Day 7

Test

weird curve. to be applied

Lab 3

no starter code!
use lab 1 & lab 2

Calculate height field
for loop can go across & create vertices everywhere
with varying heights
the challenge is then to come up with an algorithm
that turns it into a triangle mesh
and calculate vertex normals

(Face and vertex normals)
low resolution height field means bigger triangles

Face normals means all vertices are the
same normals,

use cameras to fly through terrain - to come
textures - to come also,
check piazza for starter code type magic syntax.

load height field from file
create world
be able to navigate

textures/ normals/ lightings are necessary
first challenges
other than that its fun

map. ppm - height map data (want everything smooth)
loadable. ZSS means high, 0 means low
Going over basic requirements:

display list / VBO's cache, height field, geometry
store all geometry on graphics card, so less loading, should go faster
same idea as vertex buffer

change viewpoint / field of view
shorten / widen camera field view

Projection
early paintings didn't really get it
then Renaissance -
then formalized perspective rules

Pinhole imaging

depth of field / blur
doesn't exist in graphics

usually projection plane is between us and world

OpenGL implementation

object space

\[ \text{model/view matrix} \] (transformations = model, camera moves)
basically same thing = view

Camera/model on same stack

eye coordinates

rendering, viewing, projection matrix

Relevant
Viewing Transformation
transformation properly so camera is at origin
same thing as looking from certain point
where camera at, where camera viewing

\[ \text{eye, at, up} \]
what's the up? rotation
center, where it's aimed at location

each have x, y, and z
(look at sample code this slide)
void gluLookAt(...) 

Draw object in model view

~ Question about gluLookAt function ~

 gluLookAt can also be accomplished
by correct translation/rotations
(matrices are matrices)

answer was D because an object's notation
 told you y needed to be "up"