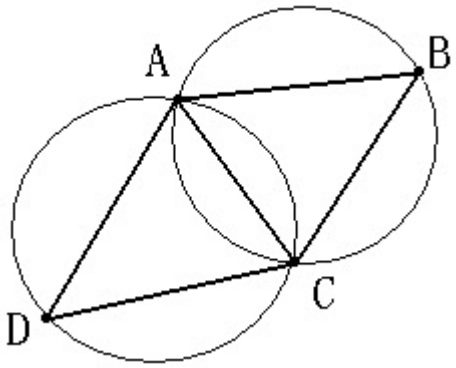
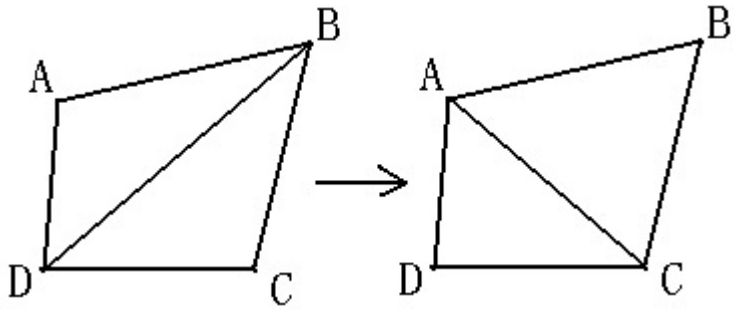
Tutorial 5 : Texture

Chenqi Luo

Quiz

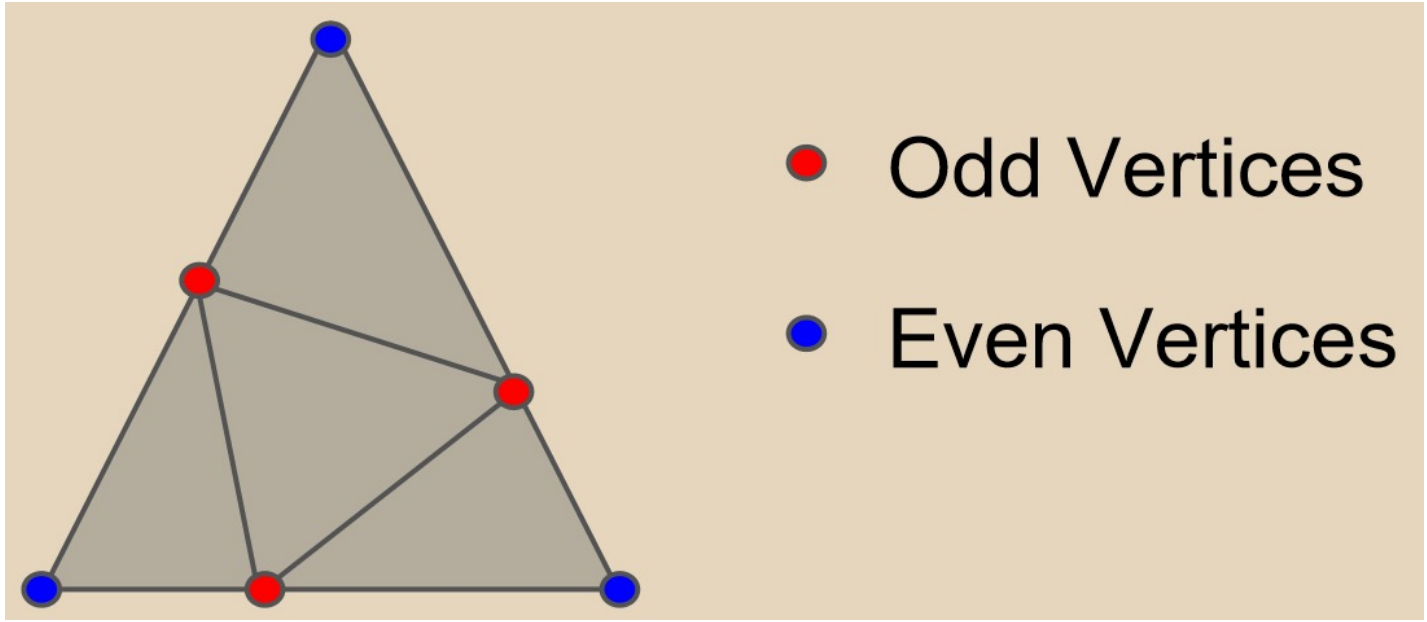
- 1. Empty Circumscribed circle
- 2. Maximum minimum angle



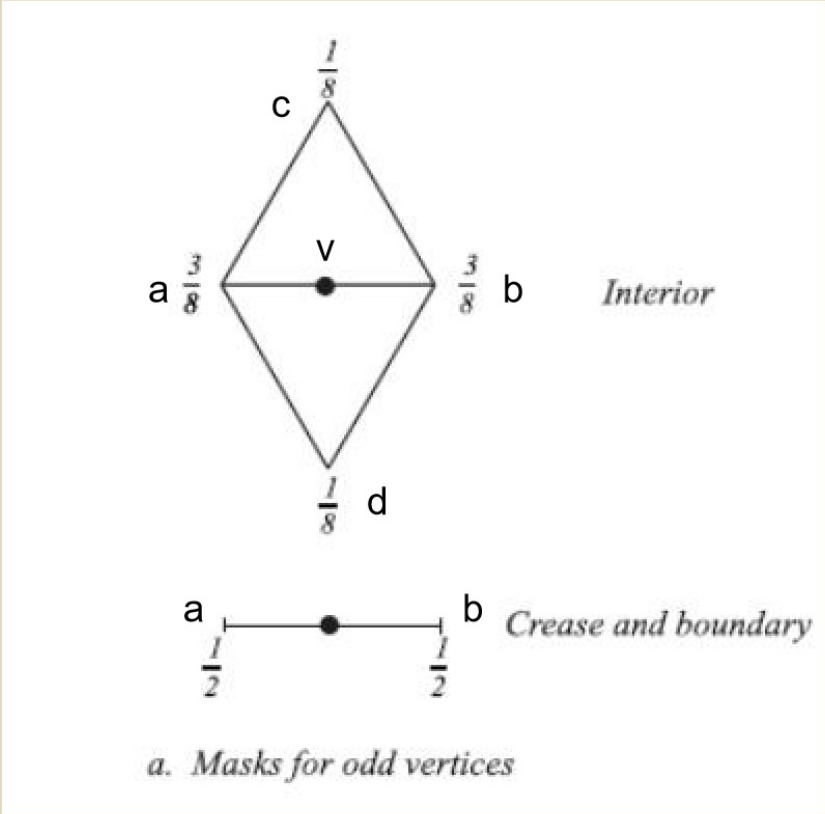


Loop subdivision

Newly created vertices are called odd vertices
Original vertices are called even vertices



- Computing odd vertices



a. *Masks for odd vertices*

Interior:

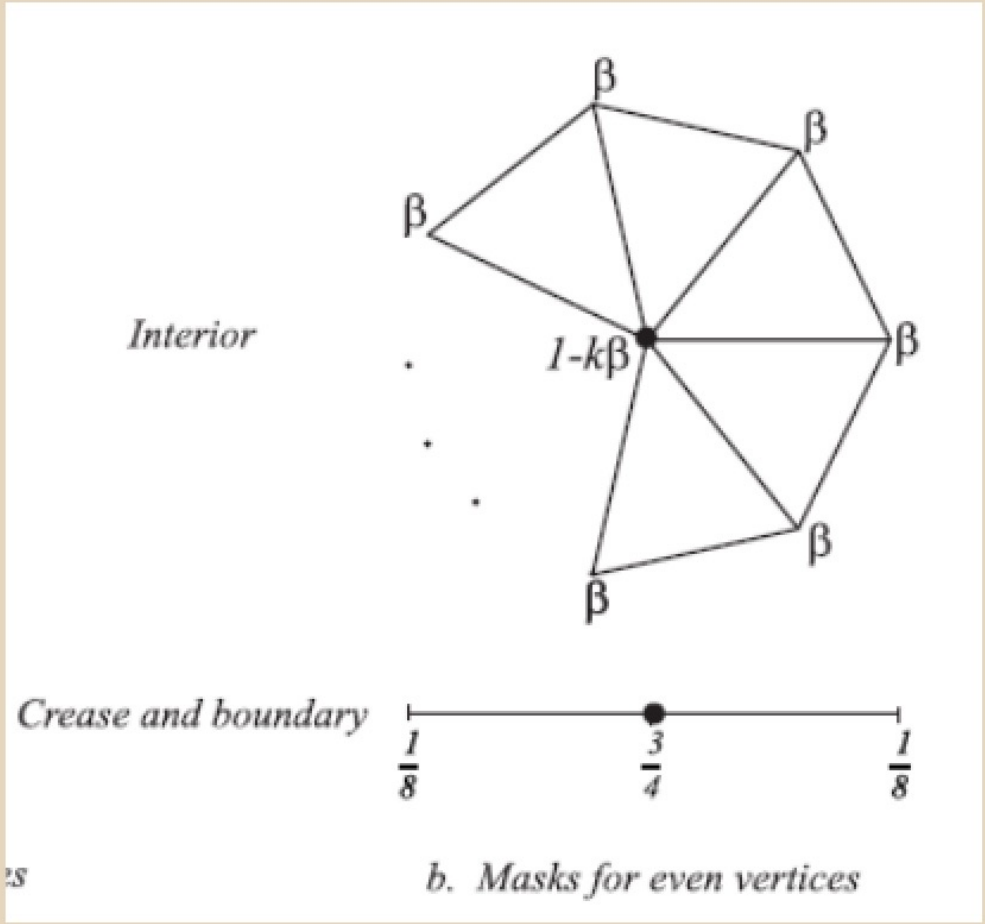
```
v = 3.0/8.0*(a + b)+
    1.0/8.0*(c + d)
```

Boundary:

```
v = 1.0/2.0*(a + b)
```

Notice that to compute v we need some to know the nearby vertices.

- Computing even vertices



Interior:

$$v = (1 - n\beta)v_0 + \beta \sum_{i=1}^{n} v_i$$

```
v = v*(1-k*BETA) +
(sum of all k neighbor
vertices)*BETA
```

$$\beta = \frac{1}{n}\left[\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right]$$

Boundary:

```
v = 1.0/8.0*(a + b) +
    3.0/4.0*(v)
```

Notice that to compute v we need know all neighboring vertices

TexParameter

{ x, y, z, w }	Useful when accessing vectors that represent points or normals
{ r, g, b, a }	Useful when accessing vectors that represent colors
{ s, t, p, q }	Useful when accessing vectors that represent texture coordinates

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

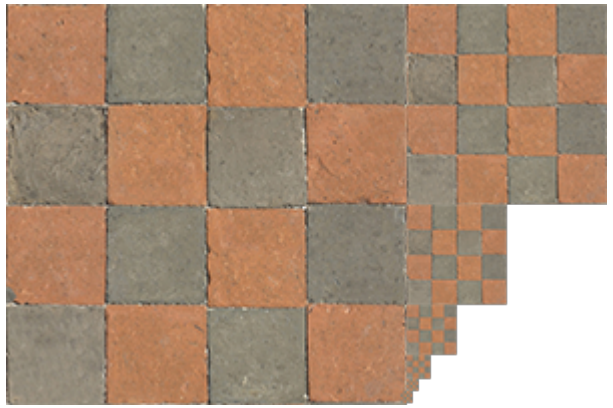
Line1-2: configure it for both the S and T axis.
Line3: minifying operations
Line4: magnifying operations

GL_NEAREST

GL_LINEAR

Mipmap

- `GL_NEAREST_MIPMAP_NEAREST`: takes the nearest mipmap to match the pixel size and uses nearest neighbor interpolation for texture sampling.
- `GL_LINEAR_MIPMAP_NEAREST`: takes the nearest mipmap level and samples that level using linear interpolation.
- `GL_NEAREST_MIPMAP_LINEAR`: linearly interpolates between the two mipmaps that most closely match the size of a pixel and samples the interpolated level via nearest neighbor interpolation.
- `GL_LINEAR_MIPMAP_LINEAR`: linearly interpolates between the two closest mipmaps and samples the interpolated level via linear interpolation.



Genereate texture

```
unsigned int texture;
glGenTextures(1, &texture);
glBindTexture(GL_TEXTURE_2D, texture);
// set the texture wrapping/filtering options (on the currently bound texture object)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
// load and generate the texture
int width, height, nrChannels;
unsigned char *data = stbi_load("container.jpg", &width, &height, &nrChannels, 0);
if (data)
{
    glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB, GL_UNSIGNED_BYTE, data);
    glGenerateMipmap(GL_TEXTURE_2D);
}
else
{
    std::cout << "Failed to load texture" << std::endl;
}
stbi_image_free(data);
```

GLSL texture function

- FragColor = texture(ourTexture, TexCoord);
- First Parameter: texture sampler
- Second, texture coordinates

- ourShader.setInt("texture2", 1);

Other application

- Normal mapping
- Shadow mapping