ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εισαγωγή στα Γραφικά Υπολογιστών

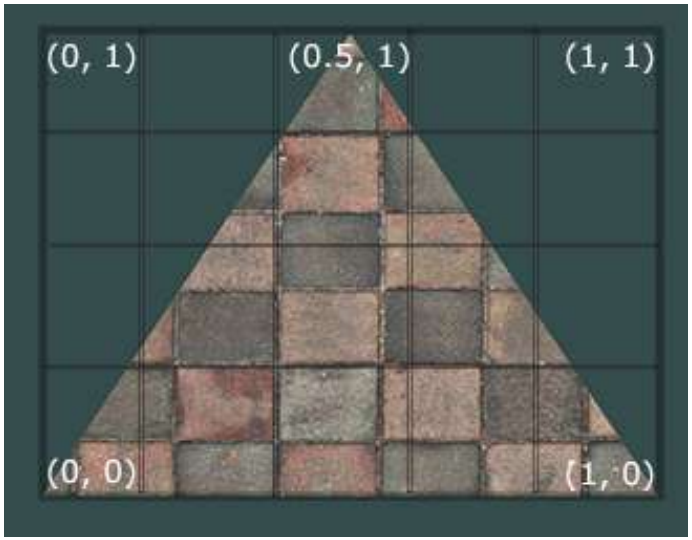
Εργαστήριο 6

Υφές

- 2Δ εικόνες με συντεταγμένες (s, t) στο [0, 1]

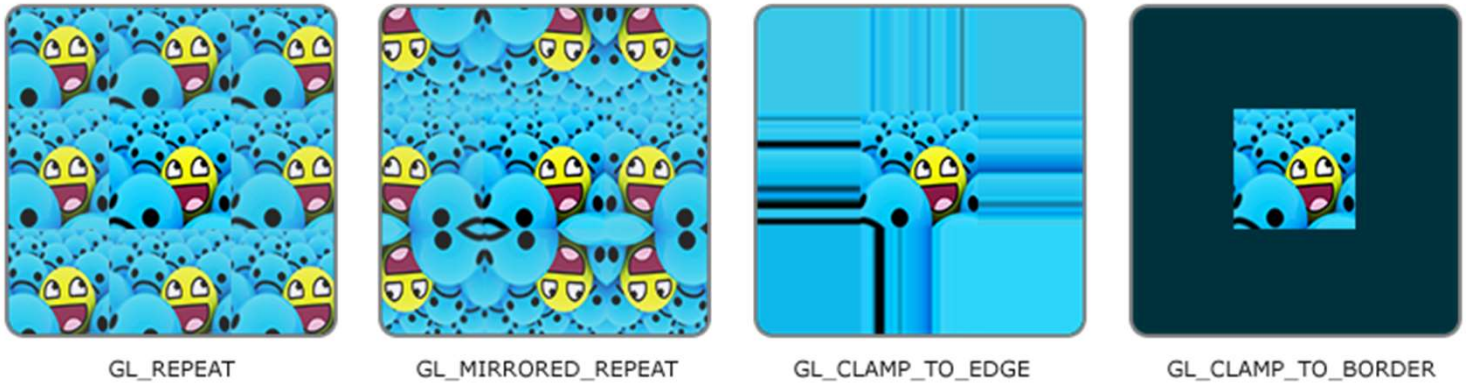
- Κάθε κορυφή αντιστοιχίζεται σε ένα ζεύγος συντεταγμένων της εικόνας → παίρνει το χρώμα που βρίσκεται σε αυτές τις συντεταγμένες

- Δειγματοληψία (**sampling**): η ανάκτηση του χρώματος της υφής από τις συντεταγμένες



Υφές

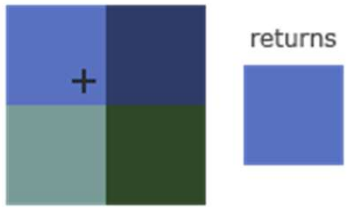
- Συντεταγμένες εκτός του εύρους [0, 1] ?
- **Texture Wrapping**:



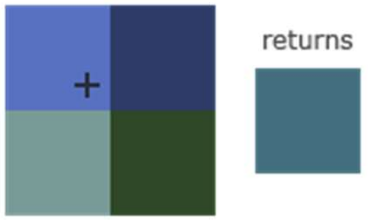

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_MIRRORED_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_MIRRORED_REPEAT);
```

Υφές

- Οι συντεταγμένες υφής είναι floats, αλλά οι 2Δ εικόνες έχουν ανάλυση → αντιστοίχιση: **Texture Filtering**
- Φιλτράρισμα κοντινότερου γείτονα:



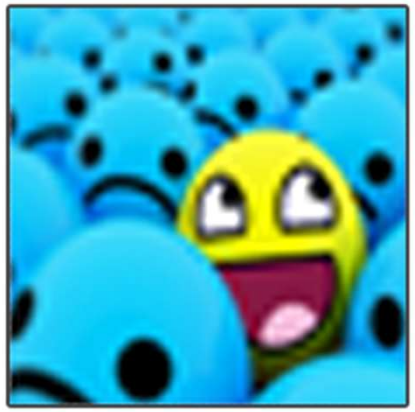
- Φιλτράρισμα γραμμικής παρεμβολής:



Υφές

- Το φιλτράρισμα των υφών μπορεί να οριστεί και για μεγέθυνση (**magnifying**) και για σμίκρυνση(**minifying**) των εικόνων.

GL_NEAREST



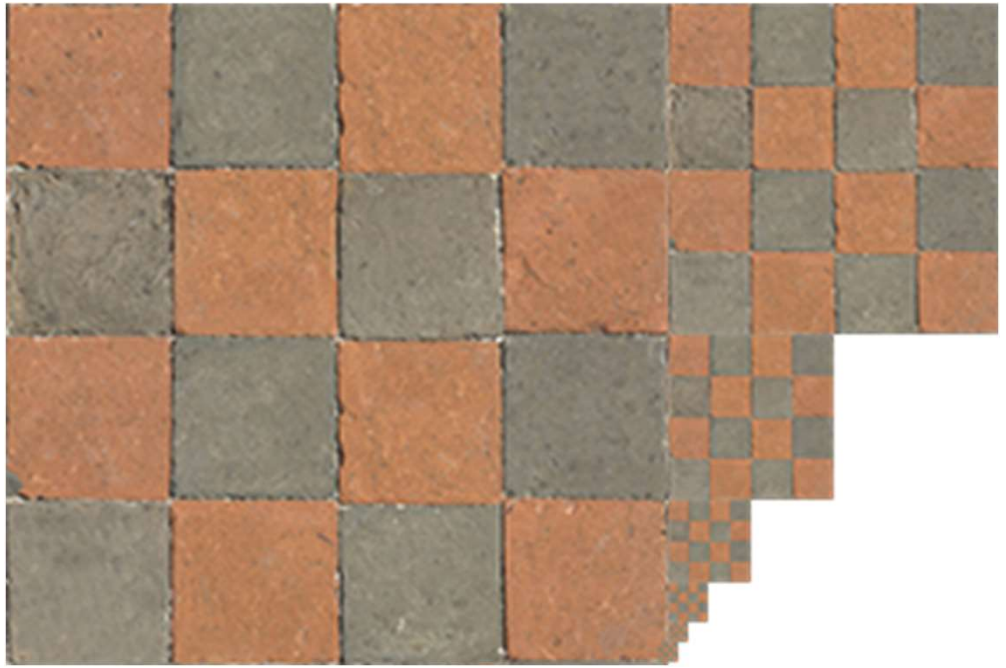
GL_LINEAR

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
```

Υφές

- **Mipmaps**

- Κάθε εικόνα έχει τις μισές διαστάσεις της προηγούμενης



Υφές

- Τα mipmaps χρησιμοποιούνται μόνο για σμίκρυνση της εικόνας:

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
GL_NEAREST_MIPMAP_LINEAR);
```

Το mipmap
δειγματοληπτείται
με κοντ. γείτονα

Το mipmap
επιλέγεται με
γραμμική παρεμβολή
από τα κοντινότερα mipmaps

Υφές

- Φόρτωση υφής στο πρόγραμμα: "stb_image.h" και "stb_image.cpp"

- **stbi_load()**:

int width, height, nrChannels;

unsigned char *data = stbi_load("Textures/container.jpg", &width, &height, &nrChannels, 0);

Υφές

```
unsigned int textureID; //generate texture ID
glGenTextures(1, &textureID);

if (data)
{
    glBindTexture(GL_TEXTURE_2D, textureID); //bind ID to the context

    //texture wrapping parameters
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

Υφές

```
    //texture filtering parameters
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_LINEAR);

glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST_MIPMAP_LINEAR);


   glTexParameteri(GL_TEXTURE_2D, GL_GENERATE_MIPMAP, GL_TRUE);
   //fill texture with image data
   glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, width, height, 0, GL_RGB,
GL_UNSIGNED_BYTE, data);
}
else
{ std::cout << "Failed to load texture" << std::endl; }

stbi_image_free(data);
```

Εισαγωγή u,v Συντεταγμένων

```
float vertices[] = {
    // positions          // normals           // texture coords
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  0.0f,  0.0f,
     0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  1.0f,  0.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  1.0f,  1.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  1.0f,  1.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  0.0f,  1.0f,
    -0.5f, -0.5f, -0.5f,  0.0f,  0.0f, -1.0f,  0.0f,  0.0f,

    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  0.0f,  0.0f,
     0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  1.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  1.0f,  1.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  1.0f,  1.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  0.0f,  1.0f,
    -0.5f, -0.5f,  0.5f,  0.0f,  0.0f,  1.0f,  0.0f,  0.0f,

    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f, -0.5f, -1.0f,  0.0f,  0.0f,  1.0f,  1.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,  0.0f,  1.0f,
    -0.5f, -0.5f, -0.5f, -1.0f,  0.0f,  0.0f,  0.0f,  1.0f,
    -0.5f, -0.5f,  0.5f, -1.0f,  0.0f,  0.0f,  0.0f,  0.0f,
    -0.5f,  0.5f,  0.5f, -1.0f,  0.0f,  0.0f,  1.0f,  0.0f,

     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,  1.0f,  0.0f,
     0.5f,  0.5f, -0.5f,  1.0f,  0.0f,  0.0f,  1.0f,  1.0f,
     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,  0.0f,  1.0f,
     0.5f, -0.5f, -0.5f,  1.0f,  0.0f,  0.0f,  0.0f,  1.0f,
     0.5f, -0.5f,  0.5f,  1.0f,  0.0f,  0.0f,  0.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  1.0f,  0.0f,  0.0f,  1.0f,  0.0f,

    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,  0.0f,  1.0f,
     0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,  1.0f,  1.0f,
     0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,  1.0f,  0.0f,
     0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,  1.0f,  0.0f,
    -0.5f, -0.5f,  0.5f,  0.0f, -1.0f,  0.0f,  0.0f,  0.0f,
    -0.5f, -0.5f, -0.5f,  0.0f, -1.0f,  0.0f,  0.0f,  1.0f,

    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,  0.0f,  1.0f,
     0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,  1.0f,  1.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,  1.0f,  0.0f,
     0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,  1.0f,  0.0f,
    -0.5f,  0.5f,  0.5f,  0.0f,  1.0f,  0.0f,  0.0f,  0.0f,
    -0.5f,  0.5f, -0.5f,  0.0f,  1.0f,  0.0f,  0.0f,  1.0f
};
```

Υφές

- Προσθήκη Vertex Shader:

```
layout (location = 2) in vec2 in_coordinate;
out vec2 texCoord;

void main()
{
	texCoord = in_coordinate;
}
```

Υφές

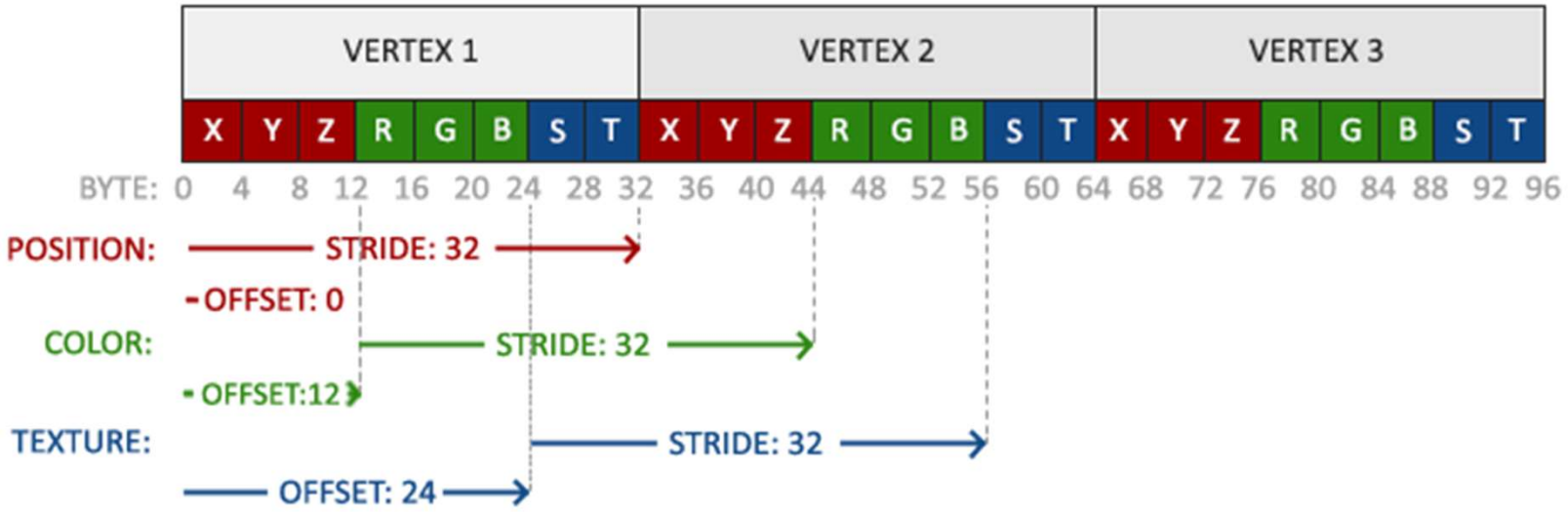
- Προσθήκη Fragment Shader:

```
uniform sampler2D myTexture;
in vec2 texCoord;

void main()
{
    gl_FragColor = texture2D(myTexture, texCoord);
}
```

Υφές

- Vertex attributes:



Υφές

```
//bind the texture and send it to the shaders
glBindTexture(GL_TEXTURE_2D, textureID);
int texLoc = glGetUniformLocation(ourShader.ID, "myTexture");
glUniform1i(texLoc, 0);
```

Ασκήσεις

- Συνδυασμός υφής και χρώματος
- Δημιουργία βοηθητικής συνάρτησης

```
int load_texture(const char* path)
```

Υφές

- Πολλές υφές στον fragment shader ?
- Κάθε υφή αποθηκεύεται σε ένα **Texture Unit**
- Τουλάχιστον 16 υφές ταυτόχρονα στον fragment shader.

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, textureID);
```

Υφές

```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, texture1);
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, texture2);

int texLoc = glGetUniformLocation(ourShader.ID, "texture1");
glUniform1i(texLoc, 0);

int texLoc2 = glGetUniformLocation(ourShader.ID, "texture2");
glUniform1i(texLoc2, 1);
```

Υφές

- Ανάμειξη δύο υφών στον fragment shader:

```
uniform sampler2D texture1;
uniform sampler2D texture2;
…

void main()
{
    gl_FragColor = mix(texture2D(texture1, texCoord), texture2D(texture2, texCoord), 0.2);
}
```

Υφές

Συντεταγμένες υφής OpenGL: 0.0 κάτω αριστερά
Συντεταγμένες εικόνας: 0.0 πάνω αριστερά

```
//before loading the image
stbi_set_flip_vertically_on_load(true);
```

- Κάποια αρχεία εικόνας περιέχουν περισσότερη πληροφορία:

JPEG αρχεία: GL_RGB
PNG αρχεία: GL_RGBA

Υφές

- Προσθήκη των εικόνων GL_RGBA στην συνάρτηση:

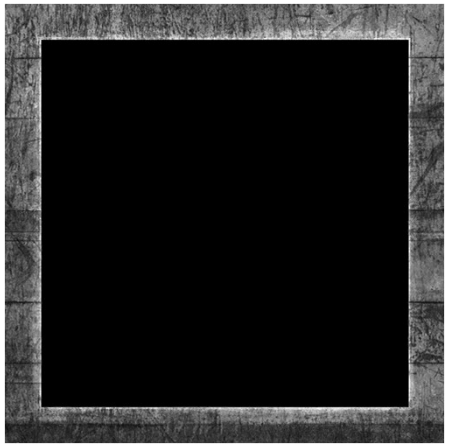
```
GLenum format;
        if (nrChannels == 1)
            format = GL_RED;
        else if (nrChannels == 3)
            format = GL_RGB;
        else if (nrChannels == 4)
            format = GL_RGBA;

glTexImage2D(GL_TEXTURE_2D, 0, format, width, height, 0, format,
GL_UNSIGNED_BYTE, data);
```

Χάρτες φωτισμού

- Χάρη στις υφές μπορούμε να σχεδιάσουμε αντικείμενα που αποτελούνται από πολλά υλικά, με πολλές λεπτομέρειες.

- Χάρτες φωτισμού (**Lighting maps**): **diffuse maps** και **specular maps**



Χάρτες φωτισμού

- Fragment shader:

```
struct Material {
        sampler2D diffuseMap;
        sampler2D specularMap;
        float shininess;
};
```

Χάρτες φωτισμού

- Αλλαγές στη main():

material.ambient → vec3(texture2D(material.diffuseMap, texCoord))

material.diffuse → vec3(texture2D(material.diffuseMap, texCoord))

material.specular → vec3(texture2D(material.specularMap, texCoord))

(Η συνάρτηση texture2D επιστρέφει ένα vec4 με τα κανάλια rgba γι' αυτό πρέπει να το κάνουμε vec3 ή στο τέλος να βάλουμε .rgb δηλαδή texture2D(…).rgb).

Χάρτες φωτισμού