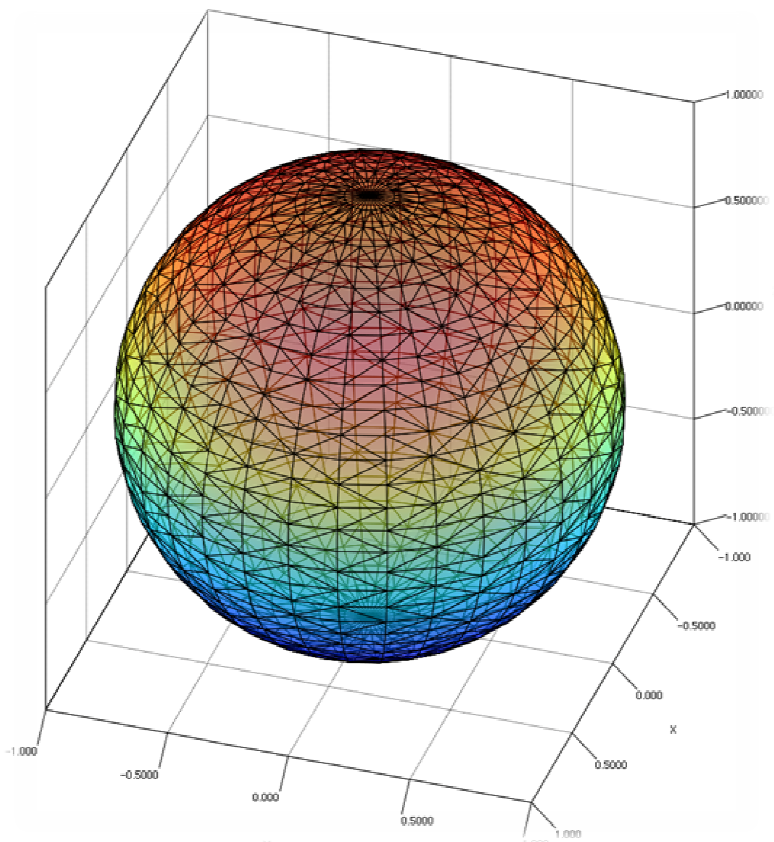
بسم الله الرحمن الرحیم

راهنمای opengl



مقدمه :

ابتدا از نام OpenGL شروع می کنیم که نشان دهنده مفهوم آن نیز می باشد ، OpenGL مخفف کلمات Open source Graphic Library می باشد ، خوب ابتدا مختصری درباره نرم افزار های open source یا همان متن باز توضیح می دهیم :

حتما با نرم افزار های متن باز یا به اصطلاح اصلی open source آشنایی دارید و می دانید که این نرم افزار ها به صورت رایگان در اختیار عموم قرار می گیرند و تهیه کننده گان آنها سورس اصلی برنامه را بیشتر به منظور عیب یابی و کاهش خطا ها و bug های احتمالی برنامه در اختیار برنامه نویسان قرار می دهند تا برنامه نویسان و علاقمندان بتوانند به راحتی کد آن برنامه را ویرایش کرده و تغییرات دلخواه را روی آن انجام دهند (یکی از معروفترین برنامه های open surce سیستم عامل لینوکس است و البته پیشرفت ها و موفقیت های جدید آن را ، که مهمترین عامل این پیشرفت ها همین متن باز بودن لینوکس بوده و است) .

opengl هم همانطور که از نام آن مشخص است جزء نرم افزار متن باز به شمار می رود که یک از مزیت ها و امتیازات آن نیز به شمار می رود ، حال نوبت به graphic library می رسد :

graphic library یا همان کتابخانه گرافیکی به مجموعه کتابخانه ها یا کامپوننت های گفته می شود که برای استفاده از دستورات مخصوص گرافیکی تهیه شده است و با الصاق (include) آن ها به برنامه اصلی (هنگام برنامه نویسی) می توانیم ، از دستورات گرافیکی مجاز برای ایجاد رابطه با بخش گرافیک سیستم استفاده نماییم (از جمله کار هایی که با گرافیک سیستم سر و کار دارد نوشتن بازی ها و اشکال سه بعدی و دو بعدی و کلا برنامه هایی که با مانیتور در ارتباط باشند).

کتابخانه ها یا همان فایل های با پسوند lib که مخفف شده library می باشد ، به منظور آسان تر شدن کار برنامه نویسان در موقع برنامه نویسی ایجاد و تهیه می شوند ، به این صورت که مثلا برنامه نویسی داخل برنامه های خود از تابع خاصی چندین بار (شاید هم صد ها و هزاران بار) به صورت متوالی استفاده می کند ، خوب نوشتن این تابع خاص به این تعداد در کد اصلی برنامه حجم فایل نهایی را بالا برده و البته وقت برنامه نویس را هم خواهد گرفت ، اینجاست که کتابخانه ها یا lib ها وارد عمل می شوند و تابع خاص را داخل خود قرار می دهند تا موقع نیاز به تابع از آن استفاده شود .

این هم از مفهوم graphic library ، حال شما با مفهوم اسم OpenGL یا همان Open source Graphic Library آشنایی کامل را دارید .

پس در اینجا نتیجه می گیریم که OpenGL یک زبان برنامه نویسی نیست ، یک سری کتابخانه از پیش تعریف شده می باشد که کار این کتابخانه ارتباط راحت تر با بخش سخت افزاری گرافیک سیستم می باشد ، برای درک بیشتر این مطلب باید توضیحی هم درباره رابط های نرم افزاری یا همان API ها بدهیم :

برای اینکه بتوانیم با سخت افزار های موجود در سیستم خود مثل کارت شبکه ، کارت گرافیک ، ... به صورت درست و صحیح ارتباط برقرار کنیم (به زبان خود ماشین) احتیاج به رابط هایی داریم که بتوانند این کار را به درستی انجام دهند ، فرض کنید یک فارسی زبان (که زبان دیگری هم نمی داند) با یک انگلیسی زبان (که او هم زبان فارسی را نمی داند) برخورد می کنند و می خواهند با هم صحبت کنند ، به نظر شما چاره کار چیست ؟ خوب اولین و تنها گزینه یک مترجم می باشد که هم زبان فارسی بداند و هم زبان

انگلیسی ، حال آن دو به راحتی می توانند با هم مکاتبه داشته باشند . در مورد کامپیوتر هم دقیقا به همین صورت است ، نرم افزار و سخت افزار کامپیوتر زبان هم دیگر را نمی دانند و احتیاج به یک سری مترجم هایی به نام API دارند .

API ها رابط های نرم افزاری هستند که زبان سیستم (زبان ماشین) و چگونگی ارتباط با سخت افزار را می دانند . به عنوان مثال من برنامه نویس اگر بخواهم یک پیکسل از مانیتور را روشن کنم به API مربوطه فرمان می دهم که این کار را (با زبانی که بلد هستی) برای من انجام بده و مستقیم با سخت افزار صحبت نخواهم کرد ، API مربوطه دستور را گرفته و به سخت افزار می رساند ، به همین راحتی ...

API های مختلفی برای بخش های مختلف سخت افزار وجود دارد ، اما در اینجا و این مقاله API های گرافیکی مد نظر ما می باشد ، که معروفترین آنها OpenGl و DirectX می باشند .

البته سخت افزار های موجود نیز باید از API ها پشتیبانی کنند ، که بعضی شرکت های سازنده سخت افزار و کارت های گرافیکی از API های OpenGL پشتیبانی و همایت می کنند و بعضی هم از DirectX که محصول شرکت ماکروسافت است . هر یک از این واسط های نرم افزاری OpenGl و DirectX معایب و مزایای خواص خود را دارند که در اینجا قصد مقایسه این دو را ندارم .

تاریخچه و نحوه تکامل OpenGl :

سازنده OpenGL یا بهتر بگوییم توسعه دهنده و بهینه کننده آن شرکت Silicon Graphics می باشد (که به اختصار SGI نیز نامیده می شود و یکی از بزرگان صنعت گرافیک کامپیوتری می باشد) ، خود OpenGL توسعه یافته کتابخانه دوبعدی IRIS GL که آن هم محصول شرکت SGL بود ، می باشد . شرکت های دیگری نیز به توسعه این محصول متن باز کمک کرده اند از جمله شرکت D Labs3 که تکنیک و قابلیت سایه زنی را به OpenGL اضافه نمود . در سالهای اخیر کتابخانه های OpenAL و OpenIL نیز به موازات OpenGL در حال پیشروی و کسترش هستند که اولی برای کنترل و مدیریت سخت افزار های صوتی (Audio) و ایجاد صدا های سه بعدی و دومی برای مدیریت و کنترل دستگاه های ورودی (Input) ها ایجاد شده اند .

رقیب اصلی OpenGL همان DirectX محصول شرکت Microsoft می باشد که در سال 1995 وارد عرصه رقابت API ها شد ، در اوایل رقابت بین این دو اپن جی ال کاملا یک سر و گردن از حریف خود جلو بود ، اما کار به همین روال پیش نرفت و شرک ماکروسافت یا استفاده از قدرت تجاری خود توانست از رقیب خود جلو بزند ، البته متن باز بودن OpenGL هم بی تقصیر نبود به گونه ای که ماکروسافت از این ویژگی استفاده می کرد و محصولات خود را تقریبا مشابه با محصولات OpenGL ارائه می داد تا اینکه در تاریخ 2000 میلادی DirectX 8 منتشر شد که امکانات بیشتر و بهتری نسبت به OpenGL داشت ، در این برهه زمانی بود که OpenGL از رقیب خود عقب افتاد ولی همچنان رقابت بین این دو ادامه دارد .

انواع گرافیک رایانه ای[1] :

دو نوع گرافیک رایانه ای وجود دارد: نوع اول گرافیک رستری(Raster) که به گرافیک Bitmap نیز شهرت دارد. گاهی نیز به گرافیک رستری گرافیک پیکسلی هم می گویند. نوع دوم آن گرافیک وکتور(Vector) می باشد که به نوع "برداری" نیز شهرت دارد.

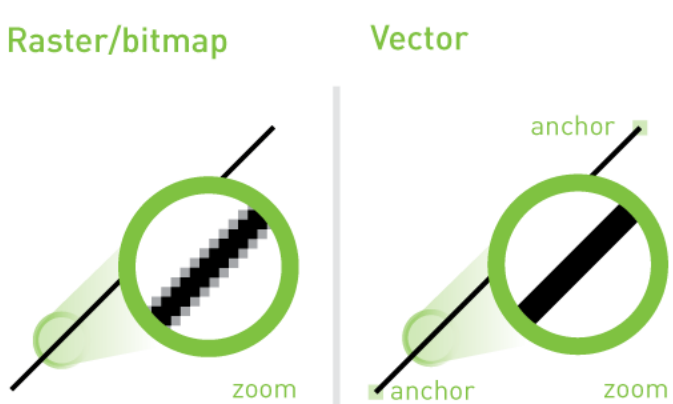
گرافیک پیکسلی یا رستری :

این نوع گرافیک تشکیل شده است از خانه های پیکسلی که دارای درجه مشخصی از یک رنگ می باشند و وقتی این پیکسل ها در کنار یکدیگر با درجه رنگهایی مختلف از هم، قرار می گیرند، تصاویر را تشکیل می دهند. در گرافیک پیکسلی به دلیل اینکه شما با رنگ پیکسل ها سرو کار دارید، قادر به ویرایش، حذف یا ایجاد تصویر یا عکس نیستید. تنها می توانید عکس یا تصاویر دیجیتال را توسط دسترسی به پیسلهای رنگی آن، ویرایش نمایید. آنچه در گرافیک رستری اهمیت دارد، تعداد پیکسل های رنگی در اینچ می باشد، زیرا این تعداد، کیفیت رنگی و وضوع تصویری عکس دیجیتال را مشخص می کند. به تعداد پیکسل های رنگی در اینچ، رزولوشن می گویند. و مقدار رزولوشن، همان مقدار پیکسل ها را در واحد اینچ مشخص می کند که اصطلاحا به آن واحد dpi می گویند (یعنی Dot Per Inch یا تعداد نقاط در اینچ).همین واحد dpi است که باعث می شود وقتی شما یک تصویر رستری را بزرگ نمایی می کنید، کیفیت تصویر پایین تر می آید. زیرا هرچه هم که آنرا بزرگ تر کنید نمی توانید رزولوشن یا تعداد پیکسل های آن را در واحد اینچ افزایش دهید. کیفیت این تصاویر با سایز ارتباط مستقیم دارد.

گرافیک وکتور یا برداری :

این نوع گرافیک، گرافیکی رایانه ای است که با فرمولهای ریاضی سر و کار دارد. از خط ها، منحنی ها و اشکالی که دارای بعد و مختصات ریاضی می باشند و مکان آنها با x و y تعریف می گردد. دارای طول و عرض ریاضی می باشند و رزولوشن آنها با تغییر سایز، تغییر نمی یابد. بنابراین کیفیت تصاویر در هر سایزی یکسان می ماند و این کار از طریق محاسبات ریاضی انجام می شود.این گرافیک یکی از بهترین و پرکاربردترین انواع گرافیک رایانه است، برای کار تصویر سازی استفاده می شود و طراحی آرم ها و گاهی تصویر سازی کتاب های کودک و ... توسط برنامه های وکتوری انجام می شود

در تصویر زیر به صراحت میتوانید تفاوت مابین این دو نوع گرافیک را مشاهده فرمائید :



[1] http://www.howtogeek.com/howto/32597/whats-the-difference-between-pixels-and-vectors/

قوانین نوشتاری در opengl :

- دستورات opengl با پیشوند gl شروع میشود و با یک حرف بزرگ ادامه می یابند مانند دستورهای glBegin() و یا glutDisplayFunc( )
- ثوابت با پیشوند GL_ شروع میشود و تمام حروف بعد از آن حروف بزرگ است مانند: **GL_COLOR_BUFFER_BIT**
- بعضی از دستورات نیز با یک عدد وچندین حرف در انتها برای یادآوری تعداد و نوع آرگومانهای ورودی,خاتمه می یابند . برای مثال در دستور glVertex3fv() ‘3’ به معنای وجود 3 آرگومان ورودی و ‘f’ به معنای اعشاری بودن ورودی ها و ‘v’ به معنای برداری[۲] بودن ورودی ها است .

۲. Vector

انواع داده ها به همراه نحوه تعریف در opengl :

Suffix	Data Type	Typical Corresponding C-Language Type	OpenGL Type Definition
b	8-bit integer	signed char	Glbyte
s	16-bit integer	Short	Glshort
i	32-bit integer	int or long	GLint, Glsizei
f	32-bit floating-point	Float	GLfloat, Glclampf
d	64-bit floating-point	Double	GLdouble, Glclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, Glboolean
us	16-bit unsigned integer	unsigned short	Glushort
ui	32-bit unsigned integer	unsigned int or unsigned long	GLuint, GLenum, Glbitfield

قالب کلی نوشتن برنامه ها در opengl **به زبان** Visual C++ **:**

```
#include <GL/glut.h>

.

.

.

تعریف متغییرها و ثوابت

Void init()

{

مقداردهی اولیه

}

تعریف سایر توابع در صورت نیاز//

int main(int  argc,char **argv)

{

فراخوانی توابع رسم و ارسال پارامتر ورودی به توابع//

Init();

glutMainLoop();

return 0;

}
```

راهنمای برخی از دستورات opengl :

○ **glClearColor (R,G,B,A)** : رنگ پاک کننده جاری صفحه را به مقدار RGBA داده شده تنظیم میکند . که R(قرمز) و G(سبز) و B(آبی) و A(آلفا یا شفافیت) است .

رنگ پیشفرض (0,0,0,0) است که رنگ سیاه است . مقادیر مورد نظر ما بین 0 و 1 است .

○ **glClear**(GLbitfield mask) : بافر همواره باید قبل از ترسیم صفحه پاک شود زیرا مقادیر رنگ هر پیکسل در بافر قرار داده میشود و این دستور نیز همین کار را میکند مقادیر GLbitfield **mask** میتواند شامل مقادیر زیر باشد :

GL_COLOR_BUFFER_BIT	بافر رنگی
GL_DEPTH_BUFFER_BIT	بافر عمق
GL_ACCUM_BUFFER_BIT	بافر انباشتگی
GL_STENCIL_BUFFER_BIT	بافر استنسیل

با فراخوانی این تابع فقط بافرهایی که در آرگومان معرفی شده اند پاک می شوند و بقیه بافرها همچنان حالت قبلی خود رو حفظ می کنند.

○ **glColor{3,4}{b,s,f,i,d,ub,us,ui}**(red,green, blue,alpha) : این تابع برای تعیین رنگ اشیائ روی صفحه به کار میرود و در صورت سه آرگومانه بودن مقادیر RGB را میگیرند و در صورت چهار آرگومانه بودن مقادیر RGBA را میگیرد . برای مثال :

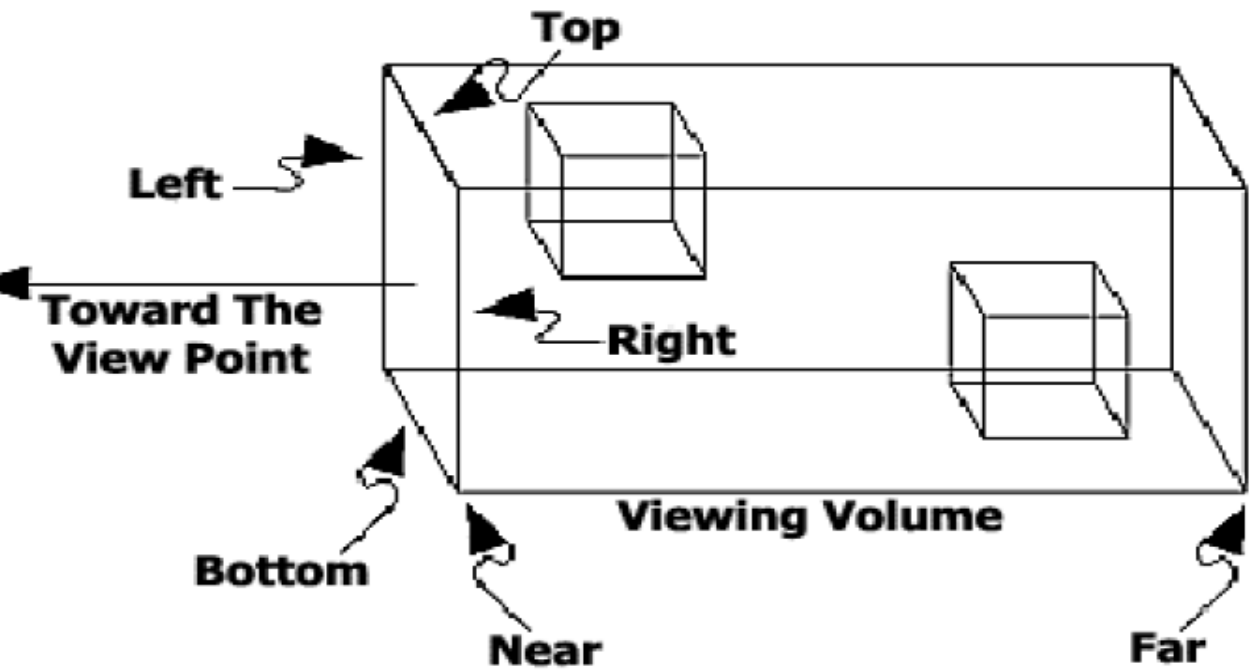
```
glColor3f(0.0, 0.0, 0.0); سیاه
glColor3f(1.0, 0.0, 0.0); قرمز
glColor3f(0.0, 1.0, 0.0); سبز
glColor3f(1.0, 1.0, 0.0); زرد
glColor3f(0.0, 0.0, 1.0); آبی
glColor3f(1.0, 0.0, 1.0); سرخ آبی
glColor3f(0.0, 1.0, 1.0); فیروزه ای
glColor3f(1.0, 1.0, 1.0); سفید
```

نوع دیگر ارسال پارامتر ورودی این تابع به صورت برداری است که در این حالت باید ورودی ها آرایه باشند به صورت زیر :

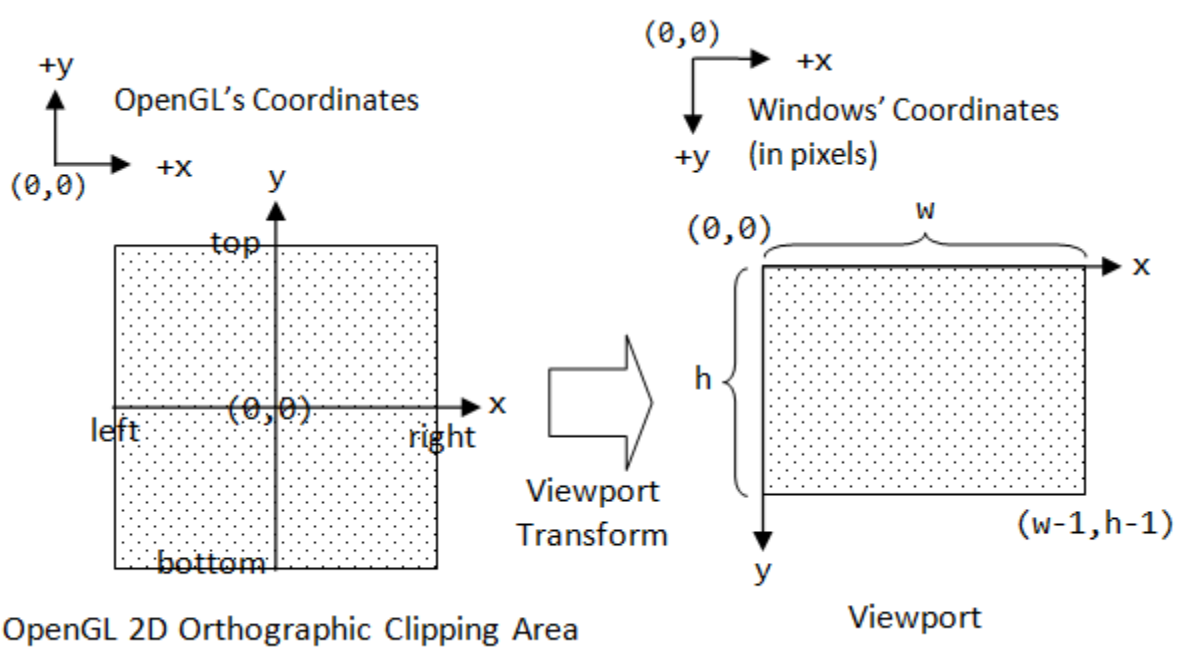
```
GLfloat color_array[] = {1.0, 0.0, 0.0};
glColor3fv(color_array);
```

○ **glOrtho**(GLdouble left, GLdouble right,GLdouble bottom, GLdouble top,GLdouble zNear, GLdouble zFar) :

دستوری که دید اورتوگرافیک حجم را به وجود می آورد در این حالت حجم درحال مشاهده به شکل یک جعبه دیده میشود (شکل زیر) :



آرگومانهای left و right مختصات صفحات عمودی برش چپ و راست را معین میسازد . top و bottom مختصات صفحات افقی برش بالا و پائین را مشخص میکند .و آرگومانهای near و far فاصله با صفحات برش نزدیکتر و دورتر هستند .
دستور **gluOrtho2D(left,right,bottom,top)** با دستور **glOrtho(left,right,bottom,top,-1,1)** برابر است با این تفاوت که دستور اول برای ایجاد سیستم دستگاه مختصات دوبعدی به کار میرود .



○

glBegin(GLenum mode);

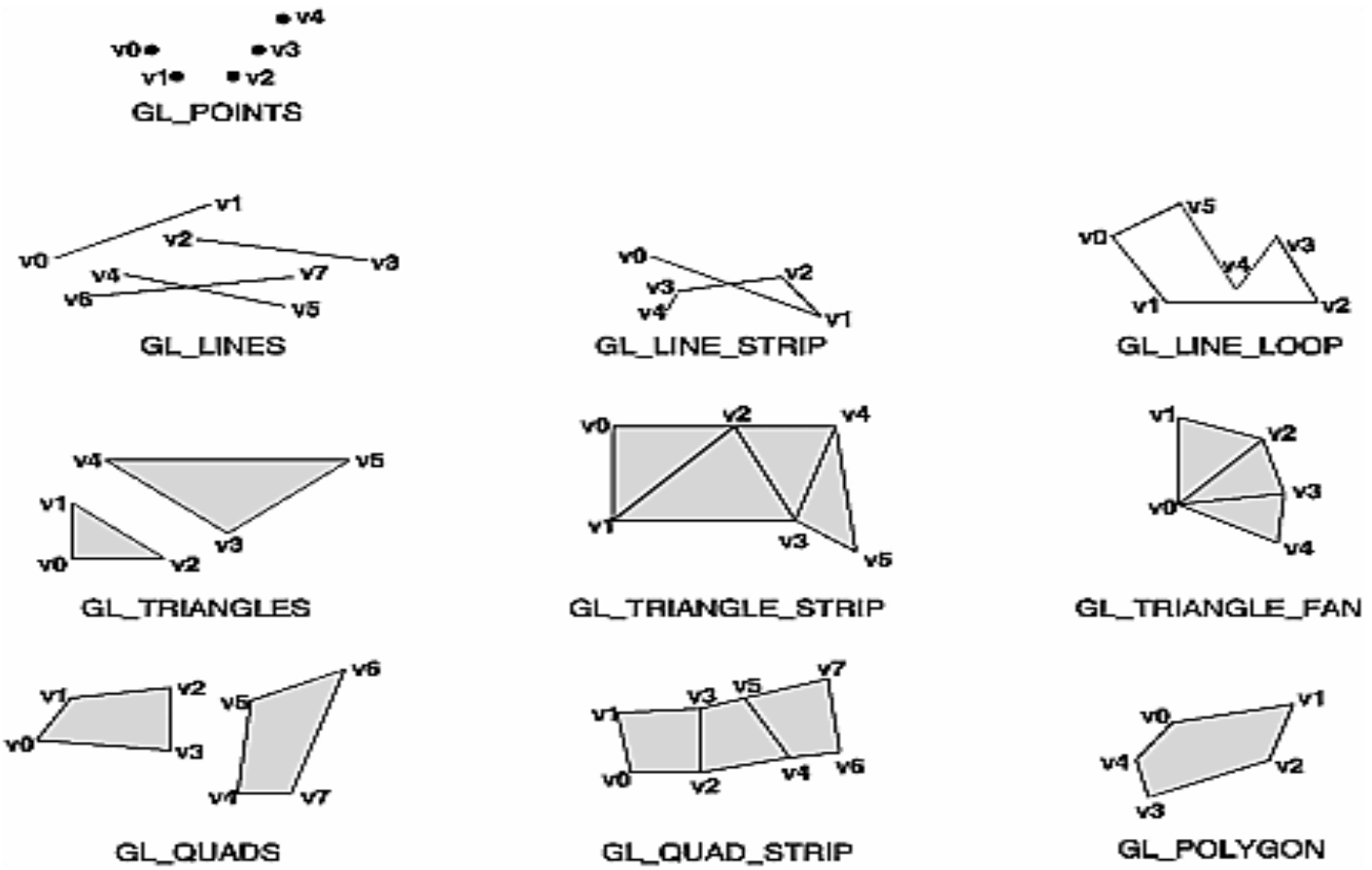
.

.

glEnd(); :

برای شروع و پایان دستورات ترسیمی(اشکال اولیه) که با رئوسشان مشخص میشوند بکار میروند . آرگومان GLenum mode نوع ترسیمی که باید شروع شود را معین میسازد برای مثال : **glBegin(GL_TRIANGLES)** به معنای این است که میخواهیم رسم یک مثلث را شروع کنیم و **glEnd**() به opengl میگوید که رسم مثلث به پایان رسیده است . و ما بین این دو دستور رئوس مورد نیاز برای رسم را وارد میکنیم .مقادیر مجاز برای GLenum mode در دستور **glBegin** به صورت زیر است .

GL_POINTS	با هر راس بصورت یک نقطه برخورد می شود . (حداقل یک راس نیاز است .)
GL_LINES	با هر جفت راس بصورت یک خط مستقل برخورد می شود . (حداقل دو راس نیاز است.)
GL_LINE_STRIP	یک گروه به هم متصل از خطوط را از راس اول تابه آخر ترسیم می کند.
GL_LINE_LOOP	یک گروه به هم متصل از خطوط را از راس اول تابه آخر ترسیم می کند و سپس به اولین باز می گردد .
GL_TRIANGLES	با هر سه راس به صورت یک مثلث مستقل برخورد می کند . (حداقل سه راس نیاز است.)
GL_TRIANGLE_STRIP	یک گروه متصل به هم از مثلث ها را ترسیم می کند . یک مثلث به ازای هر راس پس از دو راس اول تعریف می شود .
GL_TRIANGLE_FAN	یک گروه متصل به هم از مثلث ها را ترسیم می کند . یک مثلث به ازای هر راس پس از دو راس اول تعریف می شود .
GL_QUADS	با هر گروه چهارتایی از رئوس بصورت یک چهار ضلعی مستقل رفتار می کند . (حداقل چهار راس نیاز است.)
GL_QUAD_STRIP	یک گروه متصل به هم از چهار ضلعی ها را ترسیم می کند . یک چهار ضلعی به ازای هر راس پس از جفت اول تعریف می شود .
GL_POLYGON	یک چند ضلعی محدب ترسیم می کند . رئوس ۱ تا n این چند ضلعی را تعریف می کنند . (حداقل سه راس نیاز است .)



- **glVertex{2,3,4}{d,f,I,s}{v}(**input parameters**) :**

یک رأس از شکلی را که باید بین ()**glBegin** و ()**glEnd** رسم شود را ایجاد میکند در opengl تمامی اشکال هندسی با مجموعه ای از رئوس بیان میشوند . باید بدانید که ترتیب ارائه رئوس بسیار مهم است .

برای مثال دستور زیر برای ترسیم یک خط است :

```
glBegin(GL_LINES);
glVertex2i(10,10);
glVertex2i(90,120);
glEnd();
```

- **glFlush**() : برای اطمینان حاصل کردن از ترسیم تمام مواردی که رسم شوند در پایان دستورات ترسیمی از این تابع استفاده میشود . در واقع این تابع بافر را خالی میکند و تمامی دستورات در حال انتظار را به طور سریع اجرا میکند

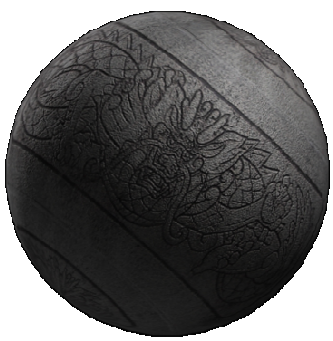
- **glMatrixMode**(GLenum mode) : برای تغییر مکان شئ و همچنین مکان دوربینها از ماتریس ها استفاده میشود و این تابع نیز ماتریس جاری را تعیین میکند مقدار **GLenum mode** میتواند شامل مقادیر زیر باشد :

GL_MODELVIEW [۳] برای انتقال و تغییر مکان و دوران و ... اشیائ استفاده میشود و بصورت پیشفرض این حالت است

GL_PROJECTION [۴] برای حالت دوبعدی استفاده میشود

GL_TEXTURE برای حالتی که میخواهیم از تکستچر(تصویر به جاری رنگ) استفاده میشود

تصویر زیر یک کره ساده است با این تفاوت که دز آن از بافت تکستچر به جای رنگ استفاده شده :



- **glLoadIdentity**() : ماتریس قابل تغییر جاری را به ماتریس واحدتبدیل میکند.(ماتریس واحد ماتریسی است که مولفه های روی قطر اصلی 1 ومابقی 0 هستند.)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **glEnable**(GLenum cap) : برای فعال نمودن یک قابلیت استفاده میشود مانند: **glEnable**(GL_POINT_SMOOTH)
- **glDisable**(GLenum cap) : برای غیرفعال نمودن یک قابلیت استفاده میشود مانند: **glDisable**(GL_POINT_SMOOTH)
- **glIsEnabled**(GLenum cap) : بررسی میکند که آیا یک قابلیت فعال است یا نه و درصورت فعال بودن مقدار GL_TRUE و در صورت غیر فعال بودن مقدار GL_FALSE را برمیگرداند .

[۳] http://www.songho.ca/opengl/files/matrixModelView.zip

[۴] http://www.songho.ca/opengl/files/matrixProjection.zip

در توابع بالا مقدار GLenum cap می تواند ثابت های زیر را به عنوان پارمترهای خود بگیرد :

GL_ALPHA_TEST
GL_AUTO_NORMAL
GL_BLEND
GL_CLIP_PLANEi
GL_COLOR_LOGIC_OP
GL_COLOR_MATERIAL
GL_CULL_FACE
GL_DEPTH_TEST
GL_DITHER
GL_FOG
GL_INDEX_LOGIC_OP
GL_LIGHTi
GL_LIGHTING
GL_LINE_SMOOTH
GL_LINE_STIPPLE
GL_LOGIC_OP
GL_MAP1_COLOR_4
GL_MAP1_INDEX
GL_MAP1_NORMAL
GL_MAP1_TEXTURE_COORD_1
GL_MAP1_TEXTURE_COORD_2
GL_MAP1_TEXTURE_COORD_3
GL_MAP1_TEXTURE_COORD_4
GL_MAP1_VERTEX_3
GL_MAP1_VERTEX_4
GL_MAP2_COLOR_4
GL_MAP2_INDEX
GL_MAP2_NORMAL
GL_MAP2_TEXTURE_COORD_1
GL_MAP2_TEXTURE_COORD_2
GL_MAP2_TEXTURE_COORD_3
GL_MAP2_TEXTURE_COORD_4
GL_MAP2_VERTEX_3
GL_MAP2_VERTEX_4
GL_NORMALIZE
GL_POINT_SMOOTH
GL_POLYGON_OFFSET_FILL
GL_POLYGON_OFFSET_LINE
GL_POLYGON_OFFSET_POINT
GL_POLYGON_SMOOTH
GL_POLYGON_STIPPLE
GL_SCISSOR_TEST
GL_STENCIL_TEST
GL_TEXTURE_1D
GL_TEXTURE_2D
GL_TEXTURE_GEN_Q
GL_TEXTURE_GEN_R
GL_TEXTURE_GEN_S
GL_TEXTURE_GEN_T

- **glutInit(**int *argc, char **argv**) :** این تابع در برنامه های openGL که از کتابخانه glut استفاده می کند و یک بار مقداردهی اولیه می شود .در واقع این تابع کتابخانه glut رو معرفی می کنه.
  این تابع دارای 2 آرگومان هست :
  argc: که اشاره گری به متغیر argc تابع main است . argc تعداد آرگومانهای ارسالی رو ذخیره می کند.
  argv: اشاره گری به متغیر argv تابع main است و argv مقدار آرگومانها رو ذخیر ه می کند .

- **glutInitWindowPosition**(int x,int y) : همانطور که از اسم این تابع پیداست برای مشخص کردن موقعیت پنجره بکار میرود و دارای دو آرگومان x وy است .

- **glutInitWindowSize**(int width,int heigth) : همانطور که از اسم این تابع پیداست برای مشخص کردن اندازه پنجره بکار میرود و دارای دو آرگومان طول وعرض است .

- **glutCreateWindow**("Test Windows") : با فراخوانی این تابع پنجره ما ساخته میشود و در عنوان آن عبارت Test Windows را قرار می گیرد.

- **glutDisplayFunc**(Display Function) : این تابع یک تابع دیگر را فراخوانی می کند ،که ما نوشتیم در واقع آرگومان آن یک تابع است.به اینگونه توابع CallBack گفته میشود .

- **glutMainLoop()** : باعث شروع رخدادهای glut میشود و به محض رسیدن برنامه به این خط تمام تعاریفی که برای ایجاد پنجره کرده ایم را به کار می اندازد و باعث نمایش پنجره ما میشود .این پنجره یک حلقه پردازشی بی نهایت ایجاد می کند و تا زمان بستن پنجره به کار خودش ادامه میدهد و بارها و بارها تمام دستوراتی که با glut نوشته ایم را اجرا می کند این حلقه تمام حرکات مانند موس ،کیبورد رو لحظه به لحظه پردازش و باعث میشود ما تغییرات را در صفحه مشاهده کنیم.

- **glutInitDisplayMode(**unsigned int display mode**)** : این تابع mode نمایشی را برای ما مشخص می کند .

آرگومانهای آن می تواند هر یک یا ترکیبی از مقادیر زیر باشد.

GLUT_RGBA	GLUT_ALPHA
GLUT_RGB	GLUT_DEPTH
GLUT_INDEX	GLUT_STENCIL
GLUT_SINGLE	GLUT_MULTISAMPLE
GLUT_DOUBLE	GLUT_STEREO
GLUT_ACCUM	GLUT_LUMINANCE

مانند :

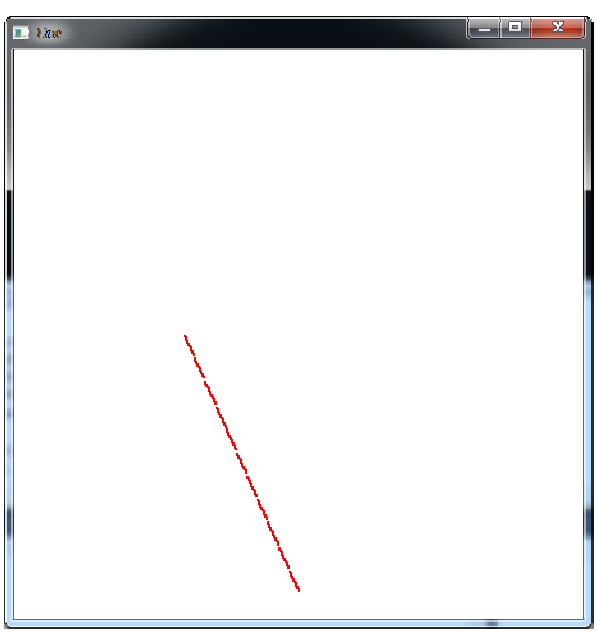
```
glutInitDisplayMode(GLUT_SINGLE,GLUT_RGB);
```

دستور بالا mode پنجره تک بافری در mode رنگی RGB رو مشخص کرده است .

مثال : برنامه رسم یک خط در صفحه :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,100,0,200);

}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0,0.0,0.0);
	glBegin(GL_LINES);
	glVertex2i(30,100);
	glVertex2i(50,10);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);
		glutCreateWindow("Line");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

که خروجی زیر را تولید میکند :

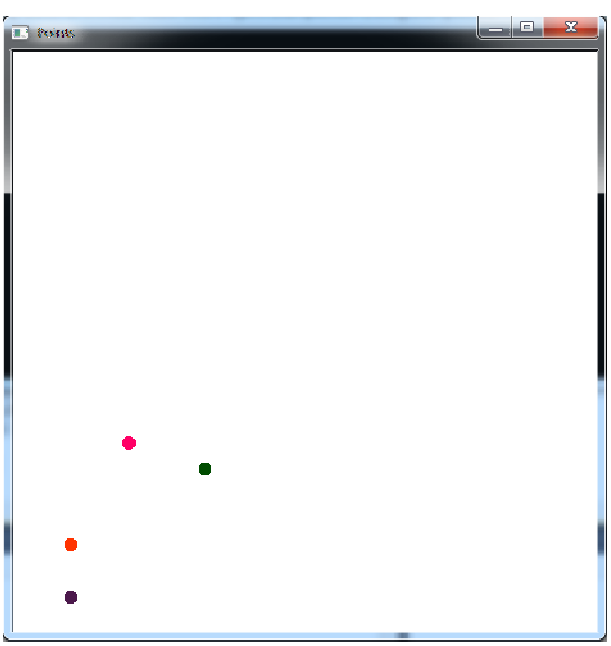


نکته : عرض خط را میتوان به وسیله تابع `glLineWidth(glfloat linewidth)` مشخص نمود .

مثال : برنامه رسم چهار نقطه با گوشه های صاف در صفحه :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,100,0,200);

}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glEnable(GL_POINT_SMOOTH);
	glPointSize(10);
	glBegin(GL_POINTS);
	glColor3f(1.0,0.0,0.4);
	glVertex2i(20,65);
	glColor3f(0.0,0.3,0.0);
	glVertex2i(33,56);
	glColor3f(0.3,0.1,0.3);
	glVertex2i(10,12);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);
		glutCreateWindow("Points");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

که خروجی زیر را تولید میکند :



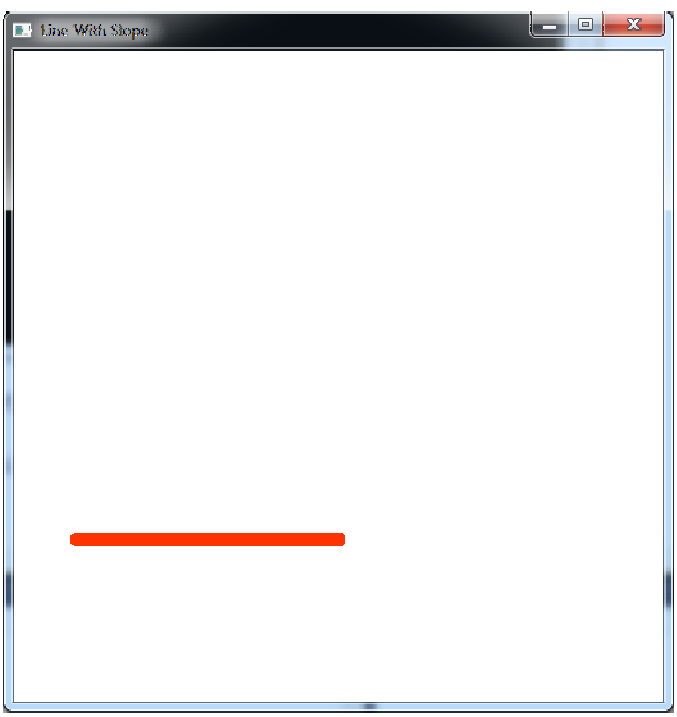
نکته : بزرگی نقطه ها را به وسیله تابع **glPointSize**(glfloat glPointSize) میتوان مشخص نمود .

مثال : برنامه رسم خط به وسیله پیداکردن شیب :

```
#include<gl/glut.h>
#include<math.h>
double round(double x)
 {
 return floor(x + 0.5);
 }
void DDALine( GLint X1, GLint Y1, GLint X2, GLint Y2)
{
       GLint iX, iY, Temp;
       float m, X, Y;
if( (X2-X1) > (Y2-Y1))
             {
                    if( X1 > X2)
                           {
                                  Temp=X1; X1=X2; X2=Temp;
                                  Temp=Y1; Y1=Y2; Y2=Temp;
                           }
       Y=Y1;
       m = (GLfloat) (Y2-Y1)/(X2-X1);
for( iX = X1; iX <= X2; iX ++)
       {
              glBegin(GL_POINTS);
              glColor3f(1.0,0.2,0.0);
              glVertex2i(iX,round(Y));
              glEnd();
              Y += m;
       }
}
else
       {
              if( Y1 > Y2)
       {
     Temp=X1; X1=X2; X2=Temp;
     Temp=Y1; Y1=Y2; Y2=Temp;
    }
  X = X1;
  m = (float) (X2-X1)/(Y2-Y1);
for( iY = Y1; iY <= Y2; iY ++)
       {
              glBegin(GL_POINTS);
              glColor3f(1.0,0.2,0.0);
              glVertex2i(round(X),iY);
              glEnd();
              X += m;
       }
}
       }
void init()
{
       glClearColor(0,0,0,0);
       glMatrixMode(GL_PROJECTION);
       gluOrtho2D(0,100,0,200);

}
void display()
{
       glClear(GL_COLOR_BUFFER_BIT);
       glEnable(GL_POINT_SMOOTH);
       glPointSize(10);
       DDALine(10,50,50,50);
       glFlush();
}
```

```
int main(int argc,char **argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(500,500);
	glutInitWindowPosition(200,200);
	glutCreateWindow("Line With Slope");
	init();
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

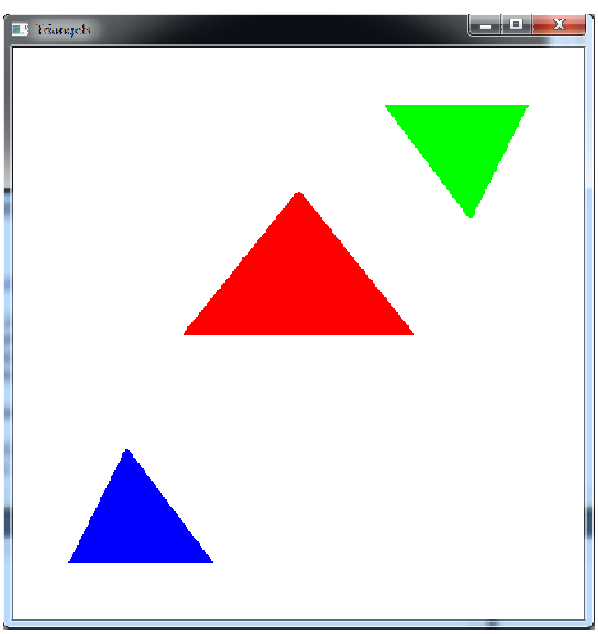
که خروجی زیر را تولید میکند :



مثال : برنامه ای بنویسید که چندین مثلث جدا از هم رسم کند :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(100,100,0,200);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_TRIANGLES);
	glColor3f(0,0,1.0);
	glVertex2f(-0.8, -0.8);
	glVertex2f(-0.6, -0.4);
	glVertex2f(-0.3, -0.8);
	glColor3f(0,1,0);
	glVertex2f(0.8, 0.8);
	glVertex2f(0.6, 0.4);
	glVertex2f(0.3, 0.8);
	glColor3f(1,0,0);
	glVertex2f(-0.4, 0.0);
	glVertex2f(0.0, 0.5);
	glVertex2f(0.4, 0.0);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);

		glutCreateWindow("Triangels");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

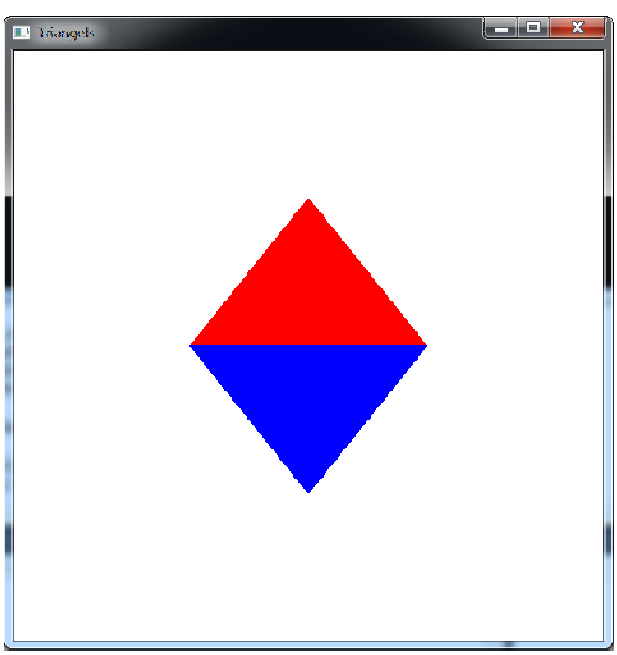
که خروجی زیر را تولید میکند :



مثال : برنامه ای بنویسید که دو مثلث به هم چسبیده(مشترک در یک ضلع) رسم کند :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(100,100,0,200);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glBegin(GL_TRIANGLE_STRIP);
	glColor3f(0,0,1.0);
	glVertex2f(0.4, 0.0);
	glVertex2f(0.0, -0.5);
	glVertex2f(-0.4, 0.0);
	glColor3f(1,0,0);
	glVertex2f(-0.4, 0.0);
	glVertex2f(0.0, 0.5);
	glVertex2f(0.4, 0.0);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);

		glutCreateWindow(Strip Triangels");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

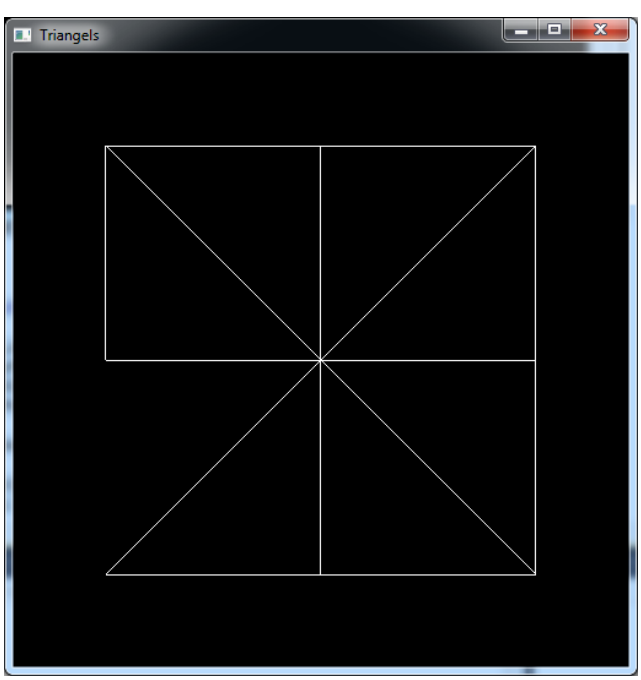
که خروجی زیر را تولید میکند :



مثال : مثلثهایی رسم کنید که در یک راس مشترک باشند :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(100,100,0,200);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
	glBegin(GL_TRIANGLE_FAN);
	glVertex2f(-0.0, 0.0);
	glVertex2f(-0.7, 0.0);
	glVertex2f(-0.7, 0.7);
	glVertex2f(0.0, 0.7);
	glVertex2f(0.7, 0.7);
	glVertex2f(0.7, 0.0);
	glVertex2f(0.7, -0.7);
	glVertex2f(0.0, -0.7);
	glVertex2f(-0.7, -0.7);
glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);

		glutCreateWindow("FAN Triangels");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

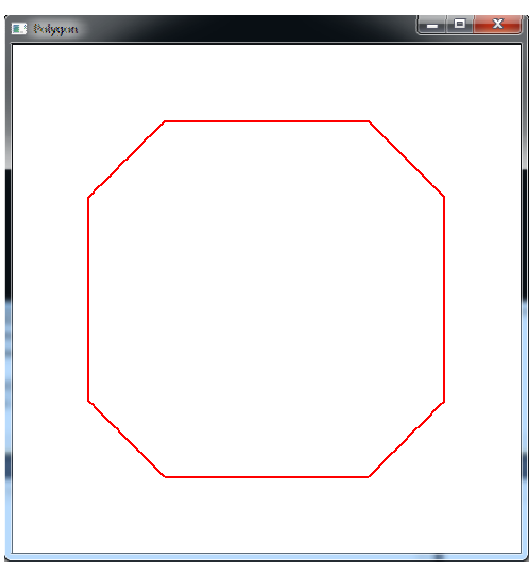
که خروجی زیر را تولید میکند :



مثال : برنامه ای بنویسید که یک 8 ضلعی رسم کند :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(100,100,0,200);

}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0,0.0,0.0);
	glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
	glBegin(GL_POLYGON);
	glVertex2f(-0.7, -0.4);
	glVertex2f(-0.7, 0.4);
	glVertex2f(-0.4, 0.7);
	glVertex2f(0.4, 0.7);
	glVertex2f(0.7, 0.4);
	glVertex2f(0.7, -0.4);
	glVertex2f(0.4, -0.7);
	glVertex2f(-0.4, -0.7);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);
		glutCreateWindow("Polygon");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

که خروجی زیر را تولید میکند :



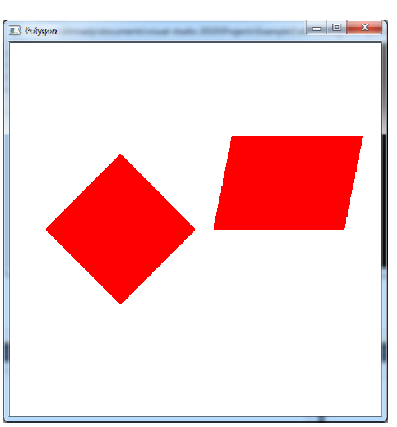
نکته : برای رسم چند ضلعی از دستور GL_POLYGON استفاده میشود که تعداد اضلاع آن توسط رئوس تعریف شده مشخص میشود .

نکته : دستور **glPolygonMode**(face, mode) نوع ترسیم شکل را مشخص میکند که آیا توپر باشد یا توخالی و پارامتر face میتواند مقادیر GL_FRONT_AND_BACK یا GL_FRONT یا GL_BACK را به خود اختصاص دهد و پارامتر mode نیز مقادیر GL_FILL یا GL_POINT یا GL_LINE را میتواند بگیرد که پیشفرض آن GL_FILL است .

مثال : برنامه ای بنویسید که یک لوزی و متوازی الاضلاع را به وسیله 4ضلعی رسم کند :

```
#include<gl/glut.h>
void init()
{
	glClearColor(0,0,0,0);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(100,100,0,200);

}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0,0.0,0.0);
	glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
	glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
	glBegin(GL_QUADS);
	glVertex2f(-0.8, 0);
	glVertex2f(-0.4, 0.4);
	glVertex2f(0.0, 0.0);
	glVertex2f(-0.4, -0.4);
	glVertex2f(0.9, 0.5);
	glVertex2f(0.2, 0.5);
	glVertex2f(0.1, 0.0);
	glVertex2f(0.8, 0.0);
	glEnd();
	glFlush();
}
	int main(int argc,char **argv)
	{
		glutInit(&argc,argv);
		glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
		glutInitWindowSize(500,500);
		glutInitWindowPosition(200,200);
		glutCreateWindow("QUADS");
		init();
		glutDisplayFunc(display);
		glutMainLoop();
		return 0;
	}
```

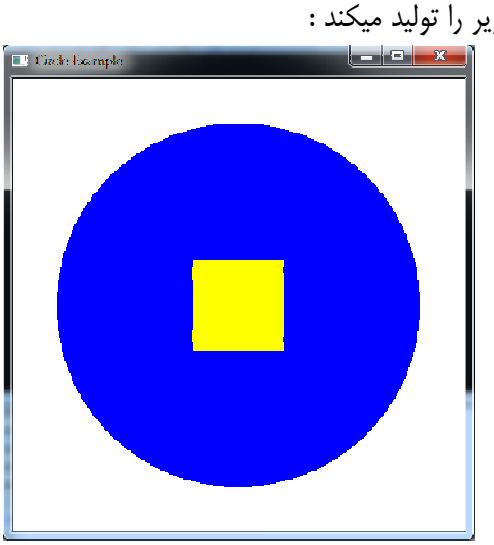
که خروجی زیر را تولید میکند :



مثال : برنامه ای بنویسید که دایره ای توپر به شعاع 9 که با رنگ آبی پر شده و در وسط آن مربعی به ضلع 4 و با رنگ زرد رسم کند:

```
#include <GL/glut.h>
#include <math.h>
int Height=400, Width=400;
#define edgeOnly 0
 void DrawCircle(double radius, int numberOfSides)
 {

 // if edge only, use line strips; otherwise , use polygons
if(edgeOnly)
 glBegin(GL_LINE_STRIP);
 else
 glBegin(GL_POLYGON);
 // calculate each vertex on the circle
 for ( int vertex = 0; vertex < numberOfSides; vertex++ )
 {
 // calculate the angle of current vertex
 // ( vertex # * 2 * PI ) / # of sides
 float angle_c = (float) vertex * 2.0 * 3.14159 / numberOfSides;
 glColor3f(0,0,1); // Blue Color
 // draw the current vertex at the correct radius
 glVertex2f(cosf(angle_c)*radius, sinf(angle_c)*radius);
 }
 // if drawing edge only, then need to complete the loop with first vertex
 if(edgeOnly)
glVertex2f(radius, 0.0);
 glEnd();
 }
 void display(void)
 {
 glClear(GL_COLOR_BUFFER_BIT);
 DrawCircle(0.8, 100);
 glColor3f(1, 1, 0);
 glBegin(GL_QUADS);
 glVertex2f(-0.2, 0.2);
 glVertex2f(-0.2, -0.2);
 glVertex2f(0.2, -0.2);
 glVertex2f(0.2, 0.2);
 glEnd();
 glutSwapBuffers();
 }
 int main(int argc, char **argv)
 {

 glutInit(&argc, argv);

 glutInitDisplayMode(GLUT_DOUBLE);
 glutInitWindowSize(Width, Height);
 glutCreateWindow("Circle Example");
 glutDisplayFunc(display);

 glutMainLoop();
 }
```

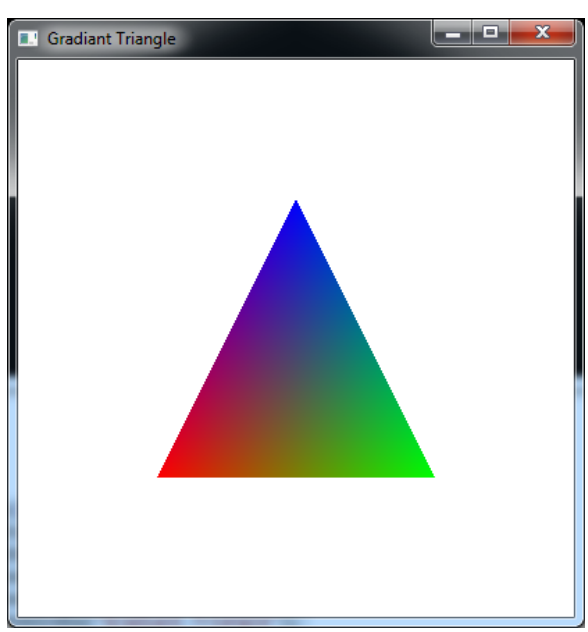
که خروجی زیر را تولید میکند :



مثال : برنامه ای بنویسید که یک مثلث با طیف رنگی ایجاد کند :

```
#include <GL/glut.h>


void Init(void)
{
glClearColor(1.0,1.0,1.0,0.0);
};
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glPushMatrix();
glRotatef(0,0.0,0.0,1.0);
glBegin(GL_POLYGON);
glColor3f(1.0, 0.0, 0.0);
glVertex2f(-0.5, -0.5);
glColor3f(0.0, 1.0, 0.0);
glVertex2f(0.5, -0.5);
glColor3f(0.0, 0.0, 1.0);
glVertex2f(0.0, 0.5);
glEnd();
glPopMatrix();
glutSwapBuffers();
}


int main(int argc, char **argv)
{glutInit(&argc,argv);
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
glutInitWindowPosition(100,100);
glutInitWindowSize(400,400);
glutCreateWindow("Gradiant Triangle");
Init();
glutDisplayFunc(display);


glutMainLoop();
return 0;
}
```

که خروجی زیر را تولید میکند :



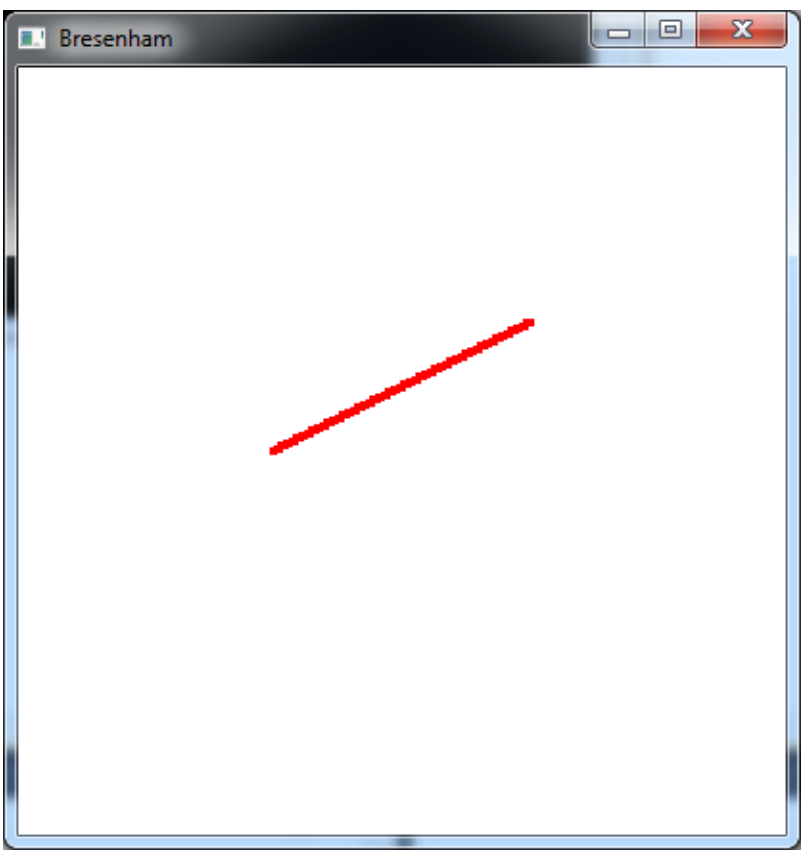
مثال : رسم خط به وسیله الگوریتم برزنهام[۵] :

```
#include<GL/glut.h>
#include <stdlib.h>
#include <math.h>
void init(void)
{
glClearColor(1.0,1.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void setPixel(GLint xCoordinate, GLint yCoordinate)
{
glBegin(GL_POINTS);
glVertex2i(xCoordinate,yCoordinate);
glEnd();
glFlush(); //executes all OpenGL functions as quickly as possible
}
//Bresenham line-drawing procedure for |m| < 1.0
void lineBres(GLint x0, GLint y0, GLint xEnd, GLint yEnd)
{
GLint dx = abs(xEnd - x0);
GLint dy = abs(yEnd - y0);
GLint p = 2 * dy - dx;
GLint twoDy = 2 * dy;
GLint twoDyMinusDx = 2 * (dy-dx);
GLint x,y;
// determine which endpoint to use as start position
if (x0 > xEnd){
x = xEnd;
y = yEnd;
xEnd = x;
}else{
x = x0;
y = y0;
}
setPixel(x,y);
while(x<xEnd){
x++;
if(p<0)
p += twoDy;
else{
y++;
p += twoDyMinusDx;
}
setPixel(x,y);
}
}
void drawMyLine(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0,0.0,0.0);
glPointSize(4.0);
GLint x0 = 100;
GLint y0 = 150;
GLint xEnd = 200;
GLint yEnd = 200;
lineBres(x0,y0,xEnd,yEnd);
}
void main(int argc, char**argv)
{
```

[۵] http://en.wikipedia.org/wiki/Bresenham's_line_algorithm

```
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400,400);
glutInitWindowPosition(0,0);
glutCreateWindow("Bresenham");

init();
glutDisplayFunc(drawMyLine);
glutMainLoop();
}
```

که خروجی زیر را تولید میکند :

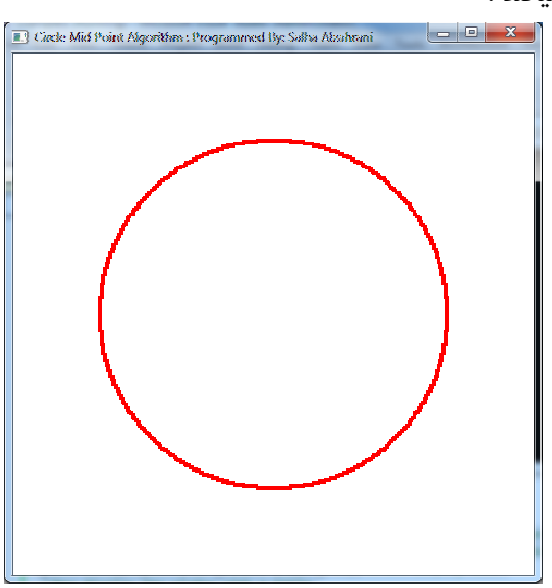


مثال : رسم دایره به وسیله الگوریتم نقطه میانی :

```
#include<GL/glut.h>
class screenPoint
{
private:
GLint x,y;
public:
/* Defult constructor: initializes coordinates position to (0,0) */
screenPoint()
{
x = 0;
y = 0;
}
//Methods
void setCoordinates(GLint xCoordinateValue, GLint yCoordinateValue)
{
x = xCoordinateValue;
y = yCoordinateValue;
}
GLint getx () const
{
return x;
}
GLint gety () const
{
return y;
}
void incrementx ()
{
x++;
}
void decrementy ()
{
y--;
}
};
void setPixel(GLint xCoordinate, GLint yCoordinate)
{
       glBegin(GL_POINTS);
glVertex2i(xCoordinate, yCoordinate);
glEnd();
glFlush(); //process all OpenGL functions as quickly as possible
}
void circleMidPoint(GLint xc, GLint yc, GLint raduis)
{
screenPoint circlePoint;
//initialize value for midpoint parameter
GLint p = 1-raduis;
//set coordinates for top point of circle
circlePoint.setCoordinates(0,raduis);
//function prototype
void circlePlotPoints(GLint, GLint, screenPoint);
//plot the initial point in each quadrant
circlePlotPoints(xc, yc, circlePoint);
//calculate the next point and plot in each octant
while(circlePoint.getx() < circlePoint.gety())
{
circlePoint.incrementx();
if(p<0)
p += 2 * circlePoint.getx() + 1;
else{
circlePoint.decrementy ();
p += 2 * (circlePoint.getx () - circlePoint.gety ()) + 1;
```

```
}
circlePlotPoints(xc, yc, circlePoint);
}
}
void circlePlotPoints(GLint xc, GLint yc, screenPoint circPoint)
{
setPixel(xc + circPoint.getx () , yc + circPoint.gety ());
setPixel(xc - circPoint.getx () , yc + circPoint.gety ());
setPixel(xc + circPoint.getx () , yc - circPoint.gety ());
setPixel(xc - circPoint.getx () , yc - circPoint.gety ());
setPixel(xc + circPoint.gety () , yc + circPoint.getx ());
setPixel(xc - circPoint.gety () , yc + circPoint.getx ());
setPixel(xc + circPoint.gety () , yc - circPoint.getx ());
setPixel(xc - circPoint.gety () , yc - circPoint.getx ());
}
//Initialize OpenGL
void init(void)
{
glClearColor(0.0,0.0,0.0,0.0); //set display-window color to black
glMatrixMode(GL_PROJECTION); //set projection parameters
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void drawMyCircle(void)
{
glClear(GL_COLOR_BUFFER_BIT); //clear display-window
glColor3f(1.0,0.0,0.0); //set pixel color to red
glPointSize(3.0);
GLint xCenter = 150;
GLint yCenter = 150;
GLint raduis = 100;
circleMidPoint(xCenter, yCenter, raduis);
}
void main(int argc, char**argv)
{
glutInit(&argc, argv); //initialize GLUT
glutInitWindowPosition(10,10); //set top-left display-window position
glutInitWindowSize(500,500); //set display-window width and height to 500
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); //set display mode
//Now display the window with a title
glutCreateWindow("Circle Mid Point ");
init(); //execute initialization procedure of OpenGL
glutDisplayFunc(drawMyCircle); //send graphics to display window
glutMainLoop(); //display everything and wait
}
```

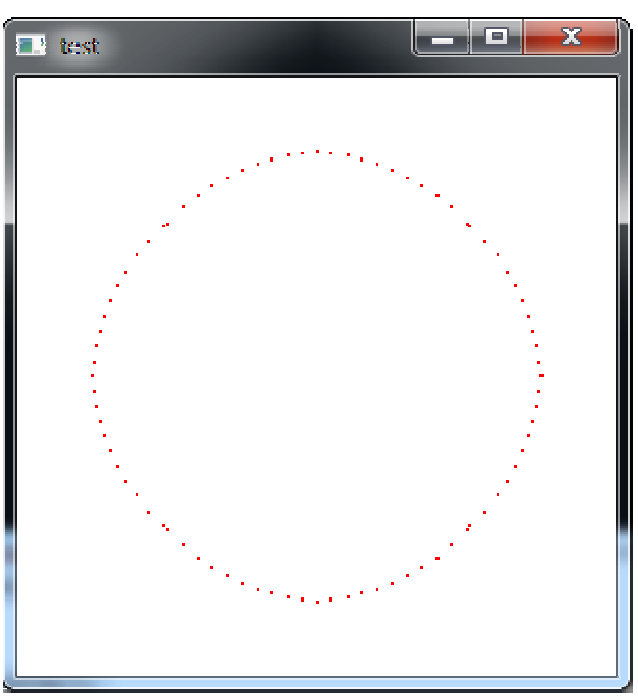
که خروجی زیر را تولید میکند :



مثال : برنامه رسم دایره به وسیله الگوریتم برزنهام :

```
#include<GL/glut.h>
#include<stdlib.h>
#include<math.h>
GLfloat x=10.0 ;
GLfloat y=0.0;
GLfloat x1;
void init()
{
       glClearColor(0,0,0,0);
       gluOrtho2D(-20,20,-20,20);
}
void display()
{
       glClear(GL_COLOR_BUFFER_BIT);
       x=15;
       y=0;
       x1=0;

       glColor3f(1,0,0);
       while(x>y)
       {

  glBegin(GL_POINTS);
  {
         glVertex2f(x,y);
         glVertex2f(-x,y);
         glVertex2f(x,-y);
         glVertex2f(-x,-y);
         glVertex2f(y,x);
         glVertex2f(-y,x);
         glVertex2f(-y,-x);
         glVertex2f(y,-x);
  }

  glEnd();
  y++;
   x1=sqrt((x*x)-2*y-1);
  x=x1;
       }
       glFlush();
}

       int main(int argc,char **argv)
       {
       glutInit(&argc,argv);
       glutInitWindowSize(300,300);
       glutInitWindowPosition(200,200);
       glutCreateWindow("test");
       glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB);
       init();
       glutDisplayFunc(display);
       glutMainLoop();
       return 0;
}
```

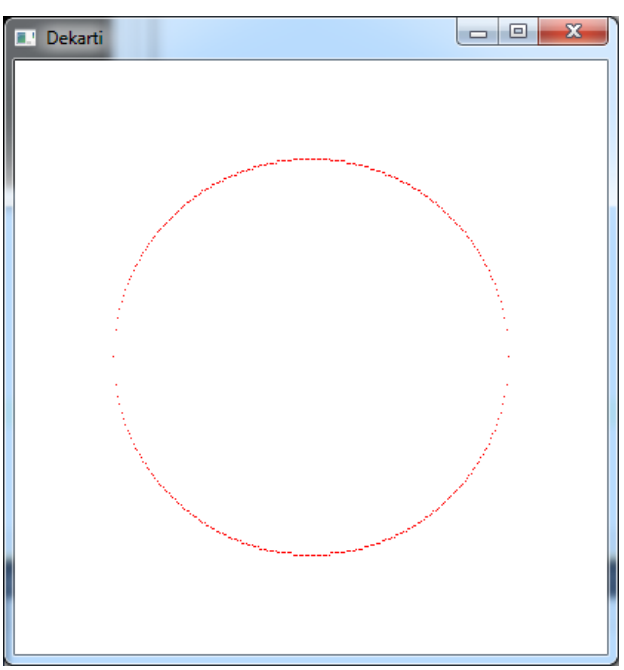
که خروجی زیر را تولید میکند :



مثال : رسم دایره به روش دکارتی :

```
#include<GL/glut.h>
#include <stdlib.h>
#include <math.h>
void init(void)
{
glClearColor(1.0,1.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void setPixel(GLint xCoordinate, GLint yCoordinate)
{
glBegin(GL_POINTS);
glVertex2i(xCoordinate,yCoordinate);
glEnd();
glFlush(); //executes all OpenGL functions as quickly as possible
}

double round(double x)
 {
 return floor(x + 0.5);
 }

void floatCircle(GLint XCenter, GLint YCenter, GLint r)
{
GLint x, y;
GLfloat yr;
GLdouble s;
for( x = -r; x <= r ; x ++)
{
s=r*r -x*x;
yr = sqrt(s);
y = round(yr);
setPixel(x+XCenter, y+YCenter);
 setPixel(x+XCenter, -y+YCenter);
}
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    floatCircle(150,150,100);
    glFlush();
}

void main(int argc, char**argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400,400);
glutInitWindowPosition(0,0);
glutCreateWindow("Dekarti");

init();
glutDisplayFunc(display);
glutMainLoop();
}
```

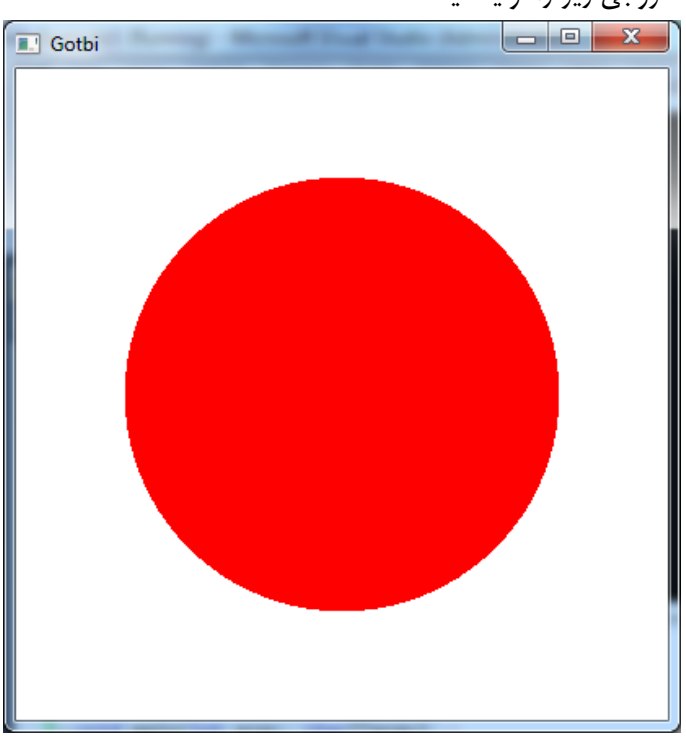
که خروجی زیر را تولید میکند :



مثال : رسم دایره به روش قطبی :

```
#include<GL/glut.h>
#include <stdlib.h>
#include <math.h>
const float PI=3.14;
int circle_points=100;
void init(void)
{
glClearColor(1.0,1.0,1.0,0.0);
glMatrixMode(GL_PROJECTION);
gluOrtho2D(0.0,300.0,0.0,300.0);
}
void setPixel(GLint xCoordinate, GLint yCoordinate)
{
glBegin(GL_POINTS);
glVertex2i(xCoordinate,yCoordinate);
glEnd();
glFlush(); //executes all OpenGL functions as quickly as possible
}

double round(double x)
 {
 return floor(x + 0.5);
 }

void MyCircle(GLfloat centerx, GLfloat centery, GLfloat radius)
{
GLint i;
GLdouble angle;
glBegin(GL_POLYGON);
for (i = 0; i < circle_points; i++) {
angle = 2*PI*i/circle_points;
glVertex2f(centerx+radius*cos(angle),
centery+radius*sin(angle));
}
glEnd();
 }
void display()
{
       glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    MyCircle(150,150,100);
       glFlush();
}

void main(int argc, char**argv)
{
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(400,400);
glutInitWindowPosition(0,0);
glutCreateWindow("Gotbi");

init();
glutDisplayFunc(display);
glutMainLoop();
}
```

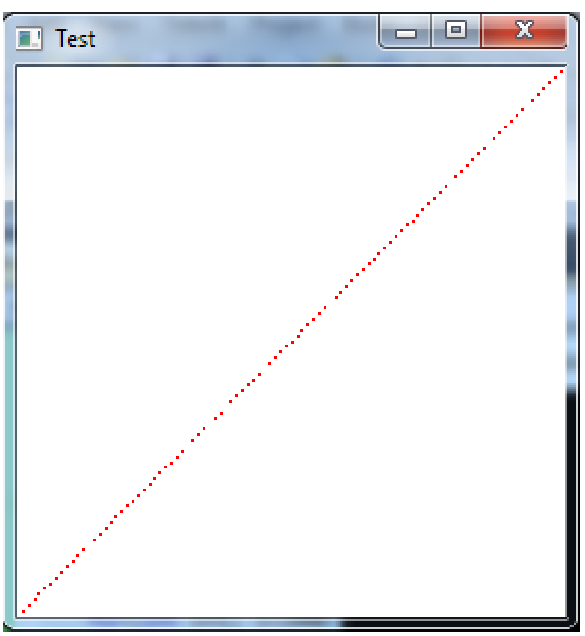
که خروجی زیر را تولید میکند :



مثال : برنامه رسم خط با نقطه:

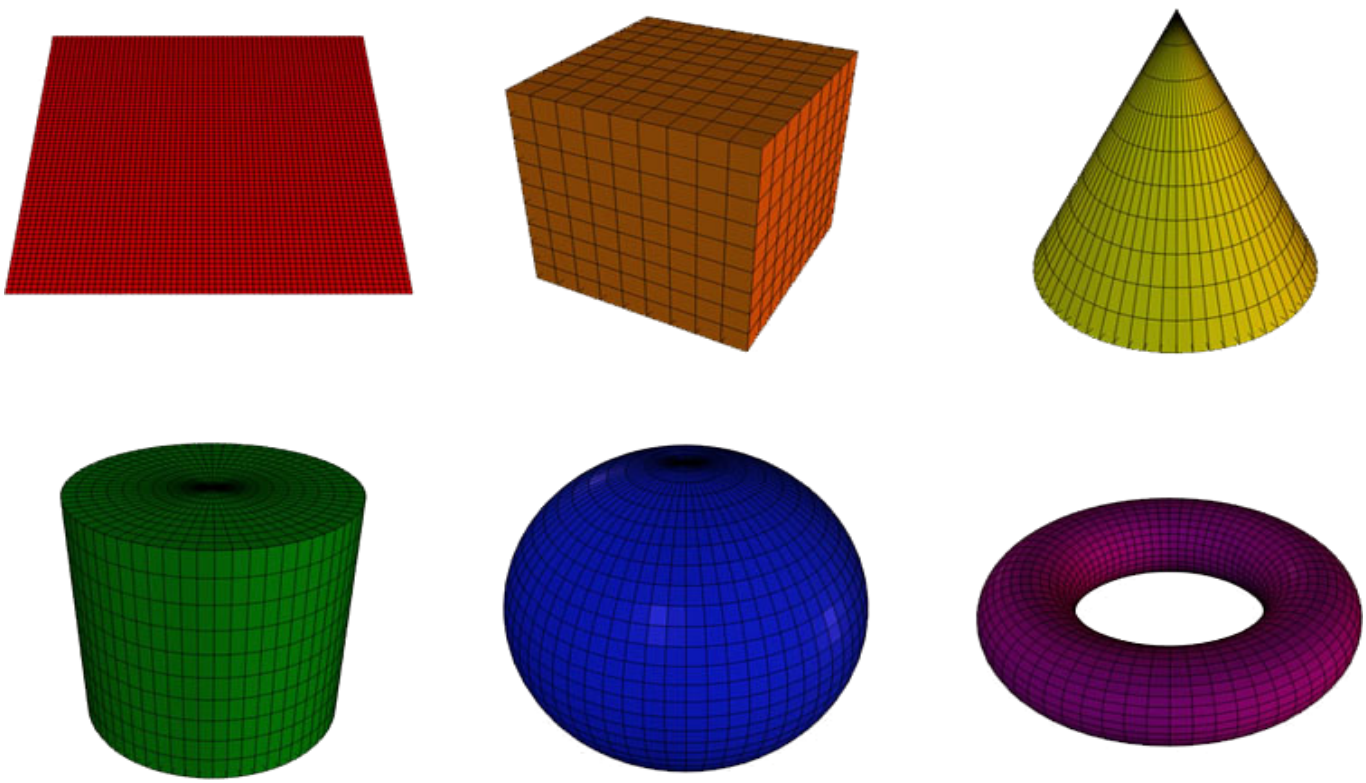
```
#include<GL/glut.h>

void init()
{
	glClearColor(0.1,0.2,0.1,0);
	glutInitWindowPosition(200,60);
	glutInitWindowSize(800,400);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,100,0,200);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0,0.0,0.0);
	for(int i=1; i<2000 ;i++)
	{
	glBegin(GL_POINTS);
	glVertex2i(i,i*2);
	glEnd();

	}
	glFlush();
}
int main(int argc,char**argv)
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutCreateWindow("Test");
	init();
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```

که خروجی زیر را تولید میکند :



گرافیک سه بعدی در opengl :

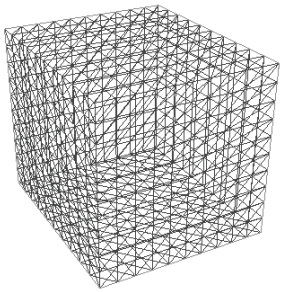
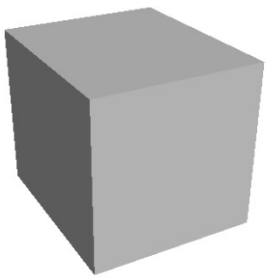
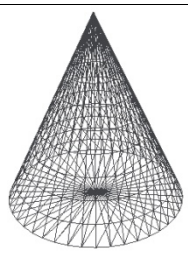
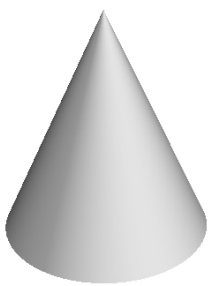
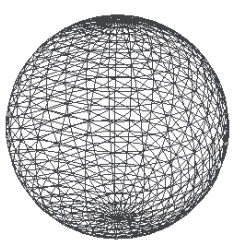
برای رسم اشیای سه بعدی همانند رسم اشیای دو بعدی عمل میکنیم . رسم اشیای سه بعدی بسیار بسیار ساده و آسان است و برای برنامه نویسان OpenGL لذت بخش ترین بخش برنامه نویسی است.

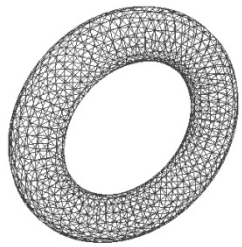
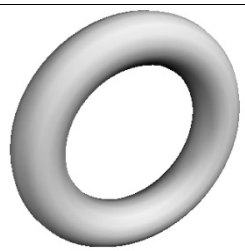


ترسیم اشکال سه بعدی :

قاب سیمی(Wire frame) : شی را با استفاده از خطوط رسم می نماید تا لبه های مرئی چند ضلعی ها را نمایش دهد .

هموار (Smooth) : سطح شی را به صورت همراه رنگ و به صورت هموار نمایش میدهد.و این حالت بالاترین سطح واقع گرایی را داراست.

	این دستور برای رسم مکعب به صورت سیمی است	glutWireCube(GLdouble size)
	این دستور برای رسم مکعب به صورت هموار است	glutSolidCube(GLdouble size)
	این دستور برای رسم مخروط به صورت سیمی است	glutWireCone(GLdouble base, GLdouble height,GLint slices, GLint (stacks
	این دستور برای رسم مخروط به صورت هموار است	glutSolidCone(GLdouble base, GLdouble height,GLint slices, GLint (stacks
	این دستور برای رسم کره به صورت سیمی است	glutWireSphere(GLdouble radius,GLint slices, GLint stacks)
	این دستور برای رسم کره به صورت هموار است	glutSolidSphere(GLdouble radius,GLint slices, GLint stacks)

	این دستور برای رسم تیوب به صورت سیمی است	glutWireTorus(GLdouble innerRadius,GLdouble outerRadius,GLint nsides, GLint rings)
	این دستور برای رسم تیوب به صورت هموار است	glutSolidTorus(GLdouble innerRadius,GLdouble outerRadius,GLint nsides, GLint rings)
	این دستور برای رسم قوری به صورت سیمی است	glutWireTeapot(GLdouble size)
	این دستور برای رسم قوری به صورت هموار است	glutSolidTeapot(GLdouble size)

راهنمای برخی از دستورات :

- **glShadeModel**(GLenum Mode) : یکی از قابلیت های مهم OpenGL ترکیب رنگ های مختلف در یک شئ است. که در این کار هر راس از شئ رنگ متفاوتی میگیرد. این تابع به OpenGL میگوید که اشیای ما فقط یک رنگ بگیرند یا با چند رنگ مختلف رنگ آمیزی شوند. این تابع به جای GLenum Mode میتواند مقادیر زیر را بگیرد :

۱- GL_FLAT : فقط یک رنگ میتوانیم به شئ اختصاص دهیم.

۲- GL_SMOOTH : می توانیم ترکیبی از رنگها را به شی اختصاص دهیم.

در حالت پیش فرض مقدار تابع glShadeModel برابر GL_FLAT است که ما در این صورت فقط یک رنگ میتوانیم به شئ مان اختصاص دهیم.

باید همیشه این نکته را توجه داشته باشید که تابع glShadeModel قبل از بلوک glBegin / glEnd بیاید که در غیر این صورت خطا رخ میدهد.

- **glViewport**(GLint x,GLint y,GLsizei width,GLsizei height) : برای انتخاب ناحیه دید استفاده میشود .

که در آن (x,y) مختصات پیکسلی گوشه سمت چپ پایین دریچه دید نسبت به گوشه سمت چپ پایین پنجره می باشد. height و Width عرض و ارتفاع دریچه دید بر حسب پیکسل می باشد .

- ✓ فرض OpenGL این نیست که شکل را بر روی تمام پنجره بنگارد.
- ✓ پنجره گرافیکی را می توان به چند دریچه کوچک تقسیم کرد.
- ✓ هر تصویر مجزا بر روی یک دریچه قابل رسم است.
- ✓ دریچه ای که رسم اشکال در آن انجام می شود، دریچه دید نام دارد.
- ✓ دریچه دید می تواند کل پنچره باشد و یا هر ناحیه مستطیل شکل از آن باشد.

- **glutSwapBuffers()** : در برنامه نویسی با opengl میتوان از دو بافر استفاده کرد که این دستور بافر پشتی را با بافر جلویی جابجا میکند . تمام رسمها در بافر پشتی اتفاق می افتد و هنگامی که آماده شد جای دو بافر با هم عوض میشوداین کار از پرپر زدن تصویر جلوگیری میکند (به نمایش بدون پرپر تصویر فلیکر نیز میگویند) .

اگر پنجره تک بافر باشد از glFlush استفاده میکنیم .

یادآوری : همانطور که قبلا گفتیم برای مشخص کردن تعداد بافرها از دستور `glutInitDisplayMode` استفاده میکنیم .

- **glutPostRedisplay()** : با استفاده از این تابع به صورت مجازی رخداد refresh را برای پنجره فعال می شود.

زمانی از این تابع استفاده می کنیم که می خواهیم بین یک تابع و تابع display ارتباط برقرار کنیم.

- **glutIdleFunc**(void (*func)(void)) : این تابع , تابع نمایش را مکررا تکرار میکند تازمانی که دیگر هیچ رخدادی دریافت نشود .

- **glutMouseFunc**(void (*func)(int button, int state,int x, int y)) : تابع callback برای ماوس در پنجره جاری را مشخص میکند.تابعی که بعنوان تابع پردازش ماوس به تابع فوق معرفی می شود باید چهار ورودی داشته باشد:

پارامتر اول معین می کند که کدام دکمه ماوس فشرده یا رها شده است که سه حالت دارد :

GLUT_LEFT_BUTTON

GLUT_MIDDLE_BUTTON

GLUT_RIGHT_BUTTON

پارامتر دوم وضعیت دکمه مربوطه، یعنی فشرده شدن یا رها شدن آن است که دو حالت زیر را دارد:

GLUT_DOWN

GLUT_UP

پارامتر سوم و چهارم نیز موقعیت ماوس در مختصات می باشد.

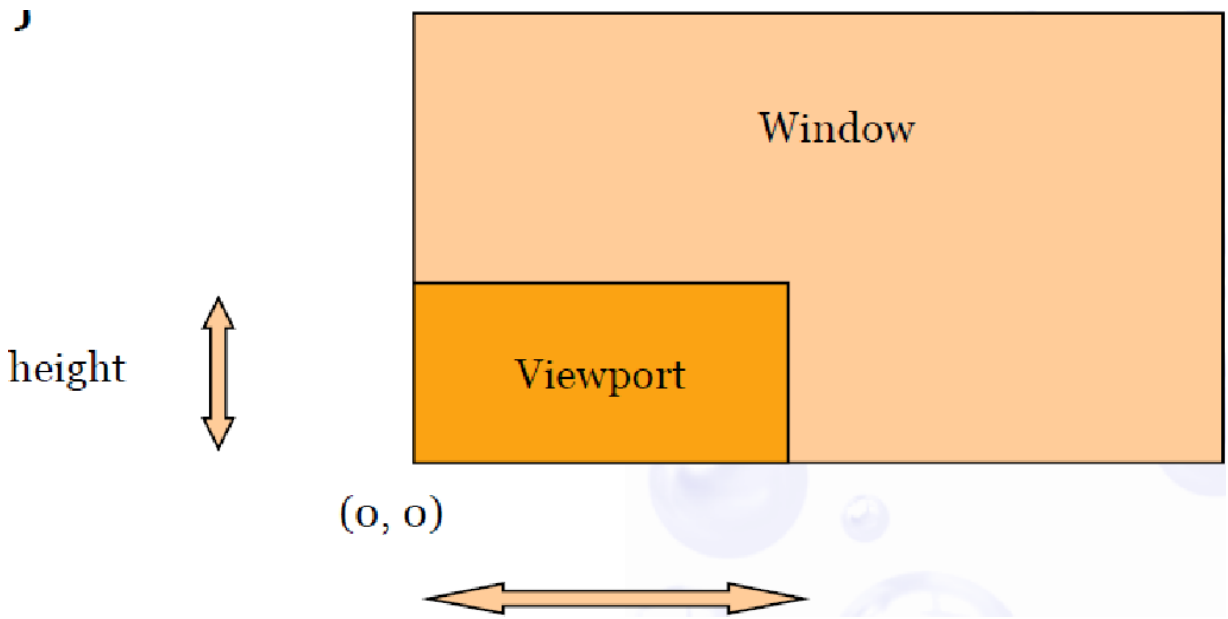
- **glutKeyboardFunc**(void (*func)(unsigned char key,int x, int y)) : تابع callback برای کی بورد در پنجره جاری را مشخص میکند. برای پردازش فشرده شدن کلیدهای معمولی به کار برده می شود.کلیدهای معمولی مثل حروف الفبا و اعداد و همه کاراکترهایی که دارای کد اسکی هستند.باشد در صورتیکه پارامتر تابع فوقNULL از کلیدهای معمولی صرفنظر می شود.

تابعی که بعنوان پارامتر تابع فوق معرفی می شود بایستی سه پارامتر ورودی داشته باشد :

- ✓ کد اسکی کلید فشرده شده
- ✓ مختصات ماوس در هنگام فشرده شدن کلید نسبت به گوشه سمت چپ بالا پنجره (x,y) بر حسب پیکسل

- **glutReshapeFunc**(reshape) :
  - ✓ به هنگام تغییر اندازه پنجره تابع reshape فراخوانی خواهد شد .
  - ✓ شامل اولین رسم پنجره نیز می باشد .
  - ✓ تابع، عرض و ارتفاع پنجره جدید را به بعنوان پارامتر دریافت می کند.

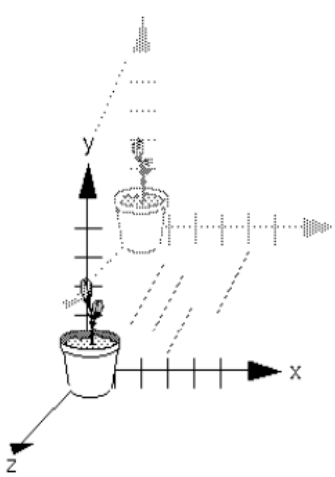
```
void reshape (int width, int height)
{
glViewport (0, 0, width, height);
}
```



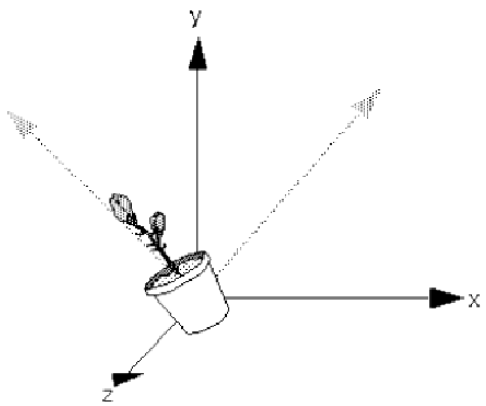
- glPopMatrix() : ماتریس بالای پشته را برداشته و بعنوان ماتریس فعلی قرار می دهد.

- glPushMatrix() : تمام توابع دیگر ماتریس فعلی به غیر از بالای پشته را تخریب می کند.

- **glTranslate**{f,d}(TYPE x, TYPE, y, TYPE z) : برای انتقال یک جسم از نقطه ای به نقطه دیگر استفاده میشود و ماتریس جاری را در ماتریس تبدیل ضرب می کند .

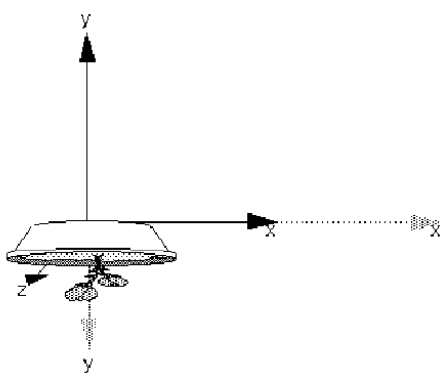


- **glRotate**{f,d}(TYPE angle, TYPE x, TYPE y, TYPE z) : برای دوران یک جسم استفاده میشود .
  - ✓ دوران حول محور دلخواه با زاویه ای به اندازه angle
  - ✓ دوران حول محور x (angle,1.0,0.0,0.0)
  - ✓ دوران حول محور y (angle,o.0,1.0,0.0)
  - ✓ دوران حول محور z (angle,0.0,0.0,1.0)



glRotatef(45.0, 0.0, 0.0, 1.0)

- **glScale**{f,d}(TYPE x, TYPE y, TYPE z) : برای مقیاس یک جسم استفاده میشود .



glScalef(2.0, -0.5, 1.0)

○ **glLight**{f,i}{v}(GLenum light,GLenum pname,const GLfloat *params) :

نور پردازی در OpenGL باعث میشود که صحنه هایی که خلق میشوند طبیعی به نظر برسند. در opengl ما چهار نوع نور پردازی داریم Ambient و Diffuse و Specular و Emmisive .

در تابع بالا آرگومان GLenum light نام منبع نور است و GLenum pname نوع دسترسی به نور است که اگر یکی از مقادیر(GL_DIFFUSE, GL_ AMBIENT, GL_ SPECULAR, GL_ EMMISIVE) باشد بیانگر این است که آرگومان بعدی رنگ نور را پردازش خواهد کرد که این آرایه دارای ۴ اندیس یا عضو است . سه عضو اول آرایه مشخص کننده ی رنگ R,G,B است و عضو آخر مقدار Alpha است که شدت نور رو مشخص می کند . اما اگر آرگومان دوم برابر مقدار GL_POSITION باشد بیانگر این است که آرگومان سوم محل نور را پردازش خواهد کرد و این آرایه باید سه عنصری باشد که مشخص کننده ی X,Y,Z هستند . به مقادیر زیر توجه کنید :

```
GLfloat A[4]={1,1,1,1};
GLfloat Pos1[4]={9,5,1,0};
glLightfv(GL_LIGHT0,GL_DIFFUSE,A);
glLightfv(GL_LIGHT0,GL_POSITION,Pos1);
```

لازم به ذکر است که برای استفاده از منبع نور باید آن را فعال کنیم و همچنین مقادیر GL_LIGHTING و GL_DEPTH_TEST و GL_COLOR_MATERIAL نیز باید فعال گردند .

```
glEnable(GL_LIGHTING);
glEnable(GL_DEPTH_TEST);
glEnable(GL_COLOR_MATERIAL);
glEnable(GL_LIGHT0);
```

○ **glMaterialf{v}**(GLenum face, GLenum pname, GLfloat param) :

glMaterialf :

این توابع پارامترهای ماده را برای مدل نور پردازی تعیین می کنند.

Face : وجه یا وجوهی است که باید به روز در آورده شوند و یکی از سه مقدار , GL_FRONT GL_FRONT_AND_BACK , GL_BACK می تواند باشد .

Pname : در این حالت فقط GL_SHININESS می تواند باشد و param مقداری است که پارامتر یاد شده به آن تنظیم خواهد شد.

glMaterialfv :

آرگومان face در آن همانند تابع قبلی است و pname مقادیر زیر را می تواند داشته باشد :

GL_AMBIENT و GL_DIFFUSE و GL_SPECULAR : همانند آنچه که در قسمت قبل گفته شد می باشند .

GL_EMISSION : در این حالت پارامتر param حاوی چهار عدد خواهد بود که شدت RGBA نور ساتع شده را معین می کنند

GL_SHININESS : در این حالت پارامتر param حاوی چهار عدد خواهد بود که جزء آینه ای RGBA را معین می کنند و در بازه ۰ و ۱۲۸ قرار دارند.

GL_AMBIENT_AND_DIFFUSE : که کاملا واضح است .

GL_COLOR_INDEXES : در این param حاوی سه عدد خواهد بود که اندیس های رنگی محیطی انتشاری و آینه ای را معین می کنند. این سه مقدار و GL_SHININESS تنها مقادیر مواد بکار گرفته شده در این حالت نور پردازی می باشند.

- **glutTimerFunc**(unsigned int msecs, void (*func)(int value), value) :

برای داشتن حرکت در برنامه ها (انیمیشن) ما نیاز به رخدادی داریم که در فواصل زمانی مشخص بیاید برای این از رخداد تایمر استفاده میکنیم.رجیستر کردن این رخداد با تابع glutTimerFunc است ولی این تابع با توابع مشابه 3 تفاوت دارد :

1- فراخوانی این رخداد در دست برنامه نویس است (پارامتر اول زمان آن را مشخص می کند)

2-این تابع یک مقدار به event handler پاس می کند .(پارامتر دوم)

3- این دستور را هم در main می نویسیم و هم در آخر تابع event handler (تابع تایمر)

توجه : در آخر تمامی توابع بالا باید دستور glutPostRedisplay را نوشت تا تغییرات به تابع display رود .

```
glutTimerFunc(2000,timerFunc,0);
```

- **glFog**{f,i}{v}(GLenum pname,GLfloat param) :

تکنیک مه در اکثر تصاویر سه بعدی به خصوص بازی های کامپیوتری مورد استفاده قرار می گیرد. پارامتر GLenum pname میتواند مقادیر GL_FOG_MODE, GL_FOG_DENSITY, GL_FOG_START, GL_FOG_END, GL_FOG_INDEX, GL_FOG_COORD_SRC را بگیرد و پارامتر GLfloat param مقدار ست شده نسبت به آرگومان قبلی را مشخص میکند .
تکنیک fog به طور پیشفرض غیر فعال است شما برای استفاده اول باید آن را فعال کنیدبه وسیله تابع glEnable(GL_FOG) .
به این مثالها دقت کنید :

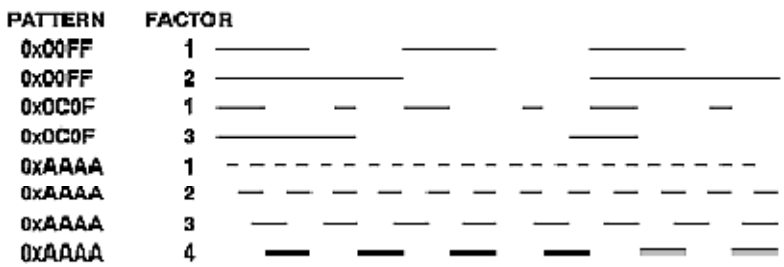
```
GLfloat fogColor[4] = {0, 01, 0, 1.0};
glFogfv (GL_FOG_COLOR, fogColor);
```

یا

```
glFogf (GL_FOG_DENSITY, 0.1);
```

○ **glLineStipple**(GLint factor, GLushort pattern) :

الگوی نقش یک خط را معین می کند . پارامتر GLint factor در هر بیت الگوی نقش زدن ضرب می شود
Pattern : عدد ۱۶ بیتی صحیح ، بطوریکه الگوی بیتی آن (سری صفر و یک ها) معین می سازد
که کدام اجزاء از خط رسم خواهند شد (عدد یک به معنای ترسیم و صفر به معنای عدم ترسیم می باشد).
بیت صفر در ابتدا بکار می رود و تمام الگوی پیش فرض از یک تشکیل شده است.
قبل از بکا ر بردن این تابع باید از دستور **glEnable**(GL_LINE_STIPPLE) برای فعال سازی این حالت استفاده کرد
برای مثال با الگوی **$3F07** که در مبنای دو معادل 0011111100000111 می باشد ، خطی رسم خواهد شد با سه نقطه روشن ، ۵ نقطه خاموش ، ۶ نقطه روشن و ۲ نقطه خاموش. اگر factor مساوی ۲ باشد ، الگو کشیده می شود : ۶ نقطه روشن ، ۱۰ نقطه خاموش ، ۱۲ نقطه روشن و ۴نقطه خاموش . در تصویر زیر مثالهای بیشتری ارائه شده اند . باید خاطر نشا ن کرد که اعداد مبنای ۱۶ در زبان C با 0X آغاز می شوند .

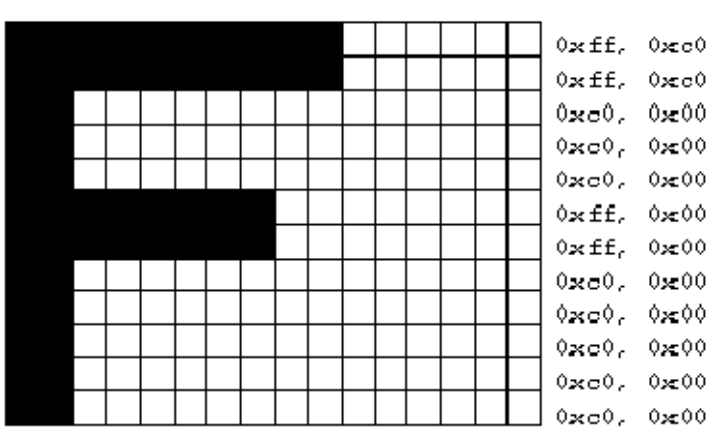


○ **glPolygonStipple**(const GLubyte *mask) :

الگوی نقش یک چندضلعی را معین می کند . mask اشاره گری است به یک الگوی32در32 بیتی . قبل از بکار گیری تابع باید این حالت را بوسیله دستور glEnable(GL_POLYGON_STIPPLE) فعال نمود . بصورت پیش فرض تمام چند ضلعی ها با یک الگوی توپر ترسیم می شوند.

○ **glBitmap**(GLsizei width, GLsizei height,GLfloat xorig, GLfloat yorig,GLfloat xmove, GLfloat ymove,const GLubyte *bitmap) :

بیت مپ ها آرایه ای مستطیلی از نقاط می باشند که در هر نقطه آن یک تک بیت قرار دارد دستورات glBitmap و glRasterPos برای قرار دادن و نمایش یک بیت مپ بر روی صفحه بکار می روند . عموما بیت مپ ها برای نمایش فونت ها ک اربرد دارند. برای مثال کاراکتر F را بصورت زیر می توان طراحی کرد.



داده های بیت مپ ها همواره در قطعاتی که مضربی از 8 بیت هستند ذخیره می شوند . بیت هایی که بیت مپ ها را تشکیل می دهند از گوشه سمت چپ پایین شروع می شوند ، سپس ردیف پایین رسم می شود در ادامه ردیف بالایی و به همین ترتیب.

در دستور بالا بیت مپی را که توسط آرگومان **const GLubyte *bitmap** معین گشته ترسیم می کند . اگر محل قرار گیری آن در صفحه مجاز نباشد ، چیزی رسم نخواهد شد.

GLsizei height , GLsizei width : طول و عرض بیت مپ را مشخص می کنند.

GLfloat yorig, GLfloat xorig : موقعیت مبدا را در بیت مپ مشخص می کنند . مبدا از گوشه پایین ، سمت چپ اندازه گیری می شود.

GLfloat ymove, GLfloat xmove : x و y ایی هستند که به موقعیت raster جاری پس از اینکه بیت مپ رسم می شود ، اضافه می گردند.

- **glRasterPos**{2,3,4}{i,d,f,s}{v}(Glint x,Glint y,...): برای تعیین موقعیت شطرنجی برای عملیات پیکسلی است .

- **glPixelStore{f,i}**(GLenum pname,GLfloat param) : حالت ذخیره سازی پیکسل را تنظیم میکند پارامتر GLenum pname میتواند شامل یکی از مقادیر :

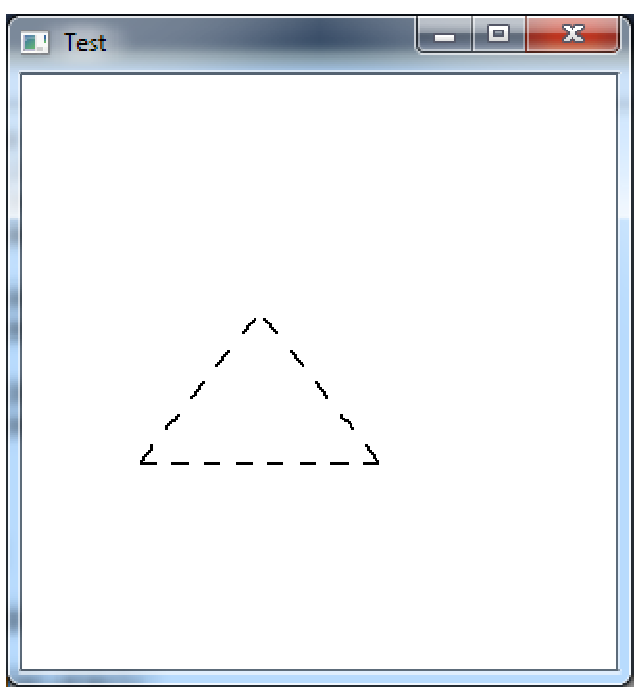
GL_PACK_SWAP_BYTES, GL_PACK_LSB_FIRST, GL_PACK_ROW_LENGTH, GL_PACK_IMAGE_HEIGHT, GLenum pname
GL_PACK_SKIP_PIXELS, GL_PACK_SKIP_ROWS, GL_PACK_SKIP_IMAGES,GL_PACK_ALIGNMENT
GL_UNPACK_SWAP_BYTES, GL_UNPACK_LSB_FIRST, GL_UNPACK_ROW_LENGTH, GL_UNPACK_IMAGE_HEIGHT,
GL_UNPACK_SKIP_PIXELS, GL_UNPACK_SKIP_ROWS, GL_UNPACK_SKIP_IMAGES,GL_UNPACK_ALIGNMENT

باشد و پارامتر GLfloat param نیز مقدار مربوط به پارامتر اول را مشخص میکند .

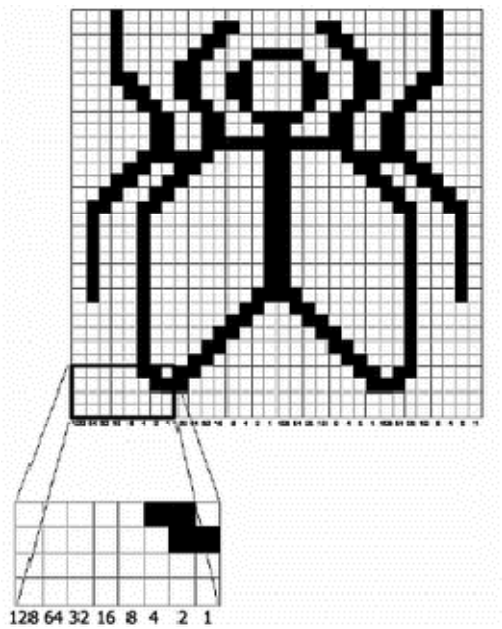
مثال : برنامه ای بنویسید که یک مثلث با خطوط خطچین رسم کند :

```
#include<GL/glut.h>

void init()
{
	glClearColor(0.1,0.1,0.4,0);
	glutInitWindowPosition(200,600);
	glutInitWindowSize(800,800);
	glMatrixMode(GL_PROJECTION);
	gluOrtho2D(0,100,0,200);
}
void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(1.0,0.2,0.8);
	glEnable (GL_LINE_STIPPLE);

   glLineStipple (1, 0x00FF);
	glBegin(GL_LINES);
	glVertex2i(20,70);
	glVertex2i(40,120);

	glVertex2i(60,70);
	glVertex2i(40,120);

	glVertex2i(20,70);
	glVertex2i(60,70);


	glEnd();

	glFlush();
}
int main(int argc,char**argv)
{
	glutInit(&argc,argv);
	glutCreateWindow("Test");
	glutDisplayFunc(display);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);

	init();

	glutMainLoop();
	return 0;
}
```

که خروجی زیر را تولید میکند :



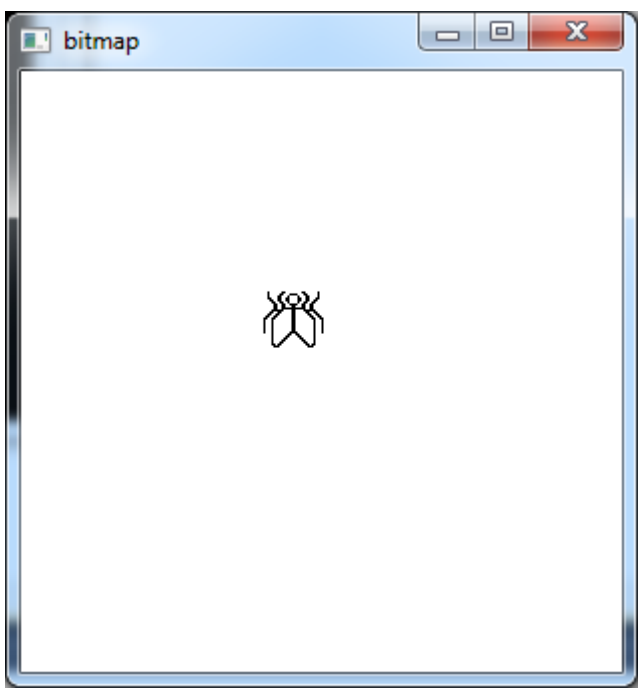
مثال : برنامه ای بنویسید که شکل زیر را با استفاده از مفهوم bitmap رسم کند :




```
#include<GL/glut.h>

void init()
{
	glutInitWindowPosition(100,200);
	glutInitWindowSize(900,900);
	glClearColor(1,1,1,0);
	glMatrixMode(GL_PROJECTION   );
	glOrtho (0,100,0,150,0,1);

}

GLubyte bitShape[128]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x03, 0x80, 0x01, 0xC0, 0x06, 0xC0, 0x03, 0x60,
0x04, 0x60, 0x06, 0x20, 0x04, 0x30, 0x0C, 0x20,
0x04, 0x18, 0x18, 0x20, 0x04, 0x0C, 0x30, 0x20,
0x04, 0x06, 0x60, 0x20, 0x44, 0x03, 0xC0, 0x22,
0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
0x44, 0x01, 0x80, 0x22, 0x44, 0x01, 0x80, 0x22,
0x66,0x01, 0x80, 0x66, 0x33,0x01, 0x80, 0xCC,
0x19,0x81, 0x81, 0x98, 0x0C, 0xC1, 0x83,0x30,
0x07,0xe1,0x87, 0xe0, 0x03, 0x3f, 0xfc,0xc0,
0x03,0x31,0x8c, 0xc0,0x03, 0x33, 0xcc, 0xc0,
0x06,0x64, 0x26, 0x60, 0x0c, 0xcc, 0x33, 0x30,
0x18, 0xcc, 0x33, 0x18,0x10,0xc4, 0x23, 0x08,
0x10,0x63,0xC6,0x08, 0x10, 0x30, 0x0c, 0x08,
0x10, 0x18,0x18, 0x08,0x10, 0x00, 0x00, 0x08};

void display()
{
	glClear(GL_COLOR_BUFFER_BIT);
	glColor3f(0,0,0);
	glRotatef(0,1,2,3);
	glPixelStorei(GL_UNPACK_ALIGNMENT,1);
	glRasterPos2i(40,80);

	glBitmap(31,30,0,0,0,0,bitShape);
	glutSwapBuffers ();
	////////////////
```

```
            glRasterPos2i(20,60);
            glBitmap(31,30,0,0,0,0,bitShape);
            ////////////////
            glRasterPos2i(60,60);
            glBitmap(31,30,0,0,0,0,bitShape);
}
int main(int argc,char**argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE |GLUT_RGB);
        glutCreateWindow("bitmap");
        init();

        glutDisplayFunc(display);

        glutMainLoop();
        return 0;
}
```

که خروجی زیر را تولید میکند :

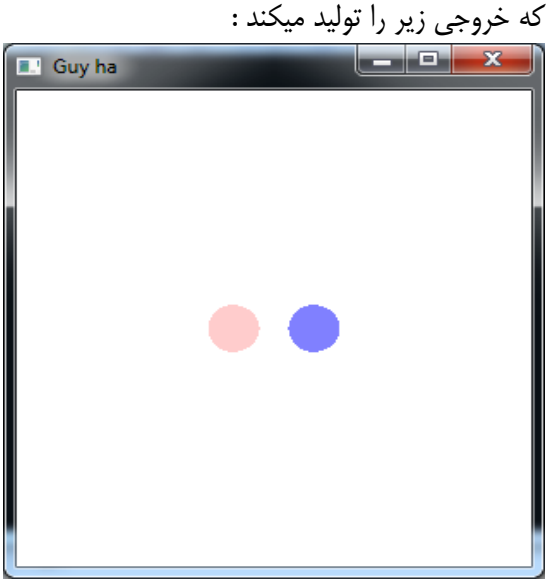


مثال : برنامه ای بنویسید که دو کره را رسم کند و این کره ها به طرف هم حرکت کنند و اگر روی هم افتادند ترکیب رنگشان مشخص شود :

```
#include <windows.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include <gl/glut.h>
#include <math.h>

// Rotation amounts
static GLfloat xRot = 0.0f;
static GLfloat yRot = 0.0f;

// Called to draw scene
void RenderScene(void)
       {
       glClearColor(1,1,1,0);
       // Angle of revolution around the nucleus
       static float fElect1 = 0.0f;
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
       glMatrixMode(GL_MODELVIEW);
       glLoadIdentity();
       glTranslatef(0.0f, 0.0f, -100.0f);
    glClear(GL_COLOR_BUFFER_BIT);

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

       glColor4f(0.0,0.0,1,0.50);
       glPushMatrix();
       glRotatef(fElect1, 0.0f, 1.0f, 0.0f);
       glTranslatef(90.0f, 0.0f, 0.0f);
       glutSolidSphere(10.0f, 15, 15);
       glPopMatrix();



       glColor4f(1,0,0,0.20);
       glPushMatrix();
       glRotatef(fElect1, 0.0f, 1.0f, 0.0f);
       glTranslatef(0.0f - 90, 0.0f, 0.0f);
       glutSolidSphere(10.0f, 15, 15);
       glPopMatrix();



       // Increment the angle of revolution
       fElect1 += 10.0f;
       if(fElect1 > 360.0f)
              fElect1 = 0.0f;

       // Show the image
       glutSwapBuffers();

       }


// This function does any needed initialization on the rendering
// context.
void SetupRC()
       {
       glEnable(GL_DEPTH_TEST);    // Hidden surface removal
       glFrontFace(GL_CCW);        // Counter clock-wise polygons face out
```

```
        glEnable(GL_CULL_FACE);            // Do not calculate inside of jet


    }



void TimerFunc(int value)
    {
    glutPostRedisplay();
    glutTimerFunc(500, TimerFunc, 1);
    }


void ChangeSize(int w, int h)
        {
        GLfloat nRange = 100.0f;

        // Prevent a divide by zero
        if(h == 0)
                h = 1;

        // Set Viewport to window dimensions
    glViewport(0, 0, w, h);


        // Reset coordinate system
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();

        // Establish clipping volume (left, right, bottom, top, near, far)
    if (w <= h)
                glOrtho (-nRange, nRange, nRange*h/w, -nRange*h/w, -nRange*2.0f, nRange*2.0f);
    else
                glOrtho (-nRange*w/h, nRange*w/h, nRange, -nRange, -nRange*2.0f, nRange*2.0f);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        }

int main(int argc, char* argv[])
        {
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
        glutCreateWindow("Guy ha");
         glutReshapeFunc(ChangeSize);
        glutDisplayFunc(RenderScene);
    glutTimerFunc(500, TimerFunc, 1);
        SetupRC();
        glutMainLoop();

        return 0;
        }
```

که خروجی زیر را تولید میکند :

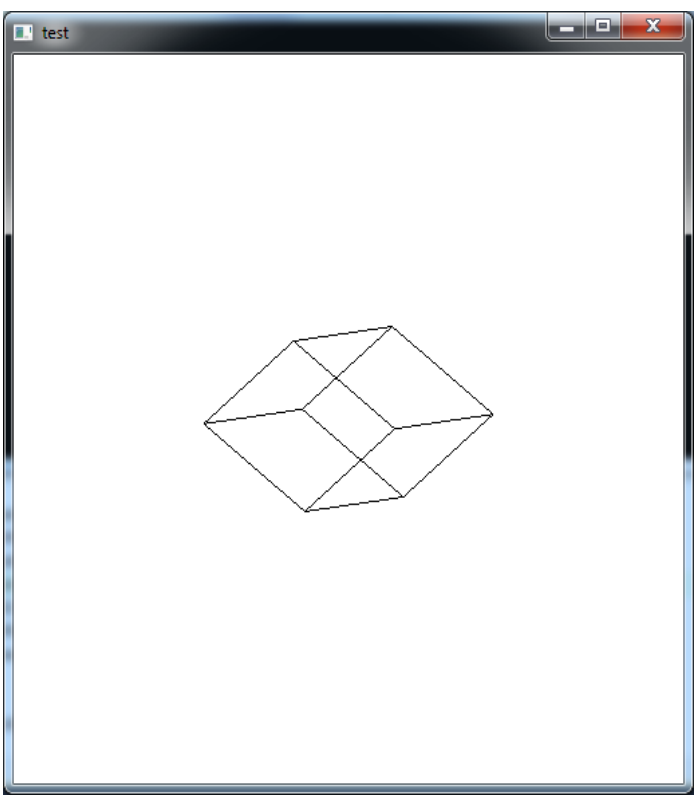


مثال : برنامه ای بنویسید که مکعبی در صفحه ایجاد کند و با کلیک چپ این مکعب چرخش کند و با کلیک دکمه وسط ماوس چرخش متوقف شود :

```
#include <GL/glut.h>
#include <stdlib.h>

static GLfloat spin = 0.0;

void init(void)
{
   glClearColor(0.0, 0.0, 0.0, 0.0);
   glShadeModel(GL_FLAT);

}

void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   glPushMatrix();
   glRotatef(spin, 0.3, 0.2, 1.0);
   glColor3f(1.0, 1.0, 1.0);
   glutWireCube(50);
   glPopMatrix();
   glutSwapBuffers();
}

void spinDisplay(void)
{
   spin = spin + 2.0;
   if (spin > 360.0)
      spin = spin - 360.0;
   glutPostRedisplay();
}

void reshape(int w, int h)
{
       glViewport(0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
glOrtho(-100,100,-150,150,-100,100);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void mouse(int button, int state, int x, int y)
{
   switch (button) {
      case GLUT_LEFT_BUTTON:
         if (state == GLUT_DOWN)
            glutIdleFunc(spinDisplay);
         break;
         case GLUT_MIDDLE_BUTTON :
         if (state == GLUT_UP)
            glutIdleFunc(NULL);
         break;
      default:
         break;
   }
}
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("test");
   init ();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMouseFunc(mouse);
   glutMainLoop();
   return 0;
}
```

که خروجی زیر را تولید میکند :



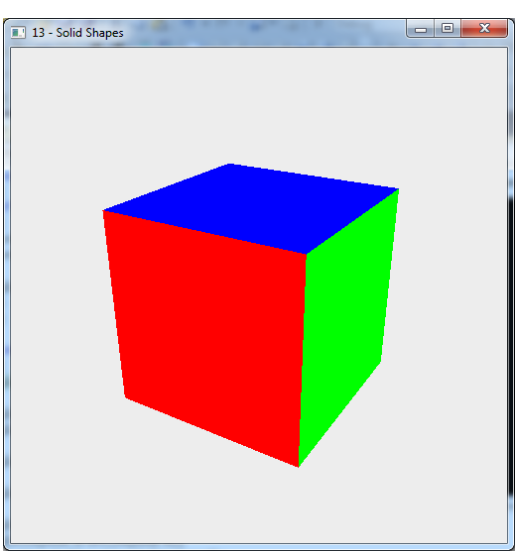
مثال : برنامه ای بنویسید که مکعبی در صفحه ایجاد کند و با حرکت ماوس مکعب نیز چرخش کند :

```
#include <gl/glut.h>

bool fullscreen = false;
bool mouseDown = false;

float xrot = 0.0f;
float yrot = 0.0f;

float xdiff = 0.0f;
float ydiff = 0.0f;

void drawBox()
{
glBegin(GL_QUADS);

glColor3f(1.0f, 0.0f, 0.0f);
// FRONT
glVertex3f(-0.5f, -0.5f, 0.5f);
glVertex3f( 0.5f, -0.5f, 0.5f);
glVertex3f( 0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);
// BACK
glVertex3f(-0.5f, -0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f( 0.5f, 0.5f, -0.5f);
glVertex3f( 0.5f, -0.5f, -0.5f);

glColor3f(0.0f, 1.0f, 0.0f);
// LEFT
glVertex3f(-0.5f, -0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
// RIGHT
glVertex3f( 0.5f, -0.5f, -0.5f);
glVertex3f( 0.5f, 0.5f, -0.5f);
glVertex3f( 0.5f, 0.5f, 0.5f);
glVertex3f( 0.5f, -0.5f, 0.5f);

glColor3f(0.0f, 0.0f, 1.0f);
// TOP
glVertex3f(-0.5f, 0.5f, 0.5f);
glVertex3f( 0.5f, 0.5f, 0.5f);
glVertex3f( 0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
// BOTTOM
glVertex3f(-0.5f, -0.5f, 0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
glVertex3f( 0.5f, -0.5f, -0.5f);
glVertex3f( 0.5f, -0.5f, 0.5f);
glEnd();
}

bool init()
{
glClearColor(0.93f, 0.93f, 0.93f, 0.0f);

glEnable(GL_DEPTH_TEST);
glDepthFunc(GL_LEQUAL);
glClearDepth(1.0f);

return true;
```

```
}

void display()
{
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glLoadIdentity();

gluLookAt(
0.0f, 0.0f, 3.0f,
0.0f, 0.0f, 0.0f,
0.0f, 1.0f, 0.0f);

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

drawBox();

glFlush();
glutSwapBuffers();
}

void resize(int w, int h)
{
glMatrixMode(GL_PROJECTION);
glLoadIdentity();

glViewport(0, 0, w, h);

gluPerspective(45.0f, 1.0f * w / h, 1.0f, 100.0f);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

void idle()
{
if (!mouseDown)
{
xrot += 0.3f;
yrot += 0.4f;
}

glutPostRedisplay();
}

void keyboard(unsigned char key, int x, int y)
{
switch(key)
{
case 27 :
exit(1); break;
}
}

void specialKeyboard(int key, int x, int y)
{
if (key == GLUT_KEY_F1)
{
fullscreen = !fullscreen;

if (fullscreen)
glutFullScreen();
else
{
glutReshapeWindow(500, 500);
```

```
glutPositionWindow(50, 50);
}
}
}

void mouse(int button, int state, int x, int y)
{
if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
mouseDown = true;

xdiff = x - yrot;
ydiff = -y + xrot;
}
else
mouseDown = false;
}

void mouseMotion(int x, int y)
{
if (mouseDown)
{
yrot = x - xdiff;
xrot = y + ydiff;

glutPostRedisplay();
}
}

int main(int argc, char *argv[])
{
glutInit(&argc, argv);

glutInitWindowPosition(50, 50);
glutInitWindowSize(500, 500);

glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);

glutCreateWindow("13 - Solid Shapes");

glutDisplayFunc(display);
glutKeyboardFunc(keyboard);
glutSpecialFunc(specialKeyboard);
glutMouseFunc(mouse);
glutMotionFunc(mouseMotion);
glutReshapeFunc(resize);
//glutIdleFunc(idle);

if (!init())
return 1;

glutMainLoop();

return 0;
}
```

که خروجی زیر را تولید میکند :

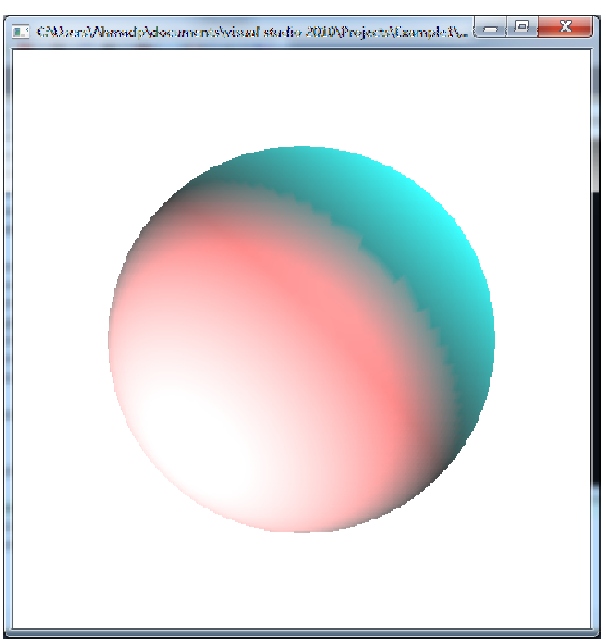


مثال : برنامه ای بنویسید که دوعدد نور ایجاد کند و با زدن کلید 1 نور اول روشن شود و با زدن کلید 2 نور دوم روشن شود و با زدن کلید 3 هر دو نور همزمان روشن شوند :

```
#include <GL/glut.h>

void init(void)
{
       GLfloat ambient[]={1,1,1,1};
       GLfloat diff[]={0,1,1,1};
   GLfloat mat_specular[] = { 1.0, 0.0, 0.0, 1.0 };
   GLfloat mat_shininess[] = { 2.0 };
   GLfloat light1_position[] = { 1.0, 1.0, -.2, 0.0 };
   GLfloat light0_position[] = { -1.0, -1.0, 1.0, 0.0 };
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_SMOOTH);

   glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
   glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
   glLightfv(GL_LIGHT1, GL_POSITION, light1_position);
    glLightfv(GL_LIGHT0, GL_POSITION, light0_position);
   glLightfv(GL_LIGHT1,GL_AMBIENT,ambient);
   glLightfv(GL_LIGHT1,GL_DIFFUSE ,diff);
   glEnable(GL_LIGHTING);

   glEnable(GL_DEPTH_TEST);
}

void display(void)
{
   glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glutSolidSphere (1.0, 200, 50);
   glFlush ();
}





void reshape (int w, int h)
{
   glViewport (0, 0, (GLsizei) w, (GLsizei) h);
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity();
   if (w <= h)
      glOrtho (-1.5, 1.5, -1.5*(GLfloat)h/(GLfloat)w,
         1.5*(GLfloat)h/(GLfloat)w, -10.0, 10.0);
   else
      glOrtho (-1.5*(GLfloat)w/(GLfloat)h,
         1.5*(GLfloat)w/(GLfloat)h, -1.5, 1.5, -10.0, 10.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void key(unsigned char k,int x,int y)
{
       switch(k)
       {
       case '1': glEnable(GL_LIGHT1);
                     glDisable(GL_LIGHT0);
                     glutPostRedisplay();
                     break;
       case '2':
```

```
            glEnable(GL_LIGHT0);
                    glDisable(GL_LIGHT1);
                    glutPostRedisplay();
                    break;
        case '3':
            glEnable(GL_LIGHT0);
                    glEnable(GL_LIGHT1);
                    glutPostRedisplay();
                    break;
        }
}


int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
   glutInitWindowSize (500, 500);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   init ();
   glutKeyboardFunc(key);
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMainLoop();
   return 0;
}
```

که خروجی زیر را تولید میکند :

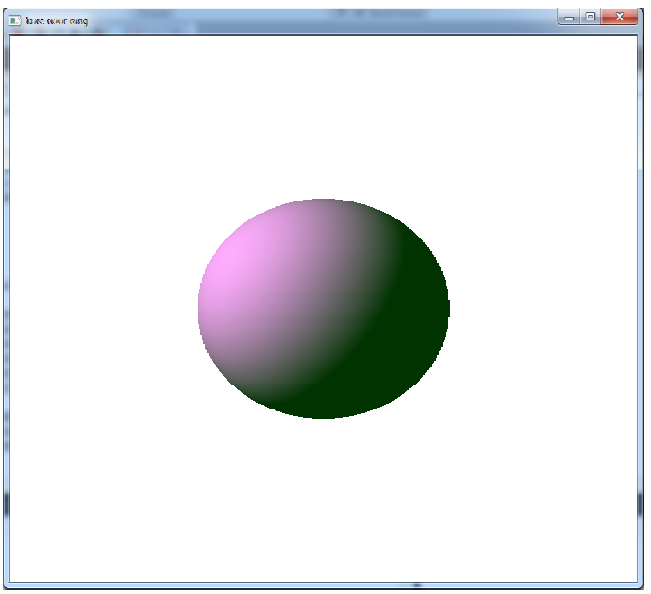


مثال : برنامه ای بنویسید که کره ای در وسط صفحه رسم کند سپس نوری به رنگ سفید از یک طرف تابیده شده و با گذشت زمان هم مکان و هم رنگ نور تغییر یابد :

```
#include<gl/glut.h>

GLfloat A[4]={1,1,1,1};
GLfloat Pos1[4]={9,5,1,0};
GLfloat D=0;
int i=0;
int j=1;
int k=0;
int cheaker=0;
void init()
{
       glClearColor(0,0,0,0);
    glShadeModel(GL_SMOOTH);
    glLightfv(GL_LIGHT0,GL_DIFFUSE,A);

       glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);
       glEnable(GL_COLOR_MATERIAL);
}


void display()
{

  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  glPushMatrix();

  glColorMaterial(GL_FRONT_AND_BACK,GL_AMBIENT);
  glColor3f(0,1,0);
  glEnable(GL_LIGHT0);
glutSolidSphere(0.4,100,100);


  glPushMatrix();

  glRotatef(D,i,j,k);
  glLightfv(GL_LIGHT0,GL_POSITION,Pos1);

  glPopMatrix();


       glutSwapBuffers();
       D+=.3;
       A[1]-=.001;
  glLightfv(GL_LIGHT0,GL_DIFFUSE,A);
}




void main (int argc, char **argv)
{
       glutInit(&argc,argv);
       glutInitDisplayMode(GLUT_DOUBLE);
       glutInitWindowSize(800,600);
       glutInitWindowPosition(100,100);
       glutCreateWindow("kore noor rang");
       init();
```

```
    glutDisplayFunc(display);
    glutIdleFunc(display);
    glutMainLoop();


}
```

که خروجی زیر را تولید میکند :



منابع

Addison Wesley OpenGL SuperBible 5 th Edition

OpenGL Programming Guide: The Official Guide to Learning OpenGL (7 th Edition)(Red Book)

مقدمه ای برOpenGL در C++ مهندس بهرام بهرام بیگی

راهنمای برنامه نویسی سه بعدی OpenGL مهندس وحید نصیری

اصول گرافیک کامپیوتری 1 مهندس جواد مهری

www.opengl.org

www.lighthouse3d.com

www.songho.ca/opengl

www.fly.cc.fer.hr/~unreal/theredbook

www3.ntu.edu.sg/home/ehchua/programming/index.html#OpenGL

www.nurani.ir

www.s1390.mihanblog.com

www.irnt.blogfa.com