СРЕДСТВА ЗА СЪЗДАВАНЕ НА РЕАЛИСТИЧНИ ИЗОБРАЖЕНИЯ НА ТРИМЕРНИ ОБЕКТИ В OPENGL*

РОСИЦА П. ХРИСТОВА

TOOLS FOR CREATING REALISTIC VISUALIZATION OF THREE-DIMENSIONAL OBJECTS IN OPENGL

ROSSICA P. HRISTOVA

ABSTRACT: *The article discusses some of the standard tools of the Open Graphics Library (OpenGL) used to create realistic 3D objects and images. Types of lights and defined light sources are commented. Some examples of different ways for lighting 3D objects and some of the possibilities of light reflections are given. Different kind of fog has been considered and recommendations of how to use them are presented. This paper describes the concept of the Alpha Blending and some cases management Alpha Blending are discussed.*

KEYWORDS: *OpenGL, 3D images, lighting 3D objects, fog, Alpha Blending.*

С развитието на новите технологии компютърни изображения се виждат по вестниците, телевизията, кината, при медицинки изследвания и др. Днес компютърната графика разработва мощни инструменти за визуализация на данни в двумерното и тримерното пространство, и чрез компютърна анимации. През последното десетилетие за различни специализирани области са разработени техники за визуализация на тримерни обекти и явления (архитектурни, метерологични, медицински, биологични и т.н.), където акцента е върху реалистично изобразяване на обеми, повърхнини, светлинни отражения и други, включително и с динамичен във времето компонент.

Целта на настоящата статия е да се разгледат средствата, които предлага OpenGL за създаване на реалистични изображения на тримерни обекти. OpenGL (Open Graphics Library) [2] е приложно-програмен интерфейс (API - Application Programming Interface) за изчертаване на двумерни и тримерни графични изображения. Разработен от Silicon Graphics Inc. през 1992 г. [3], днес библиотеката се използва като индустриален стандарт, управляван от консорциума Khronos group [4].

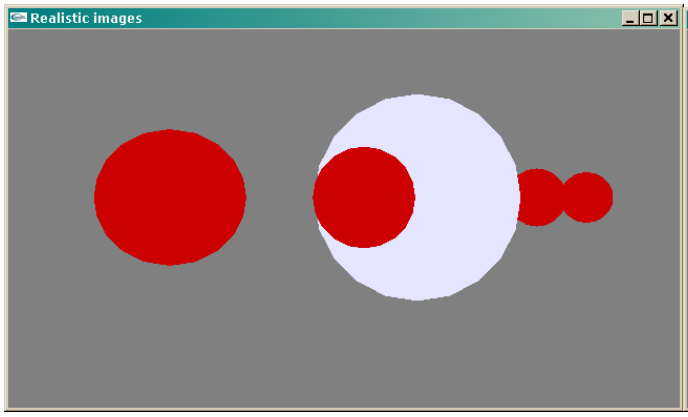
Предимствата на OpenGL са, че е много платформен графичен API, който обикновенно работи директно с графичния процесор, което прави възможна визуализацията в реално време. Използва се широко в CAD системи, виртуална реалност, научна визуализация, различни симулации и видео игри.

В статията ще бъдат разгледани само стандартните средства на OpenGL (без допълнителните библиотеки), които се използват за подобряване на 3D изображения и създаване на реалистични ефекти.

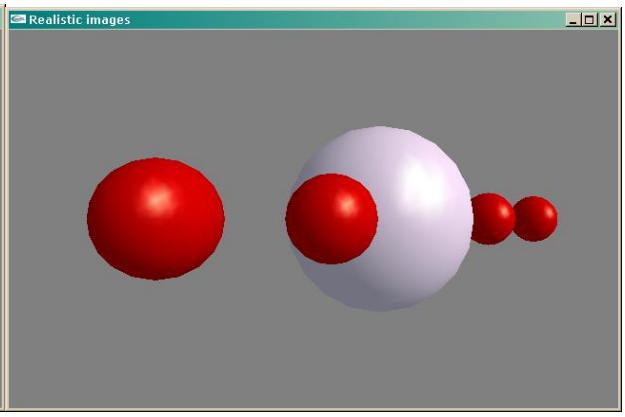
Осветяване на обекти

Едно от основните неща, които прави тримерните визуализация да изглежда реалистично, е осветлението (**Lighting**) или т.нар. динамична светлина. На фиг. 1 и фиг. 2 се вижда разликата между тримерна сцена без осветяване и същата сцена с включени светлини и отблясък. **OpenGL** приема дефиниции на светлинни източници и определения на отраженията на светлините от материалите, от които са изградени обектите, и прави всички изчисления по създаването на реалистични ефекти.

* Статията е частично финансирана по НИП № РД 05 334/12.03.2012 от ФНИ на ШУ „Епископ К.Преславски”.



Фиг. 1 Без осветяване

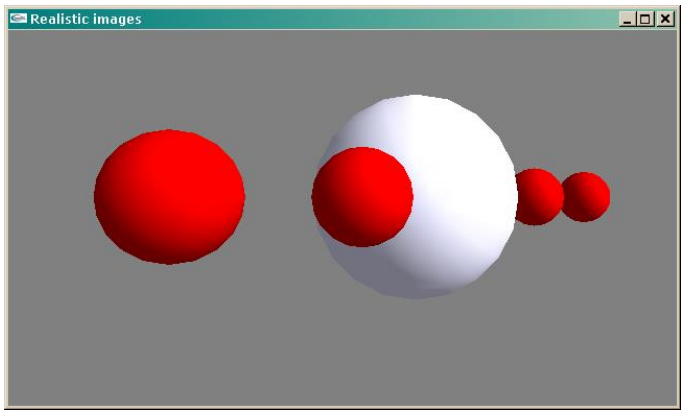


Фиг. 2 Всички светлини и отблясък

Съществуват три вида светлини:

- **Ambient** - тази светлина идва отвсякъде и се отразява навсякъде. Този вид светлина се получава, когато светлината от даден източник се отрази многократно във всички обекти. Т.е. това е глобалната светлина, която ни позволява да виждаме обектите (фиг. 3).
- **Diffuse** - тази светлина идва от определена посока и се отразява навсякъде. Пример за такава е една обикновена лампа (фиг. 4).
- **Specular** - тази светлина идва от определена посока и се отразява в определена посока. Пример за това е отблясъка на даден предмет (фиг. 6).

За да се включи осветление, трябва да се изпълнят **две стъпки. Първата** е дефиниране на източника/източниците на светлина, втората е определяне на начина, по който обектите ще отразяват даден тип светлина. Освен това, за да се използва динамичната светлина, тя трябва да бъде включена с функцията **`glEnable(GL_LIGHTING)`.**



Фиг. 3 Ambient светлина с цвят на обектите

```
float pos[]={1.5,1.,1.0,0.0};
float amb[]={0.3,0.3,0.3,1};

glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT0, GL_POSITION, pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glEnable(GL_LIGHT0);
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
glEnable(GL_COLOR_MATERIAL);
```

Програмен фрагмент за включване на източник с Ambient светлина и включен цвят на обектите.

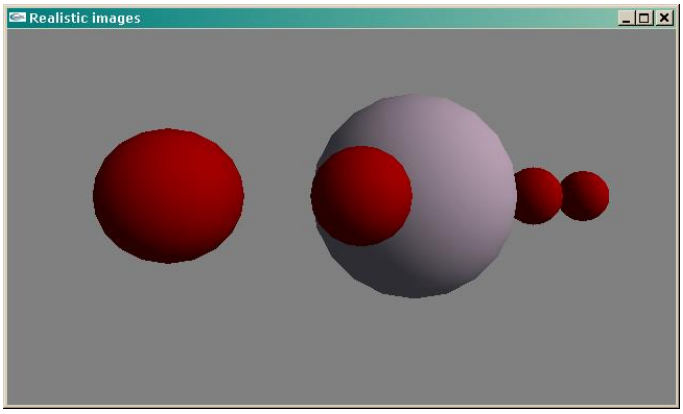
Определянето на всеки светлинен източник става с едно или няколко обръщения към функцията:

```
glLightfv(GLenum light, GLenum pname, const GLfloat *params)
```

Тя приема три аргумента:

- Първият е името на светлинния източник, което се определя чрез една от константите GL_LIGHT0, GL_LIGHT1, GL_LIGHT2, . . . GL_LIGHT6 или GL_LIGHT7. В OpenGL може да има едновременно до осем действащи светлинни източника.
- Вторият може да е една от стойностите:
  - **GL_AMBIENT** - определя Ambient излъчването на източника;
  - **GL_DIFFUSE** - определя Diffuse излъчването на източника;
  - **GL_SPECURAL** - определя Specular излъчването на източника;
  - **GL_POSITION** - определя координатите на светлинния източник.

- Третият аргумент е float масив от стойности, определящи цвят на светлината чрез RGBA модела.



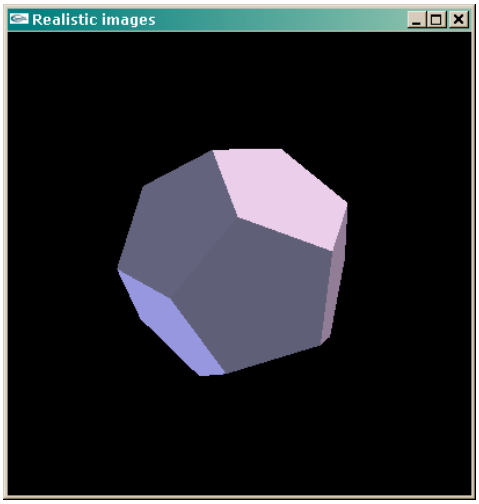

```
float pos[]={1.5,1.,1.0,0.0};
float diff[]={0.6, 0.5,0.5,1};

glEnable(GL_LIGHTING);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diff);
glLightfv(GL_LIGHT1, GL_POSITION, pos);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diff);
glEnable(GL_LIGHT1);
glEnable(GL_COLOR_MATERIAL);
```

Програмен фрагмент за включване на източник с Diffuse светлина и включен цвят на обектите.

Фиг. 4 Diffuse светлина с цвят на обектите

Светлинният източник започва да действа с извикване на функцията glEnable( ), като за параметър се подаде съответната константа.

Важно уточнение е, че един светлинен източник може да излъчва повече от един вид светина. Такъв е примерът от фиг. 2, в който източникът излъчва едновременно Ambient, Diffuse и Specular светлина. От друга страна различни светлинни източници могат да излъчват еднотипна светлина с различен цвят (фиг. 5).




```
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightfv(GL_LIGHT0, GL_POSITION, pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
glEnable(GL_LIGHT1);
glLightfv(GL_LIGHT1, GL_POSITION, pos1);
glLightfv(GL_LIGHT1, GL_DIFFUSE, diff1);
glLightfv(GL_LIGHT1, GL_SPECULAR, spec);
glMaterialfv(GL_FRONT, GL_AMBIENT, amb);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diff);
glMaterialfv(GL_FRONT, GL_DIFFUSE, diff1);
glMaterialfv(GL_FRONT, GL_SPECULAR, spec);
glMateriali(GL_FRONT, GL_SHININESS, 51);
```

Програмен фрагмент за включване на два източника с различна Diffuse светлина.

Фиг. 5 Два източника с Diffuse светлина

Втората стъпка при осветяването на обекти е определяне на начина, по който те ще отразяват даден тип светлина, от което зависи и как ще изглеждат те. Това става с една от разновидностите на следната функция:

```
glMaterialfv(GLenum face, GLenum pname, const GLfloat *params)

glMateriali(GLenum face, GLenum pname, GLfloat param)
```

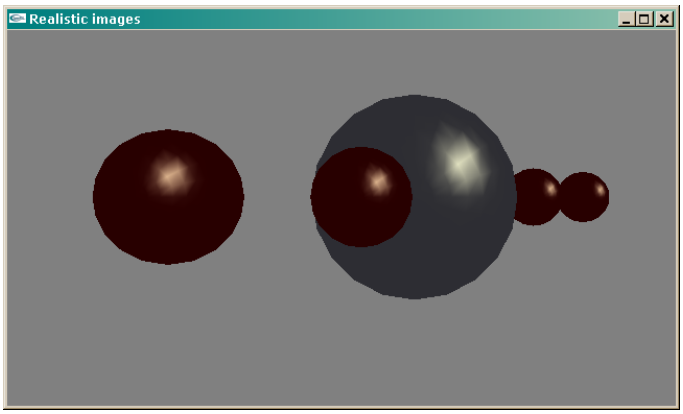
Тя приема три аргумента:

- Първият определя за коя част от обекта се отнася съответната дефиниция. Може да е една от константите GL_FRONT, GL_BACK или GL_FRONT_AND_BACK.
- Вторият аргумент може да е една от стойностите:
  - **GL_AMBIENT** - определя начина на отразяване на светлина тип Ambient;
  - **GL_DIFFUSE** - определя начина на отразяване на светлина тип Diffuse;
  - **GL_SPECULAR** - определя начина на отразяване на светлина тип Specular;
  - **GL_AMBIENT_AND_DIFFUSE** - определя еднакъв начин на отразяване за Ambient и

Diffuse светлина;

- **GL_EMISSION** - определя излъчваната от обекта светлина, която се отразява единствено на цвета на обекта и може да се използва за създаване например на лампа.
- **GL_SHININESS** - определя големината на най-яркото отражение, причинено от светлина тип Specular. Единствено с тази стойност се използва glMateriali( ), тогава третия аргумент е число от 0 до 128. Колкото по-голямо е то, толкова по-точно фокусирано и по-малко е светлинното петно.

♦ Третият аргумент на glMaterialfv( ), е float масив от стойности, определящи RGB цвят на съответната светлина.

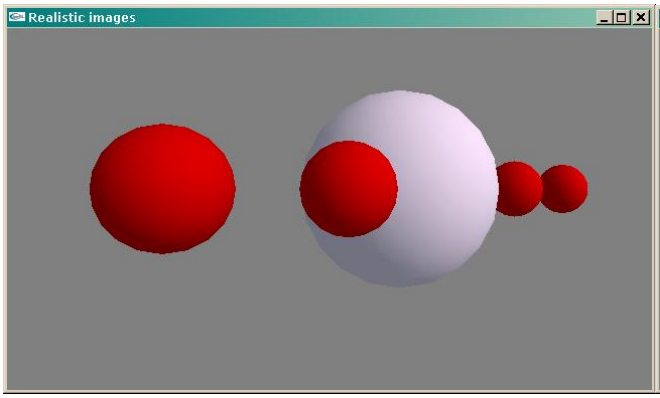



```
float pos[]={1.5,1.,1.0,0.0};
float spec[]={0.9,0.9,0.8,1};
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT2);
glLightfv(GL_LIGHT2, GL_SPECULAR, spec);
glLightfv(GL_LIGHT2, GL_POSITION, pos);
glMaterialfv(GL_FRONT, GL_SPECULAR,spec);
glMateriali(GL_FRONT, GL_SHININESS, 64);
glEnable(GL_COLOR_MATERIAL);
```

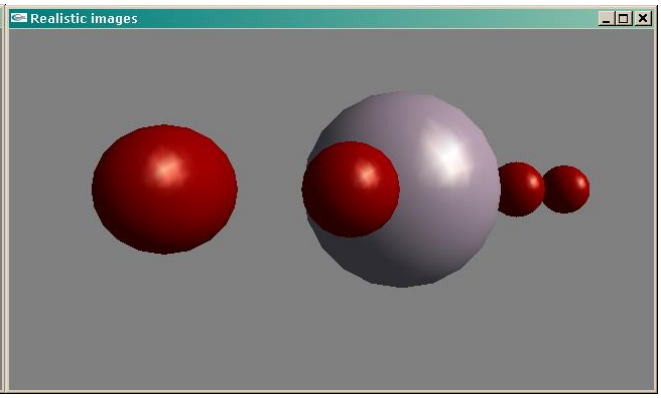
Програмен фрагмент за включване на източник със Specular светлина, отблясък и включен цвят.

Фиг. 6 Specular светлина с отблясък и цвят на обектите

Всъщност дали обектът ще изглежда грапав, гладък или полиран зависи от т.нар. отблясък, който се задава със Specular светлина, яркост на отражението и големина на светлинното петно. Демонстрация за значението на този вид характеристики е сравнението на фиг. 7 (няма дефинирана Specular светлина) и фиг. 8 (има Specular светлина със зададено светлинно петно).



Фиг. 7 Само Ambient и Diffuse светлина



Фиг. 8 Diffuse и Specular светлина със shininess

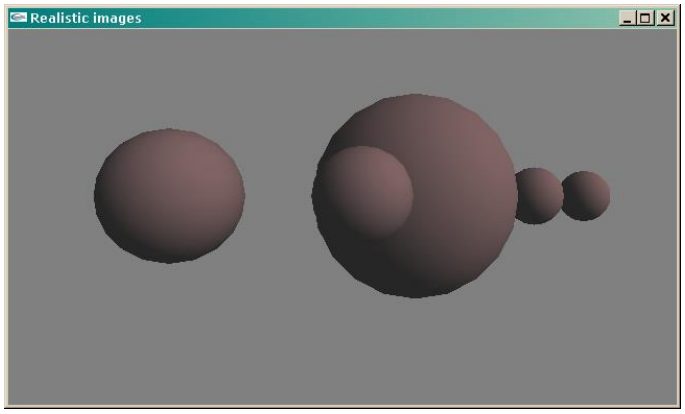
Ако обектите са били с предварително дефиниран цвят, то при включване на осветление техният цвят не се взема предвид (фиг. 9), а се генерира нов цвят в зависимост от светлинните източници и начина на отразяване на светлината. В примерите в статията предварителния цвят на обектите взема участие в създаването на крайния цвят. Това става с извикване на функцията **`glEnable(GL_COLOR_MATERIAL).`**

Друг начин за включване на началният цвят на обекта при включване на осветление е чрез дадената по-долу функция, която определя за кои видове светлина да се взема предвид началният цвят.

```
glColorMaterial( GLenum face, GLenum mode)
```

Първият аргумент определя за коя част от обекта се отнася съответната дефиниция, а вторият определя светлината, при която ще влиза в сила и цветът на обекта, т.е. може да е

GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT_AND_DIFFUSE или GL_EMISSION.



Фиг. 9 Само осветление без цвят на обектите

Препоръки:

1. Препоръчва се цветът на различните видове светлина да е съставен от трите компоненти на RGB модела т.е. да се основава на някакъв нюанс на сивия цвят.

2. В случай, че се искаме обектът да има отблясък, препоръчително е цветът на Specular светлината да е значително по-светъл от цвета на обекта и/или Ambient светлината.

3. В случай, че се искаме да има обща Ambient светлина, е добре цветът и да е тъмен.

4. За постигане на по-добър резултат в някои ситуации е по-добре да се използва един или два вида светлина.

5. Различни светлинни източници могат да излъчват еднотипна светлина с различен цвят.

6. Светлинните източници могат да се движат.

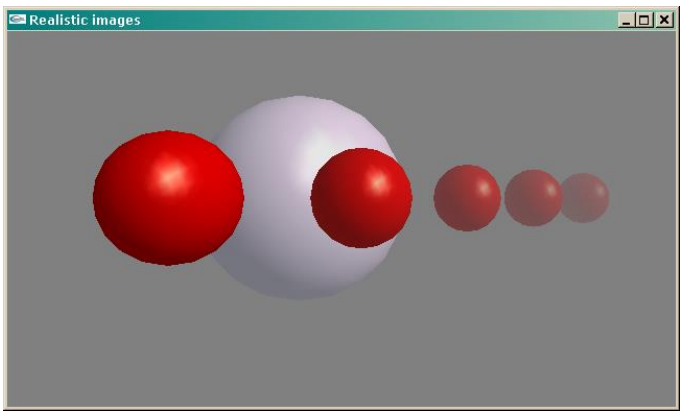
Замъгляване

OpenGL предоставя функцията **glFog( )**, която може да създаде реалистична мъгла или да направи по-далечните обекти по-бледи. Функцията има няколко разновидности, всяка от които има определено предназначение. С първия вариант се определя типа „мъгла”, които ще се използва. Видът е:

```
glFogi(GL_FOG_MODE, GLint param)
```

За втори аргумент може да се използва:

- **GL_LINEAR** – задава стандартна линейна мъгла, за която се определя откъде започва и къде обектите стават невидими (фиг. 10).
- **GL_EXP** – задава т.нар. експоненциална мъгла (фиг. 11а и 11б), която има плътност и се разпространява навсякъде, стартирайки от точката на наблюдателя.
- **GL_EXP2** – подобна на GL_EXP, но по-силна от нея (фиг. 12).



Фиг. 10 Мъгла от тип GL_LINEAR

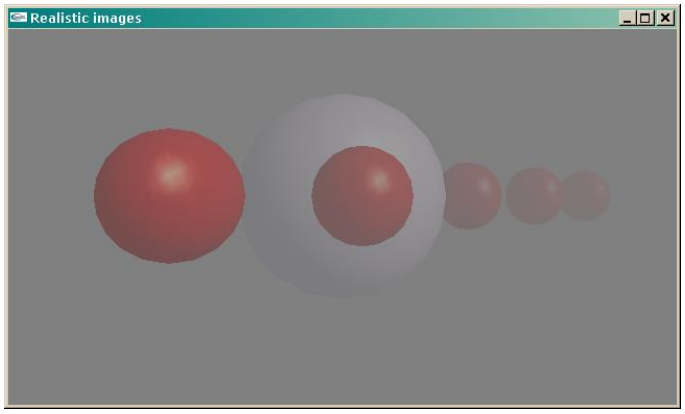
```
float colfog[]={0.5,0.5,0.5,1};

void foggy()
{
   glEnable(GL_FOG);
   glFogfv(GL_FOG_COLOR, colfog);
   glFogf(GL_FOG_START, 3);
   glFogf(GL_FOG_END, 9);
   glFogi(GL_FOG_MODE, GL_LINEAR);
}
```

Програмен фрагмент, определящ линейна мъгла.

Вторият вариант на функцията определя цвета на мъглата:

`glFogfv(GL_FOG_COLOR, const GLfloat *param )`

Тук втори аргумент е името на float масива, в който се съхранява **RGB** цвета на мъглата. За постигане на реалистична мъгла цветът на фона трябва да е еднакъв с цвета на мъглата или много близък.



Фиг. 11а Мъгла от тип GL_EXP с пътност 0.35

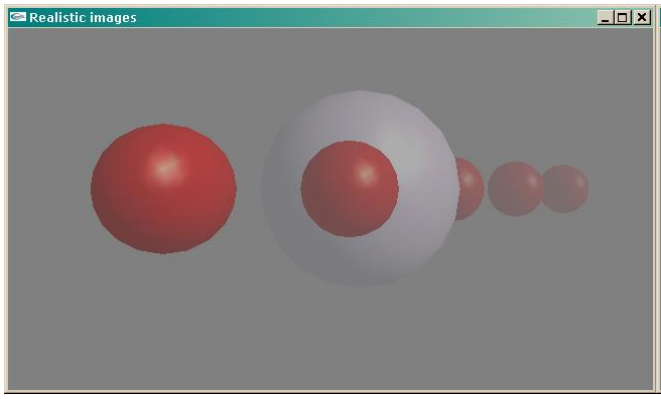
```
float colfog[]={0.5,0.5,0.5,1};

void foggy()
{
   glFogi(GL_FOG_MODE, GL_EXP);
   glFogfv(GL_FOG_COLOR, colfog);
   glFogf(GL_FOG_DENSITY, 0.35);
   glEnable(GL_FOG);
}
```

Програмен фрагмент, определящ мъгла тип Exp.

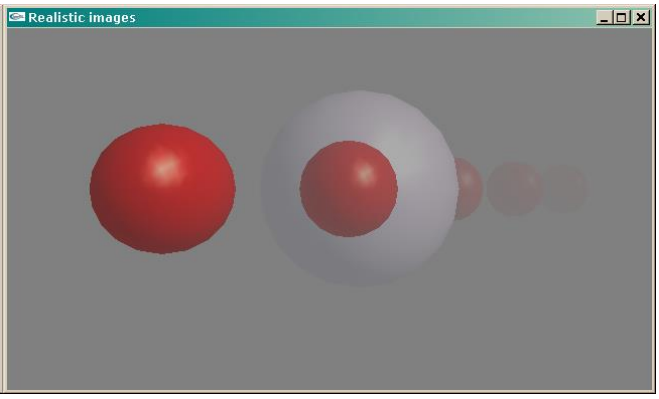
Друг вариант на функцията задава допълнителни характеристики на мъглата:

`void glFogf(GLenum pname, GLfloat param)`

Ако мъглата е тип **GL_LINEAR** се определя, откъде (на колко единици от камерата) започва мъглата и след колко единици обектите да са напълно скрити в нея.



Фиг. 11б Мъгла от тип GL_EXP с пътност 0.25



Фиг. 12 Мъгла от тип GL_EXP2 с пътност 0.25

За първи аргумент на функцията се задава съответно **GL_FOG_START** или **GL_FOG_END**, а за втори аргумент на функцията определената числова стойност.

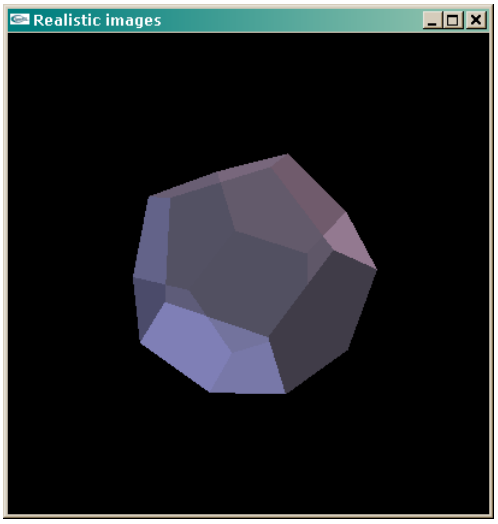
Ако се използва мъгла тип **GL_EXP** или **GL_EXP2** е нужно да се определи плътност на мъглата. Тогава първи аргумент на функцията е **GL_FOG_DENSITY**, а втори аргумент - стойност в интервала от 0.0 до 1.0.

Мъглата започва да действа с стартиране на функцията **`glEnable(GL_FOG).`**

Препоръки:

1. За постигане на реалистична мъгла цветът на фона трябва да е еднакъв или много близък с цвета на мъглата, защото мъглата се отразява само на цвета на обектите.
2. За постигане на по-добър разултат е добре пространството непосредствено пред камерата да се оставя без мъгла.

Полупрозрачност



Фиг. 13 Dodecahedron с alpha 0.6

В тримерната графика голяма част от добрите ефекти са реализирани благодарение на полупрозрач-остта (**Alpha Blending)**. Това е технология, при която цветовете на по-близо стоящия и по-далечния обект се смесват, за да се получи един среден цвят, създаващ ефект на прозрачност на предния обект. Тук се намесва четвъртата компонента от **RGBA** модела, която се използва за установяване на степен на прозрачност.

За да се активира полупрозрачност не е достатъчно само задаването на подходящ четвърти аргумент във функцията **glColor4f( )**. Трябва да се активира **Alpha Blending** с функцията **`glEnable(GL_BLEND)`** и да се определи начина, по който се смесват цветовете на предния и задния обект.

Формулата, по която се изчислява полупрозрачността в **OpenGL** е:

```
( Rs*Sr+Rd*Dr, Gs*Sg+Gd*Dg, Bs*Sb+Bd*Db, As*Sa+Ad*Da ),
```

където **Rs**, **Gs**, **Bs** и **As** са съответно **RGBA** стойностите на цвета на предния обект, които се задават с **glColor4f( ),** а **Rd**, **Gd**, **Bd** и **Ad** са съответно **RGBA** стойностите на задния обект.

Sr, **Sg**, **Sb**, **Sa**, както и **Dr**, **Dg**, **Db**, **Da** са т.нар. фактори на полупрозрачност, от които зависи как ще се смесят цветовете на двата обекта. Благодарение на тях може да се възпроизведат, най-различни ефекти на полупрозрачност. Първата поредица от четири аргумента се отнася за предния обект, а втората за задния обект.

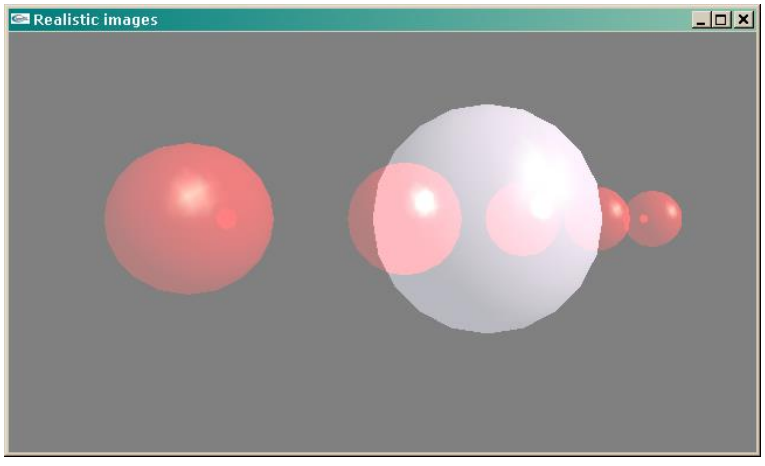
Тези фактори са скрити в параметрите на функцията:

```
glBlendFunc(GLenum sfactor, GLenum dfactor),
```

която приема два аргумента, от които зависи какви стойности ще получат факторите на полупрозрачност съответно за предния и задния обект. Ето някои от възможните аргументи:

GL_ZERO - цветът на обекта не се взима предвид. За стойности на факторите на полупрозрачност, на който и да е от двата обекта се задават стойностите (0, 0, 0, 0), т.е. обектът става невидим. Може да се използва както за преден, така и за заден обект.

GL_ONE - използва се цвета на обекта. За стойности на факторите на полупрозрачност, на който и да е от двата обекта се задават стойностите (1, 1, 1, 1), т.е. обектът става непрозрачен. Също няма значение за кой от двата обекта се определя фактор на полупрозрачност.



Фиг. 14

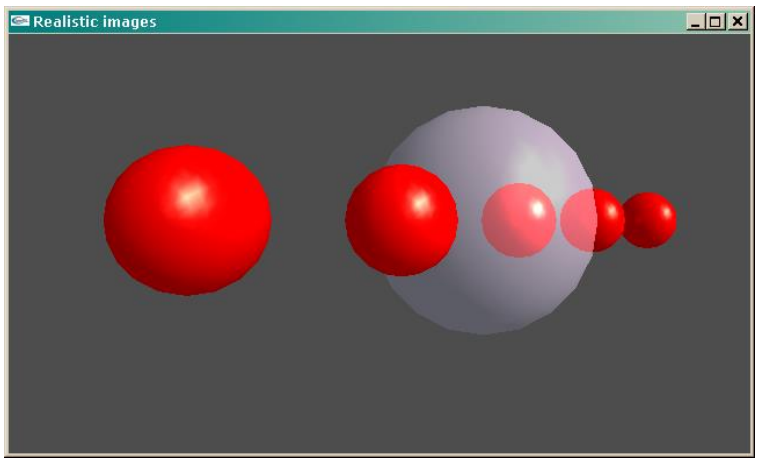
GL_DST_COLOR - цветът на обекта се умножава по този на задния обект. Може да се

използва единствено за предния обект, т.е. само като първи аргумент на функцията. За фактори на полупрозрачност на предния обект се предават стойностите **Rd**, **Gd**, **Bd** и **Ad,** т.е. RGBA цвета на задния обект. Резултатът е избледняване на по-близкия обект, тъй като всички стойности в RGBA модела на **OpenGL** са между 0 и 1.

На фиг. 14 полупрозрачна е бялата сфера (коефицент alpha 0.5), а обръщението към функцията glBlendFunc има вида:

```
glBlendFunc(GL_DST_COLOR,GL_DST_ALPHA);
```

GL_SRC_COLOR - цветът на обекта се умножава по този на предния обект. Може да се използва единствено за задния обект, т.е. само като втори аргумент на функцията. За фактори на полупрозрачност на задния обект се предават стойностите **Rs**, **Gs**, **Bs** и **As** т.е. RGBA цвета на предния обект. Резултатът е избледняване на по-далечния обект.



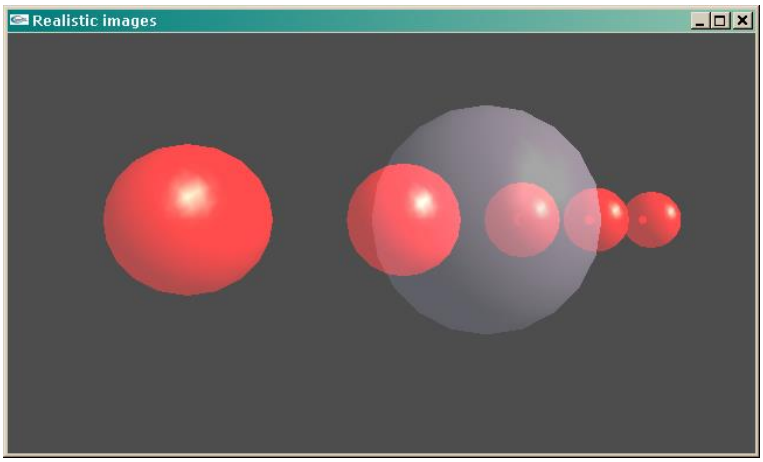
Фиг. 15

На фиг. 15 полупрозрачна е бялата сфера (коефицент alpha 0.5), а обръщението към функцията glBlendFunc има вида:

```
glBlendFunc(GL_SRC_ALPHA,GL_SRC_COLOR);
```

GL_ONE_MINUS_DST_COLOR - цветът на обекта се умножава с (1, 1, 1, 1) минус цвета на задния обект (**Rd**, **Gd**, **Bd**, **Ad**). Може да се използва само като **sfactor** и реално факторите за полупрозрачност на предния обект получават стойности, които са обратно пропорционални на цвета на задния обект.

GL_ONE_MINUS_SRC_ALPHA - цветът на обекта се умножава с (1, 1, 1, 1) минус цвета на предния обект (**Rs**, **Gs**, **Bs**, **As**). Може да се използва само за аргумент на **dfactor**. Съответно тук за факторите на полупрозрачност на задния обект се получават стойности, които са обратно пропорционални на цвета на предния обект.



Фиг. 16

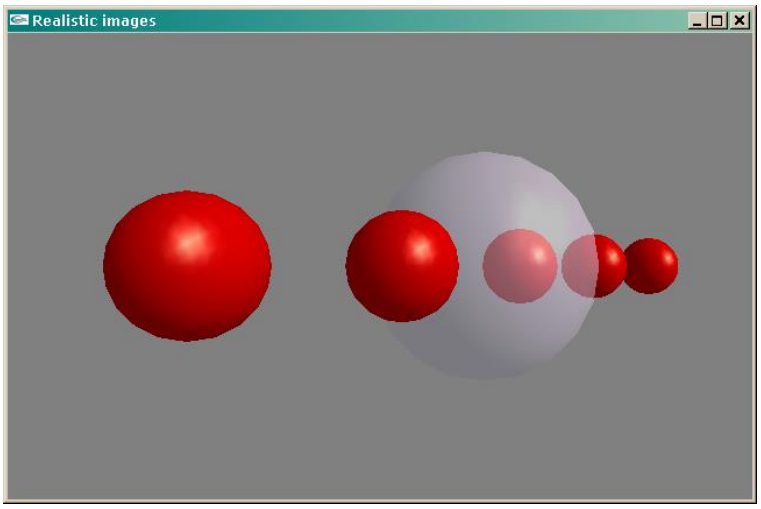
На фиг. 16 полупрозрачна е бялата сфера (коефицент alpha 0.5), а обръщението към функцията glBlendFunc има вида:

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

GL_SRC_ALPHA – цветът, на който и да е от двата обекта (може да се използва, както за **sfactor**, така и за **dfactor**) се умножава с четвъртата стойност от цвета на предния обект, т.е. за фактори на полупрозрачност обектът получава стойностите (**As**, **As**, **As**, **As**).

GL_DST_ALPHA - същото с изключение на това, че за фактори на полупрозрачност обектът получава четвъртата Alpha стойност от цвета на задния обект (**Ad**, **Ad**, **Ad**, **Ad**).

GL_ONE_MINUS_DST_ALPHA - цветът на обекта се умножава с (1, 1, 1, 1) минус четвъртата Alpha стойност от цвета на задния обект), т.е. за фактор на полупрозрачност се получават стойностите (1, 1, 1, 1) - (**Ad**, **Ad**, **Ad**, **Ad**). Може да се използва както за предния, така и за задния обект.



Фиг. 17

На фиг. 17 полупрозрачна е бялата сфера (коефицент alpha 0.5), а обръщението към функцията glBlendFunc има вида:

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

GL_ONE_MINUS_SRC_ALPHA - същото като по-горе с изключение на това, че се използва Alpha стойност от цвета на предния обект, т.е. за фактор на полупрозрачност се приемат стойностите (1, 1, 1, 1) - (**As**, **As**, **As**, **As**).

Важно е да се отбележи, че ако се използва полупрозрачност трябва първо да се изрисува задния обект и след това този отпред.

Препоръки:

1. При включване на **Alpha Blending** трябва да се задава () за цвета на всички обекти, като за плътните обекти стойността на Alpha е 1, а за полупрозрачните обекти е число по-малко от 1.
2. При включване на **Alpha Blending** цветът на фона обикновено има Alpha стойност 1 и участва във формулата за полупрозрачност като цвят на най-далечния обект.
3. При използване на полупрозрачност трябва първо да се изрисуват по-далечните обекти, а след това по-близкостоящите, за които е разрешена полупрозрачност.

Изглаждане

При изобразяване на екрана в компютърната графика, линиите и краищата на самите обекти не са съвсем гладки и при по-добро вглеждане няма как да не се забележи назъбване, особено по диагоналните линии. Начупването има своето обяснение - екранът се състои от пиксели, които представляват миниатюрни квадратчета. На помощ идва технологията **Antialiasing**, която се изразява в следното: при изобразяването на линии се използват по-голям брой пиксели като различното е, че за крайните пикселни единици се задава полупрозрачност. Така окото ни не може да разграничи ясно начупването на линията и като ефект се получава

гладка линия. Изглаждането на краищата на запълнени обекти е доста по-трудно и вместо него се използва пълноекранно изглаждане на всичко по сцената.

За изглаждане на линиите е нужно да се изпълнят функциите:

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

Направените тук демонстрации са с готови обекти, за които нормалните вектори са добре изчислени. За нестандартни обекти, които потребителя сам ще моделира и създаде, за да се постигне реалистична визуализацияа, трябва да са коректно дефинирани нормалите на всички части на обектите, защото сложните обекти се състоят от отделни парчета, които трябва да бъдат добре съединени.

ЛИТЕРАТУРА

1. The Red Book OpenGL Programming Guide, http://fly.cc.fer.hr/~unreal/theredbook/
2. URL: http://www.opengl.org/
3. URL: http://www.opengl.org/registry/doc/glspec10.pdf
4. URL: http://www.khronos.org/news/press/khronos-releases-opengl-4.3-specification-with-major-enhancements
5. URL: http://www.opengl.org/wiki/Code_Resources
6. URL: http://www.opengl.org/sdk/docs/man2
7. URL: http://virtualreality.headoff.com/