CSE 142
Computer Programming I

Functions and Design

L-1

Overview

Design process
Functional decomposition
Top down vs. bottom up
Graphics primitives

L-2

Drawing a House

L-3

Drawing a House
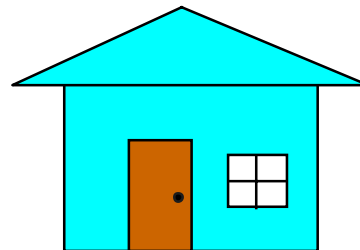


L-4

Drawing a (Similar) House

L-5

Draw House (Pseudo-code)

draw_house (color, ll_x, ll_y, num_windows)
    draw body as a colored rectangle
    draw roof as a colored triangle
    if num_windows is one
      draw door
      draw window
    if num_windows is two
      draw door
      draw window
      draw window

L-6

Functional Decomposition

Draw House

Draw Body   Draw Roof   Draw Door   Draw Window

Rectangle   Triangle

Rectangle   Circle   Rectangle   Line
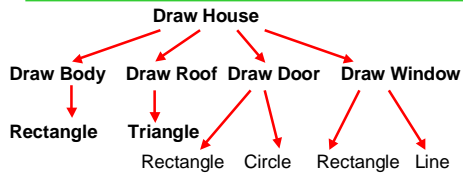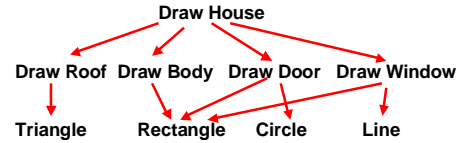
This is a "calling tree" or "static call graph."
Each function is shown, with an arrow down to
each function called.

L-7

Functional Decomposition

Draw House

Draw Roof   Draw Body   Draw Door   Draw Window

Triangle   Rectangle   Circle   Line

Each function shown only once (preferred)

L-8

Analysis to Design to Programming

Analyze the problem
Then design a "big-picture" solution
   A functional decomposition shows how the
   pieces fit together
Then design individual functions
   May depend on low-level ("primitive")
   functions available
Final programming may be very detailed

L-9

Top Down vs. Bottom Up

Sometimes designers start from the big picture
   Gradually work down to smaller pieces and
   then to fine details
   Called the "top down approach"

Sometimes people start with small pieces
   Figure out how they can fit together pieces to
   solve ever larger and larger problems
   Called the "bottom up approach"

L-10

Top Down or Bottom Up?

Which approach are we following with
DrawHouse?

Answer: Generally, top down. But we
have to look ahead and know what low
level functions will be available

Eventually, there will be graphics
programming to do. Fortunately, most
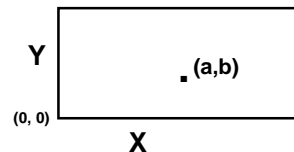systems supply a library of graphics
"primitives"

L-11

Graphics Primitive

Typical functions: clearscreen, draw circle,
rectangle, line, ellipse, etc.
Typical parameters: location, color, fill, etc.
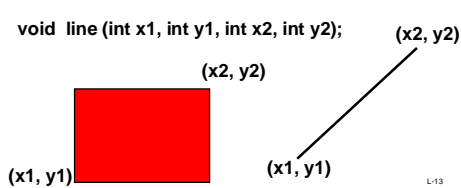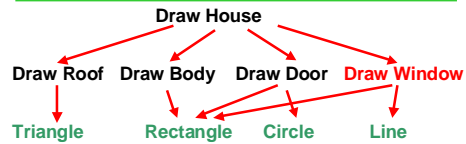Requires a coordinate system

Y

. (a,b)

(0, 0)

X

L-12

Typical 'rectangle' and 'line'

void
rectangle (int color, int x1, int y1, int x2, int y2);
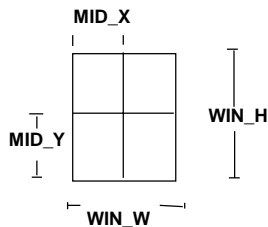
void line (int x1, int y1, int x2, int y2);

(x2, y2)

(x2, y2)

(x1, y1)

(x1, y1)

L-13

Big Picture Again

Draw House

Draw Roof   Draw Body   Draw Door   **Draw Window**

Triangle   **Rectangle**   **Circle**   **Line**

Fill in the pieces one at a time

L-14

Window Constants

Our analysis of how to describe a window

MID_X

WIN_H

MID_Y

WIN_W

L-15

Map Analysis to C Code

Identify and declare constants
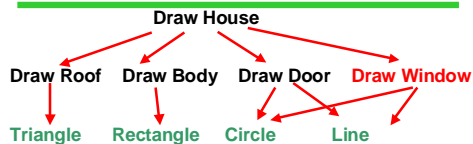**Choose parameters**
Utilize primitives
Get the picky details right, too!
void draw_window(int x, int y)
   /* (x,y) is the lower left corner of the window */
{
    rectangle(WHITE, x, y, x + WIN_W, y + WIN_H);
    line(x+MID_X, y, x + MID_X, y + WIN_H);
    line(x,y + MID_Y, x + WIN_W, y + MID_Y);
}

L-16

Keep Filling in Pieces

Draw House

Draw Roof   Draw Body   Draw Door   **Draw Window**

Triangle   **Rectangle**   **Circle**   **Line**

Analyze and code remaining functions
Does the order matter?
  Coding could be bottom-up, even if design was top-down, and vice-versa
  If the design is good, the functions can be implemented independently

L-17

Implementing Independently: Preconditions and Postconditions

Each function in the decomposition has a "contract":
 – **precondition: what the caller must ensure**
 – **postcondition: what the function promises**
This gives you a way to _assume_ that even unwritten functions work!
 – **establish the preconditions**
 – **assume the postconditions!**

L-18

L-3

draw_house: preconds and postconds

void **draw_house** (int color, int ll_x,
                          int ll_y, int windows)

preconds: what are they?

postconds: what are they?

Draw House (Gory Detail I)

```
void draw_house (int color, int ll_x,
                     int ll_y, int windows)
{
    int roof_ll_x, roof_ll_y ;

    /* Draw Body */
    draw_body (color, ll_x, ll_y) ;

    /* Draw Roof */
    roof_ll_x = ll_x - OVERHANG ;
    roof_ll_y = ll_y + BODY_HEIGHT ;
    draw_roof (color, roof_ll_x , roof_ll_y) ;
```

Draw House (Gory Detail II)

```
    if (windows == 1)
    {
        draw_door (ll_x + DOOR_OFFSET_1, ll_y) ;
        draw_window (ll_x + WINDOW_OFFSET_1,
            ll_y + WINDOW_RAISE) ;
    }
    else if (windows == 2)
    {
        draw_door (ll_x + DOOR_OFFSET_2, ll_y) ;
        draw_window (ll_x + WINDOW_OFFSET_2A,
            ll_y + WINDOW_RAISE) ;
        draw_window (ll_x + WINDOW_OFFSET_2B,
            ll_y + WINDOW_RAISE) ;
    }
}
```


Draw House (gory details)

```
void draw_house (int color, int ll_x,
                     int ll_y, int windows)
{
    int roof_ll_x, roof_ll_y ;                              else if (windows == 2)
                                                            {
    /* Draw Body */                                             draw_door (ll_x + DOOR_OFFSET_2, ll_y) ;
    draw_body (color, ll_x, ll_y) ;                             draw_window (ll_x + WINDOW_OFFSET_2A,
                                                                    ll_y + WINDOW_RAISE) ;
    /* Draw Roof */                                             draw_window (ll_x + WINDOW_OFFSET_2B,
    roof_ll_x = ll_x - OVERHANG ;                                   ll_y + WINDOW_RAISE) ;
    roof_ll_y = ll_y + BODY_HEIGHT ;                        }
    draw_roof (color, roof_ll_x , roof_ll_y) ;          }

    /* Draw Door and Window(s) */
    if (windows == 1)
    {
        draw_door (ll_x + DOOR_OFFSET_1, ll_y) ;
        draw_window (ll_x + WINDOW_OFFSET_1,
                ll_y + WINDOW_RAISE) ;
    }
```

Next Step: A Neighborhood
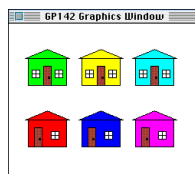
We could write 6 different functions...

Smarter - call 1 function 6 times...

Summary of Functional Decomposition
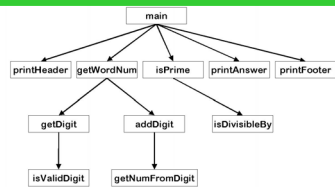
Look for common elements (similarities)

Parameterize for special features (differences)

Determine which functions will use others
    Draw a graph to show their relationships

QOTD: A Study in Decomposition

Which of these functions will include a loop?
Which will need to read data (scanf)/write data (printf)?
Which of them (really) returns a Boolean value?
Which ones might call "**exit**"?