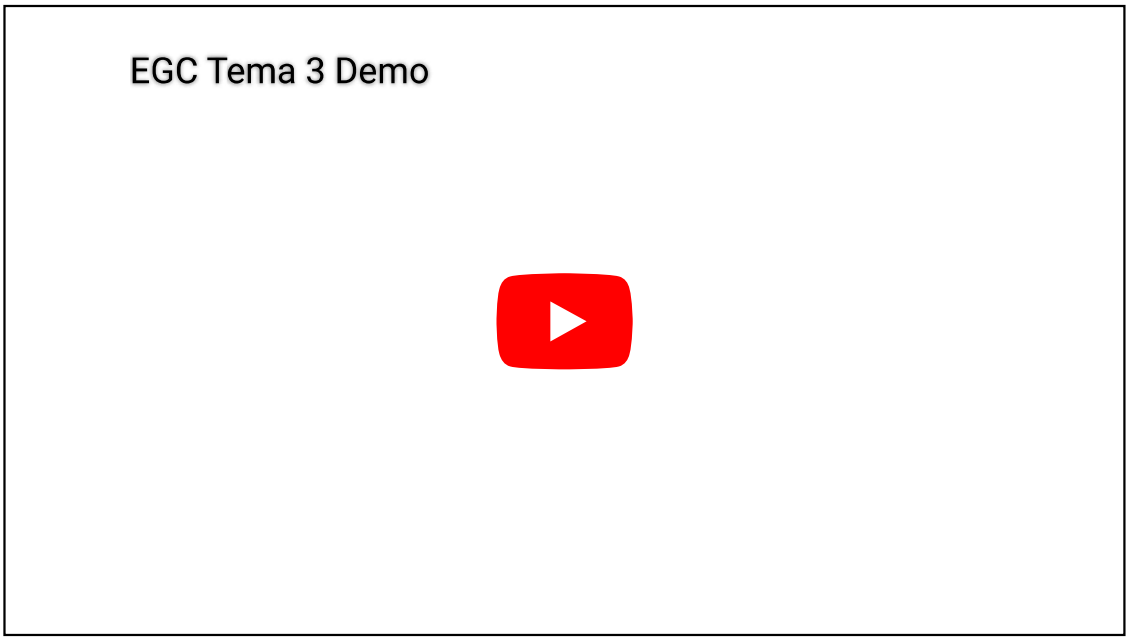
Tema 3 - Disco Party

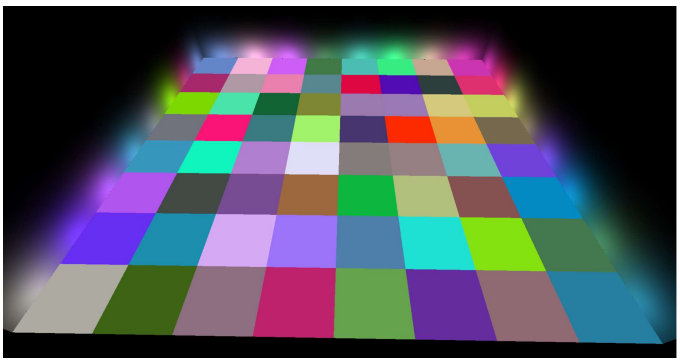
- **Responsabili:** Robert Caragicu, Alex Dinu, Cristi Lambru
- **Lansare:** 17 decembrie
- **Termen de predare:** 23 ianuarie 2021, ora 23:55
- **Regulament:** Regulament General
- **Notă: Orice informație ce nu a fost acoperită în acest document este la latitudinea voastră!**

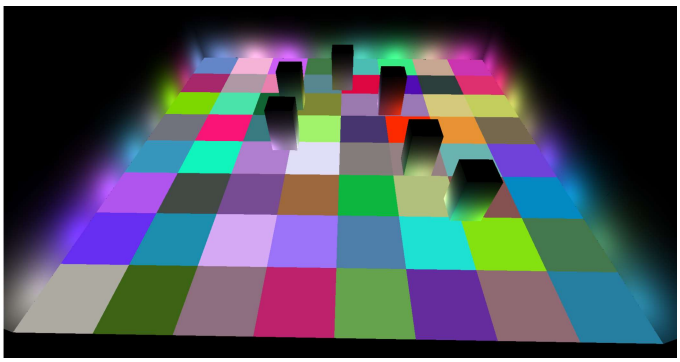
În cadrul temei 3 veți realiza un ring de dans ce va simula iluminarea venită din podea, de la un set de reflectoare și de la un glob disco atașat de tavan. Ringul de dans va fi înconjurat de ziduri și în centru se vor afla câteva forme geometrice ce se vor deplasa aleator și vor reprezenta dansatorii noștri :) . Un exemplu al unui astfel de demo poate fi vizualizat mai jos:



Rezultatul vizual poate diferi foarte mult în implementarea realizată de voi. Culorile, intensitățile și factorii de atenuare utilizați pentru lumini pot oferi rezultate vizuale foarte diferite.

Iluminarea emisă de suprafața ringului de dans





Podeaua ringului de dans va fi formată dintr-un grid de 8×8 celule, unde fiecare celulă va avea o culoare aleasă aleator. Iluminarea emisă de o singură celulă va fi simulată prin utilizarea unei surse de lumină punctiformă ce se află în centrul celulei. Geometria unei celule va fi desenată doar cu componenta emisivă, fără a se lua în calcul celelalte componente ale sursei de lumină utilizate pentru aproximarea iluminării emise de către celulă. Componenta emisivă va avea aceeași culoare ca a sursei de lumină asociate celulei.

Pentru a nu se introduce în calculul de iluminare a zidurilor și a dansatorilor toate sursele de lumină de pe ringul de dans, se vor introduce anumite constrangeri. Se va limita distanța până la care poate ajunge lumina de la o sursă de lumină a unei celule. Dacă presupunem că un dansator este luminat doar de celula în care se află și de celulele vecine, sunt suficiente doar 9 lumini pentru a lumina dansatorul. Zidurile se pot crea modular din mai multe coloane unite pentru a limita numărul de lumini.

Limitarea distanței la care ajunge lumina de la o sursă se realizează cu factorul de atenuare. Astfel, este necesar un factor de atenuare care să permită acest control. Pentru demo s-a folosit în fragment shader:

```
float att = 0.0f;

// light_radius reprezinta distanta maxima pana la care ajunge lumina unei surse
// dist reprezinta distanta de la sursa de lumina la punctul pentru care se calculeaza iluminarea

// chiar daca s-au luat cele mai apropiate lumini fata de obiect,
// la nivel de pixel, distanta poate fi mai mare decat raza
if (dist < light_radius)
        att = pow(light_radius - dist, 2);
```

Puteți utiliza orice alt factor de atenuare care permite controlul asupra distanței la care ajunge lumina de la o sursă de lumină.

Calculul de iluminare pentru mai multe surse de lumină

Pentru a putea calcula iluminarea ce provine de la mai multe surse de lumină, trebuie trimise toate aceste surse către shader (poziție, culoare). Iluminarea finală este suma iluminării fiecărei surse de lumină în parte, conform formulei de mai jos.

$$culoarea = K_e + I_a \cdot K_a + \sum_i f_{at_i} \cdot I_{sursă_i} (K_d \cdot max(\vec{N} \cdot \overrightarrow{L_i}, 0) + K_s \cdot lum_i \cdot max(\vec{N} \cdot \overrightarrow{H_i}, 0)^n)$$

- În cadrul acestui demo se va folosi modelul de shading Phong. Astfel, toate calculele de iluminare se vor realiza în fragment shader.

- Pentru niciuna din sursele de lumină din demo **NU** este necesara calcularea componentei ambientale. Este important calculul componentelor difuze și speculare, iar pentru celulele ringului de dans a celei emisive.
- Factorul de atenuare care limitează distanța de iluminare poate fi folosit doar pentru sursele de lumină punctiformă, prezentate anterior. Pentru celelalte surse de lumină se pot folosi alți factori de atenuare.
- Se poate folosi intensitatea luminii pentru a controla luminozitatea unei surse de lumină. Această intensitate poate fi supraunitară.

Structurare cod

În GLSL se pot defini funcții pentru a modulariza codul. Putem folosi acest lucru pentru calculele de iluminare să putem ușor lucra cu multe surse de lumină. Execuția shaderului începe din funcția main ce nu are parametri și nu întoarce o valoare.

O idee de funcție ar fi:

```
vec3 PointLightContribution(vec3 lightPos, vec3 lightColor)
{
        vec3 color;
        //calculele componentelor difuze si speculare din modelul Phong de iluminare pentru lumina punctiforma
        return color;
}
```

Putem accesa orice variabilă globală din orice funcție din cod, inclusiv uniforme. Astfel putem citi în funcția `PointLightContribution` normala suprafeței și constante de material, fără să le trimitem ca parametri.

Putem specifica ce tip de parametrii are funcția:

- `in` înseamnă că valoarea va fi copiată când se apelează funcția. Funcția poate modifica parametrul cum dorește
- `out` înseamnă că valoarea nu fi inițializată de apelant și după ce funcția modifică parametrul valoarea va fi copiată în variabila corespunzătoare apelantului
- `inout` le combină pe cele două

Dacă nu se specifică, parametrul este de tip `in`. Astfel, putem construi funcții ce întorc mai multe valori.

Metoda de declarare și definiție a funcțiilor este similară cu cea din C.

```
vec3 PointLightContribution(vec3 lightPos, vec3 lightColor);

void main()
{
	//...
}

vec3 PointLightContribution(vec3 lightPos, vec3 lightColor)
{
	//...
}
```

Nu este permisă recursivitate în GLSL.

Pentru a trimite ușor multe surse de lumină, putem defini vectori de uniforme. De exemplu, pentru a trimite mai multe surse de lumină punctiforme se declară în shader un vector de uniforme:

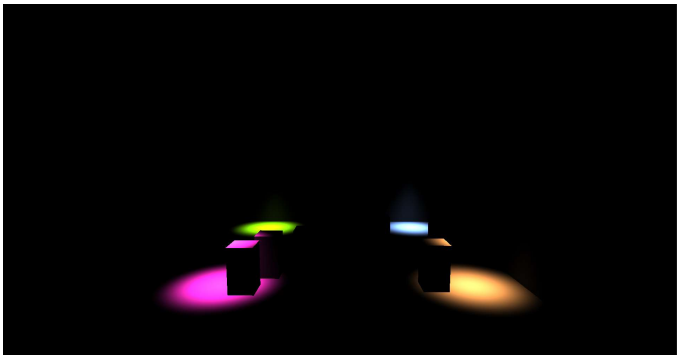
```
uniform vec3 pointLightPos[9];
uniform vec3 pointLightColor[9];
```

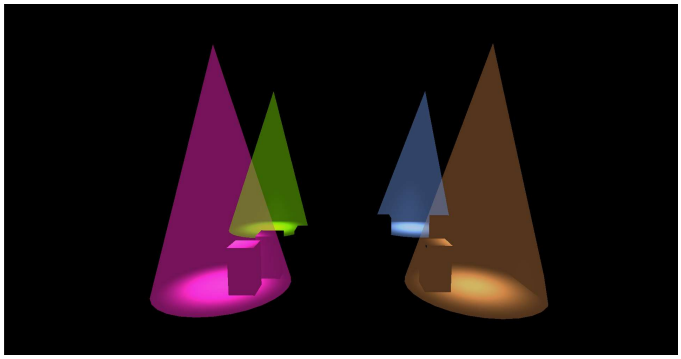
Pe urmă putem trimite uniformele cu un apel în cod:

```
glm::vec3 pointLightPos[9];
glm::vec3 pointLightColor[9];

GLuint location = glGetUniformLocation(program, "pointLightPos");
glUniform3fv(location, 9, glm::value_ptr(pointLightPos[0]));
```

Iluminarea ce provine de la reflectoare

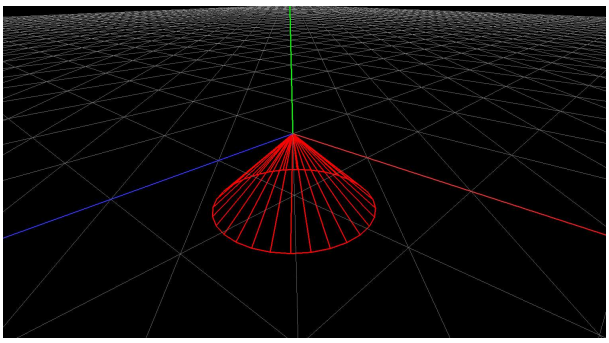




Iluminarea ce provine de la reflectoare se va simula cu surse de lumină de tip spot.

În plus, pentru a se îmbunătăți efectul vizual, se va simula zona de lumină a spot-ului prin desenarea unui con transparent suprapus cu direcția de iluminare a spot-ului.

Pentru a obține conul de lumină, trebuie generată geometria unui con. Pentru simplitate, geometria conului este similară cu cea a unui disc de cerc de rază 1, cu centrul la poziția (0, -1, 0) și creată în planul XOZ. Singura modificare față de discul de cerc este că vertexul din centrul discului se află în originea sistemului de coordonate. Poligoanele bazei nu este necesar sa fie generate. Acesta poate fi viziualizat în figura de mai jos.



Dacă geometria conului este **aceeași** cu cea descrisă mai sus, conul generat poate fi suprapus cu direcția de iluminare a sursei de lumină de tip spot prin transformările:

- Scalare cu `glm::vec3 (tan(spot_angle), 1, tan(spot_angle)) * inaltime`, unde `spot_angle` este unghiul sursei de lumină de tip spot și `inaltime` reprezintă înălțimea pe care o va avea conul.
- Transformările de rotație față de OX, OY sau OZ aplicate asupra direcției de iluminare a sursei de lumină de tip spot.
- Translație la poziția sursei de lumină.

Pentru a se aplica transparența, aceste conuri vor fi desenate ultimele. Pentru procesul de transparență se va folosi in framework:

```
// se vor desena doar fatetele fata
glEnable (GL_CULL_FACE);
glCullFace(GL_BACK);

// aceasta directiva este folosita pentru nu se scrie in depth buffer
glDepthMask(GL_FALSE);

glEnable (GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

// desenare conuri
...

// se dezactiveaza actiunile tuturor directivelor apelate anterior
glDepthMask(GL_TRUE);
glDisable (GL_BLEND);
glDisable (GL_CULL_FACE);
```

În fragment shader se asociază o valoare subunitară canalului alpha al culorii:

```
out_color = vec4(color, 0.5f);
```

Pentru a simula atmosfera unui ring de dans, reflectoarele vor avea o mișcare de rotație aleatoare.

Iluminarea globului disco


Iluminarea globului disco se va simula cu o sursă de lumină punctiformă pentru care culoarea se va obține dintr-o textură.

Această textură va fi generată pe CPU din cod și va conține o culoare aleasă aleator pentru fiecare pixel. Pentru o simulare mai bună a globului disco, se va utiliza o rezoluție de 16×16 pixeli. Proprietățile `GL_TEXTURE_MAG_FILTER` și `GL_TEXTURE_MIN_FILTER` vor fi `GL_NEAREST`.

La poziția globului disco se va desena o sferă pe care va fi aplicată această textură. Coordonatele de textură utilizate pentru eșantionare vor fi calculate din fragment shader. Se consideră că poziția sursei de lumină punctiformă utilizată pentru simularea iluminării globului disco este în centrul acestei sferei. Coordonatele de textură se pot calcula în fragment shader cu o mapare specială cunoscută sub numele de mapare sferică:

```
// light_dir  este directia de iluminare
vec3 light_dir = world_position - disco_ball_position;

// texcoord este coordonata de textura utilizata pentru esantionare
vec2 texcoord;
texcoord.x = (1.0 / (2 * 3.14159)) * atan (light_dir.x, light_dir.z);
texcoord.y = (1.0 / 3.14159) * acos (light_dir.y / length (light_dir));

// color este culoarea corespunzatoare pozitiei world_position
vec3 color = texture (textura_generata, texcoord).xyz;
```

Calcularea iluminării sursei de lumină punctiformă pe restul obiectelor din scenă se va realiza similar ca pentru orice altă lumină de acest tip, cu excepția faptului că se va extrage culoarea sursei de lumină din textura generată. Această culoare se va obține în fragment shader cu același tip de mapare descris în codul de mai sus. În această situație, `color` va fi culoarea sursei de lumină pentru poziția `world_position`.

Pentru o atmosferă mai apropiată unui ring de dans, se va simula o rotație continuă a globului disco, realizată prin translația coordonatelor de textură.

Bonusuri Posibile

- Modele mai complexe decât o primitivă la dansatori (importate sau generate în cod)
- Dansuri mai complexe (să danseze în perechi de exemplu, să sară din când în când etc.)
- Luminile de tip reflector să urmărească din când în când un dansator sau un grup de dansatori
- Schimbarea culorilor luminilor la intervale aleatoare și tranziția graduală între culori:

Pentru a simula atmosfera unui ring de dans, culoarea luminii fiecărei celule din grid se poate schimba la intervale aleatoare. Trecerea de la o culoare la alta poate să nu fie instantă. Pentru un interval scurt de timp se poate face o trecere graduală de la culoarea anterioară la cea curentă. Pentru calcularea culorii de la frame-ul curent se poate folosi un proces de interpolare cu factorul de interpolare `timp_de_la_inceperea_tranzitiei/timp_alocat_tranzitiei`. Culoarea per frame, obținută prin interpolare, se poate calcula înainte de a se trimite la shader. Pentru interpolare se poate folosi din framework:

```
// lastColor este culoarea anterioara si color este culoarea curenta
// t este factorul de interpolare descris mai sus: timp_de_la_inceperea_tranzitiei/timp_alocat_tranzitiei
// de exemplu, daca t este 0.2f, inseamna ca 80% din newColor este lastColor si 20% este color
glm::vec3 newColor = glm::mix (lastColor, color, t);

// in spate, calculul de interpolare este: lastColor * (1.0f-t) + color * t
```

Analog pentru sursele de lumină de tip spot, utilizate pentru simularea iluminării reflectoarelor, se poate schimba culoarea la intervale aleatoare. Tranziția de la culoarea anterioară la cea curentă se poate face gradual timp de un interval scurt. Culoarea de la un anumit frame poate fi calculată prin interpolare înainte de a se trimite la shader.

De asemenea, similar cu celelalte două tipuri de iluminare, se poate regenera textura de culori a sursei de lumină punctiformă utilizată pentru simularea iluminării globului disco la intervale aleatoare. Culorile pixelilor din noua textură vor fi alese aleator. Tranziția de la culorile texturii anterioare la culorile texturii curente va fi realizată gradual într-un interval scurt de timp. Culoarea de la un anumit frame se va calcula prin interpolare în fragment shader, astfel că este necesară trimiterea și eșantionarea în shader a texturii anterioare și a celei curente. Pentru interpolare în fragment shader se poate folosi analog ca la sursele de lumină descrise anterior:

```
// last_color si color reprezinta culorile esantionate din textura
// anterioara si cea curenta, t este factorul de atenuare
vec3 light_color = mix (last_color, color, t);
```

Notare (150p)

- Iluminarea emisă de suprafața ringului de dans (30p)
  - Desenarea grid-ului de celule (5p)
  - Calcularea iluminării în fragment shader pentru dansatori și ziduri (10p)
  - Utilizarea unui factor de atenuare ce limitează iluminarea pe o distanță aleasă (5p)
  - Calcularea și trimiterea spre shader doar a luminilor care au influență asupra unui obiect desenat (10p)
- Iluminarea ce provine de la reflectoare (50p)
  - Calcularea iluminării a cel puțin 4 surse de lumină spot, în fragment shader, pentru dansatori, suprafața ringului de dans și ziduri (10p)
  - Desenarea conurilor de lumină (30p)
  - Rotația aleatoare a direcției surselor de lumină (10p)
- Iluminarea globului disco (40p)
  - Generarea texturii ce va conține o culoare aleasă aleator pentru fiecare pixel (5p)
  - Calcularea iluminării în fragment shader pentru ziduri, dansatori, suprafața ringului de dans, tavan și glob disco (30p)
  - Rotația continuă a globului disco, realizată prin translația coordonatelor de textură (5p)
- Switch între cele 3 tipuri de iluminare la apăsarea unei taste (10p)
- Deplasarea aleatoare a dansatorilor (20p)

Intrebari si raspunsuri

Pentru intrebari vom folosi forumurile de pe moodle. Orice nu este mentionat in tema este la latitudinea fiecarui student!

Notare

Baremul este orientativ. Fiecare asistent are o anumita libertate in evaluarea temelor (de exemplu, sa dea punctaj partial pentru implementarea incompleta a unei functionalitati sau sa scada pentru hard coding). Acelasi lucru este valabil atat pentru functionalitatile obligatorii, cat si pentru bonusuri.

Tema trebuie incarcata pe moodle. Pentru a fi punctata, tema trebuie prezentata la laborator. Vor exista laboratoare speciale de prezentare a temelor (care vor fi anuntate).

Indicatii suplimentare

Tema va fi implementata in OpenGL si C++. Este indicat sa folositi framework-ul si Visual Studio.

Pentru implementarea temei, in folderul **src/lab_m1** puteti crea un nou folder, de exemplu Tema3, cu fisierele Tema3.cpp si Tema3.h (pentru implementare POO, este indicat sa aveti si alte fisiere). Pentru a vedea fisierele nou create in Visual Studio in Solution Explorer, apasati click dreapta pe filtrul lab_m1 si selectati Add→New Filter. Dupa ce creati un nou filtru, de exemplu Tema3, dati click dreapta si selectati Add→Existing Item. Astfel adaugati toate fisierele din folderul nou creat. In fisierul lab_list.h trebuie adaugata si calea catre header-ul temei. De exemplu: #include "lab_m1/Tema3/Tema3.h"

Arhivarea proiectului

- in mod normal arhiva trebuie sa contina toate resursele necesare compilarii si rularii
- inainte de a face arhiva asigurati-va ca ati curatat proiectul Visual Studio:
  - click dreapta pe proiect in **Solution Explorer** → **Clean Solution**
  - si stergeti folderul **/build/.vs** (daca nu il vedeti, **este posibil sa fie ascuns**)
- SAU stergeti complet folderul **/build**
- in cazul in care arhiva tot depaseste limita de 50MB (nu ar trebui), puteti sa stergeti si folderul **/deps** sau **/assets** intrucat se pot adauga la testare. Nu este recomandat sa faceti acest lucru intrucat ingreuneaza mult testarea in cazul in care versiunea curenta a librariilor/resurselor difera de versiunea utilizata la momentul scrierii temei.