VGP352 – Week 9

- Agenda:
  - Interior mapping
  - Parallax textures
  - Displacement mapping

9-March-2010

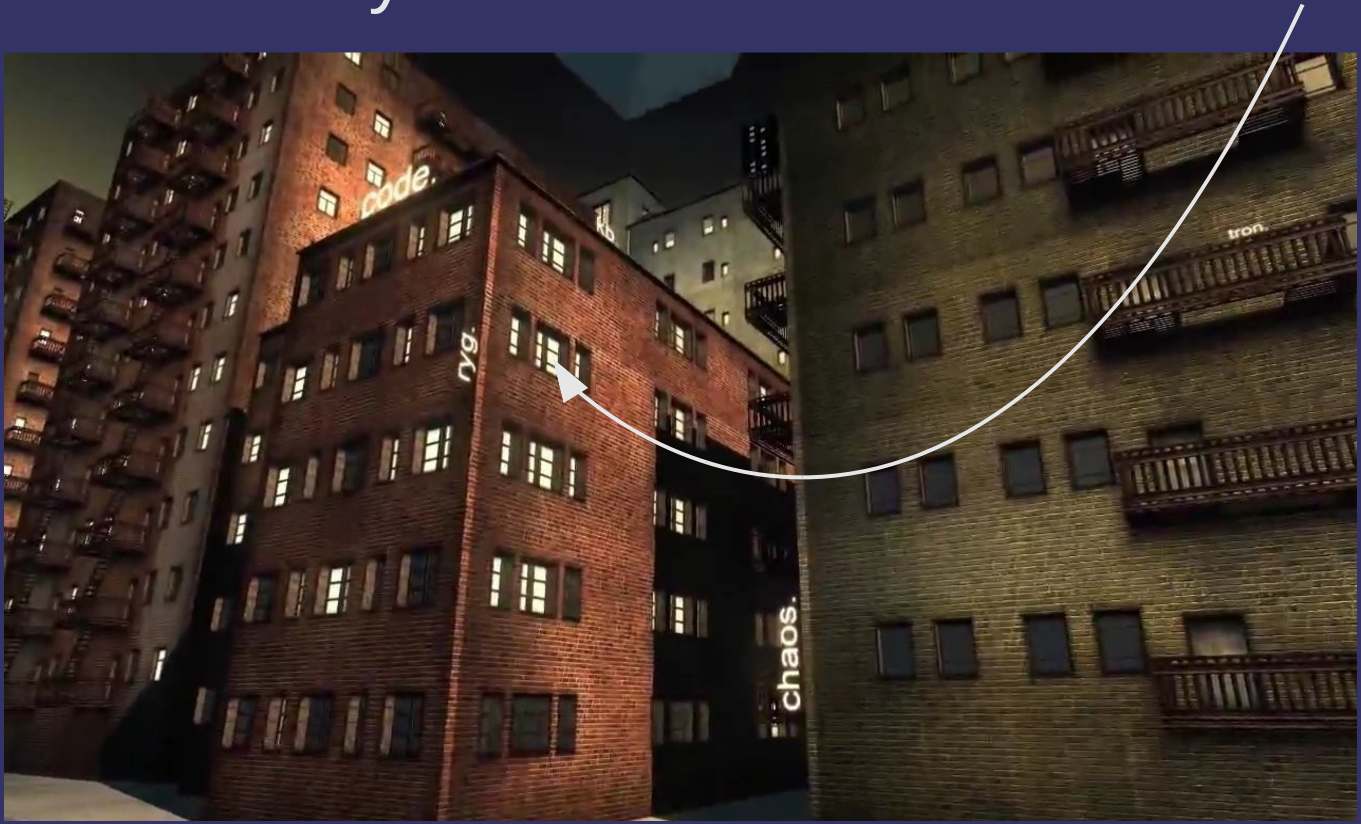
Interior Mapping

- Remember the excellent “debris” demo by Farbrausch?

code.
kb
ryg.
chaos.
tron.

Interior Mapping

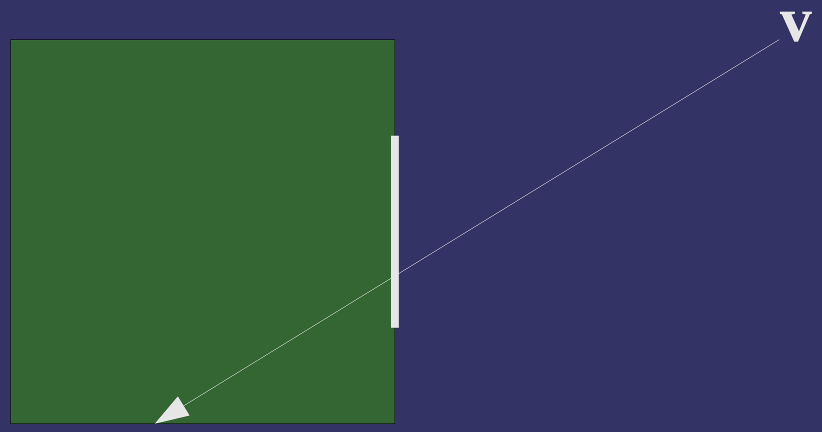
- Remember the excellent “debris” demo by Farbrausch?
  - Isn't it odd that you can't see inside the windows?



9-March-2010

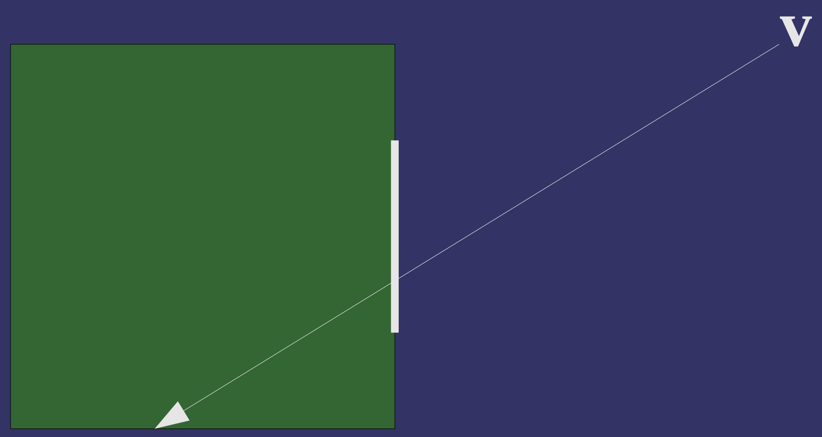
Interior Mapping

- Determine the location *inside* the building that is visible *without* adding geometry



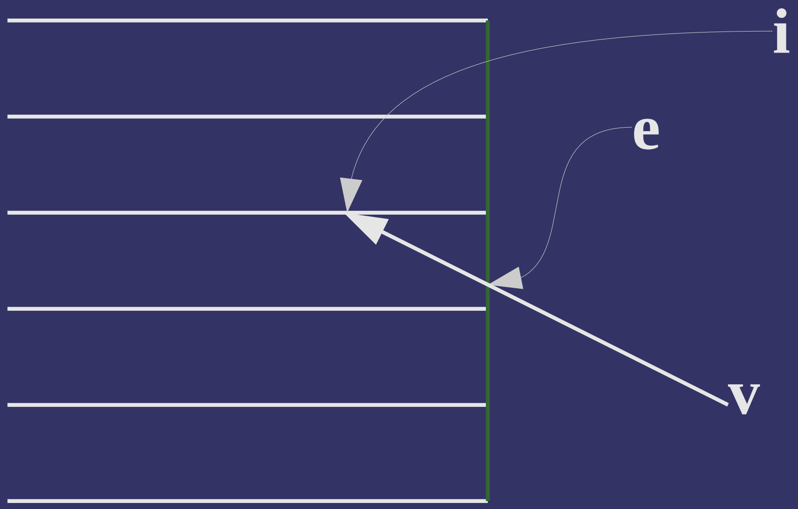
Interior Mapping

- Determine the location *inside* the building that is visible *without* adding geometry
  - The drawing suggests the answer: use raycasting
  - Create virtual walls, ceilings, and floors inside the building at regular intervals



Interior Mapping

- Calculate the point of ceiling intersection
  - Assume all calculations are in object space
  - Exterior intersection point and ray direction are given



$$\mathbf{i}_y = \left\lceil \mathbf{e}_y / h \right\rceil \times h$$

9-March-2010

Interior Mapping

- Parametric equation of V:

$$\mathbf{p} = \mathbf{v} + \hat{\mathbf{v}}\, t$$
$$\text{where } \hat{\mathbf{v}} = \mathbf{e} - \mathbf{v}$$

  - Calculate the value of t where $\mathbf{p}_{\mathrm{y}} = \mathbf{i}_{\mathrm{y}}$

$$\begin{aligned} \mathbf{i}_{\mathrm{y}} &= \mathbf{v}_{\mathrm{y}} + \hat{\mathbf{v}}_{\mathrm{y}}\, t \\ \mathbf{i}_{\mathrm{y}} - \mathbf{v}_{\mathrm{y}} &= \hat{\mathbf{v}}_{\mathrm{y}}\, t \\ \frac{\mathbf{i}_{\mathrm{y}} - \mathbf{v}_{\mathrm{y}}}{\hat{\mathbf{v}}_{\mathrm{y}}} &= t \end{aligned}$$

  - Use t to calculate the rest of $\mathbf{i}$

Interior Mapping

- Perform similar calculations for walls
  - The intersection with the smallest t is used
  - Use resulting $\mathbf{i}_{xy}$ to generate a texture coordinate
- Can add extra fake walls to represent items in the rooms
  - Textures for the fake walls should be mostly transparent
  - Has issues if the viewer can see in corners
    - See paper for more details

References

van Dongen, Joost, "Interior Mapping - A new technique for rendering realistic buildings." In *Computer Graphics International Conference* (CGI). 2008. http://interiormapping.oogst3d.net

Parallax Textures

- Normal / bump maps give shading cues to surface shape
  - No changes to silhouette
  - No self occlusion

Parallax Textures

- Normal / bump maps give shading cues to surface shape
  - No changes to silhouette
  - No self occlusion
- Parallax textures address the second problem
  - Does so by exploiting the parallax effect

Parallax Textures

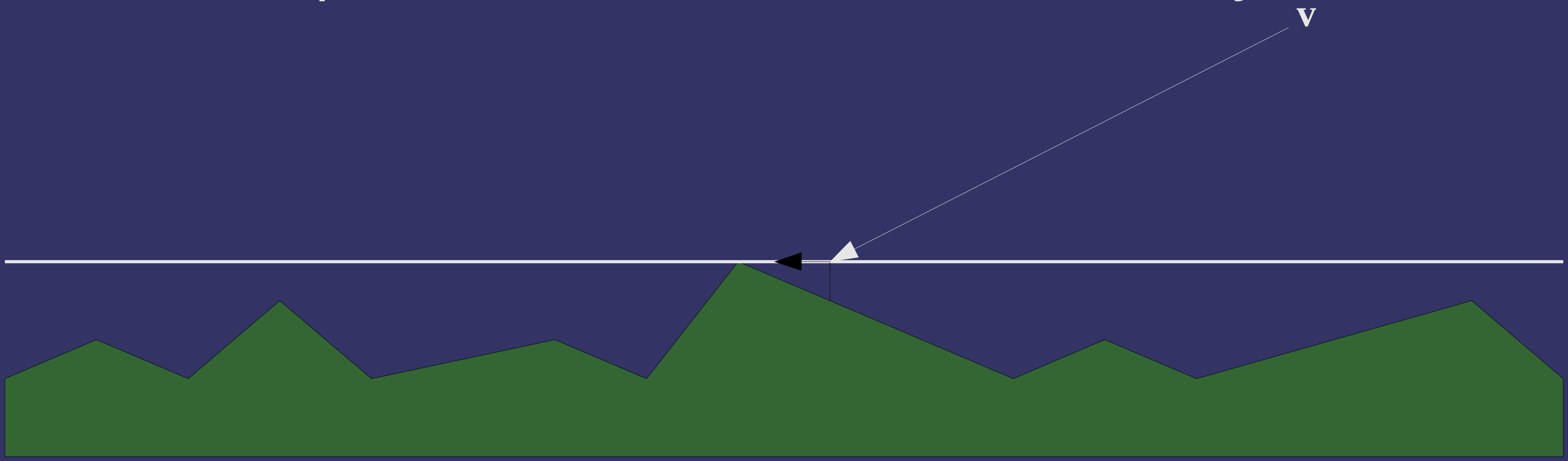
- From wikipedia:

  > Parallax is an apparent displacement or difference of orientation of an object viewed along two different lines of sight, and is measured by the angle or semi-angle of inclination between those two lines....Nearby objects have a larger parallax than more distant objects when observed from different positions, so parallax can be used to determine distances.

  - 2D side-scrolling games use this effect all the time
    - Nearer background objects scroll by faster than farther background objects

Parallax Textures

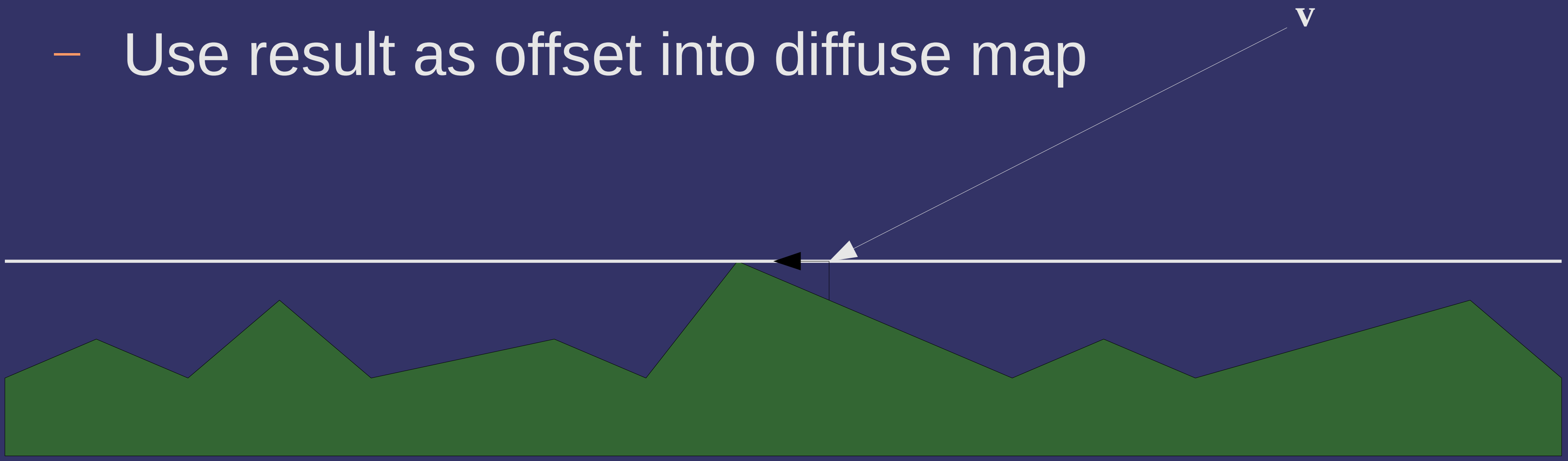
- Implement this for a 3D surface:
  - Use a height (bump) map to set per-fragment distance from viewer
  - As the viewer moves side-to-side *in surface space*, nearer portions of the texture will “scroll by faster”



9-March-2010

Parallax Textures

- At each fragment:
  - Sample the depth from the surface using the bump map
  - Use this value to scale projection of the view vector on to the surface
  - Use result as offset into diffuse map



Parallax Textures

- What could go wrong? What are the short comings?

Parallax Textures

- What could go wrong? What are the short comings?
  - Assumes a smoothly varying height field
    - Can't handle large displacements
    - Can't handle high-frequency data
  - Doesn't properly handle occlusion

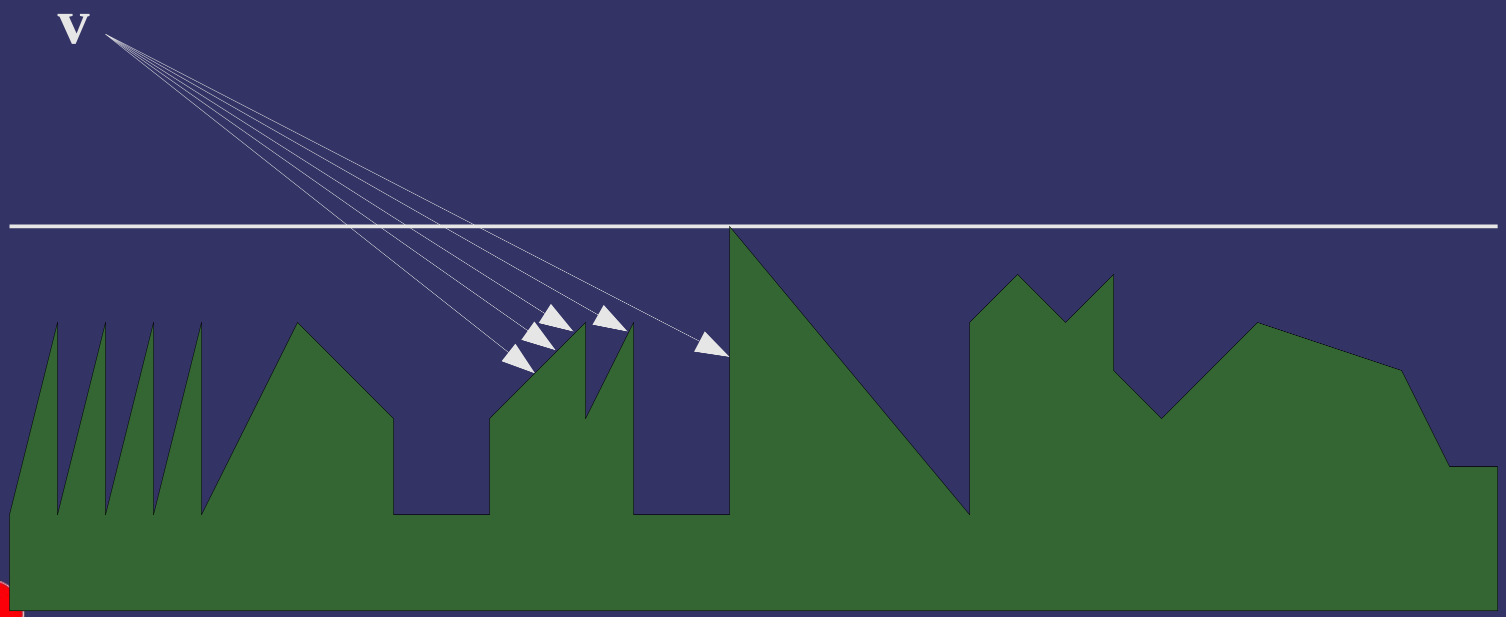
References

Kaneko, Tomomichi, Toshiyuki Takahei, Masahiko Inami, Naoki Kawakami, Yasuyuki Yanagida, Taro Maeda, and Susumu Tachi. 2001. “Detailed Shape Representation with Parallax Mapping.” In *Proceedings of the ICAT 2001 (The 11th International Conferences on Artificial Reality and Telexistence)*, Tokyo, December 2001, pp. 205 – 208. http://vrsj.t.u-tokyo.ac.jp/ic-at/ICAT2003/papers/01205.pdf

West, Mick. “Parallax Mapped Bullet Holes.” Game Developer, May 2006. http://cowboyprogramming.com/2007/01/05/parallax-mapped-bullet-holes/

Displacement Mapping

- We really want to raytrace into arbitrary volume data representing our surface
  - Would require a linear search through a volume texture *per fragment*



Displacement Mapping

- What if we knew, at every position, the distance to the nearest voxel?
  - As we walk the ray through voxel space, we could step by the distance to the nearest voxel
  - Reduces the search from n to $\log n$

Displacement Mapping

- Algorithm:

```
For some number of steps:
    distance = sample distance texture at
    position
    position += distance * direction
```

  - Dynamic branching hardware can end loop early if `distance` is below some preset threshold
  - `direction` is the normalized viewing direction vector
    - Must be rescaled from surface space to texel space

Displacement Mapping

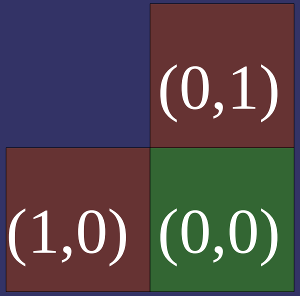
- Result of raytracing is a 3D position
  - Project the 3D position onto the surface
    - i.e., just use the x and y components
  - Use the resulting projection to sample texture and normal maps

Euclidean Distance Map

- Generate distance map using Danielsson's algorithm
  - Initialize a texture with (0, 0) for elements “inside” the surface or (∞, ∞) for elements outside
  - Perform 4 passes over the image propagating distances among neighbors
  - This is the 2D version... it can be trivially extended to 3D

Euclidean Distance Map

- Pass 1:
  - Move the mask top-to-bottom, left-to-right
  - The green element is the pixel being examined, the others are its neighbors
  - Add the specified offsets to the pixel distance values, store the minimum in the pixel

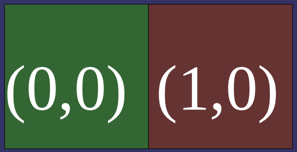


(0,1)

(1,0) (0,0)

Euclidean Distance Map

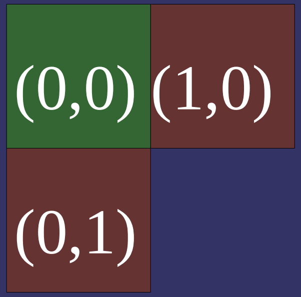
- Pass 2:
  - Move the mask top-to-bottom, right-to-left
  - The green element is the pixel being examined, the others are its neighbors
  - Add the specified offsets to the pixel distance values, store the minimum in the pixel



(0,0)	(1,0)

Euclidean Distance Map

- Pass 3:
  - Move the mask bottom-to-top, right-to-left
  - The green element is the pixel being examined, the others are its neighbors
  - Add the specified offsets to the pixel distance values, store the minimum in the pixel

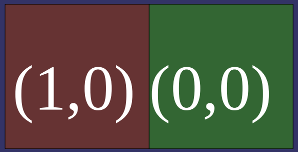


(0,0) (1,0)

(0,1)

Euclidean Distance Map

- Pass 4:
  - Move the mask bottom-to-top, left-to-right
  - The green element is the pixel being examined, the others are its neighbors
  - Add the specified offsets to the pixel distance values, store the minimum in the pixel



(1,0)	(0,0)

Euclidean Distance Map

- Final pass:
  - Convert the distance vectors to distance scalars

Displacement Mapping

- Caveats:
  - Take partial derivatives of input texture coordinate and use those when sampling the final texture
    - Otherwise the texture filtering will be wrong in weird ways
    - Use `dFdx()` and `dFdy()` functions

References

Donnelly, William. "Per-Pixel Displacement Mapping with Distance Functions" in Fernando, Randima (editor) GPU Gems 2, Addison Wesley, 2005. http://download.nvidia.com/developer/GPU_Gems_2/GPU_Gems2_ch08.pdf

Fabbri, R., Costa, L. F., Torelli, J. C., and Bruno, O. M. 2008. 2D Euclidean distance transform algorithms: A comparative survey. ACM Computing Surveys 40, 1 (Feb. 2008), 1-44. http://distance.sourceforge.net/

- You'll have to Google (with some effort!) for a live link to the actual paper. :(

Next week...

- Multiple render targets
- Deferred shading
- Quiz #4

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.