Name & RCSID: ___________________________

ECSE-4750 Computer Graphics, spring 2000

Midterm Exam

February 24, 2000

Exam Rules:

1. You may have one 2-sided 8.5x11 inch note sheet, which may be mechanically printed. Keep your crib-sheet since you can use it again on the final.
2. You may have your blank paper, calculator, pens, etc.
3. You may not communicate with anyone, except Giampiero or me.
4. Answer all questions. Brief, concise, answers are preferred.
5. Spend time on a question proportional to its number of points.
6. Note that the last page is number *midterm–8*.
7. Start immediately. You have until 1:50.
8. Try to write legibly.
9. Write your NAME on top of this page.
10. Try to write your answers on these question sheets, extra paper is allowed. If an answer is on an extra sheet, say so in the normal space on this sheet.
11. Leave the small oval boxes blank; they're for our grade.
12. *Warning:* Be careful of questions that appear to be identical to ones that you've seen before, something might have changed.

Exam

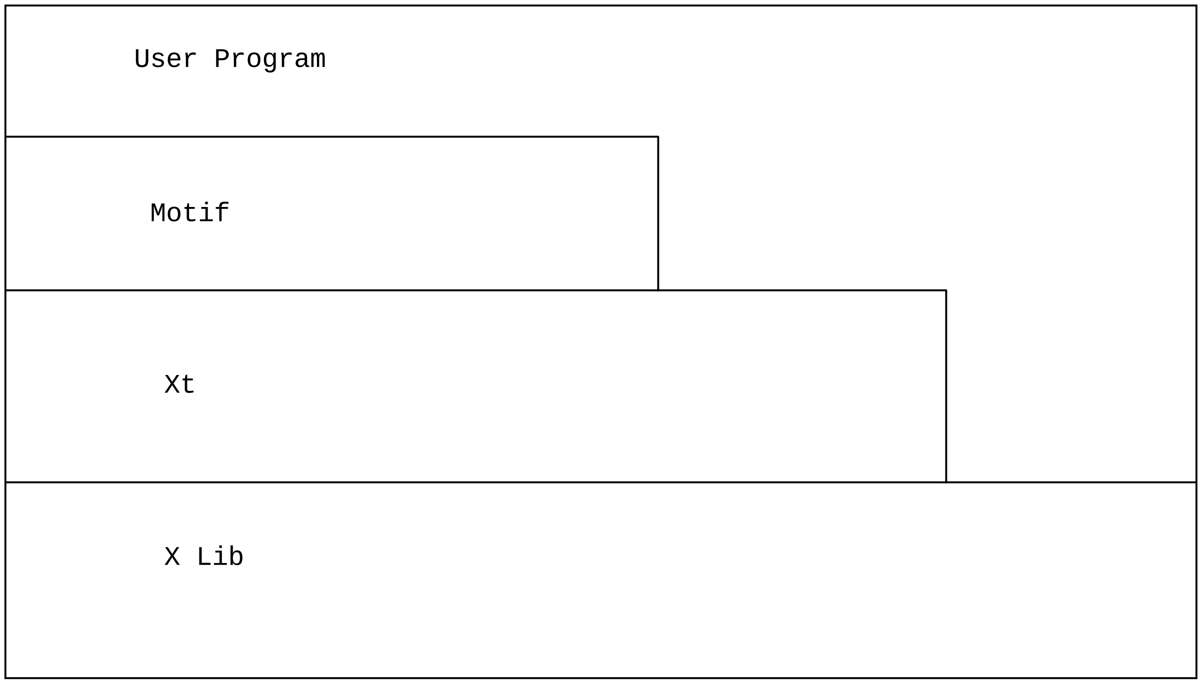
1. `[4] (points)` Name four things that `XtVaAppInitialize` does.

```
- Initializes application context
- Makes connection to X Server
- Parses application resources.
- Creates a toplevel widget
- Parses and removes X specific command line args
-
```

2. `[3]` Name 3 places where resources can be specified.

```
- Source code
- In the application .ad resource file
- In the X resource database read in via xrdb
- On the command line.
```

3. `[4]` In the following diagram, fill in the blanks with the following labels: *motif*, *xlib*, *xt*, and *user progra*m.



4. Consider the following X program:

```
#include<stdio.h>
#include <Xm/Xm.h>
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>

char greek[4] = {"plato","socrates","alex","philip"};

Widget widgets[4], quit, topLevel;
void
Proc(Widget w, caddr t client data, caddr t call data)
  Position x, y;
  int i;
  i=(int) client data;
  printf ("Client data= %d\n", i);
  XtVaGetValues(topLevel, XmNx, &x, XmNy, &y, NULL);
  switch (i) {
    case 0:
      x = 100;
    case 1:
      x += 100;
      break;
    case 2:
      y = 100;
    case 3:
      y += 100;
      break;
  }
  XtVaSetValues(topLevel, XmNx, x, XmNy, y, NULL);
}

/*quit button callback function*/
void
Quit(w, client data, call data)
Widget w;
XtPointer client_data, call-data;
{
}

main(int argc, char **argv)
  XtAppContext app context;
  Widget box;
  int i;
  topLevel = XtVaAppInitialize(&app, "XExam",
               NULL, 0, &argc, argv, NULL,  NULL);

  box = XtVaCreateManagedWidget("box",
         xmRowColumnWidgetClass, topLevel,  NULL);
  for (i = 0; i < 4; i++)
  widgets[i] = XtVaCreateManagedWidget(greek[i],
               xmPushButtonWidgetClass, box, NULL);
  quit = XtVaCreateManagedWidget("quit",
             xmPushButtonWidgetClass, box, NULL);
  XtAddCallback(quit, XmNactivateCallback, Quit, 0);
  for (i = 0; i < 4; i++)
    XtAddCallback(widgets[i], XmNactivateCallback,
                (XtPointer) Proc, (XtPointer) i);
  XtRealizeWidget(topLevel);
  XtAppMainLoop(app context);
}
```

(a) `[4]` What does this program do? Be specific about the action of each button.

```
- 5 Pushbuttons
- 4 are labeled with names
- 1 is labeled quit but does nothing
- plato sets the window's x to 100
- socrates adds 100 to the windows x
- alex sets the y to 100
- philip adds 100 to the windows y
```

(b) `[2]` Sketch the tree of widgets from the top level widget to the leaves.

```
- toplevel ---  box --- - plato
                        - socrates
                        - alex
                        - philip
                        - quit
```

(c) `[2]` What are the labels that will print in the command widgets by default?

- names listed above

(d) `[1]` What file in your account will resources be read from (assuming that things are configured as I've described in class)?

`- XExam.ad`

(e) `[1]` Suppose you want to change the label printed inside (only) the first leaf widget. Give a possible line to add to the resources file.

`- XExam.plato.labelString: Some Text Value`

5. [1] How can your X program tell the user ran it with command-line arguments that are not X resources and values?

```
- On the return from XtVaAppInitialize the argc variable is not zero.
```

6. [1] In `xtext.c`, what does the following code accomplish:

```
XtUnmanageChild(XmMessageBoxGetChild(help_widget, XmDIALOG_CANCEL_BUTTON));
```

```
- Removes the cancel button from a dialog.
```

7. [1] What is the purpose of a cascade button on a menu bar?

```
- Its what actually shows the menu
```

8. [1] Why does X use special data types, like `Point`, instead of just saying `struct { int x,y }`?

```
- Its more readable
- The values are type cast and compiler yaps when things are not quite right
- The actual storage types can change and we don’t need to know.
```

9. [1] According to the *Motif Programming Guide*, chapter 6, “The windows associated with Pop-up Menus and PulldownMenus are top-level windows. That is, the parent window of such a menu is the root window of the screen, not the window associated with the parent widget.” What advantage is there to this?

```
- It can manage itself and display anywhere on the screen, in case its too long for
the display
- You can create “Tear away” menus.
```

10. [3] Three types of manager widgets are *rowColum*n, *for*m, and *bulletinBoar*d. Distinguish between them.

```
- rowColumn is a container widget that forces child widgets into rows and columns
- The form widget is a free form widget allowing placement using constraints.
- The bulletin board widget can split up its client area and allow placement using
x,y or i,j
```

11. Window managers:
(a) [2] Name 2 things that a window manager does.

```
- Positions Windows on the screen
- Shows window decorations: title bar, resize borders, icons
- Handles input focus
- Can supply virtual desktops.
```

(b) [2] If you kill the window manager, what changes in the appearance on the screen? What functionality do you lose?

```
- You loose the title bar and resize/movement controls
- All windows are shown on the screen
- No ability to pull a window to the “top”
```

(c) [1] Name any one window manager in general use on RCS.

- MWM, TWM, TVWM, FVWM, OLWM, 4DWM, yadda, yadda, yadda

12. `[1]` Consider this code, which creates two widgets.

```
Widget but;
but = XtVaCreateManagedWidget("but",xmPushButtonWidgetClass,top_level,NULL);
but = XtVaCreateManagedWidget("but2",xmLabelWidgetClass,top_level,NULL);
*but.labelString: jambo!
```

Does this affect the button widget, the label widget, both, or neither?

`- It changes the label of the push button widget`

13. `[6]` Explain the corresponding function in Tcl/Tk to perform the following X Functions:

(a) `XtVaAppInitialize`.

`- Starting Tk (wish) creates a toplevel widget for us and initializes X`

(b) `XtAddCallback`.

`- The –command option for widget creation binds the comand to the widget`

(c) `XtVaCreateManagedWidget`.

`- Any widget creation statement (button, text, etc.) which creates a Tk Widget`

(d) `XtRealizeWidget`.

`- Closest here is pack`

(e) `XtAppMainloop`.

`- Its implicit to the interpreter when the end of the script is read`

(f) `Translations: #override  <Key>F2: string("ls -FC") string(0x0d)`

`- Use the Tcl/Tk bind command to bind an event to a command`

14. Look at the following Tcl/Tk program.

```
#! /home/wrf/bin/wish -f
proc getdate { } {
  global date
  set datef [open "|date"]
  set date [read $datef]
  close $datef
}
getdate
button .hello -textvariable date –fg red
        -command getdate
pack .hello
```

(a) `[1]` What does `-textvariable` do?

- Link the variable to the value of the button

(b) `[1]` What does `-command` do?

- Adds the associated command as a Callback to the button

(c) `[1]` What does the program as a whole do?

- Displays one button that is labeled with the start time, when pressed it resets the buttons label to the current time.

15. Motif vs. Tcl/Tk

(a) [2] Describe the different coding processes for X/Motif and Tcl/Tk. What are the steps for each and how do they differ?

- Motif is a traditional coding style. First you write, then compile, then link, and finally test.
- Tcl/Tk is a rapid proto-typing process. You write the code, and then run/test it.

(a) [4] List at least two strengths and two weaknesses each for Motif and Tcl/Tk.

```
Motif
  Strength:
      Fast execution, very flexible, Mature interface, highly tested, cross platform
      (UNIX)
  Weakness:
      Longer development cycle, steep learning curve, costs $, separate from OS,
      cross platform (Windows/MAC?)

Tcl/Tk
  Strength:
      High level language, can do a lot in a little code, easy to learn, short
      development cycle, cross platform
  Weakness:
      Limiting in its functionality, slow execution, memory intensive,
```

Total: 50 points

End of exam