COMPOUTER GRAPHICS-MODELING

JEFWA NGOMBO NICHOLAS

ngombonj@gmail.com

0722641884

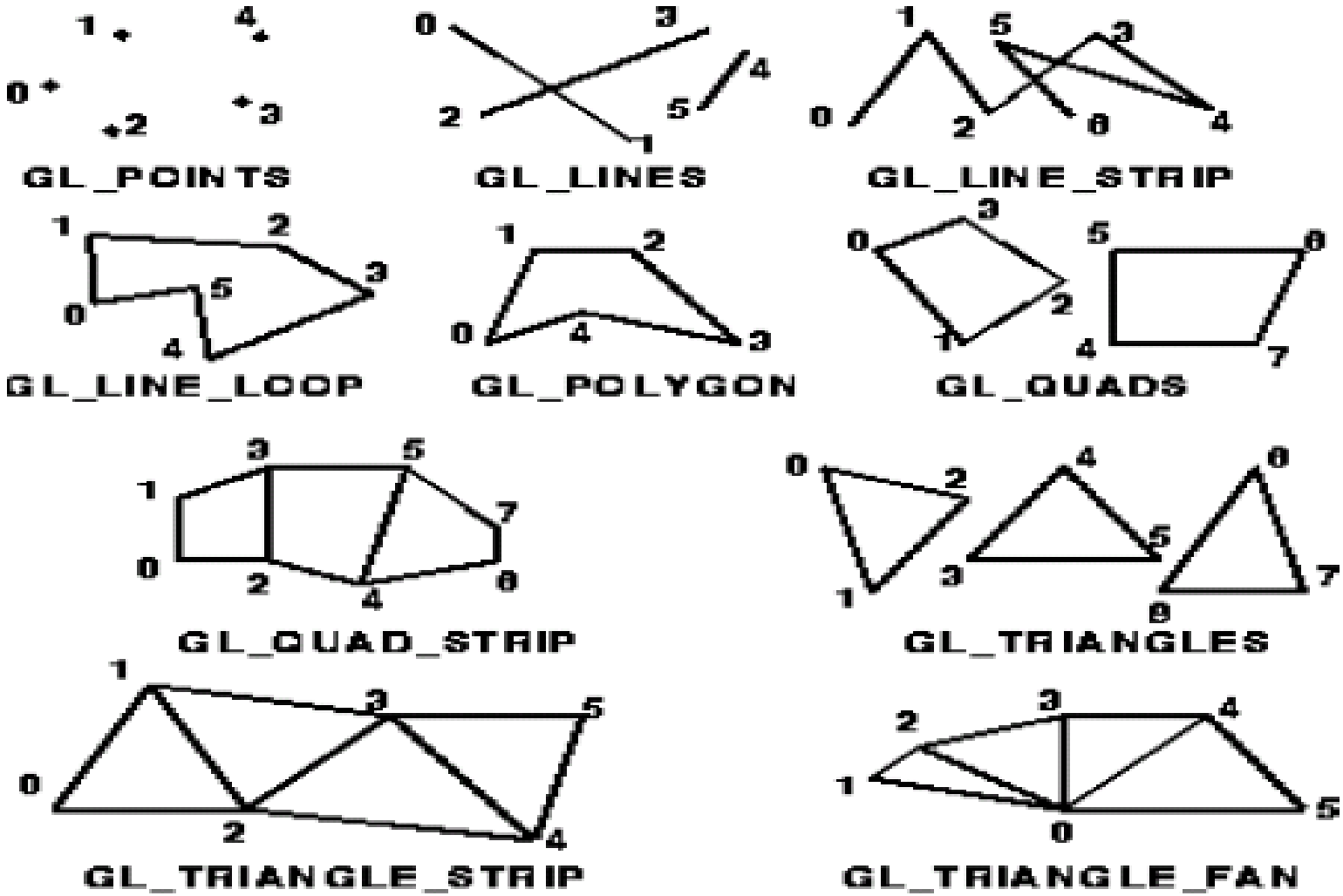
GRAPHICS-MODELING

Also called Scene modeling

The modeling involves shaping individual objects that are later used in the scene.

A model is created as a composition of graphic primitives

OpenGL primitives



2D Modeling-Line primitive

```
void drawLine(void){
/* clear all pixels */
glClear (GL_COLOR_BUFFER_BIT);
/* draw white line
* (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
*/
glColor3f (1.0, 1.0, 1.0);
glBegin(GL_LINES);
glVertex3f (0.25, 0.25, 0.0);
glVertex3f (0.75, 0.75, 0.0);
glEnd();
glFlush ();}
```

OpenGL program 2D Modeling-rectangle primitive

```
void display(void){
/* clear all pixels */
glClear (GL_COLOR_BUFFER_BIT);
/* draw white polygon (rectangle) with corners at
* (0.25, 0.25, 0.0) and (0.75, 0.75, 0.0)
*/
glColor3f (1.0, 1.0, 1.0);
glBegin(GL_POLYGON);
glVertex3f (0.25, 0.25, 0.0);
glVertex3f (0.75, 0.25, 0.0);
glVertex3f (0.75, 0.75, 0.0);
glVertex3f (0.25, 0.75, 0.0);
glEnd();
glFlush ();}
```

Initialisation part of OpenGL program

```
void init (void)
{
/* select clearing (background) color */
glClearColor (0.0, 0.0, 0.0, 0.0);
/* initialize viewing values */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

Main function starts the C program

```
 Declare initial window size, position, and display mode
* (single buffer and RGBA). Open window with "hello"
* in its title bar. Call initialization routines.
* Register callback function to display graphics.
* Enter main loop and process events.
*/
int main(int argc, char** argv)
glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (250, 250); // window size
glutInitWindowPosition (100, 100); // window position
glutCreateWindow ("hello");
init();// call init function
glutDisplayFunc(display); // callback function to display graphic
glutMainLoop();
return 0; /* ISO C requires main to return int. */
}
```

2D modeling-triangle primitive – data structure in openGL program

```
#include <C:\glut-3.7.6-bin\glut.h>
#include <stdlib.h>
void chora(void){
static GLint vertices[] = {25, 25,
100, 325,
175, 25,
175, 325,
250, 25,
325, 325};
static GLfloat colors[] = {1.0, 0.2, 0.2,
0.2, 0.2, 1.0,
0.8, 1.0, 0.2,
0.75, 0.75, 0.75,
0.35, 0.35, 0.35,
0.5, 0.5, 0.5};

}
```

2D modeling-triangle primitive – OpenGL program-poorly structured

```
glEnableClientState (GL_VERTEX_ARRAY);
glColorPointer (3, GL_FLOAT, 0, colors);
glVertexPointer (2, GL_INT, 0, vertices);
/*glBegin(GL_TRIANGLES);*
/*glArrayElement (2);*/
/*glArrayElement (3);*/
/*glArrayElement (5);*/
/*glEnd();*/
```

2D modeling-triangle primitive – OpenGL program-well structured

```
glEnableClientState (GL_VERTEX_ARRAY);
glColorPointer (3, GL_FLOAT, 0, colors);
glVertexPointer (2, GL_INT, 0, vertices);
int i, count;
count=2;
glBegin (GL_TRIANGLES);
for (i = 0; i < count; i++)
glArrayElement(vertices[i]);
glEnd();
glFlush();
```

2D modeling-triangle primitive – OpenGL program-initialisation part

```
void init (void)
{
/* select clearing (background) color */
glClearColor (0.0, 0.0, 0.0, 0.0);
/* initialize viewing values */
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

2D modeling-triangle primitive –OpenGL program- Excecution starting point

```
int main(int argc, char** argv)
{

glutInit(&argc, argv);
glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize (250, 250);
glutInitWindowPosition (100, 100);
glutCreateWindow ("VIPI");
init();
glutDisplayFunc(chora);
glutMainLoop();
return 0; /* ISO C requires main to return int. */
}
```

3D Object modeling

- Modeling Methods
  - Polygonal method
  - Curve modeling
  - Digital sculpting

Polygonal modeling

Polygonal modeling - Points in 3D space, called vertices, are connected by line segments to form a **polygonal mesh**. The vast majority of 3D models today are built as textured polygonal models, because they are flexible and because computers can render them so quickly. However, polygons are planar and can only approximate curved surfaces using many polygons.

3D Categories of representation using polygon surfaces

- Boundary representation
  - description of 3D object as a set of surfaces separating the object interior from the environment
- Space-partition representation
  - Describes the interior properties by partitioning the spatial region containing the object into a set of small non overlapping contiguous solids (usually cubes)

3D Polygonal modeling using triangle primitive

- Can use small triangles to compose 3D by triangulation e.g to model a sphere
  - A sphere algorithm operation would repeatedly call a triangle primitive and transformations of the three vertices

A sphere algorithm using triangle primitive

Identify the transformations that will be applied to vertices to generate a sphere from triangle primitives

3D Polygonal modeling using rectangle primitive

- Can use rectangular primitives to compose 3D e.g cuboid
  - A cuboid algorithm operation would repeatedly call a rectangular primitive and transformation of the four vertices

A cuboid algorithm using rectangle primitive

Identify the transformations that will be applied to vertices to generate a cuboid from rectangule primitives

Curve modeling method

- **Curve modeling** - Surfaces are defined by curves, which are influenced by weighted control points. The curve follows (but does not necessarily interpolate) the points. Increasing the weight for a point will pull the curve closer to that point.

Curve types

nonuniform rational B-spline (NURBS),

splines,

patches and geometric primitives

cubic bezier curve,

polyline,

polynomial

Digital sculpting

Digital sculpting - Still a fairly new method of modeling

There are currently 3 types of digital sculpting:

Displacement,

volumetric

dynamic tessellation.

Object modeling

- *Displacement*,

  Displacement uses a dense model (often generated by Subdivision surfaces of a polygon control mesh) and stores new locations for the vertex positions through use of a 32bit image map that stores the adjusted locations.

Object modeling

- *volumetric*

  Volumetric which is based loosely on Voxels has similar capabilities as displacement but does not suffer from polygon stretching when there are not enough polygons in a region to achieve a deformation.

Object modeling

- *dynamic tessellation*.

  similar to Voxel but divides the surface using triangulation to maintain a smooth surface and allow finer details.

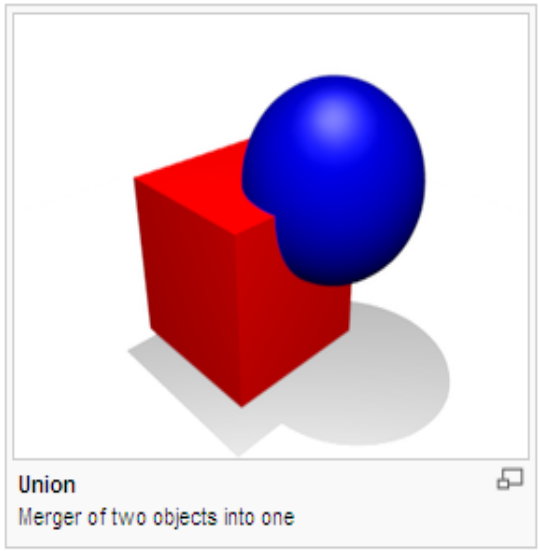
Modeling techniques

- There are a number of modeling techniques, including:
- constructive solid geometry
- implicit surfaces
- subdivision surfaces

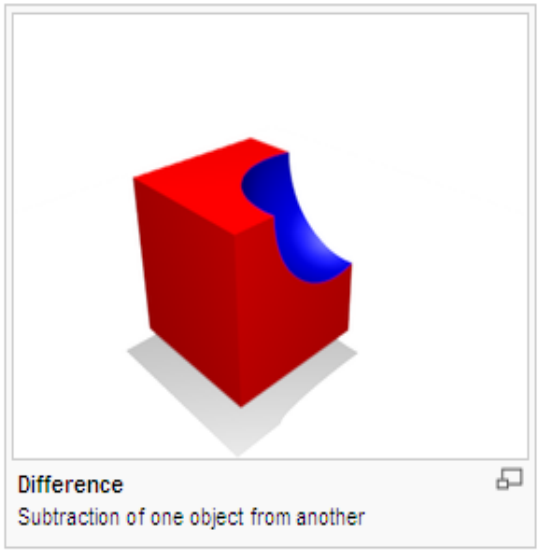
Modeling techniques

- constructive solid geometry
  - using Boolean operators to combine objects
  - The simplest solid objects used for the representation are called **primitives**. Typically they are the objects of simple shape: cuboids, cylinders, prisms, pyramids, spheres, cones. The set of allowable primitives is limited by each software package. Some software packages allow
  - an object is **constructed** from primitives by means of allowable **operations**, which are typically Boolean operations on sets: union, intersection and difference.

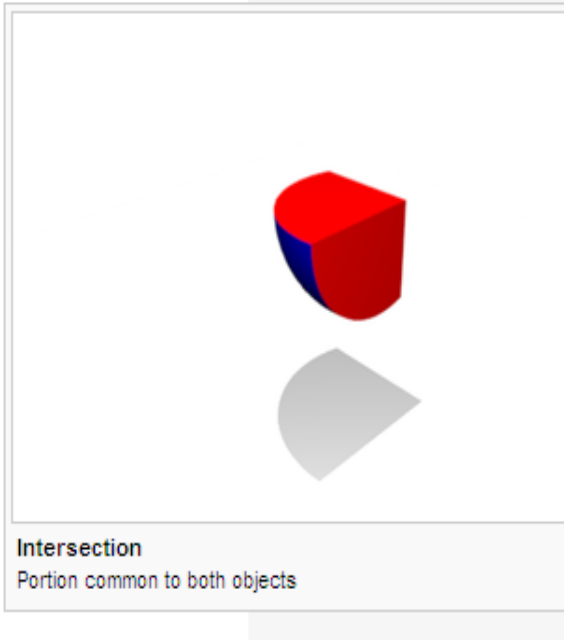
Modeling techniques



Union
Merger of two objects into one



Difference
Subtraction of one object from another



Intersection
Portion common to both objects

Modeling techniques

- an **implicit equation** is a relation of the form $R(x1,..., xn) = 0$, where R is a function of several variables (often a polynomial). The set of the values of the variables that satisfy this relation is a curve if $n = 2$ and a surface if $n=3$.
- For example, the implicit equation of the unit circle is $x^2+y^2-1=0$
- The terms **implicit curve** and **implicit surface** are usual to denote curves and surfaces defined in this way

Modeling techniques

- A **subdivision surface**, in the field of 3D computer graphics, is a method of representing a smooth surface via the specification of a coarser piecewise linear polygon mesh. The smooth surface can be calculated from the coarse mesh as the limit of a recursive process of subdividing each polygonal face into smaller faces that better approximate the smooth surface.

3D scene Modeling process of a single image

- 1. Construct shapes from geometric primitives, thereby creating mathematical descriptions of objects.
- (OpenGL considers points, lines, polygons, images, and bitmaps to be primitives.)
- 2. Arrange the objects in three-dimensional space and select the desired vantage point for viewing the composed scene.
- 3. Calculate the color of all the objects. The color might be explicitly assigned by the application,
- determined from specified lighting conditions, obtained by pasting a texture onto the objects, or some combination of these three actions.
- 4. Convert the mathematical description of objects and their associated color information to pixels on the screen. This process is called *rasterization*.

3D scene Modeling process of a single image

- Step1
  - produce a mathematical model of the object to be rendered.
  - The model should describe the *shape* of the object, its *color*, its *surface finish* (shiny, matte, transparent, fuzzy, scaly, rocky e.t.c).

3D scene Modeling process of a single image

- Step1 cont
  - Other information to define include: the location and characteristics of the *light sources* (their color, brightness), and the *atmospheric nature of the medium* through which the light travels (is it foggy or clear). the location of the *viewer* as holding a —synthetic camera, through which the image is to be photographed.
  - the *characteristics* of this camera (its focal length, for example).

3D scene Modeling process of a single image

- Step2
- **Projection**- Project the scene from 3-dimensional space onto the 2-dimensional image plane in our synthetic camera.
- step3
- **Color and shading**- For each point in our image we need to determine its color, which is a function of the object‘s surface color, its texture, the relative positions of light sources, and (in more complex illumination models) the indirect reflection of light off of other surfaces in the scene.

3D scene Modeling process of a single image

- Step3
- **Hidden surface removal-** Elements that are closer to the camera obscure more distant ones. We need to determine which surfaces are visible and which are not.
- step4
- **Rasterization** –convert the points into pixel data.