Laborator 12

1 Obiective

Acest laborator vă prezintă noțiuni de bază despre conceptele de efect de ceață, combinarea culorilor și eliminarea fragmetelor.

2 Ceață

Putem să creștem realismul unei scene 3D prin adăugarea unui efect de ceață care va crea o atmosferă specială. Prin manipularea atributelor asociate efectului de ceață putem să personalizăm atmosfera și, de asemenea, să ameliorăm percepția de adâncime (axa Z).

Culoarea de fundal trebuie aleasă în funcție de culoarea efectului de ceață.

```
glClearColor(0.5, 0.5, 0.5, 1.0);
```

În versiunile de OpenGL cu pipeline fix existau trei metode de calculare a ceții (liniară, exponențială și exponențială pătratică). Aceeași funcționalitate o putem realiza și în versiunea cu pipeline programabil prin adăugarea calculelor de ceață în shader-ul nostru de fragmente.

2.1 Ceață liniară

Prima metodă de calculare a ceții este prin folosirea unei funcții de interpolare liniară care va amesteca culoarea ceții cu aceea a fragmentului, pe baza distanței fragmentului (calculată în spațiul de vizualizare).

$$fogFactor = \frac{fogEnd - fragmentDistance}{fogEnd - fogStart}$$

Valorile factorului de ceață trebuie să fie între 0.0 and 1.0.

2.2 Ceață exponențială

Un rezultat mai bun poate fi obținut luând în considerare reducerea intensității luminii în funcție de distanță. Factorul de atenuare care va fi folosit reprezintă densitatea de ceață, această densitate fiind constantă în toată scena. Rezultatul este o scădere rapidă a factorului de ceață în comparație cu abordarea liniară.

$$fogFactor = e^{-fragmentDistance * fogDensity}$$

2.3 Ceață exponențială pătratică

Formula este practic aceeași cu cea precedentă, singura diferență provinind din faptul că am ridicat la pătrat distanța fragmentului și densitatea de ceață.

$$fogFactor = e^{-(fragmentDistance * fogDensity)^2}$$

2.4 Tutorial

Adăugați următoarea funcție în interiorul fragment shader-ului:

```
float computeFog()
{
    float fogDensity = 0.05f;
    float fragmentDistance = length(fragmentPosEyeSpace);
    float fogFactor = exp(-pow(fragmentDistance * fogDensity, 2));

    return clamp(fogFactor, 0.0f, 1.0f);
}
```

Funcția **clamp** constrânge o valoare să se afle între două limite (minValue și maxValue).

```
float fogFactor = computeFog();
vec4 fogColor = vec4(0.5f, 0.5f, 0.5f, 1.0f);
fColor = mix(fogColor, color, fogFactor);
```

Funcția **mix** interpolează liniar între două valori. Culoarea rezultată poate fi calculată și prin:

```
fColor = fogColor * (1 – fogFactor) + color * fogFactor;
```

Variabila **color** reprezintă culoarea calculată folosind sursele de lumină specificate în scenă.

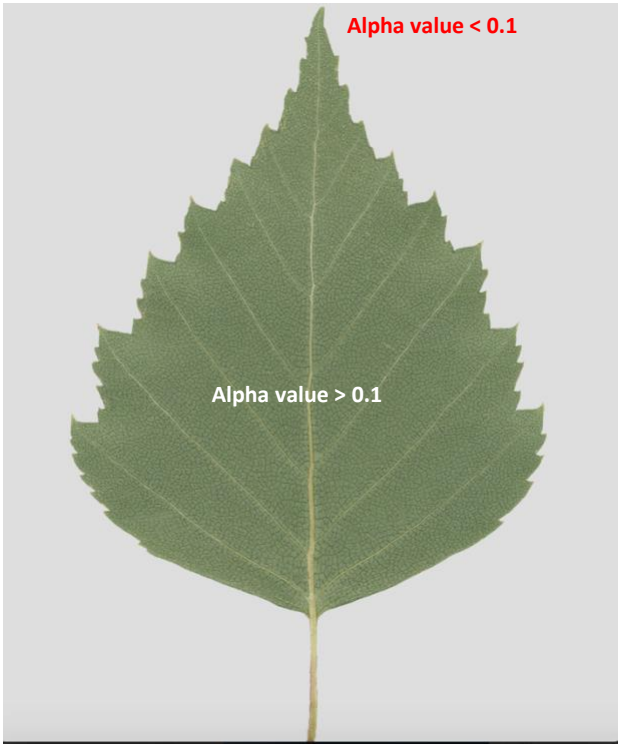
3 Eliminarea fragmentelor

Valoarea alfa din texturi poate fi utilizată pentru a elimina un fragment. De exemplu, doriți să afișați o frunză, iar modelul 3D este reprezentat de un quad pe care este mapată o textură a frunzei. Puteți verifica valoarea canalului alfa urmând să eliminați fragmentul dacă valoarea este mai mică de 0,1.

```
vec4 colorFromTexture = texture(diffuseTexture, interpolatedTexCoords);
if(colorFromTexture.a < 0.1)
        discard;
fColor = colorFromTexture;
```

Actualizați apelul funcției glTexImage2D în interiorul metodei ReadTextureFromFile (din clasa Model3D):

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, x, y, 0, GL_RGBA, GL_UNSIGNED_BYTE, image_data);
```



4 Combinarea culorilor

Putem face ca obiectele să fie transparente (sau semi-transparente) prin combinarea culorii obiectului și a culorii actuale din framebuffer (reprezentând obiectele deja redate) într-o singură culoare. Cantitatea de transparență este definită de canalul alfa din reprezentarea culorii. Dacă a patra componentă a culorii este 1.0f, atunci obiectul este opac, dacă valoarea este 0.0f atunci obiectul este transparent. Valorile din intervalul (0.0f, 1.0f) reprezintă o cantitate diferită de transparență/opacitate.

Combinarea culorilor este activată în OpenGL utilizând următoarea funcție:

```
glEnable(GL_BLEND);
```

Culoarea rezultată după aplicarea acestei operații se calculează folosind următoarea ecuație:

$color = sourceColor * sourceFactor + destinationColor * destinationFactor$ unde,

- *sourceColor* – reprezintă culoarea obiectului care este rasterizat în prezent
- *sourceFactor* – definește influența culorii sursă
- *destinationColor* – reprezintă culoarea din framebuffer
- *destinationFactor* – definește influența culorii destinație

SourceFactor și destinationFactor sunt specificați folosind glBlendFunction. În general, folosim valoarea alfa a sursei ca sourceFactor și (1 - alpha) ca destinationFactor. Pentru aceasta trebuie să apelam următoarea funcție:

```
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Ordinea în care rasterizăm obiectele în această situație este foarte importantă. Când desenați o astfel de scenă, trebuie să:

- desenați toate obiectele opace
- sortați obiectele transparente (ordinea adâncimii)
- desenați obiectele transparente în ordine (ordinea adâncimii)

5 Temă

- Implementați toate noțiunile prezentate în această lucrare de laborator.