensional, two-player game. There is nothing inherent in the game that limits it to supporting groups of two, though. It would be easy enough to allow for more dimensions and more players. A three-player version could be implemented by adjusting the code where the groups are formed and creating a three-dimensional canvas view. The canvas could be tricky, but it is likely that someone has already created something that would serve the purpose. Even a four-dimensional version could be attempted. Having a drawing program that supported four dimensions could be very useful as most people have great difficultly visualizing a shape that is more than three-dimensional.

4.2 Library Augmentation Tool

One of the main drawbacks to both copy mode and plot mode is their rigidity. The functionality of both modes is dependent upon a pre-defined library. The library currently consists of four types; circles, ellipses, rectangles, and polygons. Once a shape has been added to the library, instances are generated dynamically, so the number of different circles that can be created in the program is only limited by the size constraints of the display. However, if a shape isn't in the library it won't be supported in either copy mode or plot mode. For instance, there is no way to request the generation of a trapezoid (it is possible that requesting a four-sided polygon might return a trapezoid, but highly unlikely).

A way to remedy this would be to create a library augmentation tool. It could be used by an instructor to define a shape to add to the library. The tool could generate the code necessary for the shape to be used in the system. A tool of this nature would make MultiDraw a more flexible program and, therefore, more useful to instructors.

4.3 Individual Score Calculations

Scores are currently calculated for groups, and every member of a group is given the same score. The problem in this case is that not all group members may be at the same skill level or be doing the same amount of work. The score is as much a reflection of how well group members work together as it is of individual skill.

An intriguing idea is to apply lessons learned from analyzing the log files to in-game player scoring. Once trends were identified in the log files that seemed to indicate improvement, the game could watch for those trends during the game. When they were identified during play, score bonuses could be given.

4.4 Alternate Control Options

There is nothing inherent in MultiDraw that forces the control to be magnitude based. Instead of controlling the velocity of the cursor, the sliders could control absolute location. Another option would be to let the sliders control acceleration of the cursor. In both cases, the feel of the game would be the only thing changing.

Another direction to explore would be to implement non-intuitive modes of control. For instance, a polar system could be implemented. One user could control magnitude and the other, direction. Control could also be mapped through a function so that movements along the scrollbar didn't directly correspond to the results on the screen. In every case, it would be interesting to see what happened and how users responded to the different types of control.

4.5 Log Conversion for Sketch Replay

Log conversion is one of the more practical continuations of this project. One of the tools built into the INFACT system is a sketch recorder. Every sketch created in the context of the INFACT forum is recorded with respect to time and can be replayed like a movie. As the sketch-replay tool already exists, it would be less work to create a conversion script that could translate the image related events in a MultiDraw log to a form that is understood by the sketch player than to create a drawing player for MultiDraw.

4.6 Rework Color Feedback Scheme

The color scheme used in MultiDraw for giving feedback is purposefully similar to the color scheme used in traffic lights. The hope was that the color indicators would already

be familiar to most users. The problem that has been recently brought to my attention is that it is unwise to use green and red in the same color scheme as they appear the same to colorblind individuals. It would be worth the effort in the future to reconstruct the scheme so it was still familiar to most users, but without the aforementioned limitations.

4.7 Heading Indicator

It has been mentioned that a heading indicator might be a useful addition to MultiDraw. In a future version, it might be worth while to add a heading component to the cursor indicating direction and magnitude of movement. It could be added to the program as an extra feature that could be enabled in the option menu.

4.8 Uniform Scoring Scale

For an experienced user, the number that is most indicative of final score is the perimeter of the shape being copied. Good scores tend to fall in the 100 * perimeter range. Keeping this in mind, it might be a good idea to scale the scoring and/or final score by dividing by the perimeter. This action would make high scores a bit more uniform thus reinforcing the goal of using a scoring feature to as additional incentive to use the program.

Related Material

Lauren J. Bricker: *Cooperatively Controlled Objects in Support of Collaborative Learning*. Ph.D. dissertation, Dept. of Computer Science and Engineering, University of Washington, Seattle, WA. Spring 1998.

INFACT Documentation Index: http://kirsch.cs.washington.edu/infact_docs/

Multi-Control API Developer README:
http://kirsch.cs.washington.edu/infact_docs/infact_multi/dev_readme.txt

Multi-Control API Javadocs:
http://kirsch.cs.washington.edu/infact_docs/infact_multi/javadoc/

Jeode Runtime Documentation: http://www.cs.unc.edu/~lindsey/7ds/notes/jeode/

Acknowledgements

I would like to thank Steve Tanimoto for advising me through the duration of this project and contributing many of the ideas that have manifested themselves in MultiDraw. Thanks to Eric Bessette for the API that gave me a structure to start from, and thanks to Nick Benson who has consistently provided me with advice and examples when it comes to formatting official presentations.