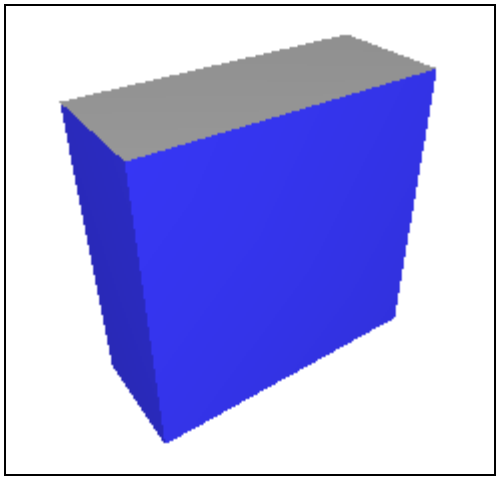
OpenGL – teksturowanie

- **Teksturowanie polega na pokrywaniu wielokątów obrazami (plikami graficznymi)**
- **Umożliwia znaczące zwiększenie realizmu sceny przy niewielkim zwiększeniu nakładu obliczeniowego**
- **Rozwój akceleratorów graficznych w ciągu ostatnich lat skupiał się na rozbudowie sprzętowych funkcji wspomagających przechowywanie, szybki transfer i rozkładanie tekstur na elementach sceny**
- **Większość efektów symulujących oświetlenie w grach komputerowych realizowana jest z zastosowaniem teksturowania**



Etapy teksturowania

- **Utworzenie zbioru tekstur na podstawie plików graficznych (np. DIB)**
  - **Wczytanie zawartości plików graficznych**
  - **Konwersja danych graficznych na tekstury**
  - **Utworzenie tablicy obiektów tekstur**
  - **Określenie filtrowania tekstur**
  - **Zdefiniowanie funkcji tekstur**
  - **Utworzenie obrazów tekstur**
- **„Rozciąganie" tekstur na wielokątach**
  - **Włączenie teksturowania dla określonych fragmentów sceny**
  - **Powiązanie współrzędnych obrazów tekstur ze współrzędnymi wierzchołków określonych wielokątów**
  - **Wyłączenie teksturowania**

Teksturowanie - wczytanie zawartości plików graficznych

```
#include <gl\glaux.h>

AUX_RGBImageRec *LoadBMP(char *Filename) // Ładuje plik DIB
{
    FILE *File=NULL;                    // Uchwyt do pliku

    if (!Filename)              // Sprawdzenie, czy podano nazwę
    {
        return NULL;
    }
    File=fopen(Filename,"r");

    if (File)                   // Czy istnieje plik?
    {
        fclose(File);           // Zamknij uchwyt

        // Załaduj plik DIB i zwróć uchwyt do tablicy plikseli
        return auxDIBImageLoad(Filename);
    }
    return NULL; // Jeśli załadowanie się nie udało, zwróć 0
}
```

Teksturowanie – utworzenie obiektów tekstur

```
GLuint texture[2];   // tablica tekstur OpenGL


int LoadGLTextures()// Załaduj bitmapy i konwertuj na tekstury
{   int Status=FALSE, i;

    AUX_RGBImageRec *TextureImage[2]; //wskaźniki do danych
    memset(TextureImage,0,sizeof(void *)*2); //zerowanie

    // Zastosuj LoadBMP do załadowania danych z plików BMP
    if ((TextureImage[0]=LoadBMP("Data/glass.bmp"))
        &&(TextureImage[1]=LoadBMP("Data/sky.bmp"))    )
    {
    Status=TRUE;

    //Utworzenie nazw tekstur:
    glGenTextures(2, &texture[0]);
```

Teksturowanie – utworzenie obiektów tekstur

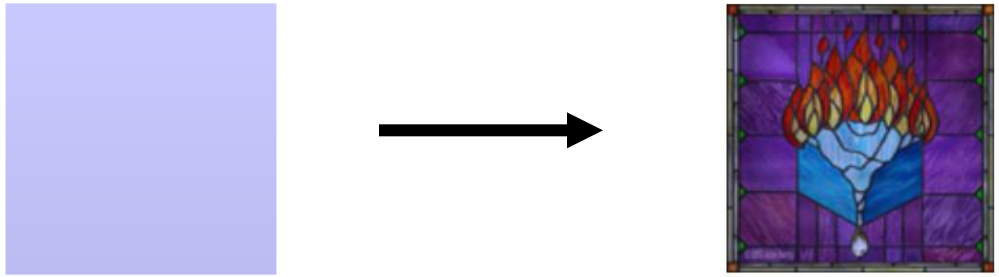
```
// Typowa generacja tekstur na podstawie danych z DIB:
    for(i=0;i<2;i++)
    { //Tworzenie i stosowanie obiektu tekstury:
      glBindTexture(GL_TEXTURE_2D, texture[i]);

      // Przekształcenie danych z pliku DIB na teksturę OpenGL
      glTexImage2D
        ( GL_TEXTURE_2D,                  // Tworzona jest tekstra 2D
          0,                              // Rozdzielczość na razie 0
          3,                              // Ilość składowych koloru
          TextureImage[i]->sizeX,         // Szerokość
          TextureImage[i]->sizeY,         // Wysokość
          0,                              // Ramka
          GL_RGB,                         // Format kolorów tekstury
          GL_UNSIGNED_BYTE,               // Typ danych obrazu
          TextureImage[i]->data);         // Wskaźnik do danych
     // Filtrowanie tekstur
     glTexParameteri( GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
                      GL_LINEAR);
     glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,
                      GL_LINEAR);
    }   }
```

Teksturowanie – utworzenie obiektów tekstur

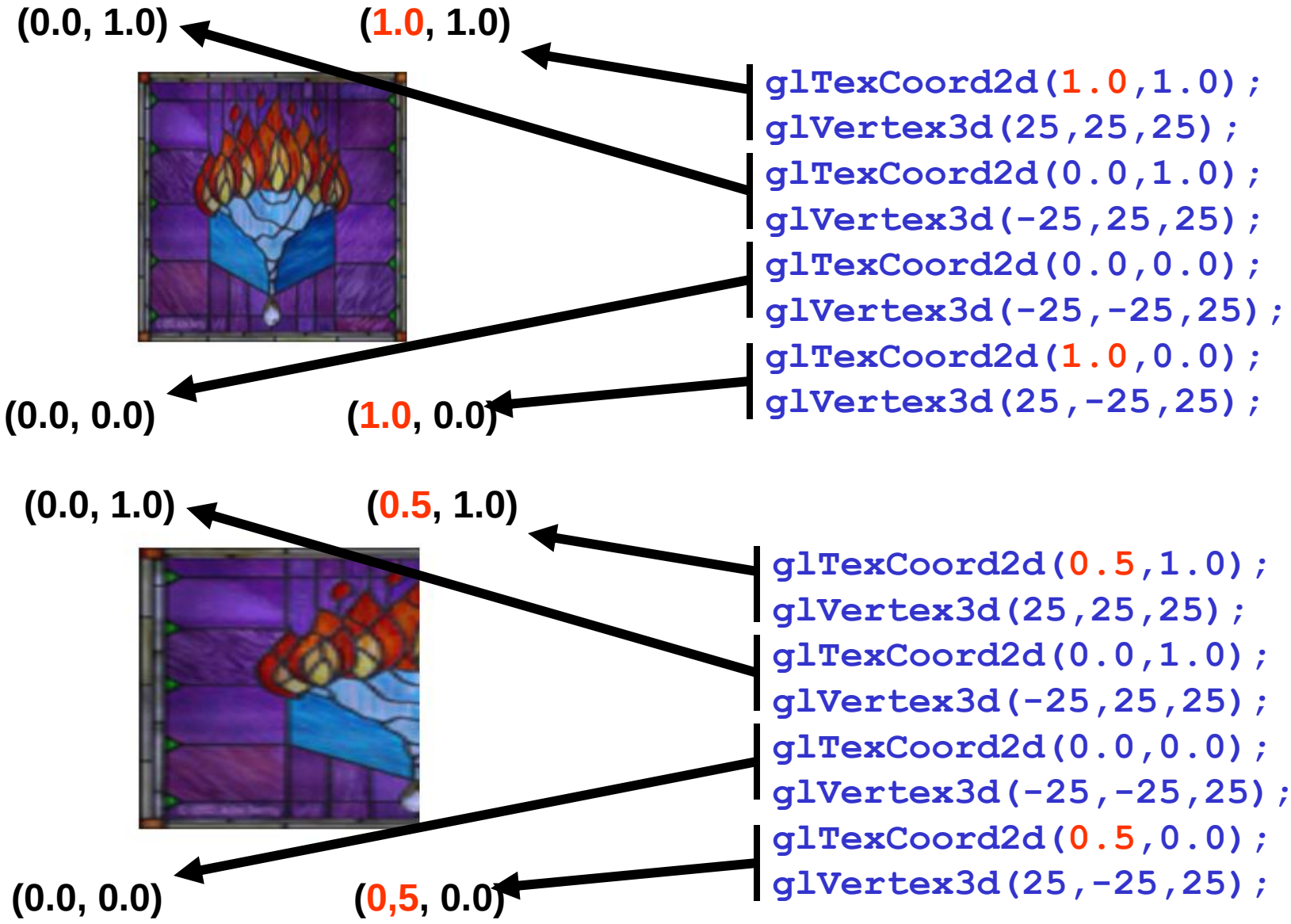
```
    // Porządkowanie pamięci:
    for(i=0;i<2;i++)
    {
    if (TextureImage[i])    //Jeśli tekstura istnieje
    {
        if (TextureImage[i]->data) //Jeśli obraz istnieje
        {
                free(TextureImage[i]->data);
                // Zwolnij pamięć zajmowaną przez obraz
        }
        free(TextureImage[i]);
                // Zwolni pamięć zajmowaną przez teksturę
    }
    }
    return Status;
}
```

Teksturowanie – pokrywanie obiektów teksturą

```
void kwadrat(void)
{   glColor3d(0.7,0.7,0.9);
    // Funkcje tekstury: GL_MODULATE, GL_DECAL, GL_BLEND
    glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_MODULATE);
    // Określ bieżący obiekt tekstury:
    glBindTexture(GL_TEXTURE_2D,texture[0]);
    glEnable(GL_TEXTURE_2D); // Włącz teksturowanie
    glBegin(GL_QUADS);
        glNormal3d(0,0,1);
        // Powiąż współrzędne tekstury z wierzchołkami:
        glTexCoord2d(1.0,1.0);      glVertex3d(25,25,25);
        glTexCoord2d(0.0,1.0);      glVertex3d(-25,25,25);
        glTexCoord2d(0.0,0.0);      glVertex3d(-25,-25,25);
        glTexCoord2d(1.0,0.0);      glVertex3d(25,-25,25);
    glEnd();
    glDisable(GL_TEXTURE_2D); // Wyłącz teksturowanie
}
```



Teksturowanie – współrzędne tekstury



Teksturowanie – mipmampy

```
glTexParameteri(     GL_TEXTURE_2D,
                     GL_TEXTURE_MAG_FILTER,GL_LINEAR);
glTexParameteri(     GL_TEXTURE_2D,
                     GL_TEXTURE_MIN_FILTER,
                     GL_NEAREST_MIPMAP_LINEAR);
gluBuild2DMipmaps(   GL_TEXTURE_2D, GL_RGB, 64, 64,
                     GL_RGB,GL_UNSIGNED_BYTE,
                     TextureImage[i]->data);
```

Kwadryki OpenGL

- **Utworzenie kwadryki:**

```
#include <gl\glu.h>

GLUquadricObj *obj;   //zdefiniowanie wskaźnika do obiektu
obj=gluNewQuadric(); //powołanie nowego obiektu
```

- **Określenie stanu kwadryki:**
  - **Styl powierzchni:**

```
gluQuadricDrawStyle(GLUquadricObj * qobj, GLenum drawStyle);
```

Parametr drawStyle	**Opis**
GLU_FILL	Kwadryga rysowana jest przy pomocy wypełnionych pasów wielokątów.
GLU_LINE	Kwadryga rysowana jest w postaci siatki składającej się z linii
GLU_SILHOUETTE	Kwadryga rysowana jest w postaci siatki złożonej z linii, widoczne są tylko zewnętrzne krawędzie
GLU_POINT	Kwadryga rysowana jest w postaci chmury wierzchołków

Kwadryki OpenGL

- **Wektory normalne:**

```
gluQuadricNormals(GLUquadricObj *qobj,GLenum normals);
```

Parametr normals	Opis
GLU_NONE	Normalne nie są generowane
GLU_FLAT	Normalne generowane są jako prostopadłe do wielokątów tworzących powierzchnię
GLU_SMOOTH	Normalne generowane są dla każdego wierzchołka osobno w celu nadania "gładkości" powierzchniom tworzącym obiekt

- **Orientacja:**

```
gluQuadricOrientation(     GLUquadricObj * quadObject,
                           GLenum orientation);
```

Parametr orientation	Opis
GLU_OUTSIDE	Normalne skierowane są na zewnątrz
GLU_INSIDE	Normalne skierowane są do wewnątrz

Kwadryki OpenGL

- **Współrzędne tekstury:**

```
gluQuadricTexture(  GLUquadricObj * quadObject,
                    GLboolean textureCoords);
```

Parametr textureCoords	Opis
GL_TRUE	Współrzędne tekstury są tworzone
GL_FALSE	Współrzędne tekstury nie są tworzone

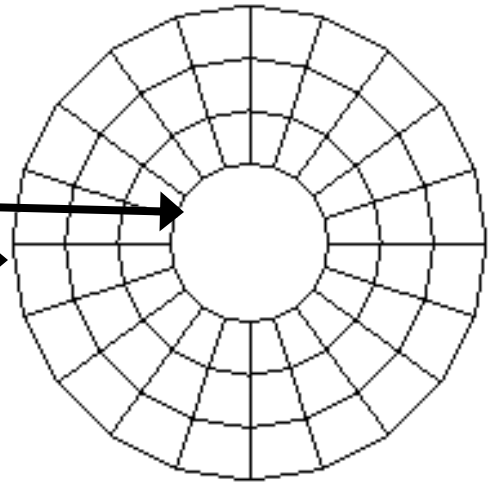
- **Usuwanie obiektów kwadryk:**

```
gluDeleteQuadric(GLUquadricObj * quadObject);
```

Kwadryki OpenGL

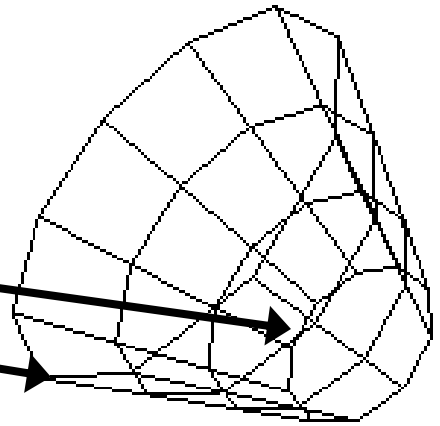
- **Dysk:**

```
void gluDisk(
  GLUquadricObj * qobj,
  GLdouble innerRadius,
  GLdouble outerRadius,
  GLint slices,
  GLint loops);
```



- **Walec:**

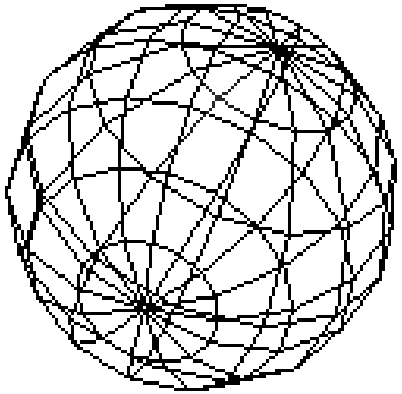
```
void gluCylinder(
  GLUquadricObj * qobj,
  GLdouble baseRadius,
  GLdouble topRadius,
  GLdouble height,
  GLint slices,
  GLint stacks
);
```



Kwadryki OpenGL

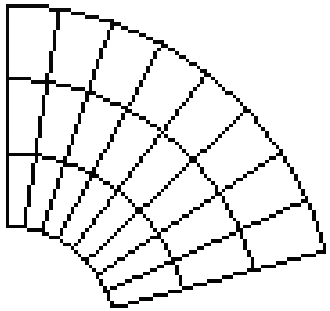
- **Kula:**

```
void gluSphere(
  GLUquadricObj * qobj,
  GLdouble radius,
  GLint slices,
  GLint stacks
);
```



- **Częściowy dysk:**

```
void gluPartialDisk(
  GLUquadricObj * qobj,
  GLdouble innerRadius,
  GLdouble outerRadius,
  GLint slices,
  GLint loops,
  GLdouble startAngle,
  GLdouble sweepAngle
);
```



Kwadryki OpenGL – przykład – oświetlony walec

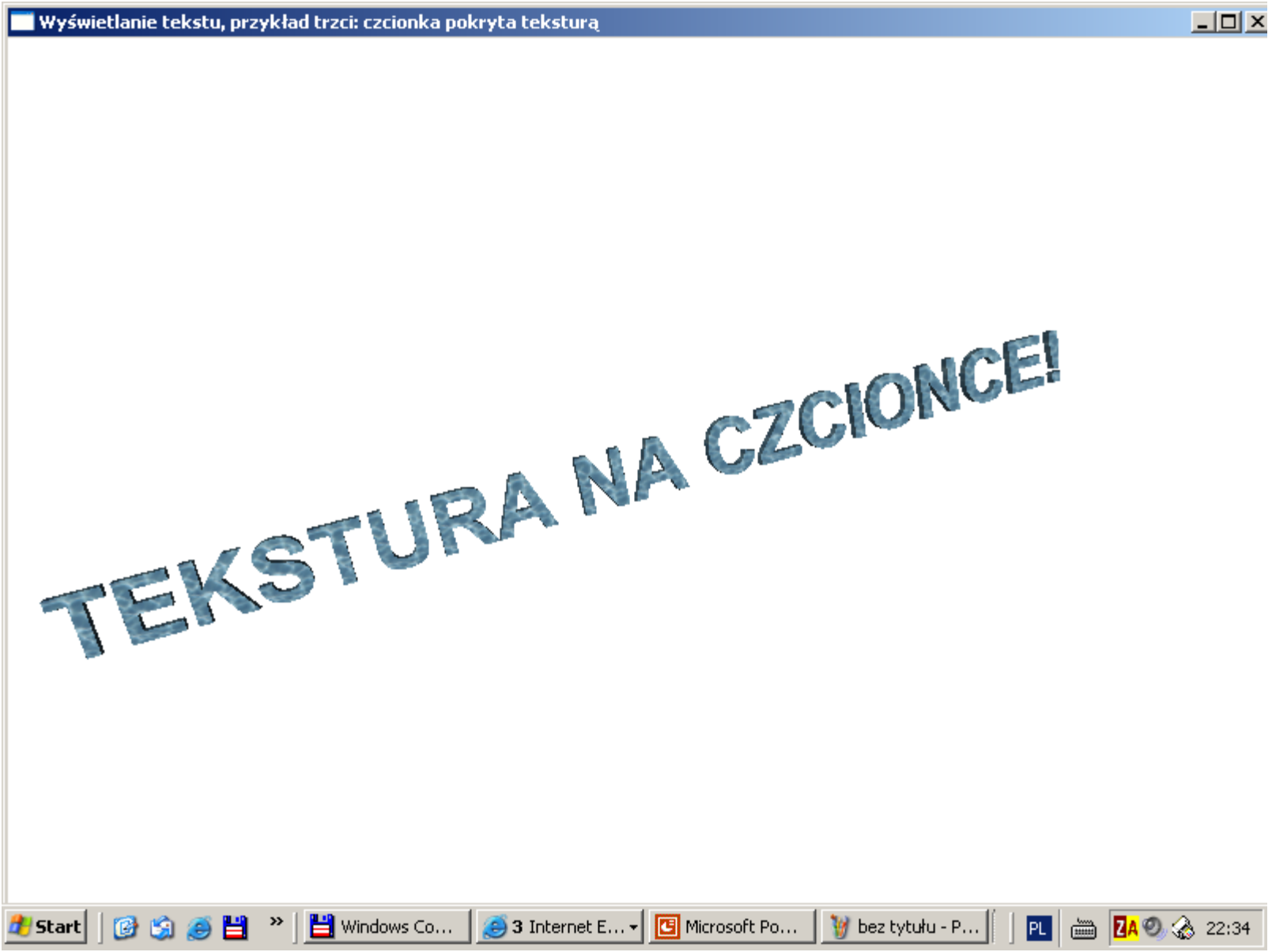
```
void walec(void)
{
        GLUquadricObj *obj;
        obj=gluNewQuadric();
        glColor3d(1,0,0);
        gluCylinder(obj,20,20,30,15,7);
}
```

Kwadryki OpenGL – przykład – teksturowana kula

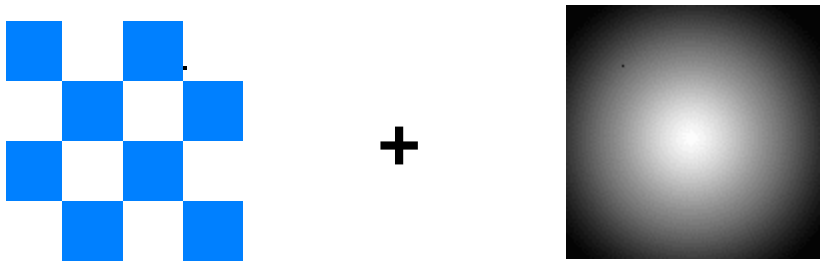
```
void kula(void)
{
 GLUquadricObj *obj;
 obj=gluNewQuadric();

 gluQuadricTexture(obj,GL_TRUE);

 glBindTexture(GL_TEXTURE_2D,texture[0]);
 glTexEnvi(   GL_TEXTURE_ENV,
              GL_TEXTURE_ENV_MODE,
              GL_MODULATE);
 glColor3d(1.0,0.8,0.8);

 glEnable(GL_TEXTURE_2D);

 gluSphere(obj,20,15,7);

 glDisable(GL_TEXTURE_2D);
}
```

Zastosowania teksturowania – odwzorowanie otoczenia

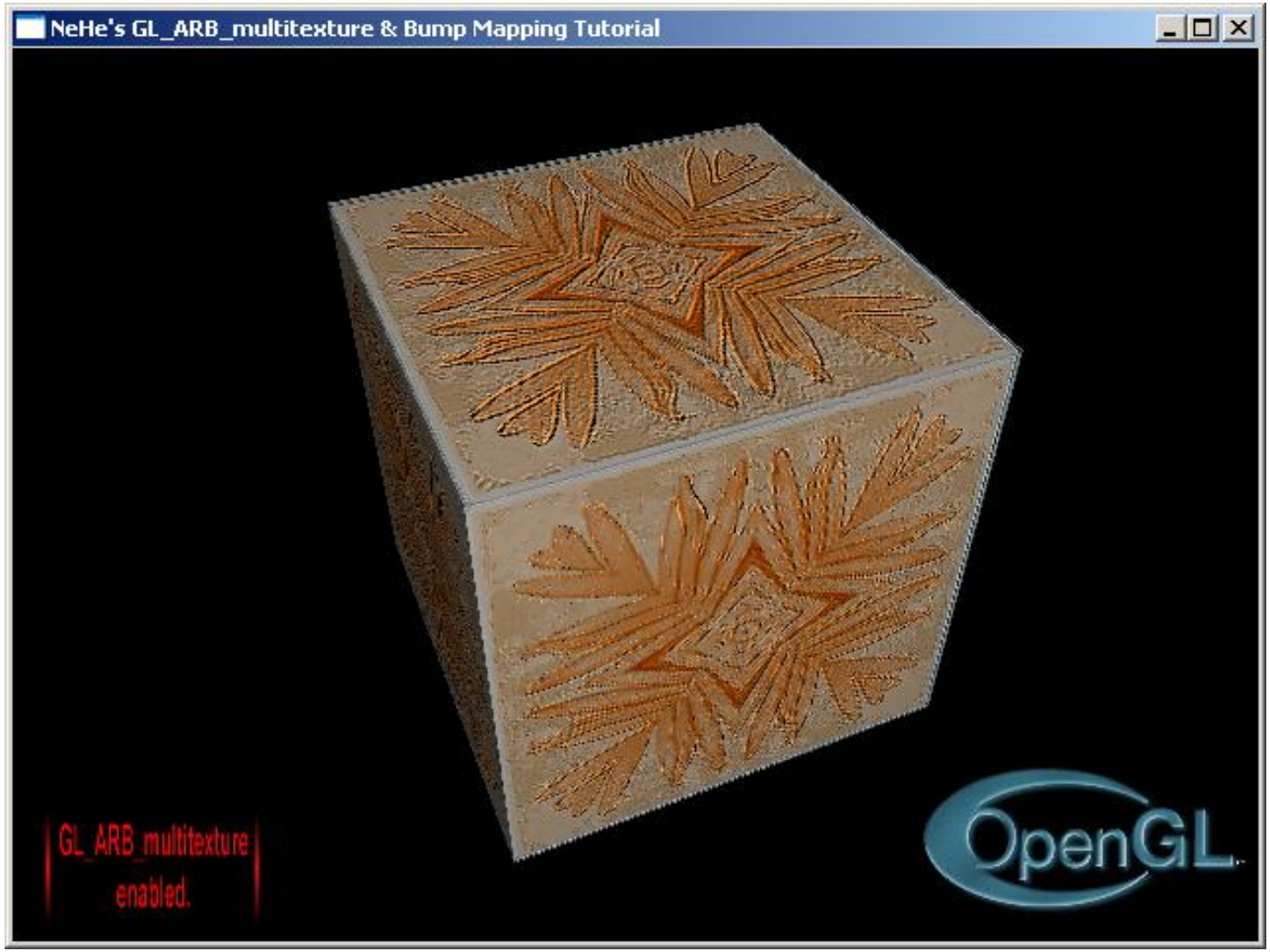
Zastosowania teksturowania – teksturowane napisy



Zastosowania teksturowania – nakładanie wielu tekstur – mapy oświetlenia



Zastosowania teksturowania – symulowanie chropowatości



Zastosowania teksturowania – mapa wysokości + „sky-box”