CMPSCI 691AD – General Purpose Computation on the GPU

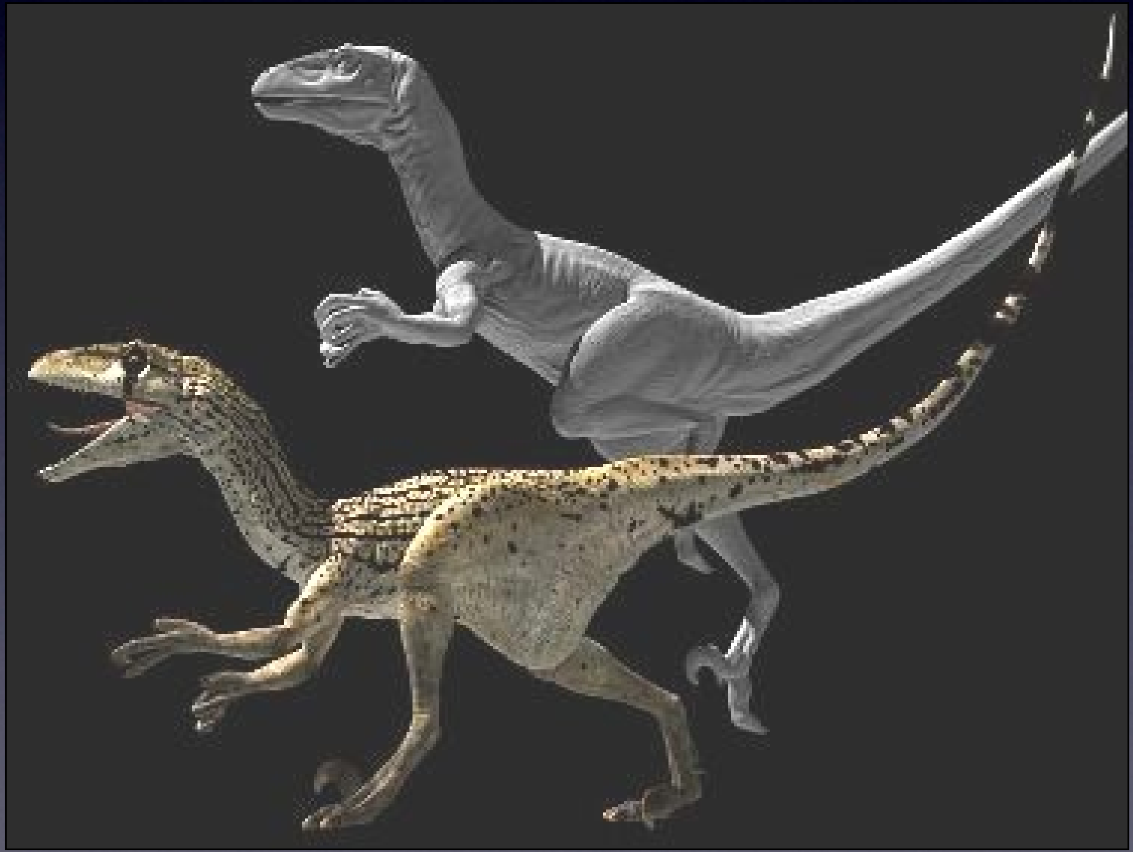
Spring 2009

Lecture 13: Texture Filtering

Rui Wang

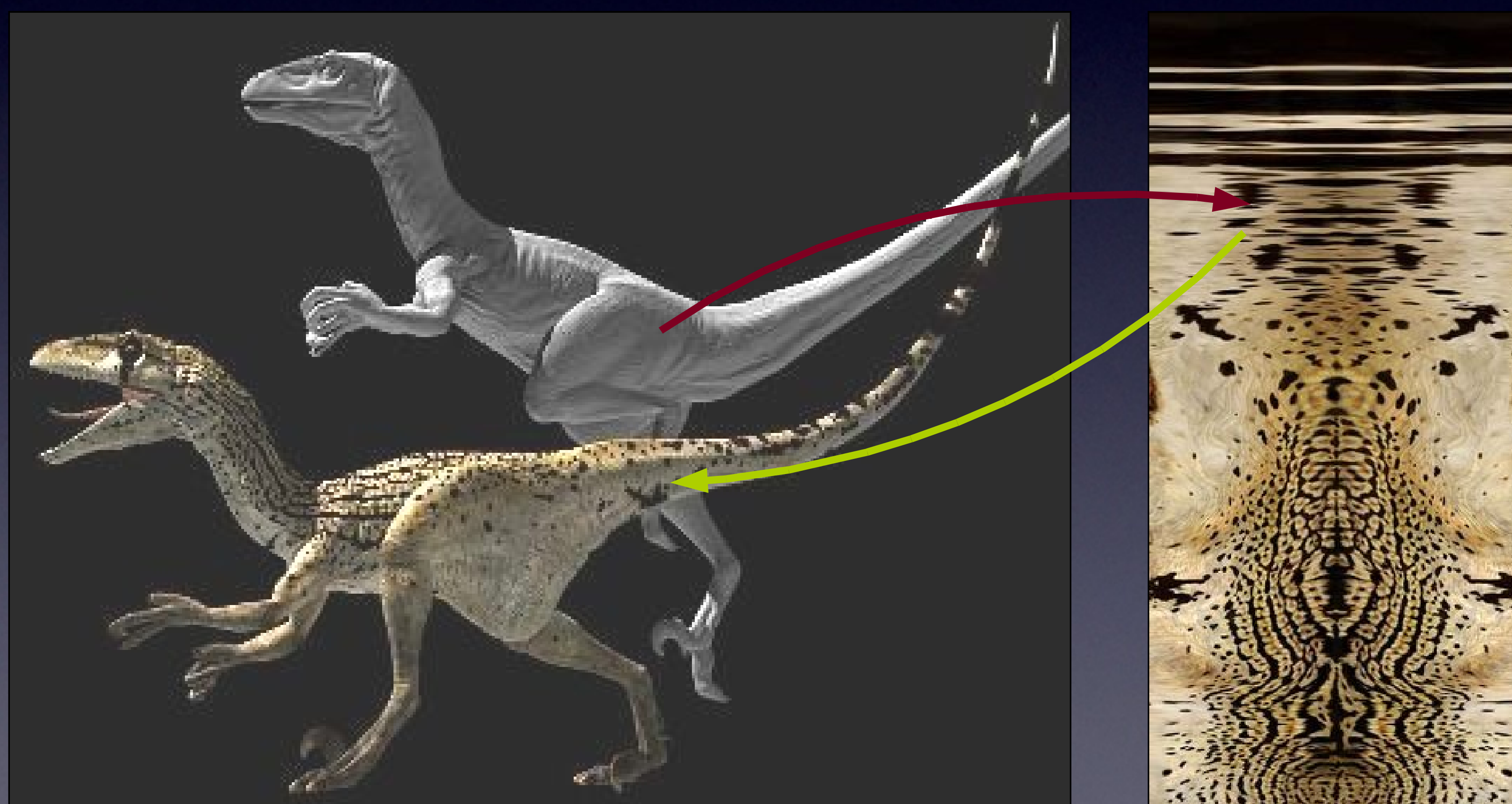
Textures

- Textures provide an efficient way to add surface details without increasing geometric complexity.



Parametrization, UV Mapping

- Determining how each point on the 3D model is mapped to a 2D texture: texture coordinates.



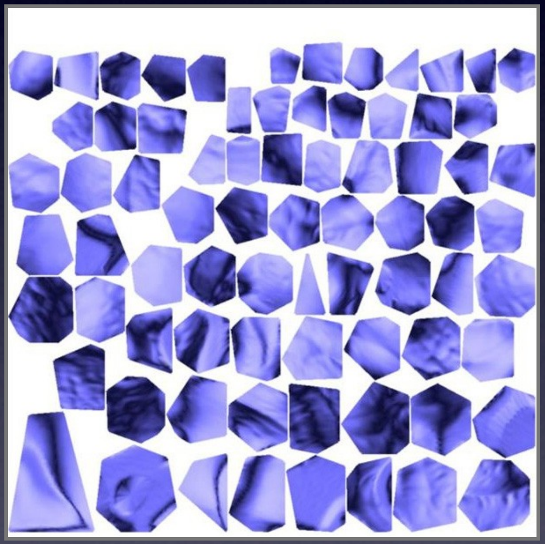
Parametrization, UV Mapping

- Determining how each point on the 3D model is mapped to a 2D texture: texture coordinates.
- In general a difficult problem:
  - Maintianing 3D coherence and continuity
  - Distortion

Parametrization, UV Mapping

- Determining how each point on the 3D model is mapped to a 2D texture.

Charts



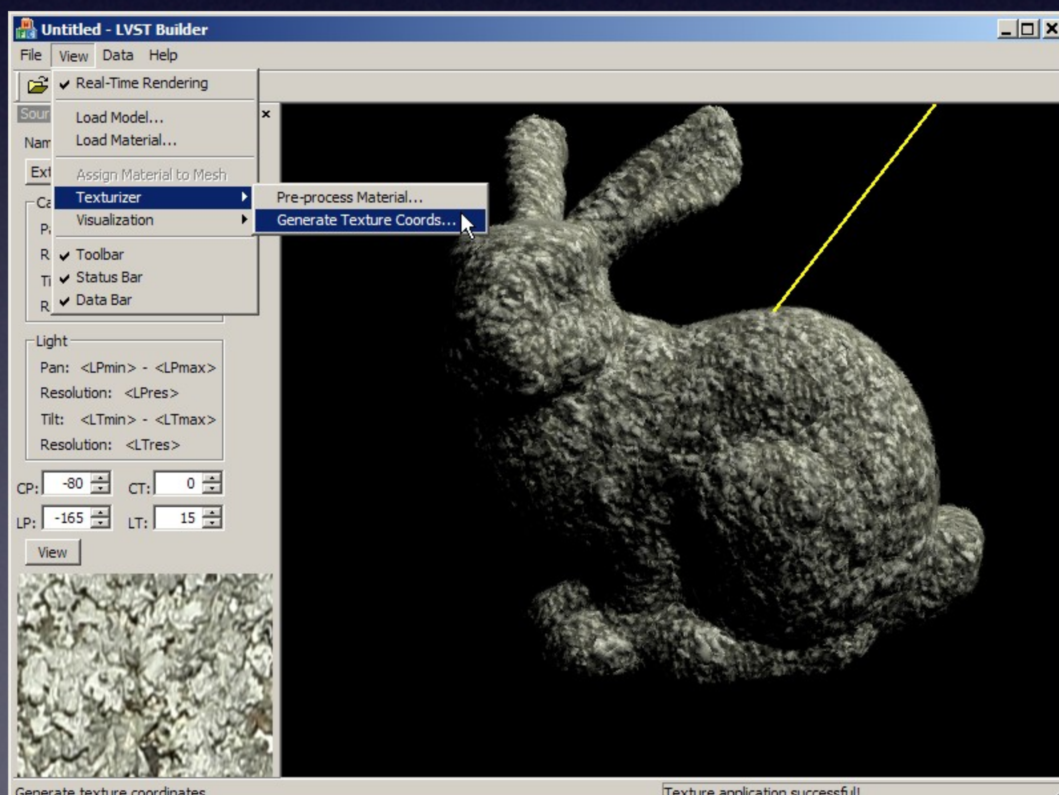
Atlas

Surface

[Sander 2001]

3D Textures

- Volumetric textures
- Can use 3D point to directly index into the texture, no parametrization needed.

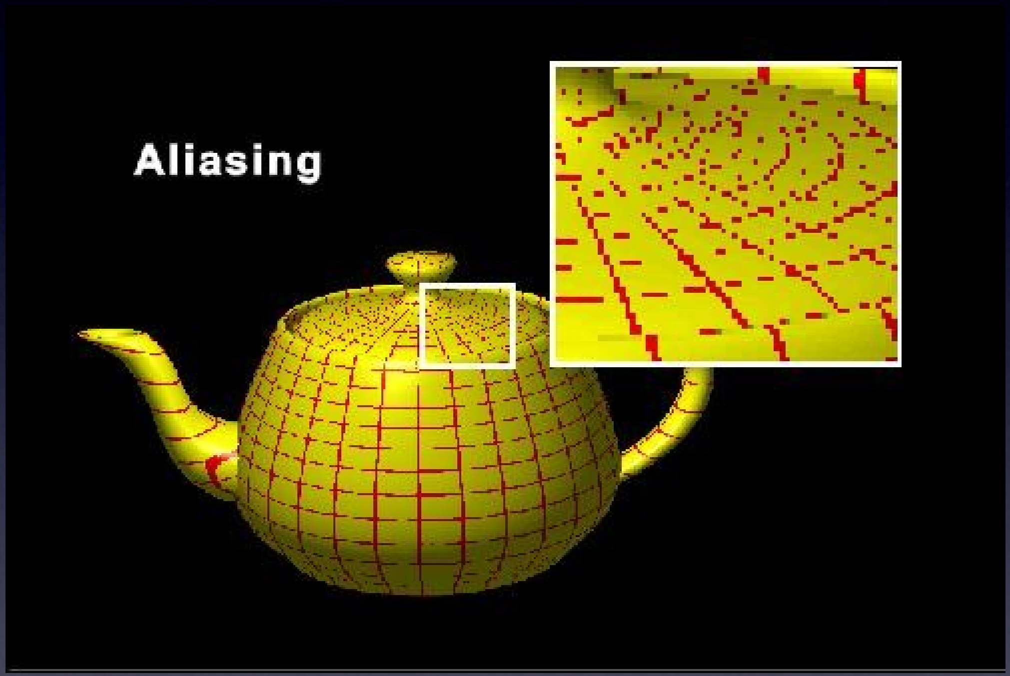


Texture Mapping

- How to apply texture mapping?
  - 1. Rendering to generate pixels
  - 2. At each pixel, figure out the texture coordinate
  - 3. Index into the texture map (using the texture coords) to obtain a texel value
  - 4. Apply the texel value to generate color for pixel.

Texture Map Aliasing

- Nearest sampling is very bad: aliasing.



Texture Map Aliasing

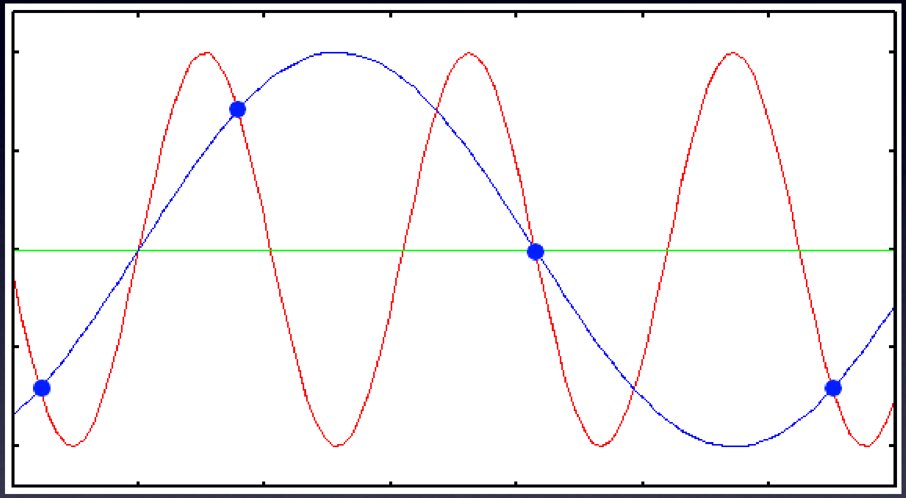
- Nearest sampling is very bad.
- (Bi)-linear interpolation improves, but still insufficient to complete eliminate aliasing. Why?

Texture Map Aliasing

- Nearest sampling is very bad.
- (Bi)-linear interpolation improves, but still insufficient.
- Fundamental problem: image resolution and texture resolution are not the same.
  - Image res = Texture res: OK
  - Image res > Texture res: bilinear interpolation
  - Image res < Texture res: aliasing! need average/filter.
- Think about perspective projection.

Texture Map Aliasing

- Theory in digital signal sampling to explain aliasing.



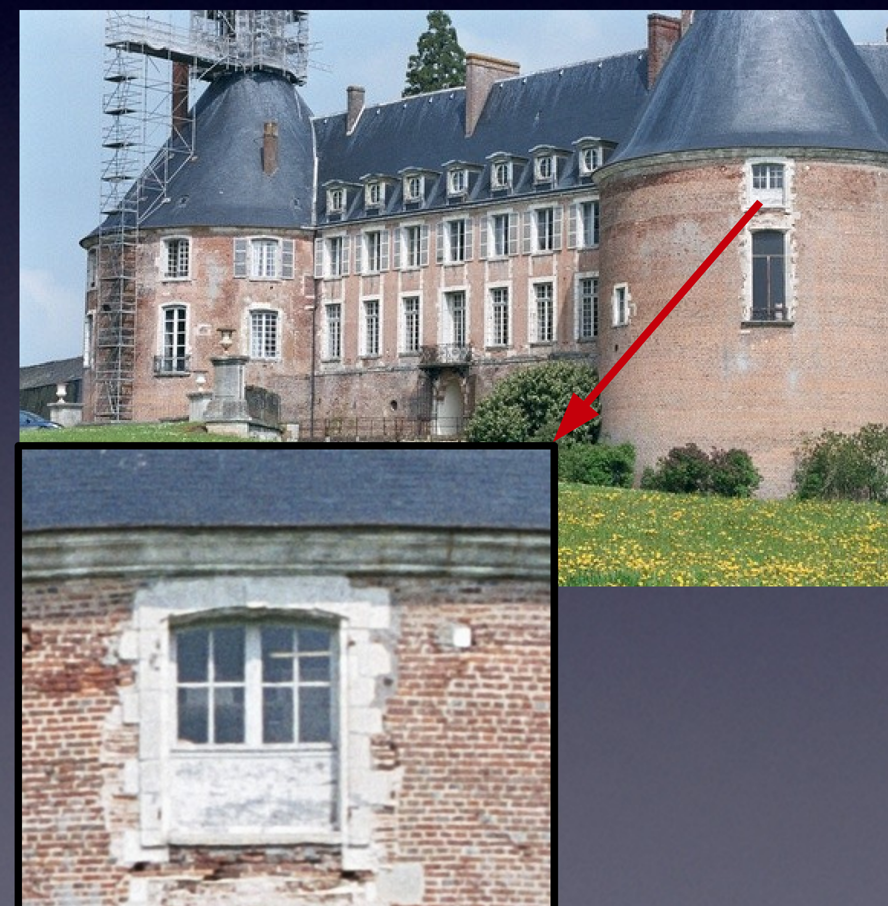
- Sampling rate must be sufficient (twice the bandwidth of the source signal) to reproduce the original signal.

Texture Map Aliasing

- Example of perspective projection:

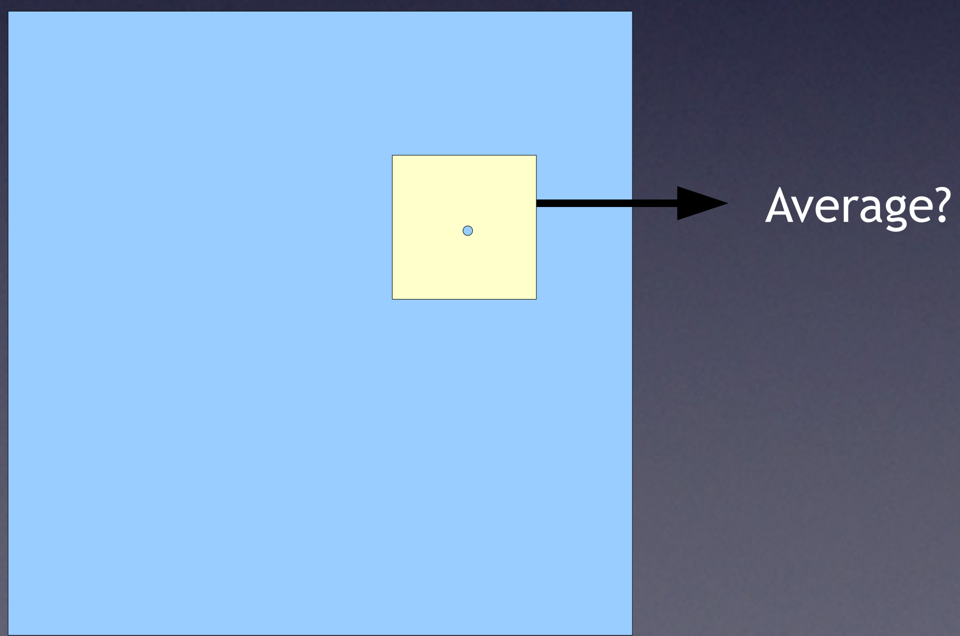
Texture Map Aliasing

- What if sampling rate is limited anyways?
- Solution: filter (smooth) the input signal.



Texture Filtering

- Basically, we want to very efficiently compute the average of a neighborhood of texels.
  - The size of the neighborhood maybe arbitrary and spatially varying.
  - Must be efficient to compute and store.

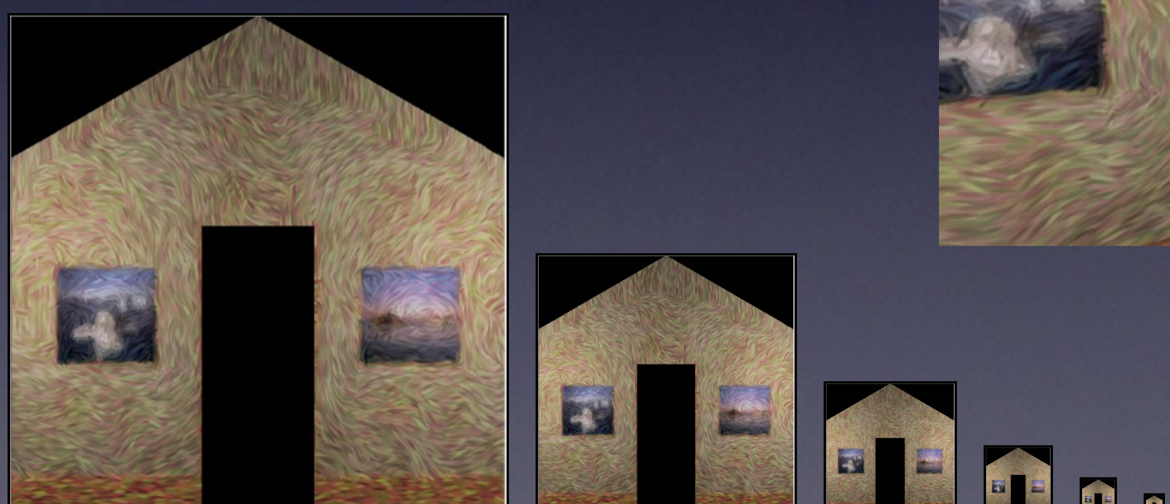


Mip Maps

- Keep textures prefiltered: a hierarchy of textures, each layer is a 2x2 reduction from the previous layer.
- Mip: comes from latin, meaning “much in a small space”

Mip Maps

- Interpolate between two mipmap levels to approximate arbitrary neighborhood size: trilinear interpolation.



Mip Maps

- No filter vs. Mip Map Filtering

AAAAAAAGH
MY EYES ARE BURNING

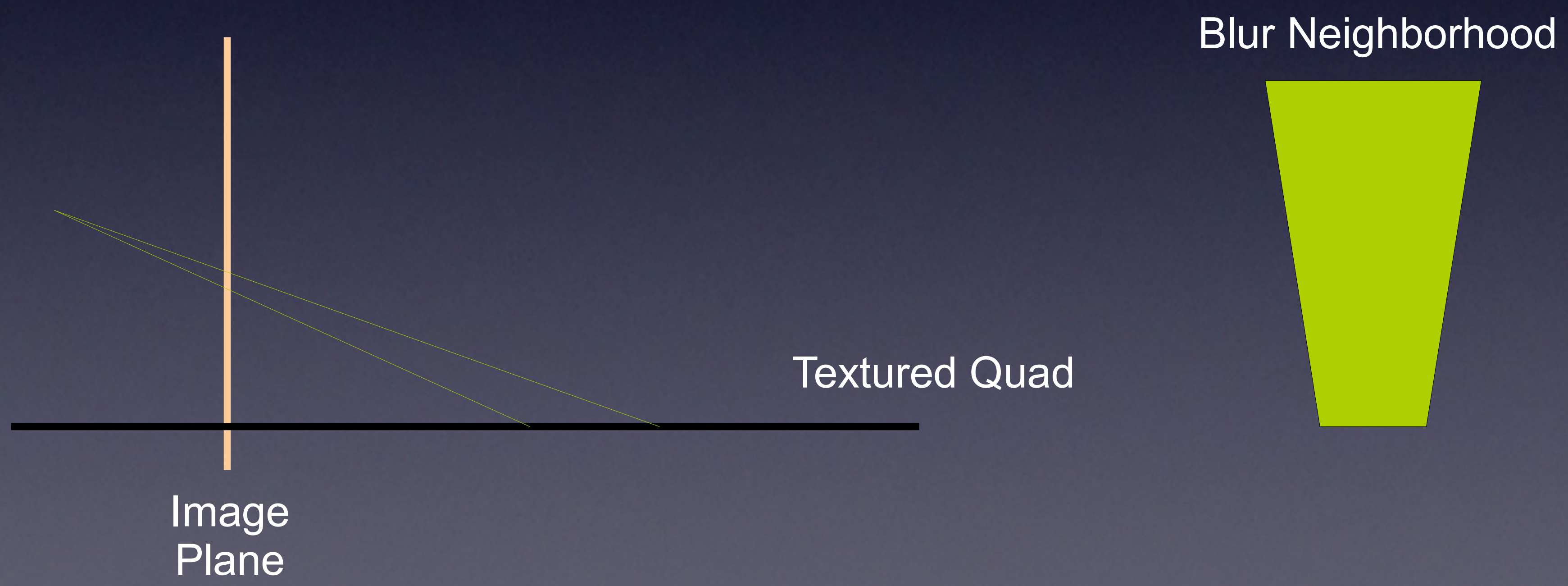
Where are my glasses?

Mip Maps

- How to build Mip Maps?
  - 2D reduction
  - Hardware accelerated in OpenGL
- Advantages
  - Fast, efficient

Mip Maps

- Disadvantages
  - Isotropic filtering, resulting in over-blurring
  - Can be solved by anisotropic mipmapping

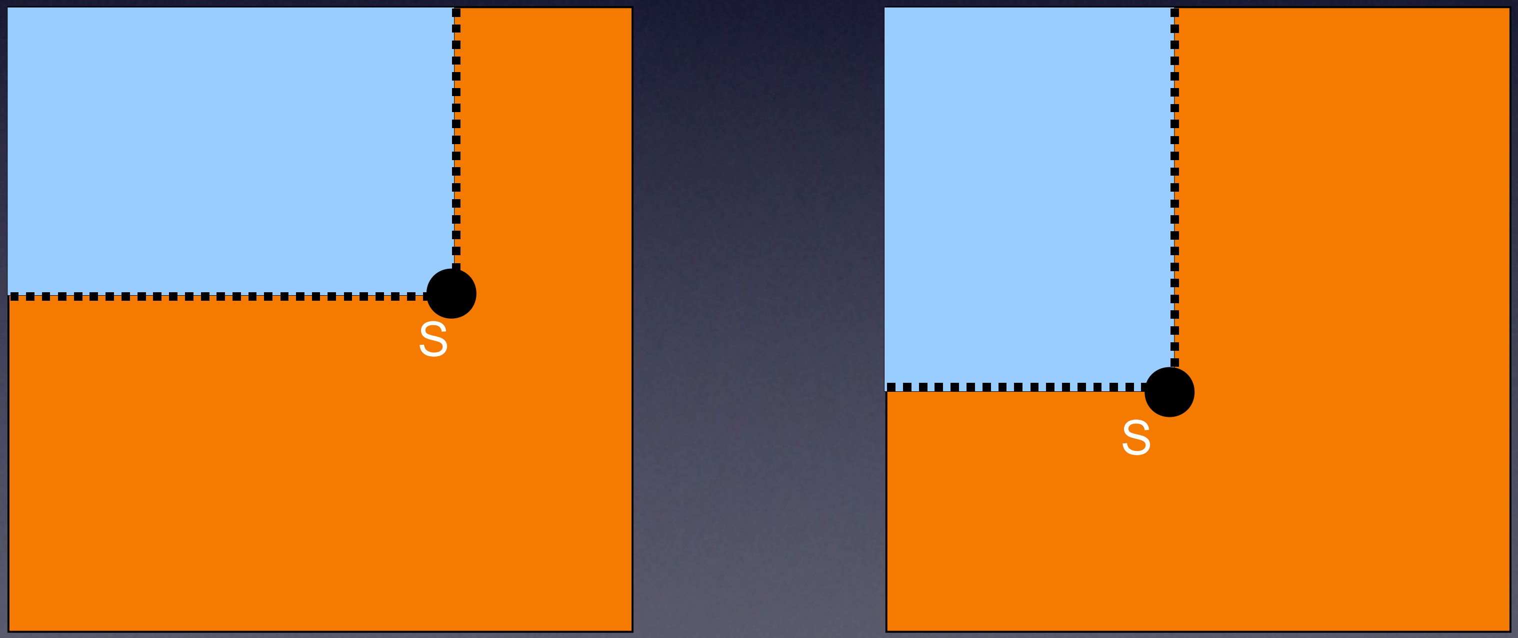


Summed-Area Table (SAT)

- SAT is a way to quickly computing the sum of a rectangular subset of a 2D array.
  - Idea can similarly apply in 1D or 3D. 1D as example.

Summed-Area Table (SAT)

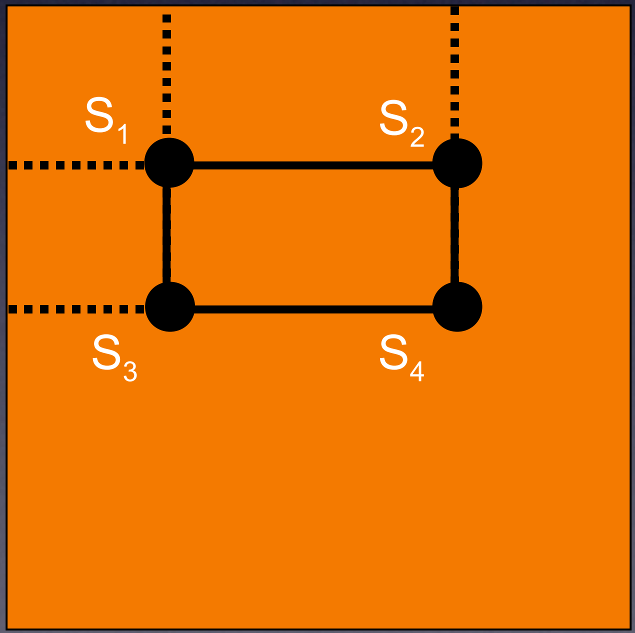
- Pre-compute partial sums
  - Similar to pre-fix sum, but in 2D
- Compute the sum of an arbitrary rectangular subset?



Summed-Area Table (SAT)

- Pre-compute partial sums
  - Similar to pre-fix sum, but in 2D
- Compute the sum of an arbitrary rectangular subset?
  - 2 subtracts, 1 add

$S = S4 - S2 - S3 + S1$



Summed-Area Table (SAT)

- SAT can quickly integrate texels covered by a rectangular sub-area: more anisotropic capability.

- Mip Map Filtering vs. SAT Filtering

Summed-Area Table (SAT)

- How to build an SAT?
- Advantages and Disadvantages.

Summed-Area Table (SAT)

- How to build an SAT?
  - Can apply +Scan on rows and subsequently on columns
  - If the texture is big enough, can process entire rows (then columns) in parallel using sequential scan.
- Advantages
  - Fast, more anisotropic
- Disadvantages
  - High precision (more bits) needed
  - What about non-axis-aligned rectangle, or neighborhood with more weird shapes?

OpenGL Interoperability

- OpenGL Buffer Objects can be mapped to CUDA address space.

1. In the very beginning, need to initialize:

```
cudaGLSetGLDevice();
```

2. Register a buffer object with CUDA

```
cudaGLRegisterBufferObject(GLuint buffer);
```

3. Map the buffer object to CUDA memory space

```
cudaGLMapBufferObject(void **devPtr, Gluint
buffer);
```

4. Launch CUDA kernel to process the buffer
5. Unmap and unregister

OpenGL Interoperability

- To handle textures, currently need to use glTexImageSub2D and glReadPixels to transfer data between pixel buffer objects (PBOs) and textures.
    - PBOs can be mapped to CUDA space as before.