Informatyka praktyczna. Import modeli 3D

Aleksander Denisiuk
Uniwersytet Warmińsko-Mazurski
Olsztyn, ul. Słoneczna 54
aleksander.denisiuk@uwm.edu.pl

Import modeli 3D

Wprowadzenie
Import

Najnowsza wersja tego dokumentu dostępna jest pod adresem

http://wmii.uwm.edu.pl/~denisjuk/uwm

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

Wprowadzenie

Przykład modelu zaimportowanego

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import



- Blender $\longrightarrow$ Wavefront .obj $\longrightarrow$ OpenGL

Eksport modeli z Blendera

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- rozłożenie modelu na płaszczyźnie (UV mapping)
- *Wypiekanie* tekstury
- eksport w obj

UV-mapping

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- Dobrze się sprawdza **Smart UV Project** (zaznaczyłem **Scale to Bounds**)
  - dla latarni warto dodać modyfikator **Solidify**

Wypiekanie tekstury

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- Stworzyć nowy obraz (bez przezroczystości)
- Dla każdego materiału obiektu dodać niepodłączony nod **Texture Image** z tym obrazem
- Silnik **Cycles**
- Zakładka **Render Properties** $\rightarrow$ **Bake**
  - ✦ wypiekamy tylko **Diffuse**, bez oświetlenia
- Zapisać teksturę jako **Targa Raw**

Przykład tekstury

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import



Eksport w obj

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- menu **File**, **Export**, **Wavefront (.obj)**
- opcje **Normals**, **Triangulated Mesh**, **Selection Only**, **UV Coordinates**, **Apply Modifiers**, **Materials**
- **Apply Transform**:
  - zaznaczyć
  - nie zaznaczać, zmodyfikować `model_martix_`

Format Wavefront .obj

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- Wavefront Technologies
- format jest legalny, przyjęty przez wszystkich wytwórców 3W aplikucji
- dwa pliki

- `*.obj` — informacja o wierzchołkach
- `*.mtl` — informacja o materiałach

Przykładowy plik `.mtl`

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
# Blender MTL File: 'kostka.blend'
# Material Count: 3
newmtl Material.001_untitled
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.640000 0.075960 0.174290
Ks 0.500000 0.500000 0.500000
Ni 1.000000
d 1.000000
illum 2
map_Kd tekstura.png

.....................................
```

Drugi materiał

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
newmtl Material.002_untitled
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.640000 0.640000 0.640000
Ks 0.000000 0.000000 0.000000
Ni 1.000000
d 1.000000
illum 1
map_Kd tekstura.png

.....................................
```

Trzeci materiał

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
newmtl Material_untitled
Ns 96.078431
Ka 0.000000 0.000000 0.000000
Kd 0.640000 0.640000 0.640000
Ks 0.500000 0.500000 0.500000
Ni 1.000000
d 1.000000
illum 2
map_Kd tekstura.png
```

Przykładowy plik `.obj`

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
# Blender v2.63 (sub 0) OBJ File: 'kostka.blend'
# www.blender.org
mtllib kostka.mtl
o Cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -0.999999
v 0.999999 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000

.....................................
```

Współrzędne tekstury

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
vt 0.249999 0.250000
vt 0.250000 0.500000
vt 0.000000 0.500000
vt 0.000000 0.250000
vt 0.499999 0.250000
vt 0.749999 0.250000
vt 0.750000 0.500000
vt 0.499999 0.500000
vt 1.000000 0.500000
vt 1.000000 0.249999
vt 0.249999 0.000000
vt 0.499999 0.000000
vt 0.499999 0.749999
vt 0.249999 0.749999

.....................................
```

Wektory normalne

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
vn 0.000000 -1.000000 0.000000
vn 0.000000 1.000000 0.000000
vn -1.000000 -0.000000 -0.000000
vn 0.000000 0.000000 -1.000000
vn 1.000000 0.000000 0.000000
vn -0.000000 -0.000000 1.000000

.....................................
```

Grupa wierzchołków

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
usemtl Material_untitled
s off
f 1/1/1 2/2/1 3/3/1
f 1/1/1 3/3/1 4/4/1
f 5/5/2 8/6/2 7/7/2
f 5/5/2 7/7/2 6/8/2
f 3/9/3 7/7/3 8/6/3
f 3/9/3 8/6/3 4/10/3
f 5/5/4 1/1/4 4/11/4
f 5/5/4 4/11/4 8/12/4

.....................................
```

Druga i trzecia grupa wierzchołków

Wprowadzenie
- Przykład
- Eksport
- Format .obj

Import

```
.....................................
usemtl Material.001_untitled
f 1/1/5 5/5/5 6/8/5
f 1/1/5 6/8/5 2/2/5
usemtl Material.002_untitled
f 2/2/6 6/8/6 7/13/6
f 2/2/6 7/13/6 3/14/6

.....................................
```

Importujemy właściwości

Wprowadzenie
❖ Przykład
❖ Eksport
❖ Format .obj
Import

- `newmtl` —nazwę materiału
- `Ka` — trójwymiarowy współczynnik odbicia światła naturalnego (ambient), dodajemy czwartą współrzędną $\alpha = 1$
- `Kd` — trójwymiarowy współczynnik odbicia rozproszonego (diffuse), dodajemy czwartą współrzędną $\alpha = 1$
- `Ks` — trójwymiarowy współczynnik odbicia zwierciadlanego , dodajemy czwartą współrzędną $\alpha = 1$
- `Ns` — wskaźnik odbicia zwierciadlanego

Wprowadzenie

Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

Import

Struktury pomocnicze

Wprowadzenie

Import

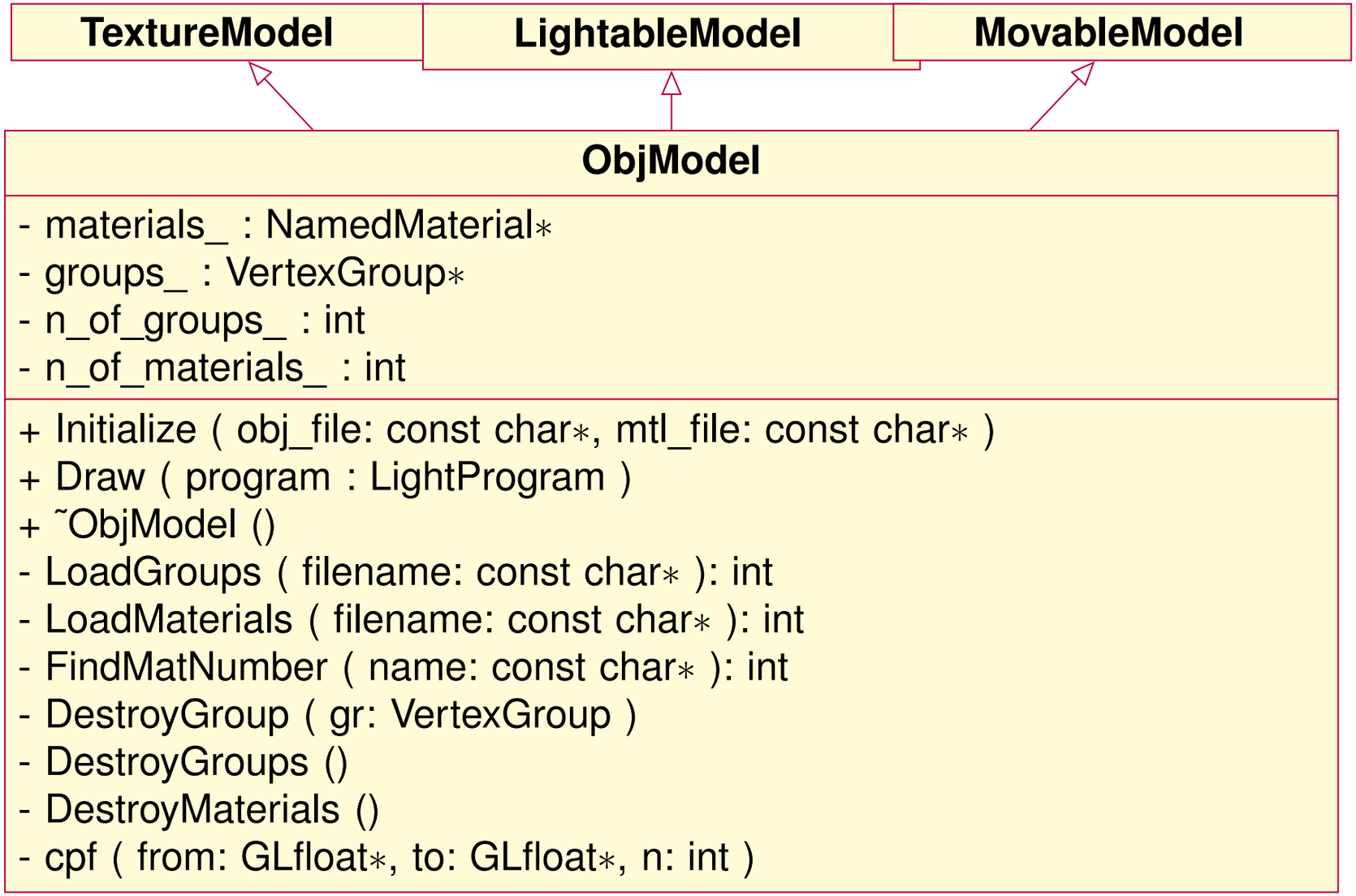
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
typedef struct NamedMaterial{
  char name_[30];
  Material m;
} NamedMaterial;

typedef struct VertexGroup{
  GLuint vao;
  GLuint vertices;
  int mat_number;
  int n_of_vertices;
} VertexGroup;

typedef GLfloat Vec2[2];
typedef GLfloat Vec3[3];
```

ObjModel

Wprowadzenie

Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik



Klasa ObjModel

Wprowadzenie
Import
❖ Struktury pomocnicze
❖ ObjModel
❖ ::Initialize
❖ ::LoadMaterials
❖ ::LoadGroups
❖ ::Destruktor
❖ ::Draw
❖ Cube
❖ Wynik

```
class ObjModel{
public:
    void Initialize(const char *obj_file,
      const char *mtl_file);
    void Draw(const PointLightProgram &);
    ~ObjModel();
```

Dane prywatne

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
private:
    NamedMaterial * materials_;
    VertexGroup * groups_;
    int n_of_groups_;
    int n_of_materials_;

    int LoadGroups(const char * filename);
    int LoadMaterials(const char* filename);
    int FindMatNumber(const char* MatName);
    void DestroyGroup(VertexGroup gr);
    void DestroyGroups();
    void DestroyMaterials(void);
```

Initialize

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
void ObjModel::Initialize(
                const char *obj_file,
                const char *mtl_file )
{
  n_of_materials_ = LoadMaterials(mtl_file);
  n_of_groups_ = LoadGroups(obj_file);
}
```

LoadMaterials

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik


```
int ObjModel::LoadMaterials(const char* filename)
{
  int n_of=-1;
  int maxMat=2;
  int d_mat=2;
  char mode[10];
  NamedMaterial * new_materials;

  ifstream file (filename, ios::in|ios::binary);

  if (file.is_open()) {
    materials_ = (NamedMaterial*)
          malloc(maxMat*sizeof(NamedMaterial));

   while(file>>mode){
```

Nowy materiał. Alokacja nowej pamięci

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
if(strcmp("newmtl",(const char*)mode)==0){
 n_of++;
 if(n_of>=maxMat){
  maxMat+=d_mat;
  new_materials
    =(NamedMaterial*)realloc(
    materials_, maxMat*sizeof(NamedMaterial) );
  if(NULL!=new_materials) materials_=new_materials;
  else {
    glfwTerminate();
    exit(EXIT_FAILURE);
  }
 }
```

Nowy materiał. Nazwa i światło emitowane

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
    file>>materials_[n_of].name_;
    materials_[n_of].m.emission[0]=0.0f;
    materials_[n_of].m.emission[1]=0.0f;
    materials_[n_of].m.emission[2]=0.0f;
    materials_[n_of].m.emission[3]=1.0f;
}
```

Odbicie światła rozproszonego

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("Kd",(const char*)mode)==0){
    // diffuse
    file >> materials_[n_of].m.diffuse[0]
         >> materials_[n_of].m.diffuse[1]
         >> materials_[n_of].m.diffuse[2];
    materials_[n_of].m.diffuse[3]=1.0f;
}
```

Odbicie światła zwierciadlanego

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("Ks",(const char*)mode)==0){
    // specular
    file >> materials_[n_of].m.specular[0]
         >> materials_[n_of].m.specular[1]
         >> materials_[n_of].m.specular[2];
    materials_[n_of].m.specular[3]=1.0f;
}
```

Wskaźnik odbicia zwierciadlanego

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("Ns",(const char*)mode)==0){
    // shininess
    file>> materials_[n_of].m.shininess;
}
```

Odbicie światła naturalnego

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("Ka",(const char*)mode)==0){
    // ambient
    file >> materials_[n_of].m.ambient[0]
         >> materials_[n_of].m.ambient[1]
         >> materials_[n_of].m.ambient[2];
   materials_[n_of].m.ambient[3]=1.0f;
}
```

Końcówka

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
            else file.ignore(1024,'\n');
        }
        file.close();
    }
    else{
        cerr<<"ERROR: " << filename<<endl;
        glfwTerminate();
        exit(EXIT_FAILURE);
     }
    return n_of+1;
}
```

LoadGroups: zmienne pomocnicze

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik


```
int ObjModel::LoadGroups(const char * filename){
  int n_of=-1; // Groups
  int max=2;
  int delta=2;
  int mat_number=-1;
  int n_of_vert=-1; //vertices
  int max_vert=2;
  int d_vert=2;
  int n_of_vert_in_gr=-1; // vertices in group
  int max_v_in_g=6;
  int d_v_in_g=6; // >=3
  int n_of_tex=-1; // Texture
  int max_tex=2;
  int d_tex=2;
  int n_of_norm=-1; // normals
  int max_norm=2;
  int d_norm=2;
```

Jeszcze zmienne pomocnicze

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
VertexGroup * new_groups;
Vec3 * vertices;
NormalTextureVertex * vertices_in_gr;
Vec2 * tex_coords;
Vec3 * normals;
Vec3 * new_vertices;
NormalTextureVertex * new_vertices_in_gr;
Vec2 * new_tex_coords;
Vec3 * new_normals;

int v,t,n; //vertex data
```

Otwieramy plik...

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
ifstream file (filename, ios::in|ios::binary);

if (file.is_open()) {
    groups_ = (VertexGroup*)
       malloc(max*sizeof(VertexGroup));
    vertices = (Vec3*) malloc(max_vert*sizeof(Vec3));
    vertices_in_gr = (NormalTextureVertex*)
       malloc(max_v_in_g*sizeof(NormalTextureVertex));
    tex_coords = (Vec2*) malloc(max_tex*sizeof(Vec2));
    normals = (Vec3*) malloc(max_norm*sizeof(Vec3));

    while (file>>mode) {
```

Nowy wierzchołek

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
if(strcmp("v",(const char*)mode)==0){
    //Vertex
    n_of_vert++;
    if(n_of_vert>=max_vert){
        max_vert+=d_vert;
        new_vertices = (Vec3*)
            realloc(vertices,max_vert*sizeof(Vec3));
        if(NULL!=new_vertices) vertices=new_vertices;
        else {
            cerr<< "ERROR: ...\n";
            glfwTerminate();
            exit(EXIT_FAILURE);
        }
    }
    file >> vertices[n_of_vert][0]
         >> vertices[n_of_vert][1]
         >> vertices[n_of_vert][2];
}
```

Nowy wektor normalny

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("vn",(const char*)mode)==0){
    //Normal
    n_of_norm++;
    if(n_of_norm>=max_norm){
        max_norm+=d_norm;
        new_normals =  (Vec3*)
          realloc(normals,max_norm*sizeof(Vec3));
        if(NULL!=new_normals) normals=new_normals;
        else {
            cerr<< "ERROR: ...";
            glfwTerminate();
            exit(EXIT_FAILURE);
        }
    }
    file >> normals[n_of_norm][0]
         >> normals[n_of_norm][1]
         >> normals[n_of_norm][2];
}
```

Nowa para współrzędnych teksturowych

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("vt",(const char*)mode)==0){
    //Texture
    n_of_tex++;
    if(n_of_tex>=max_tex){
        max_tex+=d_tex;
        new_tex_coords =  (Vec2*)
          realloc(tex_coords,max_tex*sizeof(Vec2));
        if (NULL!=new_tex_coords)
                          tex_coords=new_tex_coords;
        else {
            cerr<< "ERROR: ...\n";
            glfwTerminate();
            exit(EXIT_FAILURE);
        }
    }
    file >> tex_coords[n_of_tex][0]
         >> tex_coords[n_of_tex][1];
}
```

Ściana. Realokacja pamięci

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
else if(strcmp("f",(const char*)mode)==0){
    //Face
    if(n_of_vert_in_gr+3>=max_v_in_g){
        max_v_in_g+=d_v_in_g;
        new_vertices_in_gr = (NormalTextureVertex*)
          realloc(vertices_in_gr,
              max_v_in_g*sizeof(NormalTextureVertex));
        if(NULL!=new_vertices_in_gr)
                  vertices_in_gr=new_vertices_in_gr;
        else {
            cerr << "ERROR: ...\n";
            glfwTerminate();
            exit(EXIT_FAILURE);
        }
    }
```

Ściana. Dane

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
    for(i=0;i<3;i++){
        // three vertices
        file >> v >> g >> t>> g >>n;
        n_of_vert_in_gr++;
        cpf(vertices[v-1],
           vertices_in_gr[n_of_vert_in_gr].position, 4);
        cpf(tex_coords[t-1],
           vertices_in_gr[n_of_vert_in_gr].texture, 2);
        cpf(normals[n-1],
            vertices_in_gr[n_of_vert_in_gr].normal, 3);
    }
}
```

- `void cpf(GLfloat* from, GLfloat* to, int n)` — pomocnicza funkcja do kopiowania danych z uzupełnieniem

Pomocnicza funkcja do kopiowania danych

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik


```
void ObjModel::cpf(GLfloat* from,
    GLfloat* to, int n){
        if(n==4){
            to[3]=1.0f;
            n--;
        }
        while(n-->0){
            to[n]=from[n];
        }
        return;
}
```

Nowa grupa wierzchołków.

Wprowadzenie
Import
❖ Struktury pomocnicze
❖ ObjModel
❖ ::Initialize
❖ ::LoadMaterials
❖ ::LoadGroups
❖ ::Destruktor
❖ ::Draw
❖ Cube
❖ Wynik

- Jeżeli grupa już została wczytana, jej dane wysyłamy na GPU
  - VAO

```
else if(strcmp("usemtl",(const char*)mode)==0){
//New group
if(n_of>=0){

    //send current group to GPU
    glGenVertexArrays(1, &groups_[n_of].vao);
    glBindVertexArray(groups_[n_of].vao);
```

Wysyłanie poprzedniej grupy na GPU. VBO

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
glGenBuffers(1, &groups_[n_of].vertices);
glBindBuffer(GL_ARRAY_BUFFER,
 groups_[n_of].vertices);
glBufferData(GL_ARRAY_BUFFER,
 (n_of_vert_in_gr+1)*sizeof(NormalTextureVertex),
 vertices_in_gr, GL_STATIC_DRAW);
```

Argumenty dla shadera

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
glVertexAttribPointer(0, 4, GL_FLOAT, GL_FALSE,
   sizeof(vertices_in_gr[0]), (GLvoid*)0);
glVertexAttribPointer(2, 2, GL_FLOAT, GL_FALSE,
   sizeof(vertices_in_gr[0]),
   (GLvoid*)sizeof(vertices_in_gr[0].position));
glVertexAttribPointer(3, 3, GL_FLOAT, GL_FALSE,
   sizeof(vertices_in_gr[0]),
   (GLvoid*)(sizeof(vertices_in_gr[0].position)
      + sizeof(vertices_in_gr[0].texture)));
glEnableVertexAttribArray(0);
glEnableVertexAttribArray(2);
glEnableVertexAttribArray(3);

glBindVertexArray(0);

groups_[n_of].n_of_vertices=n_of_vert_in_gr+1;
```

Ilość grup, realokacja

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
} //
n_of++;
if(n_of>=max){
    max+=delta;
    new_groups= (VertexGroup*)
       realloc(groups_,max*sizeof(VertexGroup));
    if(NULL!=new_groups) groups_=new_groups;
    else {
       cerr<< "ERROR: ...\n";
       glfwTerminate();
       exit(EXIT_FAILURE);
    }
}
```

Materiał nowej grupy

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
file >> mat_name;
mat_number=FindMatNumber(mat_name);
if(mat_number<0){
    cerr<< "ERROR: Could not find material\n";
    glfwTerminate();
    exit(EXIT_FAILURE);
}
groups_[n_of].mat_number=mat_number;
n_of_vert_in_gr=-1;
```

Końcówka

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
}
else file.ignore(1024,'\n') ;

}// While
file.close();
if(n_of>=0){
   //send current group to GPU
    }

free(vertices);
free(vertices_in_gr);
free(tex_coords);
free(normals);
file.close();
```

Destruktor

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
ObjModel::~ObjModel(){
    DestroyGroups();
    DestroyMaterials();
}
```

DestroyGroups

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
void ObjModel::DestroyGroups(void){
    glDisableVertexAttribArray(1);
    glDisableVertexAttribArray(0);
    glBindBuffer(GL_ARRAY_BUFFER, 0);
    glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, 0);
  int i;
  for (i=0;i< n_of_groups_; i++){
    DestroyGroup(groups_[i]);
  }
  free(groups_);
  n_of_groups_=0;
}
```

DestroyGroup

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
void ObjModel::DestroyGroup(VertexGroup gr){
    glDeleteBuffers(1, &gr.vao);
    glDeleteBuffers(1, &gr.vertices);
}
```

DestroyMaterials

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
void ObjModel::DestroyMaterials(void)
{
  free(materials_);
  n_of_materials_=0;
}
```

Draw

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
void ObjModel::Draw(const LightProgram &prog){
  glUseProgram(prog);
  prog.SetModelMatrix(model_matrix_);
  prog.SetNormalMatrix(normal_matrix_);

  glEnable(GL_CULL_FACE);
  glCullFace(GL_BACK);
  glFrontFace(GL_CCW);

  glActiveTexture(texture_unit_);
  glBindTexture(GL_TEXTURE_2D, texture_);
```

Draw. Pętla

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

```
  for (int i=0; i<n_of_groups_; i++){
      glBindVertexArray(groups_[i].vao);
      prog.SetMaterial(
         materials_[groups_[i].mat_number].m);
      glDrawArrays(GL_TRIANGLES, 0,
         groups_[i].n_of_vertices);
      glBindVertexArray(0);
  }
  glDisable(GL_CULL_FACE);
  glUseProgram(0);
}
```

Draw. Pętla

Wprowadzenie
Import
- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

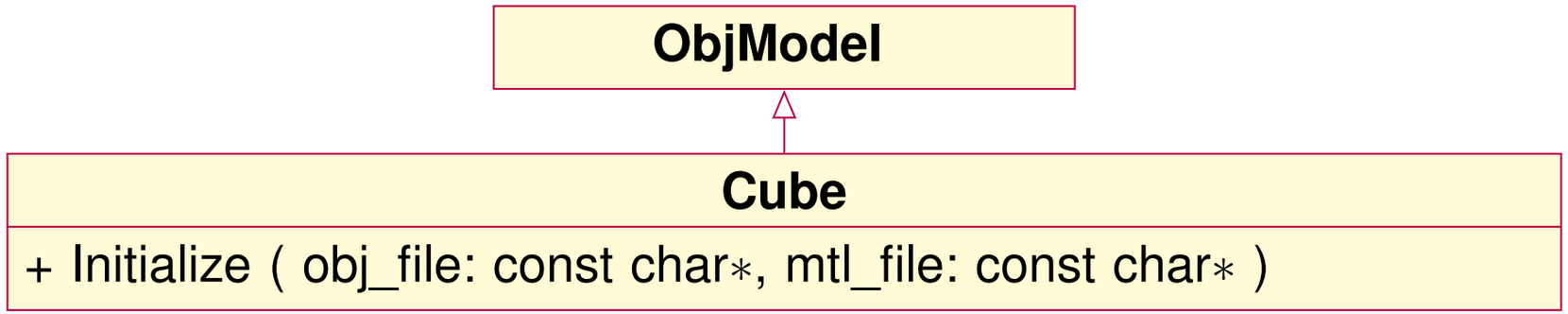
```
  for (int i=0; i<n_of_groups_; i++){
      glBindVertexArray(groups_[i].vao);
      prog.SetMaterial(
         materials_[groups_[i].mat_number].m);
      glDrawArrays(GL_TRIANGLES, 0,
         groups_[i].n_of_vertices);
      glBindVertexArray(0);
  }
  glDisable(GL_CULL_FACE);
  glUseProgram(0);
}
```

Cube

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik




```
void Cube::Initialize(const char *obj_file,
                      const char *mtl_file){
  model_matrix_.RotateAboutX(-45);
  model_matrix_.RotateAboutY(45);

  normal_matrix_.RotateAboutY(-45);
  normal_matrix_.RotateAboutX(45);

  ObjModel::Initialize(obj_file, mtl_file);
}
```

Wynik

Wprowadzenie

Import

- Struktury pomocnicze
- ObjModel
- ::Initialize
- ::LoadMaterials
- ::LoadGroups
- ::Destruktor
- ::Draw
- Cube
- Wynik

