




Object-Based Image Editing

Alan S. Cheney
William A. Barrett
Brigham Young University




Image Editing Work

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo





Image Editing Work

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping




Image Editing Work

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping
 - Beier & Neely '92 Field Morphing





Image Editing Work

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping
 - Beier & Neely '92 Field Morphing
 - Lee, et. al '94 Thin-plate Splines





Image Editing Work

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping
 - Beier & Neely '92 Field Morphing
 - Lee, et. al '94 Thin-plate Splines
- Region-of-Interest Methods
 - Image-based methods could be applied to regions-of-interest

 **Image Editing Work** 

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping
 - Beier & Neely '92 Field Morphing
 - Lee, et. al '94 Thin-plate Splines
- Region-of-Interest Methods
 - Image-based methods could be applied to regions-of-interest
 - Elder & Goldberg '98, '01



 **Image Editing Work** 

- Pixel-based Methods
 - Adobe Photoshop, GIMP, SuperGoo
- Image-based Methods
 - Wolberg '90-'00 Image Warping
 - Beier & Neely '92 Field Morphing
 - Lee, et. al '94 Thin-plate Splines
- Region-of-Interest Methods
 - Image-based methods could be applied to regions-of-interest
 - Elder & Goldberg '98, '01



 **Texture Synthesis Work** 

- Efros & Leung '99 Non-parametric Sampling 
- Wei & Levoy 2000 Faster with TSVD
- Harrison 2000 Resynthesizer plug-in
- Praun, et al. 2000 Lapped Textures 
- Liang, et al. '01 Real-Time – Patch-Based Sampling
- Efros & Freeman '01 Image Quilting 

 **Where does Object-Based Image Editing fit in?** 

Pixel-based Methods

↑

↓

Image-based Methods

 **Where does Object-Based Image Editing fit in?** 

SuperGoo, clone tool, nudge tool, etc.

↑

Object-Based Image Editing

↓

Image Warping, Morphing, Spline warping

 **Video** 



What we're going to cover:



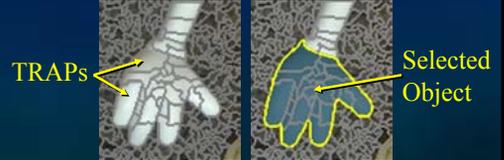
1. Object selection & representation
2. Object editing operations
3. Rendering
4. Background filling and texture painting
5. Applications



Object Selection



1. Segment image into *catchment basins* (TRAPs) (*Tobogganed Regions of Accumulated Plateaus*) using a Toboggan-based watershed algorithm.



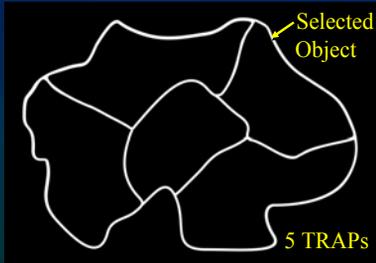
2. Manually "tag" TRAPs to select object



Object Representation



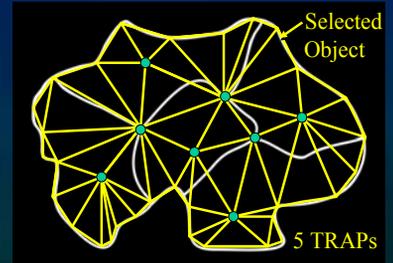
Fit Triangular Mesh to selected Object
- for efficient OpenGL Rendering



Object Representation



Fit Triangular Mesh to selected Object
- for efficient OpenGL Rendering



Object Representation



1. Recursive Divide & Conquer Polygonalization of Object Boundary



Object Representation



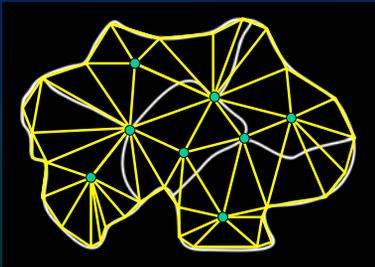
2. Insert Nodes (corners and basepoints)



BYU

Object Representation

3. Final Triangulation
– using Delaunay algorithm



BYU

BYU

Object Editing Operations

1. Object selection & representation
2. Object editing operations
3. Rendering
4. Background filling and texture painting
5. Applications

BYU

BYU

Object Editing Operations

2. Object editing operations
 - a. Direct Editing
 - b. Indirect Editing

BYU

BYU

Object Editing Operations

2. Object editing operations
 - a. Direct Editing
 - b. Indirect Editing

BYU

BYU

Stretch

1. After the object has been selected



BYU

BYU

Stretch

1. After the object has been selected
2. And vertices (x,y) identified



BYU

Stretch

1. After the **object** has been selected
2. And **vertices** (x,y) identified
3. An **anchor/pivot point** is specified

The diagram shows a rectangular object with a blue square anchor point on the left edge. The left edge is labeled with $-x$ and $+x$. The top-right corner is labeled (x,y) . The object is filled with vertical lines of varying colors.

Stretch

1. After the **object** has been selected
2. And **vertices** (x,y) identified
3. An **anchor/pivot point** is specified
4. And the user clicks on or near the **object** to stretch it with respect to the **anchor point**.

The diagram shows the same object as in the previous slide, but it is now stretched to the right. A black arrow points to the right from the right edge of the object.

Stretch

Object vertices (x,y) , are stretched

$$x' = x \left(1 + \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + \frac{\Delta y}{b_y} \right)$$

The diagram shows the object with vertical reference lines. A blue square anchor point is on the left. A point x is marked on the top edge. The lines are labeled "Reference Lines".

Stretch

Object vertices (x,y) , are stretched to (x',y') ,

$$x' = x \left(1 + \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + \frac{\Delta y}{b_y} \right)$$

where $(\Delta x, \Delta y)$ captures cursor movement
 b_x, b_y are object dimensions

The diagram shows the object stretched to a new width x' . The reference lines are now more widely spaced, indicating a uniform stretch. The text "Reference Lines show uniform stretch" is at the bottom.

Simple Linear Stretch

We now introduce an attenuation function, $A[i]$

$$x' = x \left(1 + A[i] \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + A[i] \frac{\Delta y}{b_y} \right)$$

The graph shows a constant horizontal line for $A[i]$ versus i . A green arrow points from the graph to the $A[i]$ term in the equation above.

The diagram shows the object stretched, with the text "With the same result ... for constant $A[i]$ " at the bottom.

Nonlinear Stretch

But if we vary $A[i]$

$$x' = x \left(1 + A[i] \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + A[i] \frac{\Delta y}{b_y} \right)$$

The graph shows a line with a positive slope for $A[i]$ versus i . A green arrow points from the graph to the $A[i]$ term in the equation above.

The diagram shows the object stretched, with the text "We get nonlinear stretch" at the bottom.

BYU **Nonlinear Stretch**

But if we vary $A[i]$

$$x' = x \left(1 + A[i] \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + A[i] \frac{\Delta y}{b_y} \right)$$

Less Stretch

BYU **Nonlinear Stretch**

But if we vary $A[i]$

$$x' = x \left(1 + A[i] \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + A[i] \frac{\Delta y}{b_y} \right)$$

More Stretch

BYU **Nonlinear Stretch**

And if we vary $A[i]$ even more

$$x' = x \left(1 + A[i] \frac{\Delta x}{b_x} \right) \quad y' = y \left(1 + A[i] \frac{\Delta y}{b_y} \right)$$

Extreme Nonlinear Stretch

BYU **Rotate**

Simple rotation matrix, with attenuated θ

$$(x', y') = \begin{bmatrix} \cos(\theta A[i]) & -\sin(\theta A[i]) \\ \sin(\theta A[i]) & \cos(\theta A[i]) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

BYU **Rotate** \rightarrow **Rotational Bend**

Simple rotation matrix, with attenuated θ

$$(x', y') = \begin{bmatrix} \cos(\theta A[i]) & -\sin(\theta A[i]) \\ \sin(\theta A[i]) & \cos(\theta A[i]) \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

BYU **Bend-Stretching**

- Calculate the bend based on stretch values

$$x' = x \left(1 + A[i] \frac{y_n - y_o}{b_x} \right) \quad y' = y$$

$$(x'', y'') = \begin{bmatrix} \cos(\theta A[i]) & -\sin(\theta A[i]) \\ \sin(\theta A[i]) & \cos(\theta A[i]) \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \end{bmatrix}$$

User Actions
Computation

BYU

Indirect Editing

2. Object editing operations

- Direct Editing
- Indirect Editing

BYU

Curve Deformers - Defaults

Anchor Point

Cursor Point

Length Stretch

Thickness Stretch

Rotation Falloff

Length

Thickness

Rotation

Reset

Reset

Reset

Reset

OK

BYU

Length Flattening

Anchor Point

Cursor Point

Length Stretch

Thickness Stretch

Rotation Falloff

Length ~ flat

Thickness

Rotation

Reset

Reset

Reset

Reset

OK

BYU

Bend-Stretching

Anchor Point

Cursor Point

Length Stretch

Thickness Stretch

Rotation Falloff

Length Leveling

Thickness

Rotation Level-off

Reset

Reset

Reset

Reset

OK

BYU

Indirect Editing - Interactive

BYU

Rendering

- Object selection & representation
- Object editing operations
- Rendering
- Background filling and texture painting
- Applications

BYU MIDDLEBURY COLLEGE

Antialiasing using OpenGL

Variables:

- Layer Height
- Layer Width

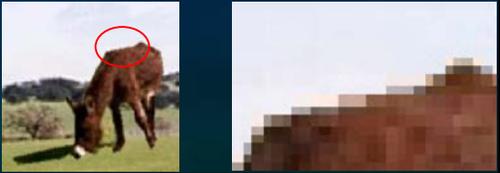


BYU MIDDLEBURY COLLEGE

Antialiasing using OpenGL

Variables:

- Layer Height
- Layer Width
- Number of Layers

BYU MIDDLEBURY COLLEGE

Background Filling

1. Object selection & representation
2. Object editing operations
3. Rendering
4. Background filling and texture painting
5. Applications

BYU MIDDLEBURY COLLEGE

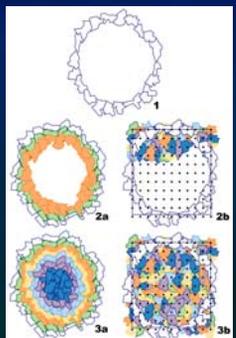
Background Filling

2 Methods

1. Scale Down Filling (2a-3a)
2. Gridded (2b-3b)

Sampling

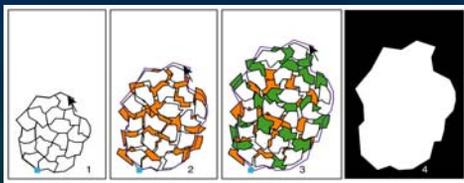
1. Automatic
2. User Selects



BYU MIDDLEBURY COLLEGE

Texture Preservation

- When an object stretches, if its texture is high in detail, the texture becomes overly smoothed, looking unnatural.
- To fix this, we keep object TRAP sizes constant, warping only their basepoint positions.



BYU MIDDLEBURY COLLEGE

Applications

1. Object selection & representation
2. Object editing operations
3. Rendering
4. Background filling and texture painting
5. Applications

