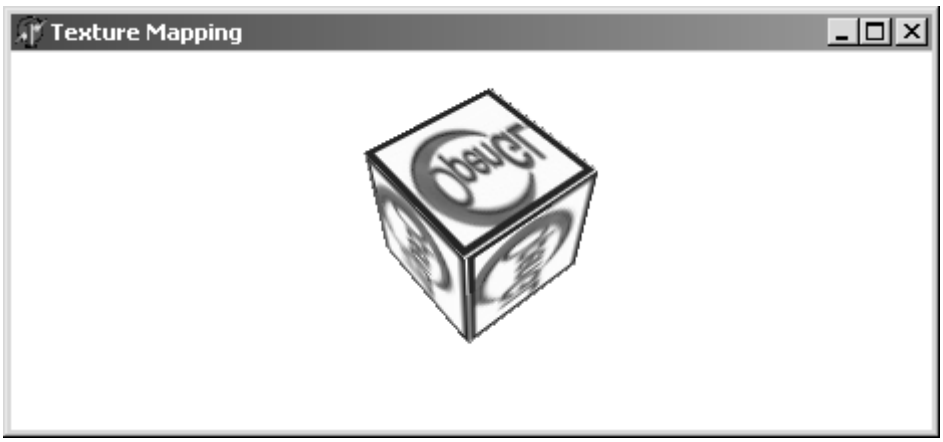
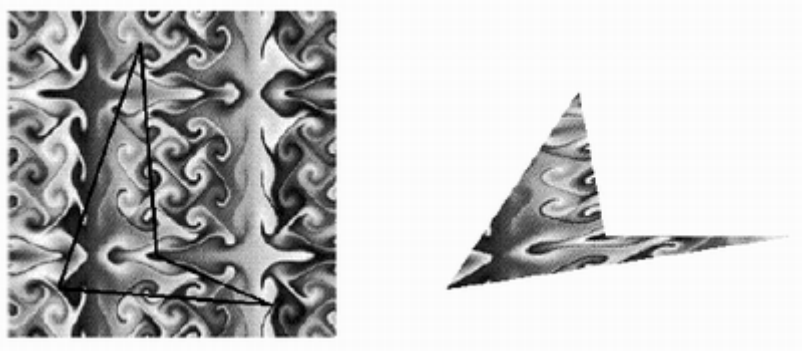
فصل هشتم

نگاشت بافت ها (Texture Mapping)



مقدمه :

دراین فصل می آموزیم که چگونه بیت مپ ها را به سطوح ایجاد شده توسط OpenGL اعمال کنیم ، تا ظاهری واقع گرایانه تر پدید آوریم . بیت مپ ها که بعنوان Texture (بافت) شناخته می شوند می توانند تصویری از درخت ، سنگ مرمر یا هر الگوی جالب دیگری باشند و به پروسه اعمال یک Texture به رویه ای مفروض (مانند شکل زیر) ، Texture Mapping گویند .



نگاشت بافت ها در OpenGL :

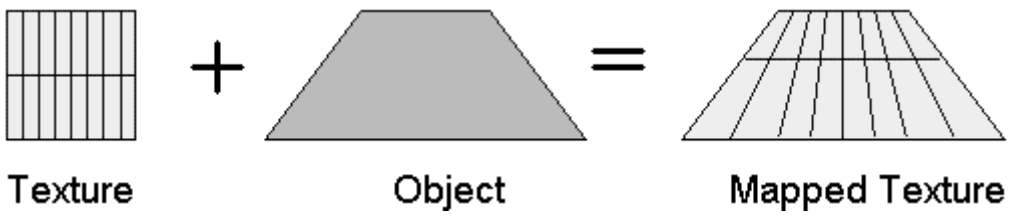
قدم اول در نگاشت بافت یک شیء ، بارگذاری آن بافت می باشد . می توان بیت مپ ها را در زمان اجرای برنامه نیز تولید نمود ، اما در اغلب اوقات بهتر است که از تصاویر حاضر و آماده استفاده شود . بهتر است قبل از بارگذاری تصویر آنرا به دلخواه OpenGL اصلاح نمائیم . توابعی مانند glTexImage2D که تصویری را برای نگاشت بافت تعریف می کنند ، به تصویری نیاز دارند که طول و عرض آن توانی از ۲ باشد . تصویر آماده شده باید حداقل ۶۴ نقطه طول و عرض داشته و بدلایل سازگاری حداکثر ۲۵۶ رنگ باشد . به نظر می رسد که کم حجم ترین و محبوب ترین نرم افزای که می تواند این کار را به سرعت و با کیفیت انجام دهد ، نرم افزار ACDsee بوده و آنرا می توانید از یکی از سایت های www.download.com و یا www.acdsystems.com تهیه کنید . با این حال از تابع gluScaleImage هم می توان بدین منظور استفاده نمود .

باید بخاطر داشته باشید که هرچه اندازه تصویر بارگذاری شده کوچکتر باشد ، OpenGL سریعتر کار خواهد کرد . پس بهتر است که از تصاویری بزرگتر از ۱۲۸×۱۲۸ استفاده نکنید .

پس از بارگذاری تصویر ، تابع glGenTextures به OpenGL می گوید که می خواهیم یک بافت بسازیم . سپس با استفاده از تابع glBindTexture ، حافظه لازم را برای انجام این کار اختصاص خواهیم داد . در ادامه با دستور glTexImage2D ، بافتی واقعی را ایجاد می نمائیم .

مطلبی را که باید بخاطر داشته باشید این است که نمی توان بافتی را درون جفت glBegin و glEnd متصل نمود و اینکار حتما باید خارج از این جفت صورت گیرد .

برای اطمینان حاصل کردن از پوشیده شدن چند ضلعی با بافت ، سمت راست ، بالای بافت باید بر سمت راست بالای چند ضلعی منطبق شودو به همین ترتیب. این امر توسط تابع glTexCoord2f انجام می گردد .



در ادامه مروری خواهیم داشت بر توابع یاد شده :

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glGenTextures(GLsizei n, GLuint *textures);	procedure glGenTextures(n: GLsizei; var textures: GLuint); stdcall; external opengl32;

توضیح :

این تابع نام بافت ها را تولید می کند . آرگومان n ، تعداد نام بافت هایی است که باید تولید شوند و آرگومان textures ، اشاره گری است به اولین عنصر آرایه ای که نام بافت ها را در خود ذخیره کرده است . بطور خلاصه این تابع ، n نام بافت در پارامتر textutes را بر می گرداند.

هر عدد صحیح مثبت غیر صفر می تواند بعنوان نام یک بافت بکار رود . سپس با استفاده از تابع glBindTexture این نام تولید شده به داده های بافت مربوطه شده متصل خواهد شد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glBindTexture(GLenum target, GLuint texture);	procedure glBindTexture(target: GLenum; texture: GLuint); stdcall; external opengl32;

توضیح :

بافت دارای نام را به داده های مربوط به آن متصل می نماید ، درحقیقت کار ایجاد و استفاده از اشیاء بافتی را انجام می دهد . آرگومان target ، هدفی است که بافت به آن متصل خواهد شد و می تواند GL_TEXTURE_1D (یک بعدی) و یا GL_TEXTURE_2D (دوبعدی) باشد. آرگومان texture نام بافتی است که هم اکنون در حال استفاده نمی باشد .

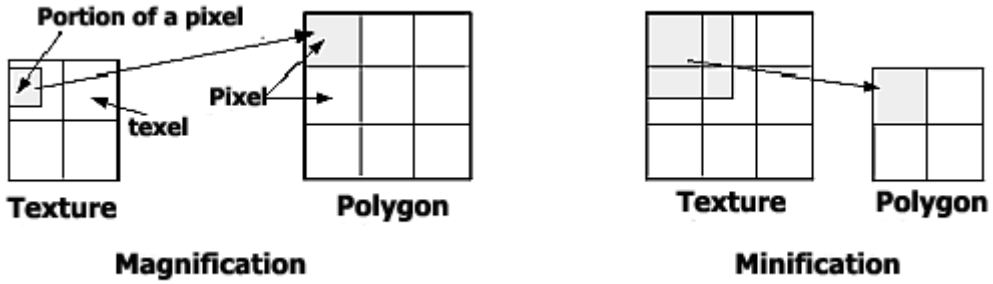
هنگامیکه بافتی به هدف متصل می گردد ، اتصال قبلی دیگر تاثیری نخواهد داشت . توابعی مانند glTexImage2D داده ها را درون اشیاء بافتی ذخیره می کنند .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glGetTexParameteriv(GLenum target, GLenum pname, GLint *params);	Procedure glGetTexParameteriv(target: GLenum; pname: GLenum; params: PGLint); stdcall; external 'OPENGL32.DLL';
void glTexParameteri(GLenum target, GLenum pname, GLint param);	Procedure glTexParameteri(target: GLenum; pname: GLenum; param: GLint); stdcall; external 'OPENGL32.DLL';

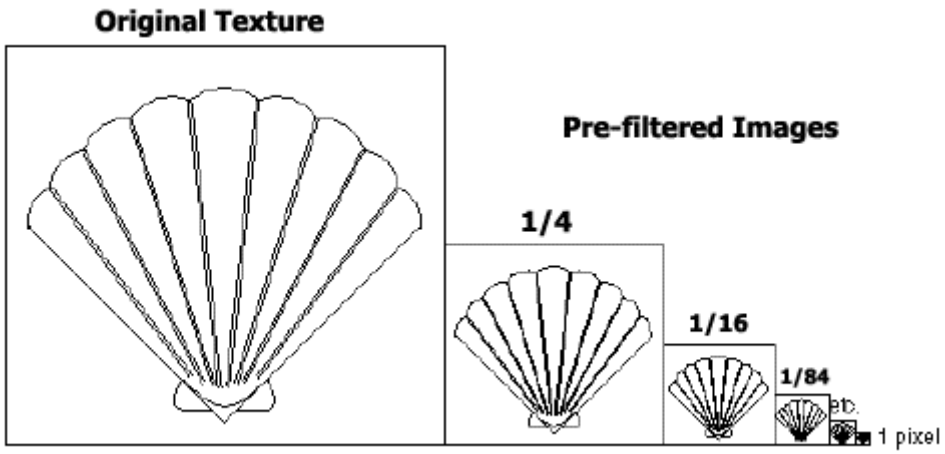
توضیح :

تـابع glTexParameteri پارامترهـای بافـت را تنظـیم مـی کنـد . آرگومـان target آن هماننـد glBindTexture می باشد. آرگومان pname نام سمبولیک پارامتر بافت می باشد . تنها سمبول های زیر مجاز هستند :

GL_TEXTURE_MIN_FILTER : تابع کوچک کردن (minifying) بافت هنگامی بکار می رود کـه نقطـه در حال نگاشت ، باید بر روی ناحیه ای بزرگتر از یک المان بافتی ، نگاشت شود (شکل زیر) .



شش تابع کوچک کردن تعریف شده انـد . دوتـای آنهـا از نزدیکتـرین ۴ عنصـر بـافتی اسـتفاده می کنند و مابقی از Mipmaps . Mipmaps مجموعه از آرایـه هـای مرتـب هسـتند کـه انـدازه هـای کوچکتری از تصویر اصلی را در خود ذخیره می کنند (شکل زیر).



اگر بافت ، ابعادی معادل $2^m \times 2^n$ داشته باشد تعداد Mipmaps معـادل Max(n,m)+1 خواهـد بـود . اولین Mipmap ، بافت اصلی به ابعاد ذکرشده می باشد. Mipmaps متعاقب،ابعادی معـادل $2^l-1 \times 2^k-1$ را خواهنـد داشـت ، بطوریکـه $2^l \times 2^k$ ابعـاد Mipmap قبلـی اسـت . Mipmaps توسـط تـوابعی ماننـد glTexImage2D تولید می شوند .

GL_TEXTURE_MAG_FILTER : تابع بزرگنمایی(Magnification) هنگامی بکار برده می شود که نقطه در حال نگاشت بر روی ناحیه ای کوچکتر یا مساوی آن ، نگاشت شود . آن تابع بزرگنمایی را به GL_NEAREST ویا GL_LINEAR تنظیم می کند .

GL_TEXTURE_WRAP_S : پارامترهای پوشانیدن (Wrap) مختصات بافتی S را به GL_CLAMP و یا GL_REPEAT تنظیم می کند . GL_CLAMP سبب می شود بافت بدون تکرار شدن کل شیء را بپوشاند و GL_REPEAT سبب خواهد شد که الگوی پوشاننده شکل ، تکرار شود . المان بافت حاشیه ای تنها هنگامی قابل دستیابی است که پوشانیدن به GL_CLAMP تنظیم شود .

GL_TEXTURE_WRAP_T : پارامتر پوشانیدن مختص t را به GL_CLAMP یا GL_REPEAT تنظیم می کند .

آرگومان param مقدار pname است .

در حالت glGetTexParameter*iv* :

آرگومان target همانند آنچه که ذکر شد می باشد و Pname : علاوه برGL_TEXTURE_MIN_FILTER ، GL_TEXTURE_MAG_FILTER ، GL_TEXTURE_WRAP_S و GL_TEXTURE_WRAP_T شامل موارد زیر نیز می باشد :

GL_TEXTURE_BORDER_COLOR : رنگ حاشیه را تنظیم می کند . در این حالت آرگومان param حاوی چهار جزء رنگ RGBA می باشند .

GL_TEXTURE_PRIORITY : حق تقدم بافت مستقر شده در حافظه را مشخص می کند . مقادیر مجاز آن در بازه صفر و یک قرار دارند .

آرگومان param اشاره گری است به آرایه ای که مقادیر pname در آن ذخیره می شود .

توابع کوچک کردن که توسط پارامتر param تامین می شوند به صورت زیر می باشند :

GL_NEAREST : مقدار المان بافتی را که به مرکز نقطه در حال نگاشت نزدیکترین است را بر می گرداند .

GL_LINEAR : میانگین وزنی چهار المان بافتی را که به مرکز نقطه در حال نگاشت نزدیکترین هستند را بر می گرداند . GL_NEAREST عموما سریعتر از GL_LINEAR می باشد ، اما تصاویر بافتی با گوشه های تیزتری را ایجاد می کند .

GL_NEAREST_MIPMAP_NEAREST : Mipmap ایی را که تا حد زیادی با اندازه نقطه در حال نگاشت منطبق است ، انتخاب می کند و از معیار GL_NEAREST کمک می گیرد .

GL_LINEAR_MIPMAP_NEAREST : Mipmap ایی را که تا حد زیادی با اندازه نقطه در حال نگاشت منطبق است ، انتخاب می کند و از معیار GL_LINEAR کمک می گیرد .

GL_NEAREST_MIPMAP_LINEAR : دو Mipmap ایی را که تا حد زیادی با اندازه نقطه در حال نگاشت منطبق است ، انتخاب می کند و از معیار GL_NEAREST کمک می گیرد .

GL_LINEAR_MIPMAP_LINEAR : دو Mipmap ایی را که تا حد زیادی با اندازه نقطه در حال نگاشت منطبق است ، انتخاب می کند و از معیار GL_LINEAR کمک می گیرد .

توابع بزرگنمایی که توسط پارامتر param تامین می شوند به صورت زیر می باشند :

GL_NEAREST و GL_LINEAR .

بطور خلاصه پارامترها و مقادیر فیلترهای بزرگ کردن و کوچک کردن در جدول زیر آورده شده اند :

Parameter	Values
GL_TEXTURE_MAG_FILTER	GL_NEAREST or GL_LINEAR
GL_TEXTURE_MIN_FILTER	GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, or GL_LINEAR_MIPMAP_LINEAR

پارامترهای توابع glTexParameter*() :

Parameter	Values
GL_TEXTURE_WRAP_S	GL_CLAMP, GL_REPEAT
GL_TEXTURE_WRAP_T	GL_CLAMP, GL_REPEAT
GL_TEXTURE_MAG_FILTER	GL_NEAREST, GL_LINEAR
GL_TEXTURE_MIN_FILTER	GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_LINEAR
GL_TEXTURE_BORDER_COLOR	any four values in [0.0, 1.0]
GL_TEXTURE_PRIORITY	[0.0, 1.0] for the current texture object

و همچنین **کیفیت** تصاویر ایجاد شده توسط ترکیب **پرکاربردترین** پارامترهای فوق را می توان به صورت زیر خلاصه کرد :

Quality	OpenGL Parameter	Value
High	GL_TEXTURE_MAG_FILTER GL_TEXTURE_MIN_FILTER	GL_LINEAR GL_LINEAR_MIPMAP_NEAREST
Medium	GL_TEXTURE_MAG_FILTER GL_TEXTURE_MIN_FILTER	GL_LINEAR GL_LINEAR
Low	GL_TEXTURE_MAG_FILTER GL_TEXTURE_MIN_FILTER	GL_NEAREST GL_NEAREST

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Procedure glTexImage2D(target: GLenum; level: GLint; components: GLint; width: Glsizei ; height: GLsizei; border: GLint; format: GLenum; atype: GLenum; pixels: Pointer); stdcall; external 'OPENGL32.DLL';	void glTexImage2D(GLenum target, GLint level, GLint internalformat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels);

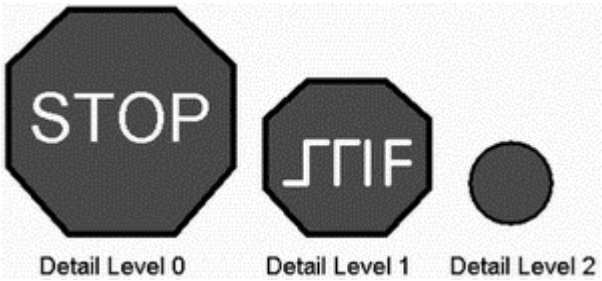
توضیح :

بافتی ۲ بعدی را تعریف می کند .

آرگومان target در این حالت تنها GL_TEXTURE_2D می تواند باشد .

آرگومان level : سطح جزئیات است . سطح n ، معادل n امین mipmap کاهش یافته تصویر است.

برای مثال هنگامیکه به شیءایی نزدیک می شویم ، شیء بزرگتر می شود و جزئیات بیشتری را می بینیم و هنگامیکه از آن دور می شویم برعکس (شکل زیر) .



ازتابع gluBuild2DMipmaps نیز ، برای این منظور بکار گرفته می شود .

Internalformat : **تعداد اجزاء رنگ در بافت . باید ۱ ، ۲ ،۳ یا ۴ و یا یکی از ثوابت زیر باشد :**

Constant	R Bits	G Bits	B Bits	A Bits	L Bits	I Bits
GL_ALPHA	—	—	—	—	—	—
GL_ALPHA4	—	—	—	4	—	—
GL_ALPHA8	—	—	—	8	—	—
GL_ALPHA12	—	—	—	12	—	—
GL_ALPHA16	—	—	—	16	—	—
GL_LUMINANCE	—	—	—	—	—	—
GL_LUMINANCE4	—	—	—	—	4	—
GL_LUMINANCE8	—	—	—	—	8	—
GL_LUMINANCE12	—	—	—	—	12	—
GL_LUMINANCE16	—	—	—	—	16	—
GL_LUMINANCE_ALPHA	—	—	—	—	—	—
GL_LUMINANCE4_ALPHA4	—	—	—	4	4	—
GL_LUMINANCE6_ALPHA2	—	—	—	2	6	—
GL_LUMINANCE8_ALPHA8	—	—	—	8	8	—
GL_LUMINANCE12_ALPHA4	—	—	—	4	12	—
GL_LUMINANCE12_ALPHA12	—	—	—	12	12	—
GL_LUMINANCE16_ALPHA16	—	—	—	16	16	—
GL_INTENSITY	—	—	—	—	—	—
GL_INTENSITY4	—	—	—	—	—	4
GL_INTENSITY8	—	—	—	—	—	8
GL_INTENSITY12	—	—	—	—	—	12
GL_INTENSITY16	—	—	—	—	—	16
GL_RGB	—	—	—	—	—	—
GL_R3_G3_B2	3	3	2	—	—	—
GL_RGB4	4	4	4	—	—	—
GL_RGB5	5	5	5	—	—	—
GL_RGB8	8	8	8	—	—	—
GL_RGB10	10	10	10	—	—	—
GL_RGB12	12	12	12	—	—	—
GL_RGB16	16	16	16	—	—	—
GL_RGBA	—	—	—	—	—	—
GL_RGBA2	2	2	2	2	—	—
GL_RGBA4	4	4	4	4	—	—
GL_RGB5_A1	5	5	5	1	—	—
GL_RGBA8	8	8	8	8	—	—
GL_RGB10_A2	10	10	10	2	—	—
GL_RGBA12	12	12	12	12	—	—
GL_RGBA16	16	16	16	16	—	—

Width : **عرض تصویر بافتی می باشد و باید معادل** $2^n+2(border)$ **باشد (n عددی صحیح است).**

Height : ارتفاع تصویر بافتی بوده و باید معادل $2^m+2(border)$ باشد (m عددی صحیح است).

Border : عرض حاشیه است . باید صفر یا یک باشد .

Format : فرمت داده نقاط می باشد و یکی از مقادیر زیر برای آن مجاز است :

GL_COLOR_INDEX : هر المان یک اندیس رنگ می باشد و در بازه صفر و یک قرار دارد .

GL_RED ، GL_GREEN ، GL_BLUE و GL_ALPHA : هر المان به ترتیب یک جزء قرمز ، سبز ، آبی و یا قرمز خواهد بود . در این حالت ها ،در رنگ RGBA بقیه عناصر بجز آلفا و عنصر ذکر شده صفر بوده و آلفا ، یک می باشد .

GL_RGB و GL_RGBA : هر المان به ترتیب یک RGB و یا RGBA می باشد ، در بازه صفر و یک و آلفا در حالت RGB ، یک است .

GL_BGR_EXT : هر نقطه به ترتیب گروهی از سه جزء : آبی ، سبز و قرمز می باشد . این فرمت مطابق DIBs ویندوز است .

GL_BGRA_EXT : هر نقطه به ترتیب گروهی از سه جزء : آبی ، سبز ، قرمز و آلفا می باشد .

GL_LUMINANCE : هر المان یک مقدار تشعشع می باشد . این مقدار سپس به RGBA تبدیل خواهد شد . در این حالت آلفا ، یک است .

GL_LUMINANCE_ALPHA : هر المان یک جفت تشعشع / آلفا می باشد و سپس به RGBA تبدیل خواهد شد .

آرگومان type : نوع داده ای نقاط را معین می کند . مقادیر زیر قابل قبول هستند :

Type	Meaning
GL_UNSIGNED_BYTE	Unsigned 8-bit integer
GL_BYTE	Signed 8-bit integer
GL_BITMAP	Single bits in unsigned 8-bit integers
GL_UNSIGNED_SHORT	Unsigned 16-bit integer
GL_SHORT	Signed 16-bit integer
GL_UNSIGNED_INT	Unsigned 32-bit integer
GL_INT	32-bit integer
GL_FLOAT	Single-precision floating-point

آرگومان pixels : اشاره گری به داده های تصویر در حافظه است .

این تابع توسط دستور glEnable(GL_TEXTURE_2D); فعال می شود .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glTexCoord2f(GLfloat s, GLfloat t);	Procedure glTexCoord2f(s: GLfloat; t: GLfloat); stdcall; external 'OPENGL32.DLL';

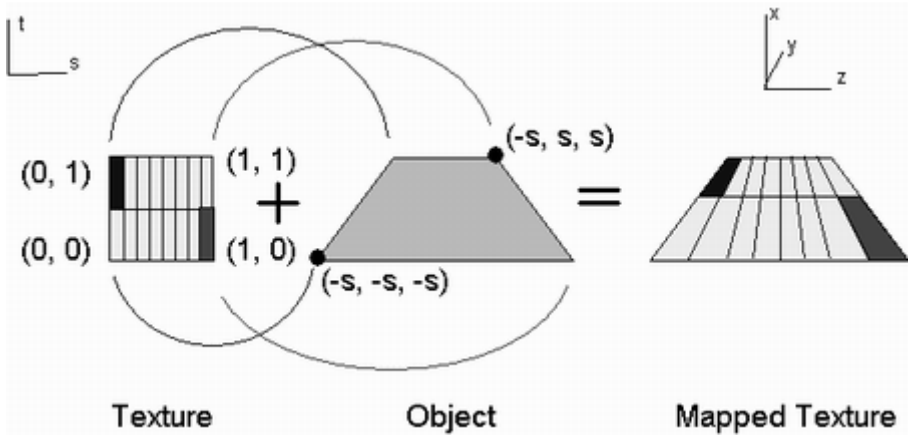
توضیح :

مختصات بافت جاری را تنظیم می کند . نگارش های مختلف این تابع توانایی تعیین مختصات بافت را در ۱ ، ۲ ، ۳ و یا ۴ بعد ، دارا هستند .

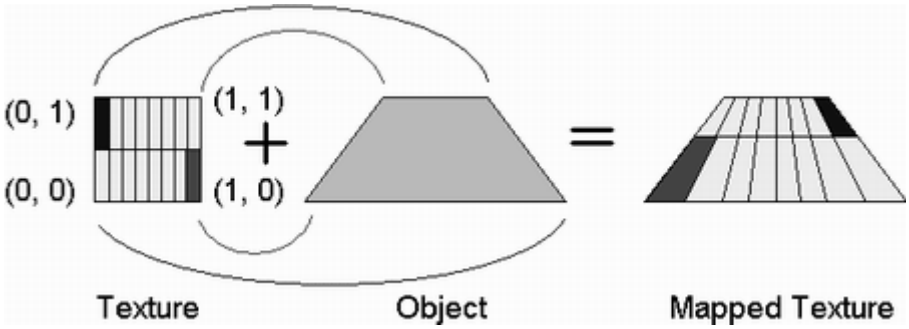
glTexCoord1	(*s*, 0, 0, 1);
glTexCoord2	(*s*, *t*, 0, 1);
glTexCoord3	(*s*, *t*, *r*, 1);
glTexCoord4	(*s*, *t*, *r*, *q*);

مختصات بافت جاری را در هر لحظه می توان به روز در آورد . بخصوص اینکه این تابع را می توان در بین جفت glBegin و glEnd فراخوانی نمود .

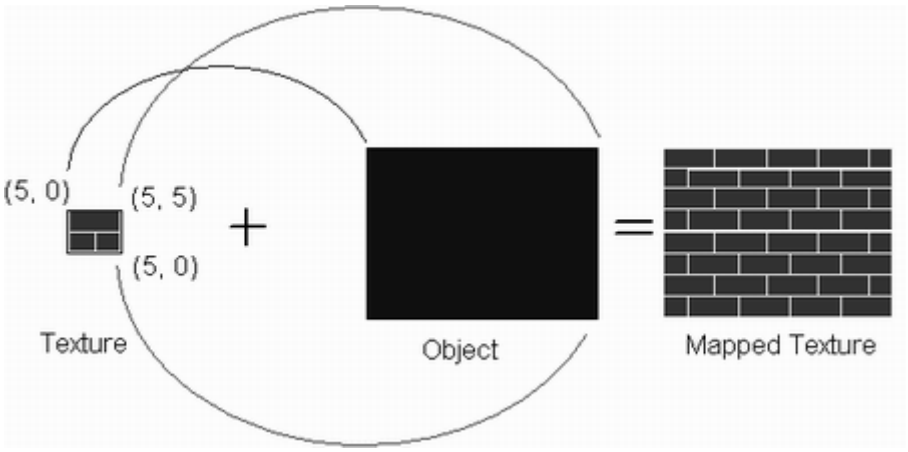
اولین آرگومان glTexCoord2f مختص X بافت می باشد . اگر صفر باشد در سمت چپ ، 0.5 ، وسط و یک، سمت راست، قرار می گیرد . دومین آرگومان مختص Y است . صفر در پایین ، 0.5 ، در میانه و ۱ در بالای بافت قرار دارد . اکنون ما می دانیم که (0,1) مختصات گوشه سمت چپ بالای **بافت** و (-1,1) مختصات گوشه سمت چپ بالای **چند ضلعی** است (شکل زیر) .

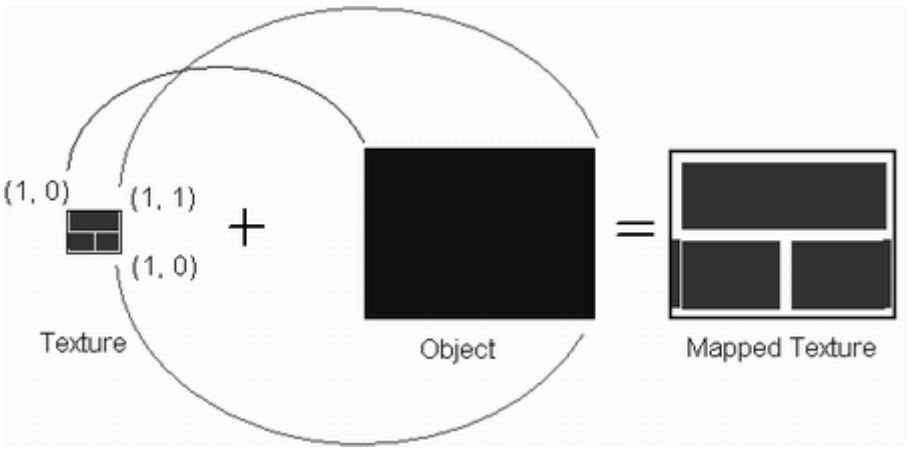


اگر مختصات بافت را با دقت انتخاب کنیم ، می توان بافت را معکوس کرد (شکل زیر).



برای اینکه بتوان مربعی را شبیه به دیواری ساخته شده با آجر نمود ، باید تصویر را تکرار نمائیم . برای تکرار بافت از مختصات بافتی بیشتر از ۱ استفاده کنید (شکل زیر) . البته پارامتر GL_REPEAT را نیز نباید فراموش کرد .





بکار نبردن مختصات بیش از یک سبب می شود که بافت روی سطح تکرار نشود .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
int gluBuild2DMipmaps(GLenum target, GLint components, GLint width, GLint height, GLenum format, GLenum type, const void *data);	function gluBuild2DMipmaps(target: GLenum; components, width, height: GLint; format,atype: GLenum; Data: Pointer): GLint; stdcall; external glu32;

توضیح :

این تابع Mipmaps دوبعدی را پدید می آورد . برای این منظور ، دستور فوق تصویری را توسط تابع glTexImage2D دریافت نموده و تمام Mipmaps آنرا تولید می کند .

آرگومان target در این حالت فقط GL_TEXTURE_2D می باشد .

components : تعداد اجزاء رنگ در بافت یم باشد که باید ۱ ، ۲ ، ۳ یا ۴ باشد .

width, height : ارتفاع و عرض ، تصویر بافت می باشند .

format : فرمت داده نقاط بوده و یکی از مقادیر ذکر شده زیر می تواند باشد :

GL_COLOR_INDEX, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_BGR_EXT, GL_BGRA_EXT, GL_LUMINANCE, or GL_LUMINANCE_ALPHA

Type : نوع داده ای آرگومان data است و یکی از مقادیر زیر را می پذیرد :

GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, or GL_FLOAT

Data : اشاره گری است به داده های تصویر در حافظه .

این تابع در صورت موفقیت صفر بر می گرداند و در غیر اینصورت کد خطای GLU متناظر را تولید خواهد کرد .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glTexEnvi(GLenum target, GLenum pname, GLint param);	Procedure glTexEnvi(target: GLenum; pname: GLenum; param: GLint); stdcall; external 'OPENGL32.DLL';
void glTexEnviv(GLenum target, GLenum pname, const GLint *params);	Procedure glTexEnviv(target: GLenum; pname: GLenum; params: PGLint); stdcall; external 'OPENGL32.DLL';

توضیح :

این توابع پارامترهای محیطی بافت و حالت ترسیم را تنظیم می کنند . محیط بافتی معین می سازد که مقادیر بافت ، هنگامیکه جزئی در حال نگاشت است ، چگونه باید ترجمه شوند . برای

مثال می توان رنگ بافت و رنگ سطحی که باید ارائه شود را بدون نگاشت بافت ، تلفیق نمود و یا پس از نگاشت ، رنگ های بافت و سطح را با هم مخلوط کرد .

در حالت glTexEnv*i* :

آرگومان target : یک بافت محیطی می باشد و باید معادل GL_TEXTURE_ENV قرارگیرد .

آرگومان pname : نام سمبولیک پارامتر بافت محیطی است و معادل GL_TEXTURE_ENV_MODE می باشد .

آرگومان param : یک ثابت سمبولیک بوده و یکی از موارد زیر را می پذیرد :

GL_MODULATE, GL_DECAL, GL_BLEND, or GL_REPLACE

در حالت GL_DECAL ، رنگ بافت ، جایگزین رنگ سطح می شود (با استفاده از فرمت داخلی RGB) و با استفاده از فرمت داخلی RGBA ، اجزاء رنگ با رنگ بافت به نسبت تعیین شده توسط آلفا ، مخلوط خواهند شد و جزء آلفا تغییری نخواهد کرد . از حالت GL_DECAL برای اعمال بافتی مات بر روی یک شی ء می توان استفاده کرد . برای مثال ترسیم یک برچسب مات بر روی سوپ ! و یا اعمال نشانی رسمی روی بال هواپیما .

حالت GL_MODULATE برای حالتی که نور پردازی وجود دارد مناسب می باشد . برای فرمت های داخلی رنگی یاد شده ، هر رنگ در حال ترسیم ، در رنگ موجود در بافت ضرب خواهد شد.

هر یک از این موارد خود می توانند موضوع فصلی جداگانه باشند .

جدول زیر طریقه تولید رنگ RGBA توسط توابع فوق را نشان می دهد .

number of componets	texture functions		
	GL_MODULATE	**GL_DECAL**	**GL_BLEND**
1	$C_v = L_t C_f$ $A_v = A_f$	undefined	$C_v = (1 - L_t) C_f + L_t C_c$ $A_v = A_f$
2	$C_v = L_t C_f$ $A_v = A_t A_f$	undefined	$C_v = (1 - L_t) C_f + L_t C_c$ $A_v = A_f A_f$
3	$C_v = C_t C_f$ $A_v = A_f$	$C_v = C_t$ $A_v = A_f$	undefined
4	$C_v = C_t C_f$ $A_v = A_t A_f$	$C_v = (1 - A_t) C_f + A_t C_t$ $A_v = A_f$	undefined

در جدول فوق : C ، معادل مقدار سه گانه RGB و A معادل مقدار آلفا مـی باشـد . انـدیس f ، بیـانگر جزء در حال ورود و اندیس t ، مربوط به تصویر بافتی هستند . اندیس C برای رنگ محیط بافت و اندیس V بیانگر مقدار تولید شده توسط تابع بافتی است .

در حالت glTexEnv*iv* :

آرگومان target : یک بافت محیطی می باشد و باید معادل GL_TEXTURE_ENV قرارگیرد .

آرگومان pname : نام سمبولیک پارامتر بافت محیطی است و معـادل GL_TEXTURE_ENV_MODE و یا GL_TEXTURE_ENV_COLOR می باشد .

params : اشاره گری است به آرایه ای از پارامترها ، که یا یک ثابت سمبولیک می باشد و یا یـک رنگ RGBA .

تذکر :

نگاشت بافت ها درحالت اندیس رنگی کار نمی کند .

تابع به فرمت زبان C	تابع به فرمت زبان دلفی
void glPixelStoref(GLenum pname, GLfloat param);	Procedure glPixelStoref(pname: GLenum; param: GLfloat); stdcall; external 'OPENGL32.DLL';
void glPixelStorei(GLenum pname, GLint param);	Procedure glPixelStorei(pname: GLenum; param: GLint); stdcall; external 'OPENGL32.DLL';

توضیح :

طریقه ذخیره شدن داده هلـی تصـویر را در حافظـه کـامپیوتر معـین مـی کنـد و روی عملکـرد تعدادی تابع منجمله glTexImage2D تاثیرگذار است . ثوابت مجاز برای آرگومـان pname بـه همـراه نوع ، مقدار اولیه و بازه مجاز آن در جدول زیر ارائه شده انـد . پارامترهـای GL_UNPACK* طریقـه باز شدن داده ها از حافظه را بیان می کنند و GL_PACK* طریقه فشرده سازی داده ها را بـه درون حافظه .

Pname	Type	Initial Value	Valid Range
GL_PACK_SWAP_BYTES	Boolean	false	true or false
GL_PACK_SWAP_BYTES	Boolean	false	true or false
GL_PACK_ROW_LENGTH	integer	0	[0,∞)
GL_PACK_SKIP_ROWS	integer	0	[0,∞)
GL_PACK_SKIP_PIXELS	integer	0	[0,∞)
GL_PACK_ALIGNMENT	integer	4	1, 2, 4, or 8
GL_UNPACK_SWAP_BYTES	Boolean	false	true or false
GL_UNPACK_LSB_FIRST	Boolean	false	true or false
GL_UNPACK_ROW_LENGTH	integer	0	[0,∞)
GL_UNPACK_SKIP_ROWS	integer	0	[0,∞)
GL_UNPACK_SKIP_PIXELS	integer	0	[0,∞)
GL_UNPACK_ALIGNMENT	integer	4	1, 2, 4, or 8

تابع به فرمت زبان دلفی	تابع به فرمت زبان C
Function gluScaleImage(format: GLenum; widthin: GLint; heightin: GLint; typein: GLenum; const datain; widthout: GLint; heightout: GLint; typeout: GLenum; var dataout): INT; stdcall; external 'GLU32.DLL';	int gluScaleImage(GLenum format, GLint widthin, GLint heightin, GLenum typein, const void *datain, GLint widthout, GLint heightout, GLenum typeout, void *dataout);

توضیح :

مقیاس و ابعاد یک تصویر را به اندازه ای دلخواه تنظیم می کند .

آرگومان format : فرمت داده ای نقاط بوده و یکی از مقادیر سمبولیک زیر را می پذیرد :

GL_COLOR_INDEX, GL_STENCIL_INDEX, GL_DEPTH_COMPONENT, GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_BGR_EXT, GL_BGRA_EXT, GL_LUMINANCE, and GL_LUMINANCE_ALPHA.

آرگومان های widthin, heightin : عرض و طول تصویر اصلی می باشند .

typein : نوع داده ای آرگومان datain که یکی از مقادیر زیر می تواند باشد :

GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, or GL_FLOAT.

datain : اشاره گری است به تصویر اصلی .

widthout, heightout : عرض و طول تصویر نهایی می باشند .

typeout : نوع داده ای برای آرگومان dataout که یکی از مقادیر زیر می تواند باشد :

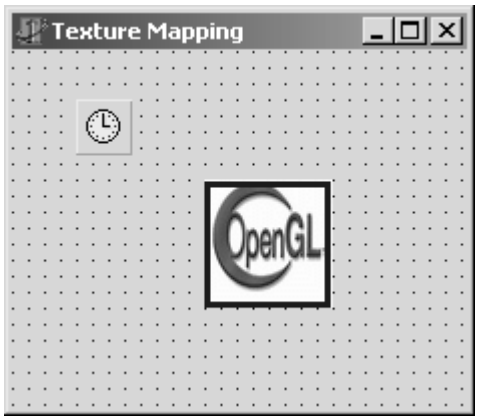
GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, or GL_FLOAT.

dataout : اشاره گری است به تصویر نهایی .

این تابع در صورت موفقیت صفر بر می گرداند و در غیر اینصورت کد خطای GLU متناظر را تولید خواهد کرد .

برنامه فصل :

برای اجرای این برنامه به کنترل های Timer و Image نیاز می باشد (شکل زیر) .



سپس تصویری دلخواه را با توجه به مطالب ذکرشده در مورد اندازه و تعداد رنـگ هـای لازم، به درون کنترل تصویر ، بارگذاری کنید (در مثال زیر از تصویری به ابعاد ۶۴×۶۴ نقطـه در ۲۵۶ رنگ ، استفاده شده است .)

```
unit Ch08;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls,
   Forms, Dialogs, OpenGL, SPF, ExtCtrls ;
type
  TForm1 = class(TForm)
    Timer1: TTimer;
    Image1: TImage;
    procedure FormResize(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
{$R *.DFM}
var
 f_Hdc : LongInt;

procedure glGenTextures(n: GLsizei;
        var textures: GLuint); stdcall; external opengl32;
procedure glBindTexture(target: GLenum;
        texture: GLuint); stdcall; external opengl32;
```

```
// The gluBuild2DMipmaps declaration in the Delphi 4 version
//of the OpenGL unit was improperly declared. I've redeclared it here:
function gluBuild2DMipmaps(target: GLenum;
          components, width, height: GLint;
          format,atype: GLenum;
          Data: Pointer): GLint; stdcall; external glu32;
var
  xrot: GLfloat;    // X Rotation
  yrot: GLfloat;    // Y Rotation
const
  z: GLfloat = -5.0;  // Depth Into The Screen
  LightAmbient: array [0..3] of GLfloat = ( 0.5, 0.5, 0.5, 1.0 );
  LightDiffuse: array [0..3] of GLfloat = ( 1.0, 1.0, 1.0, 1.0 );
  LightPosition: array [0..3] of GLfloat = ( 0.0, 0.0, 2.0, 1.0 );
  filter: GLuint = 0;  // Which Filter To Use
var
  texture: array [0..2] of GLuint;    // Storage for 3 textures

procedure getRGB(num : LongInt; var r , g , b : Integer);
begin
//  extract R,G,B from a long format RBG
       b := Trunc((num And 16711680)/ 65536) ;
       g := Trunc((num And 65280)/ 256 );
       r :=  num  And 255 ;
End;

// Load Bitmaps And Convert To Textures
procedure LoadGLTextures (pctPicToLoad : TImage );
 var
  x , y : LongInt;
   c , bitmapWidth, bitmapHeight : LongInt ;
   Red , Green , Blue : Integer ;
   texture1: array[0..63,0..63,0..2] of  GLubyte;
begin
   // Load Texture
    pctPicToLoad.AutoSize := True ;
    bitmapHeight := pctPicToLoad.Height ;  // Set the array
    bitmapWidth := pctPicToLoad.Width ; // size.

 For x := 0 To  bitmapWidth-1  do
     For y := 0 To bitmapHeight-1  do
       begin
         c := ColorToRGB(pctPicToLoad.Canvas.Pixels[x,y]);
         getRGB(c,Red,Green,Blue);
         texture1[ x, bitmapHeight - y - 1,0] := Red ;   //GetRed
         texture1[ x, bitmapHeight - y - 1,1] := Green ; //GetGreen
         texture1[ x, bitmapHeight - y - 1,2] := Blue ;  //GetBlue
       end;

       pctPicToLoad.Visible:=false;

           if not Assigned( @texture1 ) then Halt(1);

           // Create Nearest Filtered Texture
           glGenTextures(3, texture[0]);
           glBindTexture(GL_TEXTURE_2D, texture[0]);
           glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_NEAREST);
```

```
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_NEAREST);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, bitmapWidth, bitmapHeight,
          0, GL_RGB, GL_UNSIGNED_BYTE, @texture1);

        // Create Linear Filtered Texture
        glBindTexture(GL_TEXTURE_2D, texture[1]);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
        glTexImage2D(GL_TEXTURE_2D, 0, 3, bitmapWidth, bitmapHeight ,
        0, GL_RGB, GL_UNSIGNED_BYTE, @texture1);

        // Create MipMapped Texture
        glBindTexture(GL_TEXTURE_2D, texture[2]);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
        glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,
         GL_LINEAR_MIPMAP_NEAREST);
        gluBuild2DMipmaps(GL_TEXTURE_2D, 3, bitmapWidth, bitmapHeight,
           GL_RGB, GL_UNSIGNED_BYTE, @texture1);
end;

// This Will Be Called Right After The GL Window Is Created
procedure InitGL(Width: GLsizei; Height: GLsizei);
  begin
        LoadGLTextures(form1.Image1);                // Load The Texture(s)
        glEnable(GL_TEXTURE_2D);      // Enable Texture Mapping
      // This Will Clear The Background Color To White
        glClearColor(1.0, 1.0, 1.0, 0.0);
        glClearDepth(1.0);            // Enables Clearing Of The Depth Buffer
        glDepthFunc(GL_LESS);     // The Type Of Depth Test To Do
        glEnable(GL_DEPTH_TEST);    // Enables Depth Testing
        glShadeModel(GL_SMOOTH);   // Enables Smooth Color Shading

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();  // Reset The Projection Matrix

      // Calculate The Aspect Ratio Of The Window
        gluPerspective(45.0,Width/Height,0.1,100.0);

        glMatrixMode(GL_MODELVIEW);

        glLightfv(GL_LIGHT1, GL_AMBIENT, @LightAmbient);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, @LightDiffuse);
        glLightfv(GL_LIGHT1, GL_POSITION,@LightPosition);
        glEnable(GL_LIGHT1);
end;

procedure DrawGLScene();

  begin
      // Clear The Screen And The Depth Buffer
        glClear(GL_COLOR_BUFFER_BIT or GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();   // Reset The View
        glTranslatef(0.0,0.0,z);

        glRotatef(xrot,1.0,0.0,0.0);
        glRotatef(yrot,0.0,1.0,0.0);

        glBindTexture(GL_TEXTURE_2D, texture[filter]);
```

```
        glBegin(GL_QUADS);
                // Front Face
                glNormal3f( 0.0, 0.0, 1.0);
                glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
                glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
                glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0,  1.0);
                glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0,  1.0);
                // Back Face
                glNormal3f( 0.0, 0.0,-1.0);
                glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
                glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
                glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
                glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0, -1.0);
                // Top Face
                glNormal3f( 0.0, 1.0, 0.0);
                glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
                glTexCoord2f(0.0, 0.0); glVertex3f(-1.0,  1.0,  1.0);
                glTexCoord2f(1.0, 0.0); glVertex3f( 1.0,  1.0,  1.0);
                glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
                // Bottom Face
                glNormal3f( 0.0,-1.0, 0.0);
                glTexCoord2f(1.0, 1.0); glVertex3f(-1.0, -1.0, -1.0);
                glTexCoord2f(0.0, 1.0); glVertex3f( 1.0, -1.0, -1.0);
                glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
                glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
                // Right face
                glNormal3f( 1.0, 0.0, 0.0);
                glTexCoord2f(1.0, 0.0); glVertex3f( 1.0, -1.0, -1.0);
                glTexCoord2f(1.0, 1.0); glVertex3f( 1.0,  1.0, -1.0);
                glTexCoord2f(0.0, 1.0); glVertex3f( 1.0,  1.0,  1.0);
                glTexCoord2f(0.0, 0.0); glVertex3f( 1.0, -1.0,  1.0);
                // Left Face
                glNormal3f(-1.0, 0.0, 0.0);
                glTexCoord2f(0.0, 0.0); glVertex3f(-1.0, -1.0, -1.0);
                glTexCoord2f(1.0, 0.0); glVertex3f(-1.0, -1.0,  1.0);
                glTexCoord2f(1.0, 1.0); glVertex3f(-1.0,  1.0,  1.0);
                glTexCoord2f(0.0, 1.0); glVertex3f(-1.0,  1.0, -1.0);
        glEnd();

      swapBuffers(f_Hdc);
end;

procedure TForm1.FormResize(Sender: TObject);
begin
  wglMakeCurrent(f_Hdc,hrc); //activate the RC
  // Prevent A Divide By Zero If The Window Is Too Small
  if (Height=0) then   Height:=1;
  // Reset The Current Viewport And Perspective Transformation
  glViewport(0, 0, Width, Height);
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  gluPerspective(45.0,Width/Height,0.1,100.0);
  glMatrixMode(GL_MODELVIEW);
  InvalidateRect(Handle, nil, False);// DrawGLScene;  Draw the scene.
end;

procedure TForm1.FormDestroy(Sender: TObject);
```

```
begin
 CleanUp(f_Hdc);// Clean up and terminate.
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 f_Hdc := GetDC(handle);
 SetDCPixelFormat(f_Hdc,16,16);// Create a rendering context.
 InitGL(width,height);
{
 // loading the texture from file
  if  OpenPictureDialog1.Execute  then
    begin
      Image1.AutoSize := True;
      Image1.Picture.LoadFromFile (OpenPictureDialog1.FileName);
    end
  else
      halt(1);
}
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
 DrawGLScene();
 wglMakeCurrent(f_Hdc,hrc); //activate the RC
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
       z:=z-0.05 ;
       if z <= -10 then
       begin
        z:=-5;
        filter:=filter+1;
        if (filter>2) then filter:=0;
       end;
          xrot := xrot + 3 ;
          yrot := yrot + 3 ;
  Application.ProcessMessages; //DoEvents
  InvalidateRect(Handle, nil, False);//   DrawGLScene();
 end;
end.
```

برای بهبود برنامه می توان از قطعه کد زیر در روتین InitGL برنامه فوق کمک گرفت .

```
{
 // loading the texture from file
  if  OpenPictureDialog1.Execute  then
    begin
      Image1.AutoSize := True;
      Image1.Picture.LoadFromFile (OpenPictureDialog1.FileName);
    end
  else
      halt(1);
}
```