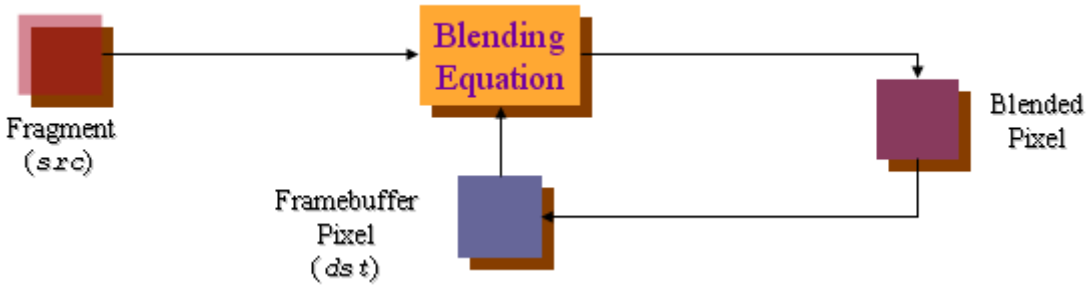
الجلسة الثامنة

الدمج

- عندما يتم تمكين الدمج، تستخدم قيمة ألفا لدمج قيمة اللون الجديدة لبكسل معطى (مرحلة Fragment) مع لون البكسل الموجود مسبقاً والمخزن في framebuffer. أفضل طريقة لفهم الدمج بالنظر إلى المركبة RGB بأنها تمثل اللون وقيمة المركبة ألفا تمثل درجة الشفافية (A=0 شفاف ، A=1 كتيم).
- عند النظر إلى جسم عبر زجاج أخضر، اللون الناتج يكون لون الجسم بالإضافة إلى دمجه مع اللون الأخضر ونسبة الدمج تختلف اعتماداً على خواص البث للزجاج فإذا كان الزجاج يرسل 80% من الضوء الواقع عليه (له درجة كمد 20%) فاللون الذي نراه هو مزيج 20% من لون الزجاج و 80% من لون الجسم.



عوامل المصدر والهدف

- إذا كانت عوامل دمج المصدر والهدف هي: (Sr,Sg,Sb,Sa) للمصدر (القيم اللونية للبكسلات الجديدة) و (Dr,Dg,Db,Da) للهدف (القيم اللونية للبكسلات الموجودة مسبقاً) وقيم RGBA للمصدر والهدف تدل عليها اللاحقة s و d عندها فإن قيمة RGBA المدمجة النهائية هي:

(RsSr+RdDr, GsSg+GdDg, BsSb+BdDb, AsSa+AdDa)

- وكل مركبة تصبح داخل المجال [0,1]
- يمكن استخدام التابع glBlendFunc() لتحديد معاملات المصدر والهدف ولكن يجب تمكين الدمج أولاً عن طريق glEnable(GL_BLEND) ولإلغاء تفعيل الدمج glDisable(GL_BLEND).

الشكل العام لأمر الدمج

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

- يتحكم بكيفية دمج لون fragment مع اللون الموجود في framebuffer.
  - *sfactor* يدل على كيفية حساب معامل دمج المصدر .
  - *dfactor* يدل على كيفية حساب معامل دمج الهدف

- القيمة الافتراضية لـ sfactor هي GL_ONE و dfactor هي GL_ZERO وتعطي نفس النتيجة في حال عدم تفعيل الدمج.
- القيم الممكنة لهذه المعاملات موجودة في الجدول التالي:

Constant	Relevant Factor	Computed Blend Factor
GL_ZERO	source or destination	(0, 0, 0, 0)
GL_ONE	source or destination	(1, 1, 1, 1)
GL_DST_COLOR	source	(Rd, Gd, Bd, Ad)
GL_SRC_COLOR	destination	(Rs, Gs, Bs, As)
GL_ONE_MINUS_DST_COLOR	source	(1, 1, 1, 1)-(Rd, Gd, Bd, Ad)
GL_ONE_MINUS_SRC_COLOR	destination	(1, 1, 1, 1)-(Rs, Gs, Bs, As)
GL_SRC_ALPHA	source or destination	(As, As, As, As)
GL_ONE_MINUS_SRC_ALPH A	source or destination	(1, 1, 1, 1)-(As, As, As, As)
GL_DST_ALPHA	source or destination	(Ad, Ad, Ad, Ad)
GL_ONE_MINUS_DST_ALPH A	source or destination	(1, 1, 1, 1)-(Ad, Ad, Ad, Ad)
GL_SRC_ALPHA_SATURATE	source	(f, f, f, 1); f=min(As, 1-Ad)

نماذج عن استخدام الدمج

- ليست كل التركيبات من معاملات المصدر والهدف لها معنى، أغلب التطبيقات تستخدم عدد صغير من التركيبات.

أمثلة:

- إحدى الطرق لرسم رسمة مؤلف نصفها من صورة والنصف الآخر من صورة ثانية بدمج متساوي هي بجعل معامل المصدر GL_ONE ورسم الصورة الأولى ثم جعل معاملي المصدر والهدف GL_SRC_ALPHA ثم رسم الصورة الثانية مع قيمة ألفا 0.5.
- إذا طُلب دمج 0.75 من الصورة الأولى مع 0.25 من الثانية: ارسم الصورة الأولى كما هو سابقاً ثم ارسم الصورة الثانية مع قيمة ألفا 0.25 لكن مع المعاملات GL_SRC_ALPHA للمصدر و GL_ONE_MINUS_SRC_ALPHA للهدف.

مثال عن الدمج:

- يرسم المثال التالي أربعة مستطيلات متداخلة ملونة. كل منها بقيمة ألفا 0.75 بحيث أن المستطيلان السفلي اليساري (سماوي مع أصفر أصلي) والعلوي اليميني (أصفر مع سماوي أصلي) من الإطار يتم تغطيتهما مرتين. نعدل فقط التابع redraw ليصبح:

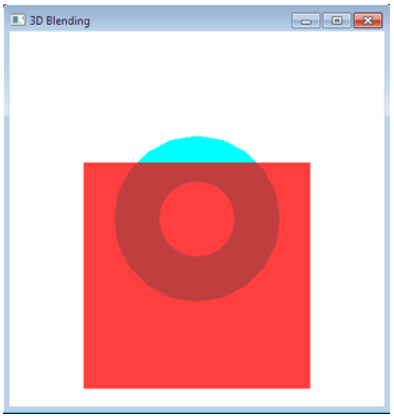
```
static void redraw(void)
{
   glClearColor(1.0,1.0,1.0,0.0);//خلفية بيضاء شفافة
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(-5.0f,-5.0f,-20.0);
   glEnable(GL_BLEND);//تفعيل المزج
   glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);//اختيار معامل المزج للمصدر والهدف
   glShadeModel(GL_FLAT);
   glColor4f(1.0, 1.0, 0.0, 1.0);// أصفر بدون شفافية
   glRectf(0.0, 0.0, 5.0, 10.0);
   glColor4f(0.0, 1.0, 1.0, 0.75);//0.75 سماوي مع شفافية
   glRectf(0.0, 0.0, 10.0, 5.0);
/* draw colored polygons in reverse order in upper right */
   glColor4f (0.0, 1.0, 1.0, 1.0);
   glRectf (5.0, 5.0, 10.0, 10.0);
   glColor4f (1.0, 1.0, 0.0, 0.75);
   glRectf (5.0, 5.0, 10.0, 10.0);
   glutSwapBuffers();

}
```

الدمج ثلاثي الأبعاد مع بفر العمق

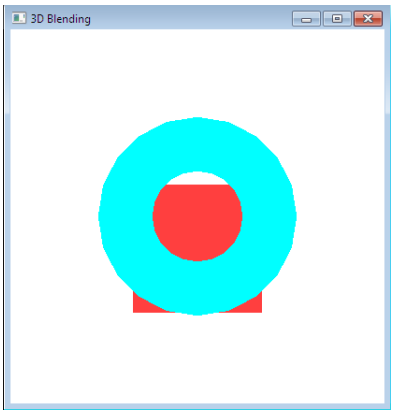
- ترتيب رسم المضلعات يؤثر على نتيجة الدمج. وعند رسم الأجسام شبه الشفافة ثلاثية البعد، سنحصل على عدة أشكال اعتماداً على رسم المضلعات من الخلف إلى الأمام أو من الأمام إلى الخلف. كما يجب أن نأخذ بعين الاعتبار تأثير بفر العمق عند تحديد الترتيب الصحيح.نريد استخدام بفر العمق لإزالة أجزاء الأجسام المخفية والتي تقع خلف الأجسام المعتمة.
- إذا وقع جسم معتم أمام جسم شبه شفاف أو جسم معتم آخر فإننا نريد من بفر العمق أن يزيل الأجسام الأكثر بعداً. لكن إذا كان جسم شبه شفاف أقرب فإننا نريده أن يدمج مع الجسم المعتم .
- يمكن معرفة الترتيب الصحيح لرسم الأجسام إذا كان المشهد ثابت لكن هذا يصبح صعب للغاية إذا كان viewport أو الجسم متحرك. الحل هو بتفعيل بفر العمق ولكن جعله read-only عند رسم الأجسام شبه الشفافة.
- أولاً يجب رسم كل الأجسام المعتمة (غير الشفافة) مع بفر العمق في الحالة العادية. ثم نقوم بحفظ قيم العمق هذه بجعل بفر العمق read-only. تستخدم التعليمة glDepthMask() مع القيمة GL_FALSE لجعل البفر read-only .

مثال:

```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
bool eyePosition;
static void redraw(void);
```




```
int main(int argc, char **argv);
void keyboard ( unsigned char key, int x, int y );
int main(int argc, char **argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowPosition(100,100);
glutInitWindowSize(400,400);
glutCreateWindow("3D Blending");
glutDisplayFunc(redraw);
glMatrixMode(GL_PROJECTION);
gluPerspective(45,1.0,10.0,200.0);
glMatrixMode(GL_MODELVIEW);
glutMainLoop();
return 0;
}
static void redraw(void)
{
glClearColor(1.0,1.0,1.0,0.0);
glEnable(GL_DEPTH_TEST);
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();
glPushMatrix();
if (eyePosition)
gluLookAt(0.0, 0.0, 10.0, 0.0, 0.0, 0.0, 0.0,1.0, 0.0);
else
gluLookAt(0.0, 0.0, -100.0, 0.0, 0.0, 0.0, 0.0,1.0, 0.0);
glPushMatrix();
glTranslatef(0.0, 0.0, -40.0);
glColor4f(0.0,1.0,1.0,1.0);
glutSolidTorus(3,8,10,20);
glPopMatrix();
glEnable(GL_BLEND);
glDepthMask(GL_FALSE);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
glTranslatef(0.0, -5.0, -60.0);
glColor4f(1.0, 0.0, 0.0, 0.75);
glRecti (-10,-10,10,10);
glDepthMask(GL_TRUE);
glDisable(GL_BLEND);
glPopMatrix();
glutKeyboardFunc( keyboard );
glutPostRedisplay();
glutSwapBuffers();
}
void keyboard ( unsigned char key, int x, int y )
{
switch ( key ) {
case 27: /* Escape key */
exit ( 0 );
break;
case 'o':
eyePosition=true;
break;
```



```
case 'i':
eyePosition=false;
break;
default:
break;
}
}
```

بفر العمق:

- يحتفظ بفر العمق بالمسافة بين عين الناظر والعنصر (البكسلات).
- تابع العمق:

void **glDepthFunc**(GLenum *func*)

- يعمل على فحص العمق المحدد (قيمة مطلقة) وعند نجاحه يتم استبدال القيمة الموجودة مسبقاً في بفر العمق بقيمة العمق الجديد.

مثال: القيمة الافتراضية GL_LESS وتعني أن القيم الموجودة (البكسلات) في مرحلة fragment تنجح في الفحص إذا كانت قيمة z لها أقل من قيم z المخزنة مسبقاً في بفر العمق. أي أنه في هذه الحالة قيم z تمثل المسافة من العنصر إلى عين الناظر والقيم الصغيرة تعني أن العنصر الموافق لها أقرب إلى عين الناظر.

القيم المحتملة للتابع السابق: مبينة في الجدول التالي:

القيمة	الشرح
GL_NEVER	عدم قبول fragment مطلقاً
GL_EQUAL	قبول fragment عندما
GL_LEQUAL	
GL_GREATER	
GL_NOTEQUAL	
GL_GEQUAL	
GL_ALWAYS	

- يتم تفعيل بفر العمق من خلال التعليمة:

```
glEnable(GL_DEPTH_TEST);
```

القسم العملي

تطبيق1: مزج المكعب المدروس سابقاً (التعديل فقط في التابع keyboard) :

```
#include <GL/glut.h>
#include <stdlib.h>
#include<math.h>
    GLfloat LightAmbient[]=     { 0.0f, 0.5f, 0.5f, 1.0f };
    GLfloat LightDiffuse[]=     { 1.0f, 1.0f, 1.0f, 1.0f };
    GLfloat LightPosition[]=    { 0.0f, 0.0f, 20.0f, 1.0f };
```

```
static void redraw(void);
int main(int argc, char **argv);
void keyboard ( unsigned char key, int x, int y );
int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
    glutInitWindowPosition(100,100);
    glutInitWindowSize(400,400);
    glutCreateWindow("draw Lighting&Blending rectangle");
    glutDisplayFunc(redraw);
    glMatrixMode(GL_PROJECTION);
    gluPerspective(45,1.0,10.0,200.0);
    glMatrixMode(GL_MODELVIEW);
    glLightfv(GL_LIGHT1, GL_AMBIENT, LightAmbient);          // Setup The Ambient Light
    glLightfv(GL_LIGHT1, GL_DIFFUSE, LightDiffuse);          // Setup The Diffuse Light
    glLightfv(GL_LIGHT1, GL_POSITION,LightPosition);         // Position The Light
    glEnable(GL_LIGHT1);                                     // Enable Light One
    glEnable(GL_COLOR_MATERIAL);
    glutMainLoop();
    return 0;
}
static void redraw(void)
{
    glClearColor(1.0,1.0,1.0,0.0);
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0f,0.0f,-100.0);
    glRotatef(45,1.0f,0.0f,0.0f);
    glRotatef(45,0.0f,1.0f,0.0f);
    glBegin(GL_QUADS);
            // Front Face
            glColor3f(1.0,0.0,0.0);
            glNormal3f( 0.0f, 0.0f, 1.0f);
            glVertex3f(-10.0f, -10.0f,  10.0f);
            glVertex3f( 10.0f, -10.0f,  10.0f);
            glVertex3f( 10.0f,  10.0f,  10.0f);
            glVertex3f(-10.0f,  10.0f,  10.0f);
            // Back Face
            glColor3f(1.0,1.0,0.0);
            glNormal3f( 0.0f, 0.0f,-1.0f);
            glVertex3f(-10.0f, -10.0f, -10.0f);
            glVertex3f(-10.0f,  10.0f, -10.0f);
            glVertex3f( 10.0f,  10.0f, -10.0f);
            glVertex3f( 10.0f, -10.0f, -10.0f);
            // Top Face
            glColor3f(0.0,1.0,0.0);
            glNormal3f( 0.0f, 1.0f, 0.0f);
            glVertex3f(-10.0f,  10.0f, -10.0f);
            glVertex3f(-10.0f,  10.0f,  10.0f);
            glVertex3f( 10.0f,  10.0f,  10.0f);
            glVertex3f( 10.0f,  10.0f, -10.0f);
            // Bottom Face
            glColor3f(1.0,0.0,1.0);
```

```
                glNormal3f( 0.0f,-1.0f, 0.0f);
                glVertex3f(-10.0f, -10.0f, -10.0f);
                glVertex3f( 10.0f, -10.0f, -10.0f);
                glVertex3f( 10.0f, -10.0f,  10.0f);
                glVertex3f(-10.0f, -10.0f,  10.0f);
                // Right face
                glColor3f(0.0,0.0,1.0);
                glNormal3f( 1.0f, 0.0f, 0.0f);
                glVertex3f( 10.0f, -10.0f, -10.0f);
                glVertex3f( 10.0f,  10.0f, -10.0f);
                glVertex3f( 10.0f,  10.0f,  10.0f);
                glVertex3f( 10.0f, -10.0f,  10.0f);
                // Left Face
                glColor3f(0.0,1.0,1.0);
                glNormal3f(-1.0f, 0.0f, 0.0f);
                glVertex3f(-10.0f, -10.0f, -10.0f);
                glVertex3f(-10.0f, -10.0f,  10.0f);
                glVertex3f(-10.0f,  10.0f,  10.0f);
                glVertex3f(-10.0f,  10.0f, -10.0f);
      glEnd();
      glutKeyboardFunc    ( keyboard );
      glutPostRedisplay();
      glutSwapBuffers();
}
void keyboard ( unsigned char key, int x, int y )
{
  switch ( key ) {
    case 27:  /*  Escape key  */
      exit ( 0 );
      break;
    case 'l':
      glEnable(GL_LIGHTING);
      break;
    case 'f':
      glDisable(GL_LIGHTING);
      break;
      case 'b':
                glEnable(GL_BLEND);
                glDisable(GL_DEPTH_TEST);
      break;
      case 'd':
                glDisable(GL_BLEND);
                glEnable(GL_DEPTH_TEST);
               break;
    default:
      break;
                                }
}
```