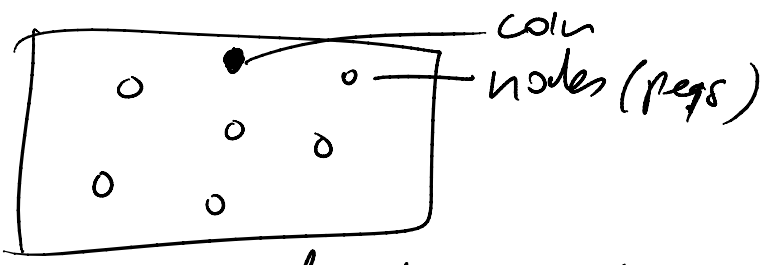
Tuesday, November 17, 2009
3:30 PM

Random notes on "Pachinko game"

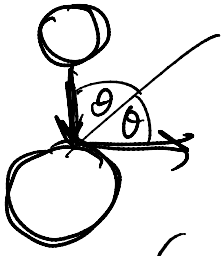
- "game engine" run in Qt idle function
- game engine: simple particle system,

⇒ Euler solver

- idea:



coin drops down;
bounces off pegs — each bounce:



Incidence = ∠response
(same as in ray tracer)

update that incoming ray is new velocity of coin

(x, y, z)

Tuesday, November 17, 2009
3:40 PM

- Drawing the range box: done in 2 steps

step1: right button press indicates start point

step2: right mouse button release 'fixes' bounding (range) box coordinates (x_2, y_2)

step 1.5 "in between" happens in mouseMove event — range box get continuously redrawn (animated)

to get step1 to work, your GL area (GLTexobj) has to have its mouse tracking state set true

in GLobjectWindow constructor:

(parent of GLTexobj ; other objects)

create the GLTexobj:

```
GLTexobj * texwin = new GLTexobj(this, "glarea");
                                 ↑
                              parent

texwin->setMouseTracking(true);
```

Tuesday, November 17, 2009
3:48 PM

these are GLTeapot::

- in mousePressEvent(QmouseEvent * e) member function

// x,y mapping
//...
// start drawing range grey box
if(e->button() & RightButton) {
{ drawingBox = true; // flag for animation
x1 = x; y1 = y; } both start & end points (corners of box) set to same coords
x2 = x; y2 = y; }
}

defined in: (data members of GLTeapot defined in glteapot.h

updateGL()

- in mouseMoveEvent(QmouseEvent * e)
// same x,y mapping
// update current bounding (range) box end point
x2 = x
y2 = y
updateGL(); // signal redraw event
(recall: don't call paintGL() directly)

- in mouseReleaseEvent(---)
// same x,y mapping
if(e->button() & RightButton) {

```
if (e->button() & RightButton) {
    drawBox = false;
    x2 = x; y2 = y;
    Point min, max;
    min[0] = x1 < x2 ? x1 : x2;
    min[1] = y1 < y2 ? y1 : y2;
    // same for max[0], max[1] ... width >
    kdTree.range(...)
}
updateGL();
```

- in paintGL:

// draw range hops
glColor4f(1.0, 1.0, 0, 1.0) // r,g,b,a = yellow
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)
if (drawingBox)
    glRectd(x1, y1, x2, y2)

normally, GL considers polygons to be filled

glPolygonMode(
    GL_FRONT_AND_BACK,
    GL_FILL);

here, I just want an outline.

// draw resulting range points
glColor4f(0,0,1,1)
for(int i=0; i<(int)range.size(); i++) {
    glBegin(GL_QUADS)
        glVertex2f((*range[i])[0]-5.0, [1]-5.0)
        " " [0]+5.0, [1]-5.0)
        " " [0]+5.0, [1]+5.0)
        " " [0]-5.0, [1]+5.0)
    glEnd()

same code as for drawing kd tree points — range points should be drawn after kd tree so that

range points if any, with br drawn
atop kdtree points

:
SwapBuffers();

Tuesday, November 17, 2009
4:10 PM

- Note that there are several things happening in each of the functions

paintGL():
- tries to draw everything (if the point, range query box, nn query point, knn query point, if any)
- implicitly checks to see if we have things to be drawn (kdtree could be empty, then might not be any range point, yet)

mousePressEvent, mouseReleaseEvent } these handle nn, knn, range, depending on mode button & "modifier" (CTRL key)

Tuesday, November 17, 2009
4:14 PM

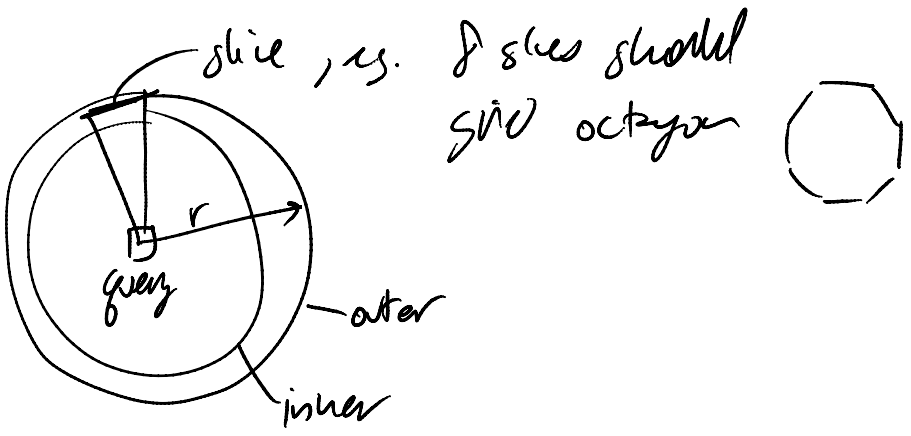
- one other tidbit: drawing circle to indicate radius around query point (for usa.kml)
- done with a GLU quadric

GL Utilities

#include <GL/glu.h>

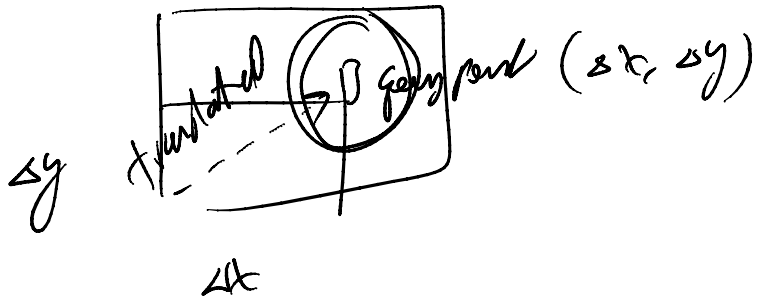
```
GLUquadricObj *qobj = gluNewQuadric();
glColor4f(1,1,0,1)
glPushMatrix();
glTranslatef( (*query)[0], (*query)[1], 0.0 );
gluDisk(qobj, radius-1.0, radius, 360, 1)
glPopMatrix()
```

innerradius outerradius #slices #loops



- what's this gluNewQuadric, gluDeleteQuadric?
- normally, the qobj would get drawn at origin

- want to translate circle to the reference frame of the grey point (as if grey point became origin)



- but glTranslate(x, y, z) would affect translation of everything being drawn, shifting by Δx, Δy
⇒ the push + pop matrix calls restrict shifting to just that object

- other important functions:

```
initializeGL() {
    glClearColor(0,0,0,0) — sets background color
}

resizeGL(int w, int h) {
    glViewport(0,0,w,h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    gluOrtho2D(-w,w,-h,h)
        ↳ this is why my mapping puts (0,0) at center — if I had (0,0) at bottom left, this call should be gluOrtho2D(0,w,0,h)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity();
}
```

- these set up world-to-screen coordinate transform