Infinite Resolution Textures
Alexander Reshetov  David Luebke
Motivation:

Computer Graphics at 30,000 feet
Motivation:

Computer Graphics at 30,000 feet
<table>
<thead>
<tr>
<th>DISTANCE</th>
<th>3D Models</th>
<th>Infinite Resolution Textures</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSETS</td>
<td><img src="image1.png" alt="Emoji" /></td>
<td><img src="image2.png" alt="Emoji" /></td>
</tr>
<tr>
<td></td>
<td><img src="image3.png" alt="Emoji" /></td>
<td><img src="image4.png" alt="Emoji" /></td>
</tr>
<tr>
<td></td>
<td><img src="image5.png" alt="Emoji" /></td>
<td><img src="image6.png" alt="Emoji" /></td>
</tr>
</tbody>
</table>
An alternative approach

<table>
<thead>
<tr>
<th>DISTANCE</th>
<th>ASSETS</th>
<th>2D Models</th>
<th>2D Textures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geometry</td>
<td>2D</td>
<td>3D</td>
<td>2D</td>
</tr>
</tbody>
</table>
An alternative approach

<table>
<thead>
<tr>
<th>DISTANCE</th>
<th>ASSETS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Models</td>
<td><img src="image1" alt="Emojis" /> <img src="image2" alt="Emojis" /> <img src="image3" alt="Emojis" /></td>
</tr>
<tr>
<td>2D Geometry</td>
<td><img src="image4" alt="Emojis" /> <img src="image5" alt="Emojis" /> <img src="image6" alt="Emojis" /></td>
</tr>
</tbody>
</table>
## Vector Graphics Evolution

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before ‘70s</td>
<td>All Graphics is 2D Vector Graphics replaced with hardware-accelerated raster graphics but continue existence in professional applications</td>
</tr>
<tr>
<td>2005+</td>
<td>GPU acceleration</td>
</tr>
<tr>
<td>Kilgard &amp; Bolz</td>
<td>Path Rendering</td>
</tr>
<tr>
<td>Ganacim et al.</td>
<td>Massively-parallel Vector Graphics</td>
</tr>
<tr>
<td>2016</td>
<td>Ellis et al.</td>
</tr>
</tbody>
</table>

Before ’70s, all computer graphics was actually 2D vector graphics. It was changed with a hardware-accelerated texture sampling. 2D graphics continue to proliferate in professional applications where it was rendered in software.

It all changed in this century, when GPUs become universal enough to accelerate rendering of smooth curves, as proposed by Loop & Blinn.

Kilgard and Bolz introduced a two-step Stencil, then Cover approach, allowing efficient GPU rendering of general vector textures. Ganacim et al. went further, employing an acceleration structure whose traversal enabled rendering parts of the image.

Now it is a part of Adobe products and you could also download NV Path Rendering SDK which is a part of GameWorks.

One of the most interesting – and unusual – papers at HPG was one by Ellis et al. who described a system that allows converting 3D scenes to vector graphics directly.
still a problem
<table>
<thead>
<tr>
<th>DISTANCE</th>
<th>ASSETS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D Geometry</th>
<th>no random sampling</th>
<th>still a problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kilgard &amp; Bolz</td>
<td>rasterization</td>
<td></td>
</tr>
<tr>
<td>Ganacim et al.</td>
<td>tiled rasterization</td>
<td></td>
</tr>
</tbody>
</table>
We aim at a more general approach seamlessly combining raster and vector representations.
IR texture = raster image + silhouettes @ grid

float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
IRT from an application standpoint

instead of

```cpp
float4 color = colorMap.SampleLevel(colorSampler, uv, lod);
```

use

```cpp
float4 color = colorMap.SampleLevel(colorSampler, uv + duv, lod);
```

IRT calculates \textit{duv} at runtime by evaluating distances to the precomputed silhouette edges

Just by tempering \textit{duv}, we can blend between

- IRT (@ closeups) and
- traditional textures at a distance
Prior Art

https://www.pinterest.com/mizzchanty/facebook-quotes
pinned from shar-enator.com
$\text{IRT} = \text{edges} + \text{raster image}$

- hi freq
- low freq

It can be filtered
I CAN'T BELIEVE YOU GOT PLASTIC SURGERY!!!
Prior Art circa 2005

- **Silmaps**
  *Pradeep Sen*

- **Bixels**
  *Jack Tumblin, Prasun Choudhury*

- **Vector Texture Maps**
  *Nicolas Ray et al*

- **Curvilinear Contours**
  *Stefan Gustavson*

- **Pinchmaps**
  *Marco Tarini, Paolo Cignoni*

- Piecewise-linear edges
- Always interpolating colors on the same side of the edge
- With a custom interpolation scheme
Prior Art circa 2005

• Silmaps
  Pradeep Sen

• Bixels
  Jack Tumblin, Prasun Choudhury

• Vector Texture Maps
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson

• Pinchmaps
  Marco Tarini, Paolo Cignoni

  • decompose the texture plane into patches with straight boundary segments
  • 10 patch functions
Prior Art circa 2005

• Silmaps
  Pradeep Sen

• Bixels
  Jack Tumblin,
  Prasun Choudhury

• Vector Texture Maps
  Nicolas Ray et al
  Curvilinear Contours
  Stefan Gustavson

• Pinchmaps
  Marco Tarini,
  Paolo Cignoni

• implicit cubic polynomials for edges
• binary classification function defines a patch
Prior Art circa 2005

- **Silmaps**
  *Pradeep Sen*

- **Bixels**
  *Jack Tumblin, Prasun Choudhury*

- **Vector Texture Maps**
  *Nicolas Ray et al*
  *Curvilinear Contours*  
  *Stefan Gustavson*

- **Pinchmaps**
  *Marco Tarini, Paolo Cignoni*

- a single quadratic silhouette edge per *pinchmap* texel
- use distance to the edge to compute new *uv*
pinchmaps  IRT
pinchmaps

IRT
Occam vs Einstein

- **Occam's Razor**
  the simpler one is usually better

- **Einstein Principle**
  “a scientific theory should be as simple as possible, but no simpler”
Pinchmaps are too simple

- pinchmap texels
- define an implicit quadratic curve,
- so all samples that have 4 pinchmap texels...
- ...will be resampled from the original texture

Issues

- No intersections

2nd degree only

- zero adjustment for all ‘outside’ samples \( \Rightarrow \) discontinuous \( \mathbf{duv} \)
To address these problems...

...IRT uses more evolved processing...

...that is easier to explain

legs-isential quandary
by Roger N. Shepard
First, we need curved edges
Those edges are split into segments and we create truncated Voronoi regions. These regions are offset from the curve by a specified distance (of a few pixels). It can also be reduced for the open-ended segments and bifurcated edges.
At run time, we just move the sample away from the edge. The samples outside Voronoi regions (×) will have zero duv.
opportunities

1. Temper* raster and vector modes just by scaling the texture coordinate adjustment \( \text{duv} \) using pixel/texel ratio as

   \[
   \text{float pixratio} = 0.5 \times \text{length(fwidth(uv * texdim))};
   \text{duv} \ast= \min(1, 2 \times (1 - \text{pixratio})) / \text{texdim};
   \]

2. Perform antialiasing in a single fetch by adjusting \text{lod}

3. Do whatever we like with it (like ‘soft landing’)

* having the elements mixed in satisfying proportions

http://www.merriam-webster.com/dictionary/tempered
soft landing

original

crisp edges ↔ tweak duv → smooth edges

trilinear
details
For each sample, we need:

- scalar distance to the curve \( d \)
- offset vector \( n_i \)
To compute $n_i$ we could

- interpolate $n_{12}$
- with weights $|w_{12}|$

Note: signs of $w_{12}$ can also be used to verify that the sample is in the curve’s Voronoi region.
Distance to the curve

• ∃ numerous prior art approaches
• To compute it even faster, we propose two algorithms:
  1. Implicit representation of cubic Bézier curves – using barycentric coordinates (savings: 6 terms instead of 10)
     (see also “Rendering Cubic Curves on a GPU with Floater's Implicitization” by Ron Pfeifle in JGT 2012)
  2. A quotient of two multivariate polynomials over variables that we choose (to make life easier)
     ≈ beefed up Phong interpolation in 1D
Using a value of implicit cubic polynomial as a distance proxy works near the curve...
...but breaks afar
\( \frac{P_{n+1}}{P_n} \) is stable by design

- unless strict reproduction of Bézier curves is required,
- it should be a method of choice
- since it is unconditionally stable; there are other interesting possibilities as well
Expectations vs Reality

The more you weigh, the harder you're to kidnap!!! Stay safe! Eat ice cream!
THE HARDER
TO K
Stay s
FOT I G
good
ȝ
IRT
raster
bad

IRT
raster
Plans