

Computer Graphics

MTAT.03.015

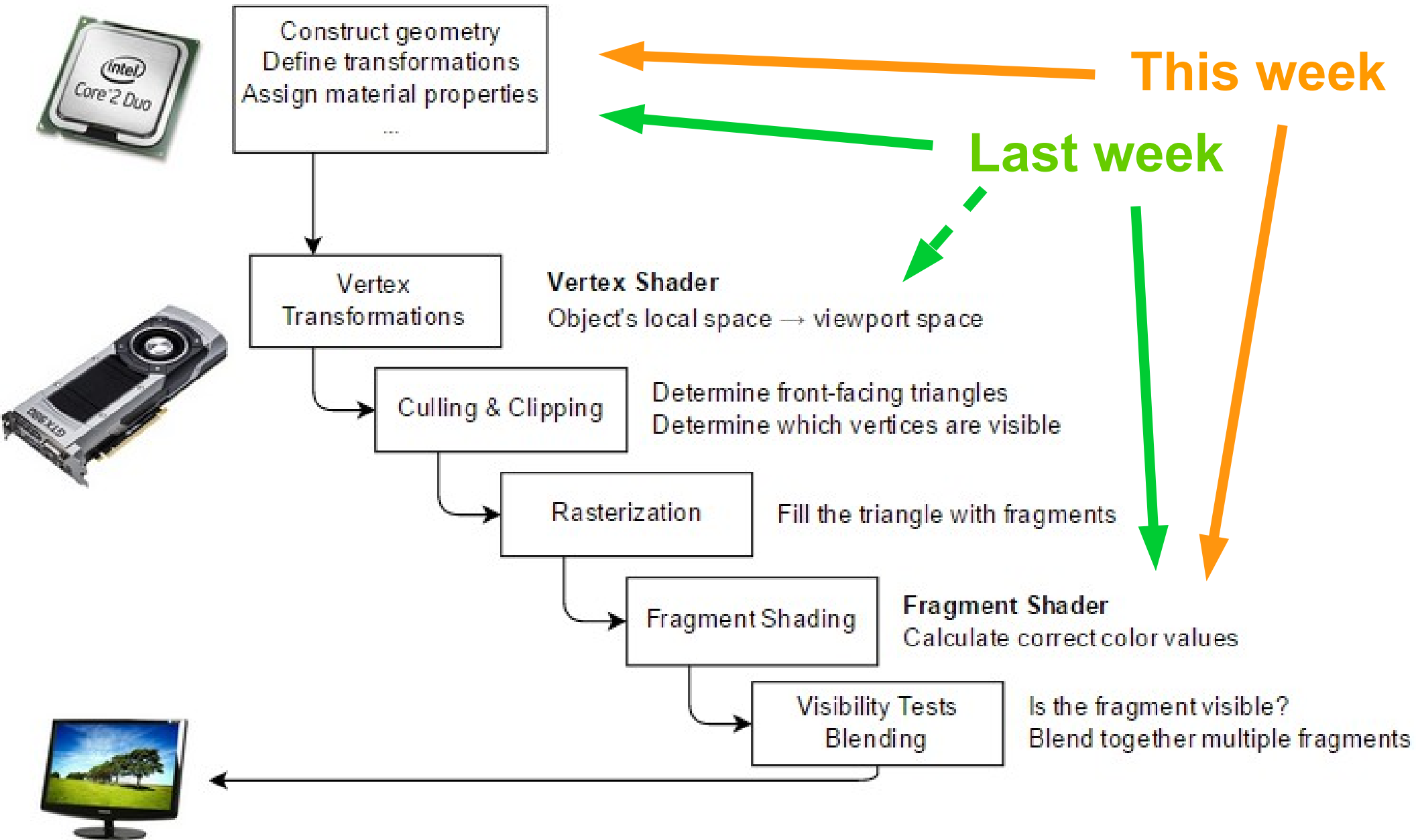
Raimond Tunnel



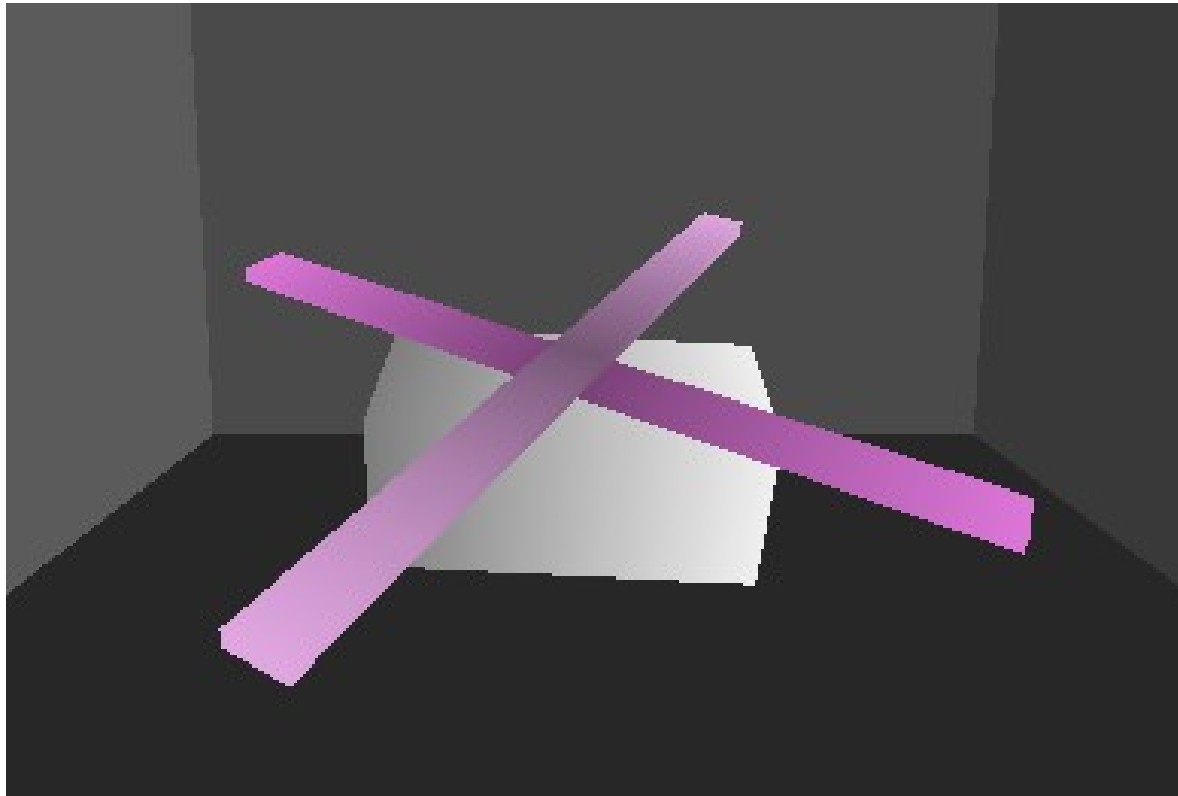
Study IT in .ee



The Road So Far...



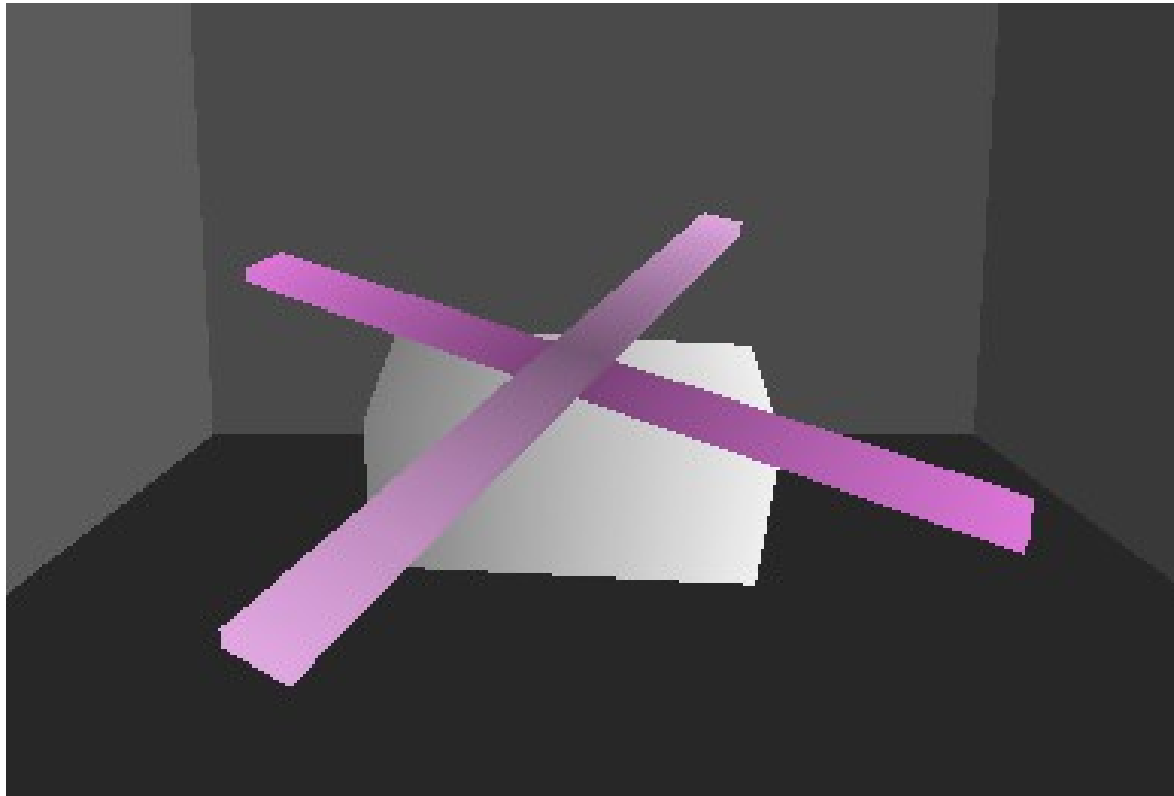
More Granular Surface Color



Chopper by Annika Hansalu

More Granular Surface Color

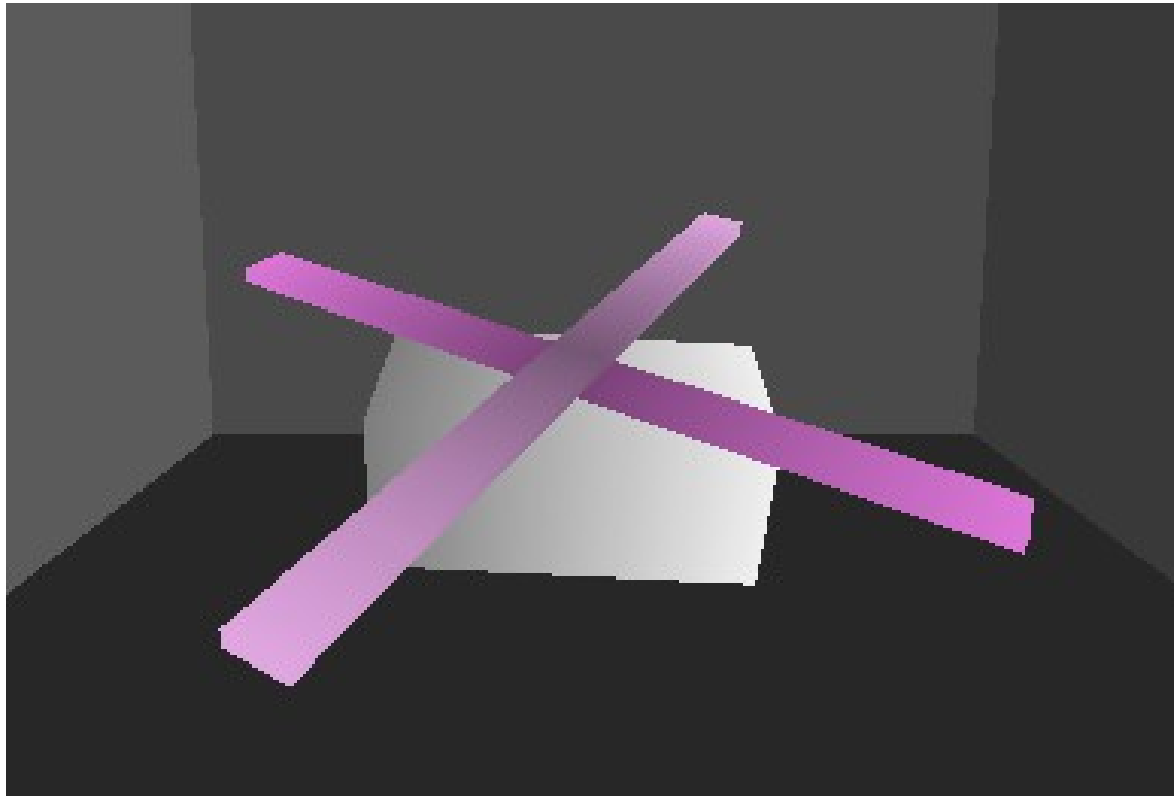
- Blades – 4 different meshes



Chopper by Annika Hansalu

More Granular Surface Color

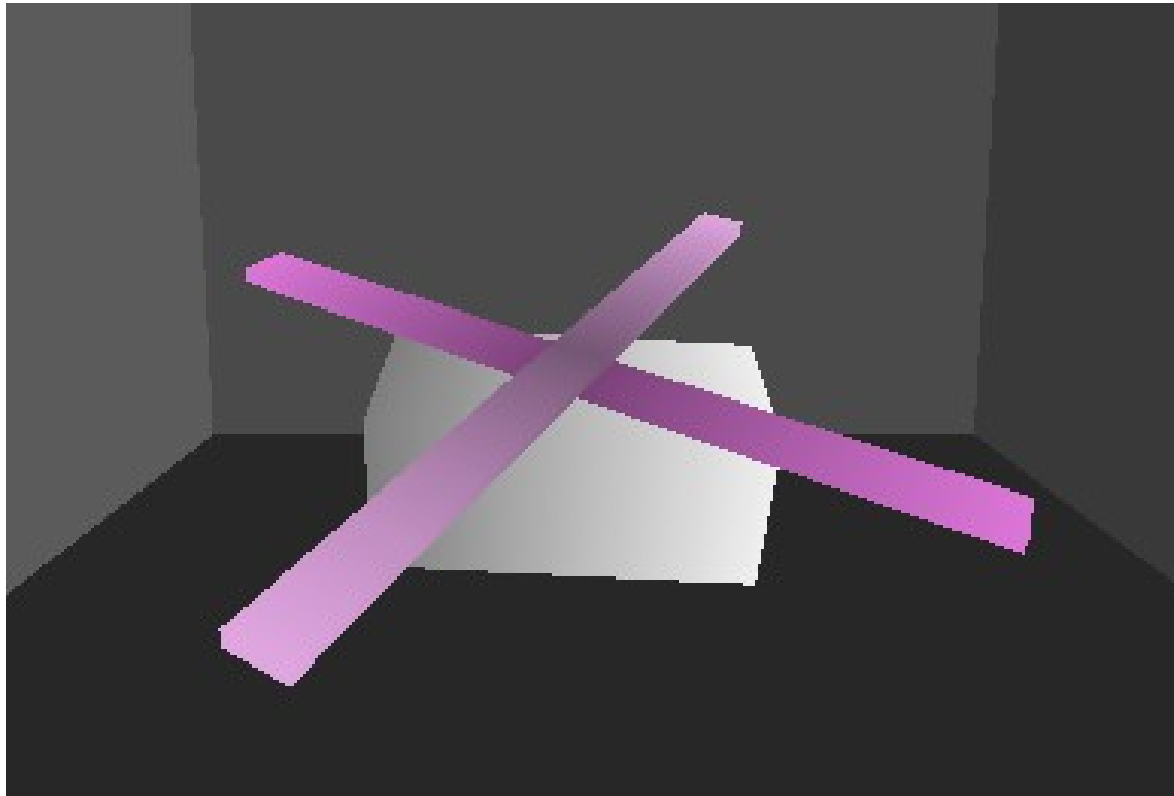
- Blades – 4 different meshes:
 - 2 blades



Chopper by Annika Hansalu

More Granular Surface Color

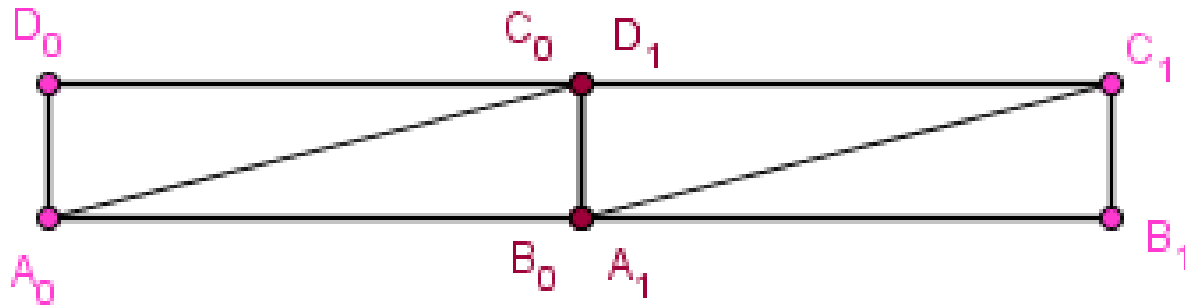
- Blades – 4 different meshes:
 - 2 blades
 - Each blade consists of 2 parts



Chopper by Annika Hansalu

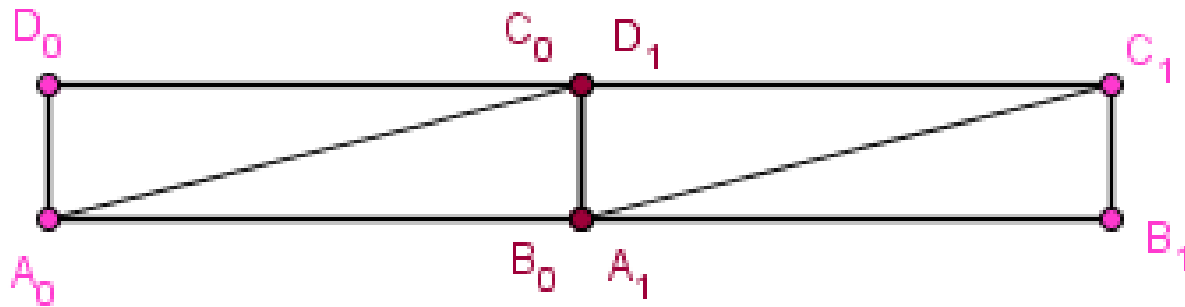
More Granular Surface Color

- Extra vertices and faces that all need parsing

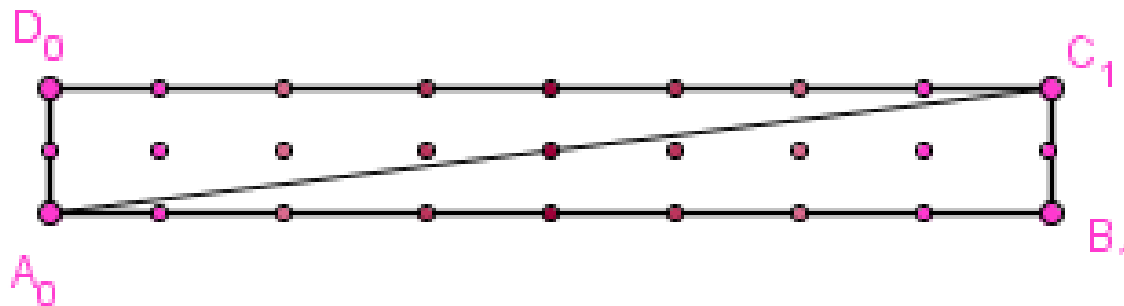


More Granular Surface Color

- Extra vertices and faces that all need parsing

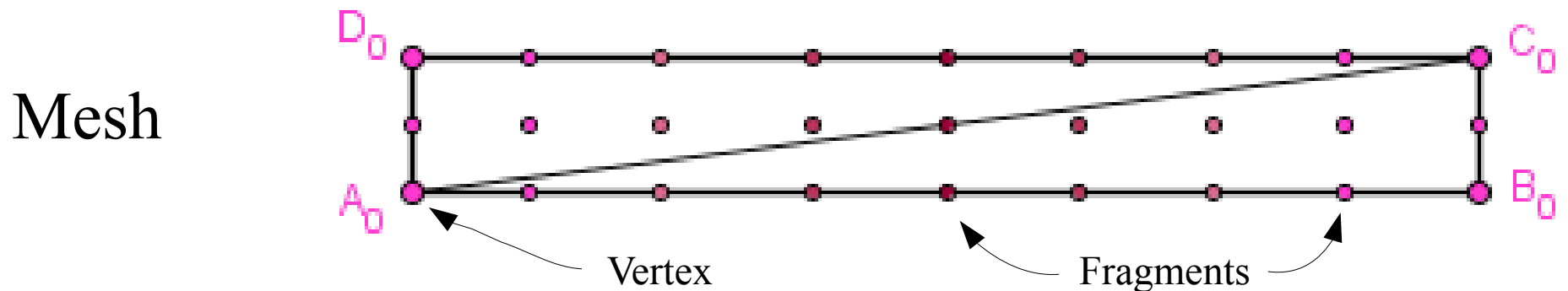


- Could we get the same result with only 4 vertices?



More Granular Surface Color

- We would need to specify at which fragment we take which color.

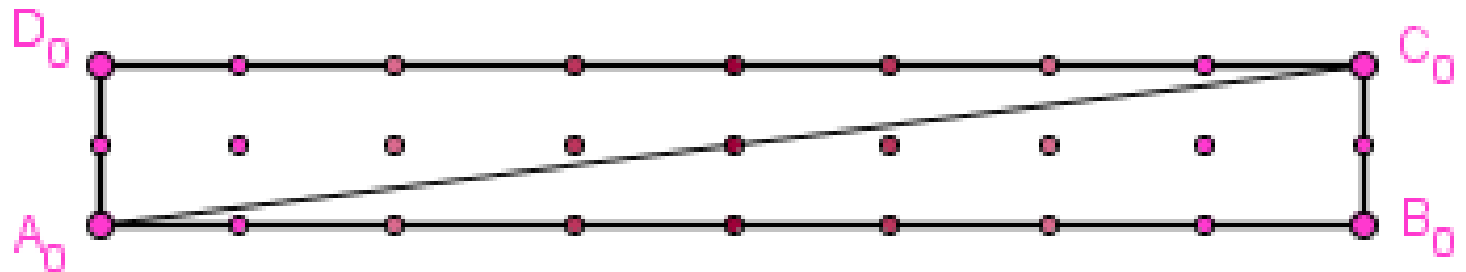


More Granular Surface Color

- We would need to specify at which fragment we take which color.

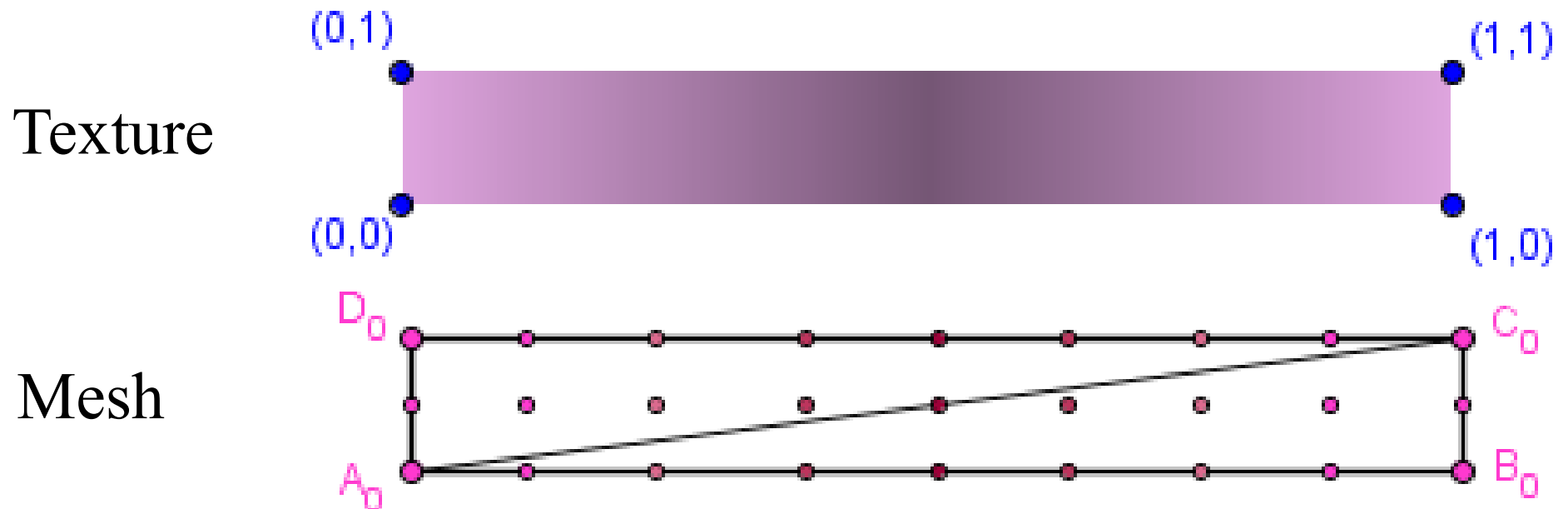


Mesh



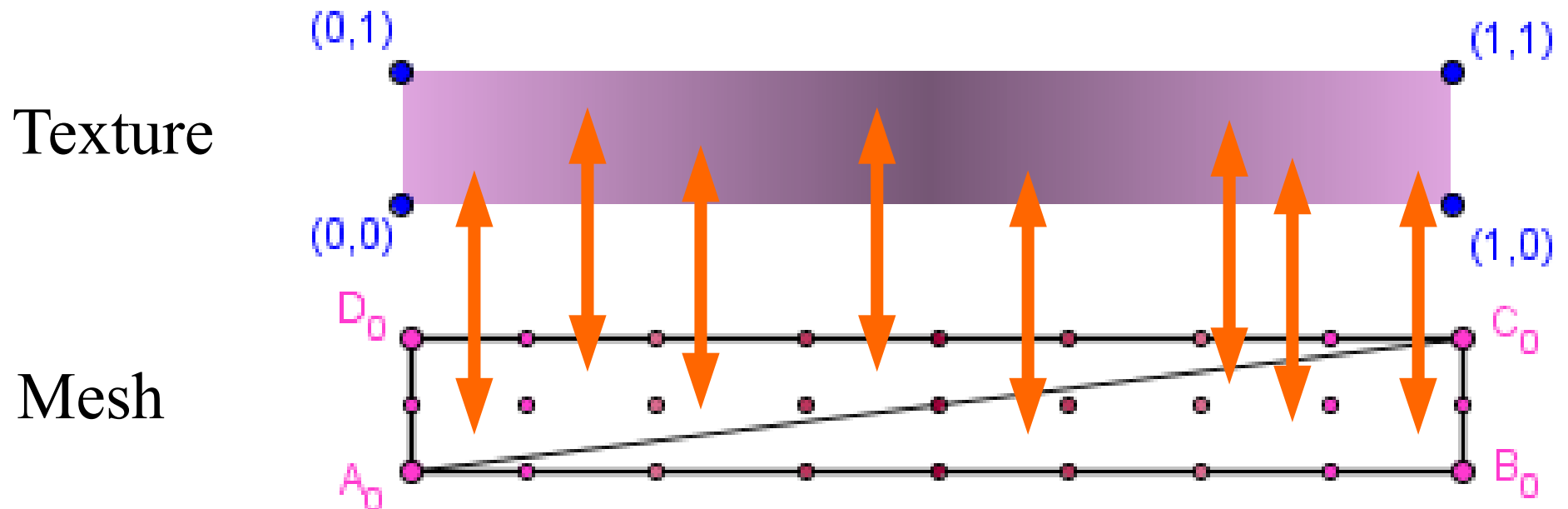
More Granular Surface Color

- We would need to specify at which fragment we take which color.
- We can no longer just interpolate the color, but should somehow specify **a mapping**.



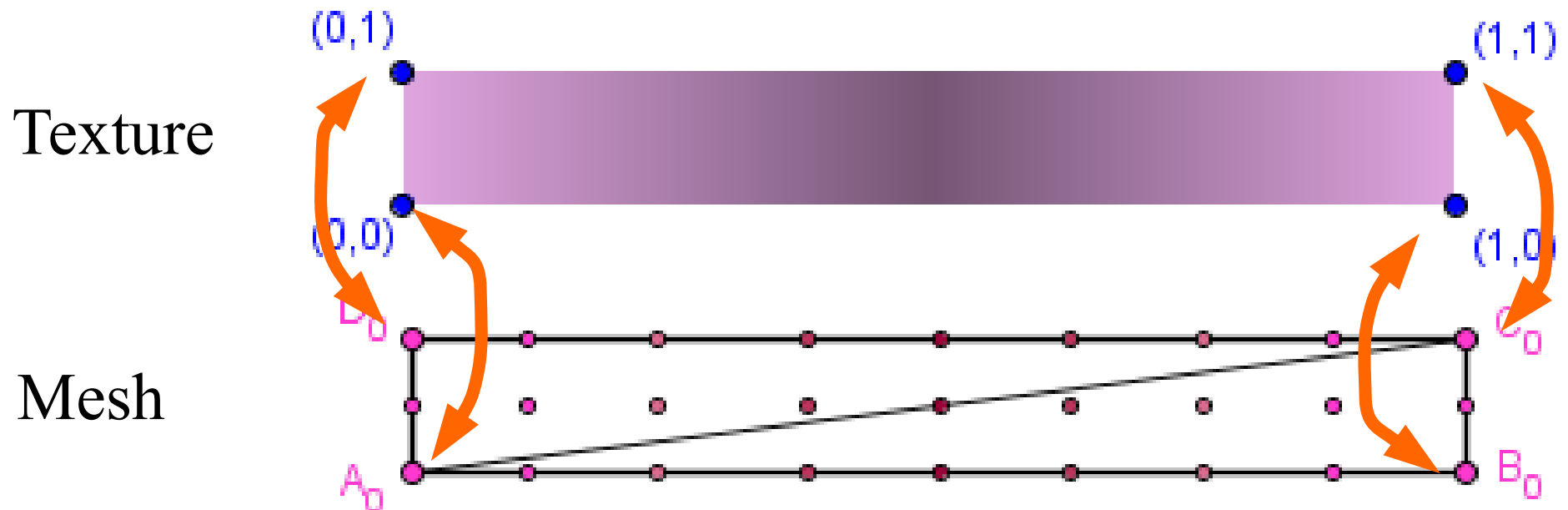
More Granular Surface Color

- We would need to specify at which fragment we take which color.
- We can no longer just interpolate the color, but should somehow specify **a mapping**.



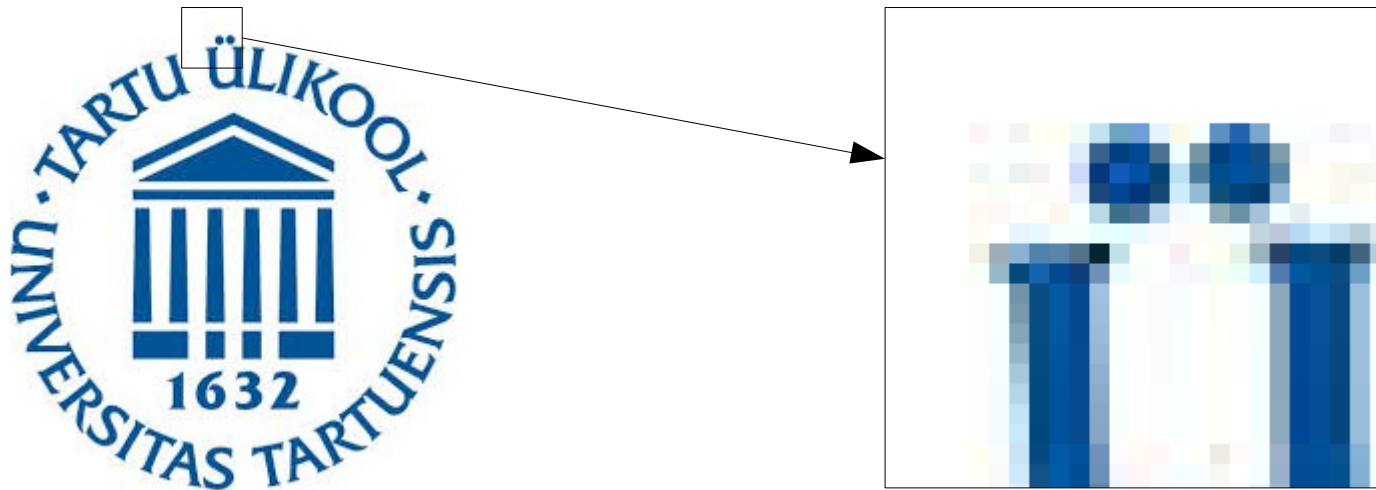
More Granular Surface Color

- We would need to specify at which fragment we take which color.
- We can no longer just interpolate the color, but should somehow specify **a mapping**.



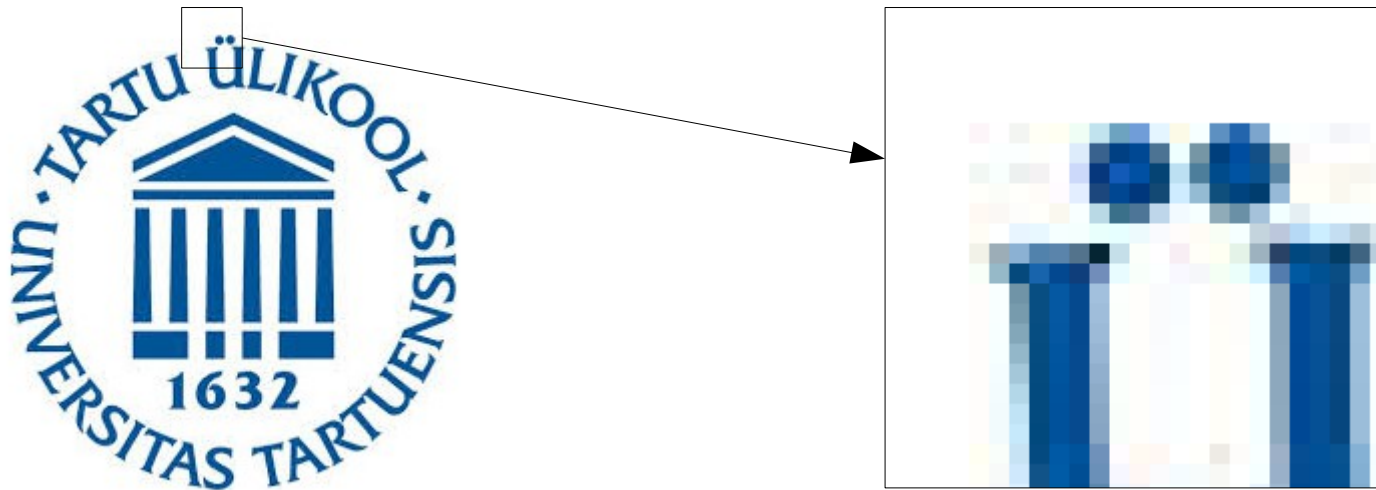
(Raster) Image

- Image is a matrix of point values.



(Raster) Image

- Image is a matrix of point values.



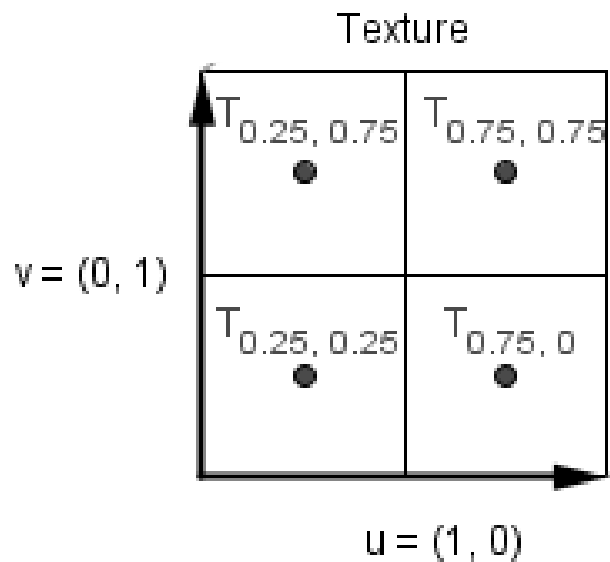
- Our 3D surface is **continuous**, we may rasterize a **varying amount** of points for a face.

Upscale

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

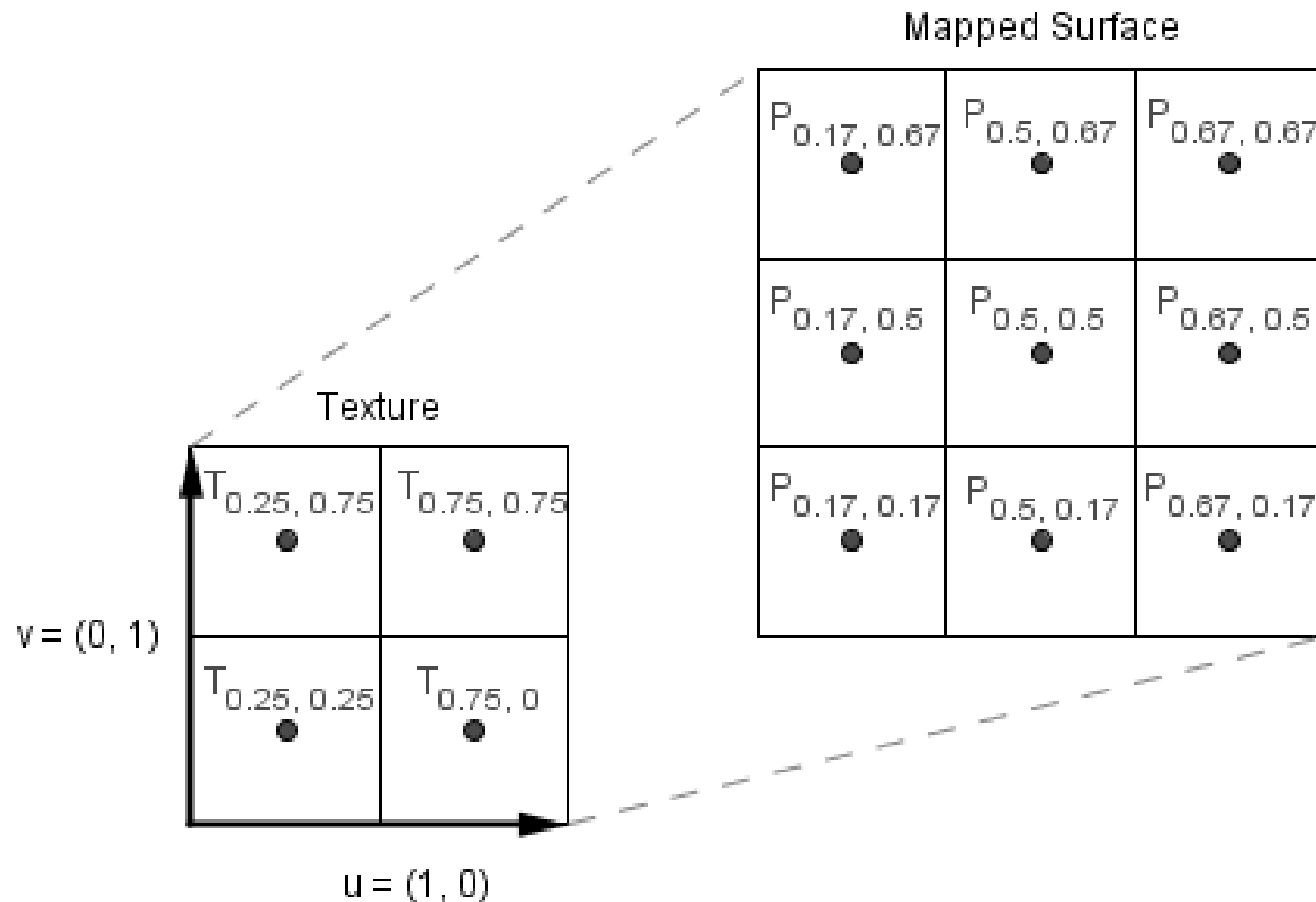
Upscale

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



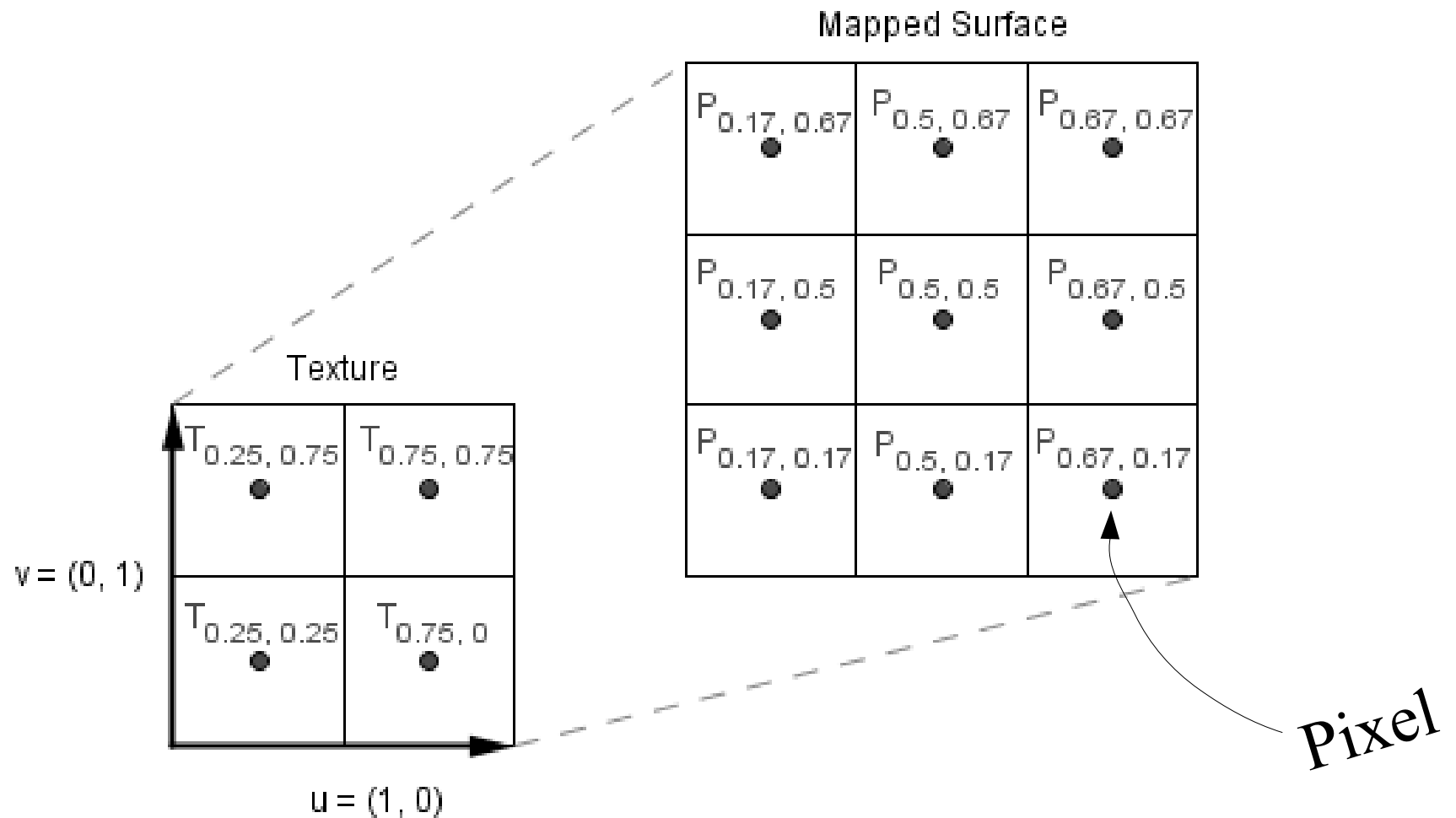
Upscale

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



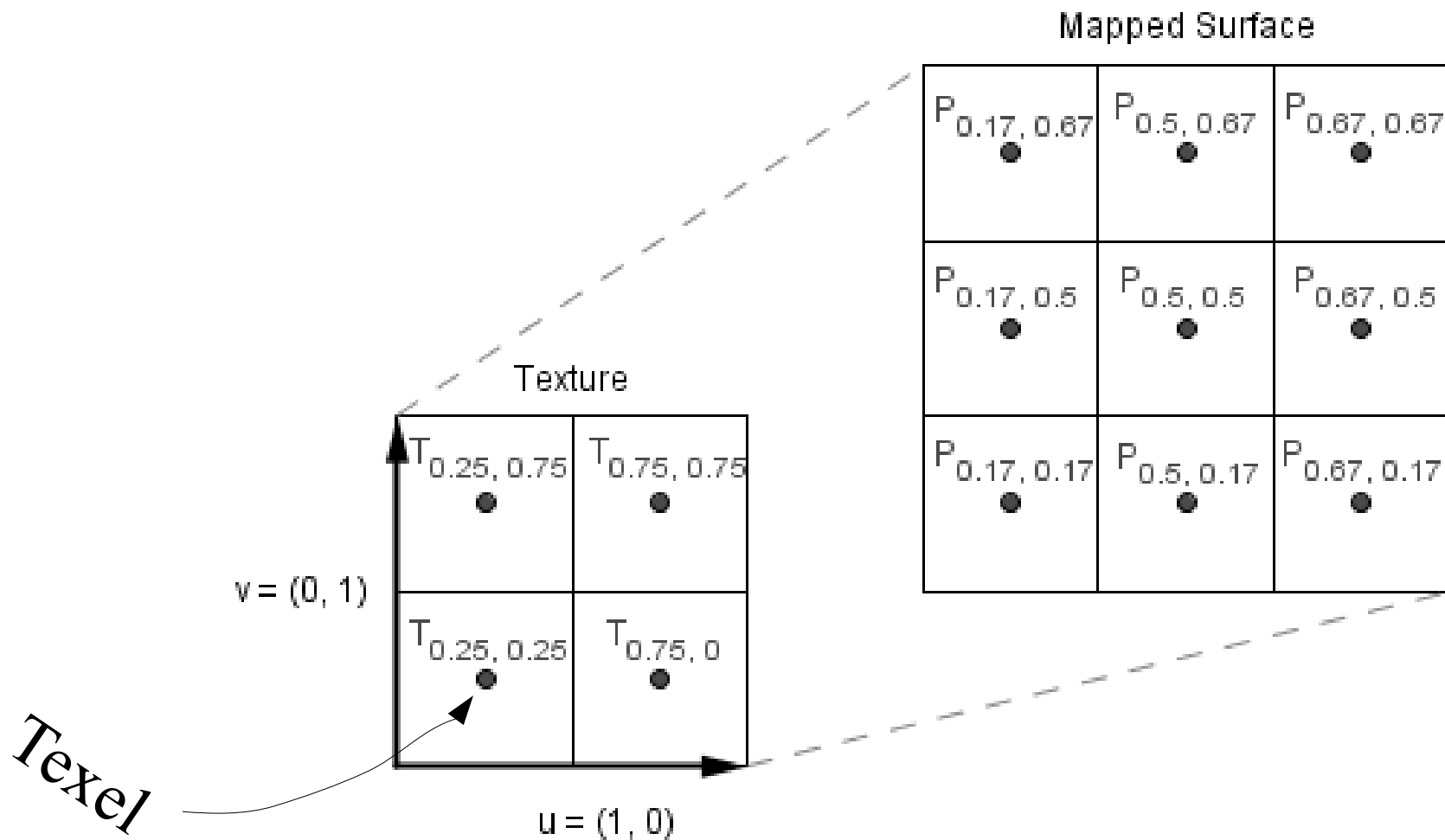
Upscale

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



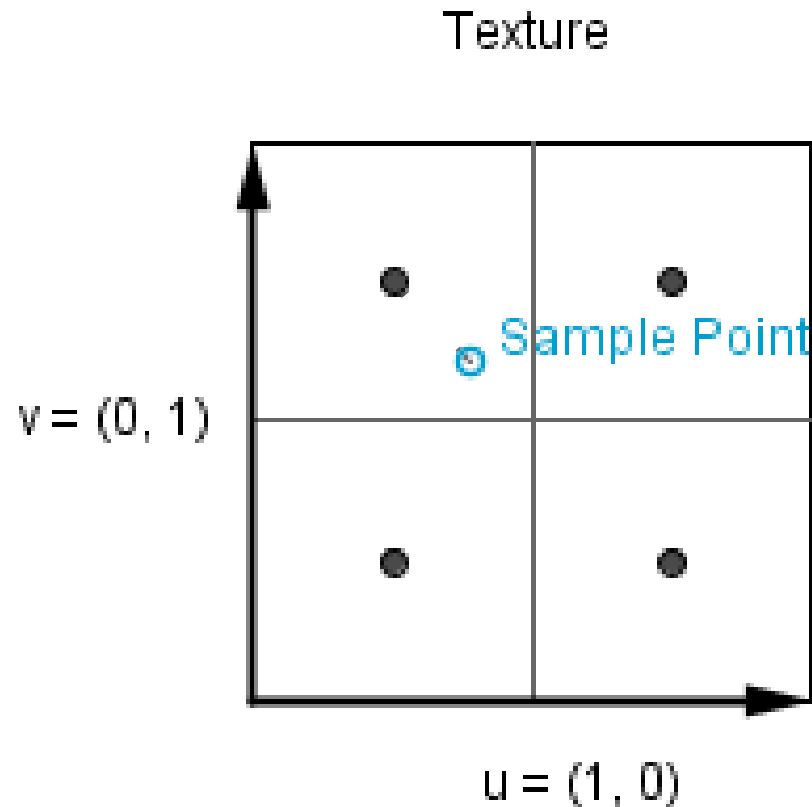
Upscale

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



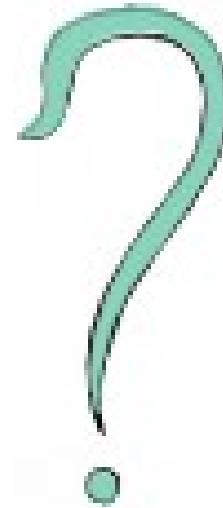
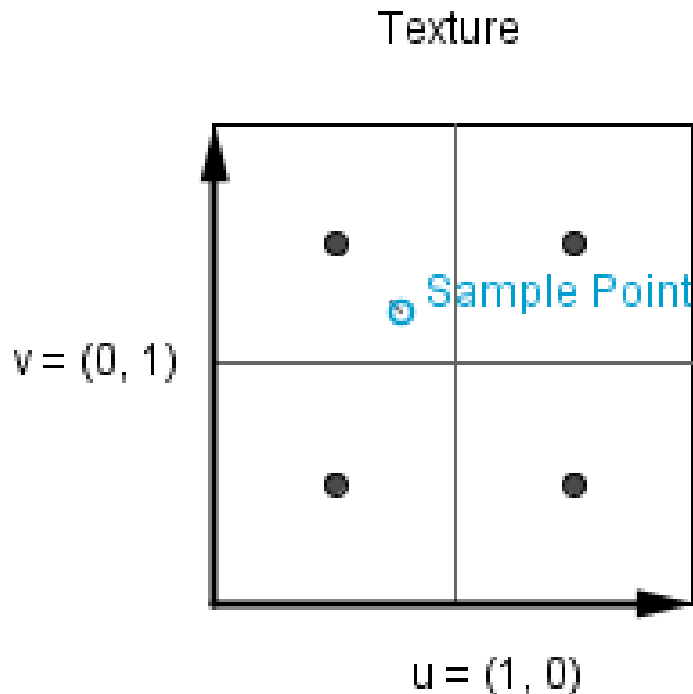
Upscale

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



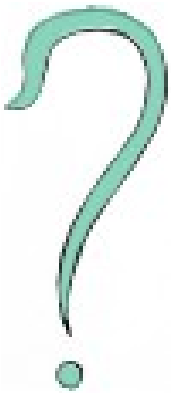
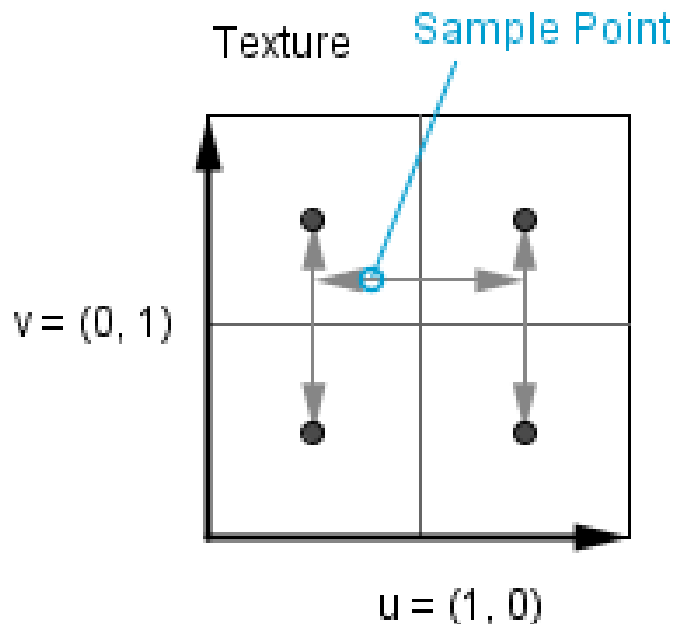
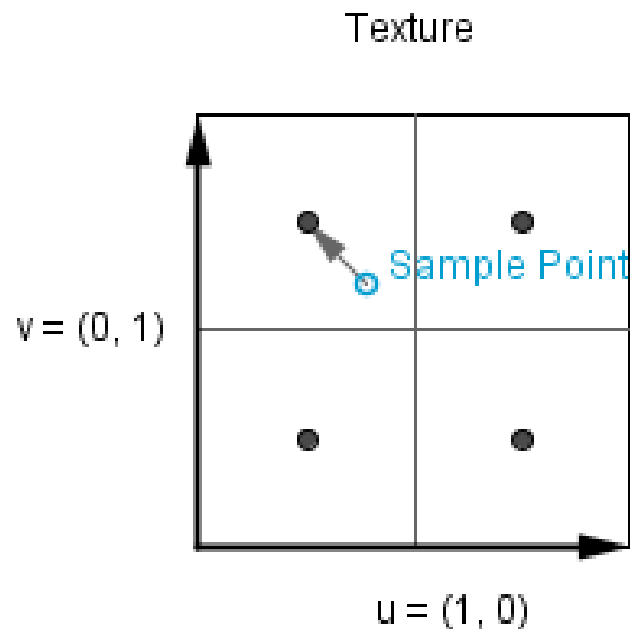
Upscale

- For a single point in the larger surface, we usually have **4 neighbours in the texture**.
- What are the exceptions?



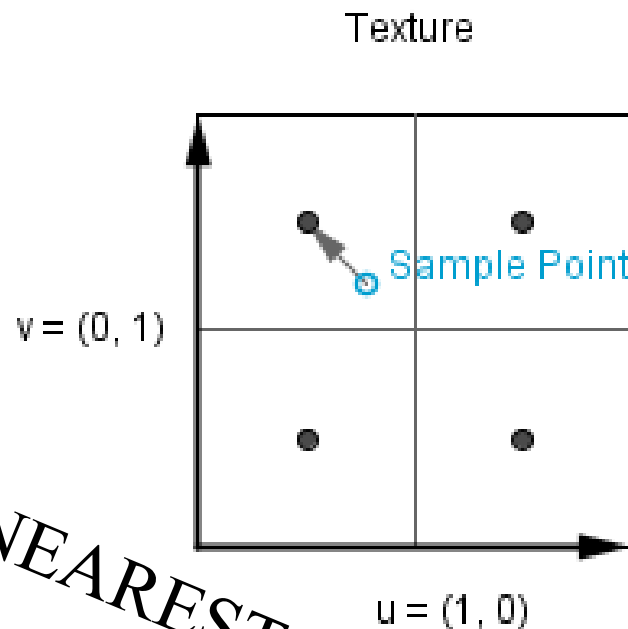
Upscale

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

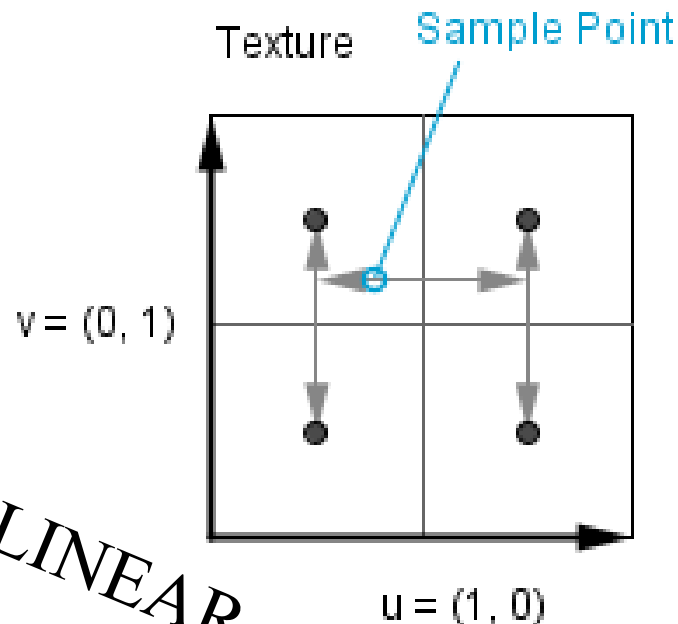


Upscale

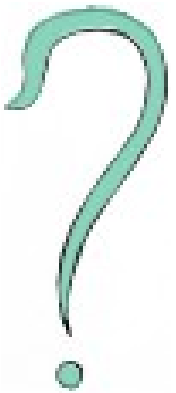
- For a single point in the larger surface, we usually have 4 neighbours in the texture.
- What are the exceptions?
- What possibilities we have to find a value?



GL_NEAREST

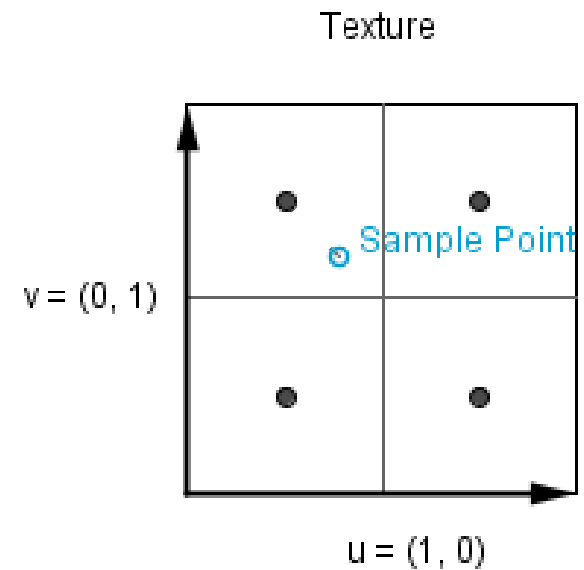


GL_LINEAR



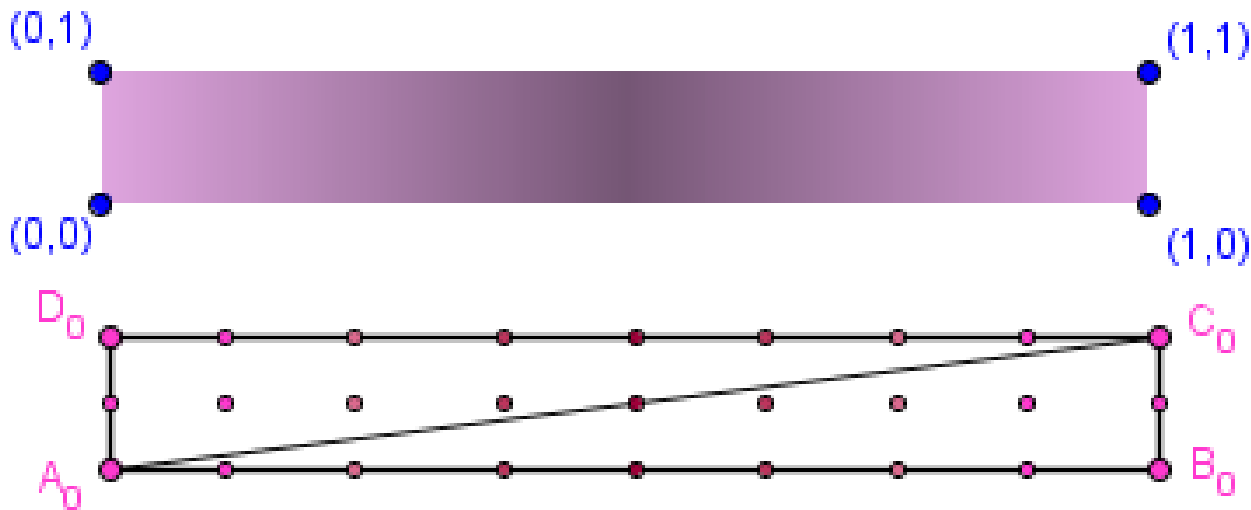
What do these do?

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



Upscale

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



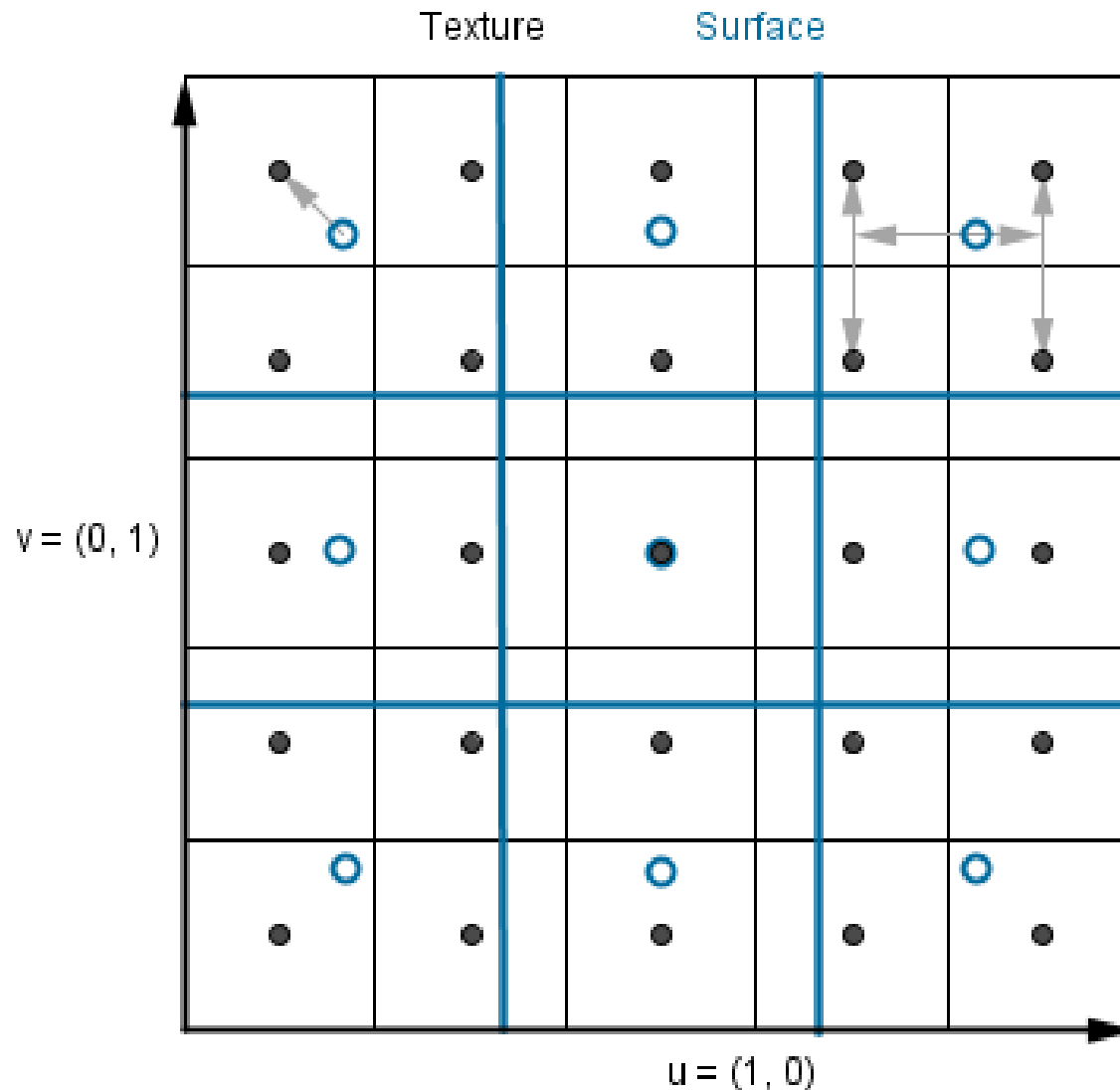
Upscale

- Given a texture with some *width*×*height*, how to find the nearest texels to an UV sample?



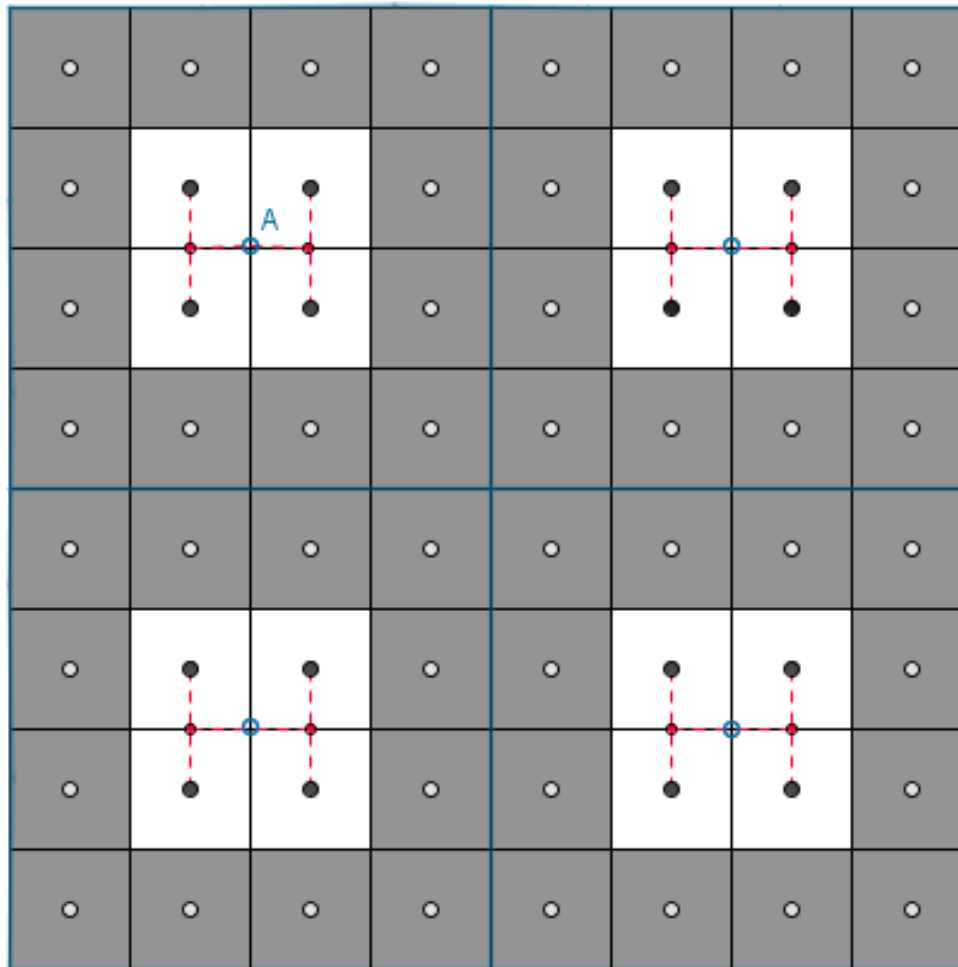
Downscale

- We can do the same interpolation for the downscale.

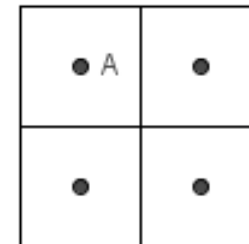


Downscale

- What can go wrong?



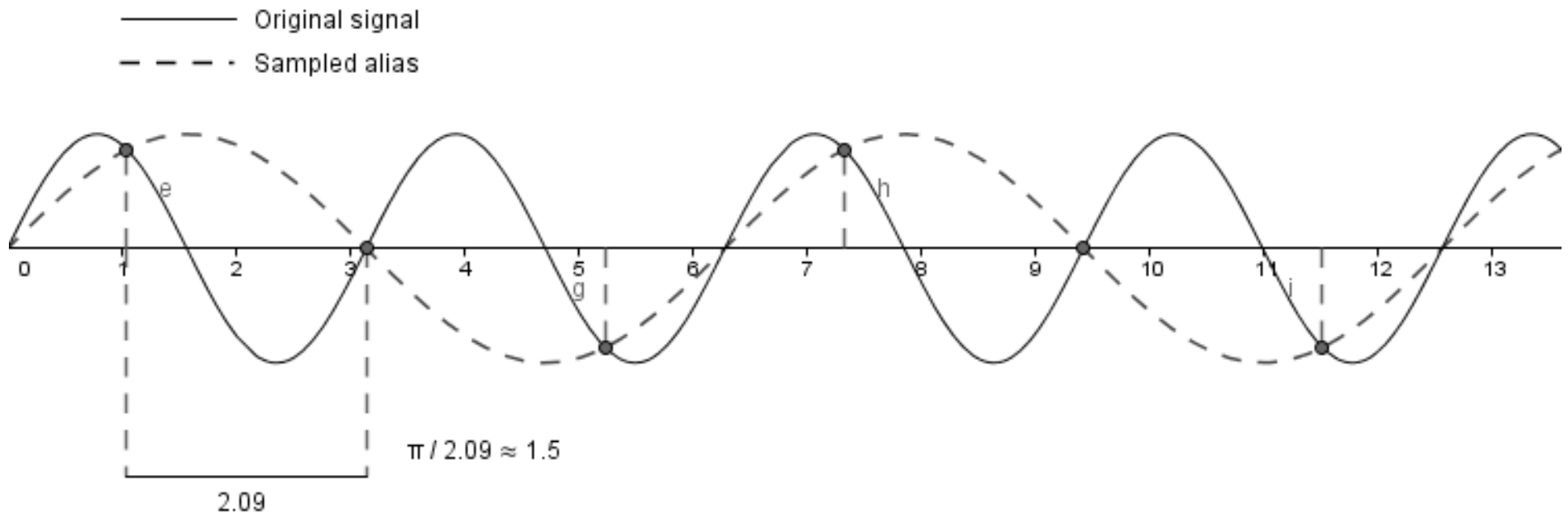
Texture



Downscaled

Nyquist–Shannon Sampling Theorem

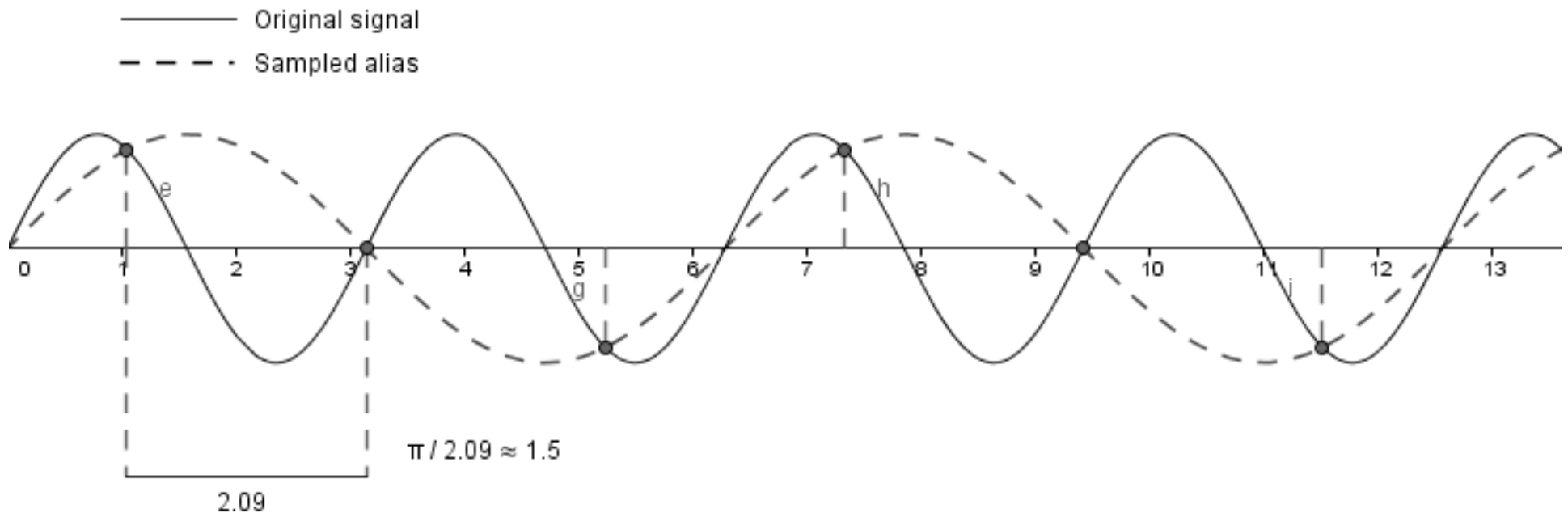
- In order to reconstruct a band-limited signal, one has to sample with sampling rate more than twice the highest frequency.*



This means more than 2 samples per period, every period.

Nyquist–Shannon Sampling Theorem

- *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

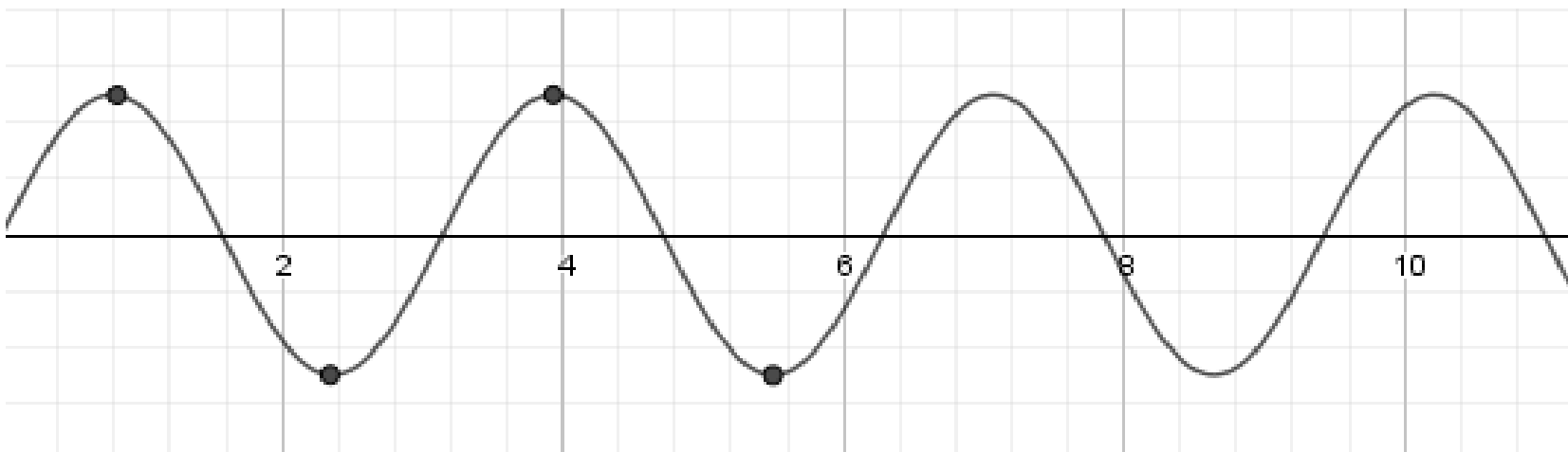
- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.

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.

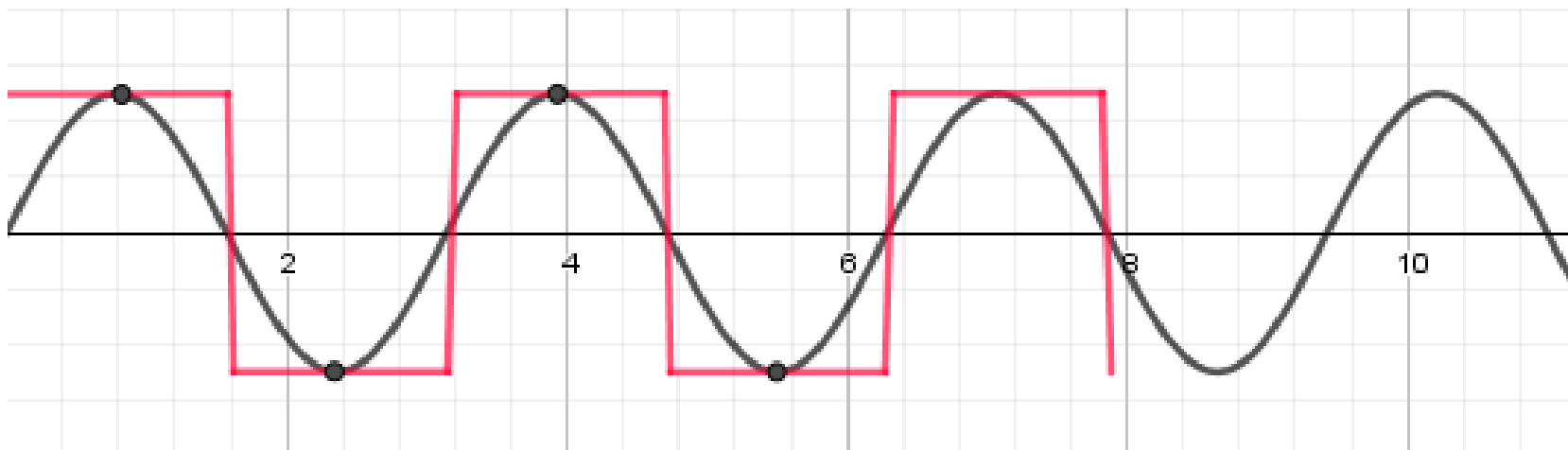
Nyquist–Shannon Sampling Theorem

- 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.

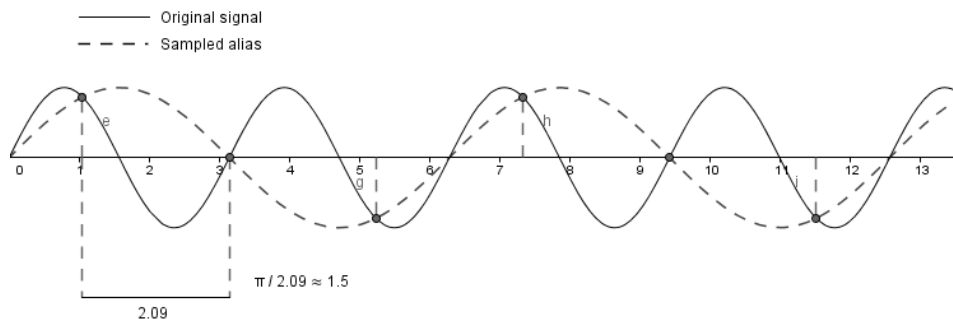


Nyquist–Shannon Sampling Theorem

- 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** – signal with a lower frequency.

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** – signal with a lower frequency.



Nyquist–Shannon Sampling Theorem

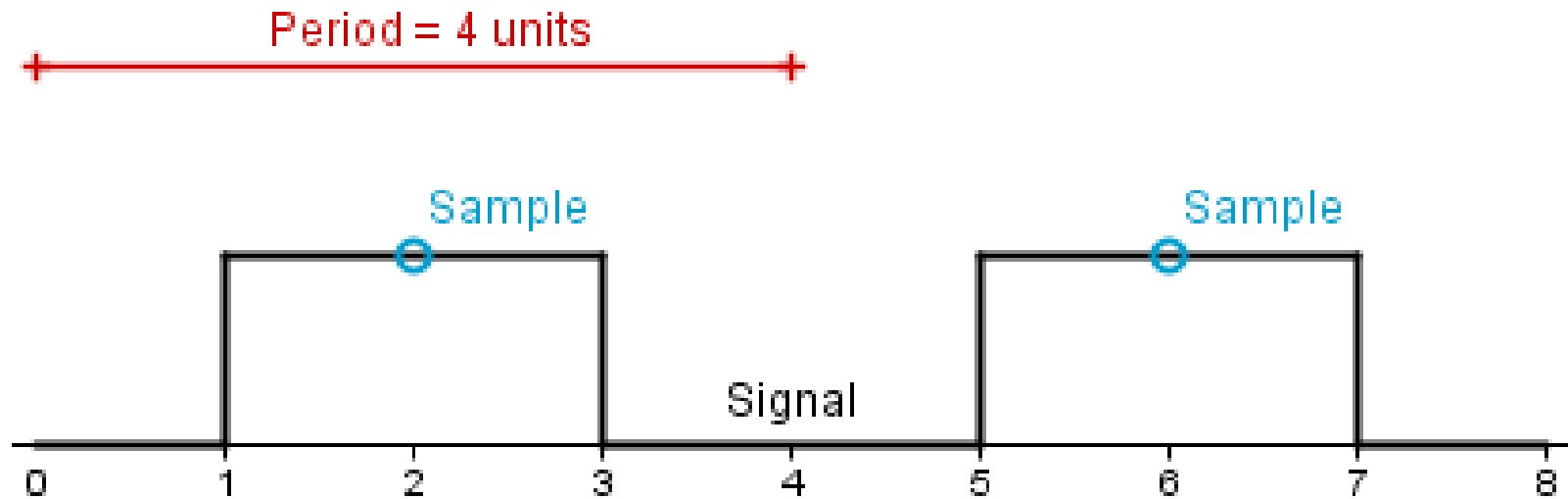
- Band-limited signal – there is a fixed highest frequency in the signal.
- The signals in real life are not band-limited.
- Reconstruction is possible only when we know the **shape** of the signal.
- Sampling less frequently, we produce **an alias** – signal with a lower frequency.
- Usually assumes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

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** – signal with a lower frequency.
- Usually assumes samples are taken over a length of time.
- **More info:** [Http://www.skillbank.co.uk/SignalConversion/rate.htm](http://www.skillbank.co.uk/SignalConversion/rate.htm)
<http://blogs.msdn.com/b/shawnhar/archive/2011/04/29/texture-aliasing.aspx>

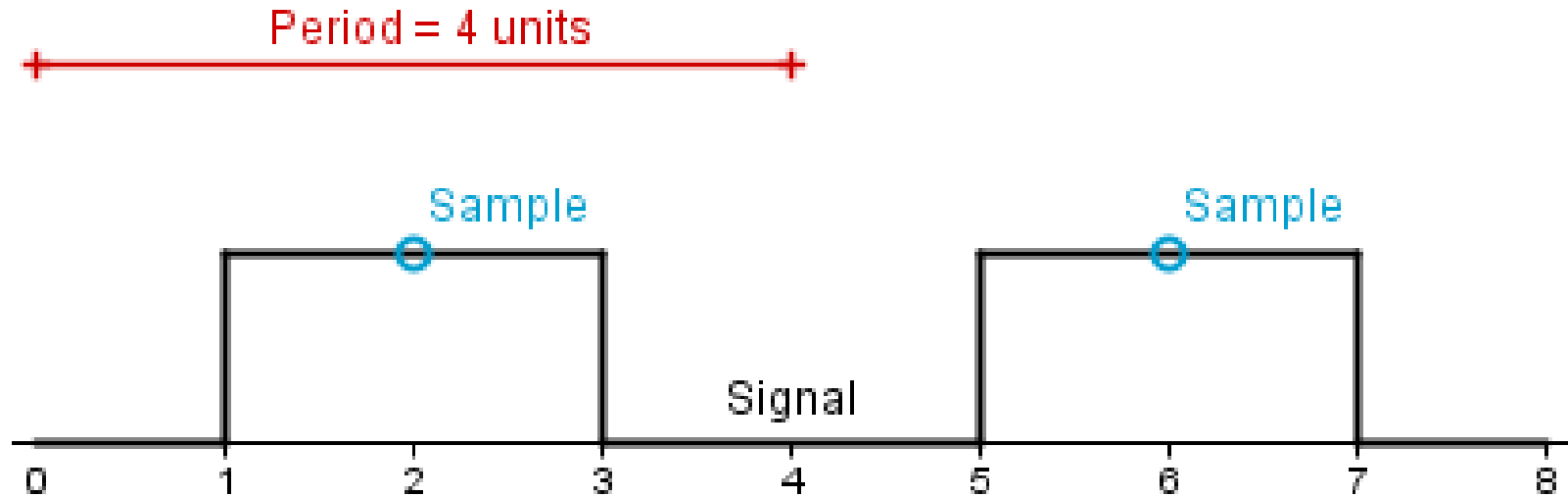
Downscale

- So, what is happening in our example?



Downscale

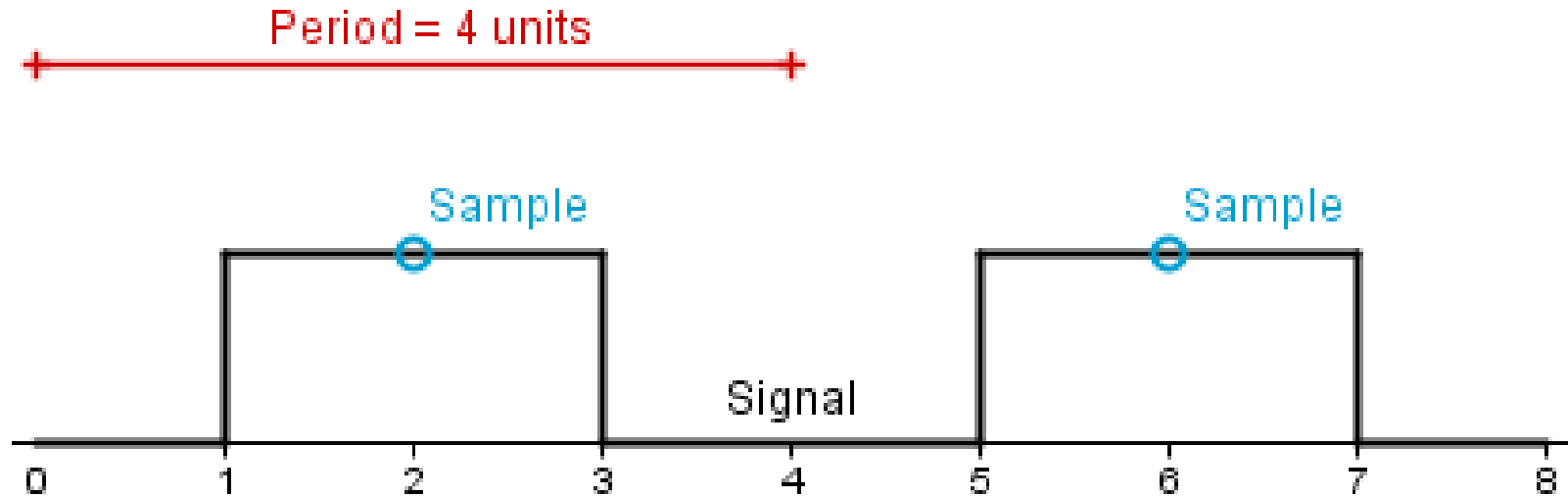
- So, what is happening in our example?



$$period = 4 \Rightarrow frequency = \frac{1}{4}$$

Downscale

- So, what is happening in our example?

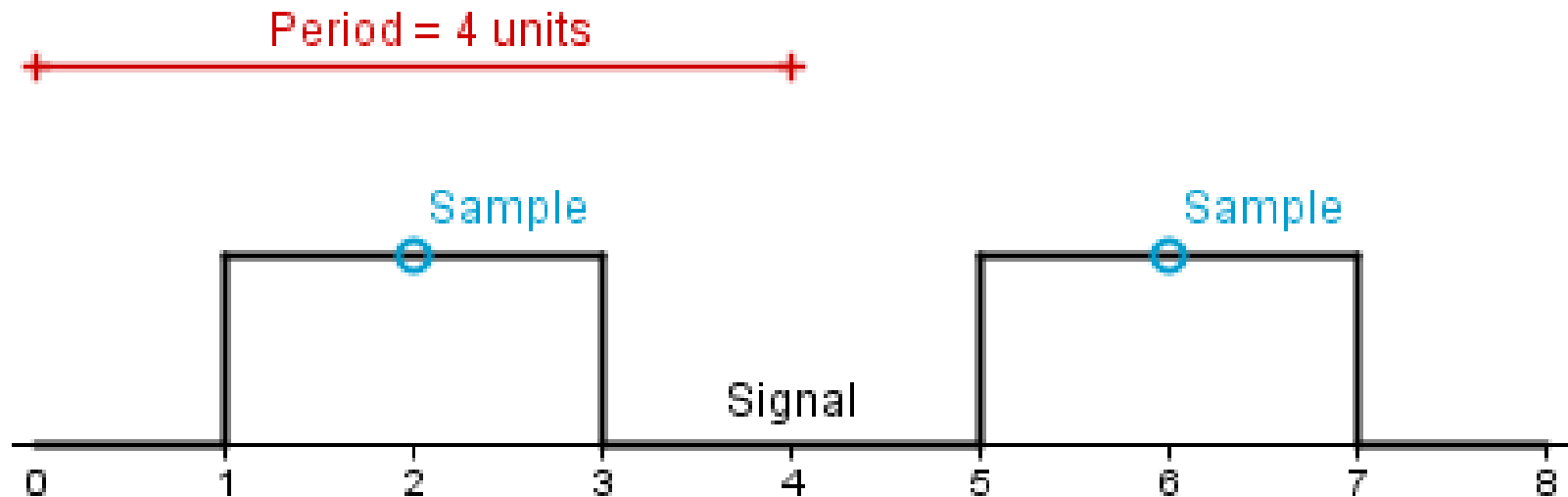


$$period = 4 \Rightarrow frequency = \frac{1}{4}$$

$$frequency_{Nyquist} = \frac{2}{4} = \frac{1}{2}$$

Downscale

- So, what is happening in our example?



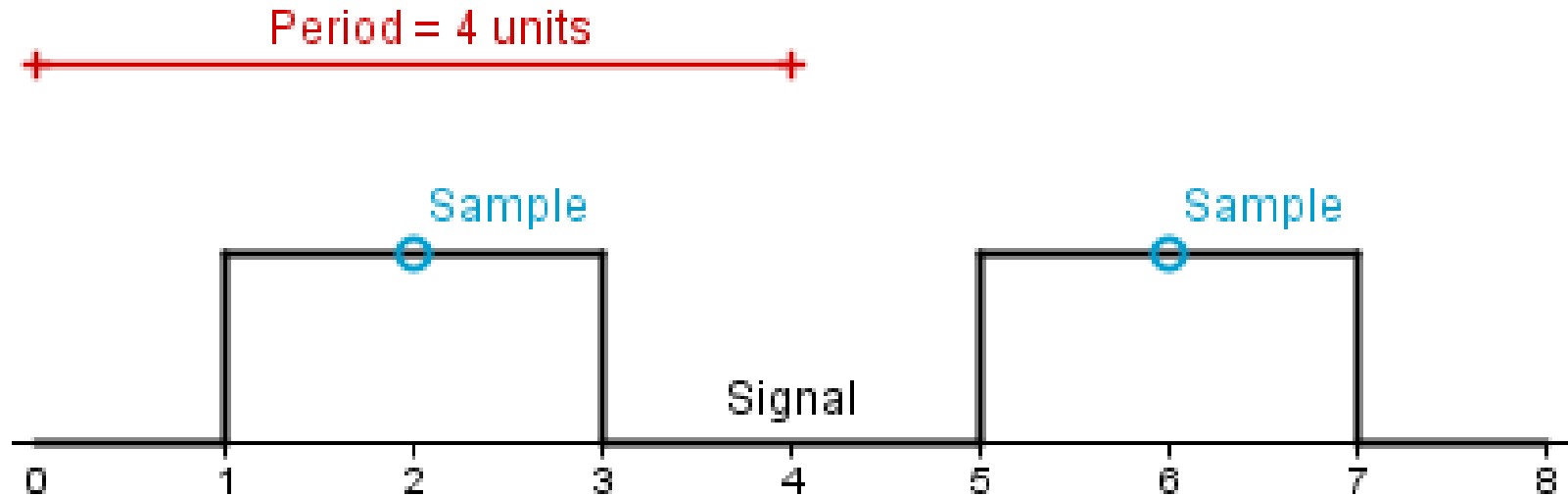
$$period = 4 \Rightarrow frequency = \frac{1}{4}$$

$$frequency_{Nyquist} = \frac{2}{4} = \frac{1}{2}$$

We need more than 1 sample per two units.

Downscale

- So, what is happening in our example?



$$period = 4 \Rightarrow frequency = \frac{1}{4}$$

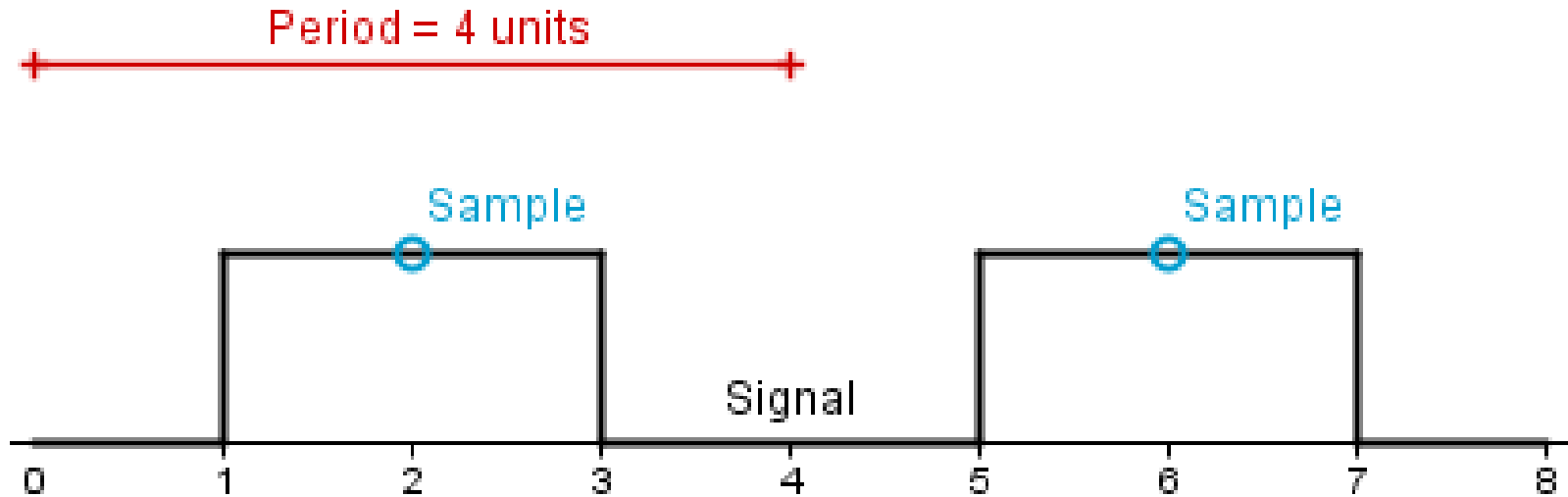
$$frequency_{Nyquist} = \frac{2}{4} = \frac{1}{2}$$

$$frequency_{Us} = \frac{1}{4} < \frac{1}{2}$$

We need more than 1 sample per two units.

Downscale

- So, what is happening in our example?



$$\text{period} = 4 \Rightarrow \text{frequency} = \frac{1}{4}$$

Aliasing

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

$$\text{frequency}_{U_s} = \frac{1}{4} < \frac{1}{2}$$

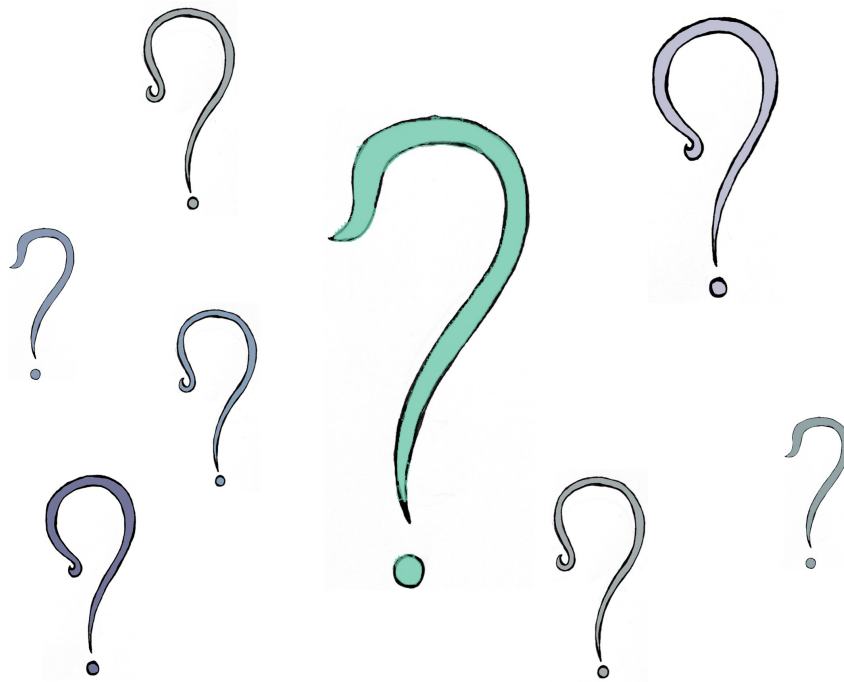
We need more than 1 sample per two units.

Downscale

- We need more than 1 sample per 2 units.

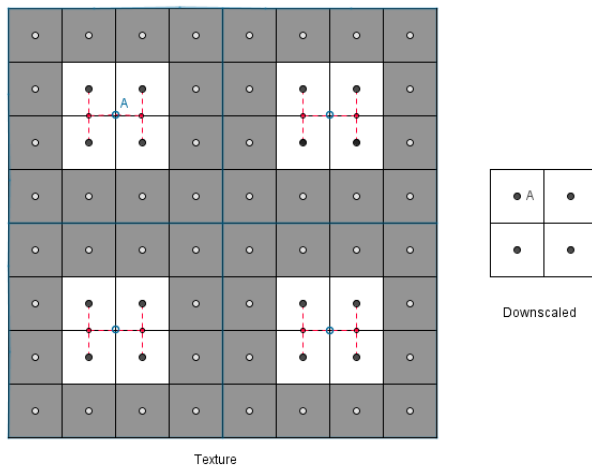
Downscale

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



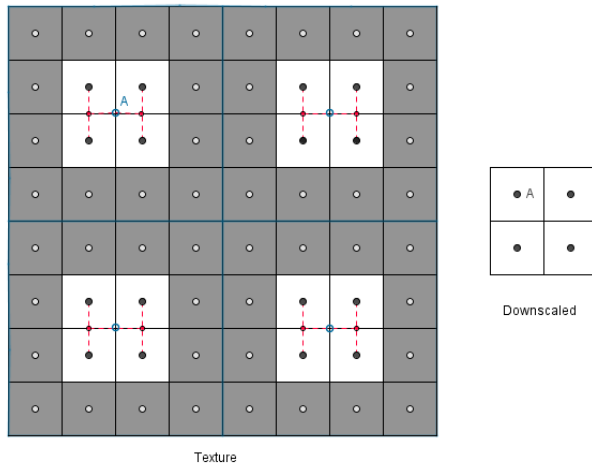
Downscale

- We do not want to create Moire aliasing.



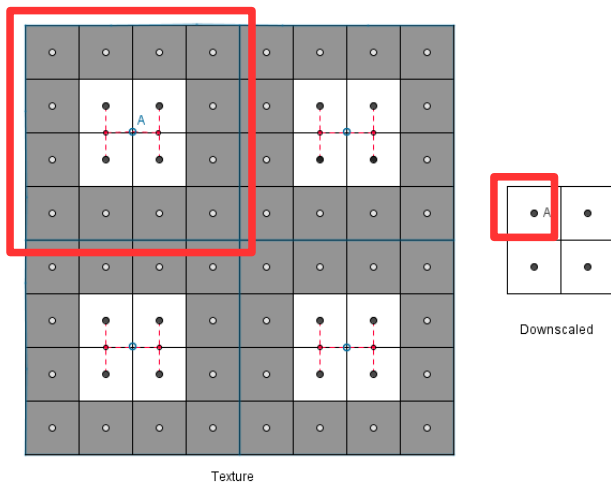
Downscale

- We do not want to create Moire aliasing.
- Our texture is not white, a 2×2 downscale should not be white either.



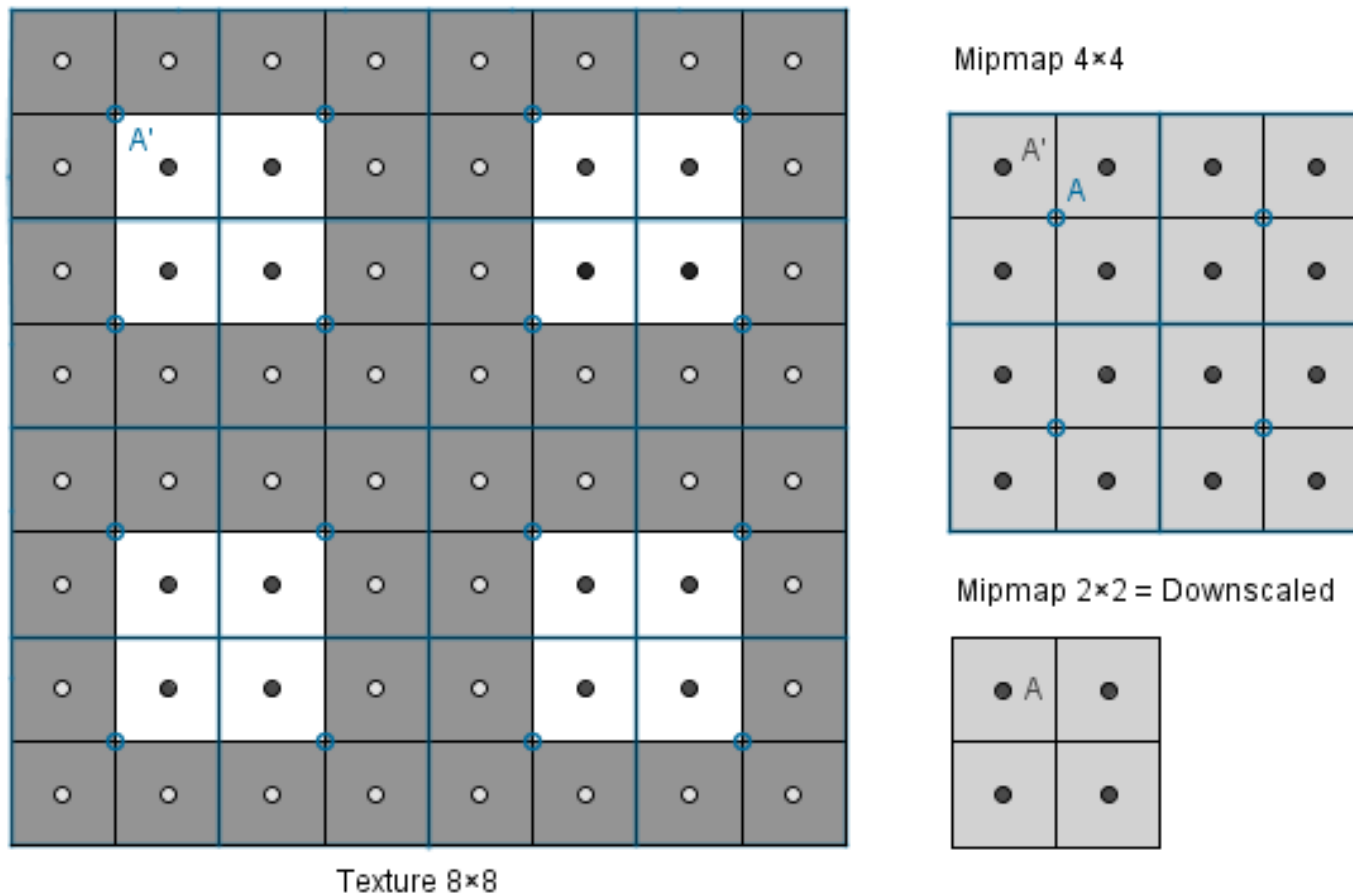
Downscale

- We do not want to create Moire aliasing.
- Our texture is not white, a 2×2 downscale should not be white either.
- 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 10×10 .

Mipmapping

- What if we have a texture that is 10×10 .
 - The first mipmap is the image itself – 10×10 .

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 10×10 .
 - The first mipmap is the image itself – 10×10 .
 - Then we take half the size – 5×5 .
 - Next we take half the size – $2.5 \dots$ Uh-oh.

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 – $2.5 \dots$ Uh-oh.
- The last mipmap we could create is 5×5 .

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 – $2.5 \dots$ 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 – $2.5 \dots$ 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.
- **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)



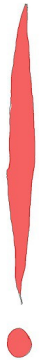
Filtering

- We have seen ways to sample the texture.
- Upscale (magnification filtering):
 - Nearest neighbour



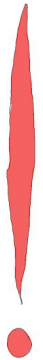
Filtering

- 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)**



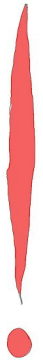
Filtering

- 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
- **Downscale (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



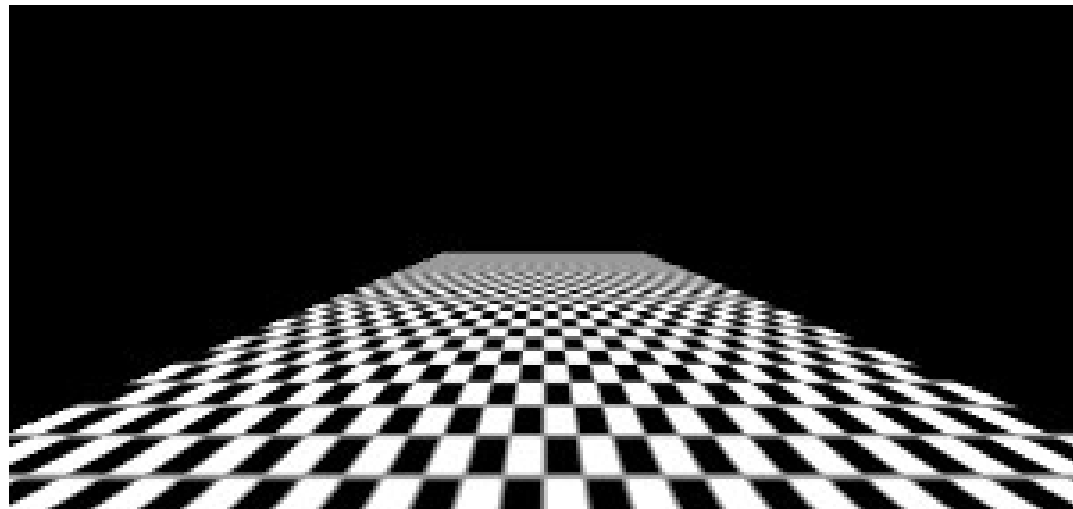
Questions?

Anisotropic Filtering

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

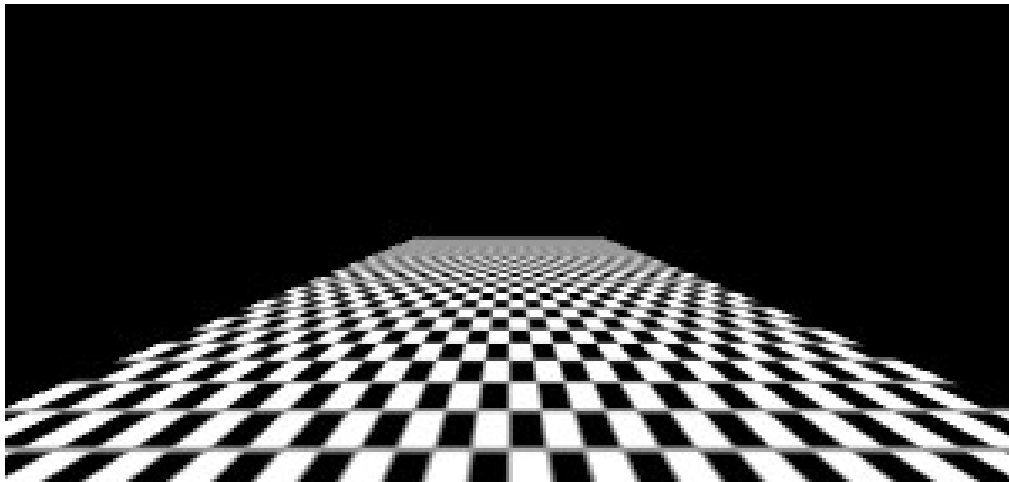
Anisotropic Filtering

- 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 10×5 area instead of 10×10 .



Anisotropic Filtering

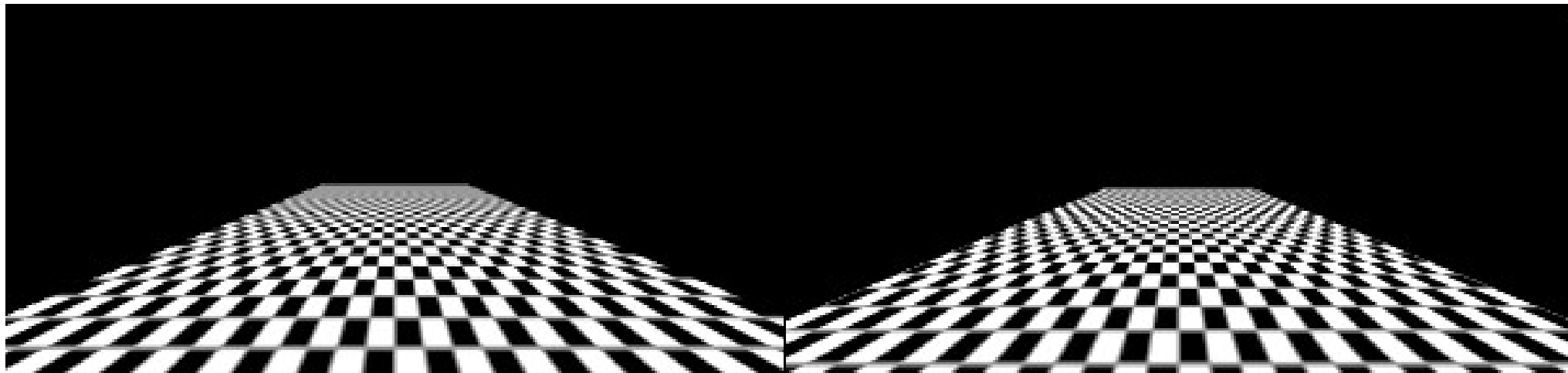
- We have more resolution in width than in height. It is unfair to average both dimensions equally.



No anisotropic filtering

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.

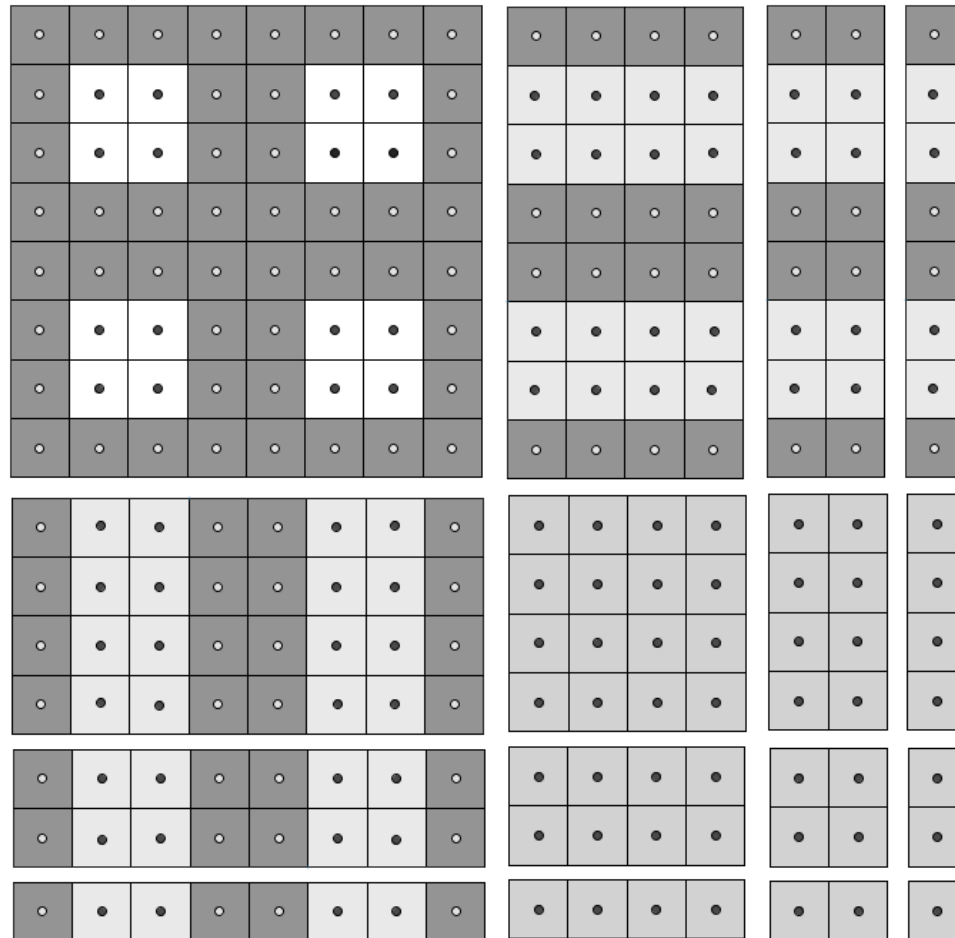


No anisotropic filtering

16x anisotropic filtering

Anisotropic Filtering

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



Textures

- There are more uses for textures than just storing granular color of a material.

Textures

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

Textures

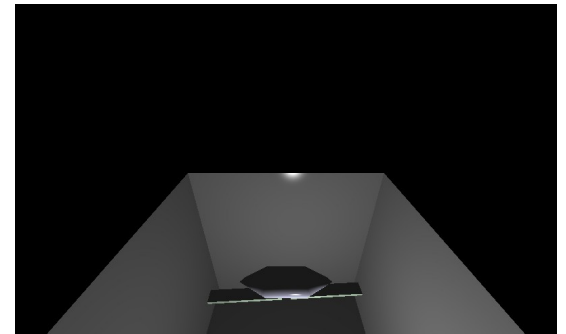
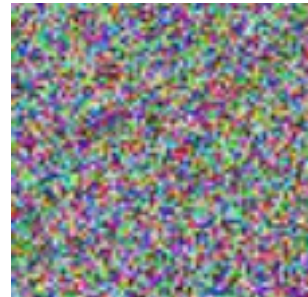
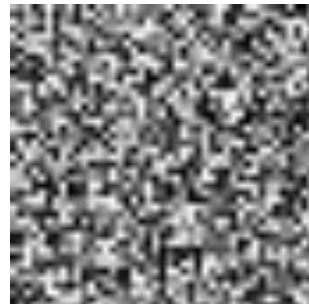
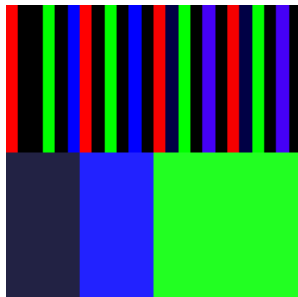
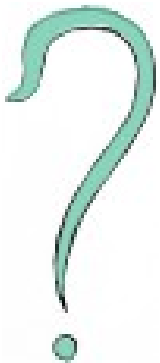
- There are more uses for textures than just storing granular color of a material.
 - **Data textures** – we can hold other data like normals or other values with 3 (or 4) coordinates.
 - **Noise texture** – we can store samples of a random function in a texture to procedurally generate things like the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

Textures

- There are more uses for textures than just storing granular color of a material.
 - **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 in a texture to procedurally generate things like the Perlin noise.
 - **Render target** – we could also render the current framebuffer to a texture.

Textures

- There are more uses for textures than just storing granular color of a material.
 - **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 in a texture to procedurally generate things like the Perlin noise.
 - **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*