IG3DA

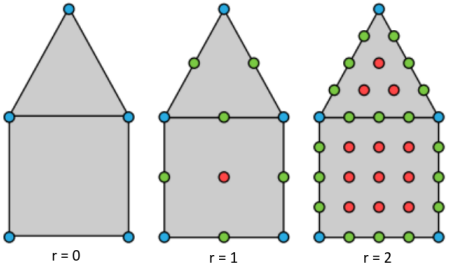
Project: Mesh Color textures

Vannvatthana Norng

Télécom Paris

09 February 2024

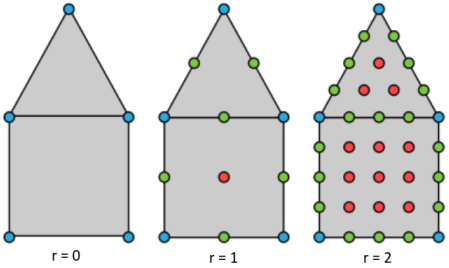
Mesh Color



Mesh color with resolution $R = 2^r - 1$

- Stores color data directly on the polygonal mesh (Vertex + Edge + Face colors)

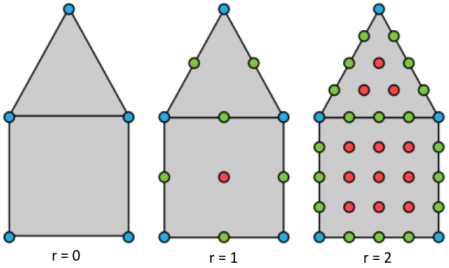
Mesh Color



Mesh color with resolution $R = 2^r - 1$

- Stores color data directly on the polygonal mesh (Vertex + Edge + Face colors)
- Avoid texture parameterization

Mesh Color



Mesh color with resolution $R = 2^r - 1$

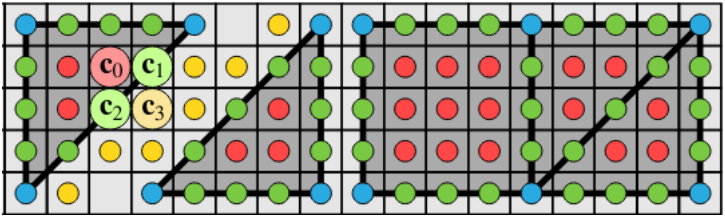
- Stores color data directly on the polygonal mesh (Vertex + Edge + Face colors)
- Avoid texture parameterization
- Avoid discontinuities occuring at texture seams

Mesh Color Texture

- **Objective :** Convert mesh color data to 2D textures that is ideal for the GPU at the time

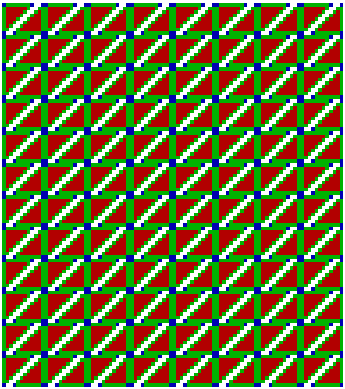
Mesh Color Texture

- **Objective :** Convert mesh color data to 2D textures that is ideal for the GPU at the time
- **How :**
  - Assign separately 2D texture coordinates for each face
  - Edges have to be vertical, horizontal or diagonal along the texel



2D Texture Layout

2D Texture Layout : My Result



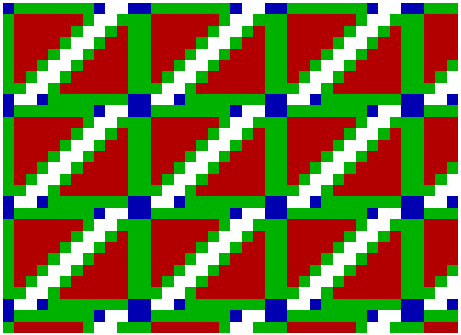


Figure – RGB for vertex :(0,0,0.7), edge : (0,0.7,0), face :(0.7,0,0)

Bilinear Filtering

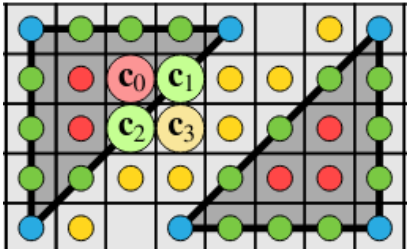


Figure – Bilinear Filtering

- For bilinear filtering, we need to interpolate the texels between the triangles (to avoid seams)
- $c_3 = c_1 + c_2 - c_0$

Bilinear Filtering

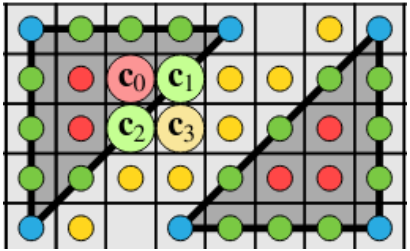
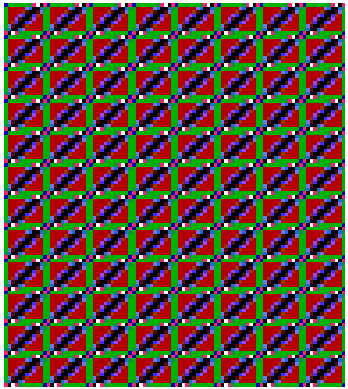


Figure – Bilinear Filtering

- For bilinear filtering, we need to interpolate the texels between the triangles (to avoid seams)
- $c_3 = c_1 + c_2 - c_0$
- if R, G or B of c_3 is greater than 1 or smaller than 0 :

$$\begin{cases} c_1 & = c_1 + \delta \\ c_2 & = c_2 + \delta \\ c_0 & = c_0 - \delta \end{cases} \text{ where } \delta = \begin{cases} -c_3/3 & \text{if } < 0 \\ (1 - c_3)/3 & \text{if } > 1 \end{cases}$$

Interpolation for bilinear filtering : result



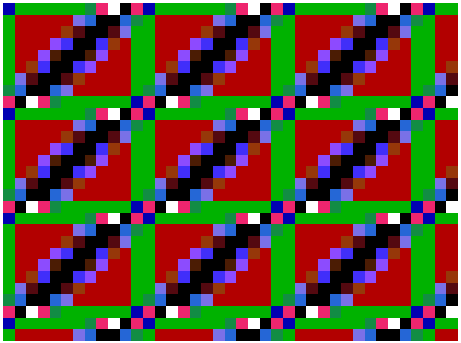


Figure – After interpolating for gap texels

4D Texture coordinates for mipmaps

Algorithm 1: Texture layout generation with 4D coordinates.

```
Form rectangles from faces.
Sort rectangles from large to small.
u_s ← [0 0]
u_δ ← [0.5 0.5]
foreach rectangle do
    if reached end of row then
        Find next row
        u_δ ← u_δ + [0 b]
        u_s ← next row starting position −u_δ
    end
    Make sure that u_s is an even multiple of 2^r_i.
    Place rectangle using the four texture coordinates:
        u_s, u_δ
        u_s + [0 2^r_i], u_δ
        u_s + [2^r_i 0], u_δ
        u_s + [2^r_i 2^r_i], u_δ
    u_s,x ← u_δ,x + 2^r_i
    u_δ,x ← u_s,x + X_i + b
end
```

Figure – 4D Texture Coordinates Algorithm

For trilinear filtering, the texture coordinates of mipmap level l can be computing as follow

$$u_l = \frac{u_s}{2^l} + u_\delta$$

, where

- u_s : scalable coordinates, it must be divisible by 2^r or $R + 1$
- u_δ : constant offsets

4D Texture coordinates for mipmaps

Algorithm 1: Texture layout generation with 4D coordinates.

```
Form rectangles from faces.
Sort rectangles from large to small.
u_s ← [0 0]
u_δ ← [0.5 0.5]
foreach rectangle do
    if reached end of row then
        Find next row
        u_δ ← u_δ + [0 b]
        u_s ← next row starting position −u_δ
    end
    Make sure that u_s is an even multiple of 2^{r_i}.
    Place rectangle using the four texture coordinates:
        u_s, u_δ
        u_s + [0 2^{r_i}], u_δ
        u_s + [2^{r_i} 0], u_δ
        u_s + [2^{r_i} 2^{r_i}], u_δ
    u_{s,x} ← u_{δ,x} + 2^{r_i}
    u_{δ,x} ← u_{s,x} + X_i + b
end
```

Figure – 4D Texture Coordinates Algorithm

For trilinear filtering, the texture coordinates of mipmap level l can be computing as follow

$$u_l = \frac{u_s}{2^l} + u_\delta$$

, where

- u_s : scalable coordinates, it must be divisible by 2^r or $R + 1$
- u_δ : constant offsets

Remark : this 4D coordinates system can only generate mipmaps level to vertex color

Mipmaps : result

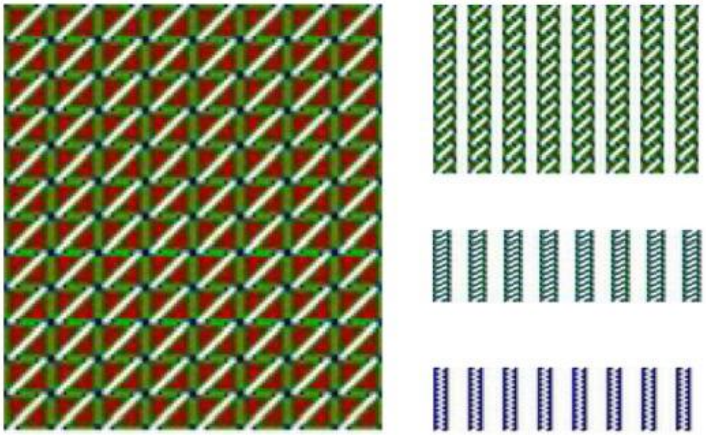


Figure – 4D Texture Coordinates Algorithm

Conclusion

- Mesh color data can be converted into a form of 2D texture that is suitable for GPU at the time (2017) in Bilinear Filtering and trilinear filtering.
- Solution for Anisotropic Filtering was not provided
- Mipmap generation beyond vertex color, and for faces with non-uniform resolutions are discussed in the paper but I did not implement them.