Computer Graphics

MTAT.03.015

Raimond Tunnel

UNIVERSITY of TARTU
Institute of Computer Science
Last week
Last week

This week

Construct geometry
Define transformations

Assign material properties...

Vertex Transformations

Vertex Shader
Object's local space → viewport space

Determine front-facing triangles
Determine which vertices are visible

Culling & Clipping

Fill the triangle with fragments

Rasterization

Fragment Shader
Calculate correct color values

Visibility Tests
Blending

Is the fragment visible?
Blend together multiple fragments
More Granular Surface Color

Chopper by Annika Hansalu
More Granular Surface Color

- Blades – 4 different meshes

Chopper by Annika Hansalu
More Granular Surface Color

- Blades – 4 different meshes:
  - 2 blades
More Granular Surface Color

- Blades – 4 different meshes:
  - 2 blades, each blade consists of 2 parts
More Granular Surface Color

- Extra vertices and faces that all need parsing
More Granular Surface Color

- Extra vertices and faces that all need parsing

- Could we get the same result with 4 vertices?
More Granular Surface Color

- We need to specify colors per fragment.
More Granular Surface Color

- We need an image to specify colors per fragment.
More Granular Surface Color

- We need an image to specify colors per fragment.
- We no longer just interpolate the color, but make and use a mapping.
More Granular Surface Color

- We need an image to specify colors per fragment.
- We no longer just interpolate the color, but make and use a mapping.
More Granular Surface Color

- We need an image to specify colors per fragment.
- We no longer just interpolate the color, but make and use a UV mapping.
(Raster) Image

- Image is a matrix (2D array) of point values.
(Raster) Image

• Image is a matrix (2D array) of point values.

• Our 3D surface is **continuous**, we may rasterize a **varying amount** of points for a face.
Upscale

- Sometimes we want to see the surface in more detail than there are point values in the image.
Upscale

- Sometimes we want to see the surface in more detail than there are point values in the image.
Upscale

- Sometimes we want to see the surface in **more detail** than there are point values in the image.
Upscale

- Sometimes we want to see the surface in more detail than there are point values in the image.
Upscale

- Sometimes we want to see the surface in more detail than there are point values in the image.
Upscale

- For a single point in the larger surface, we usually have 4 neighbours in the texture.
Upscale

• For a single point in the larger surface, we usually have 4 neighbours in the texture.

• What are the exceptions?
Upscale

- For a single point in the larger surface, we usually have 4 neighbours in the texture.
- What are the exceptions?
- How could we find the sample value?
Upscale

- For a single point in the larger surface, we usually have 4 neighbours in the texture.
- What are the exceptions?
- How could we find the sample value?

```
GL_NEAREST
```

```
GL_LINEAR
```
What do these do?

- `GL_CLAMP_TO_EDGE`
- `GL_CLAMP_TO_BORDER`
- `GL_MIRRORED_REPEAT`
- `GL_REPEAT`

Upscale

• With that in mind, what would be the smallest texture we need for the chopper blade here?
Upscale

• Given a texture with some \( width \times height \), how to find the nearest texels for a sample with UV coordinates?
Downscale

- We can try to do the same interpolation for the downscale.
Downscale

- What can go wrong?
Nyquist–Shannon Sampling Theorem

- In order to reconstruct a band-limited signal, one has to sample with sampling rate more than twice the highest frequency.
Nyquist–Shannon Sampling Theorem

- In order to reconstruct a band-limited signal, one has to sample with sampling rate more than twice the highest frequency.

This means more than 2 samples per period, every period.
Nyquist–Shannon Sampling Theorem

• Band-limited signal – there is a fixed highest frequency in the signal.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
- Sampling less frequently, we produce an alias.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
- Sampling less frequently, we produce an alias.

Alias – Something unwanted, emerging because of our algorithm, a signal non-existent in the original data, masquerading as the original signal.
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
- Sampling less frequently, we produce an alias.
- Usually assumes samples are taken over time.

This is how radio works...
Nyquist–Shannon Sampling Theorem

- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the shape of the signal.
- Sampling less frequently, we produce an alias.
- Usually assumes samples are taken over time.

More info:

- http://www.skillbank.co.uk/SignalConversion/rate.htm
Downscale

- So, what is happening in our example?
Downscale

• So, what is happening in our example?

\[ \text{period} = 4 \Rightarrow \text{frequency} = \frac{1}{4} \]
Downscale

- So, what is happening in our example?

\[ \text{period} = 4 \Rightarrow \text{frequency} = \frac{1}{4} \]

\[ \text{frequency}_{\text{Nyquist}} = \frac{2}{4} = \frac{1}{2} \]

We need more than 1 sample for every 2 units!
Downscale

- So, what is happening in our example?

\[ \text{frequency} = \frac{1}{4} \]

\[ \text{frequency}_{\text{Nyquist}} = \frac{2}{4} = \frac{1}{2} \]

\[ \text{frequency}_{Us} = \frac{1}{4} < \frac{1}{2} \]
Downscale

- So, what is happening in our example?

\[
\text{period} = 4 \Rightarrow \text{frequency} = \frac{1}{4}
\]

\[
f_{Nyquist} = \frac{2}{4} = \frac{1}{2}
\]

\[
f_{Us} = \frac{1}{4} < \frac{1}{2}
\]
Downscale

- We need more than 1 sample per 2 units.
Downscale

- We need more than 1 sample per 2 units.
- Is this even possible, if we want to downscale our pattern from $8 \times 8$ to $2 \times 2$?
Downscale

- We do not want to create *Moire aliasing*. 
Downscale

- We do not want to create *Moire aliasing*.
- The $2 \times 2$ downscale should not be white!
Downscale

- We do not want to create Moire aliasing.
- The 2×2 downscale should not be white!
- One unit in the result covers 16 units in the texture. How to represent all those 16 values?
Mipmapping

• In order not to take that many samples each time for downscaling, we take them beforehand.
Mipmapping

- What if we have a texture that is $10 \times 10$. 
Mipmapping

- What if we have a texture that is $10 \times 10$.
  - The first mipmap is the image itself – $10 \times 10$. 
Mipmapping

- What if we have a texture that is 10×10.
  - The first mipmap is the image itself – 10×10.
  - Then we take half the size – 5×5.
Mipmapping

- What if we have a texture that is $10 \times 10$.
  - The first mipmap is the image itself – $10 \times 10$.
  - Then we take half the size – $5 \times 5$.
  - Next we take half the size – $2.5...$ Uh-oh.
Mipmapping

• What if we have a texture that is $10 \times 10$.
  • The first mipmap is the image itself – $10 \times 10$.
  • Then we take half the size – $5 \times 5$.
  • Next we take half the size – $2.5 \ldots$ Uh-oh.
• The last mipmap we could create is $5 \times 5$. 
Mipmapping

• What if we have a texture that is 10×10.
  • The first mipmap is the image itself – 10×10.
  • Then we take half the size – 5×5.
  • Next we take half the size – 2.5... Uh-oh.

• The last mipmap we could create is 5×5.

• For a smaller downscale (eg 2×2, 1×1) we still need to sample more than the 4 neighbouring pixels.
Mipmapping

- What if we have a texture that is 10×10.
  - The first mipmap is the image itself – 10×10.
  - Then we take half the size – 5×5.
  - Next we take half the size – 2.5... Uh-oh.
- The last mipmap we could create is 5×5.
- For a smaller downscale (e.g., 2×2, 1×1) we still need to sample more than the 4 neighbouring pixels.
- How not to have that problem?
Mipmapping

• Assume we have mipmaps 8×8, 4×4, 2×2, 1×1.
Mipmapping

- Assume we have mipmaps 8×8, 4×4, 2×2, 1×1.
- We want to show our texture on a 6×6 area.
Mipmapping

- Assume we have mipmaps 8×8, 4×4, 2×2, 1×1.
- We want to show our texture on a 6×6 area.
- Which mipmap should we sample?
Filtering

- We have seen ways to sample the texture.
Filtering

- We have seen ways to sample the texture.
- Upscale (magnification filtering)
Filtering

● We have seen ways to sample the texture.

● Upscale (magnification filtering):
  ● Nearest neighbour
Filtering

- We have seen ways to sample the texture.

- Upscale (magnification filtering):
  - Nearest neighbour
  - Bilinear
Filtering

• We have seen ways to sample the texture.

• Upscale (magnification filtering):
  • Nearest neighbour
  • Bilinear

• Downscale (minification filtering)
Filtering

• We have seen ways to sample the texture.

• Upscale (magnification filtering):
  • Nearest neighbour
  • Bilinear

• Downscale (minification filtering):
  • Nearest neighbour (mipmap: no, NN, linear)
Filtering

• We have seen ways to sample the texture.

• Upscale (magnification filtering):
  • Nearest neighbour
  • Bilinear

• Downscale (minification filtering):
  • Nearest neighbour (mipmap: no, NN, linear)
  • Bilinear (mipmap: no, NN, linear)
Filtering

- We have seen ways to sample the texture.
- Upscale (magnification filtering):
  - Nearest neighbour
  - Bilinear
- Downsacle (minification filtering):
  - Nearest neighbour (mipmap: no, NN, linear)
  - Bilinear (mipmap: no, NN, linear)

Also called trilinear
Filtering

- We have seen ways to sample the texture.

- **Upscale (magnification filtering):**
  - Nearest neighbour
  - Bilinear

- **Downscale (minification filtering):**
  - Nearest neighbour (mipmap: no, NN, linear)
  - Bilinear (mipmap: no, NN, linear)

Also called trilinear
Anisotropic Filtering

- We assumed that the result we are showing our texture on is shown as a square. This is usually not the case.
Anisotropic Filtering

- We assumed that the result we are showing our texture on is shown as a square. This is usually not the case.

- If we rotate our quad around the x-axis for example, then we might get that the texture needs to be shown on a $10 \times 5$ area instead of $10 \times 10$. 

Anisotropic Filtering

• We have more resolution in width than in height. It is unfair to average both dimensions equally.
Anisotropic Filtering

- We have more resolution in width than in height. It is unfair to average both dimensions equally.

- Anisotropic filtering will use the higher mipmap and take more samples along the denser direction.
Anisotropic Filtering

- Actual implementations are vendor dependant.
- One way would be to just create anisotropic mipmaps.
Textures
Textures

• There are more uses than just storing granular color.
Textures

• There are more uses than just storing granular color.
  
  • **Data textures** – we can hold other data like normals or other values with 3 (or 4) coordinates.
Textures

- There are more uses than just storing granular color.
  - **Data textures** – we can hold other data like normals or other values with 3 (or 4) coordinates.
  - **Noise texture** – we can store samples of a random function or noise (eg Perlin noise) function.
Textures

- There are more uses than just storing granular color.
  - **Data textures** – we can hold other data like normals or other values with 3 (or 4) coordinates.
  - **Noise texture** – we can store samples of a random function or noise (eg Perlin noise) function.
  - **Render target** – we could also render the current framebuffer to a texture.
Textures

- There are more uses than just storing granular color.
  - **Data textures** – we can hold other data like normals or other values with 3 (or 4) coordinates.
  - **Noise texture** – we can store samples of a random function or noise (e.g., Perlin noise) function.
  - **Render target** – we could also render the current framebuffer to a texture.
What seemed useful today?

What more would you like to know?

Next time
Blending – Jaanus Jaggo