

This is not one exam! It is a collection of all the questions that I've asked on midterm exams over the last several years. Some questions refer to material that is not in this year's course.

ECSE-4750 Computer Graphics, Spring 1998
Handout bigmidterm*
Midterm Exam

October 17, 1999

Exam Rules

1. You may have one 2-sided 8.5 inch x11 inch note sheet, which may be mechanically printed. Keep your cribsheet 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 Sutha 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 bigmidterm-35.
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, tho 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.

*Copyright 1996-8, Wm. Randolph Franklin. You may copy this for nonprofit research and teaching purposes under these conditions: (1) You do not change the document. (2) You do not charge people to look at or to copy your copy.

Exam

1. /[4] (points) Name four things that `XtVaAppInitialize` does.
2. Tell me about classes of X programs.
 - (a) /[1] (point) Where is a program's class set?
 - (b) /[1] What is the biggest use of the class?
 - (c) /[1] What's the point of classes in general; i.e., why not just use program names?
3. /[3] Name 3 X resources that may be specified on the command line when running a program.
4. /[3] Name 3 places where resources can be specified.
5. /[4] In the following diagram, fill in the blanks with the following labels: *motif*, *xlib*, *xt*, *user program*.
6. /[1] A widget variable in a program is a pointer to a structure, whose components are documented. Why is it bad practice to access these components directly, instead of via things like `XtName`?
7. /[40] Here is an alphabetical list of names that occur in X, including routines, macros, and so on. Use them to answer the question starting on bigmidterm-4.


- | | |
|--------------------------|---------------------------------|
| (1) argc | (44) XDefaultScreen |
| (2) argv | (45) XDisplayHeight |
| (3) Buttonpress | (46) XDisplayHeightMM |
| (4) ButtonPressMask | (47) XDrawLine |
| (5) callback | (48) XDrawString |
| (6) CancelButtonCallback | (49) XEvent |
| (7) CapRound | (50) XExam |
| (8) cascade button | (51) XExposeEvent |
| (9) Class | (52) XFarewell |
| (10) ClearButtonCallBack | (53) XFillArc |
| (11) Coloredit | (54) XFillPolygon |
| (12) colormap | (55) XKeyEvent |
| (13) colorpanel | (56) Xlib |
| (14) Colors | (57) XLookupString |
| (15) DefaultRootWindow | (58) Xm |
| (16) Draw | (59) XMapRaised |
| (17) editres | (60) XmATTACH |
| (18) Event | (61) XmBulletinBoard |
| (19) EventMask | (62) xmCascadeButtonWidgetClass |
| (20) Expose | (63) XmCreateInformationDialog |
| (21) ExposureMask | (64) XmCreateMenuBar |
| (22) GC | (65) XmCreatePromptDialog |
| (23) JoinBevel | (66) XmCreatePulldownMenu |
| (24) Keypress | (67) XmCreatePushButton |
| (25) I18N | (68) XmDIALOG |
| (26) LineDoubleDash | (69) XmFONTLIST |
| (27) Manage | (70) xmFormWidgetClass |
| (28) Map | (71) XmLabel |
| (29) OkButtonCallback | (72) xmLabelWidgetClass |
| (30) OSF | (73) XmMainWindowSetAreas |
| (31) PrintButtonCallBack | (74) xmMainWindowWidgetClass |
| (32) PushButton | (75) XmMessageBoxGetChild |
| (33) Redraw | (76) XmNactivateCallback |
| (34) RowColumn | (77) XmNcancelCallback |
| (35) Widget | (78) XmNdragCallback |
| (36) XAllocColor | (79) XmNeditMode |
| (37) XBlackPixel | (80) XmNheight |
| (38) XButtonEvent | (81) XmNlabelString |
| (39) XColor | (82) XmNleftAttachment |
| (40) XCreateColormap | (83) XmNmaximum |
| (41) XCreateGC | (84) XmNmenuHelpWidget |
| (42) XCreateSimpleWindow | (85) XmNminimum |
| (43) XDefaultColormap | (86) XmNokCallback |
| | (87) XmNsubMenuId |

(88) XmNtextString	(123) XtActionProc
(89) XmNvalue	(124) XtActionsRec
(90) XmNvalueChangedCallback	(125) XtAddCallback
(91) XmNwidth	(126) XtAddEventHandler
(92) XmNwordWrap	(127) XtAppAddActions
(93) XmNx	(128) XtAppContext
(94) XmNy	(129) XtAppMainLoop
(95) XMove	(130) XtCreateManagedWidget
(96) xmRowColumnWidgetClass	(131) XtCreateWidget
(97) XmScale	(132) XtDestroyWidget
(98) XmScaleCallbackStruct	(133) XtDisplay
(99) xmScaleWidgetClass	(134) XText
(100) XmSelectionBoxCallbackStruct	(135) XtFree
(101) xmSeparatorWidgetClass	(136) XtGrabNone
(102) XmString	(137) XtMalloc
(103) XmStringConcat	(138) XtManageChild
(104) XmStringCreate	(139) XtName
(105) XmStringCreateSimple	(140) XtNbackground
(106) XmStringFree	(141) XtNumber
(107) XmTextGetString	(142) XtOverrideTranslations
(108) XmTextSetString	(143) XtParent
(109) xmTextWidgetClass	(144) XtParseTranslationTable
(110) XNextEvent	(145) XtPointer
(111) XOpenDisplay	(146) XtPopup
(112) XPoint	(147) XtRealizeWidget
(113) XQueryColor	(148) XtSetArg
(114) XRootWindow	(149) XtSetValues
(115) XSelectInput	(150) XtUnmanageChild
(116) XServerVendor	(151) XtVaAppInitialize
(117) XSetBackground	(152) XtVaCreateManagedWidget
(118) XSetForeground	(153) XtVaGetValues
(119) XSetLineAttributes	(154) XtVaSetValues
(120) XSetWindowColormap	(155) XtWindow
(121) XStoreColor	(156) XWhitePixel
(122) Xt	

Here is a list of X-related operations to perform, resources, data types, concepts, etc. Before each item, write down the item NUMBER in the above list that is most applicable.

- (1) _____ A button designed for calling up a pulldown menu.
- (2) _____ A data type containing graphics properties of a line, such as color, width, and style.
- (3) _____ A data type of a string of text including font info.
- (4) _____ A data type whose components are x and y .
- (5) _____ A particular program that queries and changes another program's resources.

- (6) _____ A resource name to cause two widgets to be positioned side-by-side.
- (7) _____ A resource to tie a pulldown menu to the button that will make it appear.
- (8) _____ A struct containing components like red, green, blue, and pixel.
- (9) _____ A widget designed to take several children, which it will lay out itself without any more user control.
- (10) _____ Converts a Motif string into a C string.
- (11) _____ Copies a C variable into a widget's resource value.
- (12) _____ Copies a widget's resource value into a C variable.
- (13) _____ Creates a popup convenience widget to ask the user for a small amount of input.
- (14) _____ Creates a widget and tells its parent to determine its size and position.
- (15) _____ Creates a window for a widget that doesn't yet have one.
- (16) _____ Creates the top level widget.
- (17) _____ Deletes a widget.
- (18) _____ Determines a widget's size and location, and does the same for other widgets that have this widget as a parent.
- (19) _____ Draws a line.
- (20) _____ Draws text at the Xlib level.
- (21) _____ Find the row of the color map containing the color white.
- (22) _____ For a graphics context, sets the color to draw a solid line in.
- (23) _____ Find the address of the ok button in a dialog widget.
- (24) _____ Frees a Motif string that is no longer needed.
- (25) _____ Gets a widget's name.
- (26) _____ Gives a user routine to call when a button is activated.
- (27) _____ Installs a desired color into the color map.
- (28) _____ Internationalization.
- (29) _____ Loops while waiting for X events.
- (30) _____ Processes X arguments on the command line.
- (31) _____ Reads the resource file.
- (32) _____ Returns the display associated with a widget.
- (33) _____ Returns the number of pixels from top to bottom on your display.
- (34) _____ Returns the widget whose child this widget is.
- (35) _____ Returns the window associated with a widget.
- (36) _____ Specifies what classes of events to accept.
- (37) _____ The industrial consortium behind X.
- (38) _____ The number of arguments that an X program was run with.
- (39) _____ Tries to connect to the X server.
- (40) _____ Waits for the next user input or system status change, like a keypress or window expose.

8.  [8] Here are some lines from `xfarewell.c`. Unfortunately, I accidentally ran them thru the sort program, so that they are now in alphabetical order. Please place them in the proper order, by writing the proper number of each line, 1,2,3, or whatever, before that line. I've numbered the first three lines.

- (a) 1
`#include <Xm/PushB.h>`
- (b) 2
`#include <Xm/Xm.h>`
- (c) 3
`#include <stdio.h>`
- (d) _____
`XtAppAddActions (app_context, my_actions, XtNumber (my_actions));`
- (e) _____
`XtAppContext app_context;`
- (f) _____
`XtAppMainLoop (app_context);}`
- (g) _____
`XtRealizeWidget (top_level);`
- (h) _____
`exit (0);}`
- (i) _____
`farewell_widget =`
`XtVaCreateManagedWidget ("farewell", xmPushButtonWidgetClass,`
`top_level, NULL);`
- (j) _____
`main (int argc, char **argv)`
- (k) _____
`static XtActionsRec my_actions[] = {{"confirm", (XtActionProc) Con-`
`firm},`
`{"quit", (XtActionProc) Quit}};`
- (l) _____
`top_level = XtVaAppInitialize (&app_context, "XFarewell", NULL, 0,`
`&argc, argv, NULL, NULL);`
- (m) _____
`void Confirm (Widget w, XButtonEvent * event, String * params,`
`Cardinal * num_params)`

(n) _____

```
void Quit (Widget w, XButtonEvent * event, String * params,
          Cardinal * num_params)
```

(o) _____

```
{Widget top_level, farewell_widget;
```

(p) _____

```
{fprintf (stderr, "Are you sure you want to exit?\n\
Click with the middle pointer button if you're sure.\n");}
```

(q) _____

```
{fprintf (stderr, "It was nice knowing you.\n");
```

9. ☐ [8] Here are some lines from xgoodbye.c. Unfortunately, I accidentally ran them thru the sort program, so that they are now in alphabetical order. Please place them in the proper order, by writing the proper number of each line, 1,2,3, or whatever, before that line. Make variables local, not global, if possible.

(a) _____

```
#include <Xm/Xm.h>
```

(b) _____

```
goodbye_widget = XtVaCreateManagedWidget("goodbye",
xmPushButtonWidgetClass, top_level_widget, NULL);
```

(c) _____

```
top_level_widget = XtVaAppInitialize(&app, "XGoodbye", NULL, 0,
&argc, argv, NULL, NULL);
```

(d) _____

```
void main(int argc, char **argv)
```

(e) _____

```
Widget top_level_widget, goodbye_widget;
```

(f) _____

```
XtAddCallback(goodbye_widget, XmNactivateCallback, Quit, 0);
```

(g) _____


```
XtAppMainLoop(app);
```

(h) _____

```
XtRealizeWidget(top_level_widget);
```

10. ☐ [2] Altho Vtcl usually makes laying out a GUI much easier, there is one program that we saw that would be much harder to do with Vtcl, than in Motif or Tcl/Tk. The reason is that in Vtcl, you have lay out the widgets individually. Circle that program, from the following list, which might extend to the next page if this question is at the bottom of the page.

- (a) xbox.c
- (b) xdraw1.c
- (c) xdraw2.c
- (d) xfarewell.c
- (e) xgoodbye.c
- (f) xhello.c
- (g) xlots.c
- (h) xmove.c
- (i) xtext.c

11. /[8] Here is a list of structs that X uses. Following, is a list of names. For each name, write the number of the struct that this applies to. You are not expected ever to have seen the struct contents before. The idea is to use your knowledge of what each data type does to find a struct whose components look appropriate.

- (a)

```
typedef struct {
    short x, y;
}
```
- (b)

```
typedef struct {
    unsigned long pixel;
    unsigned short red, green, blue;
    char flags; /* do_red, do_green, do_blue */
    char pad;
}
```
- (c)

```
typedef struct {
    int type; /* of event */
    unsigned long serial; /* # of last request processed by server */
    Bool send_event; /* true if this came from a SendEvent request */
    Display *display; /* Display the event was read from */
    Window window; /* "event" window it is reported relative to */
    Window root; /* root window that the event occurred on */
    Window subwindow; /* child window */
    Time time; /* milliseconds */
    int x, y; /* pointer x, y coordinates in event window */
    int x_root, y_root; /* coordinates relative to root */
    unsigned int state; /* key or button mask */
    unsigned int button; /* detail */
    Bool same_screen; /* same screen flag */
}
```
- (d)

```
typedef struct {
    int function; /* logical operation */
    unsigned long plane_mask; /* plane mask */
    unsigned long foreground; /* foreground pixel */
    unsigned long background; /* background pixel */
    int line_width; /* line width */
    int line_style; /* LineSolid, LineOnOffDash, LineDoubleDash */
}
```




```

int cap_style;      /* CapNotLast, CapButt,
                    CapRound, CapProjecting */
int join_style;     /* JoinMiter, JoinRound, JoinBevel */
int fill_style;     /* FillSolid, FillTiled,
                    FillStippled, FillOpaqueStippled */
int fill_rule;      /* EvenOddRule, WindingRule */
int arc_mode; /* ArcChord, ArcPieSlice */
Pixmap tile; /* tile pixmap for tiling operations */
Pixmap stipple; /* stipple 1 plane pixmap for stippling */
int ts_x_origin; /* offset for tile or stipple operations */
int ts_y_origin;
    Font font;      /* default text font for text operations */
int subwindow_mode; /* ClipByChildren, IncludeInferiors */
Bool graphics_exposures; /* boolean, should exposures be generated */
int clip_x_origin; /* origin for clipping */
int clip_y_origin;
Pixmap clip_mask; /* bitmap clipping; other calls for rects */
int dash_offset; /* patterned/dashed line information */
char dashes;
}

```

- (a) _____GC
- (b) _____XButtonEvent
- (c) _____XColor
- (d) _____XPoint

12.  [5] For each of the following properties, decide whether it applies more to the basic C character strings, or to Motif compound strings. Write a **C** or an **M**, as appropriate, before each statement.

- (a) _____The data type is XmString.
- (b) _____The usual routine that catenates two of it modifies its first argument.
- (c) _____It makes GUIs that function in other languages easier to write.
- (d) _____The routines that manipulate it allocate their own storage.
- (e) _____The number of bytes that a variable of this type occupies is less likely to correspond to the number of characters in the string.

13. Consider the following X program:

```

/** xexam.c Midterm problem
Last modified: by Wm Randolph Franklin (wrf) on speed.ecse.rpi.edu at Wed Oct 19 16:00:08 1994
*/

#include <stdio.h>
#include <Xm/Xm.h>          /* Needed by all Motif programs. */
#include <Xm/Label.h>
#include <Xm/PushB.h>
#include <Xm/RowColumn.h>

```

10

```
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;
```

20

```
    i=(int) client_data;
```

```
    printf("Client data= %d\n", i);
```

```
    XtVaGetValues(topLevel, XmNx, &x, XmNy, &y, NULL);
```

```
    switch (i) {
```

```
    case 0:
```

```
        x -= 100;
```

```
        break;
```

30

```
    case 1:
```

```
        x += 100;
```

```
        break;
```

```
    case 2:
```

```
        y -= 100;
```

```
        break;
```

```
    case 3:
```

```
        y += 100;
```

```
        break;
```

40

```
    };
```

```
    XtVaSetValues(topLevel, XmNx, x, XmNy, y, NULL);
```

```
}
```

```
/*
```

```
 * quit button callback function
```

```
*/
```

```
void
```

```
Quit(w, client_data, call_data)
```

50

```
    Widget  w;
```

```
    caddr_t client_data, call_data;
```

```
{
```

```
    exit(0);
```

```
}
```

```
main(int argc, char **argv)
```

```
{
```

```

XtAppContext app_context;
Widget box;
int i;

topLevel = XtVaAppInitialize(&app_context, "XExam",
                                NULL, /* options */
                                0, /* num_options */
                                &argc, /* num cmd line */
                                argv, /* cmd line */
                                NULL, /* fallback */
                                NULL /* args */
                                );

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.
- (b) /[2] Sketch the tree of widgets from the top level widget to the leaves.
- (c) /[2] How many push button widgets are there? Which routine is called when each is clicked on?

- (d) /[2] What are the labels that will print in the command widgets by default?
- (e) /[1] What file where in your account will resources be read from (assuming that things are configured as I've described in class)?
- (f) /[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.
- (g) /[1] Suppose you run the program thus: `xexam -bg red`. Where, if anywhere, in your program is the `bg` resource recognized?

14. Do the preceding question with the following program.

```

/** xexam.c Midterm problem
Time-stamp: </mab_home1/wrf/cg/xexam2.c, Tue, 23 Feb 1999, 19:46:33 EST, wrfmab.ecse.rpi.edu>
*/

```

```

#include <stdio.h>
#include <Xm/Xm.h>                /* Needed by all Motif programs. */
#include <Xm/Label.h>
#include <Xm/PushButton.h>
#include <Xm/RowColumn.h>

```

10

```

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;

```

20

```

    i=(int) client_data;

```

```

    printf("Client data= %d\n", i);

```

```

    XtVaGetValues(topLevel, XmNx, &x, XmNy, &y, NULL);

```

```

    switch (i) {

```

```

    case 0:

```

```

        x -= 100;

```

30

```

    case 1:

```

```

        x += 100;

```

```

        break;

```

```

    case 2:

```

```

        y -= 100;
    case 3:
        y += 100;
        break;
    };
    XtVaSetValues(topLevel, XmNx, x, XmNy, y, NULL);
}
40

/*
 * quit button callback function
 */
void
Quit(w, client_data, call_data)
    Widget w;
    caddr_t client_data, call_data;
50
{
}

main(int argc, char **argv)
{
    XtAppContext app_context;
    Widget box;
    int i;
60

    topLevel = XtVaAppInitialize(&app_context, "XExam",
                                NULL, /* options */
                                0, /* num_options */
                                &argc, /* num cmd line */
                                argv, /* cmd line */
                                NULL, /* fallback */
                                NULL /* args */
                                );
70

    box = XtVaCreateManagedWidget("box",
                                xmRowColumnWidgetClass, /* class */
                                topLevel, /* parent */
                                NULL); /* args */

    for (i = 0; i < 4; i++) {
        widgets[i] = XtVaCreateManagedWidget(greek[i],
                                xmPushButtonWidgetClass,
                                box, NULL);
80
    }

    quit = XtVaCreateManagedWidget(
                                "quit", /* 12 */
                                xmPushButtonWidgetClass, /* 13 */

```

```

        box,      /* 14 */
        NULL);   /* 15 */

```

```

XtAddCallback(quit, XmNactivateCallback, Quit, 0);

```

```

for (i = 0; i < 4; i++)

```

90

```

    XtAddCallback(widgets[i], XmNactivateCallback, (XtPointer) Proc, (XtPointer) i);

```

```

XtRealizeWidget(topLevel);

```

```

XtAppMainLoop(app_context);

```

```

}

```

15. ☐ [1] How can your X program tell the user ran it with command-line arguments that are not X resources and values?

16. ☐ [1] In `xgoodbye.c`, why did I place the `Quit` routine before *main*? For reference, here is part of that program, w/o comments.

```

void Quit(w, client_data, call_data)
    Widget w;
    XtPointer client_data, call_data;
{
    printf("It was nice knowing you.\n");
    exit(0);
}

void main(int argc, char **argv)
{
    Widget top_level, goodbye_widget;
    XtAppContext app;
    top_level = XtVaAppInitialize(&app, "XGoodbye", NULL, 0, &argc, argv, NULL,
    NULL);
    goodbye_widget = XtVaCreateManagedWidget("goodbye", xmPushButtonWidgetClass,
    top_level, NULL);
    XtAddCallback(goodbye_widget, XmNactivateCallback, Quit, 0);
    XtRealizeWidget(top_level);
    XtAppMainLoop(app);
}

```

17. ☐ [1] In `xlots.c`, what are the names of the widgets created with the following code:

```

for (i = 0; i < N; i++)
{
    sprintf (s, "%d", i);
    w[i] = XtVaCreateManagedWidget (s, xmLabelWidgetClass, bb, NULL);
}

```

18. ☐ [2] What are the names of the widgets created with the following code:

```

char *a[]={ "Sun", "DEC", "IBM", "MS" };
char s[100];
char *t;

for (i = 0; i < N; i++)
{
    sprintf (s, "%d", i);
    t=strcat(s,a[i]);
    w[i] = XtVaCreateManagedWidget (s, xmLabelWidgetClass, bb, NULL);
}

```

19. ☐ [1] Program `xbox.c` has the following code:

```

/* PopupDialog:  callback for the PRESSME button. */
void PopupDialog(w, client_data, call_data)
    Widget w;
    XtPointer client_data, call_data;
{
    Widget parent;
    static Widget dialogbox = (Widget) NULL;
    parent = (Widget) client_data;

    if (!dialogbox) { /* Create this only the first time thru. */
        dialogbox = XmCreatePromptDialog(parent, "dialogbox", NULL, 0);
    }
    XtManageChild(dialogbox);
}

```

How is it that the `if` test creates the dialogbox only the first time thru?

20. ☐ [2] There is something unusual about the identity of the widget returned when you call `XmCreatePromptDialog`, in contrast to most of the other widget creation routines. What?

21. ☐ [2] In `xfarewell.c`, what is the following doing?

```

static XtActionsRec my_actions[] =
{
    {"confirm", (XtActionProc) Confirm},
    {"quit", (XtActionProc) Quit},
};
...
XtAppAddActions (app_context, my_actions, XtNumber (my_actions));

```

22. ☐ [1] Various programs have had code like the following to set a widget's label:

```

s2=XmStringCreateSimple("newlabel");
XtVaSetValues (but, XmNlabelString, s2, NULL);

```

Since I can set a widget's label in a resource file w/o going thru this mess, why can't I just say this in the program?

```
XtVaSetValues (but, XmNlabelString, "newlabel", NULL);
```

23. /[1] In `xdraw2.c`, why did I realize the top level widget before creating the graphics contexts?

24. /[1] In `xtext.c`, what does the following code accomplish:

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

25. /[1] What is the purpose of a cascade button on a menu bar?

26. /[3] What is a resource in X? Name 4 resources.

27. /[2] What does it mean to *manage* a widget?

28. /[2] When you call `XtMainLoop()` your program loses control to the X system. Name two ways for it to get back control later (without terminating the X system)?

29. (a) /[1] Why might you want to have the same callback routine for more than one widget?

(b) /[1] In the above case, name one way for the routine tell which widget it is responding to?

30. /[2] You can pass one variable to the callback routine, in the client data field. Suppose that you wish to make two variables used in the main program to be available in the callback routine. Give two different ways to do it.

31. /[1] When does a widget not have a window associated with it?

32. /[4] For each of the following routines, say whether it is an *Xlib* routine, an *Xt* routine, or a *Motif* routine:

- (a) `XtVaAppinitialize`
- (b) `XmStringCreateSimple`
- (c) `XtVaSetValues`
- (d) `XFillArc`

33. Graphics contexts:

- (a) /[1] How does a graphics context lower load on the net?
- (b) /[2] Name 3 attributes stored in a GC.
34. /[2] One Motif design principle is called *I18N*. What does this mean? Name one feature that we've seen that supports this.
35. /[2] What's wrong with this C program:
- ```
char command[4]="ls ";
char arg[5]="/tmp";
char line[100];

line=strcat(command,arg);
system(line);
```
36. /[1] Why is it advantageous for the several clients on one graphics server to use the same colors, instead of each using slightly different colors?
37. /[1] Suppose that one program wants to use 250 colors on its own. How might it do something with color maps to lessen the impact on others clients?
38. /[1] It's clear why 8-bit frame buffers use color maps (i.e., to select which 256 colors of the  $2^{24}$  possible ones to use). However, some 24-bit buffers also use color maps. (It's not one  $2^{24}$  entry cmap, but 3 256-entry cmaps, one per primary color.) Why might they want to do this since it does make things more complicated?
39. Convenience widgets:
- (a) /[1] What are they?
- (b) /[1] Name one.
40. /[1] Suppose that you create an information dialog widget, but don't want the cancel button to appear. How can you effect this?
41. /[2] How does a program know which file to read to get the resources? Give 2 steps that it goes thru.
42. /[1] What does `XtAppMainLoop` do?

43. /[2] What does `XtRealizeWidget` do? It really does several things; I'll take any one.
44. /[1] Why does X use special data types, like `Point`, instead of just saying `struct { int x,y }`?
45. /[1] Why does X use special data types, like `Position`, instead of just saying `int`?
46. /[4] Give two advantages and two disadvantages of Motif compared to the Athena widgets.
47. /[2] In your C program, Motif uses the preprocessor symbol `XmNx` to represent the resource that would be named `x` in your resource file. How does this help to catch typos in your program?

48. Motif uses compound text strings.

(a) /[1] What are they?

(b) /[2] Give 2 advantages of them. The following is not acceptable.

"It's harder to debug compound strings, so you use more CPU time. Motif comes from the OSF, of which IBM is a partner. So, compound strings help IBM to sell more computers."

49. The resource file for `charset.c` said this:

```
*pb1.fontList: *-courier*-r*--12-=charset
*pb2.fontList: *-courier*-r*--14-=charset
*pb3.fontList: *-courier*-r*--18-=charset
```

(a) /[2] What is this `charset` used for and where in the program might I refer to it?

(b) /[1] Why is it not a problem for the three widgets all to use the same `charset`? That is, won't the last use of `charset` override the previous two?

50. /[2] In `xcharset.c`, how can the 3 widgets' different resources cause their labels to have different type fonts, when labels are exactly the same compound char string? FYI, here is the relevant part of the program:

```
XmString text;

toplevel = XtVaAppInitialize(&app, "XCharset", NULL, 0,
 &argc, argv, fallbacks, NULL);
text = XmStringCreateSimple("Testing, testing...");
rowcol_widget = XtVaCreateManagedWidget("rowcol",
```

```

 xmRowColumnWidgetClass, toplevel, NULL);
XtVaCreateManagedWidget("pb1", xmPushButtonGadgetClass, rowcol_widget,
 XmNlabelString, text, NULL);
XtVaCreateManagedWidget("pb2", xmPushButtonGadgetClass, rowcol_widget,
 XmNlabelString, text, NULL);
XtVaCreateManagedWidget("pb3", xmPushButtonGadgetClass, rowcol_widget,
 XmNlabelString, text, NULL);

```

and the resource file:

```

*pb1.fontList: *-courier*-r*--12-=charset
*pb2.fontList: *-courier*-r*--14-=charset
*pb3.fontList: *-courier*-r*--18-=charset

```

51.  [1] Name one programming error wrt compound strings that can lead to a storage leak.
52.  [1] One compound string can have several fonts. Different X servers can have different fonts available. This would seem to suggest that there might be a problem with a compound string referring to a font that doesn't exist on some servers. Name one feature in the compound string format that ameliorates this possible problem.
53.  [1] What is a *font server*?
54.  [1] Compound strings usually should be freed when you are finished with them. What happens if you don't?
55.  [3] Name 3 differences between standard C character strings and Motif compound text strings.
56.  [2] Suppose that we wish to put 3 label widgets and 5 command widgets in a popup. How can we do this, given that a transient shell widget can have only one child?
57.  [2] According to the *Motif Programming Guide*, chapter 6, "The windows associated with Pop-upMenus 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?
58.  [2] Consider the following widget definition:

```
jambo=XtVaCreateManagedWidget("gday", labelWidgetClass, box, NULL);
```

and the following fragment from a resource file:

```
*jambo.command: Yes
*jambo.labelString: No
*gday.command: Maybe
*gday.labelstring: bon giorno
*end.label: Freeze!
*Command.labelString: Finish!
*labelString: Default.
*label: Itty Bitty Machines
```

What will be the label on this widget?

59. /[2] Consider the following widget definition in a program whose class is XExam

```
jambo=XtVaCreateManagedWidget("gday",labelWidgetClass,box,NULL);
```

and the following fragment from a resource file:

```
*jambo.command: Yes
*jambo.labelString: No
*gday.command: Maybe
*gday.labelstring: bon giorno
*end.label: Freeze!
*Command.labelString: Finish!
*labelString: Default
*label: Itty Bitty Machines
XExam*labelString: Forty two
```

What will be the label on this widget?

60. /[2] Sometimes you see a line such as this:

```
XtSetArg(arg[i], XtNwidth, &width); i++;
```

but sometimes it is written thus:

```
XtSetArg(arg[i], XtNwidth, width); i++;
```

Why is width passed by value sometimes and by address at others?

61. /[2] What is the difference between an event and an action?

62. Events fall into two broad categories.

- (a) /[2] What are they?

- (b) /[2] Give an example of each.

63. /[1] How do you establish the relation between an event and an action?
64. /[1] There are events that are user inputs, and then there are events that are system status changes. Name one of the latter.
65. /[3] Three types of manager widgets are *rowColumn*, *form*, and *bulletinBoard*. Distinguish between them.
66. /[1] In the mainwindow widget menu bar, what's the purpose of declaring one button to be the help button?
67. Window managers:
- (a) /[2] Name 2 things that a window manager does.
- (b) /[2] If you kill the window manager, what changes in the appearance on the screen? What functionality do you lose?
- (c) /[1] Name any one window manager in general use on RCS.
68. /[1] Resources are grouped into classes, just as programs are. E.g, foreground and bottomShadowColor are both class Foreground. However, the background resource is in a different class, Background. It would seem more logical to group all color-related resources into one class called Color. Why do think that the OSF did not do this?
69. /[2] While browsing the resources file for Netscape, I found this piece of a translation table.

```
*globalTranslations: #override \n\
 Meta ~Ctrl<Key>A: addBookmark() \n\
 Alt ~Ctrl<Key>A: addBookmark() \n\
 Meta ~Ctrl<Key>B: viewBookmark() \n\
 Alt ~Ctrl<Key>B: viewBookmark() \n\
 Meta ~Ctrl<Key>C: copy() \n\
 Alt ~Ctrl<Key>C: copy()
```

Which are the actions, and which the events here?

70. /[2] While browsing the resources file for xcalc, I found this piece of a translation table.

```
*hp.bevel.screen.LCD.Translations: #replace\n\
<Key>p:pi()\n\
<Key>i:inverse()\n\
*ti.button5.Translations: #override<Btn1Up>:off()unset()\n\
 <Btn3Up>:quit()
```

Which are the actions, and which the events here?

71. Look at this line from `xtext.c`.

```
XtVaSetValues(clear, XmNleftAttachment, XmATTACH_WIDGET,
 XmNleftWidget, print, NULL);
```

- (a) /[1] What does the Va mean as part of the routine name?
- (b) /[1] This routine claims that it is setting values. What sort of values. E.g., is it assigning C variables?
- (c) /[1] What is the approximate effect of this values setting?

72. /[1] How do you establish the relation between a routine in your program and an action?

73. The designers of X made some choices about the capabilities of a graphics server, and picked a middle road.

- (a) /[2] Name one capability that they included in the server, which they might have omitted. Give an advantage of including it.
- (b) /[2] Name one capability that they omitted, which they might have included. Give a disadvantage of omitting it.

74. /[1] Consider this code, which creates two widgets.

```
Widget but;
but = XtVaCreateManagedWidget("but",xmPushButtonWidgetClass,top_level,NULL);
but = XtVaCreateManagedWidget("but2",xmLabelWidgetClass,top_level,NULL);
```

and this extract from the resource file.

```
*but.labelString: jambo!
```

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

75. For certain technical reasons, a few resources cannot be specified in a resource file, but must be specified in the program. I'm not thinking of a reason like that it is undesirable to allow the user access to it. This resource just cannot be given a value in the resource file, even if you, the programmer, wanted to.

- (a) /[1] Name one such resource.
- (b) /[1] Why cannot it be specified in the resource file?

76. ☐/[1] Suppose that you create a widget, and then assign it to another variable:

```
Widget hello, hello2;
hello = XtVaCreateManagedWidget("hello2", xmLabelWidgetClass, top_level, NULL);
hello2= hello;
```

If you want to set a resource for the widget referenced by the C variable hello, should you do this:

```
*hello.labelString: jambo!
```

or this:

```
*hello2.labelString: jambo!
```

or does it not matter?

77. ☐/[1] Suppose that you create a widget, and then assign it to another variable:

```
Widget hello, hello2;
hello = XtVaCreateManagedWidget("hello", xmLabelWidgetClass, top_level, NULL);
hello2= hello;
```

If you want to set a resource for hello2, should you do this:

```
*hello.labelString: jambo!
```

or this:

```
*hello2.labelString: jambo!
```

or does it not matter?

78. ☐/[2] What's the difference between a rowcolumn widget, a paned widget, and a menubar widget?

79. ☐/[2] Xlib, at least as we've seen it, counts coordinates in pixels. How would you write a program to display the same on the different CRTs at RPI, which have 3 different resolutions? How would you get any information that you propose to use?

80. ☐/[1] If your screen is 1024x1024, and has 8 bits per pixel, how much memory does the frame buffer take?

81. Suppose that you want to write a calculator program to run on both the Suns, with  $900 \times 1152$ -resolution displays, and the X stations, with  $480 \times 640$ -resolution displays. You want the program with all its buttons to fill 1/2 of the screen in either case.

(a) /[2] Would it be easier to hold the buttons in a *form* widget or a *bulletinboard* widget? Why?

(b) /[1] Why should you not use a *rowcolumn* widget?

82. /[2] What's wrong with this? Fix it.

```
char *c;
c[0]='h';
c[1]='i';
c[2]='\0';
printf("The message of the day is: %s\n",c);
```

83. Suppose that Bill is running on `whitehouse.gov` and he wishes to start an `xeyes` program to display on Newt's personal workstation, `house.gov`.

(a) /[1] How does Bill tell `xeyes` to display on Newt's machine?

(b) /[1] How does Newt permit Bill's program to display on his machine?

(c) /[1] Which is the client and which the server?

84. /[2] What functionality does the `Xt` level add to the X window system?

85. /[2] Near the start of my programs, there are these lines:

```
#include <Xm/Xm.h> /* Needed by all Motif programs. */
#include <Xm/Label.h> /* This is for the Motif Label Widget */
```

When you compile the program, how does the compiler know where to look for the include files?

86. /[2] What is the difference between the client data and the call data in a callback routine?

87. /[1] Name one type of information that we saw passed in the `call_data` argument to a callback routine.

88. /[1] Why does X make it so complicated to use new colors in Xlib? Whether or not the MIT grad students who wrote much of X are sadistic is not relevant.



89. /[1] Consider the file `/campus/X11/R5/core/1.0/@sys/lib/X11/rgb.txt`, which maps from color names, like cyan, to values.

- (a) /[1] We see lines like this:

255 228 181 moccasin

What does 255 mean?

- (b) /[1] There's another line like this:

 Blue

Unfortunately the first half of the line is unreadable :-). Tell me what number(s) are hidden by the black box.

90. /[2] In `xdraw2.c`, there are expressions like this:

```
XAllocColor(display, cmap, &orange)
```

Why is `cmap` passed directly, but the address of `orange` is passed in?

91. /[1] Why might `XAllocColor` fail on a color display?

92. /[1] An X server has no protection or hiding between its various clients. Name one program that we've seen that uses this.

93. Client-server communications operate asynchronously.

- (a) /[1] What does this mean?

- (b) /[2] Name an advantage and a disadvantage of this.

94. /[1] Name a widget class in the widget class hierarchy, is subclassed from the label widget class

95. /[1] Normally the widget instance hierarchy matches the window hierarchy. Name an exception.

96. /[6] For each of the following properties, say whether it applies more to Motif and to Tcl/tk.

- (a) More industry standard.
- (b) Easier to use.
- (c) Faster to execute.

- (d) Funded by ARPA.
- (e) Funded by OSF.
- (f) Source code freely available.

97. Look at the following Tcl 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?
  - (b) ☐/[1] What does -command do?
  - (c) ☐/[1] What does the program as a whole do?
98. ☐/[2] When the Bresenham line algorithm was invented, floating point operations were much slower than fixed point. However, on current CPUs, floating point is generally as fast as fixed (maybe faster!). However, the Bresenham algorithm still is useful in modern hi-performance graphics. Why?
99. ☐/[2] What is the difference between http and html?
100. ☐/[1] Which one of the following is correct?

Position x,y;

- (a) XtVaGetValues (top\_level, XmNx, &x, XmNy, &y, NULL);
- (b) XtVaGetValues (top\_level, XmNx, x, XmNy, y, NULL);

101. ☐/[2] What file is accessed by the following URL?
- <http://www.rpi.edu/~pipes/>

102. Look at the condensed class home page below.

```
<head><title>35.475 Computer Graphics, Spring 1995, Index Page</title></head>

<BODY><P>

<H1>35.475 Computer Graphics</H1>
<hr>
<H3> When and Where</H3>
Lectures: MW 11am - noon in CC3051.

Lab: F, either 11-noon in VCC-N, or noon-1 in VCC-S.
</DL>
<hr>
<H3>Misc Pages</H3>
<MENU>
 List of lectures
 Please register yourself.
 Give us your comments or suggestions.
 Class List.
</MENU>
<hr>

<H3>Handouts and Old Homeworks</H3>

My secretary, Audrey Hayner, in JEC 6012, has copies of most old
handouts, and of old graded homeworks that were not picked up in
class.

<hr>
<i>Last updated: 27 Feb 95</i>
<hr>
<P><ADDRESS>

Wm. Randolph Franklin, ECSE Dept, Rensselaer Polytechnic
Institute, wrf@ecse.rpi.edu
</ADDRESS>
</BODY>
```

- (a) /[ 1 ] What does <h3> mean?
- (b) /[ 1 ] What does <hr> mean?
- (c) /[ 3 ] List the last URL seen on the page. What machine is the page retrieved from? What is the pathname of the file on that machine?

## X questions

103. About the Salim Abi-Ezzi patent:

- (a) /[ 1 ] What does **NURBS** stand for?

- (b) /[ 2 ] The **R** represents a choice that was made. What was the alternative, and why was the **R** choice better?

104. About *Foley*, chap 1:

- (a) /[ 1 ] Why is raster graphics better than vector graphics?

- (b) /[ 1 ] So, why didn't people use raster graphics a lot 20 years ago?

105. /[ 4 ] Calculate the pixels on a circle of radius 19 using the Bresenham method. Show your work.

106. /[ 3 ] What are bundled attributes? Give an advantage and a disadvantage of them.

107. /[ 3 ] When filling a polygon, which is defined by a list of vertices, the simple way, what problem is caused by the current scan line going through a vertex? How do you work around this problem?

108. /[ 1 ] Portability has at least one disadvantage, other than making it easier for your competitor to copy your program. Name it.

109. /[ 1 ] What is the advantage of a Bresenham-style algorithm compared to simpler, more obvious methods?

110. /[ 1 ] Why do graphics packages contain polyline routines in spite of the fact that users could easily just call the line routine many times?

111. /[ 2 ] Name an advantage and a disadvantage of specifying attributes for, e.g., a line, in every call to draw the line.

112. /[ 2 ] Suppose you used a paint program to outline a region by setting pixels around its border. Describe how the paint program could operate to fill its interior. Illustrate this by filling the interior of the letter **B**.

113. /[ 2 ] How can the above method be used when colorizing movies? The problem is that a frame of a black-and-white movie doesn't have pixels of a certain value outlining each region that should be a single color.

114. /[ 1 ] Why would you be interested in doing a Bresenham algorithm at a sub-pixel accuracy instead of simply rounding the line endpoints (or circle center and radius) to the nearest pixel before applying Bresenham?

115.  [ 1 ] Why would you be interested in doing a Bresenham algorithm at a sub-pixel accuracy instead of simply using a higher resolution device?
116.  [ 2 ] Write your name clearly on the top of your answer sheet.
117. Consider the program `coloredit.c`. Ignore any Motif features that are new to you, except for the ones summarized below; understanding the others is not necessary for answering this question.

---

```

/*****
 * coloredit.c: A simple color editor.
 * From:
 * The X Window System,
 * Programming and Applications with Xt
 * OSF/Motif Edition
 * by
 * Douglas Young
 * Prentice Hall, 1990
 *
 * Example described on pages: 178-188
 *
 *
 * Copyright 1989 by Prentice Hall
 * All Rights Reserved
 *
 * This code is based on the OSF/Motif widget set and the X Window System
 *
 * Permission to use, copy, modify, and distribute this software for
 * any purpose and without fee is hereby granted, provided that the above
 * copyright notice appear in all copies and that both the copyright notice
 * and this permission notice appear in supporting documentation.
 *
 * Prentice Hall and the author disclaim all warranties with regard to
 * this software, including all implied warranties of merchantability and fitness.
 * In no event shall Prentice Hall or the author be liable for any special,
 * indirect or cosequential damages or any damages whatsoever resulting from
 * loss of use, data or profits, whether in an action of contract, negligence
 * or other tortious action, arising out of or in connection with the use
 * or performance of this software.
 *
 * Open Software Foundation is a trademark of The Open Software Foundation, Inc.
 * OSF is a trademark of Open Software Foundation, Inc.
 * OSF/Motif is a trademark of Open Software Foundation, Inc.
 * Motif is a trademark of Open Software Foundation, Inc.
 * DEC is a registered trademark of Digital Equipment Corporation
 * HP is a registered trademark of the Hewlett Packard Company
 * DIGITAL is a registered trademark of Digital Equipment Corporation
 * X Window System is a trademark of the Massachusetts Institute of Technology
 *****/

```

10

20

30

/ 40

```

#include "coloredit.h"

main(argc, argv)
 int argc;
 char *argv[];
{
 Widget toplevel, bb, sliders, rc;
 Colormap def_colormap;
 XColor Colors[MAXCOLORS];
 int i;
 /*
 * Initialize the Intrinsics and save pointer to the display.
 */
 toplevel = XtInitialize(argv[0], "Coloredit", NULL, 0,
 &argc, argv);

 dpy = XtDisplay(toplevel);
 /*
 * Determine the number of colors to be edited.
 */
 ncolors = DisplayCells(dpy, DefaultScreen(dpy));
 if(ncolors > MAXCOLORS) ncolors = MAXCOLORS;
 /*
 * Create a base to hold everything.
 */
 rc = XtCreateManagedWidget("base", xmRowColumnWidgetClass,
 toplevel, NULL, 0);

 /*
 * Create a grid of buttons, one for each
 * color to be edited.
 */
 create_color_bar(rc);

 /*
 * A separator widget looks nice, between the colors
 * and the controls.
 */
 XtCreateManagedWidget("sep", xmSeparatorWidgetClass,
 rc, NULL, 0);

 /*
 * Put the controls inside an XmBulletinBoard widget
 */
 bb = XtCreateManagedWidget("controls",
 xmBulletinBoardWidgetClass,
 rc, NULL, 0);

 /*
 * Use an XmLabel widget to display the current rgb values.
 */
 color_display_panel =
 XtCreateManagedWidget("display",

```

50

60

70

80

90

```

 xmLabelWidgetClass,
 bb, NULL, 0);

/*
 * Create a row column widget containing three sliders,
 * one for each color component.
 */
sliders = XtCreateManagedWidget("sliderpanel",
 xmRowColumnWidgetClass,
 bb, NULL, 0);

red_slider = make_slider("red", sliders,
 red_slider_moved);
green_slider = make_slider("green", sliders,
 green_slider_moved);
blue_slider = make_slider("blue", sliders,
 blue_slider_moved);

/*
 * Add a quit button.
 */
xs_create_quit_button(bb);

/*
 * Get the ID of the default colormap.
 */
def_colormap = DefaultColormap(dpy, DefaultScreen(dpy));
for(i = 0; i < ncolors; i++) {
 Colors[i].pixel = i;
 Colors[i].flags = DoRed|DoGreen|DoBlue;
}
XQueryColors(dpy, def_colormap, Colors, ncolors);
my_colormap =
 XCreateColormap(dpy, DefaultRootWindow(dpy),
 DefaultVisual(dpy, DefaultScreen(dpy)),
 AllocAll);
XStoreColors(dpy, my_colormap, Colors, ncolors);

/*
 * Initialize the pixel member of the global color struct
 * To the first editable color cell.
 */
current_color.pixel = 0;

XtRealizeWidget(toplevel);

XSetWindowColormap(dpy, XtWindow(toplevel), my_colormap);

XtMainLoop();
}

Widget make_slider(name, parent, callback)
char *name;
Widget parent;
void (*callback)();

```

```

{
 Widget w;
 int n;
 Arg wargs[10];
 /*
 * Create an XmScale widget.
 */
 n = 0;
 XtSetArg(wargs[n], XmNminimum, 0); n++;
 XtSetArg(wargs[n], XmNmaximum, 65535); n++;
 w = XtCreateManagedWidget(name, xmScaleWidgetClass,
 parent, wargs, n);
 /*
 * Add callbacks to be invoked when the slider moves.
 */
 XtAddCallback(w, XmNvalueChangedCallback, callback, NULL);
 XtAddCallback(w, XmNdragCallback, callback, NULL);

 return (w);
}

Widget create_color_bar(parent)
 Widget parent;
{
 Widget panel;
 WidgetList colors;
 int i, n;
 char name[10];
 Arg wargs[10];
 colors = (WidgetList) XtMalloc(ncolors * sizeof(Widget));

 /*
 * Create the row column manager to hold all
 * color buttons.
 */
 n = 0;
 panel = XtCreateManagedWidget("colorpanel",
 xmRowColumnWidgetClass,
 parent, wargs, n);

 /*
 * Create ncolors widgets. Use the relative color
 * cell number as the name of each color. Add a
 * XmNactivateCallback for each cell with the color
 * index as client_data.
 */
 for(i=0; i<ncolors; i++){
 n = 0;
 XtSetArg(wargs[n], XmNbackground, i); n++;
 sprintf(name, "%d", i);

```



```

 colors[i] = XtCreateWidget(name, xmLabelWidgetClass,
 panel, wargs, n);
 XtAddEventHandler(colors[i], ButtonPressMask, False,
 set_current_color, i);
 }
 XtManageChildren(colors, ncolors);

 return panel;
}
200

void red_slider_moved(w, client_data, call_data)
 Widget w;
 caddr_t client_data;
 XmScaleCallbackStruct *call_data;
{
 /*
 * Set the red color components of the global
 * current_color structure.
 */
 current_color.red = call_data->value;
 /*
 * Update the digital rgb display and the current
 * color label.
 */
 update_color();
}
210

void blue_slider_moved(w, client_data, call_data)
 Widget w;
 caddr_t client_data;
 XmScaleCallbackStruct *call_data;
{
 /*
 * Set the blue color components of the global
 * current_color structure.
 */
 current_color.blue = call_data->value;
 /*
 * Update the digital rgb display and the current
 * color label.
 */
 update_color();
}
220

void green_slider_moved(w, client_data, call_data)
 Widget w;
 caddr_t client_data;
 XmScaleCallbackStruct *call_data;
{
 /*
230
240

```

```
 * Set the green color components of the global
 * current_color structure.
 */
 current_color.green = call_data->value;
 /*
 * Update the digital rgb display and the current
 * color label.
 */
 update_color();
} 250

update_color()
{
 Arg wargs[1];
 char str[25];

 /*
 * Update the digital display.
 */
 xs_wprintf(color_display_panel, "%3d %3d %3d", 260
 current_color.red, current_color.green,
 current_color.blue);

 /*
 * Update the current color.
 */
 XStoreColor(dpy, my_colormap, ¤t_color);
}

void set_current_color(w, number, event) 270
 Widget w;
 int number;
 XEvent *event;
{
 Arg wargs[10];
 current_color.flags = DoRed | DoGreen | DoBlue;
 /*
 * Get the current color components of the selected button.
 */
 current_color.pixel = number;
 XQueryColor(dpy, my_colormap, ¤t_color); 280
 /*
 * Use each color component to as the new
 * position of the corresponding slider.
 */
 XtSetArg(wargs[0], XmNvalue, current_color.red);
 XtSetValues(red_slider, wargs, 1);

 XtSetArg(wargs[0], XmNvalue, current_color.green);
 XtSetValues(green_slider, wargs, 1);
} 290
```

```

XtSetArg(wargs[0], XmNvalue, current_color.blue);
XtSetValues(blue_slider, wargs, 1);
}

```

---

*Summary of new routines*

- XtInitialize returns a top level widget, like XtVaAppInitialize.
- XtCreateManagedWidget and XtCreateWidget are like XtVaCreateManagedWidget; their relevant args are the first three: name, widget types, parent widget.

- (a) /[ 1 ] What is the program's class?
- (b) /[ 4 ] Sketch the tree of widget instances.
- (c) /[ 3 ] Name the three row column widgets.
- (d) /[ 5 ] Name the label widgets.

118. /[ 1 ] Correct this program.

```

main()
{
 char *s;
 s[0]='h';
 s[1]='i';
}

```

119. /[ 4 ] Print your name legibly on the top of page one, if you haven't already.

Total: 318 points  
*End of exam*

October 17, 1999, 19:7 /dept/ecse/graphics/hobigmidterm.tex