Drawing Fractals

6 Mar 2009
CMPT166
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review last time

- Applications of recursion
  - Towers of Hanoi: algorithm
  - Gray codes: algorithm to generate
  - Fractals: self-similarity
    - Iterated function systems
    - Sierpinski triangle: recursive solution
    - Sierpinski triangle: Chaos Game

Quiz 4 (10mins, 20pts)

- Describe how to design/structure an FLTK program that draws in a widget (like our Scribble example) [5]
- Name and describe at least five functions that draw FLTK "fast shapes" (we mentioned 8). Be sure to describe the parameters, too. [6]
- What are some limitations of the "fast shapes"? [4]
- Describe what fl_push_matrix() and fl_pop_matrix() do. [5]

Quiz 4 answers: #1

- Describe how to design/structure an FLTK program that draws in a widget (like our Scribble example) [5]
  - Setup UI: create a widget to draw in
    - e.g., an Fl_Box
  - In the C++ tab, change the subclass to the class we are going to write
  - In C++ file, create a subclass of Fl_Box
  - Override the draw() method
  - #include <FL/fl_draw.H> and use the FLTK drawing functions within draw()

Quiz 4 answers: #2-3

- Name and describe at least five functions that draw FLTK "fast shapes" (we mentioned 8). Be sure to describe the parameters, too. [6]
  - fl_point(x,y), fl_line(x,y,x2,y2), fl_rect/rectf(x,y,w,h), fl_loop/polygon(x,y,x2,y2,x3,y3), fl_arc/pie(x,y,w,h,a1,a2)
- What are some limitations of the "fast shapes"? [4]
  - No transforms
  - loop/polygon only up to 4 vertices, convex

Quiz 4 answers: #4

- Describe what fl_push_matrix() and fl_pop_matrix() do. [5]
  - Transform matrices: only applies to FLTK complex shapes
  - x,y coordinates of objects are transformed by the matrix to find their final position on the screen
  - push_matrix() saves old transform so that we can change transform (scale, rotate, transform) and later restore using pop_matrix()

Drawing IFS in FLTK

- Some fractals like Mandelbrot set are images:
  - Render into a pixel buffer (char *)
- Iterated function systems can be drawn using FLTK's <FL/fl_draw.H> functions
- Transform matrices are useful!
  - Use complex shapes, not fast-draw ones
    - **draw triangle**
    - **translate and scale smaller**
    - **draw triangle**
    - **recurse....**

FractalTree example

See FractalTree example

- Two parameters: angle and shrink
- On each recursive step (see drawBranch()):
  - Draw left branch, then
    - Rotate (by angle) and transl (to end)
    - Recurse to fill out left branch
    - Pop matrix
  - Draw right branch, then
    - Rotate (by angle) and transl (to end)
    - Recurse to fill out left branch
    - Pop matrix
- Stop when branches are <2 pixels long

FLTK event handling

- Two parameters: angle and shrink
  - Use horiz mouse for angle, vert for shrink
- Override the handle() method:
  - Takes one int parameter: kind of event
    - enum type Fl_Event
  - Returns int: 1 if handled, 0 if not
    - If not handled, the event is passed on to the enclosing widget (e.g., window)
  - switch on the kind of event: FL_PUSH, FL_RELEASE, FL_KEYDOWN, etc.
  - <FL/Enumerations.H> has full list

Listening for events

- We want to listen for mouse click or drag events
  - switch(e) {
    - case FL_PUSH: case FL_DRAG:
      - **// handle this event**
      - **return(1); // flag that it's taken care of**
- For all other events that we don't care about, pass it on to the superclass to handle:
  - return(Fl_Box::handle());
- If that doesn't take care of it, FLTK will send the event to the enclosing widget for handling

Mouse event info

- Now that we know it's a mouse push/drag event, where are the mouse coordinates?
  - **#include <FL/Fl.H>**
  - **Fl::event_x() and Fl::event_y()**
- Also, use Fl::event_button() to get which mouse button was pressed
- Fl::event_clicks() returns (# clicks)-1
  - So it returns “true” if double-click
- For FL_KEYDOWN events, Fl::event_key() returns which key was pressed
  - See FLTK docs ch6 for more details