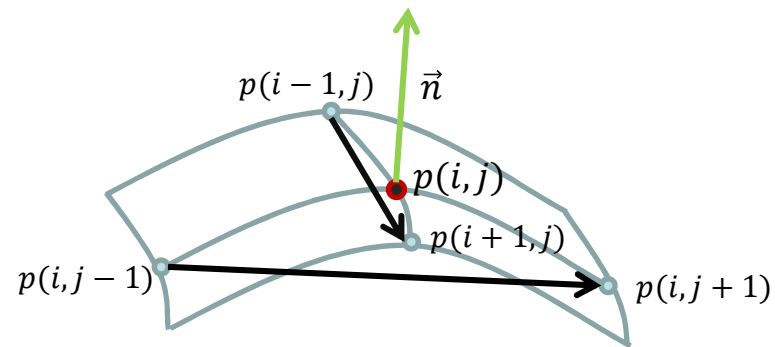# Terrain II

Adding light and texture to the terrain

Terrain Normals

- Secant approximation for partial derivatives

- Cross product of the partial derivatives provides an approximation to the surface normal
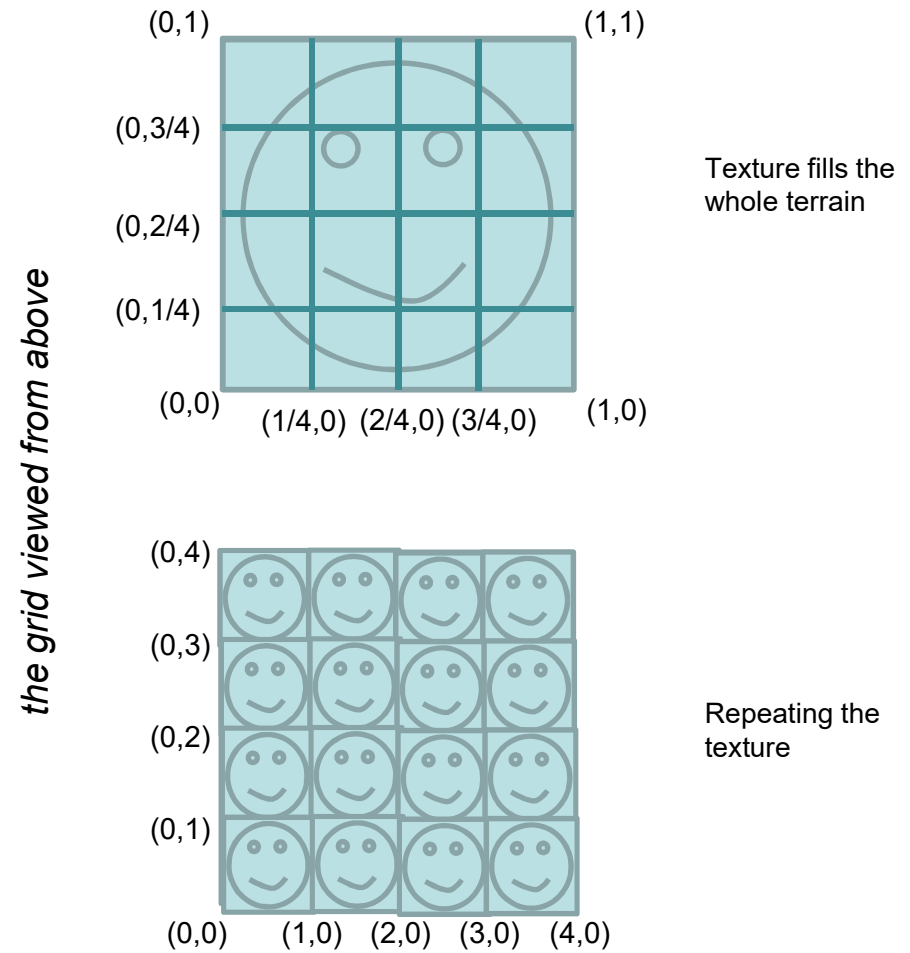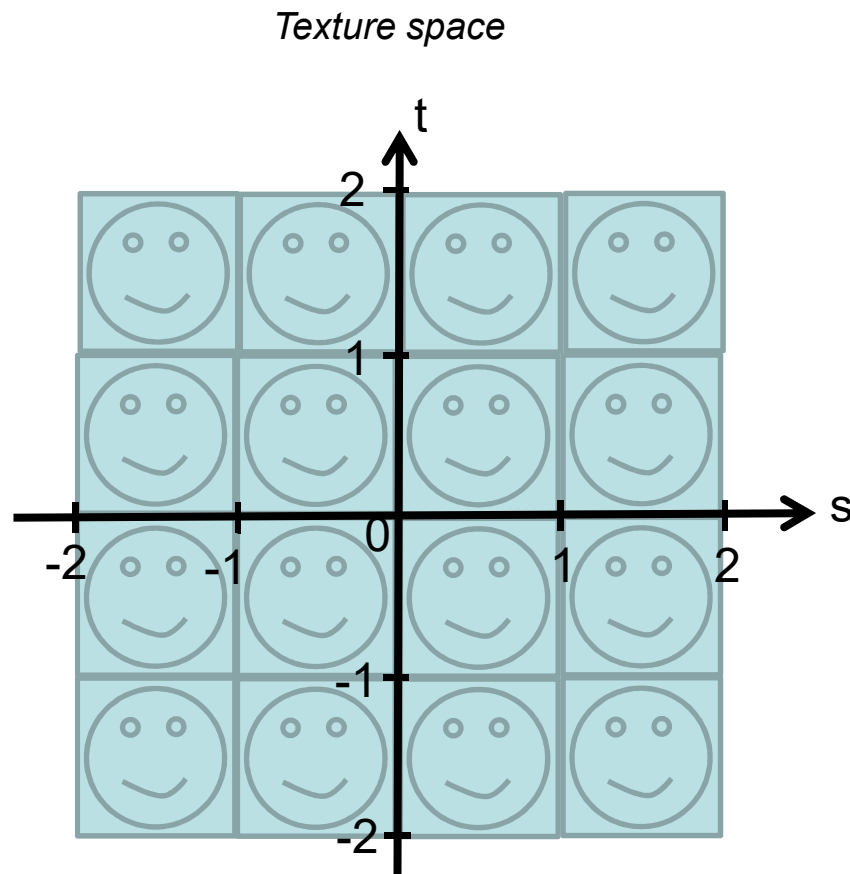

$$\overrightarrow{v_1} = p(i+1,j) - p(i-1,j)$$
$$\overrightarrow{v_2} = p(i,j+1) - p(i,j-1)$$

$$\vec{n} = \frac{\overrightarrow{v_1} \times \overrightarrow{v_2}}{|\overrightarrow{v_1} \times \overrightarrow{v_2}|}$$

Texture Coordinates

Texture space

Texture fills the whole terrain

the grid viewed from above

Repeating the texture

Texturing

- Ask OpenGL to generate mipmaps

  `glGenerateMipmap(GL_TEXTURE_2D)`

- Try the different filtering modes available for `GL_TEXTURE_MIN_FILTER`:

  `GL_NEAREST_MIPMAP_NEAREST`
  `GL_NEAREST_MIPMAP_LINEAR`
  `GL_LINEAR_MIPMAP_NEAREST`
  `GL_LINEAR_MIPMAP_LINEAR`

Assignment

- Define normals and texture coordinates for the terrain

- Compare the results with and without mipmapping

Loading a texture

```
unsigned int t,tw,th;
unsigned char *texData;
ilGenImages(1,&t);
ilBindImage(t);
ilLoadImage((ILstring)"relva1.jpg");
tw = ilGetInteger(IL_IMAGE_WIDTH);
th = ilGetInteger(IL_IMAGE_HEIGHT);
ilConvertImage(IL_RGBA, IL_UNSIGNED_BYTE);
texData = ilGetData();

glGenTextures(1,&texture);

glBindTexture(GL_TEXTURE_2D,texture);
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_MAG_FILTER,    GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,     GL_LINEAR);
//glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,     GL_LINEAR);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, tw, th, 0, GL_RGBA, GL_UNSIGNED_BYTE, texData);
```

GL init

```
glEnable(GL_TEXTURE_2D);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);

glEnableClientState(GL_VERTEX_ARRAY);
glEnableClientState(GL_NORMAL_ARRAY);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);
```

Prepare the terrain

```
void prepareTerrain() {

    for (int i = 1; i < imageWidth - 2 ; i++) {
        for(int j = 1 ; j < imageWidth -1; j++) {
            // fill arrays for position, normal and texcoord to create strips...
        }
    }

    glGenBuffers(3, buffers);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);
    glBufferData(GL_ARRAY_BUFFER, position.size() * sizeof(float), &(position[0]),GL_STATIC_DRAW);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);
    glBufferData(GL_ARRAY_BUFFER, normal.size() * sizeof(float), &(normal[0]),GL_STATIC_DRAW);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[2]);
    glBufferData(GL_ARRAY_BUFFER, texCoord.size() * sizeof(float), &(texCoord[0]),GL_STATIC_DRAW);

    glEnableClientState(GL_VERTEX_ARRAY);
    glEnableClientState(GL_NORMAL_ARRAY);
    glEnableClientState(GL_TEXTURE_COORD_ARRAY);
}
```

Render the terrain

```
void renderTerrain() {

    glColor3f(1, 1, 1);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[0]);
    glVertexPointer(3, GL_FLOAT, 0, 0);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[1]);
    glNormalPointer(GL_FLOAT, 0, 0);

    glBindBuffer(GL_ARRAY_BUFFER, buffers[2]);
    glTexCoordPointer(2, GL_FLOAT, 0, 0);

    for (int i = 1; i < imageWidth - 2; i++) {
        glDrawArrays(GL_TRIANGLE_STRIP, (imageWidth-2) * 2 * i, (imageWidth-2) * 2);
    }

}
```