GL Utility Toolkit (GLUT)

* OpenGL does NOT draw window
* GLUT only draws window (no slider, dialog boxes, etc. Need API to do so)

Rendering steps:

* Generate triangle corners
* Create GPU buffer
* GLuint vao;
* glGenVertexArrays //create Vertex Array Object
* glBindVertexArray // make VAO active
* GLuint buffer;
* glGenBuffer //make one Vertex Buffer Object VAO->VBO
* glBindBuffer //make VBO active
* Move points to GPU memory
* glBufferData( GL\_ARRAY\_BUFFER, buffer, sizeof(points), points, GL\_STATIC\_DRAW)
* Draw points (from VBO)
* glDrawArray (GL\_POINTS, GL\_LINES, GL\_LINE\_STRIP, GL\_POLYGON, GL\_LINE\_LOOP)

Slide 2a:

Sierpinski

Slide 2b

Window-to-viewport mapping

1. define world window (original drawing extents)

mat4 ortho = Ortho2D (left, right, bottom, top)

//generates 4x4 matrix that scales input drawing

1. glViewport (l, b, (r-l), (t-b)) (drawing extents on screen)
2. map drawings within window to viewport

Maintaining ratio:

R: original ratio. W = window width, H = height

If(R>W/H) glViewport (x,y,w,w/r)

If(R<W/H) glViewport (x,y,h\*r, h)

Special Keys:

glutSpecialFunc

mouse: x=x, y = screenHeight-y

Menu:

glutCreate(myMenu);

glutAddMenuEntry(“clear screen”, 1)

glutAttachMenu(Glut\_Right\_Button)

void mymenu(int value){

if(value == 1) {…};

Slide 3a. Fractals

-self-similarity

-Application (Grass, Fire, Coastline)

1. Koch Curve

- divide line into 3 equal parts

- replace middle section with triangular bump, 1/3l

- F-F++F-F

2. Hilbert Curve

- 兇

3. Gingerbread Man (115, 121, M = 40; L = 3)

- next\_x = M (1+2L) -y + abs(x-l\*m)

- next\_y = x

4. Fern (1,7,7,85)

5. Mandelbrot Set

- f(z) = s^2 +c ( s = 0; c = 0,2+0.5i)

- (x + yi)^2 = (x^2-y^2)+(2xy)i

(x + yi)^2 + (cx+cy\*i) = [(x^2-y^2)+cx]+(2xy+cy)\*i

- if no term has exceeded 2 after 100 iterations, never will

- int dwell (cx, cy) {

Temp, dx=cx, dy=cy; fsq = cx\*cx+cy\*cy

For(int count = 0; count<=100 && fsq<=4; count++){

Temp = dx;

dx = dx^2-dy^2+cx;

dy = 2\*temp\*dy+cy;

fsq = dx\*dx+dy\*dy;

}

Slide 3b

Single vs Double buffering

* glutInitDisplaytMode(GLUT\_SINGLE | GLUT\_RGB)
* double buffering (front and back framebuffers)

always render to the buffer that is NOT visible.

Call glutSwapBuffers instead of glFlush

Shader setup

1. Create a program object
2. Read shaders
3. Add + compile shaders
4. Link program
5. Link variables

Uniform Variable

1. Constant
2. Can be change in application and sent to shaders
3. Cannot change in shader
4. timeLoc = glGetUniformLocation(program, “time”) //get location of shader variable “time”)
5. glUniform1(timeLoc, etime) //connect “etime”->application variable

Passing Values

Cpp->vertices shader->fragment shader->framebuffer

Slide 3c: Linear Algebra

* Magnitude of A
* Normalizing a vector (vec/magnitude)
* Dot product

Angle between 2 vector

1. Magnitude of b and c
2. Normalize of b and c
3. Dot product of normalized b and c
4. Arc of step 3 (result is in radius!)

Cross Product (Newell Method) (important)

Vector Reflected from a surface (important)

1. r = a-2m

Forms of Equation

1. Explicit: y = mx+h
2. Implicit: ax+by+c=0;
3. Parametric: x(a) = ax0+(1-a)x1

Y(a) = ay0+(1-y)y1

Convexity

Convex (line btw 2 points in the object) vs. not convex (line not in object)

4a: 3d models

Declaring Vertex Attribute

* Position 3 (offsets)
* Color 3 (offsets)
* Tex0 2 (offsets)
* Tex1 2 (offsets)

Allocating

#define VERTEX\_ATTRIB\_SIZE

Malloc (numVertices\*VERTEX\_ATTRIB\_SIZE \* sizeof(float))

Specifying

glVertexAttribPointer() //to tell if it is color or position and where they are

Initialization I

GLunit vao;

glGenvertexArrays( 1, &vao);

glBindVertexArray( vao );

//create vertex array object

GLuint buffer;

glGenBuffers(1, &buffer );

glBindBuffer( GL\_ARRAY\_BUFFER, buffer);

glBufferData( GL\_ARRAY\_BUFFER, sizeof(points)+sizeof(colors), null, static)

//create a buffer object and move data to GPU

glBufferSubData( GL\_ARRAY\_BUFFER, 0, sizeof(points), points);

glBufferSubData( GL\_ARRAY\_BUFFER, sizeof(points), sizeof(color), points);

Slide 4b: Transformations

Translation, Scaling, shearing; (need to draw generic matrix)

Slide 4c: 3d rotation

Rotate by axie;

Rotate by point(important);