Computergrafik

Universität Bern
Herbst 2012

Reminder regarding turn-in

- Turn-in as usual in 2 weeks until noon on ilias (**november 1st**)

- Keep in mind that you cannot turn-in stuff on Ilias after the deadline, so missing it will result in a **late penalty (=50% of original score)**

Reminder regarding turn-in

- If possible use the time-slots on Thursday!

- No more time-slots after Friday 2 pm.
  - Showing it later will result in **late penalty (=50% of original score)**

Reminder regarding turn-in

- The code you upload on ilias has to be the same you show us in the pool!
  - No more advantage by taking a later time-slot!
  - We will check some of the turn-ins on ilias on a random basis.
  - Showing us different code will also result in **late penalty (=50% of original score)**.

Assignment 3: Rasterization

- Until now:
  - Rasterization done by Hardware (GPU)
  - Black box:

    Vertexdata goes in, Image comes out...

Assignment 3: Rasterization

- Now:
  - Rasterization done by Software
  - Code your own rasterizer in java...

Assignment 3: Rasterization

- Modify ***SWRenderContext.draw*** from the jrtr
  - This method is called once per shape

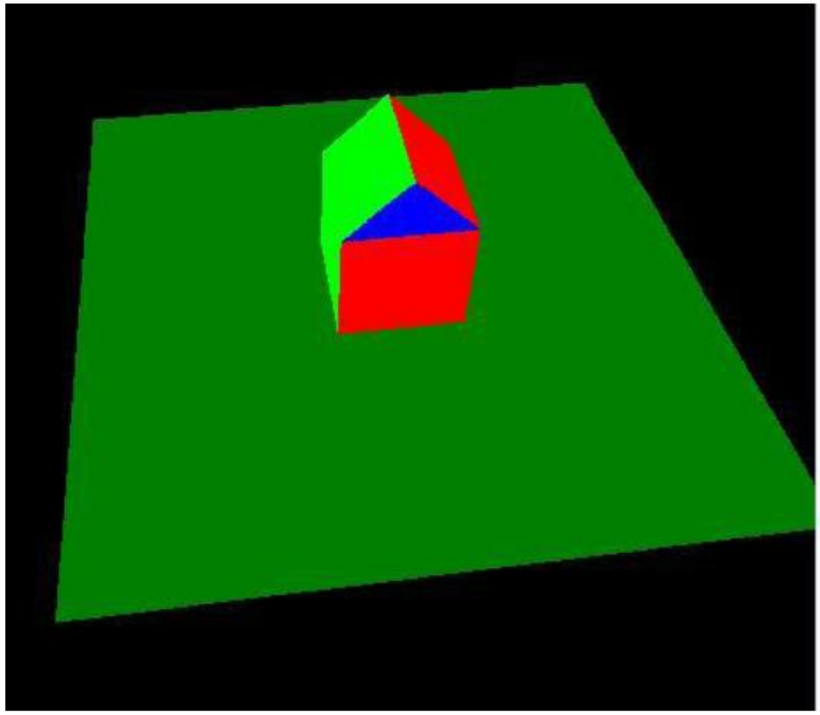
- Test it by deriving ***SimpleRenderPanel*** from ***SWRenderPanel*** instead of ***GLRenderPanel***

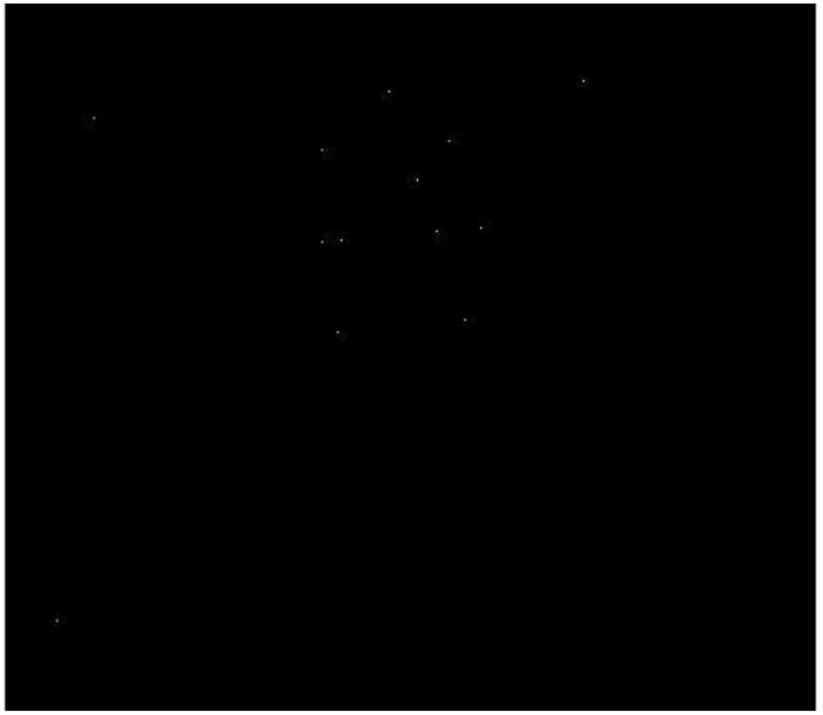
```
public final static class SimpleRenderPanel extends GLRenderPanel
```

```
public final static class SimpleRenderPanel extends SWRenderPanel
```

Assignment 3: Render Vertices

- Project vertices onto screen and set color to white





Assignment 3: Render Vertices

- Project 3D-Objectcoordinates to 2D-Pixelcoordinates

$$\mathbf{p}' = \mathbf{DPC}^{-1}\mathbf{Mp}$$

Object space

World space

Camera space

Canonic view volume

Image space

$$\mathbf{p}' = \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

Pixel coordinates $\begin{matrix} x'/w' \\ y'/w' \end{matrix}$

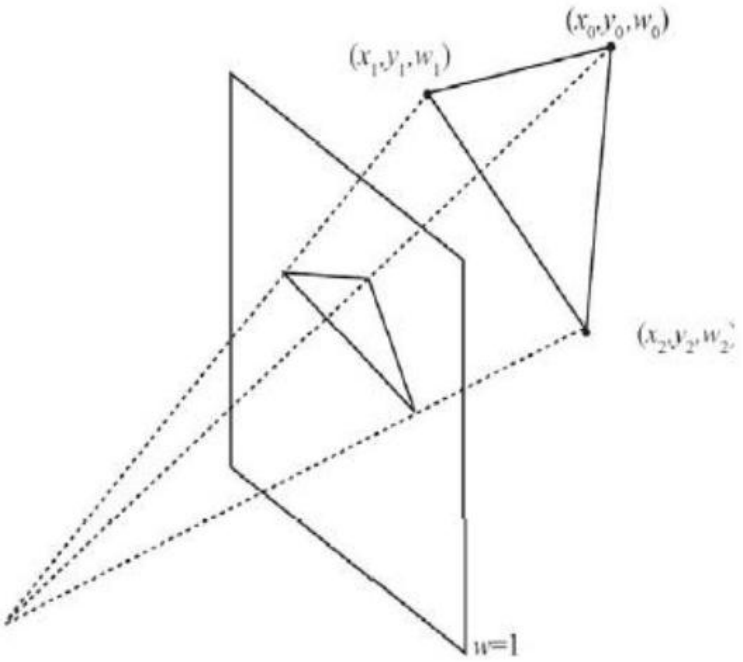
- Store Pixelcolors into a *BufferedImage*
  - Pixel 0,0 is the upper left corner

Assignment 3: Rasterization and Z-Buffering

- Homogenious Rasterization:
  - http://www.cgg.unibe.ch/teaching/previous-courses/hs-08/computergrafik/Homogeneous_rasterization.pdf

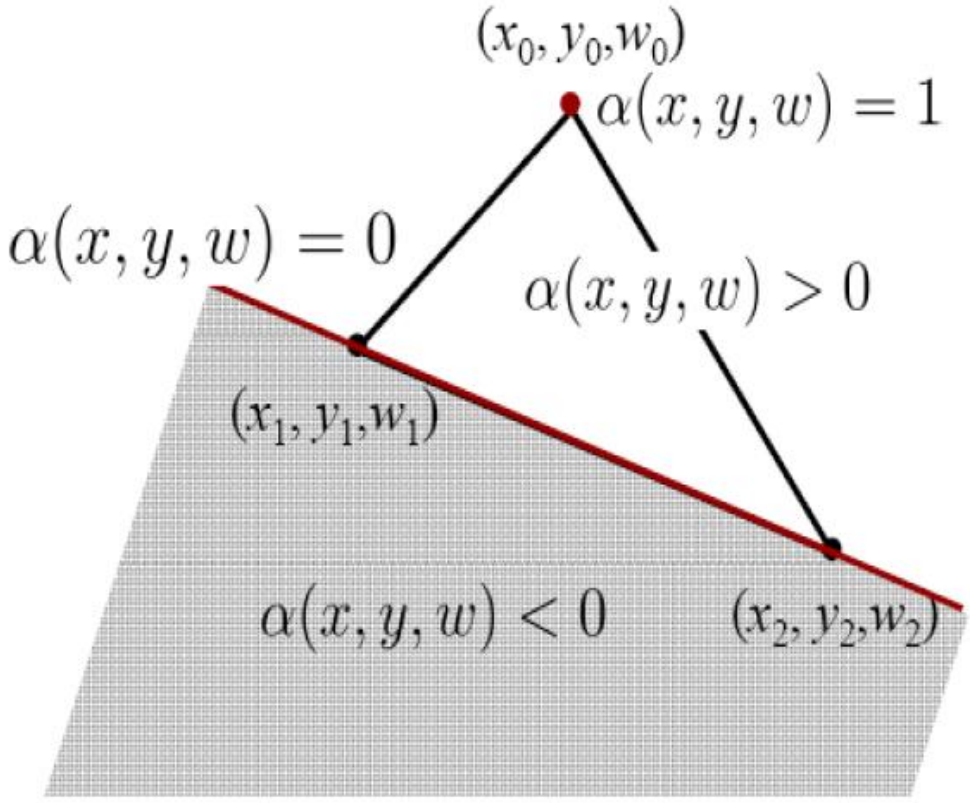
Assignment 3: Rasterization and Z-Buffering

Step 1: Transform vertex coordinates into 2D homogeneous coordinates (x,y,**w**) ***before*** performing the homogeneous division!



Assignment 3: Rasterization and Z-Buffering

Step 2: For each triangle compute edge function α,β,γ



(analogous for β and γ)

Assignment 3: Rasterization and Z-Buffering

Step 2: For each triangle compute edge function α,β,γ

- Compute the coefficients of the edge functions

$$\alpha(x,y,w) = a_\alpha x + b_\alpha y + c_\alpha w$$
$$\beta(x,y,w) = a_\beta x + b_\beta y + c_\beta w$$
$$\gamma(x,y,w) = a_\gamma x + b_\gamma y + c_\gamma w$$

$$\begin{bmatrix} a_\alpha & a_\beta & a_\gamma \\ b_\alpha & b_\beta & b_\gamma \\ c_\alpha & c_\beta & c_\gamma \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & w_0 \\ x_1 & y_1 & w_1 \\ x_2 & y_2 & w_2 \end{bmatrix}^{-1}$$

Assignment 3: Rasterization and Z-Buffering

Step 3: Check if a point (x/w, y/w) is inside a triangle

$$\alpha/w = a_\alpha(x/w) + b_\alpha(y/w) + c_\alpha$$
$$\beta/w = a_\beta(x/w) + b_\beta(y/w) + c_\beta$$
$$\gamma/w = a_\gamma(x/w) + b_\gamma(y/w) + c_\gamma$$

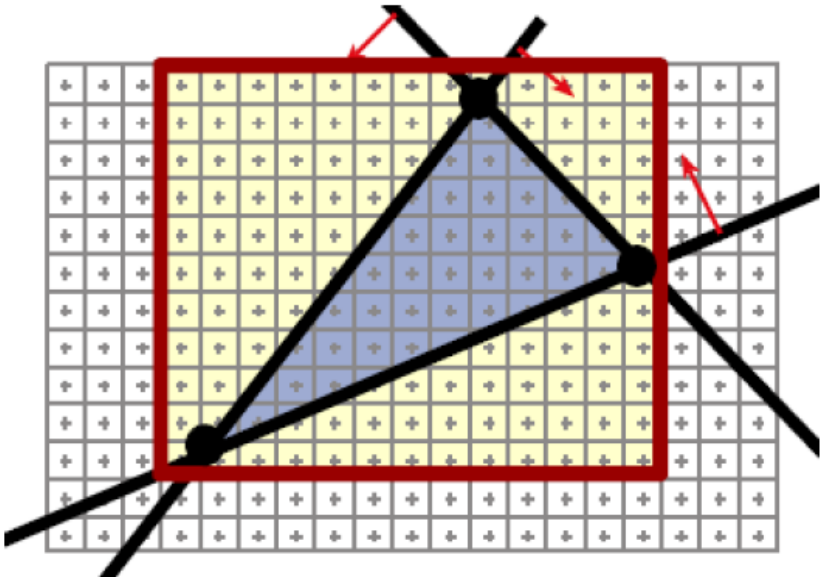
A point is inside if

$$\alpha/w > 0 \wedge \beta/w > 0 \wedge \gamma/w > 0$$

Assignment 3: Rasterization and Z-Buffering

Step 3: Check if a point (x/w, y/w) is inside a triangle

- We don't want to do this test for each triangle and pixel!
  - Perform test for only for pixels inside of the **bounding box** of a triangle



Assignment 3: Rasterization and Z-Buffering

Step 4: Perspective correct color interpolation

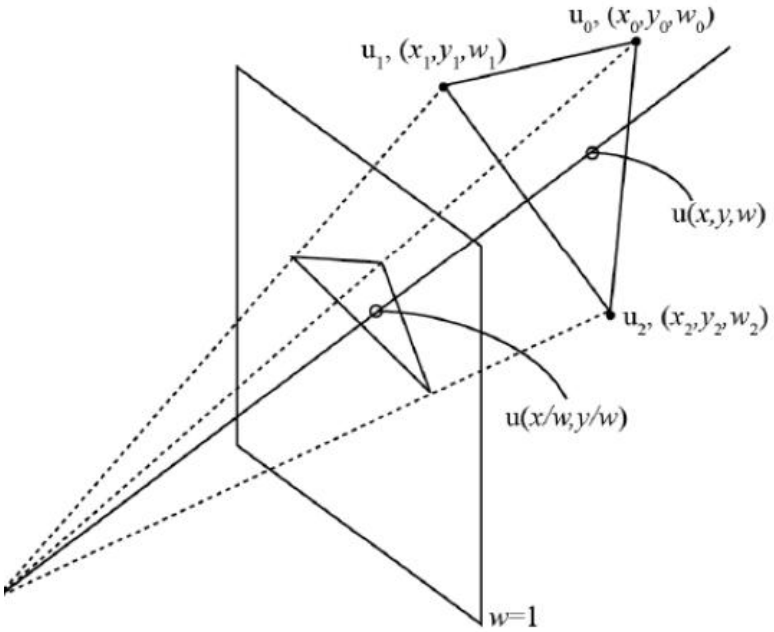
- Must be done for R-,G- and B-channel **separatly**!

Assignment 3: Rasterization and Z-Buffering

Step 4: Perspective correct color interpolation

- Must be done for R-,G- and B-channel **separatly**!

$$u(x/w, y/w) = \frac{(u/w)}{(1/w)}$$



compute (u/w) for R,G abd B
and compute also (1/w)

(see script)

Assignment 3: Rasterization and Z-Buffering

And finally... **Z-Buffering**

- The z-buffer stores for each drawn pixel (x/w, y/w) the value **1/w**

Assignment 3: Rasterization and Z-Buffering

And finally... **Z-Buffering**

- The z-buffer stores for each drawn pixel (x/w, y/w) the value **1/w**
- Before drawing a pixel into the ImageBuffer compare 1/w with the value stored in the z-buffer

Assignment 3: Rasterization and Z-Buffering

And finally... **Z-Buffering**

- The z-buffer stores for each drawn pixel (x/w, y/w) the value **1/w**

- Before drawing a pixel into the ImageBuffer compare 1/w with the value stored in the z-buffer

If z-buffer value is smaller

Draw new pixel into ImageBuffer and overwrite z-Buffer value with the new 1/w value

else do nothing

Assignment 3: Texture Mapping

Preparation

- Modify ***jrtr.SWTexture.load*** in such a way that textures can be loaded

use

```
BufferedImage texture;
File f = new File(fileName);
texture = ImageIO.read(f);
```

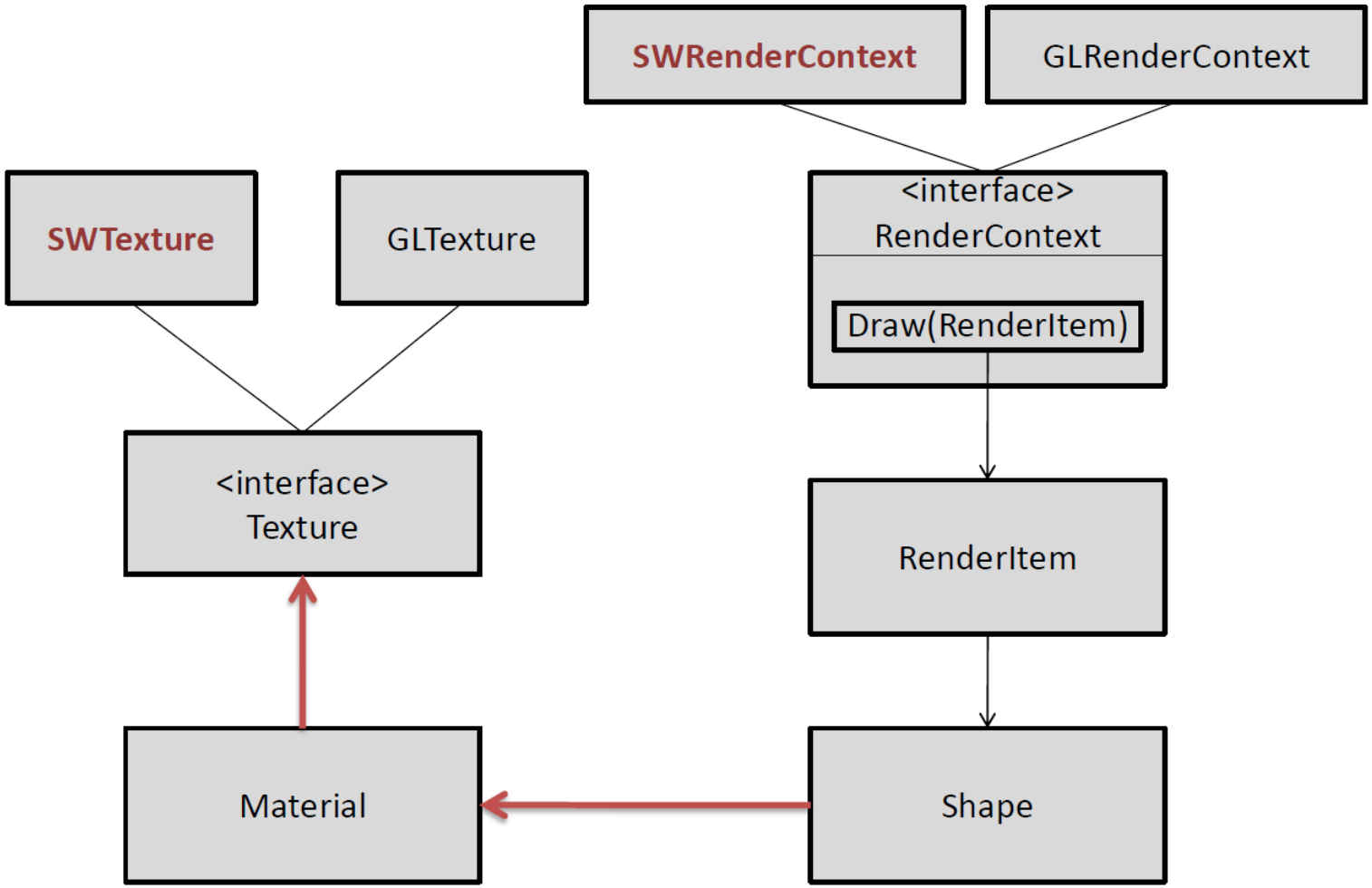
Assignment 3: Texture Mapping

Preparation

- Extend class ***jrtr.Material*** by a reference on a texture
- Extend class ***jrtr.Shape*** by a reference on a material

Assignment 3: Texture Mapping

Preparation



Assignment 3: Texture Mapping

- Texturecoordinates can be passed to the vertexdata in the same way as vertexposition, color or normals

Example:

```
vertexData.addElement(texdata, VertexData.Semantic.TEXCOORD, 2);
```

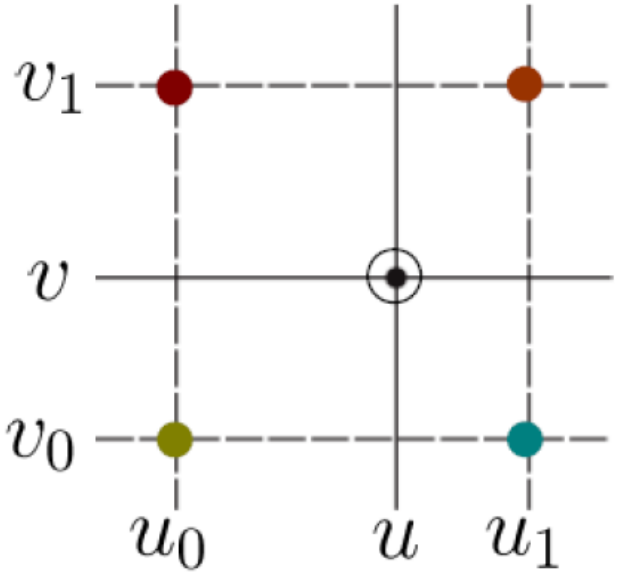
- As a convention, texturecoordinates are passed as values between (0,0) and (1,1)

Assignment 3: Texture Mapping

- Perspective correct interpolation has to be applied also to texture coordinates...
  - Attributes to interpolate are u and v
  - Can be done in the same way as the perspective correct color interpolation...

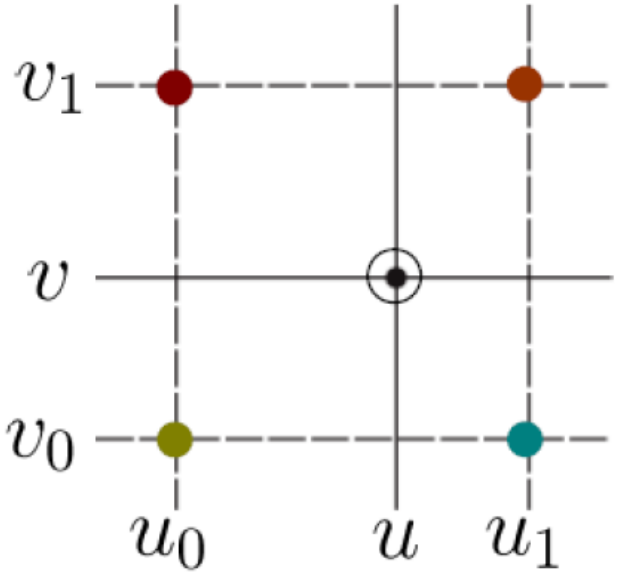
Assignment 3: Texture Mapping

- Resulting texure coordinates are not necessarely in the center of image-pixels



Assignment 3: Texture Mapping

- Resulting texure coordinates are not necessarely in the center of image-pixels



=> we also need to interpolate for in-between pixel positions

Assignment 3: Texture Mapping

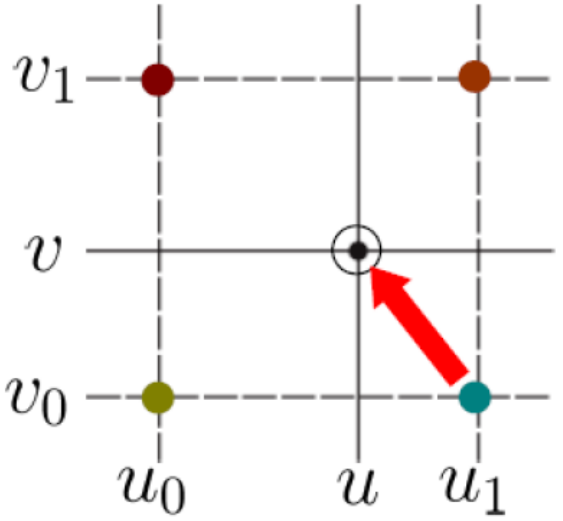
- Two different texture look-up interpolation methods have to be implemented:

Assignment 3: Texture Mapping

- Implement two different texture look-up interpolation methods:

- **nearest-neighbor interpolation**

  - Take the closest pixelvalue!

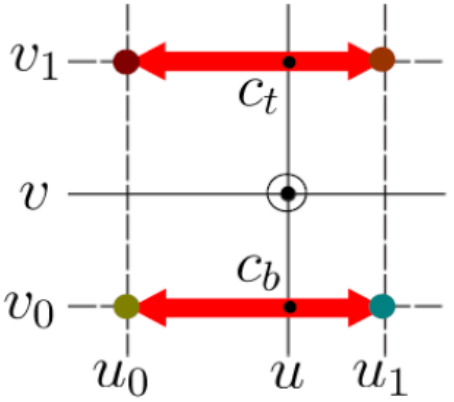


Assignment 3: Texture Mapping

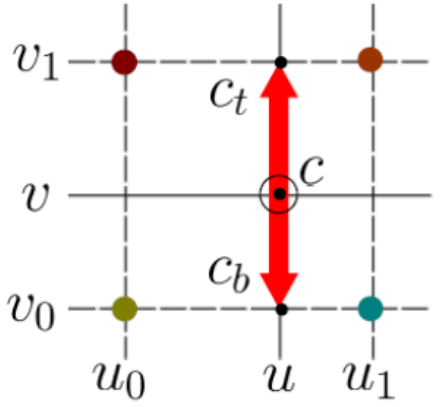
- Implement two different texture look-up interpolation methods:

- **bilinear interpolation**
- Two steps:

    1: horizontal linear interpolation



    2: vertical linear interpolation



Assignment 3: Texture Mapping

- For the demostration apply a texture onto the torus from assignment 1, the fractal landscape of assignment 2 or the simple house

...you should start early with this assignment...

Questions?