МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання лабораторних та самостійних
робіт з дисципліни
«Комп’ютерна графіка та обробка зображень»

Лабораторна робота 4

Київ 2024

Зміст

РОЗДІЛ 3
ДВОВИМІРНІ ТЕКСТУРИ
ТЕКСТУРНІ КООРДИНАТИ
ТЕКСТУРНА МАТРИЦЯ

Текстури – растрові зображення (малюнки, візерунки, фотозображення), які наносяться на грані з використанням алгоритмів масштабування та інтерполяції. Використання текстур підвищує реалістичність та естетичну якість зображень, але вимагає значних обчислювальних ресурсів. Для швидкого виведення текстур використовується блок текстурування відеоприскорювача.

В початковій версії OpenGL підтримувались тільки **1D** та **2D** типи текстур. В нових версіях та в розширеннях бібліотеки використовуються також **3D** і кубічні (**cube maps)** текстури. **1D**-текстури використовуються для створення лінійних шаблонів (наприклад, смугастих візерунків). Найчастіше використовуються **2D**-текстури, які розглядаються в даному посібнику.

Підготовка текстури та завантаження в текстурну пам'ять. Зображення для текстур готуються з допомогою графічного **2D**-редактора. Глибина кольору рекомендується 24 біти. Розміри зображення (висота і ширина) для текстур прийнято задавати рівними степеням двійки, наприклад, 128x128, 256x1024, 512x512. Деякі графічні редактори утворюють навколо зображення рамку і в них потрібно вказувати розміри на одиницю більші. Зберігають підготовлені зображення найчастіше у форматах **BMP, JPG, PNG**.

Перед використанням параметри текстур задаються командами

```
glTexParameter[i f](target, pname, param),
```

де **`target`** – тип текстури (розмірність), **`pname`** – назва параметра, **`param`** – конкретне значення параметра.

Алгоритми масштабування **2D**-текстур (фільтри) задаються командою

```
glTexParameteri(GL_TEXTURE_2D, pname, param)
```

з параметром-назвою **pname=GL_TEXTURE_MAG_FILTER** – для збільшення зображення або **GL_TEXTURE_MIN_FILTER** – для зменшення зображення. Можливі значення параметра **param GL_NEAREST** або **GL_LINEAR.**

Зі значенням **GL_NEAREST** для виведення використовується колір одного найближчого елемента текстури, а зі значенням **GL_LINEAR** – усереднений колір чотирьох сусідніх елементів. Значення за замовчуванням **GL_LINEAR.**

Формат розташування даних текстури у відеопам'яті та зв'язування з конкретним бітовим масивом задаються командою

```
glTexImage2D(GL_TEXTURE_2D,    //тип текстури
    level,           // рівень текстури
    internalFormat, // формат відображення
    width , height, // розміри текстури
    border,          // розмір границі (рамки)
    format,          // формат елементів масиву
     type,           // числовий тип
    @array )         // зв'язування з масивом
```

Параметри взаємодії текстури з об'єктом задаються командою

```
glTexEnvi(GL_TEXTURE_ENV,
              GL_TEXTURE_ENV_MODE, param);
```

З параметром **param=GL_DECAL** враховується тільки колір елементів текстури. Якщо **param=GL_MODULATE**, то результуючий колір пікселя одержується усередненням власного кольору пікселя грані (з врахуванням освітлення) і кольору відповідного елемента текстури (текселя). Якщо **param=GL_BLEND**, то колір текселя змішується із попередньо виведеним зображенням з врахуванням коефіцієнта прозорості. Значення за замовчуванням: **GL_MODULATE**. Для зменшення крайових спотворень текстури можна створити рамку на грані, задавши ненульове значення сьомому

параметру. Вмикається механізм накладання текстур на грані командою **glEnable(GL_TEXTURE_2D),** вимикається командою **glDisable(GL_TEXTURE_2D).**

Приклад 3.1. Процедура завантаження зображення текстури із файла (викликається в обробнику ініціалізації ***OnCreate***, додається до всіх прикладів з текстуруванням).

В розділі змінних головної форми модуля попередньо описуються змінні для бітової матриці та бітового масиву:

```
const tw=512; th=512; // розміри 2D-текстури
var Bmp:TBitmap;//для завантаження зображення
 // масив для передачі кольорів в текстурний об'єкт
 arrayRGB: Array [0..th-1, 0..tw-1, 0..2] of GLubyte;
```

```
procedure LoadBmpTexture(bmp_file_name:string);
var i,j:Integer;
begin bmp:=TBitmap.Create;//створення бітової матриці
   // зчитування зображення з файла
 bmp.LoadFromFile(bmp_file_name);
 for i := 0 to tw-1 do for j := 0 to th-1 do
 begin
     arrayRGB[j,i,0]:=GetRValue(
                      bmp.Canvas.Pixels[i,th-1-j]);
     arrayRGB[j,i,1]:=GetGValue(
                      bmp.Canvas.Pixels[i,th-1-j]);
     arrayRGB[j,i,2]:=GetBValue(
                      bmp.Canvas.Pixels[i,th-1-j]);
 end;    bmp.Free; //звільнення пам'яті
         // створення текстури
 glTexImage2D(GL_TEXTURE_2D,// розмірність текстури
        0,            // рівень текстури
        GL_RGBA,      // формат відображення текселів
        tw, th,       // розміри (ширина, висота)
        0,            // розмір границі (рамки)
        GL_RGB,       // формат елементів масиву
        GL_UNSIGNED_BYTE, // числовий тип
        @arrayRGB );//зв'язування з бітовим масивом
    // задання способу збільшення текстури
   glTexParameteri(GL_TEXTURE_2D,
               GL_TEXTURE_MAG_FILTER, GL_LINEAR);
      // задання способу зменшення текстури
   glTexParameteri(GL_TEXTURE_2D,
                GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

```
// задання способу обчислення кольорів об'єктів
glTexEnvi(GL_TEXTURE_ENV,
          GL_TEXTURE_ENV_MODE, GL_DECAL) end;
```

Зв'язування координат текстури з об'єктом. Команда

```
glTexCoord{1234}{sifd}(TYPE coords)
```

або

```
glTexCoord{1234}{sifd}v(TYPE @coords)
```

встановлює відповідність координат текстури **(s,t,r,q)** з вершиною, яка описана за цією командою. При використанні функції **glTexCoord2*()** небхідно задати тільки координати ***s*** і ***t***. Третя координата ***r*** зарезервована для тривимірних текстур, четверта ***q*** відіграє роль масштабуючого множника.

Координати зображення текстури змінюються в діапазоні від 0.0 до 1.0 (рис.3.1). Нижній лівий кут текстури має координати (0.0,0.0), а верхній правий – (1.0,1,0).

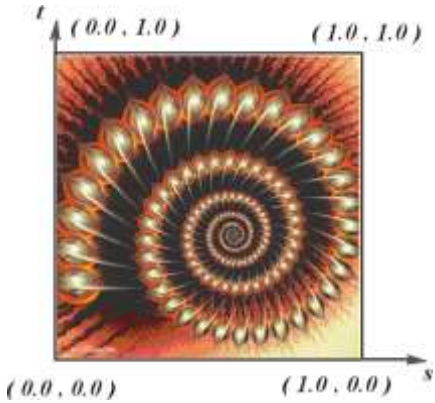


Рис.3.1. Координати 2D-текстури.

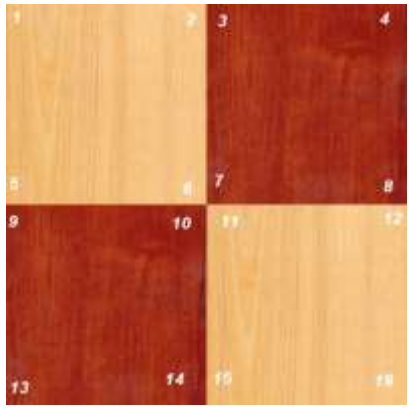


Рис.3.2. Текстура з мітками.

На гранях для кожної із вершин задаються текстурні плоскі координати, найчастіше теж в діапазоні від 0.0 до 1.0. Варіант відповідності координат текстури і вершин, тобто розміщення текстури на грані залежить від конкретної задачі. Якщо потрібно ретельно проаналізувати накладання текстури на складну поверхню, можна для контролю додати до зображення текстури цифрові мітки (рис.3.2).

Приклад 3.2. Текстурований куб (бічні грані, рис.3.3).

Для кожної грані куба перед кожною вершиною задаємо текстурні координати процедурою **`glTexCoord2d(s,t).`**

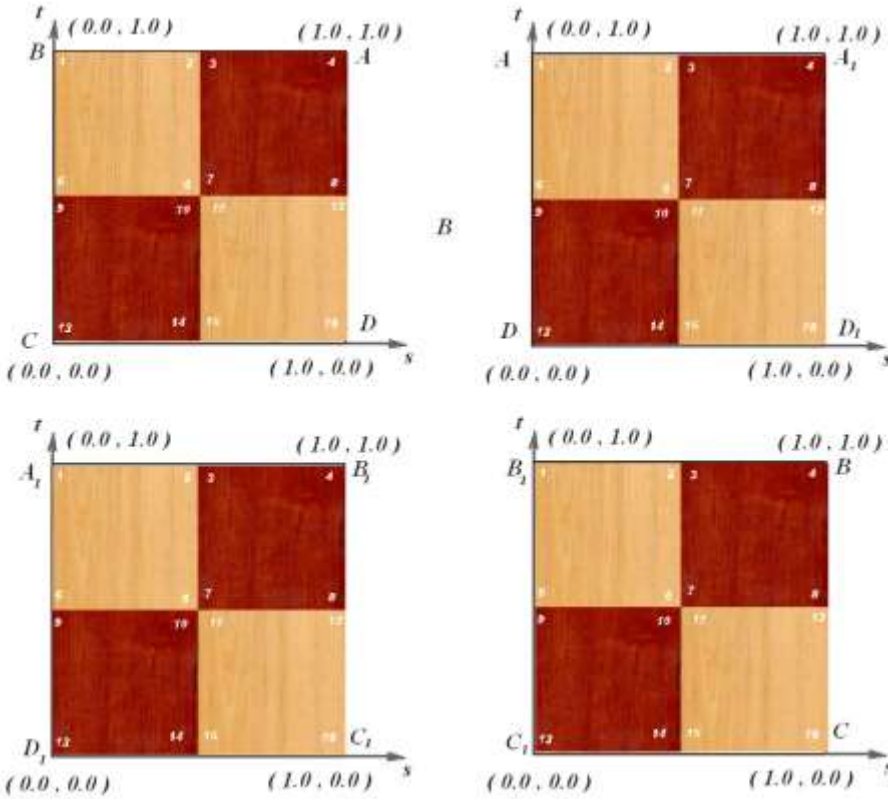


Рис.3.3. Текстурні координати бічних граней куба.

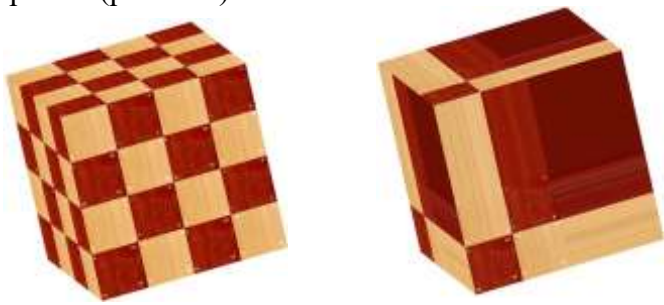
```
procedure Draw3D;// Текстурований куб (бічні грані)
const h=1.0;// h-половина довжини ребра куба
begin // вмикання механізму відображення текстури
 glEnable(GL_TEXTURE_2D);
 glBegin(GL_QUADS);// режим виведення 4-кутників
  // передня грань куба
 glTexCoord2d (1, 1); glVertex3f( h,  h,  h);  // A
 glTexCoord2d (0, 1); glVertex3f(-h,  h,  h);  // B
 glTexCoord2d (0, 0); glVertex3f(-h, -h,  h);  // C
 glTexCoord2d (1, 0); glVertex3f( h, -h,  h);  // D
```

```
   // права грань куба
  glTexCoord2d (1, 1); glVertex3f( h,  h, -h);  // A1
  glTexCoord2d (0, 1); glVertex3f( h,  h,  h);  // A
  glTexCoord2d (0, 0); glVertex3f( h, -h,  h);  // D
  glTexCoord2d (1, 0); glVertex3f( h, -h, -h);  // D1
   // задня грань куба
  glTexCoord2d (1, 1); glVertex3f(-h,  h, -h);  // B1
  glTexCoord2d (0, 1); glVertex3f( h,  h, -h);  // A1
  glTexCoord2d (0, 0); glVertex3f( h, -h, -h);  // D1
  glTexCoord2d (1, 0); glVertex3f(-h, -h, -h);  // C1
   // ліва грань куба
  glTexCoord2d (1, 1); glVertex3f(-h,  h,  h);  // B
  glTexCoord2d (0, 1); glVertex3f(-h,  h, -h);  // B1
  glTexCoord2d (0, 0); glVertex3f(-h, -h, -h);  // C1
 glTexCoord2d (1, 0);  glVertex3f(-h, -h,  h);  // C
glEnd end;
```

Якщо задати координати текстур на грані в діапазоні від ***0.0*** до ***m*** і задати спосіб продовження текстури

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

то зображення текстури буде повторюватись на грані ***m*** разів в обох напрямах (рис.3.4а).



а) з повторенням б) з продовженням крайніх пікселів

Рис.3.4. Два способи продовження текстури на гранях.

Якщо задати спосіб продовження текстури командами

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

то повторюватись до границі грані (продовжуватись) будуть тільки крайні точки текстури (рис.3.4б).

В конвеєрі OpenGL використовується спеціальна **текстурна матриця**, яка множиться на текстурні координати вершин перед виведенням текстури. Автоматично ініціалізується одинична матриця текстур. Текстурну матрицю можна використати для повороту або зміни масштабу текстури. Наприклад, для дзеркального відображення згори вниз і розтягування вдвічі, потрібно виконати таку послідовність команд:

```
// активізуємо матрицю текстур
  glMatrixMode (GL_TEXTURE);
  glLoadIdentity; // завантажуємо одиничну матрицю
// розтягуємо текстуру і відображаємо відносно OX
  glScale(-0.5,0.5,0.5);
 // активізуємо модельну матрицю
  glMatrixMode (GL_MODELVIEW);
```

Примітка. Після виконання операцій з масштабованою текстурою потрібно відновити одиничну текстурну матрицю для коректної роботи інших частин програми.

Обчислення зображень для текстур. Зображення для текстур у вигляді візерунків чи фракталів можна не завантажувати з файла, а обчислювати за формулами.

Приклад 3.3. Процедура обчислення зображення для текстури (викликається в обробнику ініціалізації ***OnCreate*** замість **LoadBMPTexture)**.

```
const tw=512; th=512; // розміри 2D-текстури
var  // масив з бітовими площинами кольорів
 arrayRGB: Array [0..th-1, 0..tw-1, 0..2] of GLubyte;
```

```
procedure CalculateTexture;
begin   CalculateArrayRGB;//обчислення масиву
// автоматична побудова рівнів пірамідальної текстури
  gluBuild2DMipmaps(GL_TEXTURE_2D, // розмірність
                GL_RGBA, // формат пікселів
                tw,  th,  // розміри текстури
                GL_RGB,   // формат текселів
                GL_UNSIGNED_BYTE, //числовий формат
                @ arrayRGB); // зв'язування з массивом
```

```
   // задання способу збільшення текстури
  glTexParameteri(GL_TEXTURE_2D,
                  GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    // задання способу зменшення текстури
  glTexParameteri(GL_TEXTURE_2D,
     GL_TEXTURE_MIN_FILTER, GL_LINEAR_MIPMAP_LINEAR);
   // задання способу обчислення кольорів об'єктів
  glTexEnvi(GL_TEXTURE_ENV,
              GL_TEXTURE_ENV_MODE, GL_DECAL) end;
```

Приклад 3.3.1. Процедура обчислення масиву з бітовими площинами зображення у вигляді чотирьох клітинок (рис.3.5).

```
procedure CalculateArrayRGB;
var i,j: integer;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
    if (i<tw div 2)and(j<th div 2) OR
       (i>tw div 2)and(j>th div 2) then
    begin arrayRGB[j,i,0]:= 100; arrayRGB[j,i,1]:= 0;
          arrayRGB[j,i,2]:= 60
    end else begin arrayRGB[j,i,0]:= 255 ;
    arrayRGB[j,i,1]:= 240; arrayRGB[j,i,2]:= 0   end
end;
```

Рис.3.5.

Рис.3.6.

Приклад 3.3.2. Процедура обчислення масиву зображення у вигляді чотирьох трикутників (рис.3.6).

```
procedure CalculateArrayRGB;
var i,j: integer;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
  begin if (i>j)and(i+j>tw)or(i<j)and(i+j<tw) then
    arrayRGB[j,i,0]:= 255 else arrayRGB[j,i,0]:= 0;
   arrayRGB[j,i,1]:= 100; arrayRGB[j,i,2]:= 0
 end end;
```

Приклад 3.3.3. Процедура обчислення масиву зображення у вигляді концентричних кілець (рис.3.7).

```
procedure CalculateArrayRGB;
var i,j:integer; x,y,r:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin x:=(i-tw div 2)/tw*8;y:=(j-th div 2)/th*8;
  r:=sqrt(x*x+y*y);
  if frac(r)<0.2  then
  begin arrayRGB[j,i,0]:= 255;
   arrayRGB[j,i,1]:= 255; arrayRGB[j,i,2]:= 0 end
  else begin arrayRGB[j,i,0]:= 0;arrayRGB[j,i,1]:= 0;
         arrayRGB[j,i,2]:= 255 end end end;
```

Рис.3.7.

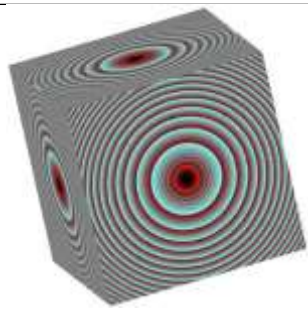


Рис.3.8.

Приклад 3.3.4. Процедура обчислення масиву зображення у вигляді градієнтних концентричних кілець (рис.3.8).

```
procedure CalculateArrayRGB;
var i,j:integer;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin arrayRGB[j,i,0]:=
  (sqr(i-tw div 2)+sqr(j-th div 2)) div 5 mod 256;
  arrayRGB[j,i,1]:=
  (sqr(i-tw div 2)+sqr(j-th div 2)) div 32 mod 256;
  arrayRGB[j,i,1]:= 0; arrayRGB[j,i,2]:= 255 end end;
```

Приклад 3.3.5. Процедура обчислення масиву зображення з візерунком у вигляді зафарбованих секторів (рис.3.9).

```
procedure CalculateArrayRGB;
var i,j:Integer; x,y,fi:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin x:=(i-tw div 2)/tw*8; y:=(j-th div 2)/th*8;
```

```
  if x<>0.0 then fi:=arctan(y/x) else fi:=pi/2;
  if frac(abs(sin(8*fi)))<0.75 then
  begin arrayRGB[j,i,0]:= 255;
   arrayRGB[j,i,1]:= 255; arrayRGB[j,i,2]:= 0 end
  else begin arrayRGB[j,i,0]:=0; arrayRGB[j,i,1]:= 0;
             arrayRGB[j,i,2]:= 255 end   end end;
```

Рис.3.9.

Рис.3.10.

Приклад 3.3.6. Процедура обчислення масиву зображення з візерунком у вигляді спіральних секторів (рис.3.10)

```
procedure CalculateArrayRGB;
var i,j:integer; x,y,r,fi:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin x:=(i-tw div 2)/tw*8; y:=(j-th div 2)/th*8;
  if x<>0.0 then fi:=arctan(y/x) else fi:=pi/2;
  r:=sqrt(x*x+y*y);
  if frac(abs(cos(8*fi-r)))<0.75 then
  begin arrayRGB[j,i,0]:= 255; arrayRGB[j,i,1]:= 255;
        arrayRGB[j,i,2]:= 0 end
  else begin arrayRGB[j,i,0]:=0; arrayRGB[j,i,1]:=0;
             arrayRGB[j,i,2]:= 255 end
 end end;
```

Приклад 3.3.7. Процедура обчислення масиву зображення з візерунком (рис.3.11).

```
procedure CalculateArrayRGB;
var i,j:integer; x,y,r,fi:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin x:=(i-tw div 2)/tw*8; y:=(j-th div 2)/th*8;
  if x<>0.0 then fi:=arctan(y/x) else fi:=pi/2;
  r:=sqrt(x*x+y*y);
  if frac(abs(sin(8*fi))*r)<0.75 then
  begin arrayRGB[j,i,0]:= 255; arrayRGB[j,i,1]:= 255;
        arrayRGB[j,i,2]:= 0 end
```

```
else begin arrayRGB[j,i,0]:=0; arrayRGB[j,i,1]:=0;
          arrayRGB[j,i,2]:= 255 end end end;
```

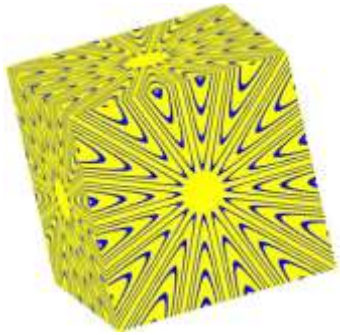


Рис.3.11.

Рис.3.12.

Приклад 3.3.8. Процедура обчислення масиву зображення з візерунком (рис.3.12).

```
procedure CalculateArrayRGB;
var i,j:integer; x,y,r,fi:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
 begin x:=(i-tw div 2)/tw*8; y:=(j-th div 2)/th*8;
  if x<>0.0 then fi:=arctan(y/x) else fi:=pi/2;
  r:=sqrt(x*x+y*y);
  if frac(abs(cos(8*fi)-cos(r)))<0.75 then
  begin arrayRGB[j,i,0]:=255;  arrayRGB[j,i,1]:=255;
        arrayRGB[j,i,2]:= 0 end
  else begin arrayRGB[j,i,0]:=0; arrayRGB[j,i,1]:=0;
             arrayRGB[j,i,2]:= 255 end end end;
```

Приклад 3.3.9. Процедура обчислення масиву зображення з візерунком (рис.3.13).

```
procedure CalculateArrayRGB;
var i,j:Integer; x,y,r,fi:GLfloat;
begin for i := 0 to tw-1 do for j := 0 to th-1 do
  begin x:=(i-tw div 2)/tw*8; y:=(j-th div 2)/th*8;
   if x<>0.0 then fi:=arctan(y/x) else fi:=pi/2;
   r:=sqrt(x*x+y*y);
   if frac(abs(cos(4*fi)-r))<0.75 then
   begin arrayRGB[j,i,0]:=255; arrayRGB[j,i,1]:= 255;
        arrayRGB[j,i,2]:= 0 end
   else begin arrayRGB[j,i,0]:=0; arrayRGB[j,i,1]:=0;
             arrayRGB[j,i,2]:= 255 end end end;
```

Для візерунку на рис.3.14 досить у вищенаведеній процедурі `cos(4*fi)` змінити на `cos(8*fi)`.

Рис.3.13.

Рис.3.14.

Лабораторна робота №4

Тема: 2D-текстури. Підготовка та завантаження зображень. Текстурні координати, текстурна матриця

Завдання 3.1. Підготувати та зберегти в окремій папці зображення текстури у форматі **BMP** з розмірами 512x512 пікселів та глибиною кольору 24 біти. Деякі графічні редактори утворюють навколо зображення рамку, тому в них потрібно вказувати розміри на одиницю більші.

Завдання 3.2 Доповнити проект із завдання 1.1 лабораторної роботи 2 процедурою `LoadBmpTexture` з прикладу 3.1 (викликати в OnCreate) та замінити процедуру `Draw3D` процедурою з прикладу 3.2. Доповнити процедуру `Draw3D` командами для виведення текстури на верхній та нижній гранях куба. Скріншоти зберегти в звіті. Зберегти всі файли проекту в окремому каталозі на диску для звіту.

Завдання 3.3. Шахівниця на кубі. Створити зображення шахових клітинок на гранях куба (рис.3.15). Для цього підготувати попередньо два зображення, які відповідають світлому та темному матеріалу з розмірами 256x256 пікселів згідно варіанту (бажано вибирати однорідні частини матеріалів без помітних дефектів).

Таблиця 3.1. Матеріали для текстурування шахівниці

№	Світлі	Темні
1	Світлий горіх.	Темний горіх.
2	Світлий дуб.	Темний дуб.
3	Сосна.	Ясен.
4	Вільха.	Чорне дерево.
5	Каштан.	Тис.
6	Білий мармур.	Чорний мармур.
7	Палісандр.	Махагон.
8	Зелена яшма.	Червона яшма.
9	Світлий бурштин.	Темний бурштин.
10	Слонова кістка.	Червоний мармур.
11	Тюльпанове дерево.	Каучукове дерево.
12	Білий нефрит.	Сірий халцедон.

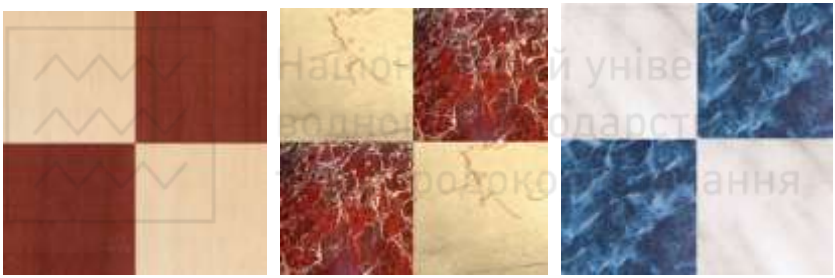


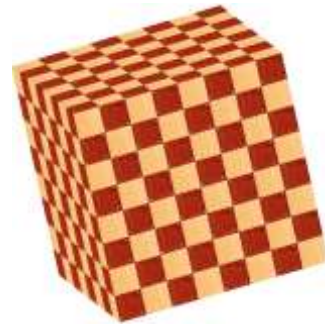
Рис.3.15. Приклади текстур для шахових клітинок

Зберегти у файлах ***1.BMP*** та ***2.BMP***. Створити нове вікно у редакторі ***Paint, Photoshop*** або аналогічному. Задати розміри зображення 512x512 пікселів. Вставити дві копії зображення із файла ***1.BMP*** і розмістити точно в протилежних кутах. У вільні квадрати вставити дві копії із файла ***2.BMP***. Одержане зображення частини шахівниці у вигляді комбінації двох темних і двох світлих клітинок з відповідних матеріалів зберегти у файлі ***2x2.bmp***. Деякі графічні редактори утворюють навколо зображення рамку, тому зберегти ще один варіант зображення у файлі ***2x2a.bmp*** з розмірами 513x513 пікселів. Скопіювати файли проекту з попереднього завдання в нову папку. Додати файл з текстурою ***2x2.BMP***.

mult = 1

mult = -1

mult = 1,5



mult = 4

mult = 0,5

mult = 0,75

Рис.3.16. Текстурований куб з масштабованою матрицею текстур.

Для повторення текстури на грані в процедурі **Draw3D** перед виведенням куба задати режим повторення

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

та масштабувати матрицю текстур (рис.3.16):

```
const mult=4.0; // множник для масштабування текстури
// активізуємо матрицю текстур
  glMatrixMode (GL_TEXTURE);
  glLoadIdentity; // завантажуємо одиничну матрицю
  glScale( mult, mult, mult); // масштабуємо текстуру
 // активізуємо модельну матрицю
  glMatrixMode (GL_MODELVIEW);
```

Скріншоти зберегти в звіті. Відлагодити та зберегти всі файли проекту в окремому каталозі на диску для звіту.

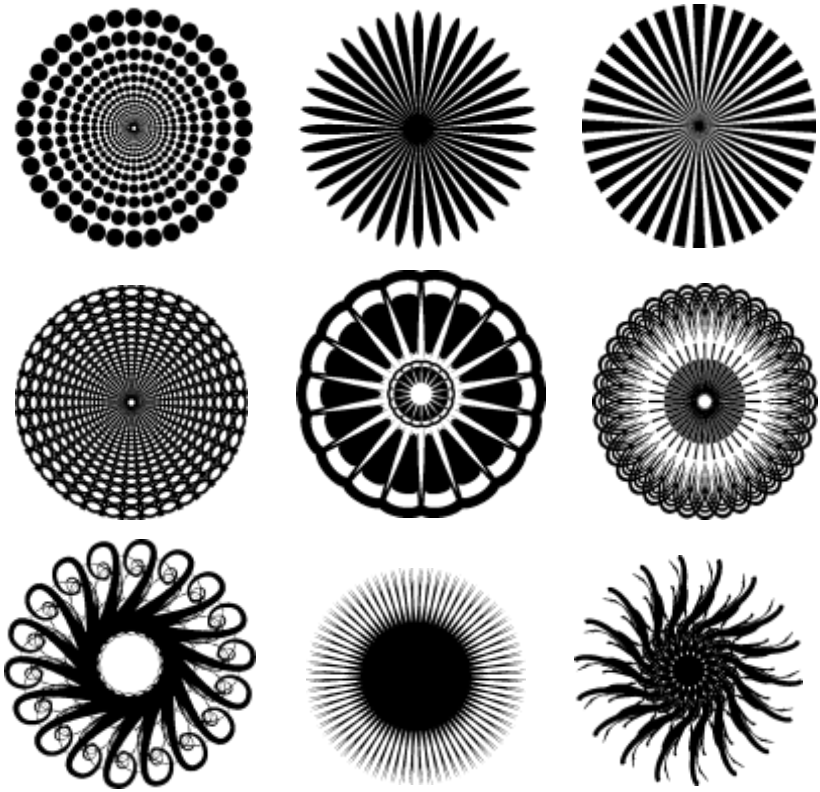


Рис.3.17. Приклади візерунків для текстур.

Завдання 3.4. Доповнити копію проекту із завдання 3.2 процедурою **`CalculateTexture`** з прикладу 3.3 (викликати в ***OnCreate*** замість **`LoadBmpTexture`**). Створити власний

варіант зображення з візерунком (рис.3.17.). Скріншоти зберегти в звіті. Відлагодити та зберегти всі файли проекту в окремому каталозі на диску для звіту.

!!! Оформити звіт з лабораторної роботи дотримуючись вимог.

Створити папку, підписати її своїм прізвищем, зберегти в ній виконані завдання та завантажити її на Гугл-диск в папку ЛБ 3.

Контрольні запитання

1. Які типи текстур використовуються в бібліотеці OpenGL та її розширеннях?
2. Опишіть порядок роботи з текстурами в OpenGL .
3. Вкажіть формати графічних файлів для зберігання растрових зображень. В яких з них використовуються алгоритми стискання?
4. Як підготувати та завантажити текстуру в OpenGL?
5. З якими параметрами викликається команда **`glTexImage2D`**?
6. Які параметри задаються для відтворення текстур?
7. Як задаються текстурні координати?
8. Як накласти певний колір на малюнок текстури?
9. Як повернути зображення на грані, не змінюючи текстурних координат?
10. Як досягається масштабування текстури на гранях?
11. Для чого можна використати повторення чи продовження текстури?