جزوه درس گرافیک کامپیوتری

دانشگاه فنی و حرفه ای دختران اهواز

مدرس: زارعی

Marziyeh.zareie@gmail.com

www.m-zareie.ir

به نام خدا

فصل اول

آشنایی با گرافیک کامپیوتری و سخت افزار آن

تعاریف گرافیک کامپیوتری:

گرافیک کامپیوتری به تصاویر و فیلم هایی اطلاق می شود که با استفاده از کامپیوتر ساخته می شوند. این اصطلاح معمولاً به تصویر تولید شده توسط کامپیوتر اشاره دارد که به کمک سخت افزار و نرم افزار تخصصی گرافیکی ایجاد میشود.

کاربردهای گرافیک کامپیوتری:

۱- نمایش اطلاعات (information display): انتقال مفاهیم، نمودار اشکال و پخش تصاویر و بازیها و...

۲- طراحی(Design): طراحی مدار و معماری و (CAD طراحی بر مبنای کامپیوتر) برای قطعات صنعتی

۳- شبیه سازی (Simulation): مدل کردن یک محیط واقعی در یک فضای مجازی بمنظور اهداف مختلف...

۴- واسط کاربری (User Interface): محیط نرم افزارها ، سیستمهای عامل و اینترنت....

فعالیت:

در مورد تاریخچه گرافیک کامپیوتری تحقیق کنید و در کلاس ارائه دهید.

انواع سیستمهای نمایش گرافیکی:

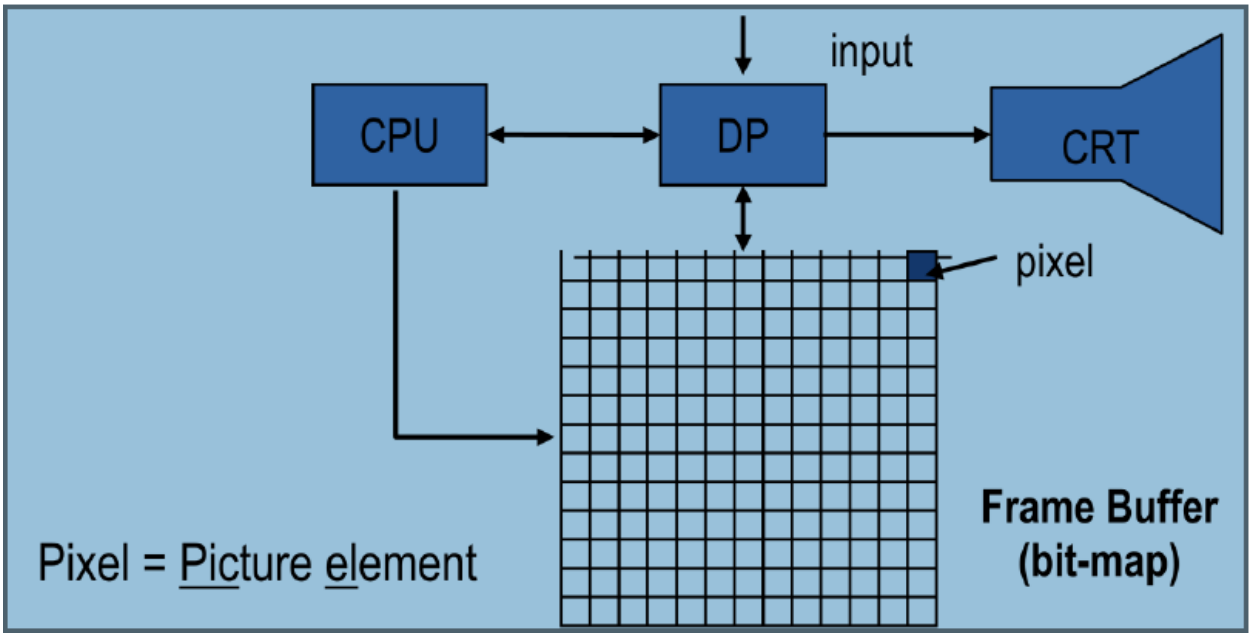
۱- گرافیک رستر Raster

۲- گرافیک برداری Vector

گرافیک raster : نمایش رستر شامل ماتریسی از نقاط مربعی ریزی بنام پیکسل pixel است که برگرفته از عبارت عنصر تصویر (Picture Element) می باشد. پیکسل کوچکترین واحد قابل آدرسدهی در صفحه نمایش است و پایه تمام اعمال گرافیکی محسوب می شود. در این سیستم اطلاعات مربوط به پیکسلها در یک حافظه ماتریسی بنام بافر فریم (Frame Buffer) که به آن بافر نوسازی (Refresh Buffer) هم میگویند، ذخیره می گردند .هر ردیف از پیکسلها، یک خط رستر (Raster line) نامیده می شود. اطلاعات پیکسلها بصورت متناوب نوسازی می گردند و بدین شکل ما نمایش پویا داریم. در بافر فریم به ازای هر پیکسل، با توجه به نوع تصویر (تک رنگ، Grayscaleو رنگی) فضایی در نظر گرفته می شود.

یک بیت برای تک رنگ، ۰ بیت برای Grayscale و ۲۴ یا ۳۲ بیت برای تصاویر رنگی.

اصلی ترین وظیفه یک برنامه گرافیکی تولید و نمایش تصویر در صفحه نمایش است. در صفحه نمایش رستری کل صفحه به مجموعه ای از پیکسلها تقسیم می شود. هر چه تصویر را به نقاط ریزتری تفکیک کنیم وضوح آن بیشتر می شود و کیفیت آن افزایش می یابد و بعلت افزایش تعداد پیکسلها بدیهی است که حجم تصویر نیز بیشتر می شود. بنابراین برای اینکه وضوح تصاویر قابل مقایسه باشند اصطلاحی بنام Resolution بشکل زیر تعریف می شود:



Resolution = تعداد پیکسلهای افقی × تعداد پیکسلهای عمودی

مهمترین مسئله در نمایش پیکسلها رنگ آنهاست . روشهای متفاوتی برای کدگذاری رنگ در کامپیوتر استفاده
میشود. معمولاً رنگ هر پیکسل بصورت یک عدد دودویی یا مجموعه ای از اعداد دودویی کدبندی می شود. بسته به اینکه کد رنگ برای هرنقطه را با چند بیت نمایش دهیم، در بافر فریم همان تعداد بیت به ازای هر پیکسل فضا در نظر گرفته میشود. یعنی با توجه به رزولوشن تصویر و نوع کدگذاری رنگ می توان فضای مورد نیاز برای ذخیره تصویر را بدست آورد.
بطور معمول از نظر رنگ بندی سه نوع تصویر داریم:

- تصاویر تک رنگ: یک بیت به ازای هر پیکسل (۱خاموش یا سیاه و ۱ روشن یا سفید)
- تصاویر Grayscale: ۸ بیت به ازای هر پیکسل (۲۵۶ سطح خاکستری متنوع)
- تصاویر رنگی: ۸ بیت به ازای هر پیکسل / ۲۴بیت به ازای هر پیکسل RGB / ۳۲ بیت به ازای هر پیکسل RGBα

بطور مثال مقدار فضای کاملی که برای ذخیره یک تصویر ۲۴ بیتی با وضوح تصویر ۶۴۰*۴۸۰ (VGA) حدود ۹۲۱۶۰۰ مگا بایت می باشد.
۳*۶۴۰*۴۸۰

در واقع وضوح تصویر ضرب در تعداد بایت (در مثال بالا ۲۴ بیت معادل ۳ بایت است.)

خصوصیات کلی گرافیک رستری:

- عمومی است (Generic) و برای عکسهای واقعی استفاده می شود.
- از تکنیکهای پردازش تصویر استفاده می کنند.
- تبدیلات هندسی در آن ها با افت کیفیت همراه است. (مثل زوم، چرخش و...)
- پیچیدگی مربوط به آنها متناسب با تعداد پیکسلهاست.
- نرم افزارهای نمونه مانند paint و photoshop

گرافیک برداری:

در این سیستم تصاویر به صورت مجموعه ای از اشیای هندسی پیوسته مانند خطوط، منحنی ها، چند ضلعی ها و... ارائه می شود و برای ذخیره تصویر، مجموعه ای از دستورات خطوط ترسیمی در حافظه نگه داری می گردد.

مشخصات گرافیک برداری

- بصورت اشیای گرافیکی، هندسه و رنگ ذخیره شده و برای کاربردهای خاص مانند نمودارها، دیاگرامها و آرمها
- مناسب می باشد.
- از توابع ریاضی و هندسی استفاده می کنند.
- تبدیلات هندسی در آنها بدون افت کیفیت یا از دست دادن اطلاعات صورت می گیرد.
- پیچیدگی مربوط به آنها متناسب با تعداد اشیاء است.
- نرم افزارهای نمونه مثل Corel Draw، Illustrator و چارت ها در آفیس

اجزای سخت افزاری سیستم گرافیک:

- اجزای داخلی
- اجزای خروجی
- اجزای ورودی

واحد های داخلی

واحدهای داخلی مربوط به سیستم گرافیکی شامل پردازنده ، حافظه و کارت گرافیک می باشند که با پردازنده و حافظه در دروس دیگر آشنا شده ایم و در اینجا به معرفی کارت گرافیک می پردازیم.

بمنظور برقراری ارتباط صفحه نمایش با برد اصلی (motherboard) و پردازنده، سخت افزاری بنام کارت گرافیک یا کارت تصویر (Video Adaptor) وجود دارد که سیگنالهای مورد نیاز جهت تشکیل تصویر از طریق این کارت و کابل واسط به نمایشگر ارسال می شود. کارت گرافیک واسط بین پردازنده و صفحه نمایش می باشد.

کارت گرافیک قطعه ای در کامپیوتر است که تصاویر را برای نمایش در صفحه نمایشگر (مانیتور) آماده می کند. در واقع کارت گرافیک وظیفه تفسیر تصاویر و ویدیو برای مانیتور را بر عهده دارد و داده های رایانه شما را به سیگنال هایی تبدیل می کند که بر روی مانیتور قابل نمایش باشند.

کارت های تصویر کامپیوتر های شخصی شامل حافظه ای هستند که اطلاعات مربوط به هر یک ار نقاط یا محل های صفحه نمایش را در خود دارد.

این حافظه ram تصویر نام دارد و با ram کامپیوتر مرتبط است که امکان مراجعه مستقیم به آن از سوی ریزپردازنده را فراهم می کند.

هر یک از انواع کارت تصویر می تواند بیش از یک حالت نمایش داشته باشد. نمایش متن و گرافیک ممکن است بسیار متفاوت باشد. نمایشگر نمی تواند این دو حالت را از هم تشخیص دهد. نمایشگر فقط اطلاعات گرافیکی ارسال شده توسط کارت تصویر را پردازش می کند. برای برنامه نویس و کارت تصویر، این دو حالت مستلزم روش های برنامه نویسی کاملا متفاوت هستند.

در حالت گرافیکی، رنگ هر نقطه صفحه نمایش در یک یا چند بیت نگهداری شده و سپس محتوای ram تصویر، به صفحه نمایش فرستاده می شود. در حالت متنی از روش دیگری استفاده می شود.

در این حالت برای هر یک از محل های صفحه نمایش، کد اسکی یک نویسه در ram تصویر نگهداری می شود.
وقتی که کنترل کننده تصویر اطلاعات را روی صفحه به نمایش در می آورد، الگوی نویسه مربوطه به آن کد اسکی را از تراشه rom روی کارت تصویر بر میدارد و کد را به کادری از نقاط نویسه تبدیل می کند. سپس این الگو به نمایشگر رد می شود و روی صفحه نمایش ظاهر می شود.

انواع کارت گرافیک:

به طور کلی دو نوع کارت گرافیک موجود است:

کارت گرافیک مجتمع یاOn-Board

کارت گرافیک مجزا یاDiscrete

کارت گرافیک مجتمع:(On-Board) این نوع کارت گرافیک به صورت یکپارچه یا مجتمع با پردازنده مرکزی (CPU) ساخته شده است و قطعه ای جداگانه محسوب نمی شود. این نوع کارت گرافیک ارزان است اما نمی توان آن ها را ارتقا داد.

کارت گرافیک مجزا:(Discrete) این نوع کارت گرافیک به صورت قطعه ای جدا بر روی مادربورد نصب می شود. این کارت های گرافیک بیشتر مناسب کسانی است که بازی های رایانه ای انجام می دهند، کارهای ویرایش ویدیو و تصاویر را انجام می دهند یا افرادی که تمایل دارند سیستم آن ها به بهترین شکل ممکن کار کند.

برای بیشتر افرادی که با استفاده از رایانه های خود کارهای استانداردی مانند وب گردی، تماشای ویدیو و شنیدن موسیقی را انجام می دهند، کارت های گرافیک On-Board کافی است. افراد حرفه ای، گیمر ها و ویرایش کنندگان ویدیو باید از یک کارت گرافیک مجزا استفاده کنند تا قدرت تفسیر و پردازش تصاویر را افزایش دهند. بدون استفاده از این کارت های گرافیک اجرای بازی با تأخیر و پرش صورت می گیرد و ویرایش ویدیوها بسیار بسیار طولانی می شود.

برای بیان وضوح تصویر واحدهای مختلفی وجود دارد از جمله : ppi و dpi

Dpi = Dot per Inch یعنی تعداد نقاط در هر اینچ مربع.

PPi = Pixel per Inch یعنی تعداد پیکسل در هر اینچ مربع.

به عبارتی یک تصویر با وضوح 72ppi دارای ۵۱۸۴ پیکسل است.

در ادامه چند فرمت استاندارد تصویر به همراه قابلیت وضوح نمایش ان ها اورده شده است.

استاندارد VGA (Video Graphics Array) : قادر به نمایش تصویری با وضوح 640*480 پیکسل است.

استاندارد SVGA (Super VGA) : قادر به نمایش تصویری با وضوح 800*600 پیکسل است.

استاندارد XGA (Extended Graphic Array) : قادر به نمایش تصویری با وضوح 1024*768 پیکسل است.

WXGA (Wide XGA): قادر به نمایش تصویری با وضوح 1365*768 پیکسل است.

استاندار UXGA (Ultra Extended Graphic Array) : قادر به نمایش تصویری با وضوح ۱۲۰۰*۱۶۰۰ پیکسل است.

امروزه تلویزیون های HDTV قابلیت نمایش تصویری با رزولوشن 1920*1080 پیکسل را دارند.

واحدهای خروجی

این واحد ها به دو دسته تقسیم میشوند:

واحدهای تابشی

واحد های غیر تابشی

نمایشگرهای تابشی: نمایشگرهایی هستند که انرژی الکتریکی را به نور تبدیل می کنند مانند نمایشگرهای پلاسما و...

نمایشگرهای غیر تابشی : نمایشگرهایی هستند که از خود نور تولید نمی کنند بلکه از اثرات تابشی چشمه های نور استفاده کرده و آنها را به الگوهای گرافیکی تبدیل می کنند. یعنی نیاز به منبع نور خارجی دارند مانند LCD

معرفی انواع صفحه نمایش ها و تکنولوژی های مرتبط با آن

CRT

Cathode Ray ، اولین نسل و ساده ترین تکنولوژی صفحه نمایش ویدئویی، نمایشگرهای لامپ اشعه کاتدی می گویند. این تکنولوژی سالیان سال بعنوان تکنولوژی استاندارد تلویزیون CRT استفاده شده است. این نمایشگرها شامل یک لامپ تصویر می باشند که این لامپ تصویر یک لامپ خلاء بوده است.

اما مطمئنا اولین نامی که درمورد صفحات نمایش به ذهن شما می رسد،LCD است. در واقع این نوع از صفحات نمایش نسل جدیدی هستند که جایگزین صفحه نمایش CRT شدند و سال هاست که در لوازم خانگی از قبیل تلویزیون استفاده می شوند. اما علاوه بر تلویزیون، کاربرد این نوع از صفحه نمایش نیز در گوشی های موبایل، تبلت و دیگر دستگاه ها رسوخ نموده، به طوری که تا امروز نام آن در مقابل نوع صفحه نمایش بسیاری از گوشی های هوشمند و تبلت ها به چشم می خورد. از آن جایی که کیفیت رنگ و وضوح صفحه نمایش LCD پایین است، با گذشت زمان انواع مختلفی از این صفحه نمایش، از قبیل پلاسما، TFT ، IPS و ... جهت بهبود آن معرفی شدند که در ادامه به معرفی آنها می پردازیم.

نمایشگرهای پلاسما:

نمایشگرهای پلاسما از دو صفحه شیشه ای موازی تشکیل شده اند که مجموعه ای از الکترودها بصورت نوارهای افقی برروی یکی از صفحات شیشه ای و مجموعه ای دیگر بصورت نوارهای عمودی برروی صفحه ای دیگر قرار داده شده اند. با برقراری ولتاژ بین الکترودهای عمودی و افقی گاز موجود بین فضای این دو صفحه، به پلاسمای درخشانی از یون ها و الکترونها تبدیل می گردد. این گاز

معمولاً ترکیبی از گاز نئون با چند ماده دیگر می باشد .الگوی تصویر در بافر فریم ذخیره شده و اختلاف ولتاژ ایجاد شده باعث Refresh وضعیت پیکسلها در حد فاصل دو الکترود می گردد.

نمایشگرهای LCD

این نوع نمایشگرها که امروزه بسیار مورد استقبال قرار گرفته اند دارای کیفیت بالا و سایزهای متنوع و قطر کم می باشند. در این نمایشگرها از ترکیب ماده ای مخصوص که دارای ساختار مولکولی قطبی است و حالتی بین جامد و مایع دارند، تشکیل شده است. این ماده خاص بلور مایع (کریستال مایع) نام دارد که با اندکی حرارت به مایع تبدیل می شوند .نوعی از این مواد وجود دارند که ملکولهای آن پیچ خورده اند و تحت تأثیر جریان برق، پیچ خوردگی آنها کم یا زیاد می شود .یعنی اندازه زاویه چرخش این بلورهای مایع به ولتاژ برق بستگی دارد .با استفاده از این خاصیت می توان میزان گذردهی نور از مولکولهای بلور مایع را بکمک جریان برق کنترل کرد.

نمایشگرهای LED

صفحه نمایش های LED یا Light-Emitting Diodes مدت زمانی است که وارد بازار شده اند که نسبت به صفحه نمایش های LCD هزینه های ساخت بالاتری دارند و در نتیجه از کیفیت رنگ بهتر، روشنایی بیشتر و همچنین مصرف کمتر باتری برخوردارند. این صفحه نمایش با گذشت زمان به انواع مختلفی تقسیم شده است.

لامپ‌های LED در تمام قسمت‌های صفحه نمایش لپ‌تاپ پخش شده‌اند و به همین دلیل نه تنها تصویر شفاف‌تر و واضح‌تر است؛ بلکه کمتر باعث خستگی چشم می‌شود.

TFT

TFT اولین نوع از صفحه نمایش های LCD می باشد که با مدیریت بهتر پیکسل، میزان وضوح و کیفیت تصویر پخش شده در آن را افزایش می دهد .TFT مخفف کلمه Thin Film Transistor می باشد و به معنی ترانزیستور لایه نازک است و به گونه ای طراحی شده است که پیکسل ها به صورت افقی عمودی در کنار هم قرار گرفته اند. هر پیکسل به یک خازن و یک ترانزیستور متصل

است، این طراحی اجازه می دهد هر پیکسل به طور مستقل شارژ شود و شارژ آن حتی بعد از رفرش شدن برای تصویر جدید باقی بماند.

IPS

IPSیکی از تحسین برانگیزترین نوع صفحات نمایش LCD است و نسخه پیشرفته تر TFT به حساب می آید و قیمت بالاتر و کیفیت بهتری نسبت به نسخه قبلی خود دارد، همچنین برخلاف نسخه TFTدر زیر نور مستقیم آفتاب هم قابل استفاده می باشد. از اصلی ترین نقاط ضعف این نوع از صفحه نمایشLCD ، می توان به عدم توانایی در تولید رنگ مشکی عمیق و باکیفیت نام برد. به طوری که رنگ مشکی تولید شده، بیشتر تمایل به رنگ خاکستری تیره دارد.

واحدهای ورودی

ابزارهایی هستند که بوسیله آنها اطلاعات مربوط به تصاویر و گرافیک و داده ها وارد سیستم کامپیوتری می شوند. مانند ماوس، صفحه کلید، قلم نوری، اسکنر، دوربین دیجیتال و ...

فصل دوم:

ترسیم اشکال هندسی پایه

- **رسم نقطه**
- **رسم خط (الگوریتم DDA، الگوریتم برسنهام)**
- **رسم دایره (الگوریتم زاویه، الگوریتم نقطه میانی)**
- **رسم بیضی (الگوریتم زاویه، الگوریتم نقظه میانی)**

هر تصویر را میتوان به روشهای گوناگون توصیف کرد. برای مثال میتوان هر تصویر را به صورت مجموعه ای از اشیاء در نظر گرفت و سپس این اشیاء را بصورت اشکال اولیه مانند نقطه، خط، دایره و ... مدل کرد و در نهایت این اشکال اولیه را توسط مجموعه ای از پیکسل های نورانی نمایش داد.

رسم نقطه:

برای رسم نقطه تنها باید به طریقی مختصات نقطه مورد نظر را به رویه ای مناسب برای دستگاه خروجی ببریم. در سیستم های راستر مکان متناظر با نقطه، در فریم بافر مقداردهی میشود و سپس هنگام تابیدن الکترون مکان مورد نظر روشن میشود. معمولاً برای مقداردهی کردن فریم بافر(ترسیم نقطه) از تابع سطح پایین زیر استفاده میشود:

setpixel (x , y, color)

رسم خط با استفاده از الگوریتم معادله خط:

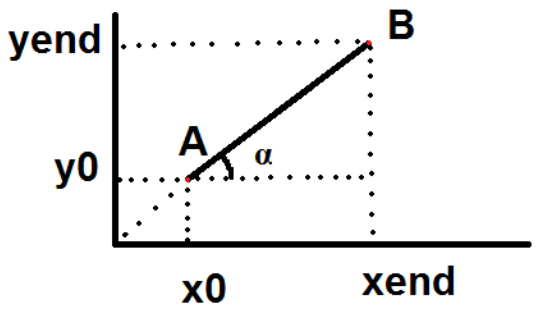
Y=mx+b

m: شیب خط نسبت به محور x ها

b: عرض از مبدا

x: جز معلوم هاست که با مقدار آن y بدست می آید.

با داشتن x و به دست آوردن y مختصات نقطه ای که قرار است رسم شود مشخص می شود.



$A(x_0, y_0)$

$B(x_{end}, y_{end})$

$m= \tan(\alpha)=$ضلع مقابل/ضلع مجاور

$Y_{end}-y_0/x_{end}-x_0$

$Y_0=mx_0+b$

$B=y_0-mx_0$

مثال:

نقطه A(8,15) و B(18,23)، جدول ردیابی ترسیم از نقطه A و B را با استفاده از الگوریتم معادله خط بنویسید.

DDA :الگوریتم

یکی از الگوریتم های ترسیم خط، الگوریتم DDA است که بر پایه معادله شیب خط عمل می کند. بسته به مقدار شیب خط یکی از فرمول های زیر را استفاده خواهیم کرد:

$$\Delta y = m\Delta x \Rightarrow \begin{cases} y_{k+1} = y_k + m\Delta x \overset{\Delta x=1}{\Rightarrow} y_{k+1} = y_k + m & (1) \\ x_{k+1} = x_k + \frac{1}{m}\Delta y \overset{\Delta y=1}{\Rightarrow} x_{k+1} = x_k + \frac{1}{m} & (2) \end{cases}$$

در صورتیکه شیب خط مقدار مثبت و کمتر از یک باشد، در این حالت باید در هر مرحله مقدار Δx را یک واحد افزایش میدهیم و سپس مقدار y را از فرمول یک محاسبه میکنیم

در صورتیکه شیب خط مقدار مثبت و بیشتر از یک باشد، در این حالت باید در هر مرحله مقدار Δy را یک واحد افزایش میدهیم و سپس مقدار x را از فرمول دو محاسبه میکنیم.

از آنجا که مقدار m میتواند هر عدد حقیقی باشد، بنابراین نیاز به گرد کردن عدد حاصل داریم. الگوریتم DDA الگوریتم سریعی برای پیدا کردن پیکسل ها است که مستقیماً از معادله خط استفاده میکند .اما به دلیل عملیات پیاپی در گردکردن میتواند منجر به خطاهای چشمگیری در خطوط طولانی شود. همچنین عملیات گرد کردن و کار با اعداد اعشاری بسیار وقت گیر است. در مقایسه الگوریتم معادله خط و الگوریتم DDA میتوان گفت که در روش DDA نسبت به الگوریتم معادله خط سرعت ترسیم خط بالاتر است اما تعداد محاسبات جمع بالاست که میتواند موجب خطا در خطوط طولانی شود.

نحوه ی اجرای الگوریتم DDA:

```
void dda_line(const int x_1,const int y_1,const int x_2,const int y_2)
{
int color=getcolor( );
int x1=x 1;
int y1=y 1;
int x2=x 2;
int y2=y2;
if(x_1>x_2)
{
x1=x_2;
```

```
y1=y2;
x2=x_1;
y2=y_1;
}
float dx=(x2-x1);
float dy=(y2-y1);
int steps=abs(dy);
if(abs(dx)>abs(dy))
steps=abs(dx);
float x_inc=(dx/(float)steps);
float y_inc=(dy/(float)steps);
float x=x1;
float y=y1;
putpixel(x,y,color);
for(int count=1;count<=steps;count++)
{
x+=x_inc;
y+=y_inc;
putpixel((int)(x+0.5),(int)(y+0.5),color);
}
}
```

الگوریتم Bresenham:

این الگوریتم یکی از الگوریتم های سریع و مناسب برای رسم خط است که توسط برسنهام ایجاد شد و تنها از محاسبات بر روی اعداد صحیح استفاده میکند ضمناً میتواند با کمی تغییر برای رسم دایره و سایر خطوط منحنی استفاده شود. این الگوریتم از دو الگوریتم قبلی سریع تر است.

نحوه اجرای الگوریتم برسنهام:

```
void bresenham_line(const int x_1,const int y_1,const int x_2,const int
y_2)
{
int color=getcolor( );
int x1=x_1;
```

```
int y1=y_1;
int x2=x2;
int y2=y_2;
if(x_1>x_2)
{
x1=x_2;
y1=y2;
x2=x1;
y2=y_1;
}
int dx=abs(x2-x1);
int dy=abs(y2-y1);
int two_dy=(2*dy);
int two_dy_dx=(2*(dy-dx));
int p=((2*dy)-dx);
int x=x1;
int y=y1;
putpixel(x,y,color);
while(x<x2)
{
x++;
if(p<0)
p+=two_dy;
else
{
y--;
p+=two_dy_dx;
}
putpixel(x,y,color);
}
}
```

الگوریتم رسم دایره:

یک دایره به صورت مجموعه نقاطی است که از یک نقطه مرکزی (X_C,Y_C) به فاصله مشخص r هستند. معادله دایره در دستگاه دکارتی به صورت زیر تعریف می شود:

$(x-x_c)^2+(y-y_c)^2=r^2$

ما میتوانیم از همین معادله برای پیدا کردن نقاط پیکسلها استفاده کنیم .بدین صورت که مؤلفه x را از بازه xc-r تا xc+r تغییر میدهیم و مقدار y را برای هر رابطه $y = y_c \pm \sqrt{r^2 - (x_c - x)2}$ بدست می آوریم.

اما این روش روش خوبی برای تولید دایره نیست، زیر اولاً در هر مرحله نیاز به انجام محاسبات فراوانی است و ثانیاً همانطور که در شکل زیر نشان داده شده است فاصله بین پیکسلها همه جا ثابت نیست.

یک روش برای حذف فضاهای خالی عوض کردن مولفه های x,y در جاییست که شیب خط بیشتر از یک یا کمتر از 1- باشد. که این روش نیاز به محاسبات فراوان دارد. روش دیگر برای حذف فضاهای بین پیکسلی استفاده از روش زاویه ای و رسم دایره در مختصات قطبی بر پایه r و θ می باشد. معمولا θ را در هر مرحله به مقدار $\frac{1}{r}$ افزایش می دهند تا دایره رسم شود.

$$\begin{cases} x = x_c + r\cos\theta \\ y = y_c + r\sin\theta \end{cases}$$

نکته مهم اینجاست که ما میتوانیم میزان محاسبات را با توجه به تقارن دایره کاهش دهیم. ما میتوانیم تنها یک هشتم دایره را رسم کنیم و سپس کل دایره را با تقارن از محورهای x,y,x=y بدست آوریم و تنها باید دایره را برای ۴۵درجه محاسبه کنیم.

ما تنها دایره را از نقطه x=0 تا x=y رسم کرده و سپس آنرا با تقارن دایره کامل می کنیم. اما همانطور که گفته شد معادلات دکارتی و قطبی روش خوبی برای رسم دایره نیستند. زیرا روش دکارتی نیاز به محاسبه ضرب و رادیکال، و روش قطبی نیاز به محاسبات مثلثاتی دارد. بدلیل وجود این محاسبات سرعت رسم کند می شود. روش بهتر الگوریتم midpoint است.

الگوریتم رسم دایره با زاویه:

```
void trigonometric_circle(const int h,const int k,const int r)
{
int color=getcolor( );
float x=0;
float y=r;
float angle=0;
float range=M_PI_4;
do
{
putpixel((int)(h+x+0.5),(int)(k+y+0.5),color);
putpixel((int)(h+y+0.5),(int)(k+x+0.5),color);
putpixel((int)(h+y+0.5),(int)(k-x+0.5),color);
putpixel((int)(h+x+0.5),(int)(k-y+0.5),color);
putpixel((int)(h-x+0.5),(int)(k-y+0.5),color);
putpixel((int)(h-y+0.5),(int)(k-x+0.5),color);
putpixel((int)(h-y+0.5),(int)(k+x+0.5),color);
putpixel((int)(h-x+0.5),(int)(k+y+0.5),color);
angle+=0.001;
x=(r*cos(angle));
y=(r*sin(angle));
}
while(angle<=range);
}
```

الگوریتم نقطه میانی:

این الگوریتم ساده تر و سریع تر است. دایره یک شکل متقارن است پس می توانیم از این خاصیت تقارن استفاده کنیم و عملیات رسم و عملیات بدست آوردن نقاط روی دایره را کاهش دهیم.

معادله دایره برای زمانیکه مرکز آن بر روی مرکز مختصات منطبق باشد $x^2+y^2=r^2$
اما زمانیکه مرکز آن x_c,y_c است:

$$(x-x_c)^2+(y-y_c)^2=r^2$$

$$F(x,y)=(x,x_c)^2+(y-y_c)-r^2$$

تابع دایره در واقع فرقی که با معادله دایره دارد این است که به ازای مقادیر مختلفx مقادیر y را می دهد و زوج x و y ممکن است بیرون از دایره، روی محیط دایره و یا داخل دایره قرار داشته باشد.

:F(x,y)

زوج x,y داخل دایره است 0>

زوج x,y روی محیط دایره است 0=

زوج x,y داخل دایره است 0>

پارامترهای تصمیم گیری در این الگوریتم باید به دست بیایند که بر اساس مطالب ریاضی است.

الگوریتم رسم دایره:

```
void midpoint_circle(const int h,const int k,const int r)
{
int color=getcolor( );
int x=0;
int y=r;
int p=(1-r);
do {
putpixel((h+x),(k+y),color);
putpixel((h+y),(k+x),color);
putpixel((h+y),(k-x),color);
putpixel((h+x),(k-y),color);
putpixel((h-x),(k-y),color);
putpixel((h-y),(k-x),color);
```

```
putpixel((h-y),(k+x),color);
putpixel((h-x),(k+y),color);
x++;
if(p<0)
p+=((2*x)+1);
else {
y--;
p+=((2*(x-y))+1);}
}
while(x<=y);
}
```

مزیت ها: محاسبات جمع و تفریق با عدد صحیح باعث افزایش سرعت کار می شود.

الگوریتم رسم بیضی:

در یک تعریف مقدماتی، بیضی یک دایره کشیده است. بنابراین اشکال بیضیوار را می توان با اصلاح الگوریتم رسم دایره رسم کرد.

خواص بیضی:

یک بیضی بصورت مجموعه ای از نقاط تعریف می شود که مجموع فاصله آن ها از دو نقطه ثابت (نقاط کانونی)، مقداری ثابت است.

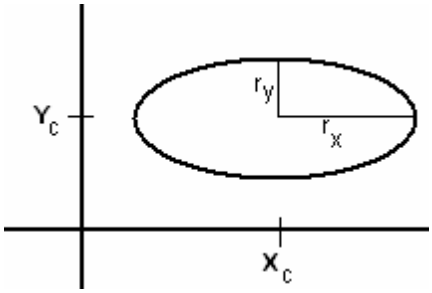
معادله بیضی به صورت زیر است:

$Ax^2+By^2=Cx^y=Dx=Ey=F=0$

که در آن ضرایب با توجه به مختصات نقاط کانونی و اندازه قطر اصلی و فرعی بیضی مشخص می شود. قطر اصلی خط مستقیمی است که از دو کانون بیضی می گذرد و قطر دوم عمود منصف قطر اول است .یک روش برای مشخص کردن بیضی داشتن مختصات نقاط کانونی و مختصات یک نقطه روی محیط بیضی است. با این سه مختصات می

$$\left(\frac{x-x_c}{r_x}\right)^2+\left(\frac{y-y_c}{r_y}\right)^2=1$$

توانیم مقدار ثابت معادله را محاسبه کرده و سپس ضرایب معادله را بدست آوریم. در صورتی که قطر اصلی و فرعی بیضی هم جهت محورهای مختصات باشند ، معادله بیضی بسیار ساده خواهد بود. شکل بیضی استاندارد بصورت شکل مقابل بوده و معادله آنرا میتوان بصورت زیر نوشت (با استفاده از معادله دکارتی):



در مختصات قطبی نیز این معادله به صورت زیر نوشته می شود:

$$\begin{cases} x = x_c + r_x \cos\theta \\ y = y_c + r_y \sin\theta \end{cases}$$

همچنین میتوان با در نظر گرفتن تقارن بیضی میزان محاسبات را کاهش داد. بر خلاف دایره که در هر یک هشتم متقارن است، یک بیضی استاندارد در هر یک چهارم متقارن خواهد بود، بنابراین باید بیضی ربع اول رسم کرده و سپس با استفاده از تقارن، بیضی را کامل کرد.

الگوریتم پیاده سازی رسم بیضی:

```
void trigonometric_ellipse(const int h,const int k,const int rx,const int
ry)
{
int color=getcolor( );
float x=0;
float y=ry;
float angle=0;
float range=rx;
do
{
putpixel((int)(h+x+0.5),(int)(k+y+0.5),color);
putpixel((int)(h+x+0.5),(int)(k-y+0.5),color);
putpixel((int)(h-x+0.5),(int)(k-y+0.5),color);
putpixel((int)(h-x+0.5),(int)(k+y+0.5),color);
angle+=0.05;
x=(rx*cos(angle));
y=(ry*sin(angle));
}
while(angle<=range);
}
```

رسم بیضی با استفاده از الگوریتم نقطه میانی:

این روش نسبت به روش های قبلی که گفته شد، ساده تر و سریع تر است. چرا که خاصیت تقارن در این الگوریتم وجود دارد و محاسبات همیشه مثبت هستند.

پیاده سازی الگوریتم به صورت زیر است:

```
void midpoint_ellipse(const int h,const int k,const int a,const int b)
{
float aa=(a*a);
float bb=(b*b);
float aa2=(aa*2);
floatbb2=(bb*2);
float x=0;
float y=b;
float fx=0;
float fy=(aa2*b);
float p=(int)(bb-(aa*b)+(0.25*aa)+0.5);
putpixel((h+x),(k+y),color);
putpixel((h+x),(k-y),color);
putpixel((h-x),(k-y),color);
putpixel((h-x),(k+y),color);
while(fx<fy)
{
x++;
fx+=bb2;
if(p<0)
p+=(fx+bb);
else {
y--;
fy-=aa2;
p+=(fx+bb-fy);
}
putpixel((h+x),(k+y),color);
putpixel((h+x),(k-y),color);
putpixel((h-x),(k-y),color);
putpixel((h-x),(k+y),color);
}
p=(int)((bb*(x+0.5)*(x+0.5))+(aa*(y-1)*(y-1))-(aa*bb)+0.5);
while(y>0) {
y--;
fy-=aa2;
if(p>=0)
p+=(aa-fy);
else {
```

```
x++;
fx+=bb2;
p+=(fx+aa-fy);
}
putpixel((h+x),(k+y),color);
putpixel((h+x),(k-y),color);
putpixel((h-x),(k-y),color);
putpixel((h-x),(k+y),color);
}
}
```

فصل سوم:

عملیات بر روی اشکال گرافیکی

- **پر کردن اشکال**
- **دوران**
- **تجانس و انتقال دو بعدی و سه بعدی (بزرگ کردن یا کوچک کردن)**

پر کردن اشکال:

دو راه برای پر کردن نواحی در سیستم های رستر وجود دارد:

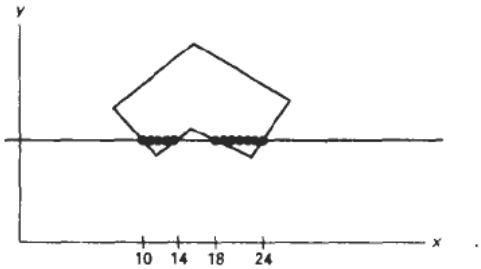
روش اول: بر پایه اسکن خطوط

روش دوم: بر پایه معادلات ریاضی

در این بخش فقط روش اول را مورد بررسی قرار می دهیم.

الگوریتم پر کردن چند ضلعی نامنتظم با استفاده از روش اسکن خطوط:

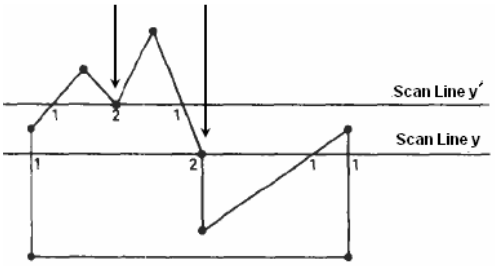
شکل زیر نحوه عملکرد این الگوریتم را برای رنکامیزی یک چند ضلعی نا منتظم نشان می دهد.



برای هر خط اسکن شد که از چندضلعی عبور کرده، الگوریتم، نقاط تقاطع خط اسکن شده با چند ضلعی را بدست می آورد. سپس این نقاط تقاطع از چپ به راست مرتب می شوند و در نهایت مکانهای بین هر جفت از این نقاط در میانگیر قاب با رنگ مشخص مقداردهی می شوند.

در شکل چهار نقطه تقاطع توسط الگوریتم تعیین می شود و پیکسل های داخلی از x=10 تا x=14 و از x=18 تا x=24 در میانگیر قاب با رنگ مشخص مقداردهی می شوند.

اما دقت کنید که اگر نقاط تقاطع خط اسکن با چندضلعی، رئوس چندضلعی باشد در این صورت نیاز به یک سری
دستکاری نقاط تقاطع داریم .برای مثال شکل زیر را در نظر بگیرید .در این صورت چگونه میتوان ناحیه داخل چندضلعی را تشخیص داد.

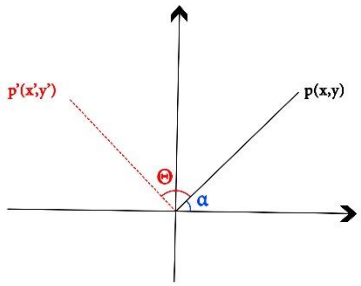


در اینجا خط اسکن y هیچ مشکلی ندارد اما خط اسکن y' دارای مشکل است. برای حل مشکل میتوان بدینگونه عمل کرد: برای هر نقطه تقاطع، قبل و بعد نقطه را نگاه میکنیم. اگر به طور یکنواخت نزولی یا صعودی بود که نقطه را به عنوان نقطه تقاطع در نظر میگیریم و در غیراین صورت نقطه را به صورت دو نقطه تقاطع در یک مکان در نظر میگیریم. برای پیاده سازی این روش نیاز به محاسبات فراوان میباشد. ولی مزیت آن نسبت به الگوریتم معادلات ریاضی توانایی در پر کردن اشکال چندضلعی نامنتظم است.

دوران:

دوران یک **نقطه** نسبت به مبدا مختصات (۰،۰)

میخواهیم نقطه p را تحت زاویه Θ دوران دهیم:

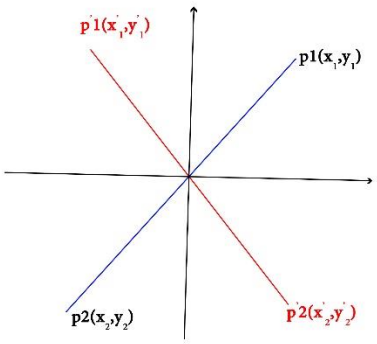


$$[x\ \ y]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}=[x'\ y']$$

مثال: نقطه (2,3) تحت زاویه 45 درجه دوران دهید.

دوران یک **خط** نسبت به مبدا مختصات:

میخواهیم خط p_1,p_2 را تحت زاویه θ دوران دهیم:



$$[x_2 \;\; y_2]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}=[x'_1 \;\; y'_1]$$

$$[x_1 \;\; y_1]\begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}=[x'_2 \;\; y'_2]$$

مثال: خطی داریم که نقطه پایانی آن (۲و۱۰) و نقطه اول آن (۴و۵-) است. آن را تحت زاویه ۹۰ درجه دوران دهید.

مثال: دوران $y=2x+3$ را نسبت به مبدا مختصات و تحت زاویه ۹۰ درجه بدست آمورید.

تجانس:

تجانس نسبت به مبدا مختصات:

ماتریس تجانس در حالت دو بعدی به صورت زیر است:

$$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

نکته: اگر s_x بزرگتر از یک یا s_y بزرگتر از یک باشد آنگاه انبساط طولی داریم. یعنی پهنای شکل بزرگ است.

اگر s_x کوچکتر از یک یا s_y کوچکتر از یک باشد آنگاه انقباض داریم.

مثال: تجانس یافته نقطه p(2,3) را بدست آورید. با ضریب تجانس s_x:1/2 و y_x:2

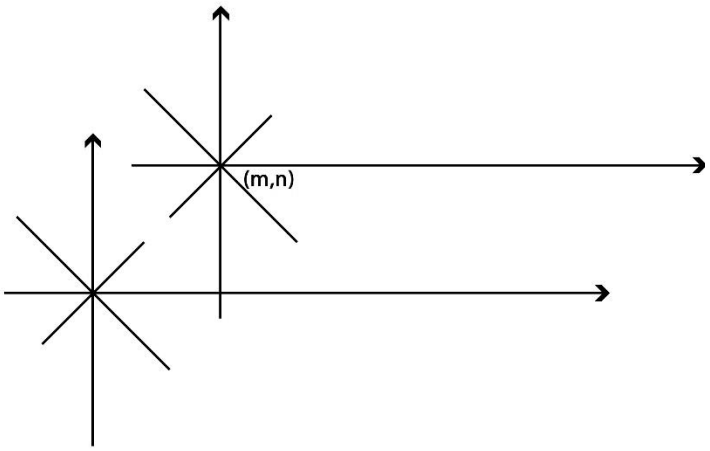
دوران خط نسبت به مرکز m, n

در این صورت باید سه مرحله ی زیر را انجام دهیم:

۱- انتقال خط از m,n به 0,0 (خط۲)

۲- دوران خط انتقال یافته نسبت به 0,0 با زاویه θ (خط ۳)

۳- انتقال خط دوران یافته از مرکز 0,0 به مرکز m,n (خط ۴)



برای بدست آوردن نقطه وسط خط:

$$\frac{x_1 + x_2}{2} = m$$

$$\frac{y_1 + y_2}{2} = n$$

فرمول های خط ۲: (انتقال)

$$p_1' \begin{cases} x_1' = x_1 - m \\ y_1' = y_1 - n \end{cases}$$

$$p_2' \begin{cases} x_2' = x_2 - m \\ y_2' = y_2 - n \end{cases}$$

فرمول های خط۳: (دوران)

$$p_1'' \{[x_1' \;\; y_1'] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = [x_1'' \;\; y_1'']$$

$$p_2'' \{[x_2' \;\; y_2'] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = [x_2'' \;\; y_2'']$$

فرمول های خط ۴: (انتقال)

$$p_1''' \begin{cases} x_1''' = x_1'' + m \\ y_1''' = y_1'' + n \end{cases}$$

$$p_2''' \begin{cases} x_2''' = x_2'' + m \\ y_2''' = y_2'' + n \end{cases}$$

مثال: خط زیر را دوران دهید:

$p_1(50,30)$ $p_2(70,100)$

فصل چهارم:

آشنایی با OpenGl

OpenGL مخفف عبارت Open Graphic Library است و همان طور که از نامش پیداست یک کتابخانه جهت دستیابی ساده به سخت افزار گرافیکی شماست. برای کاربران عادی نام OpenGL با نام بازی های سه بعدی گره خورده است. قبل از نوشتن OpenGL برنامه نویسان گرافیکی مجبور بودند که مستقیما برای سخت افزارهای گرافیکی کار کنند و برای سخت افزارها برنامه های اختصاصی بنویسند. اما با گسترش روز افزون تکنولوژی و سخت افزارهای مختلف این رویه ی برنامه نویسی کاربرد خود را از دست داد.

OpenGLدر اصل برای دو هدف اصلی توسعه داده شده و می شود:

- پنهان کردن پیچیدگی کار با سخت افزارهای مختلف گرافیکی با یک رابط سطح پایین
- ساده کردن کارهای گرافیکی خصوصا سه بعدی با ایجاد یک رابط استاندارد واحد

کتابخانه های مرتبط با OpenGL

همان طور که ذکر شد، OpenGL مستقل از سکو است. بنابراین کارهایی که اختصاص به سکوی میزبان دارد را به طور مستقیم پشتیبانی نمی کند. از جمله ی این کارها می توان به ایجاد و کنترل پنجره ها ، کنترل ورودی و خروجی، ایجاد اشیای آماده (مانند کره ، مخروط و ...) ، کنترل صداها و سیگنال های دیجیتال و ارتباط با شبکه را نام برد. دلیل پشتیبانی نکردن این موارد، وابسته بودن آن ها به سیستم عامل و سکوی میزبان است که تفاوت قابل ملاحظه ای با هم دارند. بنابراین OpenGLاین کارها را به کتابخانه های مرتبط واگذار کرده است. این کتابخانه ها قابلیت انجام تمامی موارد ذکر شده را دارند.

این کتابخانه ها بر دو نوع اند:

کتابخانه های مستقل از سکو

کتابخانه های وابسته به سکو

کتابخانه های مستقل از سکو همانند OpenGL قابلیت انتقال به هر سکویی که OpenGL به آن پورت شده باشد را دارند و دیگر نیازی به تغییر کدهایتان در آن ها ندارید. اما کتابخانه های وابسته به سکو در سیستم عامل های خاصی اجرا می شوند و در دیگر سیستم عامل ها کارایی ندارند. (نیاز به تغییر عمده ی کد وجود دارد) از جمله ی کتابخانه های مستقل از سکو می توان دو کتابخانه GLUTو SDL را نام برد.

از جمله ی کتابخانه های وابسته به سکو می توان به WGL برای ویندوز، CGL برای Mac OS X و GLX برای لینوکس اشاره کرد. البته لازم به ذکر است این کتابخانه ها تنها استفاده از سیستم Windowingرا فراهم می کنند و دیگر قابلیت ها مثل کنترل ورودی و خروجی یا صدا را پشتیبانی نمی کنند و برای اضافه کردن این قابلیت ها بایستی از زبان برنامه نویسی مورد استفاده و یا ویژگی های سیستم عامل خود استفاده کنید. پس ملاحظه می فرمایید که استفاده از کتابخانه های مستقل از سکو ضمن اینکه وابستگی به میزبان ندارند، همگی موارد نیاز برای نوشتن یک بازی رایانه ای را برایتان فراهم می آورند.

کتابخانه ی GLUT

در واقع جعبه ابزار OpenGL است. این کتابخانه تکمیل کننده ی کتابخانه ی GLU (OpenGL Utility Library) است که توابعی را ارائه می دهد که یک سطح بالاتر از سطح توابع اصلی OpenGLهستند. کتابخانه ی GLU معمولا به همراه OpenGL توزیع می شود چرا که کارهای

اصلی که با OpenGL می توان انجام داد از قبیل: تبدیلت مختصات، ایجاد texture ها، ایجاد چهارضلعی ها، تولید خطاهای OpenGL و غیره را شامل می شود.

و در واقع برنامه نویسی OpenGL بدونGLU کار بسیار سخت و طاقت فرسایی خواهد بود. حال کاری که کتابخانه ی GLUT کرده است (OpenGL Utility Toolkit) یک سطح بالتر از GLU را ارائه داده است. یعنی کار با اجزای سیستم عامل میزبان. کارهایی از قبیل کنترل پنجره ها، کنترل ورودی و ... را می توان با این کتابخانه انجام داد. سه کتابخانه یOpenGL ، GLUو GLUT یک پکیج کامل و هماهنگ را جهت ایجاد برنامه های گرافیکی بسیار قدرتمند فراهم می سازند.

مراحل ایجاد یک برنامه Open GL

برای برنامه نویسی Open GL تنها کافی است سه کتابخانه , glut32.h , glut32.dll glut32.lib را دانلود کنید و در مسیر های زیر کپی کنید:

glut32.dll <- C:\Windows\system32

glut32.lib <- C:\programfiles\Microsoft Visual Studio 11.0\Vcc\lib

glut32.h <- C:\programfiles\Microsot Visual Studoi 11.0\Vc\include\gl

سپس زمان آن است که برنامه Visual Studio را باز کرده و پروژه C++ ایجاد کنید حال باید کتابخانه ها را به برنامه خود بشناسانیم بدین صورت

Project\test properties\configuration properties\liker\input\Additional Dependencies

(توجه داشته باشید که test در قسمت test properties نام فرضی پروژه شماست)

در مسیر فوق نام کتابخانه های خود رامیینویسیم و OK

حال میتوان کدهای Open GL خود را نوشته و اجرا کرد برای مثال برنامه زیر یک خانه را ترسیم میکند:

```
#include <windows.h>
#include <gl/glut.h>
void init (void)
{
glClearColor (1.0, 0.7, 0.9, 0.0);
glMatrixMode (GL_PROJECTION);
gluOrtho2D (0.0, 600.0, 0.0, 500.0);
}
void Draw_House(void)
{
glClear (GL_COLOR_BUFFER_BIT);
glColor3f (0.0, 0.0, 1.0);
glEnable(GL_LINE_SMOOTH);
glBegin (GL_LINES);
glVertex2i (300, 400);
glVertex2i (200, 300);
glVertex2i (300, 400);
glVertex2i (400, 300);
glVertex2i (200, 300);
glVertex2i (400, 300);
glVertex2i (200, 300);
```

```
glVertex2i (200, 100);
glVertex2i (400, 300);
glVertex2i (400, 100);
glVertex2i (200, 100);
glVertex2i (400, 100);
glEnd ();
glFlush ();
}
void main (int argc, char** argv)
{
glutInit (&argc, argv);
glutInitWindowPosition (50,100);
glutInitWindowSize (600,500);
glutCreateWindow ("Draw_House");
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB );
init();
glutDisplayFunc (Draw_House);
glutMainLoop();
}
```