CMSC 498M: Chapter 5
Modeling for Games

Reading: Sects. 8.1-8.15 in Hearn and Baker's graphics text.
http://www.geisswerks.com/ryan/BLOBS/blobs.html

Overview:

- Modeling Surfaces, Contexts, and Illumination
  - Implicit surfaces
  - Parametric surfaces
- Level of Detail

Modeling for Games

Modeling for Games

Modeling for Games

Display Processor-Based Graphics

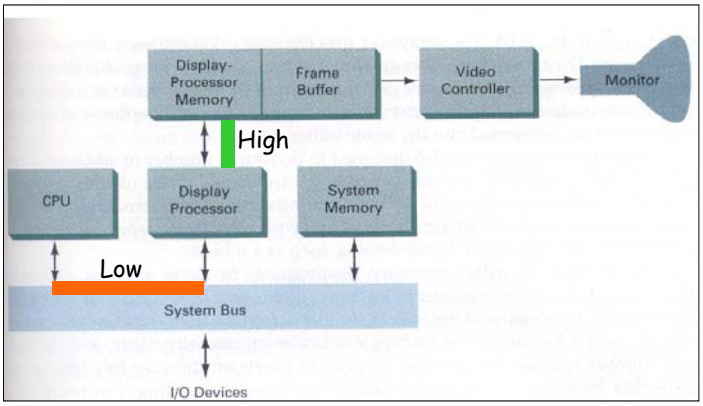


Image from Computer Graphics by Hearn and Baker

Consequences of GPU Archtecture

Low polygon counts:
- Minimize CPU-GPU communications.

Extensive use of textures for detail:
- Texture images.
- Bump maps and environment maps.
- Vertex/fragment shaders to create the impression of detail.

Limiting visibility:
- Control world view to control scene complexity.

Modeling with Low Polygon Counts

Level of Detail:

- Provide only as much geometry as needed to produce a realistic image.

69,451 polys 2,502 polys 251 polys 76 polys

Limiting Visibility

Dungeons and Corridors:

- Arrange rooms and connecting portals to control visible portion of the environment.

Limiting Visibility

Other ways to limit visibility:

- Enclosures (e.g. stadium)
- Fog
- Trees and vegetation

Modeling for Games

Elements of Modeling:

Modeling Surfaces and Geometry:

- Typically single objects.

Modeling Context:

- Complex environments.

Modeling Illumination:

- Enhancing realism.
- Creating mood.

Modeling Surface Detail: (we will discuss this later)

- Textures.
- Bumpiness and waves.
- Fur and hair.

Modeling for Games

Modeling Surfaces:

Character design:

- Personality, perception/visual psychology

Modeling Tools:

- 3ds Max, Maya, Blender, …

Free 3D model repositories:

- http://www.lodbook.com/models/
- http://www.3dvalley.com/
- http://www.3dcafe.com/
- http://www.the123d.com/3d-model.html

…**or commercial ones**:

- http://www.digimation.com/

Modeling Surfaces and Geometry

Modeling Surfaces and Object Geometry

General Surface Modeling Approaches:

Implicit modeling:

- Surface defined by the isosurface of some function.
- Example: Blobs.

Parametric modeling:

- Surface defined by functions over a parametric space.
- Examples: Bezier surfaces, NURBS surfaces.

Implicit Modeling

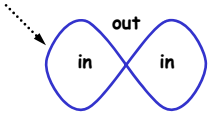
Implicit Modeling:

- Expresses a surface as the set of points that **satisfy an equation**:

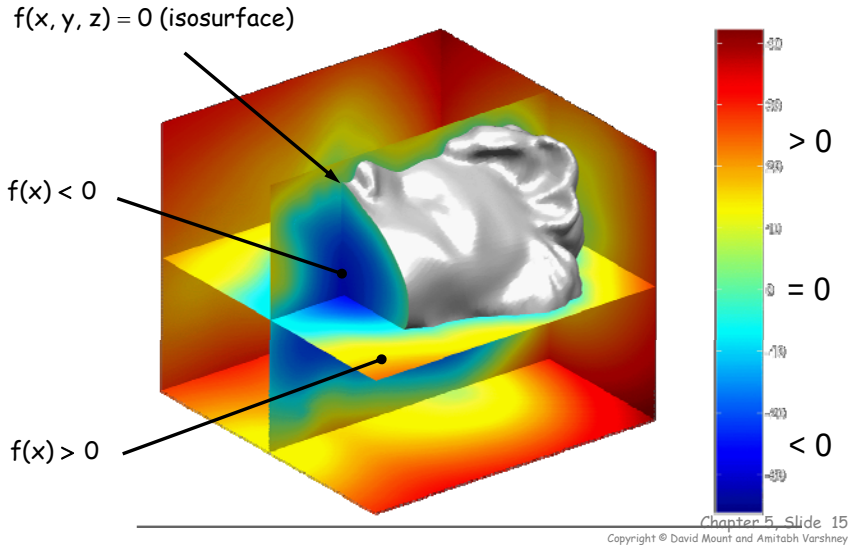
$$f(x, y, z) = 0$$

- Subdivides space into **inside/outside** regions based on whether $f(x, y, z) < 0$ or > 0. These regions need not be connected.

Isosurface: The set of points for which $f(x, y, z)$ has the same fixed value. (Need not be zero.)



Implicit Surface Representation



Implicit Surfaces: Issues

Advantages:

- **Smooth**: Surfaces are smooth and (depending on representation) easy to compute derivatives.
- **Membership**: Each to determine whether a point lies inside or outside by simply evaluating the function.
- **Full resolution**: Since representation is exact, can zoom in arbitrarily close.

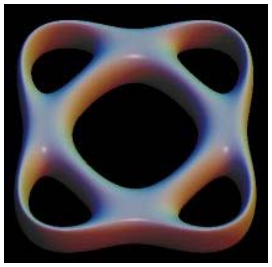


Image Source:
www.xahlee.org/surface/bretzel5/bretzel5.html

Disadvantages:

- **Hard to control**: May be difficult to determine the function that produces a desired general shape.
- **Hard to render**: Before rendering (using, say, OpenGL) need to convert to a mesh of polygon surface meshes.

Blobs and Metaballs

Blobs:

- Also known as **metaballs**, **blobby models**, **soft objects**.
- Useful for modeling objects made up of **soft rounded parts**, such as muscles for humans and animals.
- Based on combining **many simple implicit functions** into one **complex implicit function**.

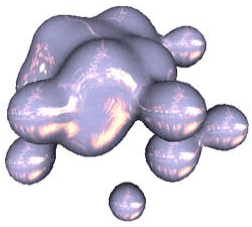


Image source: http://www.programmersheaven.com/2/Fast-metaballs

Blobs and Metaballs

Intuition:

- Consider a point $\mathbf{p} = (p_x, p_y, p_z)$, and any function f that **decreases as a function of the distance** r of point (x, y, z) from **p**.
- Let:

$$r = \sqrt{(x-p_x)^2 + (y-p_y)^2 + (z-p_z)^2}$$

Examples: (let a and b be parameters selected by the user)

$$f(x,y,z) = \frac{a}{r^2}$$

$$f(x,y,z) = e^{-ar^2}$$

$$f(x,y,z) = \begin{cases} a(1-3(r/b)^2) & 0 \le r \le b/3 \\ (3/2)a(1-r/b)^2 & b/3 \le r \le b \\ 0 & b \le r \end{cases}$$

The choice of f affects **blob shape**.

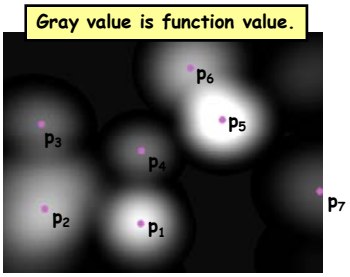
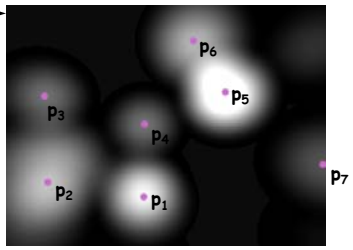


Image source: Ryan Geiss, www.geisswerks.com

Blobs and Metaballs

Intuition:

- Given number of points, sum of these functions for each such point. Like a **density** or **potential function**.
- Next, consider **isosurfaces** (like **contour lines** in a terrain map) that arise at various values of the density function.



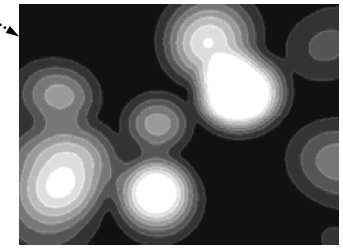


Image source: Ryan Geiss, www.geisswerks.com

Blobs and Metaballs

Intuition:

- Selecting an arbitrary **threshold value** T uniquely determines one of these isosurfaces. The result is our **blobby object**.
- **Final implicit function**: $f(x, y, z) = \sum_k c_k f_k(r) - T = 0$, where f_k is the density function for p_k, and c_k is an additional parameter.

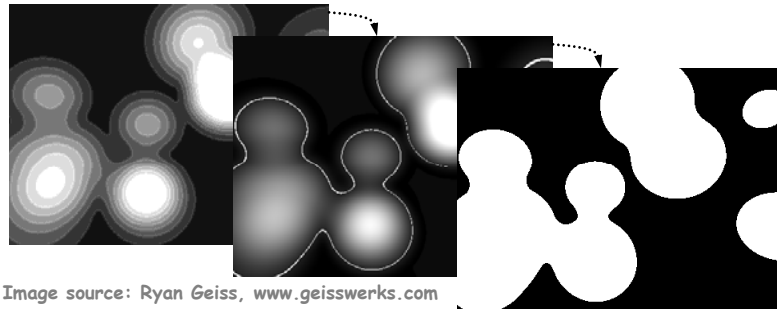


Image source: Ryan Geiss, www.geisswerks.com

Blobs and Metaballs

Example:

Image courtesy Spencer Arts

Parametric Surface Models

Parametric Model:

- Represents coordinates as **functions of parameters**.
- **Parametric curve**:

  $x(u) = \ldots \quad y(u) = \ldots \quad z(u) = \ldots \quad$ where $u_0 \le u \le u_1$.
- **Parametric surface**:

  $x(u, v) = \ldots \quad y(u, v) = \ldots \quad z(u, v) = \ldots \quad$ where $u_0 \le u \le u_1, v_0 \le v \le v_1$.

Advantages:

- Easy to **enumerate** points on the curve/surface.
- Easy to **render**: Just subdivide the surface into **quadrilateral** or **triangular patches**, which can be passed to OpenGL.

Disadvantages:

- Guaranteeing **smoothness** at patch boundaries can be tricky.
- **Sidedness test is harder**: Unlike implicit surfaces, it is hard to know whether you are inside or outside.

Parametric Modeling

- Useful for initial model creation (from which a triangulated mesh is made).
- Quake 3 (1999/2000) was the first game to use Bezier patches.

Parametric Polynomial Curves

Parametric Polynomial Curve: "Nicest" parametric representation.

- The parametric functions are **polynomials of degree d** in the parameters.

$$x(u) = \sum_{i=0}^{d} a_i u^i \qquad y(u) = \sum_{i=0}^{d} b_i u^i$$

Example: $x(u) = 7u^3 + 3u^2 - 4u + 6$, $y(u) = u^3 - 4u^2 + 6$ is a **parametric polynomial curve of degree 3**, a **cubic** parametric polynomial.

Why polynomials are nice: They are easy to **evaluate** and easy to **differentiate**.

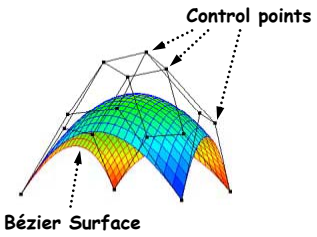
Matrix Representation: A convenient symbolic form.

Can express $x(u) = 7u^3 + 3u^2 - 4u + 6$ as: $x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 3 \\ -4 \\ 6 \end{bmatrix}$.

Bézier Curves

Bézier Curves:

- Developed in the 1960's by **Bézier** at **Renault** and **de Casteljau** at **Citroën**.
- Given **n control points**, this is a parametric polynomial curve of **degree n-1**.
- An elegant method for defining a curve that is defined by a sequence of **control points**.
- Has a number of **nice geometric properties**.
- Generalizes readily to **surfaces**.

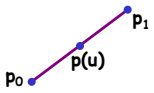


Bézier Curves: Derivation

Iterated Interpolation: The basic idea behind Bézier curves.

- Given two points $\mathbf{p}_0$ and $\mathbf{p}_1$, how can we define a **parametric curve** $\mathbf{p}(u)$ between them?
- Easy. Just use **linear interpolation**:

$$\mathbf{p}(u) = (1-u)\cdot\mathbf{p}_0 + u\cdot\mathbf{p}_1 \qquad \text{for } 0 \le u \le 1.$$



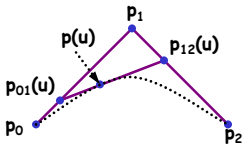
- Each point on this line segment is the **convex combination** (weighted average) of the two **control points**.

Bézier Curves: Derivation

Taking this one step further:

- Given three points $\mathbf{p}_0$, $\mathbf{p}_1$, and $\mathbf{p}_2$, how can we define a **parametric curve** $\mathbf{p}(u)$ between them?
- **de Casteljau's idea**:
  - Interpolate between $\mathbf{p}_0$ and $\mathbf{p}_1$. Call the result $\mathbf{p}_{01}(u)$.
  - Interpolate between $\mathbf{p}_1$ and $\mathbf{p}_2$. Call the result $\mathbf{p}_{12}(u)$.

$$\mathbf{p}_{01}(u) = (1-u)\cdot\mathbf{p}_0 + u\cdot\mathbf{p}_1 \qquad \mathbf{p}_{12}(u) = (1-u)\cdot\mathbf{p}_1 + u\cdot\mathbf{p}_2$$



  - Finally, interpolate between $\mathbf{p}_{01}(u)$ and $\mathbf{p}_{12}(u)$.

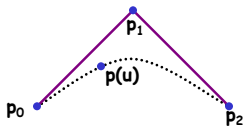
$$\begin{aligned}\mathbf{p}(u) &= (1-u)\cdot\mathbf{p}_{01}(u) + u\cdot\mathbf{p}_{12}(u)\\ &= (1-u)\cdot((1-u)\cdot\mathbf{p}_0 + u\cdot\mathbf{p}_1) + u\cdot((1-u)\cdot\mathbf{p}_1 + u\cdot\mathbf{p}_2)\\ &= (1-u)^2\cdot\mathbf{p}_0 + 2(1-u)u\cdot\mathbf{p}_1 + u^2\cdot\mathbf{p}_2\end{aligned}$$

Bézier Curves: Blending

Blending:

- The result is the **Bézier curve of degree 2**:

$$\mathbf{p}(u) = (1-u)^2\cdot\mathbf{p}_0 + 2(1-u)u\cdot\mathbf{p}_1 + u^2\cdot\mathbf{p}_2.$$



- We can express $\mathbf{p}(u)$ as a parametric blending of the control points $\mathbf{p}_0$, $\mathbf{p}_1$, and $\mathbf{p}_2$,

$$\mathbf{p}(u) = b_{02}(u)\cdot\mathbf{p}_0 + b_{12}(u)\cdot\mathbf{p}_1 + b_{22}(u)\cdot\mathbf{p}_2.$$

by the **Bézier blending functions** (of degree 2):

$$b_{02}(u) = (1-u)^2 \qquad b_{12}(u) = 2(1-u)u \qquad b_{22}(u) = u^2.$$

Convex Hull Property:

- Observe that the blending functions **sum to 1** for any u and are **nonnegative** if $0 \le u \le 1$. That is, $\mathbf{p}(u)$ is a **convex combination** of the control points.
- A **Bézier curve** lies within the **convex hull** of its **control points**.

Bézier Curves: Four Control Points

Bézier Curve for 4 control points P_0, P_1, P_2, P_3.

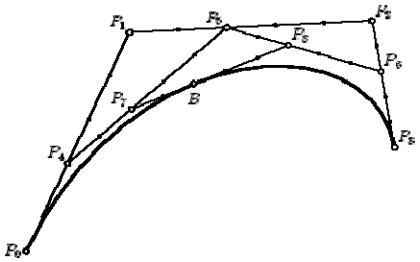


Image source: http://www.ursoswald.ch

Bézier blending functions of degree 3.

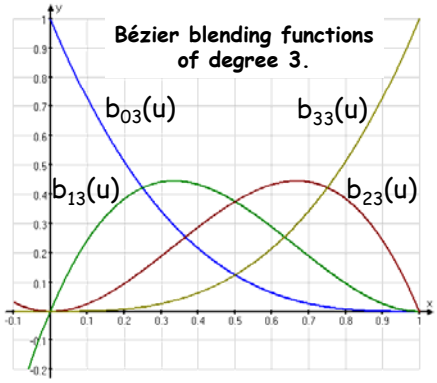


Image source: MIT

Bézier Curves: The General Case

Generalization:

- A Bézier curve of degree d over d+1 control points $\mathbf{p}_0$, $\mathbf{p}_1$, …, $\mathbf{p}_d$ is defined as follows:

$$\mathbf{p}(u) = \sum_{0 \le j \le d} b_{jd}(u)\mathbf{p}_j$$

where $b_{jd}(u)$ is the j^{th} **Bézier blending function** of degree d.

Generalizing the Blending Functions: The **Bernstein Polynomials**

Degree 0: $b_{00}(u) = 1$

Degree 1: $b_{01}(u) = 1\cdot(1-u)$ $b_{11}(u) = 1\cdot u$

Degree 2: $b_{02}(u) = 1\cdot(1-u)^2$ $b_{12}(u) = 2\cdot(1-u)u$ $b_{22}(u) = 1\cdot u^2$

Degree 3: $b_{03}(u) = 1\cdot(1-u)^3$ $b_{13}(u) = 3\cdot(1-u)^2u$ $b_{23}(u) = 3\cdot(1-u)u^2$ $b_{33}(u) = 1\cdot u^3$

… (the coefficients follow **Pascal's triangle**)

Degree d: $b_{jd}(u) = \binom{d}{j}(1-u)^{d-j}u^j$ where $\binom{d}{j} = \frac{d!}{j!(d-j)!}$

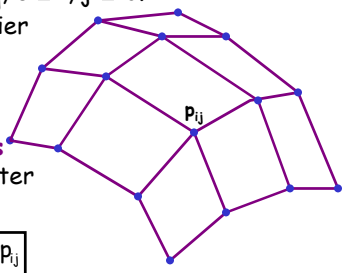
From Curves to Surfaces

We know how to define a **Bézier curve**. Can we generalize this to generate **Bézier surfaces**?

Tensor Product Construction: (of a Bézier surfaces of degree d)

- We have **two parameters** u and v.
- Rather than d+1 control points, let us assume that we have a **(d+1)×(d+1) mesh of control points**, $\mathbf{p}_{ij}$, $0 \le i, j \le d$.
- For each fixed i, we can define the Bézier curve for $\mathbf{p}_{i0}, \mathbf{p}_{i1}, \ldots, \mathbf{p}_{i,d}$ the result is

$$p_i(u) = \sum_{0 \le j \le d} b_{jd}(u) p_{ij}$$



- We can then apply the **blending process** to $\mathbf{p}_0(u), \mathbf{p}_1(u), \ldots, \mathbf{p}_d(u)$ using the parameter v. The result is a **Bézier surface**:

$$p(u,v) = \sum_{0 \le i \le d} b_{id}(v) p_i(u) = \sum_{0 \le i \le d} \sum_{0 \le j \le d} b_{id}(v) b_{jd}(u) p_{ij}$$

Other Parametric Modeling Methods

Bézier Curves/Surfaces:

- **Advantages**:
  - Mathematically **elegant**
  - Surfaces are **arbitrarily smooth** – derivatives of all degrees are continuous throughout the curve.
- **Disadvantages**:
  - Many control points implies **high degree polynomials**
  - **Difficult** to **evaluate** and control "**wiggling**"

B-Splines:

- A **piecewise approach** based on **Bézier** curves and surfaces.
- Arbitrarily **large numbers** of control points with **fixed degree**.
- Easy to evaluate and wiggling is less of a problem.

NURBS: Non-uniform rational B-Splines

- Generalizes B-Splines as a **rational function** (ratio of two B-splines).
- **Very powerful** and **widely used** in modelers.

Modeling Context

Context Modeling

Environmental context:

A sense of location:

- Indoor/outdoor
- Urban/rural.

Complexity can be HUGE:

- **Man made**: Cities, buildings.
- **Natural**: Forests, clouds, mountains.
- **Mobile**: Crowds of people, herds of animals.

Visual detail but not functional accuracy:

- Objects are part of the **background**.
- Do not require complex interactivity.
- Visual details need not be accurate.

Landscapes

http://www.xfrogdownloads.com/greenwebNew/news/images/Eco_CC04.jpg

Cityscapes

Procedurally generated cities: Roads, buildings, trees, …

Muller, Wonka et al, SIGGRAPH 2006

Battlescapes

Oc-zone.com

Animalscapes

James and Twigg, SIGGRAPH 2005

Animalscapes

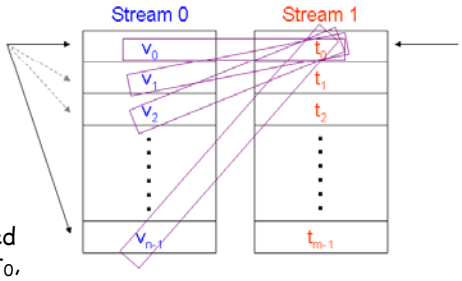
James and Twigg, SIGGRAPH 2005

Geometry Instancing using Vertex Shaders

Feature in Vertex Shader 3.0 model.

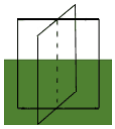
How it works:

- Given n vertices in $stream_0$ and m transformations in $stream_1$.
- Set the frequency of the vertex stream to m and the frequency of the transformation stream to 1.
- Vertex shader is first invoked with v_0t_0 followed by v_1t_0, v_2t_0, …
- After all the vertices for the first transformation (t_0) have been processed, the pointer to vertex stream ($stream_0$) is reset and the pointer to transformation stream ($stream_1$) is incremented to t_1.

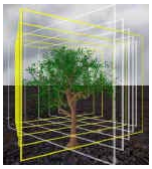


Billboards

Billboards: The illusion of depth from flat objects.



- Render a small number (1 or 2) **intersecting polygons** with **transparent textures**.
- Switch between **views** depending on the **direction** of view.



- Images can be photographic, pre-rendered, or dynamically created.
- Only works if the object will **not be viewed close up**, from above, or below.
- Great when **visual detail is important**, but there for context: clouds, trees, …

- Recent work in **hierarchical billboards**: Enhanced realism as distance decreases.

http://www.vterrain.org/

Modeling Illumination

Modeling Illumination

Functions of Lighting:

- **Illumination**: Simple ability to see what needs to be seen.
- **Revelation of form**: Adding realism to low-realism models (such as billboards).
- **Focus**: Directing the viewer's attention to a desired region of focus.
- **Mood**: Setting the tone of a scene (friendly, dangerous, pastoral).
- **Location and time of day**: Blues suggest night time. Orange and red can suggest a sunrise or sunset.
- **Plot**: A lighting event may trigger or advance the action.

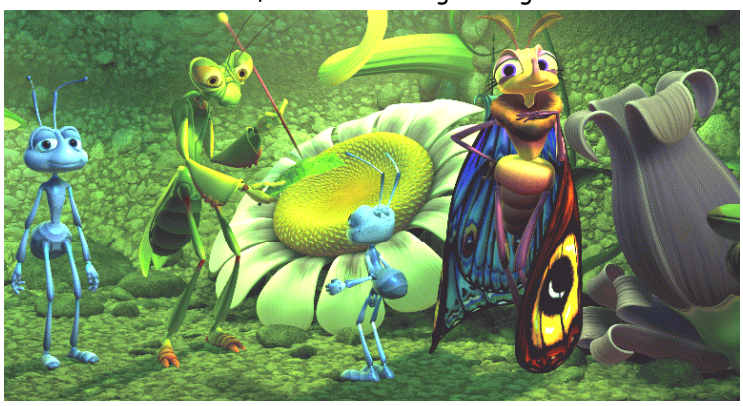
Traditional Theatrical Stage Lighting:

- **Key light**: Principal (brightest) illumination source.
- **Fill light**: Fills in shadows, softens effect of the key light.
- **Back light**: Light directed towards viewer. Enhances silhouettes.

Modeling Illumination: High Key

High-Key Lighting:

- Uniformly **bright illumination**, usually with **soft shadows**.
- Creates a **comfortable**, non-threatening feeling.



Modeling Illumination: Low Key

Low-Key Lighting:

- Often used to produce sense of **danger**, threat.
- **Chiaroscuro**: Artistic technique of using strong light-dark contrast to create strong dramatic effect.

Modeling Illumination

Soft lights and shadows: For warmth.

Harsh lights and hard shadows: For sterile environments and loneliness.

Level of Detail

Level of Detail for Polygonal Models

Level of detail (**LOD**):

- Flexible methods **managing scene complexity**.
- Also known as polygonal simplification, geometric simplification, mesh reduction, multiresolution modeling, …
- A standard tool for balancing **rendering speed** with **visual fidelity**.

Basic Idea:

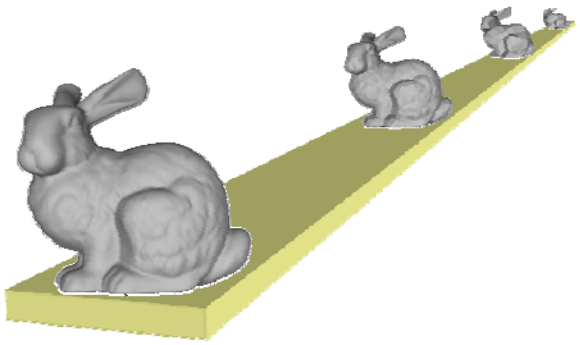
- Simplify the polygonal geometry of **small** or **distant** objects.

69,451 polys **2,502 polys** **251 polys** **76 polys**

Courtesy Stanford 3D Scanning Repository

Level of Detail

Distant objects are rendered with **coarser** LODs.



Static Levels of Detail

Static LOD:

- Create LODs for each object **separately** through preprocessing.
- At run-time, **select** object's LOD by **distance** (or similar criterion).

Advantages:

- **Easy to program**—decouples simplification and rendering.
- LOD creation need **not** affected by **real-time rendering issues**.
- **Simplifies run-time rendering**—need only select LODs.
- Fits well with **modern graphics hardware**:
  - Easy to compile each LOD into triangle strips, display lists, etc.
  - Render much faster (3-5x) than unorganized polygons.

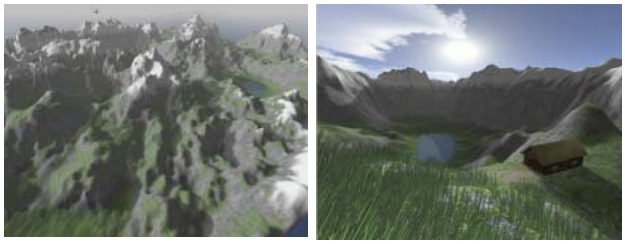
Disadvantages:

- Very large objects may simultaneously reside at **various distances**.
- Examples: Terrain fly-overs, volumetric isosurfaces, massive CAD models.

Dynamic Level of Detail

Dynamic LOD:

- Create data structure from which a desired **level of detail** can be **extracted at run time**.
- Improved **granularity** and control of detail.
- Can account for **subtler aspects of rendering**: lighting, viewing angle.
- Enables **better overall fidelity**.
- Enables drastic simplification of **very large objects**.



Design of Simplification

Simplification Operators

Error Evaluation

Perceptual Criteria

How To Simplify?

How to Simplify?

- Find a region of the mesh where a **local simplification** can be performed with a **minimal impact** on the object's geometry.
- **Repeat** this until desired degree of simplicity is achieved.

Simplification Operators: Common examples:

- **Edge Collapse**
- **Vertex-Pair Collapse**
- **Triangle Collapse**
- **Cell Collapse**
- **Vertex Removal**

Edge Collapse

Edge Collapse:

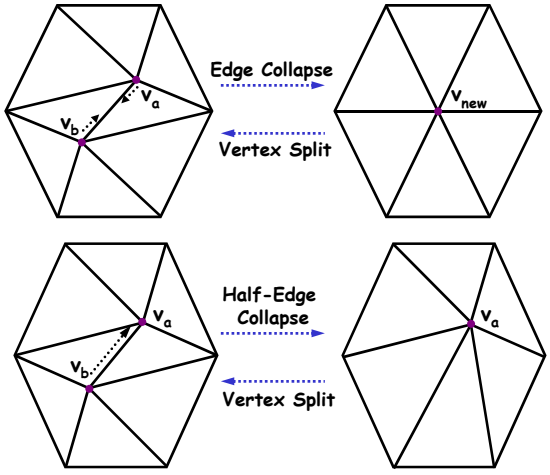
- Remove edge and merge endpoints at their **midpoint**.

Half-Edge Collapse:

- Collapse one endpoint into the other.
- Not symmetrical.
- Does not generate any new vertices.

Inverse Operator:

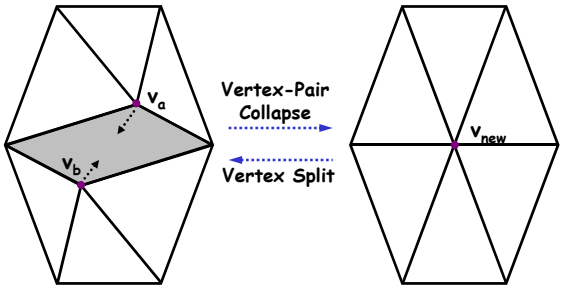
- Vertex split.



Vertex-Pair Collapse

Vertex-Pair Collapse:

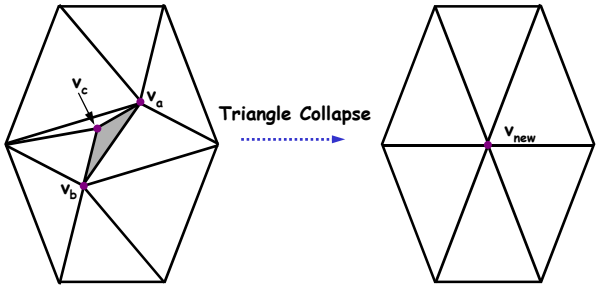
- Can remove a "**hole**" in mesh.
- Useful for **topology simplifications**.



Triangle Collapse

Triangle Collapse:

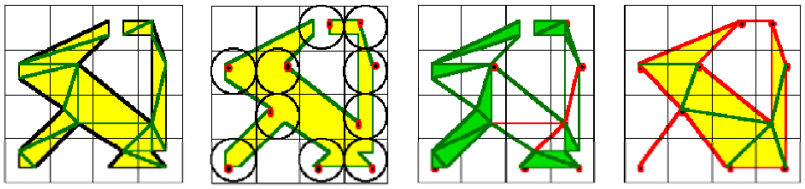
- Collapse a **single triangle** from the mesh, replacing it with its centroid.
- **Reduces height** of the hierarchy.
- Topologically equivalent to **two edge collapses**.



Cell Collapse

Cell Collapse:

- Overlay a **grid** on top of the mesh.
- Collapse all vertices **within each grid square** to a single vertex.
- **Simple** to implement; **robust**; relatively **poor visual quality**.

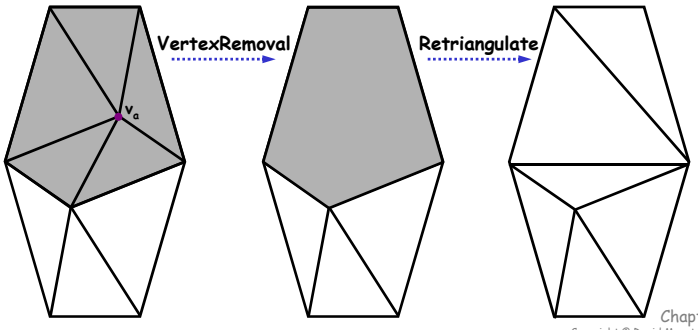


Rossignac & Borrel 1993

Vertex Removal

Vertex Removal:

- **Remove** all triangles **incident** to the given vertex, forming a "hole".
- **Retriangulate** the hole. (There are many possibilities).
- E.g., connect all to one vertex: Equivalent to **half-edge collapse**.
- **Beware**: Retriangulation is **not** generally possible. Ideally surrounding triangles nearly co-planar and form a convex ring.



Design of Simplification

Simplification Operators

Error Evaluation

Perceptual Criteria

Why Measure Error?

Guide simplification process:
- Making **better choices** produces better simplifications.

Know quality of results:
- Object-space **error bounds describe quality**.

Know when to show a particular LOD:
- **Which LOD** to achieve a desired screen-space **error**.

Balance quality for large environments:
- **What error bound** to achieved a desired **polygon count**.

Geometric Error Measures

Objectives:

- Promote accurate 3D **shape preservation**.
- Preserve **screen-space shape**:
  - Silhouettes
  - Pixel coverage

Error Metrics:

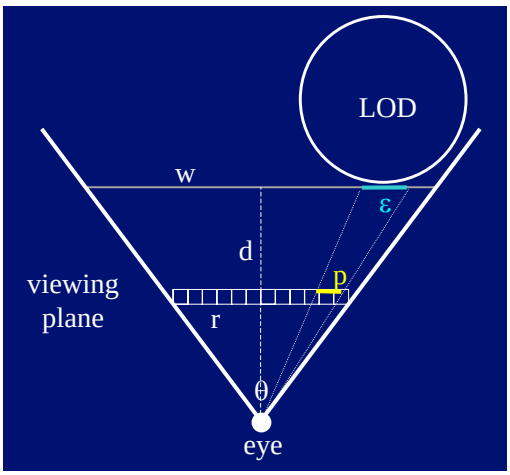
- **Vertex-Vertex Distance**
- **Vertex-Plane Distance**
- **Point-Surface Distance**
- **Surface-Surface Distance**

Screen-space Geometric Error

Screen-space error:

- What is the **number of image pixels** covered by a triangle of diameter ε in object space?
- Let:
  - ε: **element size**
  - w: **width of view frustum** at object depth
  - d: **perpendicular distance** to object plane
  - r: **width of the view plane**
  - θ: **field-of-view** angle
  - p: **number of pixels**

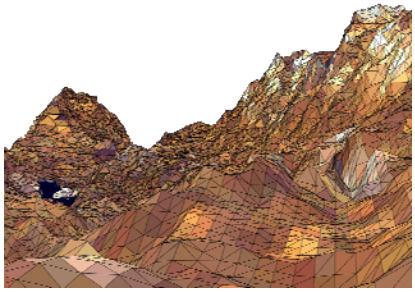
$$p = \frac{\varepsilon r}{w} = \frac{\varepsilon r}{2d \tan\frac{\theta}{2}}$$

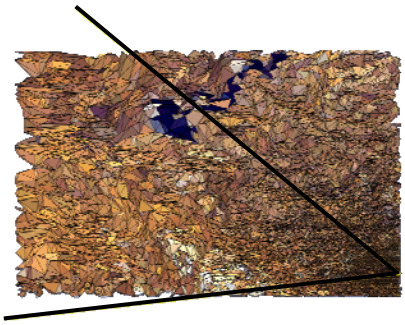


Screen-space Error Threshold

Nodes chosen by projected area:

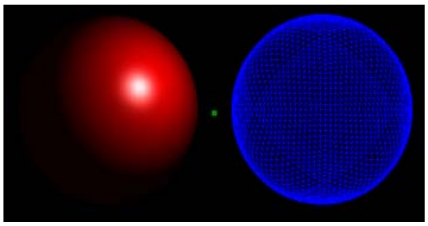
- User sets desired **screen-space size** threshold.
- Nodes that are **larger** than threshold are **unfolded**.



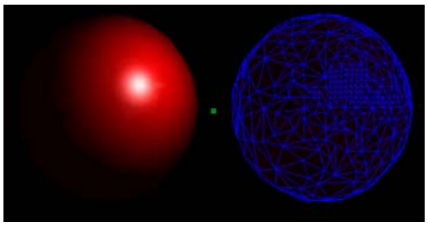


Illumination-Dependent Detail

Regions of **high illumination variation** are refined.



8192 triangles

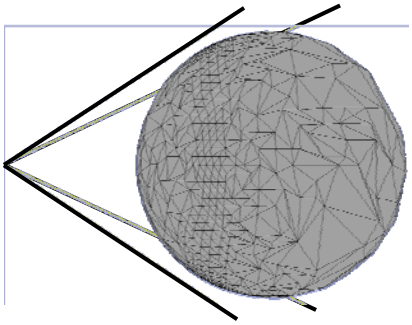


537 triangles

Silhouette Preservation

Retain more detail near silhouettes:

- Human visual system is **very sensitive to edges**.
- A **silhouette node** supports triangles on the **visual contour**.
- Use **tighter screen-space thresholds** when examining silhouette nodes.



Design of Simplification

Simplification Operators

Error Evaluation

Perceptual Criteria

Switching LOD

Primary LOD selection criteria:

- **Distance or Size**
- **Velocity**
- **Eccentricity**
- **Depth of Field**

Additional LOD constraints:

- **Fixed-frame rate schedulers** (reactive or predictive)
- **Hysteresis** (switching lag)
- **Priority schemes**
- **Alpha-blended transitions** (fading regions)
- **Geomorph transitions** (morph geometry)

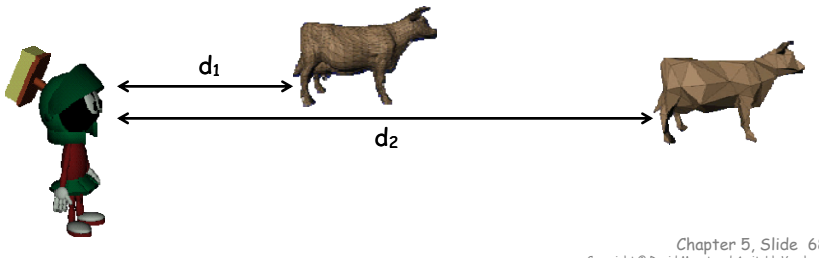
Distance LOD

Distance LOD:

- Select resolution based upon the **distance** between an element and the viewer, i.e. **coarser resolution** for **distant objects**.

Features:

- Simple to calculate (3-D Euclidean distance) - Good
- Scale dependent - Bad
- Resolution dependent - Bad
- Field-of-view dependent - Bad



Size LOD

Size LOD:

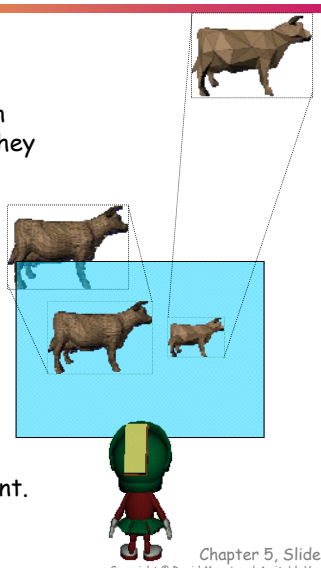
- Select resolution based upon the **projected screen size** (or area) of an element. Objects appear smaller as they move further away.

Features:

- Requires 3-D → 2-D projection - Bad
- Scale invariant - Good
- Resolution invariant - Good
- Field-of-view invariant - Good

Computation:

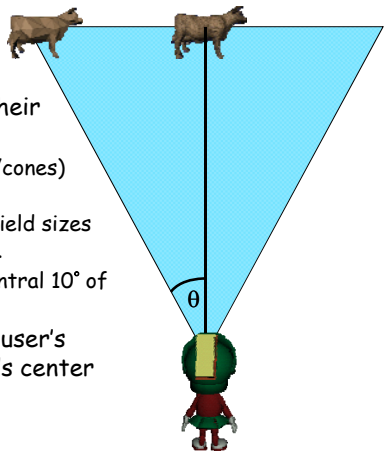
- Bounding **spheres or ellipsoids** used instead of boxes as more efficient to calculate projected extent.



Eccentricity LOD

Eccentricity LOD:

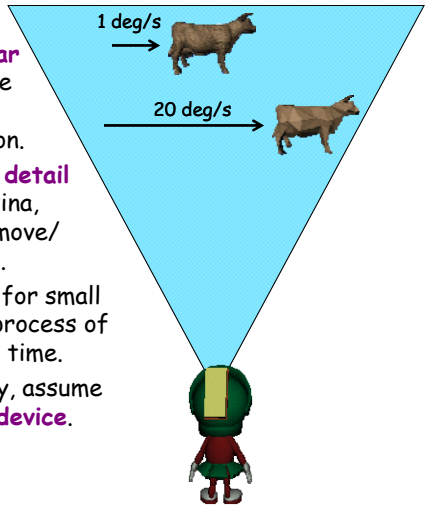
- Resolution is selected based on the degree to which an element exists in the **visual periphery**.
- Humans can resolve **less detail** in their **peripheral** field due to:
  - More retinal photoreceptors (rods/cones) near fovea.
  - Retinal and cortical cell receptive field sizes increases linearly with eccentricity.
  - 80% of cortical cells devoted to central 10° of vision.
- Use **eye-tracking system** to track user's gaze or assume user looking towards center of display.



Velocity LOD

Velocity LOD:

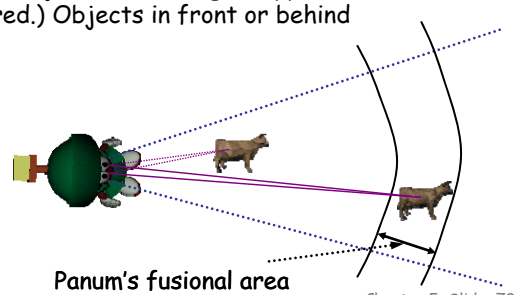
- Resolution based upon the **angular velocity** of an element across the visual field, i.e. faster moving objects appear in lower resolution.
- Humans can resolve **less spatial detail** in objects **moving** across the retina, causing objects to blur as they move/ rotate, or the user's gaze moves.
- It is believed visual information for small features are **destroyed** by the process of integrating stimulus energy over time.
- Without eye-tracking technology, assume **angular velocity** across **display device**.



Depth of Field LOD

Depth-of-Field LOD:

- Resolution of element depends on the **depth of field focus** of the user's eyes, i.e. objects outside the **fusional area** appear in lower detail.
- Under **binocular vision**, both eyes converge on object at certain distance in order to focus retinal image.
- **Panum's fusional area**: Objects in this region appear **in focus**. (Physiologically measured.) Objects in front or behind are out of focus.
- Must **track both eyes** accurately to evaluate convergence distance, or assume focus on **central object**.



Panum's fusional area

Visual Perception Software

Vermeer

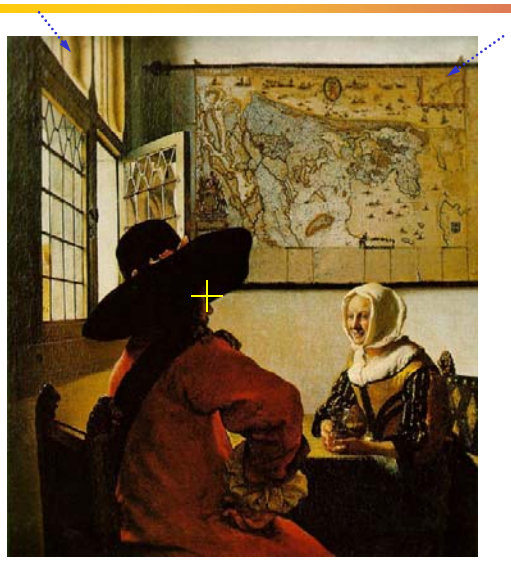
"Officer and Laughing Girl", 1658-60

120 x 135 degrees FOV

No eccentricity blurring

No velocity blurring

Visual Perception Software



Vermeer

"Officer and Laughing Girl", 1658-60

120 x 135 degrees FOV

Eccentricity blurring

No velocity blurring

Visual Perception Software

Vermeer

"Officer and Laughing Girl", 1658-60

120 x 135 degrees FOV

Eccentricity blurring

Velocity = 60 deg/s

Summary

Summary:

- Modeling Surfaces, Contexts, and Illumination
  - Implicit surfaces
  - Parametric surfaces
- Level of Detail