



Animation in Computer Graphics Using Local Deformation

Dr. A. Arokiasamy

Contents(1/4)

- The Objective of Research
- A Brief Introduction to OpenGL
- Animation
- Different types of Animation
- Various types of Computer Animation
- Animation in OpenGL

Contents(2/4)

- Deformation
- Different Types of deformation
- Regular Deformation
 - Tapering
 - Twisting
 - Bending

Contents(3/4)

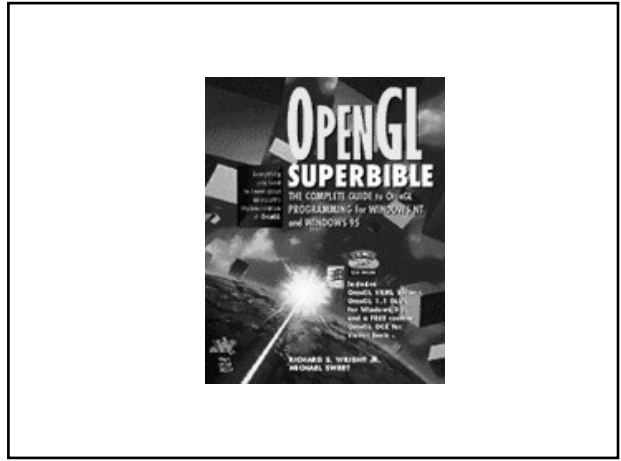
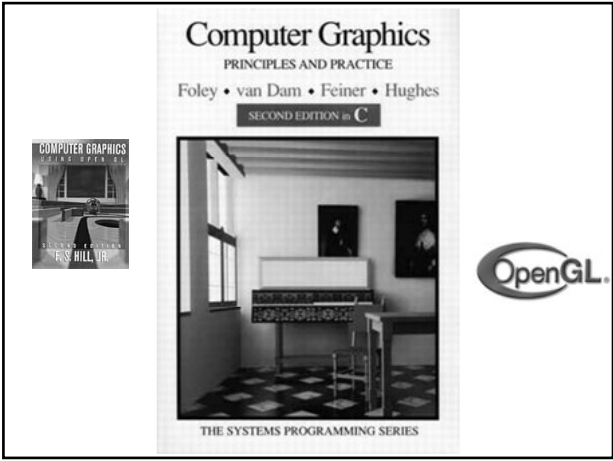
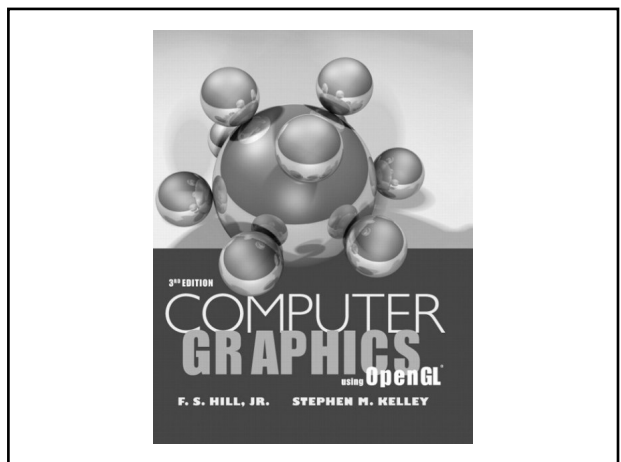
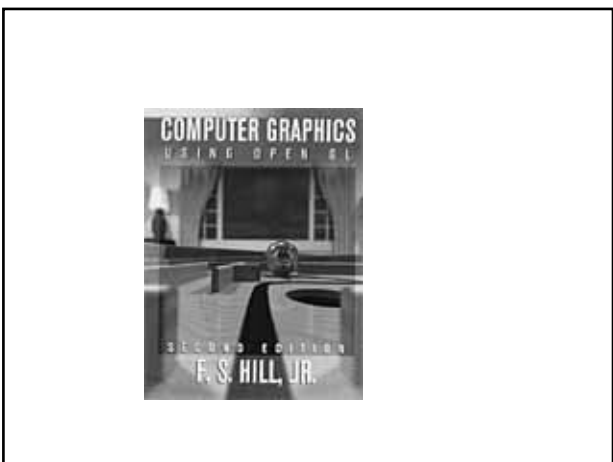
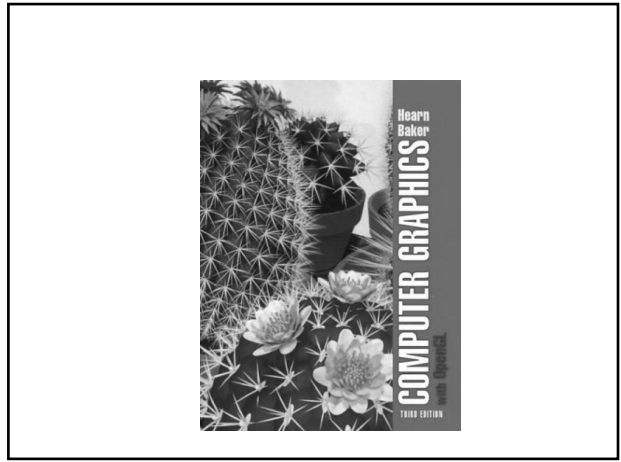
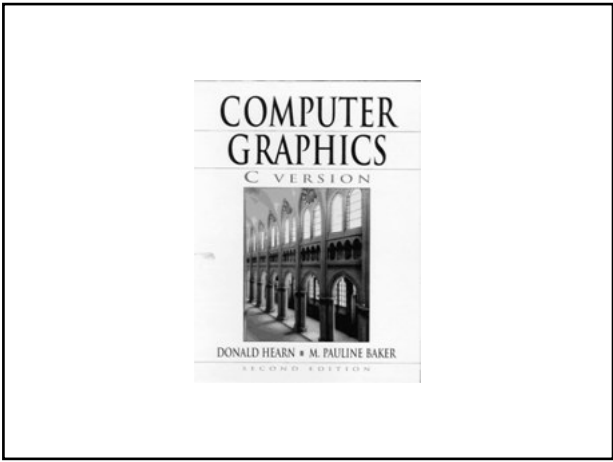
- Snapshots of deformation
- Local Deformation
- Local Deformation snapshots
- Animation by regular deformation and program execution
- Simulating rigid object animation using lighting effects in OpenGL

Contents(4/4)

- Coloring in OpenGL
- Material Properties
- Lighting in OpenGL
- Snapshots and program executions
- Closing Thoughts

Objective Of the Research

- The main aim of this research is to couple the dynamic motion with the deformation process. That is achieving animation through regular deformation technique.
- Our another work is based on how to achieve animation by rotating the position of light on the object that creates an illusion of the object itself rotating with the light.
- This is done in OpenGL which provides more flexibility to do this work.





A Brief Introduction to OpenGL(1/4)

- OpenGL is a low level, real-time 3D graphics API. It is a software interface for applications to generate interactive 2D and 3D computer graphics.
- OpenGL was originally developed in 1992 by silicon graphics, as a descendant of an API known as Iris GL for Unix.
- It was created as an open standard and is available on many different platforms. That is where the word “Open” in OpenGL derived from. Meanwhile, GL stands for graphics library.
- OpenGL is designed to be independent of operating system, window system, and hardware operations and many vendors support it.

A Brief Introduction to OpenGL(2/4)

- OpenGL is available on PCs, Macintosh and workstations.
- OpenGL provides a wide range of graphics functions from rendering a simple geometric point, line, or filled polygon, to texture mapping, NURBS, curved surfaces.
- OpenGL is a low-level, real-time 3D graphics API. By low-level we mean that OpenGL gives us a very fine degree of control as to how rendering takes place, and by real-time graphics we mean interactive 3D graphics as opposed to graphics that render over a long period of time.
- High-end applications have an extreme need for real-time performance, and OpenGL is often the API of choice for developing these kinds of applications

A Brief Introduction to OpenGL(3/4)

Developer Driven Advantages of OpenGL are it is

- an industry standard
- Stable
- Reliable
- Portable
- Evolving
- Scalable
- Easy to use
- Well-documented API.

A Brief Introduction to OpenGL(4/4)

- One of the strengths of OpenGL is visual fidelity and it is also often confused with performance issues. OpenGL is referred to as a “precise” rendering API, which should only be used when you have “exacting” rendering needs where performance is not a consideration.
- This API is designed to work efficiently even if the computer that displays the graphics is not the computer that runs the OpenGL program.
- With any compiler, the programmer will be able to create any 3D graphics within imagination with the help of OpenGL libraries.

Animation(1/2)

- Animation means “to bring to life”. By providing still images that change a number of times per second we can provide the illusion of movement, and thus of life. That is rapid display of slightly different images create the illusion of motion called animation.
- Animation speed is measured in frames per second (fps), which is the number of animation frames, or image changes presented every second. In the case of television and movies the human eye is tricked by a minimal amount of frames per second.
- Animation is a very important (and one of the most interesting) part of computer graphics. The application of graphics is increased significantly by being able to animate objects.

Animation(2/2)

Computer animation systems can generally be classified as either algorithmic or demonstrative.

1. Programming-language based systems (Algorithmic systems) require the animator to describe a movie in a written programming language. These systems generally are not interactive; the animator is provided no direct visual feedback.
2. Interactive systems based on freehand sketching (Demonstrative systems) allows the animator to sketch images and movements free-hand. These systems generally provide immediate real time playback of the resulting movie.

Different Types of Animation

- Traditional animation
- Cel animation
- Stop-motion animation
- Clay animation
- Computer animation

Various Types of Computer Animation(1/2)

- Keyframe animation
- Real time animation
- Character animation
- Motion path animation
- Hierarchical animation

Various Types of Computer Animation(2/2)

- Procedural animation
- Simulation
- Model animation
- Shape animation
- Camera animation

Animation in OpenGL(1/2)

•In OpenGL, animation is achieved by using variables for the parameters like position and rotation. The user or the program can then update these variables and when OpenGL redraws the screen, the image will be different.

•Continuous animation in OpenGL is done with the help of a GLUT Idle function. That is often we want to do some processing in our program repeatedly, or when nothing else is happening. You can do this using the GLUT idle function.

•First register the callback function to be run when OpenGL is idling (the idle function takes no parameters):

```
void glutIdleFunc(void (*func)(void));
```

Animation in OpenGL(2/2)

- By passing NULL to this function it turns off the idle function.
- OpenGL provides two display modes namely single buffer mode and double buffer mode.
- If we use single buffer mode some flickering will be there. That is a smooth animation cannot be achieved.
- We must use Double buffer mode for animation in OpenGL in order to avoid flickering.

Deformation

•Shape changes are referred to by a number of phrases, including flexible surfaces, shape metamorphosis, and shape deformations.

•Shape deformation is a deformation between the outlines-either 2D or 3D.

•In animating a rigid object we define the transformations for the object, save these transformations as keyframe and then ask the computer to interpolate the transformation values between the two frames. The interpolated transformation values yield the in-between positions of the objects.

•Similarly in animating a flexible or changing surface, we have to determine the position of the points that define the surface (the control points or vertices), save the positions of these points as a keyframe, reposition the surface-defining-points, save a new keyframe, and then ask computer to interpolate the values between the two key positions. The interpolated positions of the surface-defining-points yield the in-between shapes of the object.

Deformation Types

- Lattice Deformation
- Axis deformation
- Regular Deformation
- Spline deformation or Curve deformation
- Patch deformation
- Path deformation
- Morphing

Regular Deformation

- It is not easy to model an irregular object in a conventional solid modeling system.
- One of the most important problems of available solid modeling systems is that the range of shapes generated is limited.
- Without deformation, to simulate an irregular object, one has to save every point on the object. Then it will become a tedious process.
- However, one can create a regular object and then apply deformation to it to create an irregular object with much less data.

Types of Regular Deformation

- Tapering
- Twisting
- Bending

Tapering(1/2)

- Tapering is similar to scaling, by differentially changing the length of two global components without changing the length of the third.
- To do tapering operation along the z-axis one should choose a tapering function depending on the z-coordinates of the points.
- When the tapering function $f(z)=1$, then the portion of the deformed object is unchanged.
- The object increases in size as a function of z when $f'(z)>0$.
- The object decreases in size when $f'(z)<0$.
- The object passes through a singularity at $f(z)=0$.
- And it becomes everted when $f(z)<0$.

Tapering(2/2)

Global tapering along the z-axis is given by the following equations

$$\begin{aligned}r &= f(z) \\X &= rx \\Y &= ry \\Z &= z\end{aligned}$$

Twisting(1/2)

- For some applications, it is useful to simulate global twisting of an object.
- A twist can be approximated as differential rotation just as tapering is a differential scaling of the global basis vectors.
- We rotate one pair of global basis vectors as a function of height, without altering the third global basis vector
- The twist proceeds along the z-axis at a rate of $f'(z)$ radians per unit length in the z direction.

Twisting(2/2)

The global twist around the z axis is produced by the following equations:

$$\begin{aligned}\theta &= f(z) \\ C_\theta &= \cos(\theta) \\ S_\theta &= \sin(\theta) \\ X &= xC_\theta - yS_\theta \\ Y &= xS_\theta + yC_\theta \\ Z &= z.\end{aligned}$$

Bending (1/5)

- Bending simulates an important manufacturing process for fabricating objects.
- An example of this operation is the bending of a bar stock or sheet metal.
- For other applications, it is useful to have a simple simulation of bending.
- The length of the centerline does not change during the bending process.
- To make a bending along the y-axis, one has to specify a bent region along the y-axis.

Bending (2/5)

The following equations represent an isotropic bend along a centerline parallel to the y-axis:

$$\begin{aligned}\theta &= k(\hat{y} - y_0), \\ C_\theta &= \cos(\theta), \\ S_\theta &= \sin(\theta),\end{aligned}$$

where

$$\hat{y} = \begin{cases} y_{\min} & \text{if } y \leq y_{\min} \\ y & \text{if } y_{\min} < y < y_{\max} \\ y_{\max} & \text{if } y \geq y_{\max} \end{cases}$$

Bending (3/5)

The equation for this type of bending along the y-axis centerline is given by the following relations:

$$X = x$$

$$Y = \begin{cases} -S_\theta \left(z - \frac{1}{k} \right) + y_0, & y_{\min} \leq y \leq y_{\max} \\ -S_\theta \left(z - \frac{1}{k} \right) + y_0 + C_\theta (y - y_{\min}), & y < y_{\min} \\ -S_\theta \left(z - \frac{1}{k} \right) + y_0 + C_\theta (y - y_{\max}), & y > y_{\max} \end{cases}$$

Bending (4/5)

$$Z = \begin{cases} C_\theta \left(z - \frac{1}{k} \right) + \frac{1}{k}, & y_{\min} \leq y \leq y_{\max} \\ -C_\theta \left(z - \frac{1}{k} \right) + \frac{1}{k} + S_\theta (y - y_{\min}), & y < y_{\min} \\ -C_\theta \left(z - \frac{1}{k} \right) + \frac{1}{k} + S_\theta (y - y_{\max}), & y > y_{\max} \end{cases}$$

• The bending angle θ , is constant at the extremities, but changes linearly in the central region.

• In the bent region, the bending rate k , measured in radians per unit length, is constant, and the differential basis vectors are simultaneously rotated and translated around the third local basis vector.

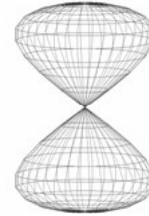
• Outside the bent region, the deformation consists of a rigid body rotation and translation.

Bending (5/5)

- The range of the bending deformation is controlled by y_{min} and y_{max} with the bent region corresponding to values of y such that $y_{min} < y < y_{max}$.
- The bending axis is located along $[s, y_0, 1/k]T$, where s is the parameter of the line.
- The center of the bending occurs at $y = y_0$ i.e., where one would "put one's thumbs" to create the bending. The radius of curvature of the bending is $1/k$.
- These functions have continuous values at the boundaries of each of the three regions for y , and in the limit, for $k = 0$.
- However, there is a jump in the derivative of the bending angle θ at the $y = y_{min}$ and $y = y_{max}$ boundaries. The discontinuities may be eliminated by using a smooth function for θ as a function of y .

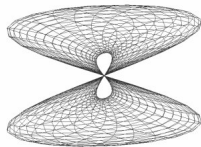
Deformation Snapshots

Tapered Sphere using simple function along y-axis



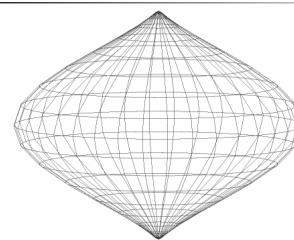
$$r1 = (y * \text{PI} / 180)$$

Tapered Torus using simple function along y-axis



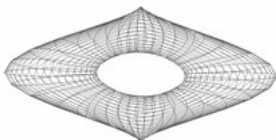
$$r1 = (y * \text{PI} / 180)$$

Tapered Sphere using cosine function along y-axis



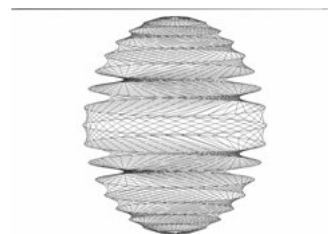
$$r1 = \cos(y * \text{PI} / 180)$$

Tapered Torus using cosine function along y-axis



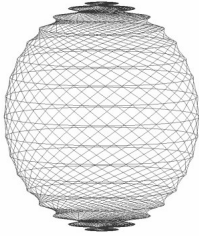
$$r1 = \cos(y * \text{PI} / 180)$$

Twisted Sphere using sine function along y-axis



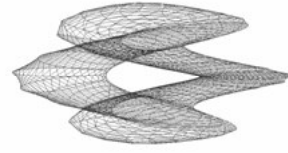
$$\text{theta} = \sin(y * \text{PI} / 180)$$

Twisted Sphere using Tan function along y-axis



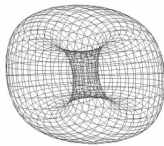
$$\theta = \tan(y * \pi / 180)$$

Twisted Torus using simple function along y-axis



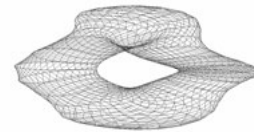
$$\theta = (y * \pi / 180)$$

Twisted Torus using cosine function along y-axis



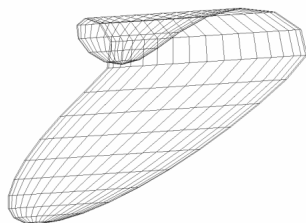
$$\theta = \cos(y * \pi / 180);$$

Twisted Torus using sine function along y-axis



$$\theta = \sin(y * \pi / 180);$$

Bent Sphere along y-axis



$$k = 1.0$$