Chapter 11

Code Samples

Assignment 2 - Code quality?

```tcl
#!/usr/local/bin/wish
if {![info exists widgetDemo]} {
    error "This script should be run from the "widget" demo."
}
set w .plot
catch {destroy $w}
toplevel $w
wm title $w "Plot Demonstration"
wm iconname $w "Plot"
positionWindow $w
label $w.msg −font $font −wraplength 4i −justify left −text "This window ..."
pack $w.msg −side top
frame $w.buttons
pack $w.buttons −side bottom −fill x −pady 2m
button $w.buttons.dismiss −text Dismiss −command "destroy $w"
button $w.buttons.code −text "See Code" −command "showCode $w"
pack $w.buttons.dismiss $w.buttons.code −side left −expand 1
canvas $c −relief raised −width 450 −height 300
pack $w.c −side top −fill x
set plotFont {Helvetica 18}
$c create line 100 250 400 250 −width 2
$c create line 100 250 100 50 −width 2
$c create text 225 20 −text "A Simple Plot" −font $plotFont −fill brown
for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $plotFont
}
for {set i 0} {$i <= 5} {incr i} {
    set y [expr {250 − ($i*40)}]
    $c create line 100 $y 105 $y −width 2
    $c create text 96 $y −text [expr $i*50].0 −anchor e −font $plotFont
}
foreach point {{12 56} {20 94} {33 98} {32 120} {61 180} {75 160} {98 223}} {
    set x [expr {100 + (3*[lindex $point 0])}]
    set y [expr {250 − (4*[lindex $point 1])/5}]
    set item [c create oval [expr $x−6] [expr $y−6] [expr $x+6] [expr $y+6] −width 1 −outline black −fill SkyBlue2]
    $c addtag point withtag $item
}
c bind point <Any-Enter> "c itemconfig current −fill red"
c bind point <Any-Leave> "c itemconfig current −fill SkyBlue2"
c bind point <1> "plotDown $c %x %y"
c bind point <ButtonRelease-1> "c dtag selected"
c bind $c <B1-Motion> "plotMove $c %x %y"
set plot(lastX) 0
set plot(lastY) 0
proc plotDown {w x y} {
    global plot
    $w dtag selected
    $w addtag selected withtag current
    $w raise current
    set plot(lastX) $x
    set plot(lastY) $y
}
```
Assignment 2 - Code quality?

#!/usr/local/bin/wish
#
# This demonstration script creates a canvas widget showing a 2-D
# plot with data points that can be dragged with the mouse.
#
#
# plotDown -−
# This procedure is invoked when the mouse is pressed over one of the
# data points. It sets up state to allow the point to be dragged.
#
# Arguments:
# w − The canvas window.
# x, y − The coordinates of the mouse press.

proc plotDown {w x y} {
    global plot
    $w dtag selected
    $w addtag selected withtag current
    $w raise current
    set plot(lastX) $x
    set plot(lastY) $y
}

set w .pixel
catch {destroy $w}
label $w.m −font $font −wraplength 4i −justify left −text "This window ..."
pack $w.m −side top
frame $w.bottom
pack $w.bottom −side bottom −fill x −pady 2m
button $w.dismiss −text Dismiss −command "destroy $w"
canvas $c −relief raised −width 450 −height 300
pack $w.canvas −side top −fill x
set pf {Helvetica 18}
$c create line 100 250 400 250 −width 2
$c create line 100 250 100 50 −width 2
for {set i 0} {$i <= 10} {incr i} {
    set x [expr {100 + ($i*30)}]
    $c create line $x 250 $x 245 −width 2
    $c create text $x 254 −text [expr 10*$i] −anchor n −font $pf
}
for {set i 0} {$i <= 5} {incr i} {
    set y [expr {250 − ($i*40)}]
    $c create line 100 $y 105 $y −width 2
    $c create text 96 $y −text [expr $i*50].0 −anchor e −font $pf
}
foreach pt {{12 56} {20 94} {33 98} {32 120} {61 180} {75 160} {98 223}} {
    set x [expr {100 + (3*[lindex $pt 0])}]
    set y [expr {250 − (4*[lindex $pt 1])/5}]
    set im [$c create oval [expr $x−6] [expr $y−6] [expr $x+6] [expr $y+6] −width 1 −outline black −fill SkyBlue2]
    $c addtag pt withtag $im
}
$c bind pt <Any−Enter> "$c itemconfig ct −fill red"
$c bind pt <Any−Leave> "$c itemconfig ct −fill SkyBlue2"

Assignment 2 - Code quality?
Assignment 2 - debugging?

Write code clearly: Edit, document, comment...

```
# GetCommandString(x,y,itemID):string
# Returns a string that is later executed as a
# command
# The parameters x and y are the current cursor
# position, and itemID is the closest visible
# item on the canvas .canv
# Requires: Uses global variable canvas .canv
# Ensures: Always returns a command of some sort
# Sets global variable ErrorID if there is
# any error...
# Last modified: 12/2/2004 - by Hugh
```

✔ Run wish, and then use `source x.tcl`

✔ ... then interact with running program...
Assignment 3

3 options:

1. Re-implement YOUR assignment 2

2. A simple (but actually useful) visualization

3. Image library assistant...

Assignment 3 (option a)

✔ The tricky thing is the graphics component

✔ Some help with it...
public void paintComponent(Graphics g) {
    super.paintComponent(g); //paint background
    //Paint a filled rectangle at user's chosen point.
    if (point != null) {
        g.drawRect(point.x, point.y, rectWidth-1, rectHeight-1);
        g.setColor(Color.yellow);
        g.fillRect(point.x+1, point.y+1, rectWidth-2, rectHeight-2);
    }
}

1. Basic/AWT - Abstract Graphics class
2. Java2D
Coordinate system

- Upper left of each component is (0,0)
- Behind the title bar of a window
- Container class has `getInsets` method
- Graphics objects contain methods for drawing

Graphics API

- Swing components have a method `paintComponent` which takes a graphics object as an argument

```java
public void paintComponent(Graphics g)
```

- Overide this to draw your objects.
- Also may call the `repaint()` method
Graphics class methods

clearRect(int x, int y, int width, int height);
draw3DRect(int x, int y, int width, int height, boolean raised);
drawImage(Image img, int x, int y, Color bgColor, ImageObserver observer);
drawLine(int x1, int y1, int x2, int y2);
drawOval(int x, int y, int width, int height);
drawPolygon(int xPoints[], int yPoints[], int nPoints);
drawRect(int x, int y, int width, int height);
drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
drawString(String str, int x, int y);

fill3DRect(int x, int y, int width, int height, boolean raised);
fillArc(int x, int y, int width, int height, int startAngle, int arcAngle);
fillOval(int x, int y, int width, int height);
fillPolygon(int xPoints[], int yPoints[], int nPoints);
fillRect(int x, int y, int width, int height);
fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight);
Graphics class methods

Color getColor();
Font getFont();
FontMetrics getFontMetrics();
setColor(Color c);
setFont(Font font);

Graphics API

✔ Use JPanel instead of JComponent
✔ UI delegate (for look-and-feel painting) is called in JPanel
✔ UI delegate not called in JComponent
Text in Graphics API

✔ Note - you paint text using `drawString()`

✔ `getFontMetrics()` to get a `FontMetrics` object

```java
getHeight()
ggetAscent()
getDescent()
charWidth()
```

✔ and so on...

Assignment 3 (option b)

✔ Start with a large number (>1000000) points to be plotted, explored, displayed.

✔ If only a 1024*768 screen there are <1000000 points on screen.

✔ In some small region with (say) 10*10 points, there might be no difference between a display with 100 dots and one with 100000 dots.
Assignment 3 (option b)

✔ So...

✔ Tile the display

✔ Black and white? Colour?
Assignment 3 (option b)
✔ Must use a slider to change the tiling.
✔ May show different zoom levels, and locations of data
✔ Processing of other tilings in background using threads... (i.e. no pauses)
Assignment 3 (option c)

✔ Java application or a Java applet

✔ User interface to assist in the management of large numbers of images.

✔ Principally display TEXT information (spreadsheet),

✔ May also display small (thumbnail) versions of the images

Assignment 3 (option c)

✔ Database

✔ Special purpose editor for ...
  ✔ classifying,
  ✔ annotating and
  ✔ querying a large number of images.
Assignment 3 (option c)

✔ Image DSCN0100.JPG (Tim at a party): It is in
  • “Friends”
  • “Trip to NZ in Dec 2003”, which is itself in the section “Trips”
  • “Hooligans”

✔ Main screen shows a list of images.

Assignment 3 (option c)

Editable and fixed annotation fields:

• The date and time the image was entered into the section (not editable).

• A unique identifier for the image

• A scrollable text box with (say) 5 visible lines of text description.
Assignment 3 (option c)

Minimum flow of operation:

1. create, locate and delete new sections,

2. import image(s), using selection or cut and paste.

3. edit image/section information annotations,

4. save and load new databases,

5. query the system with a text search.

Deliverables:

✔ Single (zipped) file with sourcecode, README, docs in PDF

✔ Documentation:

✔ A title page, Table of contents...
✔ A one page introduction to the application
✔ A one page technical section
✔ A one to three page section describing the interface
Assessment:

The assessment is as follows:

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>15%</td>
</tr>
<tr>
<td>Code style/quality</td>
<td>35%</td>
</tr>
<tr>
<td>Operation of the interface</td>
<td>50%</td>
</tr>
</tbody>
</table>
Debugging Java

✔ Netbeans debugger

✔ The java debugger jdb

On suns...difficulty with versions of java and jdb and ddd

PATH=/usr/local/java/j2sdk1_3.1.02/bin:$PATH;export PATH
MFC

✔ Microsoft Foundation Classes - classes needed to produce GUI Windows programs.

✔ Development cycle - RAD, then editing.

MFC menus

A resource file for a simple File/Quit menu:

```c
#define MYAPP_EXIT 3210
MyApp MENU
    POPUP "File"
    {
        MENUITEM "Exit",MYAPP_EXIT
    }
```

Menus

In the `create` call, you can do something like this:

```cpp
Create( NULL, "Example", ..., CRect(...), NULL, "MyApp" );
```

The `MYAPP_EXIT` message may be bound using the `DECLARE_MESSAGE_MAP()` macro, and with the following declaration:

```cpp
ON_COMMAND( MYAPP_EXIT, OnExit )
```

Message handler

```cpp
afx_msg void CMenusWin::OnExit()
{
    SendMessage( WM_CLOSE );
}
```
```cpp
#include <afxwin.h>

class CFirstWindow : public CFrameWnd {
public:
    CFirstWindow();
    ~CFirstWindow();
private:
    CStatic *m_pGreeting;
};// CFirstWindow

CFirstWindow::CFirstWindow() {
    Create(NULL,
            "First Application",
            WS_OVERLAPPEDWINDOW,
            CRect(100, 100, 400, 220));
    m_pGreeting = new CStatic;
    m_pGreeting->Create("Hello World!",
                        WS_VISIBLE | WS_BORDER,
                        CRect(80, 30, 200, 50),
                        this);
} // CFirstWindow

CFirstWindow::~CFirstWindow() {
    delete m_pGreeting;
} // CFirstWindow

class CFirstApp : public CWinApp {
public:
    BOOL InitInstance() {
        m_pMainWnd = new CFirstWindow;
        m_pMainWnd->ShowWindow(m_nCmdShow);
        m_pMainWnd->UpdateWindow();
        return TRUE;
    }
} // CFirstApp
```
Hungarian notation

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Class declaration</td>
</tr>
<tr>
<td>m_</td>
<td>Class member variable</td>
</tr>
<tr>
<td>p</td>
<td>Pointer</td>
</tr>
<tr>
<td>n or i</td>
<td>Integer</td>
</tr>
<tr>
<td>On</td>
<td>Event or message handler</td>
</tr>
</tbody>
</table>

MFC class hierarchy

![MFC class hierarchy diagram]