




Лабораторна робота №7

Тема: Апаратне прискорення растеризації тривимірного зображення OpenGL

Мета: Реалізувати каркасне тривимірних об'єктів за допомогою апаратного прискорювача OpenGL.

Теоретичні відомості

Якщо ви ще не встановили GLScene на свій Delphi або Lazarus, то спочатку встановіть його. Якщо GLScene коректно встановлений, то виконаємо дії:

1. Створюємо новий проект.
2. Подвійним клацанням по іконці компонента TGLScene  (на вкладці GLScene), додаємо GLScene1.
3. Подвійним клацанням по іконці компонента TGLSceneViewer  (на вкладці GLScene), додаємо GLSceneViewer1.
4. У об'єкта GLSceneViewer1 у властивості Align вибираємо параметр alClient, для того щоб відображається сцена розгорнулася в усі вікно.
5. Подвійним клацанням по іконці компонента TGLCadencer  (на вкладці компонентів GLScene), додаємо GLCadencer1. Він потрібен для синхронізації і обробки всіх об'єктів GLScene.
6. У об'єкта GLCadencer1 у властивості Scene вибираємо GLScene1 для того щоб GLCadencer1 функціонував в нашій сцені.
7. Подвійним клацанням по GLScene1 на нашій формі відкриваємо редактор сцени

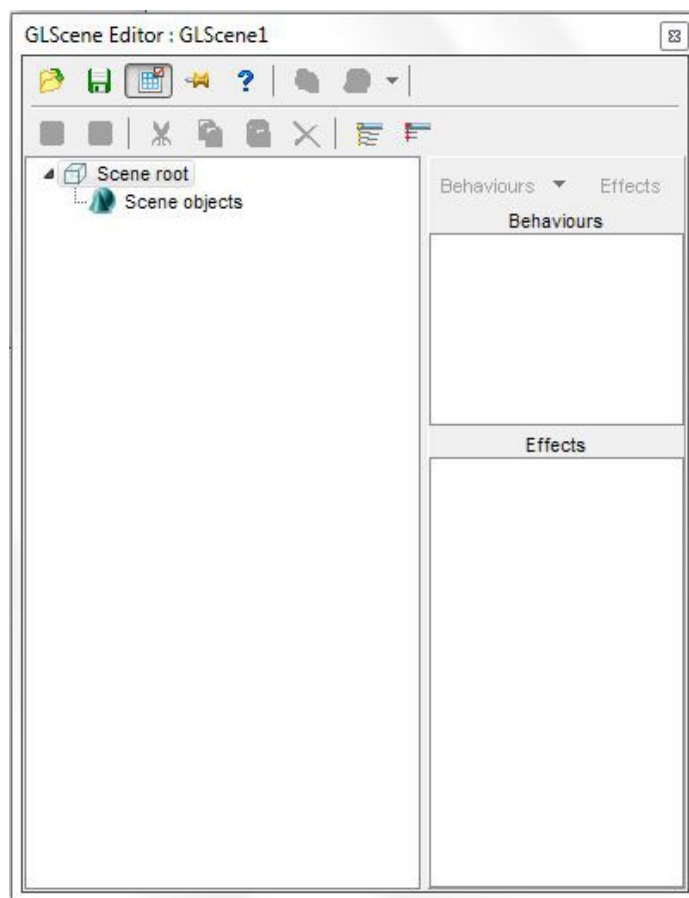


Рис. 1 — Редактор сцени

8. Тепер додамо камеру - правий клік на Scene objects в редакторі сцени, потім у випадаючому меню вибираємо Add Camera.

9. Вибираємо GLCamera1 і встановлюємо її параметри положення щодо центру сцени: Position.X: = 3, Position.Y: = 3, Position.Z: = 3.

10. Правим кліком по Scene objects в редакторі сцени - Add object - LightSource - додаємо на сцену джерело світла.

11. Вибираємо GLLightSource1 (створений раніше в пункті 10 джерело світла) і встановлюємо його параметр Position.Z: = 10 у властивостях для того щоб джерело світла був трохи віддалений від об'єкту.

12. Правим кліком по Scene objects в редакторі сцени - Add object - Basic Geometry - Cube - додаємо в середину сцени (0;0; 0) куб.

13. Вибираємо GLCamera1 (камера) і встановлюємо в її властивості TargetObject:= GLCube1, щоб наша камера була спрямована на куб.

14. Вибираємо GLSceneViewer1 і встановлюємо їй властивість Camera:= GLCamera1, щоб у вікні відображалось те, що «бачить» GLCamera1.

15. Створюємо подія Progress для GLCadencer1.

16. Додаємо в подія рядок коду:

GLCube1.TurnAngle:= GLCube1.TurnAngle + deltaTime * 100;

Цей рядок відповідає за поворот кубика навколо осі Z.

17. Тепер запустіть додаток, щоб побачити ваш перший тривимірний обертається куб.

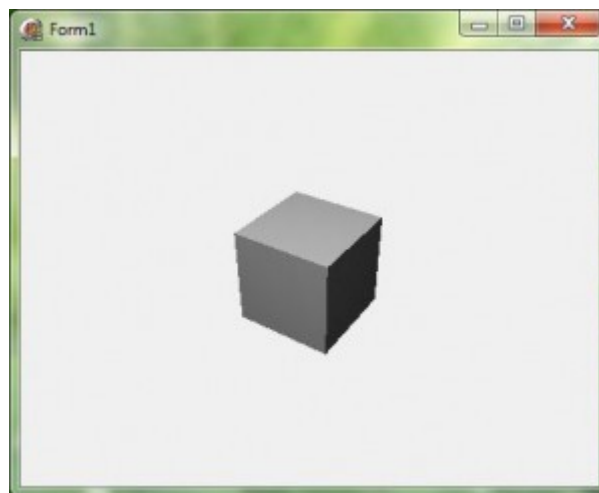


Рис. 2. - Отримане зображення куба

І все-таки обертається куб - досить тривіальний приклад. І це дійсно так з GLScene, але якщо розглянути те, що потрібно, щоб малювати як куб обертається без GLScene, то відразу стане ясно, яку роботу зробили за нас розробники цієї бібліотеки. За прикладом також варто звернути увагу, що об'єкт (куб) затінюється щодо джерела світла.

Тепер давайте поставимо перед собою завдання: нехай поруч з кубиком треба розмістити 25 сфер, по 5×5 і зафарбувати їх випадковим кольором. Створимо глобальний двовимірний масив

MySphere: array [1..5,1..5] of TGLSphere;

Тепер додамо обробник події procedure TForm1.FormCreate (Sender: TObject); і наведемо його до наступного вигляду:

```
procedure TForm1.FormCreate ( Sender  TObject );
var
  i, j: integer;
begin
  randomize;
  for i:= 1 to 5 do
```

```

begin
  for j: = 1 to 5 do
    begin
      MySphere[i,j] := TGLSphere ( GLCubel.AddNewChild( TGLSphere ) );
      // створюємо нову сферу, як дочірній об'єкт
      // елемента для GLCubel
    with MySphere[i,j] do
      begin
        Position.X := -i; // позиція по осі x
        Position.Z := -j; // позиція по осі y
        material.FrontProperties.Diffuse.RandomColor;
        // задаємо випадкову заливку
        material.FrontProperties.Ambient.RandomColor;
        // задаємо випадковий відтінок затінення на самому об'єкті
        // щодо джерела світла
      end;
    end;
  end;
end;

```



Ще один приклад демонструє завантаження об'єкта з файлу 3ds і його текстурізацію. Для роботи з текстурами, потрібно додати компонент TGLMaterialLibrary. Він знаходиться на вкладці GLScene. Створіть його. У свою властивість 'Materials' цей компонент зберігає масив матеріалів. У свою чергу кожен матеріал є сукупність текстур і різних параметрів матеріалу. Тому цей компонент і називається 'Бібліотека матеріалів'. Змінимо обробник події OnForm1Create на наведений нижче:

```

procedure TForm1.FormCreate(Sender: TObject);
var
  i, j: integer;
begin
  // читаємо текстури з файлів
  // tex1, tex2, tex3 – імена матеріалів в GLMaterialLibrary1
  GLMaterialLibrary1.AddTextureMaterial('tex1', 'tex1.jpg', true);
  GLMaterialLibrary1.AddTextureMaterial('tex2', 'tex2.jpg', true);
  GLMaterialLibrary1.AddTextureMaterial('tex3', 'tex3.jpg', true);
  for i := 1 to 5 do
    begin
      for j := 1 to 15 do
        begin
          // створюємо новий об'єкт TGLFreeForm
          MyFreeForm[i,j]:=TGLFreeForm(GLCubel.AddNewChild(TGLFreeForm));
          with MyFreeForm[i, j] do
            begin
              // задаємо нові координати об'єкту
              Position.X := i-3;
              Position.Z := -j+7;
              Position.y := sin(j);
              // завантажуюмо меш об'єкту з файлу
              LoadFromFile('obj.3ds');
              // обираємо бібліотеку матеріалів для об'єкту
            end;
          end;
        end;
      end;
    end;
  end;

```

```

material.MaterialLibrary:=GLMaterialLibrary1;
// вказуємо для об'єкту одну з текстур
if (i=1)or(j=1)or(i=5)or(j mod 5=0) then
    Material.LibMaterialName:='tex2'
else if (i=3)then Material.LibMaterialName:='tex3'
else Material.LibMaterialName:='tex1';
// повертаємо об'єкт
pitch(90);
// змінємо розміри об'єкту
Scale.X:=0.4;
Scale.y:=0.4;
Scale.z:=0.7;
end;
end;
end;
end;

```

Хід роботи

1. В графічному редакторі (рекомендується вільний до використання Blender) створіть декілька елементів графічної сцени.
2. За вказівками створіть програму для зображення створеної сцени. Елементи сцени зробіть завантаженими з файлів типу *.3ds.
3. Створіть ієрархічний об'єкт та організуйте його лінійне переміщення по сцені.
4. Для залежного елементу рухомого об'єкту створіть окреме незначне але добре помітне періодичне переміщення.
5. Прив'яжіть положення камери до рухомого об'єкту так, щоб він завжди був в центрі зображення сцени.
6. Організуйте коловий рух камери навколо рухомого об'єкту.

Контрольні питання:

1. Дайте означення поняття мешу.
2. Яке призначення компоненту TGLScene?
3. Яке призначення компоненту TGLCadencer?
4. Яке призначення компоненту TGLSceneViewer?
5. Яке призначення компоненту TGLCamera?
6. Яке призначення компоненту TGLLightSource?
7. Яке призначення компоненту TGLFreeForm?
8. Яке призначення компоненту TGLMaterial?