دانشگاه آزاد اسلامی واحد شوشتر

گرافیک کامپیوتری

شکیبا فخر

فصل اول

مقدمه

۱.۱. مقدمه

تکامل علوم، فن‌آوری، هنر، تجارت و صنعت همواره مرهون امکان برقراری ارتباط و اطلاعات بوده است. با گذشت زمان، انتقال اطلاعات از شکل اولیه یعنی سنگ نوشته تا شکل کنونی یعنی بیت های ذخیره شده در یک تراشه در آمده است. در این بین رسانه‌هایی که از جنبه بصری برخوردار بودند (مانند: کتاب- تلویزیون و ...) به دلیل برخورداری از جنبه آموزشی خوب از جایگاه ویژه‌ای برخوردار بوده‌اند. در اواخر قرن بیستم و با همگانی شدن کامپیوترهای شخصی این رسانه بیش از پیش در برقراری ارتباطات و رشد فن‌آوری سهیم گشت.

نمایش تصویر از طریق صفحه نمایش کامپیوترهای شخصی از همان ابتدا مورد توجه قرار گرفت به گونه‌ای که امروزه با گذشت حدود ۴۰ سال از ساخت اولین صفحه نمایش‌های لامپی، استفاده گرافیک کامپیوتری در بسیاری از زمینه های زندگی ما رخنه کرده است. دامنه به کارگیری گرافیک کامپیوتری از تولید چارت‌ها و گراف‌ها گرفته تا تولید تصاویر تلویزیونی وانیمیشن و طراحی، همه و همه به قدری وسیع گشته است که زندگی کردن بدون به کارگیری آنان ما را به زحمت می اندازد.

می توان گرافیک کامپیوتری را به صورت ساده بدین ترتیب تعریف نمود: «گرافیک کامپیوتری عبارت است از به کارگیری کامپیوتر در همه جنبه هایی که نیازمند ساخت تصویر هستند.»

۲.۱. نگاهی به جایگاه گرافیک کامپیوتری

بر اساس تعریف فوق، گرافیک کامپیوتری می تواند در تولید و نمایش هر آنچه به صورت طرح قابل نمایش است، مورد استفاده قرار گیرد. کاربرد گرافیک کامپیوتری را می توان به چهار زمینه زیر دسته‌بندی نمود.

۱. نمایش اطلاعات
۲. طراحی
۳. شبیه سازی
۴. واسط کاربر

۱.۲.۱. نمایش اطلاعات

گرافیک همیشه با نمایش اطلاعات توأم بوده است. تاریخچه نگارش را می توان به ۲۰۰۰ سال پیش نسبت داد که مربوط به کنده کاریهای بابلیان بوده است. شیوه‌های مکانیکی برای ایجاد ترسیمات از رنسانس به بعد رایج گردید. بتدریج مهندسین به منظور ترسیم طرح های خود و ثبت بر روی کاغذ به ابزار متوسل گردیدند. این روند ادامه یافت تا امروزه که می بینیم کلیه طرح ها و ایده هایی که به ذهن مهندسین خطور می کند طی چند دقیقه تبدیل به نقشه ای رنگی بر روی کاغذ و یا صفحه نمایش می گردد. در شاخه هایی نظیر معماری و طراحی جامدات نیز طراحی به کمک کامپیوتر (CAD) از جایگاه ویژه‌ای برخوردار است.

حتی بسیاری از پزشکان ترجیح می دهند کار تجزیه و تحلیل تصاویر را به کامپیوتر محول کنند، به گونه‌ای که در یکی از رشته های جوان مانند مهندسی پزشکی، به ثبت و تحلیل تصاویر گرفته شده توسط کامپیوتر بسیار پرداخته می شود.

۲.۲.۱. طراحی

حرفه‌هایی نظیر مهندسی و معماری با طراحی اجین می باشند. یکی از مهمترین دلایل پرداختن به طرح‌ها آن است که معایب و مشکلات ایده های افراد بر روی کاغذ و به شیوه دیداری نمایان تر می گردد. به گونه‌ای که با تحلیل و ارزیابی آنها قبل از پیاده سازی طرح ها، می توان مانع از اجرای طرح های معیوب و یا احتمالاً به هدر رفتن وقت و بودجه گردید (تصور اجرای یک پروژه عمرانی بزرگ مانند سدسازی بدون وجود طرح ها مضحک به نظر می رسد). از همین رو یافتن نقاط ضعف و اصلاح آنها موجب می

گردد این طرح‌ها دائماً در حال تغییر باشند. بدین ترتیب جایگاه رسانه‌ای که بتواند این تغییرات را با کمترین هزینه و در اسرع وقت در طرح ها اعمال نماید به خوبی روشن می گردد.

طراحی در کلیه رشته های مهندسی از جایگاه ویژه ای برخوردار است. امروزه مدارات مجتمع VLSI قبل از مرحله ساخت مورد بررسی قرار می گیرند. رسانه های ورودی گرافیک روز به روز محبوب تر گشته و تکمیل می گردند. تصور وجود یک کامپیوتر بدون موس می تواند وحشتناک باشد.

۳.۲.۱. شبیه سازی

یکی از زمینه‌های بسیار فعال در کاربرد گرافیک کامپیوتری را می توان شبیه سازی دانست. طراحی‌های انجام شده در مرحله قبل را می توان شبیه‌سازی نموده و خروجی بدست آمده را با نتایج قبلی مقایسه کرد و بدین ترتیب بهترین طراحی را بدست آورد. بازی های کامپیوتری را می توان مظهر توانایی انسان و گرافیک کامپیوتری دانست. بسیاری از افراد در حین اجرای این بازی‌ها از هنر نرم افزار و سخت افزار برای همگام سازی حرکات و جلوه های نمایشی آگاهی ندارند. شبیه سازی و مدل سازی کامپیوتری موجب گردیده است که آموزش بسیاری از فنون تجربی دچار تحول و دگرگونی گردد، مانند آموزش خلبانی توسط شبیه‌سازی محیط هواپیما. قابل رؤیت شدن طرح ها قبل از ساخت واقعی آنها شاید از ارزنده ترین خدماتی باشد که گرافیک کامپیوتری به صنایع خودرو سازی و هواپیماسازی ارائه نموده است.

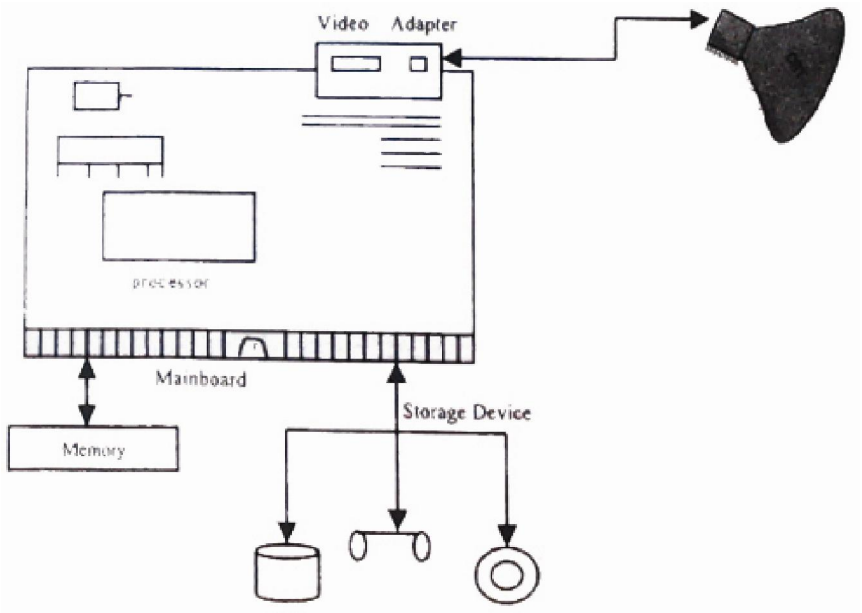
۴.۲.۱. واسط کاربر

به کارگیری گرافیک کامپیوتری موجب تحولات اساسی در برقراری ارتباط بین برنامه ها و کاربران گشته است. بسیاری از کاربران ترجیح می دهند به جای کار در محیط متنی صفحه نمایش، در محیط هایی کار کنند که با دستگاه های نقطه یابی (موس) امکان دسترسی به هر گوشه صفحه نمایش را فراهم می آورد. همین گرایش موجب گردیده است که محیط برقراری ارتباط اغلب سیستم های عامل با کاربران به صورت گرافیکی در آید. در یک چنین محیط هایی هر آیکون می تواند مبین انجام عملیات خاصی باشد. یک چنین محیطی به لحاظ سهولت به کارگیری و آموزش بر محیط های دستوری ترجیح داده می شود. به عنوان مثال برای بررسی E-Mail خود کافی است بر روی آیکون خاصی دابل کلیک کنید.

کار در محیط گرافیکی در اینترنت دیگر یک مطلب عادی به شمار می رود. برقراری ارتباط گرافیکی پیش از پیش همکاری تیم های طراحی را سهولت بخشیده است به گونه ای که افراد در نقاط مختلف جهان می توانند پیرامون طرح ها، نظرات خود را بیان نمایند.

ظهور Multimedia (چندرسانه‌ای) را بیش از هر چیز می توان مرهون اشاعه گرافیک کامپیوتری دانست پخش فیلم و اسلاید و انیمیشن موجب بروز تحول اساسی در ارتباطات و انفورماتیک گردیده است.

قبل از پایان دادن به این مبحث خاطر نشان می کنیم که گرافیک کامپیوتری در بسیاری از جنبه های دیگر مانند: هنر- علوم نظامی و فضایی، تفریح و سرگرمی، واقعیت مجازی، پردازش تصویر و ... نیز کاربرد دارد.



شکل ۱-۱- اجزاء اصلی در یک کامپیوتر معمولی

۳.۱. اجزاء پایه

اکنون نگاهی به اجزاء مورد استفاده در سیستم های گرافیکی رایج خواهیم داشت. دیدگاه ما همان دیدگاه سطح بالا و کلی است که کاربران معمول از سیستم دارند و از پرداختن به جزئیات سخت افزاری چشم پوشی کرده‌ایم. نمودار بلوکی این اجزاء به صورت شکل ۱-۱ می باشد. اجزاء کلیدی را می توان به چهار دسته زیر تقسیم بندی نمود.

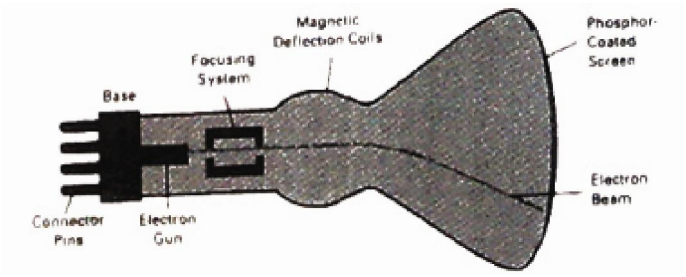
۱. پردازنده
۲. حافظه
۳. رسانه های خروجی
۴. رسانه های ورودی

برد اصلی سیستم که پردازنده بر روی آن نصب می گردد(گاهی به آن برد مادر نیز گفته می شود) امکان رد و بدل کردن اطلاعات را فراهم می آورد. برای این منظور گذرگاه های داده (Data Bus) به صورت ۳۲،۱۶ و ۶۴ بیتی تعبیه شده‌اند. برای دریافت و ارسال اطلاعات به رسانه های ورودی و خروجی معمولاً بوردهای جداگانه ای بر روی شکافهای توسعه ی سیستم نصب می گردند. این بوردها که اغلب به صورت کارت ساخته می شوند در واقع دروازه ارسال و دریافت داده ها محسوب می گردند (مانند: کارت گرافیکی- کارت شبکه- کارت صدا- کارت TV- کارت فکس مودم و ...) که استفاده کننده بسته به نوع نیاز خود آنها را خریداری و بر روی برد اصلی نصب می کند. به منظور برقراری ارتباط صفحه نمایش با برد اصلی نیز کارتی به نام آداپتور ویدئویی (کارت گرافیک) بر روی شکاف های توسعه نصب می گردد.

بدین ترتیب سیگنال مورد نیاز جهت تشکیل تصویر از طریق این کارت و کابل داده واسط به مانیتور ارسال می شود. مانیتورها علاوه بر ورودی کابل داده، ورودی منبع تغذیه نیز دارند که انرژی الکتریکی مورد نیاز جهت ساخت تصویر را تأمین می کند. ساختمان کارت های گرافیکی و صفحه نمایش‌ها به صورت جداگانه در ادامه همین فصل مورد بررسی قرار گرفته است.

۴.۱. صفحه نمایش ویدئویی

ساده ترین نوع صفحه نمایش، مانیتور ویدئویی است. عملکرد اغلب مانیتورهای ویدئویی مبتنی بر طراحی لامپ اشعه کاتدی Cathode- Ray- Tube (CRT) استاندارد است. اما بتدریج عملکرد این رسانه با پیشرفت فن‌آوری، بهبود داده شده است. شکل ۱-۲ به صورت ابتدایی،عملکرد CRT را نشان می دهد.

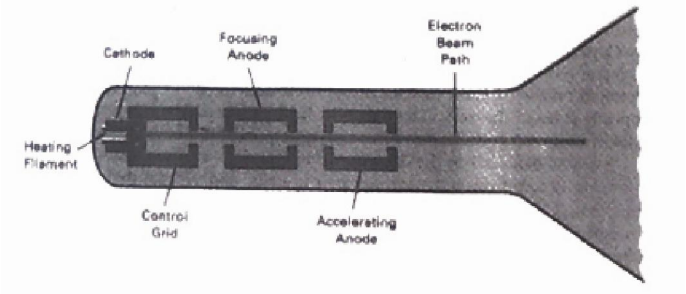


شکل ۱-۲- طرحواره ای ساده از عملکرد انحراف مغناطیسی CRT

به صورت ساده عملکرد CRT بدین صورت است که پرتو باریک الکترونی، توسط یک تفنگ الکترونی، با گذر از بخش متمرکز کننده و سیستم منحرف کننده به سوی نقطه مشخصی از صفحه‌ای

شیشه‌ای که با غشایی از فسفر پوشانده شده است، تابیده می شود. بدین ترتیب نقطه‌ی بسیار کوچکی از غشاء فسفری که در معرض تابش قرار گرفته است، روشن می شود. ولی این نقطه روشن به سرعت محو می گردد. از همین رو با استفاده از روشی سعی می شود که تصویر پایدار باقی بماند. یک راه پایدار ساختن تصویر آن است که مجدداً با تابش سریع بر روی همان نقطه، سعی کنیم تصویر را با ترسیم مجدد، پایدار نگه داریم. این نوع از صفحه نمایش،CRT با بازسازی[1] نامیده می شود.

یکی از مهمترین اجزاء یک CRT، تفنگ الکترونی است. این تفنگ یک عنصر گرمایشی(المنت) است که مستقیما باعث تابش پرتوی کاتدی می گردد. دلیل گرم شدن پشت مانیتورهای معمولی نیز همین قضیه است. پرتوهای پراکنده توسط یک شبکه کنترل شده و با گذشتن از یک آند شتاب دهنده، با گرفتن بار منفی با ولتاژ بالا به سمت سطح مقابل تابش می شوند. (شکل ۱-۳)



شکل ۱-۳- عملکرد یک تفنگ الکترونی با یک شتاب دهنده آندی

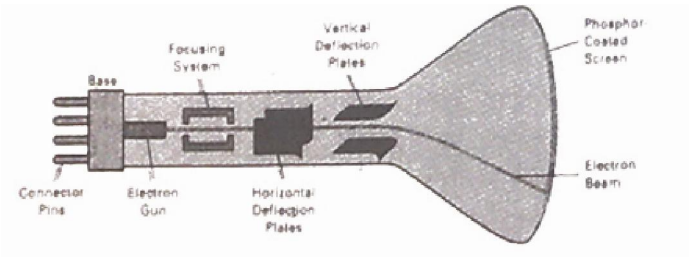
گاهی تفنگ الکترونی به همراه آند شتاب دهنده و سیستم متمرکز کننده در یک واحد مجتمع می گردند. شدت تابش الکترون ها با تنظیم سطح ولتاژ شبکه کنترلی تنظیم می شود. ولتاژ پایین در شبکه کنترلی باعث کاهش تعداد الکترون هایی که عبور می کنند می شود. شدت تابش نور بستگی به تعداد الکترون هایی دارد که به سطح غشاء فسفری شلیک می شود. بدین ترتیب ما در واقع با کنترل میزان تشعشات، میزان روشنایی را کنترل می کنیم.

وظیفه سیستم متمرکز کننده این است که پرتوهای شلیک شده را به صورت پرتو باریکی به یک نقطه کوچک بر روی غشاء فسفری هدایت کند نحوه متمرکز سازی می تواند به صورت الکتریکی یا مغناطیسی باشد. متمرکز سازی الکترواستاتیکی عموماً در تلویزیون ها و مانیتورهای گرافیکی کامپیوتر مورد استفاده واقع می شود. همانگونه که در شکل ۱-۳ دیده می شود عملکرد این سیلندر فلزی را می توان به لنزهای الکترواستاتیکی تشبیه کرد که پرتوهای نور را در یک نقطه متمرکز می سازند. عملکردی شبیه متمرکز کننده

[1]. *Refresh CRT*

الکترواستاتیکی را می توان با لنزهای مغناطیسی نیز انجام داد. این متمرکز کننده ها دارای سیم پیچی هستند که با گذشت جریان از آنها نوعی میدان مغناطیسی در اطراف پرتو به وجود آمده و آن را متمرکز می کند. لنزهای مغناطیسی کم حجم تر بوده و در رسانه های خاصی مورد استفاده قرار می گیرد.

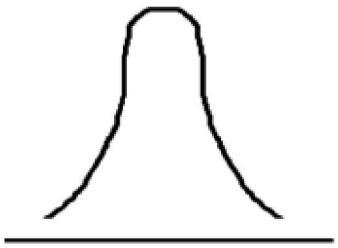
به منظور هدایت و تمرکز پرتوها به کلیه نقاط صفحه و با دقت بالا، نیازمند یک واحد سخت افزاری دیگر هستیم. تفنگ الکترونی در مرکز لامپ خلاء قرار گرفته است. از همین رو میزان تابش پرتوها در نقاط میانی صفحه به دلیل نزدیکی فاصله بیشتر از نقاط حاشیه ای خواهد بود. از همین رو تفنگ الکترونی تنها به مرکز لامپ شلیک می کند. به همین جهت دو بخش منحرف کننده با صفحاتی در راستای افقی و عمودی باعث هدایت پرتو باریک به کلیه نقاط سطح داخلی لامپ می گردد. به منظور جبران اثر دوری فاصله نقاط حاشیه‌ای لامپ، سعی کرده اند که با دادن کمی انحنا به جداره لامپ، اثر این اختلاف فاصله به حداقل برسد. (شکل ۱-۴)



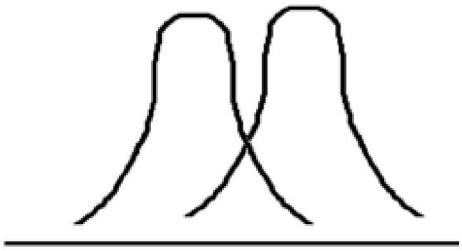
شکل ۱-۴- انحراف الکترواستاتیکی پرتوهای الکترونی در یک CRT

حداکثر تعداد نقاطی که می تواند بدون هم پوشانی در یک CRT به نمایش در آید قدرت تفکیک یا وضوح[1] نامیده می شود. تعریف آشنای دیگر، تعداد نقاط روشن قابل نمایش در هر سانتی متر در راستای افقی و عمودی است و یا گاهی حداکثر تعداد نقاط قابل نمایش در هر راستا را گویند. همجواری بیش از حد نقاط به یکدیگر موجب همپوشانی و عدم تشخیص نقاط می گردد. شدت نقاط دارای یک توزیع گوسی است (شکل ۱-۵). وقتی نقاط تشخیص داده می شوند که فواصل آنها بیش از قطر هر دایره باشد. قطری که شدت آن حدود ۶۰ درصد مرکز نقطه است. این همپوشانی در شکل ۱-۶ نشان داده شده است. قطر نقاط به قطر باریکه الکترونی و سطح انرژی آن بستگی دارد.

[1]. *Resolution*



شکل ۱-۵- توزیع شدت نمایش نقاط فسفری در یک صفحه نمایش CRT



شکل ۱-۶- دو نقطه روشن شده همجوار وقتی قابل تشخیص هستند که فاصله آنها از یکدیگر از مرکز، بیشتر از ۶۰ درصد حداکثر قطر تابش باشد.

بدین ترتیب قدرت تفکیک یک CRT به نوع فسفر، شدت تابش و سیستم منحرف کننده و متمرکز کننده بستگی دارد. وضوح یک CRT معمولی، بسته به نوع و اندازه مانیتور، متفاوت است، اما بطور کلی وضوح مانیتورها 600 * 800، 768 * 1024، 864 * 1152، 768 * 1280 و 1024 * 1280 است که بترتیب برای مانیتورهای ۱۵ اینچ، ۱۷ اینچ و بالاتر می باشد. اندازه CRT معمولی حدود ۱۲ تا ۲۷ اینچ است. ویژگی دیگر قابل بحث در مانیتورهای ویدئویی نرخ نسبت نمایش[1] است. این عدد عبارتست از نرخ تعداد نقاط در راستای عمودی بر تعداد نقاط در راستای افقی. به عنوان مثال نرخ نمایش ۳/۴ بدین مفهوم که طول سه نقطه در راستای عمودی معادل طول ۴ نقطه در راستای افقی است.

۱.۴.۱. صفحه نمایش با پویش رستر[2]

اغلب انواع مانیتورهای موجود CRT از نوع پویشی مبتنی بر فن آوری تلویزیون هستند. در صفحه نمایش با پویش رستر پرتو الکترونی، طول صفحه نمایش را از بالا به پایین جاروب می کند. وقتی پرتو الکترونی طول هر سطر را جاروب می کند، برای ایجاد تصویر مورد نظر، در نقاطی از صفحه خاموش، و در نقاط دیگر روشن است. تصویری که تعریف شده و در حافظه مستقر شده است، بافر بازسازی[3] یا بافر فریم[4] نامیده می شود. این بخش از حافظه مجموعه‌ای از مقادیر شدت تابش را نگهداری می کند. لازم به یادآوری است که مانیتور معمولی یک لامپ به همراه تجهیزات جانبی است و برخلاف کارت گرافیکی

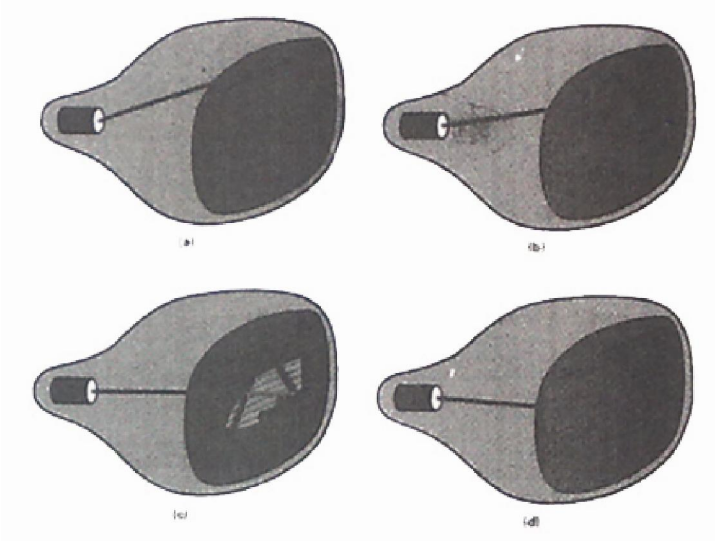
[1] . *Aspect ratio*
[2] . *Raster-Sean Display*
[3] . *Refresh buffer*
[4] . *Frame buffer*

حافظه ندارد. با تکمیل هر سطر،اطلاعات آن سطر تکمیل شده و مرئی می گردد. تلویزیون های خانگی و چاپگرها نمونه هایی از سیستم هایی هستند که بر اساس روش پویش رستر عمل می کنند.

به هر نقطه در صفحه نمایش، پیکسل گویند (Pixel مخفف Picture Element می باشد).

در سیستم های سیاه و سفید، هر نقطه نمایشی می تواند خاموش یا روشن باشد و تنها یک بیت به ازاء هر پیکسل برای کنترل شـدت و میزان تابش به نقطه کفایت می کند. در صورتی که بخواهیم دامنه‌ی متغیری از رنگ‌ها را در هر پیکسل نمایش دهیم (سطوح خاکستری و یا رنگ ها)، تعداد بیت‌های مورد نیاز، با ازاء هر پیکسل افزایش می یابد (مثلاً ۲۴ بیت بر پیکسل). بدین ترتیب ظرفیت بافر مورد نیاز برای نگهداری هر فریم بسته به وضوح صفحه ممکن است چندین مگابایت گردد. به عنوان مثال برای نگهداری یک فریم در حافظه بافر و با وضوح ۱۰۲۴ در ۱۰۲۴ و با ۲۴ بیت بر پیکسل، این حافظه حدود سه مگابایت خواهد شد (1024*1024*24) / (8*1024*1024) = 3.

در سیستم سیاه و سفید، که در هر پیکسل، فقط یک بیت وجود دارد، بافر فریم را معمولاً نگاشت بیتی (Bitmap)، و در سیستمی که به ازاء هر پیکسل چندین بیت مورد استفاده قرار می گیرد، بافر فریم را نگاشت پیکسلی (Pixmap) می نامند. (شکل ۱-۷)



شکل ۱-۷- در صفحه نمایش پویشی هر شیئی به صورت مجموعه ای از نقاط گسسته در راستای هر خط پویشی ترسیم می گردد

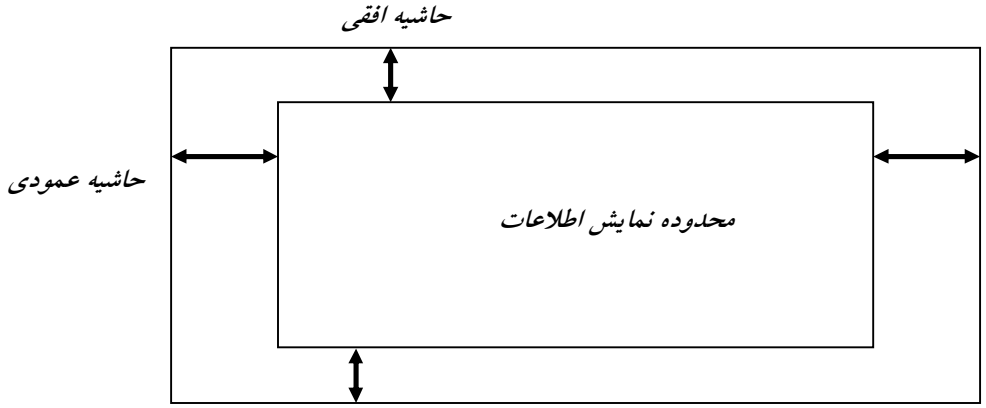
مانیتورهای تک رنگ برای ایجاد تصویر نیازمند تنها یک پرتو الکترونی هستند در حالی که در مانیتورهای رنگی از سه پرتو همزمان استفاده می شود. یعنی هر پیکسل صفحه، مشتمل بر سه جزء فسفری

با سه رنگ اصلی: قرمز- سبز و آبی است. هر رنگ یک پرتو الکترونی منطبق شونده دارد. با تنظیم شدت تابش این سه نوع پرتو می توان طیف بسیار متنوعی از رنگ ها را در صفحه بوجود آورد.

بازسازی صفحه نمایش ۶۰ الی ۸۰ بار در هر ثانیه انجام می شود. البته برخی از سیستم ها به گونه ای طراحی شده اند که نرخ بازسازی بالاتری دارند. نرخ بازسازی بر حسب واحد دور در ثانیه یا هرتز (Hertz) بیان می شود. به عنوان مثال نرخ بازسازی 60Hz یعنی 60 فریم در هر ثانیه. در پایان هر خط پویش، چشمه نور به سمت چپ خط پویش بعد باز می گردد.

- **ناحیه خارج از نمایش** (Over Scan)

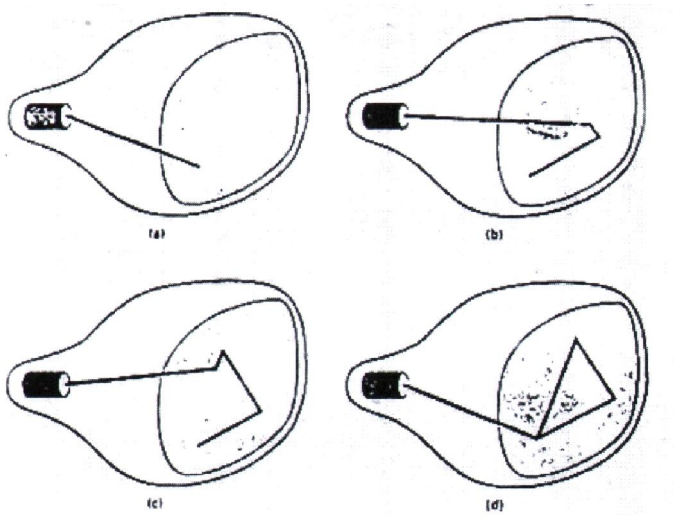
هر بار که پرتو الکترونی نیازمند بازگشت به ابتدای خط است، مدت زمان کوتاهی نیاز خواهد داشت تا اطلاعات جدید سطر بعد را از حافظه RAM ویدئو برداشت نماید. از همین رو یک وقفه بسیار کوتاه بروز می‌نماید. از طرفی پرتو الکترونی نمی تواند متوقف گردد به همین جهت حاشیه ای در سمت راست و چپ به وجود می آید که از دسترس خارج است و نمی توان اطلاعاتی را در آن به تصویر کشید. این حاشیه «ناحیه خارج از نمایش» نامیده می شود. با تنظیم این حاشیه که در قسمت بالا و پایین صفحه نیز وجود دارد (شکل۱-۸) می توان تصویر را دقیقاً در وسط صفحه نمایش مشاهده نمود.



شکل ۱-۸- ناحیه خارج از نمایش

۲.۴.۱. صفحه نمایش با پویش تصادفی

عملکرد یک صفحه نمایش با پویش تصادفی بدین گونه است که باریکه پرتو الکترونی تنها بر روی بخشی از صفحه تابیده می شود که قرار است تصویری در آن ترسیم گردد، و در هر بار تنها یک خـــط از تصویر، رسم می شود، به همین دلـیل به آن صـفحه نمایش برداری (Vector) نیز می‌گویند. اجزاء خطوط تصویر می توانند با هر ترتیبی ترسیم و بازسازی گردند، عملکرد این گونه صفحه نمایش ها را می توان بسیار شبیه به عملکرد یک پلاتر یا یک قلم سخت افزاری دانست. (شکل ۱-۹)



شکل ۱-۹- در سیستم‌های نمایش با پویش تصادفی، قطعه خطوط اشکال می توانند با هر ترتیبی ترسیم گردند.

نرخ بازسازی تصویر، به تعداد خطوطی که نمایش می یابد وابسته است. تصویر بصورت تعریفی از مجموعه دستورات خطوط ترسیمی در حافظه نگهداری می شود، که این قسمت از حافظه، لیست نمایش یا بافر بازسازی نامیده می شود. برای نمایش تصویر، سیستم به صورت دوره‌ای، اجزاء تصویر را از بافر برداشته و ترسیم می کند. پس از پایان یک دوره ترسیم، مجدداً در بازسازی بعدی، از نقطه ابتدا شروع شده و ترسیم، از نو آغاز می گردد. صفحه نمایش‌های با پویش تصادفی به گونه ای طراحی شده‌اند که بتوانند کلیه خطوط مؤلفه تصویر را ۳۰ تا ۶۰ بار در ثانیه ترسیم نمایند. سیستم‌های برداری باکیفیت بالا قادر هستند با نرخ تقریباً ۱۰۰/۰۰۰ قطعه خط درثانیه، تصویر را بازسازی کنند. نرخ بازسازی سریع‌تر می تواند باعث فرسایش لایه فسفرگردد.

نکته:

ثابت ماندن تصویر بدین مفهوم است که نقاط خاصی از غشاء فسفری مستقیماً در معرض تابش یکنواخت پرتو الکترونی قرار گیرد. این امر در طولانی مدت باعث فرسایش غشاء فسفری و افت کیفیت لامپ تصویر می گردد. از همین رو سعی می شود از بروز چنین وضعیتی جلوگیری شود. این کار، گاه به صورت سخت‌افزاری، با خاموش شدن مانیتور و گاه توسط سیستم عامل کنترل می شود. به این فرایند، محافظت از صفحه نمایش (Screen Saver) گویند.

صفحه نمایش‌های با پویش تصادفی برای کاربردهایی که در آن، خطوط ترسیم می گردند، بسیار مناسب است (مانند تجهیزات آزمایشگاهی مثل اسیلوسکوپ) اما نمی توانند تصاویر سایه دار واقعی را به خوبی نشان دهند. دلیل آن هم این است که تصویر به صورت مجموعه‌ای از خطوط ترسیمی ذخیره می شود که نمی تواند تقریب مناسبی از مقادیر همه نقاط صفحه نمایش ارائه دهد. صفحه نمایش‌های با پویش تصادفی عموماً نسبت به صفحه نمایش‌های پویش رستر، از وضوح بالاتری برخوردارند. خطوط ترسیمی در این نوع مانیتورها نیز بدلیل تابش مستقیم پرتو الکترونی به نقاط مورد نظر، صاف تر به نظر رسیده و مسیر خطوط هموارتر است.

۳.۴.۱. مانیتورهای CRT رنگی

درمانیتور های CRT رنگی، تصاویر رنگی به واسطه تلفیق پرتوها، که با رنگ های مختلف به تابش در می آیند، نمایش داده می شود. می توان طیف بسیار گسترده ای از رنگ ها را بوجود آورد. برای ایجاد تصاویر رنگی در یک مانیتور CRT دو فن‌آوری پایه بنام منافذ پرتوی و ماسک سایه وجود دارد.

۱.۳.۴.۱. روش منافذ پرتوی

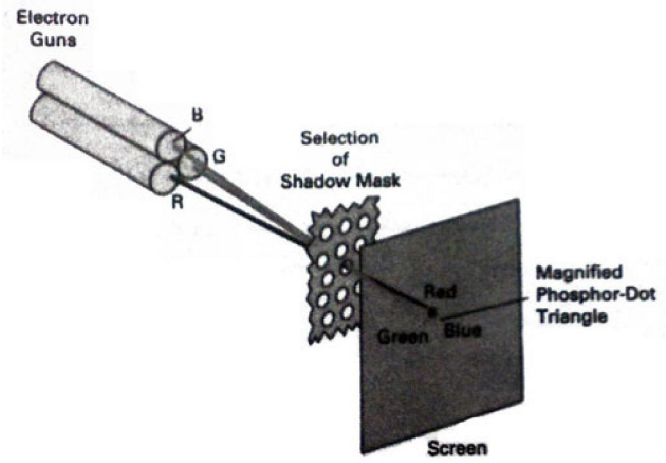
روش منافذ پرتوی اغلب برای نمایش تصاویر رنگی در مانیتورهای با پویش تصادفی مورد استفاده قرار می‌گیرد. در چنین مانیتورهایی عموماً دو لایه فسفری قرمز و سبز بر روی یکدیگر، صفحه را پوشانده‌اند. رنگ نمایش داده شده بر حسب اینکه پرتوی الکترونی تا چه اندازه به لایه ی فسفر نفوذ می کند، متفاوت خواهد بود.

پرتوی الکترونی کند، فقط لایه ی بیرونی یعنی قرمز را برانگیخته می کند، در حالیکه یک پرتوی الکترونی سریع، از لایه ی قرمز عبور می کند ولایه‌ی درونی سبز رنگ را برانگیخته می کند و پرتوی با سرعت

متوسط، ترکیبی از نور قرمز و سبز منتشر می‌کند که حاصل آن رنگهایی مثل نارنجی و زرد، خواهد بود. سرعت الکترونها هم به وسیله‌ی ولتاژ تسریع کننده‌ی الکترون تنظیم می شود.

۲.۳.۴.۱. روش ماسک سایه

روش ماسک سایه عموماً در صفحه نمایش‌هایی با پویش رستر و تلویزیون‌های رنگی بکار می رود زیرا طیف گستره‌تری از رنگها را در این روش می توان بوجود آورد. در مانیتورهای ماسک سایه، سه نقطه رنگی فسفری در هر پیکسل وجود دارد. قرمز، آبی و سبز. این نوع از مانیتورهای CRT دارای سه تفنگ الکترونی هستند که هریک، نوعی رنگ را در هر نقطه بوجود می آورند و یک ماسک سایه مشبک دقیقاً در پشت صفحه فسفری قرار دارد. در شکل ۱-۱۰ نمونه ای از ساختمان مانیتورهای ماسک سایه را مشاهده می کنید. ساختمان اغلب صفحه نمایش های CRT معمولی که مورد استفاده قرار می دهید، این گونه است. سه پرتو الکترونی بصورت گروهی در راستای یک نقطه در صفحه مشبک قرار گرفته و بر روی یک نقطه در صفحه متمرکز شده و تابش می شوند.



شکل ۱-۱۰- عملکرد یک CRT دلتا- دلتا با ماسک سایه. سه تفنگ الکترونی با الگوی نقاط رنگی در صفحه، در یک مسیر قرار گرفته و مستقیماً یک مثلث رنگی را در هر لحظه ماسک سایه به وجود می آورند

بدین ترتیب وقتی سه پرتو از یکی از حفره های ماسک سایه عبور می کنند، یک مثلث را با سه خال رنگی کوچک در صفحه پدیدار می کنند. نقاط فسفری به گونه‌ای قرار گرفته‌اند که هر پرتو الکترونی می تواند تنها نقطه رنگی خود را با گذر از ماسک، در معرض تابش قرار دهد.

در مانیتور های ماسک سایه می توان با تنظیم سطح تابش هر یک از پرتوهای الکترونی، طیف بسیار متنوعی از رنگ ها را بوجود آورد. به عنوان مثال رنگ زرد تنها با روشن شدن نقاط سبز و قرمز، پدید می آید. به همین ترتیب با روشن شدن نقاط آبی و قرمز، رنگ بنفش را می توان پدید آورد. در بعضی از سیستم های نمایشی ارزان قیمت، تنها می‌توان هر پرتو الکترونی را روشن و خاموش نمود که بدین ترتیب تنها امکان نمایش ۸ رنگ متنوع بوجود می آید. در سیستم‌های حرفه‌ای‌تر، میزان تابش به نحو دقیق‌تری قابل تنظیم است که بدین ترتیب امکان نمایش میلیون ها رنگ متنوع پدید می آید.

امروزه CRT های مسطح نیز ساخته شده و بسیار رایج هستند. در این نمایشگرها، پرتوهای الکترونی بصورت موازی شتاب داده شده و با زاویه ۹۰ درجه به صفحه مسطح مقابل برخورد می کنند.

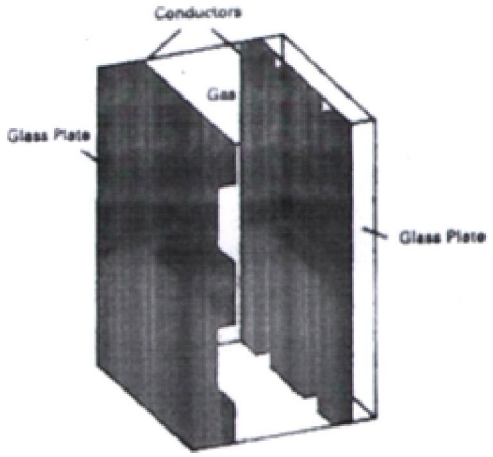
۴.۴.۱. صفحه نمایش های مسطح

اغلب صفحه نمایش‌های کنونی، صفحه نمایش‌های مبتنی بر ساختمان CRT هستند اما با ظهور فن-آوری دیگری به نام « صفحه نمایش‌های مسطح »، بتدریج این صفحه نمایش‌ها در حال منسوخ شدن هستند. اصطلاح صفحه نمایش مسطح، به آن دسته از نمایشگرهای ویدئویی اطلاق می گردد که کم حجم-تر و سبک‌تر بوده و نسبت به CRT ها، انرژی مصرفی پایین تری دارند. یکی از بارزترین ویژگی‌های صفحه نمایش‌های مسطح، آن است که بسیار باریک هستند، به گونه‌ای که به راحتی قابل نصب در دیوار بوده و حتی می توان آنها را به مچ دست بست. حتی امکان نوشتن بر روی نوعی از این صفحه نمایش‌ها نیز فراهم آمده است، و به زودی از آنها می توان بصورت دفترچه یادداشت استفاده نمود . امروزه استفاده از صفحه نمایش‌ها در معابری که با کمبود جا مواجه هستند، مانند آسانسورها، هواپیماها، بوردهای تبلیغاتی، مانیتورهای قابل حمل و ... کاربردهای فراوانی دارد.

صفحه نمایش‌های مسطح را می توان به دو دسته، تقسیم بندی نمود: ۱. نمایشگرهای ساطع کننده، ۲. نمایشگرهای غیر ساطع کننده. صفحه نمایش های ساطع کننده، نمایشگرهایی هستند که انرژی الکترونیکی را به نور تبدیل می کنند. پانل‌های پلاسما، صفحات نورافشان، نمونه‌هایی از اینگونه نمایشگرها هستند. صفحه نمایش‌های غیر ساطع کننده، از اثرات تابشی چشمه های نوری محیط، مانند نور خورشید استفاده کرده و آنها را به الگوهای گرافیکی تبدیل می کنند. یکی از مهمترین انواع صفحه نمایش های مسطح غیر ساطع کننده، نمایشگر کریستال مایع است که به آنها نمایشگرهای تخلیه گازی نیز گفته می شود. چهار نوع اصلی صفحه نمایش‌های مسطح بصورت زیر می باشند.

۱.۴.۴.۱. صفحه نمایش پلاسما:

اساساً ساختمان این نوع صفحه نمایش‌ها را دو صفحه مسطح تشکیل می دهد که بین آنها با ترکیبی از گازها، که بخش اعظم آن را نئون تشکیل می دهد، پرشده است. مجموعه ای از الکترودها به صورت نوارهایی افقی برروی یکی از صفحات شیشه‌ای و مجموعه‌ی دیگری بصورت نوارهایی عمودی بر روی صفحه دیگر قرار داده شده‌اند (شکل ۱– ۱۱). با برقراری اختلاف ولتاژ بین الکترودهای عمودی و افقی، گاز موجود بین فضای این دو، به پلاسمایی درخشان از یون ها و الکترون ها تبدیل می گردد. الگوی تصویری که در بافر بازسازی ذخیره شده، توسط ولتاژ آستانه، موجب بازسازی موقعیت پیکسل ها در حد فاصل بین الکترودها می گردد. این بازسازی می تواند در هر ثانیه تا ۶۰ مرتبه، تکرار گردد.

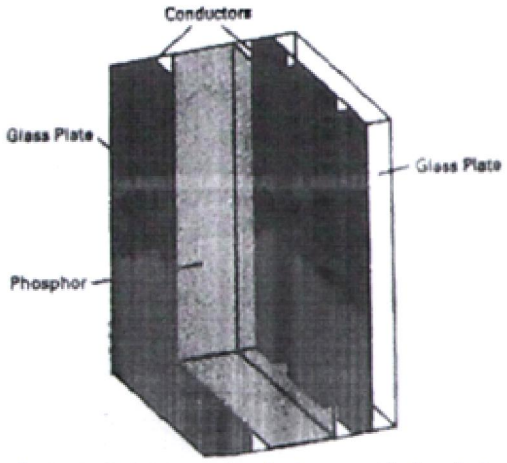


شکل ۱–۱۱– الگوی ساده‌ای از ساختمان نمایشگر مسطح پلاسما

۲.۴.۴.۱. صفحه نمایش نور افشان الکتریکی:

نمایشگرهای نورافشان الکتریکی نوع دیگری از نمایشگرهای مسطح با عمق کم هستند که ساختمانی شبیه نمایشگرهای پلاسما دارند. مهمترین اختلاف بین ساختمان این دو نوع نمایشگر در ماده حائل بین الکترودها است. این ماده در نمایشگرهای نورافشان، عموماً از فسفر و یا سولفید روی ترکیب شده با منگنز می باشد. (شکل۱–۱۲). وقتی ولتاژ عبوری بین جفت الکترودهای گذرنده از یک نقطه، به میزان مؤثری می رسد، فسفر در ناحیه بین الکترود ها به یک هادی تبدیل شده وانرژی الکتریکی توسط اتم‌های منگنز جذب می گردد. انرژی جذب شده بصورت نقطه نورانی شبیه آنچه در نمایشگرهای پلاسما شرح دادیم، آزاد

مـی گردد. این گونه نمایشگرها برای تغذیه، به انرژی الکتریکی بیشتری نسبت به نمایشگرهای پلاسما، نیاز داشته و به سختی می توان به کیفیت مطلوب رنگ وسطوح خاکستری در آنها دست یافت.



شکل ۱-۱۲- الگوی ساده‌ای از ساختمان نمایشگرهای نورافشان الکتریکی

۳.۴.۴.۱. صفحه نمایش LED[1]:

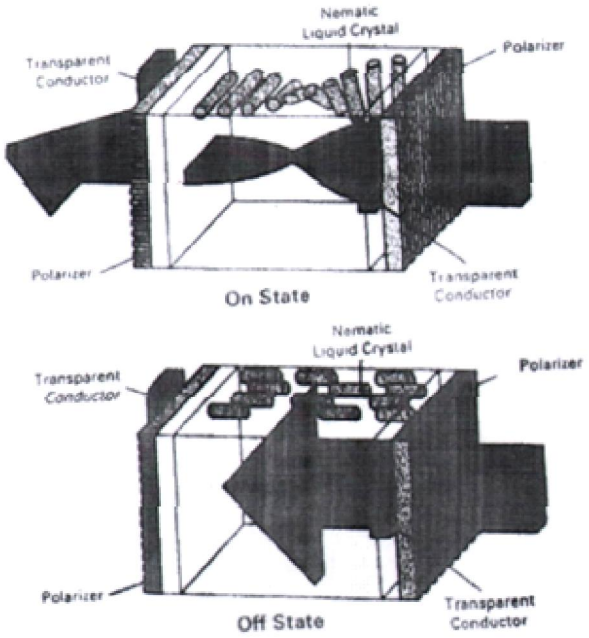
نوع سوم نمایشگرهای مسطح، دیودهای نورافشان LED می باشند. در این گونه نمایشگرها، به جای هر پیکسل، یک دیود نورانی کار گذاشته شده است. بدین ترتیب ماتریسی از دیودها را در اختیار خواهیم داشت که روشن شدن هر یک از آنها به الگوی تصویر ذخیره شده در بافر باز سازی، بستگی دارد. در اینجا نیز بازسازی تصویر می تواند به صورت خط پویش صورت گیرد.

۴.۴.۴.۱. صفحه نمایش کریستال مایع (LCD[2]):

صفحه نمایش‌های کریستال مایع (LCD)، نمایشگرهایی هستند که عموماً در سیستم‌های کوچک نظیر ماشین حساب‌ها، نمایشگر کامپیوترهای قابل حمل و مواد این چنینی، کاربرد دارند. در این نمایشگرها، از ترکیب مایعی استفاده می شود، به گونه‌ای که این مایع دارای ساختار مولکول قطبی است که بین دو الکترود شفاف با قابلیت گذردهی قرار می گیرد. هرگاه یک میدان الکتریکی در این ترکیب بوجود آید مولکول‌ها بصورت منظم دور میدان قرار گرفته و آرایشی کریستال مانند ایجاد می کنند، به گونه ای که نور گذرنده از آن را قطبی می کند و یک فیلتر قطبی شده روی بلاک های الکترود نور قطبی قرار می گیرد. بدین ترتیب

[1] . *Light Emitting Diode*
[2] . *Liquit Crystal Displays*

یک شبکه از الکترودها، یک نقطه از تصویر (پیکسل)، را روشن می کنند. یعنی آن را به رنگ تیره تبدیل می کنند. (به نمایش اعداد در ماشین حساب توجه کنید). در شکل ۱-۱۳ دو وضعیت روشن وخاموش یک مؤلفه نمایشی نشان داده شده است. در برخی صفحه نمایش‌های کریستال مایع، صفحه‌ای با تابش الکتریکی در پشت صفحه نمایش قرار داده شده است تا این صفحه را روشن کند. با اعمال پاره ای تغییرات در ساختمان این گونه نمایشگرها، قابلیت ایجاد رنگ را در آنها به وجود آورده اند.



شکل ۱-۱۳- پیچش پرتوها، اساس کار اغلب صفحه نمایش‌های کریستال مایع را تشکیل می دهد

۵.۱ . کارت گرافیک

یک کارت گرافیک پیشرفته، یک برد مدار چاپی به همراه حافظه و یک پردازنده اختصاصی است. پردازنده با هدف انجام محاسبات مورد نیاز گرافیکی، طراحی شده است. اکثر پردازنده‌های فوق دارای دستورات اختصاصی بوده که به کمک آنها می توان عملیات گرافیک را انجام داد. کارت گرافیک دارای اسامی متفاوتی نظیر کارت ویدئو، برد ویدئو، برد نمایش ویدئویی، برد گرافیک، آداپتور گرافیک و آداپتور ویدئو است.

شکل ۱-۱۴- کارت گرافیکی با رابط PCI-EXPRESS

شکل ۱-۱۵- کارت گرافیکی با رابط AGP

۱.۵.۱. اجزای کارت گرافیک

۱.۱.۵.۱. حافظه

اولین چیزی که یک کارت گرافیک به آن نیاز دارد، حافظه است. این حافظه مجموع اطلاعات مورد نیاز پیرامون هر صفحه نمایش را نگهداری می کند. همانطور که قبلا گفته شد برای ذخیره سازی رنگ هر پیکسل، در صفحه نمایش تک رنگ، از یک بیت، و در صفحه نمایش رنگی، بسته به وضوح صفحه، ممکن است چندین مگابایت استفاده گردد.

۲.۱.۵.۱. رابط کامپیوتر

این رابط جهت تغییر محتویات حافظه کارت گرافیک است. امکان فوق با اتصال کارت گرافیک به گذرگاه مربوطه بر روی برد اصلی تحقق پیدا خواهد کرد. کامپیوتر قادر به ارسال سیگنال، از طریق این گذرگاه برای تغییر محتویات حافظه خواهد بود. در گذشته از گذرگاه‌های ISA، PCI و AGP، استفاده می شد، اما امروزه از PCI-Express، استفاده می شود، که سرعت انتقال اطلاعات در آن چندین برابر گذرگاه-های قبلی است.

۳.۱.۵.۱. رابط ویدئو

این رابط به منظور تولید سیگنال برای مانیتور است. کارت گرافیک می بایست سیگنال‌های رنگی را تولید، تا باعث حرکت اشعه در CRT گردد. فرض کنید که صفحه نمایش در هر ثانیه شصت فریم را بازخوانی/ بازنویسی می نماید، این بدان معنی است که کارت گرافیک تمام حافظه مربوطه را بیت به بیت اسکن و این عمل را شصت مرتبه در ثانیه انجام می دهد. سیگنال‌های مورد نظر برای هر پیکسل موجود بر هر خط، ارسال، و در ادامه یک پالس افقی Sync، نیز ارسال می گردد. عملیات فوق برای ۴۸۰ خط تکرار شده و در نهایت یک پالس عمودی Sync ارسال خواهد شد.

۴.۱.۵.۱. پردازنده‌های کمکی گرافیک

یک کارت گرافیک ساده، Frame Buffer نامیده می شود. در این کارت، یک فریم از اطلاعاتی که برای نمایشگر ارسال شده است نگهداری می شود. ریزپردازنده کامپیوتر «CPU» مسئول بهنگام سازی هر بایت در حافظه کارت گرافیک است. در صورتی که عملیات گرافیکی پیچیده‌ای داشته باشیم، ریزپردازنده کامپیوتر، مدت زمان زیادی را صرف بهنگام سازی حافظه کارت گرافیک کرده و برای سایر عملیات مربوطه، زمانی باقی نخواهد ماند. مثلا اگر یک تصویر سه بعدی دارای ۱۰۰۰۰ ضلع باشد، ریزپردازنده می بایست هر ضلع را رسم، و عملیات مربوطه در حافظه کارت گرافیک را نیز انجام دهد. عملیات فوق، زمان بسیار زیادی را طلب می کند.

کارت‌های گرافیک جدید، به طرز قابل توجهی، عملیات مربوط به پردازنده اصلی کامپیوتر را کاهش می‌دهند. این نوع کارت‌ها دارای یک پردازنده اصلی پر قدرت بوده که مختص عملیات گرافیکی طراحی شده است. با توجه به نوع کارت گرافیک، پردازنده فوق می تواند یک «شتاب دهنده گرافیکی» یا «کمک پردازنده گرافیکی» باشد.

در صورتی که از شتاب دهنده گرافیکی استفاده شود، دستورات لازم از طریق پردازنده اصلی برای شتاب دهنده ارسال می شود و شتاب دهنده مسئولیت انجام آنها را بر عهده خواهد داشت. در این حالت درایور کارت گرافیک هر چیز را در ابتدا برای پردازنده اصلی کامپیوتر ارسال می کند، و پردازنده اصلی کامپیوتر، شتاب دهنده گرافیکی را به منظور انجام عملیات خاصی هدایت می کند. مثلا پردازنده اصلی ممکن است به شتاب دهنده اعلام نماید که "یک چند ضلعی رسم کن" در ادامه، شتاب دهنده فعالیت تعریف شده فوق را انجام خواهد داد.

اگر از کمک پردازنده گرافیکی استفاده شود، درایور کارت گرافیک، عملیات مربوط به کارهای گرافیکی را مستقیما برای کمک پردازنده ارسال می کند.

Graphic BIOS .۵.۱.۵.۱

کارت‌های گرافیک، دارای یک تراشه کوچک BIOS می باشند. اطلاعات موجود در تراشه فوق، به سایر عناصر کارت، نحوه انجام عملیات (مرتبط به یکدیگر) را تبیین خواهد کرد. BIOS همچنین مسئولیت تست کارت گرافیک (حافظه مربوطه و عملیات ورودی و خروجی) را بر عهده خواهد داشت.

Digital- To- Analog- Converter (DAC) .۶.۱.۵.۱

تبدیل کننده فوق را RAMDAC نیز می گویند. این قسمت تبدیل سیگنال‌های دیجیتال به آنالوگ را انجام می دهد، و سرعت آن تاثیر مستقیمی در ارتباط با مشاهده تصویر بر روی صفحه نمایشگر خواهد داشت.

۲.۵.۱. استانداردهای کارت گرافیک

اولین کارت گرافیک در سال ۱۹۸۱ توسط شرکت IBM عرضه گردید. کارت فوق بصورت تک رنگ و با نام Monochrome Display Adapters (MDAs) ارائه گردید. صفحات نمایشگری که از کارت فوق استفاده می کردند، متنی بودند. رنگ نوشته، سفید یا سبز، و زمینه سیاه بود. در ادامه کارت‌های چهار رنگ Hercules Graphic Card(HGC) ارائه گردید. سپس کارت‌های هشت رنگ Color Graphic Adapter (CGA) و کارت‌های شانزده رنگ Enhanced Graphic Adapter (EGA) ارائه گردیدند. تولید کنندگان دیگر نظیر کمودور، کامپیوترهایی را معرفی کردند که دارای کارت‌های گرافیک از قبل تعبیه شده و ساخته شده در سیستم بودند. این کارت‌ها قادر به نمایش تعداد زیادی رنگ بودند.

زمانی که شرکت IBM در سال ۱۹۸۷ کارت Video Graphic Array (VGA) را معرفی کرد، استاندارد جدیدی در این راستا مطرح گردید. نمایشگرهای VGA قادر به ارائه ۲۵۶ رنگ و وضوح تصویر ۴۰۰ * ۷۲۰ بودند. یک سال بعد، استاندارد Super Video Graphic Array (SVGA) مطرح گردید. استاندارد فوق قادر به ارائه ۸/۱۶ میلیون رنگ با وضوح تصویر ۱۰۲۴ * ۱۲۸۰ است.

۳.۵.۱. ویژگی‌های کارت گرافیک

۱.۳.۵.۱. سرعت

تعویض سریع صفحات در برنامه های واژه پرداز و در برخی دیگر از برنامه‌های کاربردی، حائز اهمیت است. به عنوان مثال هنگامی که قصد دارید از ویرایش یک سند بلافاصله به ویرایش سند دیگری بپردازید، می بایست اطلاعات یک صفحه کامل (یعنی ۲۰۰۰ کاراکتر =۸۰ × ۲۵ سطر و ستون) در یک لحظه از طریق کارت ویدیویی ارسال گردد. این مطلب در ارسال یک صفحه کامل گرافیکی با وضوح بالا از اهمیت بیشتری برخوردار است. بعنوان مثال یک کارت گرافیکی رنگی با وضوح ۱۰۲۴ × ۱۲۸۰ نیازمند ارسال اطلاعات حدود ۱.۳۰۰.۰۰۰ پیکسل گرافیکی در یک لحظه است.

۲.۳.۵.۱. مود گرافیک **ومتن**

صفحه نمایش در دو مود متن و گرافیک کار می کند. مانیتور نمی تواند اختلاف بین این دو را تشخیص دهد و تنها اطلاعات گرافیکی ارسال شده توسط کنترلر کارت گرافیک را پردازش می کند. در مود گرافیک، تنها رنگ پیکسل‌ها (به ازاء هر پیکسل یک یا چند بیت) از طریق حافظه ویدیویی به مانیتور ارسال می شود. مود متن از شیوه دیگری استفاده می کند. در این مود، به ازاء هر موقعیت در صفحه نمایش، کد اسکی یک کاراکتر در حافظه ویدیویی ذخیره می شود. وقتی کنترلر ویدیو صفحه‌ای را نمایش می دهد، الگوی کاراکتر متناسب با کد اسکی هر کاراکتر از حافظه موجود بر روی کارت گرافیکی به دست می آید، سپس این الگو به ماتریس پیکسلی کاراکتر تبدیل می گردد. نهایتاً این الگو برای ظهور در صفحه، به مانیتور ارسال می گردد.

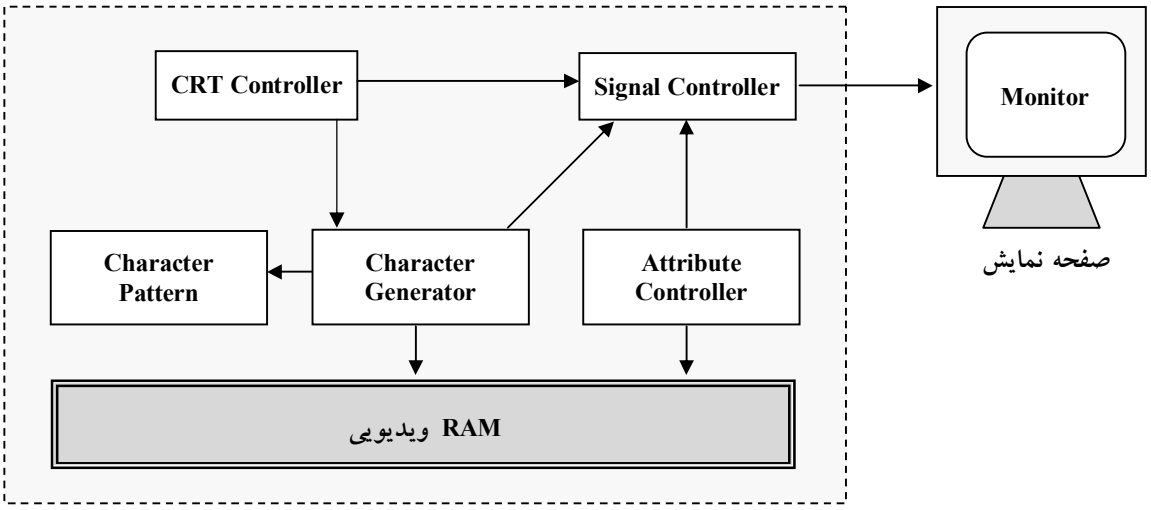
مود متن از مجموعه‌ای ۲۵۶ کاراکتری استفاده می کند. این کاراکترها به ترتیب از ۰ تا ۲۵۵ شماره گذاری شده و جهت رسم، کافی است که موقعیت نمایش هر کاراکتر در صفحه تعیین گردد.

۳.۳.۵.۱. بایت خصوصیت (Attribute Byte)

هر مکان در مود متن، در صفحه نمایش، دارای یک بایت خصوصیت است که ویژگی نمایشی کاراکتر (نظیر زیرخطدار بودن- چشمک‌زن بودن- سایه‌دار بودن و...) را تعیین می کند. این بدین مفهوم است که برای نمایش هر موقعیت نیازمند دو بایت هستیم. در مود گرافیک، تعداد بیت‌های مورد نیاز بستگی به وضوح تصویر و تنوع رنگ دارد.

۴.۵.۱. ساختار سخت افزاری کارت گرافیگ

درشکل ۱-۱۶ ساختار سخت افزاری اجزاء تشکیل دهنده کارت گرافیک نشان داده شده است. نقطه شروع ایجاد تصویر، همیشه Ram ویدیویی (حافظه ویدیویی) می باشد. Ram ویدیویی حاوی اطلاعاتی پیرامون کاراکترهای مورد نمایش و خصوصیات نمایش آنها (مانند رنگ - سبک نمایش و...) می باشد.



شکل ۱-۱۶- نمودار بلوکی از یک کارت گرافیک

ابتدا بخش تولید کاراکتر (Character Generator) با دسترسی به Ram ویدیویی، کاراکترها را خوانده و با استفاده از جدول الگوی کارکترها (Character Pattern) نگاشت بیتی کاراکترها را برای نمایش بوجود می آورد. آنگاه کنترل کننده صفات کاراکترها (Attribute Controller)، اطلاعات خصوصیات کاراکتری را از Ram ویدیویی برداشت می کند. حال هر دو ماژول فوق این اطلاعات را به بخش کنترل کننده سیگنال (Signal Controller) ارسال می کنند.

کنترل کننده سیگنال نیز، عـــلائم مناسب را با ترکیب این دو اطلاعـات بوجود آورده و به مانیتور ارســـال می کند. بخش کنترل کننده سیگنال، توسط واحد دیگری به نام CRT Controller، کنترل می شود. این واحد، در واقع نقطه مرکزی عملکرد یک کارت گرافیک است که از وظایف آن، می توان به تولید مکان نما، کنترل Ram ویدیویی و کنترل قلم های نوری اشاره کرد. معمولاً از CRT Controller در کنار Ram ویدیویی و مانیتور، به عـنوان اجزاء اصـــلی تشکیل دهنده تصویر نام مـی برند.

فصل دوم

الگوهای ریاضی ترسیمات گرافیکی

۱.۲. توابع کتابخانه‌ای

در C استاندارد هیچگونه تابعی در مورد گرافیک و یا سایر توابع در مورد صفحه نمایش پیش بینی نشده است. ولی توربو C دارای چندین تابع کتابخانه‌ای در این زمینه است که در این قسمت مورد بحث قرار می‌گیرند. الگوی توابع گرافیکی در قایل graphics.h و الگوی سایر توابع در مورد صفحه نمایش که دراین قسمت مورد بحث قرار می‌گیرند در فایل conio.h قرار دارند. برای درک توابع گرافیکی و دیگر توابع صفحه نمایش، باید مفهوم پنجره[1] مشخص گردد، زیرا بسیاری از توابع با آن سر و کار دارند. پنجره، قسمتی از صفحه نمایش است که خروجی برنامه در آنجا قرار می‌گیرد. پنجره می‌تواند به اندازهٔ صفحه نمایش (در حالت عادی) و یا کوچکتر از آن باشد. به پنجره، در حالت گرافیکی «محدوده گرافیک»[2] گفته می‌شود.

صفحه نمایش کامپیوتر به دو صورت می‌تواند مورد استفاده قرار گیرد:

۱- حالت متن (text)

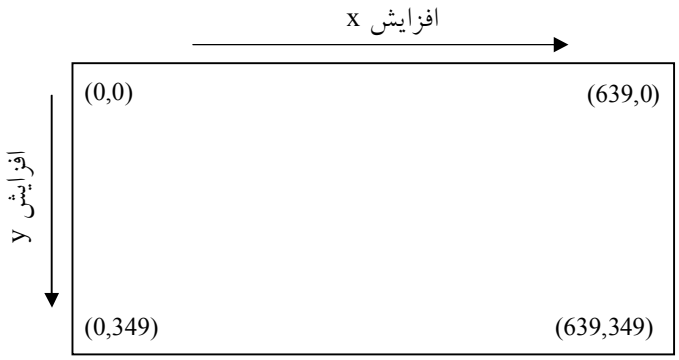
۲- حالت گرافیکی

هر یک از حالتهای متن و گرافیک می‌توانند وضعیتهای متفاوتی داشته باشند. همانطور که می‌دانید در ریز کامپیوترها بوردهای گرافیکی مختلفی مورد استفاده قرار می گیرند که بعضی از آنها عبارتند از: ۱. monochorome ۲. CGA ۳. EGA ۴. VGA، هر کدام از بوردهای گرافیکی، وجوه (حالات) مختلفی را برای صفحه نمایش فراهم می‌کنند که در ادامه این فصل مورد بررسی قرار می گیرند.

[1] - *Window*
[2] - *Viewport*

بعضی از وجوه صفحه نمایش مربوط به متن و بعضی دیگر مربوط به گرافیک است. در وجوه گرافیکی امکان ظاهر شدن متن در صفحه نمایش وجود دارد ولی در وجوه متن، انجام امور گرافیکی ممکن نیست.

کوچکترین واحد قابل دسترسی (آدرس دهی) در وجه گرافیکی، Pixel نام دارد. مختصات گوشه بالای سمت چپ صفحه نمایش در وجه گرافیکی (0,0) است. مختصات هر نقطه از صفحه نمایش با دو مقدار x و y مشخص می‌شود که x محور افقی و y محور عمودی است (x,y) شکل (۲-۱).



شکل ۲-۱- وضعیت صفحهٔ نمایش در یکی از وجوه گرافیکی (۳۵۰×۶۴۰)

در وجوه گرافیکی همانند وجوه متن می‌توان از امکان رنگ‌آمیزی استفاده نمود که چگونگی این عمل، در ادامه این فصل مورد بررسی قرار می‌گیرد. در حالت متن رنگ‌های زمینه و متن، و در حالت گرافیکی رنگ‌های زمینه و شکل گرافیکی قابل تغییر می‌باشند. انتخاب رنگ شکل گرافیکی با استفاده از جعبه رنگ انجام می‌شود. بعنوان مثال، در بورد گرافیکی CGA تعداد ۴ جعبه رنگ وجود دارد که از ۰ تا ۳ هستند و شماره رنگ ۰، همان رنگ زمینه می‌باشد جدول (۲-۱).

شماره رنگ / جعبه رنگ	۱	۲	۳
۰	سبز	قرمز	زرد
۱	کبود	بنفش	سفید
۲	سبز روشن	قرمز روشن	زرد
۳	کبود روشن	بنفش روشن	سفید

جدول ۲-۱- رنگهای موجود در جعبه رنگ بورد CGA

در بوردهای گرافیکی EGA، VGA و SVGA، هر جعبه رنگ دارای ۱۶ رنگ است که رنگهای آنها قابل تغییر است. چون ۴ جعبه رنگ وجود دارد، ۶۴ رنگ قابل استفاده است.

۱.۱.۲. تابع ( ) initgraph

تابع initgraph برای انتقال یک مبدل گرافیک مناسب به حافظه مورد استفاده قرار می‌گیرد. قبل از انجام هر کار گرافیکی باید یک مبدل گرافیک مناسب به حافظه منتقل شود، زیرا در غیر اینصورت هیچک از توابع گرافیکی عمل نخواهند کرد. تابع ( ) initgraph دارای الگوی زیر است:

Void far initgraph (int far *driver, int far *mode, char far *path)

در الگوی فوق، driver به درایور کارت گرافیکی که کاربر استفاده می کند، اشاره دارد. با مقدار دهی آن به DETECT، کامپایلر، خود بهترین درایور موجود را انتخاب می کند. فایلهای مبدل گرافیک دارای انشعاب BGI. می‌باشند. برای معرفی این فایلها لزومی به حفظ کردن اسامی آن نیست بلکه می‌توان از شماره‌ها و ماکروهایی که در فایل graphics.h تعریف شده‌اند استفاده نمود (جدول ۲-۲).

نام ماکرو	شماره	نام ماکرو	شماره
DETECT	0	EGAMONO	5
CGA	1	IBM8514	6
MCGA	2	HERCMONO	7
EGA	3	ATT400	8
EGA64	4	VGA	9
		PC3270	10

جدول ۲-۲- مبدل‌های گرافیک

Mode، وجه گرافیکی (مود گرافیکی) مورد نظر را تعیین می‌کند. مقادیر و ماکروهای معتبر برای mode در جدول ۲-۳ آمده‌اند.

مولفه سوم، مسیر فایلی بنام EGAVGA.BGI است، که در پوشه TC\BGI قرار دارد، اگر این فایل به پوشه BIN، منتقل شود، در این قسمت مسیری نوشته نمی شود و فقط از علامت " " استفاده می شود.

مبدل	وجه گرافیکی (ماکرو)	معادل عددی وجه گرافیکی	دقت
CGA		0	320 × 200
		1	320 × 200
		2	320 × 200
		3	320 × 200
		4	640 × 200
MCGA		0	320 × 200
		1	320 × 200
		2	320 × 200
		3	320 × 200
		4	640 × 200
		5	640 × 480
EGA		0	640 × 200
		1	640 × 350
EGA64		0	640 × 200
		1	640 × 350
EGAMONO		3	640 × 350
HERC		0	720 × 348
ATT400		0	320 × 200
		1	320 × 200
		2	320 × 200
		3	320 × 200
		4	640 × 200
		5	640 × 400
VGA		0	640 × 200
		1	640 × 350
		2	640 × 480
PC3270		0	720 × 350
IBM8514		0	640 × 480
		1	1024 × 760

جدول ۲-۳: وجوه گرافیکی معتبر برای تابع initgraph()

بعنوان مثال، مجموعه دستورات زیر را در نظر بگیرید:

```
int driver,mode;
driver=DETECT;
mode=0;
initgraph (&driver, &mode, "");
```

مجموعه دستورات فوق موجب می‌شوند تا بورد گرافیکی توسط سیستم تشخیص داده شده و در وجه گرافیکی مناسبی قرار گیرد. چون بجای path رشته تهی قرار گرفته است، برای پیدا کردن مبدل مورد نیاز در مسیر جاری جستجو می‌شود. بعنوان مثال دیگر، مجموعه دستورات زیر را در نظر بگیرید:

```
int driver,mode;
driver=EGA;
mode=EGAHI;
initgraph (&driver, &mode, "");
```

مجموعهٔ دستورات فوق، مبدل گرافیک بورد EGA را به حافظه منتقل کرده و وجه گرافیکی را برابر با EGAHI قرار می‌دهد.

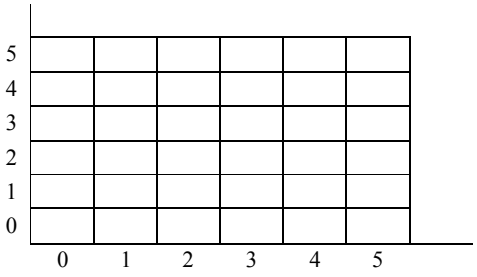
۲.۲. الگوهای ریاضی ترسیمات گرافیکی

برای توصیف یک تصویر در صفحه نمایش، راه‌های متعددی وجود دارد. همان گونه که دیدیم، می‌توان تصویر را مجموعه‌ای از نقاط همجوار دانست که هر یک ویژگی‌های نمایشی خود را دارند. برای تشریح ساده‌تر تصاویر ترسیمی، می‌توان آنها را مجموعه‌ای از اشکال ساده (مانند نقطه، پاره‌خط، منحنی و...) دانست که در کنار یکدیگر، اشکال پیچیده‌تر را به بوجود آورده‌اند. با توجه به اینکه پیکسل‌ها، کوچکترین عنصر نمایشی هستند، ترسیم شکل پایه می تواند به واسطه بارگذاری آرایه‌ای از پیکسل‌ها به بافر فریم و یا پویش نقاط سازنده با ساختار هندسی مشخص، صورت گیرد. از همین رو پرداختن به الگوهای هندسی اشکال پایه ترسیمی، میتواند بعنوان راه حلی جهت تعیین پیکسل‌های نمایش شونده در مسیر ترسیم اشکال باشد.

۱.۲.۲. رسم نقطه

نقطه، پایه‌ای ترین جز ترسیمی است. ترسیم نقطه می تواند به واسطه تبدیل مختصات در یک برنامه کاربردی برای یافتن موقعیت متناظر در صفحه نمایش صورت گیرد. به عنوان مثال در مانیتورهای CRT ، نقطه روشن شونده فسفری به واسطه تابش باریکه الکترونی تشکیل می یابد. در مانیتورهای پویشی تصادفی(vector) ، دستورالعمل های ترسیم نقاط در لیست نمایش ذخیره می شود و در حین چرخه بازسازی پرتو تابش، نقطه مربوطه را روشن می کند. در سیستم های پویشی سیاه و سفید ، ارزش بیت نقطه نمایشی در موقعیت مربوطه در بافر فریم، به مقدار یک تغییر می یابد. در پویش نقاط کل مانیتور، تابش بر نقاطی انجام می شود که ارزش بیت متناظر با آنها برابر با یک است. از دید کاربر ، مراجعه به این نقاط بر اساس شماره سطر و ستون مربوطه انجام می شود.

شکل غربال گونه از صفحه نمایش را در شکل ۲-۲ مشاهده می کنید:

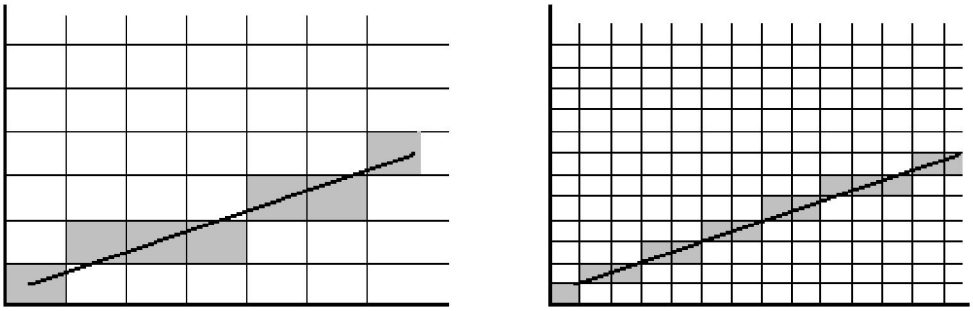


شکل ۲-۲ - موقعیت پیکسل بر اساس سطر و ستون پویش

۲.۲.۲. رسم خط

همانطور که گفته شد، ترسیم اشکال پایه هندسی به واسطه روشن شدن نقاطی به نام پیکسل در راستای شکل انجام می شود. برای ترسیم یک خط هموار مورب در صفحه نمایش با پویش تصادفی، هیچ مانعی وجود ندارد زیرا در این رسانه قیاسی(Analog) قلم ترسیمی می تواند هر راستایی از نقطه شروع تا پایان حرکت کند.

در صفحه نمایش رستر، ترسیم خط می بایست با روشن شدن نقاطی در راستای خط انجام شود. انجام این کار را می توان به ترسیم خط در یک کاغذ شطرنجی مقایسه کرد که در آن ارزش هر نقطه صفر یا یک است. طبیعی است که هرچه ارزش مقیاس گذاری نقاط در راستای x و y کوچکتر باشد، از نقاط بیشتری برای ترسیمات استفاده شده و خط مورب هموارتر به نظر خواهد رسید. (شکل ۲-۳)



شکل ۲-۳- ترسیم خط مورب؛ مقیاس گذاری کوچک، و مقیاس گذاری بزرگ

مقیاس گذاری محورهای مختصات، متناظر با قدرت تفکیک صفحه نمایش است. از این به بعد، به منظور ارجاع به نقاط، از موقعیت سطر و ستون آنها استفاده می کنیم. شماره سطر و ستون، در صفحه مختصات، از گوشه سمت چپ و پایین، به ترتیب از (0, 0)، و در صفحه نمایش، از گوشه سمت چپ و بالا، از (0, 0) شروع می شود. جهت رسم هر پیکسل، از تابع putpixel(x, y) استفاده می کنیم.

- **الگوریتم‌های ترسیم خط**

هدف از بررسی الگوریتم‌های ترسیم خط آن است که نقاط روشن شونده در راستای مسیر خط، به بهترین نحو، مشخص گردند به گونه‌ای که خط با تقریب دقیقی از خط واقعی ترسیم گردد.

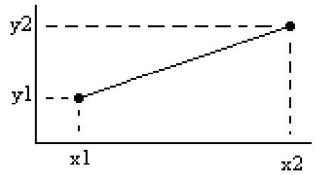
قبل از هر چیز به توصیف هندسی خط می پردازیم. معادله خط در هندسه به صورت زیر است:

$$y=m\ .x\ +b$$

m) شیب خط (ضریب زاویه) b) عرض از مبدا

در صورتی که نقاط (x_1, y_1) و (x_2, y_2) به ترتیب همانند شکل۲-۳ مختصات نقاط شروع و پایان خط باشند، مقادیر m و b از طریق مقادیر زیر بدست می آید:

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} \qquad b = y_1 - m.x_1$$



شکل ۲-۳- مسیر خط در راستای نقاط (x1, y1) الی (x2, y2)

تغییرات مقادیر x در راستای محور طول و تغییرات y در راستای محور عرض را می توان بصورت رابطه زیر بیان کرد:

$$\Delta y = m.\Delta x$$

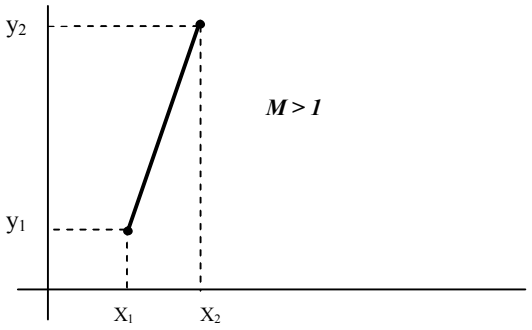
Δy: تغییرات در راستای y

Δx: تغییرات در راستای x

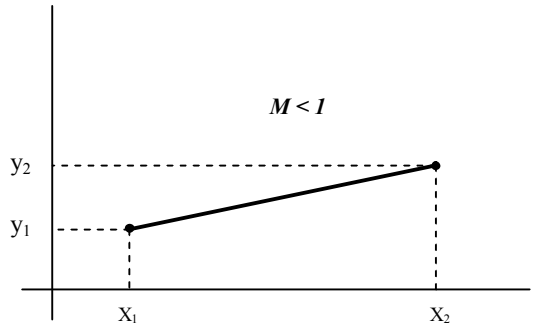
می توان تغییرات x (Δx) را نیز بر حسب تغییرات y (Δy) بیان کرد:

$$\Delta x = \frac{\Delta y}{m}$$

با استفاده از روابط فوق می توان مسیر و شیب خط را تحلیل نمود. در صورتی که شیب خط ملایم باشد (یعنی $|M| < 1$) می توان نتیجه گرفت که میزان تغییرات در راستای محور عمودی (y) نسبت به محور افقی (x) کمتر است. (شکل ۲-۵)



شکل ۲-۴: در خطی با شیب بیشتر از یک (M>1) تغییرات y نسبت به x بیشتر است



شکل ۲-۵: در خطی با شیب کمتر از یک (M<1) تغییرات x نسبت به y بیشتر است

بعنوان مثال:

For Fig 2.2: if (X_1= 1, X_2= 3) And (Y_1=1 , Y_2=5) Then M=Y_2-Y_1 / X_2-X_1= 4/2= 2/1= 2 => M>1

For Fig2.3: if (X_1= 1, X_2= 5) And (Y_1=1 , Y_2=3) Then M=Y_2-Y_1 / X_2-X_1= 2/4= 1/2= 0.5 => M<1

در خطوط با زاویه ۴۵درجه (یعنی $M = 1$ و $\Delta x = \Delta y$)میزان تغییرات x و y، کاملا متناسب است.

در سیستم‌های پویشی که خط به واسطه روشن شدن نقاط، ترسیم می گردد، اندازه گام‌ها برای پیشروی در راستای افقی و عمودی با تفکیک پیکسل ها متناسب است. بدین ترتیب می بایست به دنبال روشی بود که نزدیکترین پیکسل در راستای خط تعیین گردد. برای این منظور راستای تعیین پیکسل‌ها براساس محور با بیشترین تغییرات تعیین می شود. به عنوان مثال در شکل ۲–۵ با توجه به تغییرات زیاد در راستای محور x نسبت به محور y ها، تقریب در راستای محور طول انجام خواهد شد. بنابراین، گام افزایش x در هر مرحله ۱ واحد است.

۱.۲.۲.۲. الگوریتم *DDA*

الگوریتم تحلیل اختلافات رقومی (*DDA*) بر اساس محاسبه و تحلیل تغییرات در راستای x و y $(\Delta x, \Delta y)$ استوار است. به سادگی می توان در محور مختصات بر مبنای واحد صحیح، تغییرات را بدست آورده و نزدیکترین مقادیر صحیح نسبت به مسیر خط را تعیین نمود.

در ساده‌ترین حالت شکل ۲–۵ را با شیب مثبت در نظر بگیرید. در صورتی که شیب خط کوچکتر از یک است $(m < 1)$، واحد تغییرات x را می توان برابر یک ($x_{inc}=1$) در نظر گرفت و مقادیر متناظر y را با رابطه زیر به دست آورد:

اندیس k عدد صحیحی است که برای نقطه شروع برابر مقدار یک است و تا رسیدن به نقطه پایان در هر مرحله یک واحد افزایش می یابد. از آنجایی که عدد m ، هر مقدار حقیقی بین صفر و یک می تواند باشد، در محاسبه مقدار y ، ارزش آن از طریق رند کردن به نزدیکترین عدد صحیح به دست می آید.

در مورد خطوطی با شیب مثبت بزرگتر از یک $(m > 1)$، می توانیم نقش x و y را معکوس نماییم بنابراین هر واحد پیشروی (گام)، می تواند از طریق رابطه زیر به دست آید:

$$y_{k+1} = y_k + 1$$

توجه داشته باشید که در دو رابطه فوق فرض بر این بوده است که خطوط از سمت چپ به عنوان نقطه شروع به سمت راست به عنوان نقطه پایان تکمیل می شوند. در صورتی که از سمت راست شروع به تکمیل کرده و به نقطه سمت چپ برسیم روابط فوق به صورت زیر تغییر می کنند: $(m < 1)$

$$x_{k+1} = x_k - 1$$
$$y_{k+1} = y_k - m$$

و یا برای شیب های بزرگتر از یک خواهیم داشت: $(m > 1)$

$$x_{k+1} = x_k - \frac{1}{m}$$
$$y_{k+1} = y_k - 1$$

روابطی که در الگوریتم *DDA* تشریح شدند، می توانند برای محاسبه موقعیت پیکسل ها با شیب منفی نیز مورد استفاده قرار گیرند.

در ادامه برنامه‌ای به نام *LineDDA* به زبان *C++* مشاهده می کنید که برای رسم یک خط در صفحه نمایش بکار می رود.

در ابتدا مختصات دو نقطه به عنوان نقاط شروع و پایان را دریافت کرده، سپس اختلاف بین نقاط شروع و پایان در راستای افقی و عمودی در متغیرهای dx , dy قرار می گیرد.

در صورتی که مقدار dx > dy، آنگاه *M<1* می باشد، مطابق روابط فوق، گام افزایش x در هر مرحله یک واحد بوده و گام افزایش y، برابر با شیب خط (M) می باشد. و در صورتی که dx < dy، آنگاه *M>1* می باشد، بنابراین گام افزایش x در هر مرحله برابر با معکوس شیب خط $(\frac{1}{M})$ بوده و گام افزایش y، یک واحد می باشد.

شرط دوم جهت بررسی رسم خط از سمت راست یا چپ می باشد. بنابراین الگوریتم زیر بدون هیچگونه تغییری می تواند جهت رسم هر دو نوع خط مورد استفاده قرار گیرد.

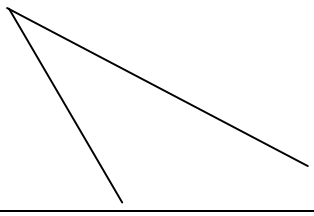
شروع از پیکسل (x_a, y_a) خواهد بود. در هر مرحله تفاوت مکان موقعیت پیکسل را در راستای مسیر خط مشخص می کنیم. این روند در یک حلقه تکرار می شود.

ترسیم خط توسط الگوریتم *DDA*

از این الگوریتم جهت رسم خط با شیب کمتر و بیشتر از یک (M<1 و M>1)، و همچنین برای رسم خطوطی که نقطه آغاز، از نقطه پایان بزرگتر است، استفاده می شود.

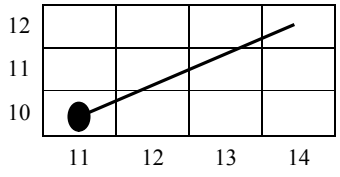
```
/*********************************/
/* Author: Mohsen Shakiba
/*********************************/
#define ROUND(a)  ((int)(a+0.5))
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>
void main (void)
{
int xa,ya,xb,yb,dx,dy,Xend;
float xinc,yinc,x,y;
int driver,mode;
driver=DETECT;
mode=0;
initgraph(&driver,&mode,"");
cout<<"******** PLOT DDA ********\n\n";
cout<<"Please Enter FOUR NUMBER
(Xa,Ya,Xb,Yb):\n";
cin>>xa>>ya>>xb>>yb;
dx=abs(xb-xa);
dy=abs(yb-ya);
if (dx>dy)
  {xinc=1;
  yinc=dy/(float)dx;}              //=M
else
  {xinc=dx/(float)dy;              //=1/M
  yinc=1;}
if(xa<xb)
  {x=xa; y=ya; Xend=xb;}
else
  {x=xb; y=yb; Xend=xa;}
putpixel(ROUND(x),ROUND(y),RED);
//////////////////////Loop//////////////////////////
while(x<=Xend)
  {x=x+xinc;
  y=y+yinc;
  putpixel(ROUND(x),ROUND(y),WHITE);}
cout<<"Xa="<<xa<<" Ya="<<ya<<"
Xb="<<xb<<" Yb="<<yb<<" DX="<<dx<<"
DY="<<dy<<" Xinc="<<xinc<<" Yinc="<<yinc;
getch();
restorecrtmode();
}
```

```
********************************* PLOT DDA *********************************
Please Enter FOUR Number (Xa,Ya,Xb,Yb):
200
200
400
300
Xa=200 Ya=200 Xb=400 Yb=300 Dx=200 Dy=100 Xinc=1.000000 Yinc=0.5000000
Please Enter FOUR Number (Xa,Ya,Xb,Yb):
200
200
300
400
```

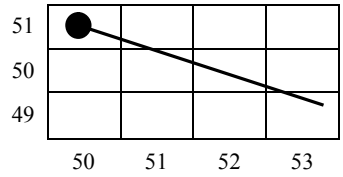


۲.۲.۲.۲. الگوریتم برسنهام[1]

یکی از الگوریتم های دقیق و موثر برای ترسیم خط توسط برسنهام تکوین یافت. در این الگوریتم تنها از مقادیر صحیح افزایشی برای تبدیل خطوط استفاده می شود. این الگوریتم می تواند برای ترسیم دایره و سایر منحنی ها نیز با کمی تغییر مورد استفاده قرار گیرد. در شکل ۲-۶ و ۲-۷ بخشی از خطی که به صورت مستقیم می بایست ترسیم گردد، نشان داده شده است.



شکل ۲-۶- بخشی از یک صفحه نمایش که خطی با شیب مثبت از آن گذشته است. نقطه شروع پیکسلی در سطر پویش ۱۰ و ستون ۱۱ است.



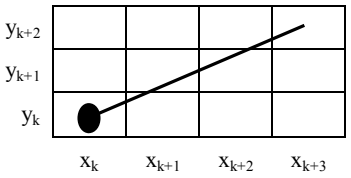
شکل ۲-۷- بخشی از یک صفحه نمایش که خطی با شیب منفی از آن گذشته است. نقطه شروع پیکسلی در سطر پویش ۵۱ و ستون ۵۰ است.

به شکل ۲-۶ توجه نمایید. با فرض این که از نقطه (۱۰ و ۱۱) شروع کرده‌ایم. مایلیم بدانیم در گام بعدی، کدامیک از پیکسل های (۱۰ و ۱۲) و یا (۱۱ و ۱۲) در راستای خط روشن خواهد شد به همین ترتیب در خط با شیب منفی، شکل ۲-۷ مایلیم بدانیم کدامیک از پیکسل های (۵۰ و ۵۱) و یا (۵۱ و ۵۱) در مسیر خط روشن خواهد شد. اینها سوالاتی هستند که توسط الگوریتم برسنهام پاسخ داده شده‌اند. در این الگوریتم علامت یک پارامتر صحیح بررسی می شود. مقدار این پارامتر نسبت اختلاف بین موقعیت‌های دو پیکسل از مسیر واقعی خط است.

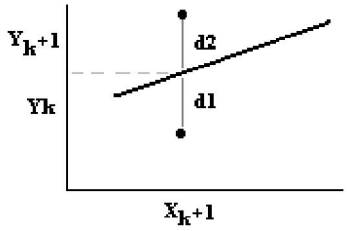
به منظور شرح روش برسنهام، ابتدا حالت خاصی که در آن شیب خط مثبت و کمتر از یک است را در نظر بگیرید. موقعیت پیکسل ها در راستای خط با افزایش گام ها در راستای x تعیین می گردد. نقطه‌ی شروع را سمت چپ و (x_0 , y_0) در نظر گرفته و با پیشروی به اندازه یک واحد در راستای محور x شماره سطر متناسب با آن ستون را بسته به میزان نزدیکی به مسیر واقعی خط تعیین می کنیم. شکل ۲-۸ مرحله k ام این فرآیند را نشان میدهد. فرض کنید می دانیم پیکسلی در موقعیت (x_k , y_k) روشن است و می خواهیم در قدم بعد، تصمیم بگیریم که کدامیک از پیکسل‌ها در ستون x_{k+1} باید روشن شود. انتخاب‌هایی که ما خواهیم داشت یکی از پیکسل‌های $(x_k + 1 , y_k + 1)$ و یا $(x_k + 1 , y_k)$ خواهد بود.

[1] . *Bresenham*

در موقعیت موردی $x_k + 1$ ، فاصله عمودی پیکسل‌ها از مسیر هندسی خط را با d_1 , d_2 نشان می دهیم (شکل ۲-۹). مقدار y در مسیر خط در راستای ستون پیکسل $x_k + 1$ به صورت زیر است :



شکل ۲-۸- بخشی از خط که از نقطه (x_k, y_k) می گذرد ($0<m<1$)



شکل ۲-۹- اختلاف بین موقعیت پیکسل x_k+1 و خط در راستای ستون

از طرفی: $$d_1 = y - y_k = m(x_k + 1) + b - y_k$$

و داریم: $$d_2 = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$$

اختلاف بین این دو مقدار برابر خواهد بود با:

با اعمال پاره ای تغییرات مختصر کاری می کنیم که رابطه فوق، تنها مشتمل بر مقادیر صحیح گردد. این مقدار را پارامتر تصمیم p_k (یعنی تصمیم در مرحله k- ام در ترسیم خط) می نامیم. با جایگزین کردن $m = \frac{\Delta y}{\Delta x}$, p_k به صورت زیر تعریف می شود :

علامت p_k همان علامت $d_1 - d_2$ می باشد (در مثال ما $\Delta x > 0$ بود). پارامتر C مقدار ثابتی بوده و برابر مقدار $2\Delta y + \Delta x(2b - 1)$ می باشد. این مقدار از موقعیت پیکسل مستقل بوده و در الگوریتم بازگشتی محاسبات از p_k حذف خواهد شد. در صورتی که $d_1 < d_2$ باشد این بدین مفهوم است که پیکسل y_k به مسیر خط نزدیکتر از y_k+1 است. بدین ترتیب p_k منفی خواهد بود. در این حالت پیکسل پایینی روشن می شود. طبیعتاً در غیر این صورت پیکسل بالایی روشن خواهد شد.

در هر مرحله میزان پیشروی در راستای x , y هماهنگ بوده و به اندازه یک واحد می باشد. از این رو می توان مقادیر پارامترهای تصمیم را به صورت افزایشی صحیح محاسبه نمود. در مرحله $k+1$ ، پارامتر تصمیم براساس رابطه بالا عبارت خواهد بود از :

با تفریق معادله p_k از رابطه فوق خواهیم داشت :

ولی میدانیم $x_{k+1}=x_k+1$ پس خواهیم داشت : $p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k)$

بسته به علامت p_k، مقدار جمله $y_{k+1} - y_k$ برابر یکی از مقادیر صفر یا یک خواهد بود. محاسبه بازگشتی پارامتر تصمیم در موقعیت صحیح x انجام پذیر است. اولین پارامتر (یعنی p_0) با شروع از نقطه (x_0 , y_0) و با تقسیم $m = \frac{\Delta y}{\Delta x}$ قابل محاسبه است.

در زیر الگویتم ترسیم خط برسنهام با شیب مثبت کوچکتر از یک آورده شده است. ثابت های $2\Delta y$ و $2\Delta y - 2\Delta x$ یک بار محاسبه شده و در کلیه مراحل مورد استفاده قرار می گیرد.

الگوریتم ترسیم خط برسنهام برای $m<1$

1. **دو نقطه ابتدا و انتهای پاره خط دریافت شده و نقطه سمت چپ به نام (x_0, y_0) در حافظه ذخیره می شود.**
2. **نقطه (x_0, y_0) به بافر فریم بارگذاری می گردد. این نقطه، اولین نقطه‌ای است که ترسیم می گردد.**
3. **مقادیر ثابت Δx ، Δy ، $2\Delta y$ و $2\Delta y- 2\Delta x$ محاسبه شده و پارامتر تصمیم اولیه با رابطه زیر بدست می آید: $P_0= 2\Delta y- \Delta x$**
4. **به ازای هر x_k در طول خط، با شروع $k=0$، آزمون زیر را پیگیری نمایید:**

اگر $P_k<0$ باشد، نقطه بعدی برابر (x_k+1, y_k) بوده و $P_{k+1}= P_k+2\Delta y$ ،

اگر $P_k>0$ باشد، نقطه بعدی برابر (x_k+1, y_k+1) و $P_{k+1}= P_k+2\Delta y- 2\Delta x$ می باشد.

5. **مرحله ۴ را به اندازه Δx بار تکرار کنید.**

مثالی از ترسیم خط برسنهام

برای شرح بهتر چگونگی ترسیم خط قصد داریم نقاط قرار گیرنده در راستای خطی از نقطه (*20. 10*) الی (*30. 18*) را تعیین کنیم. شیب این خط *0.8* می باشد.

$\Delta y=8 , \Delta x= 10$

اولین پارامتر تصمیم دارای مقدار *6* است. $p_0 = 2\Delta y - \Delta x = 6$

مقادیر اضافه شونده برای محاسبه پارامترهای تصمیم در هر مرحله عبارتند از :

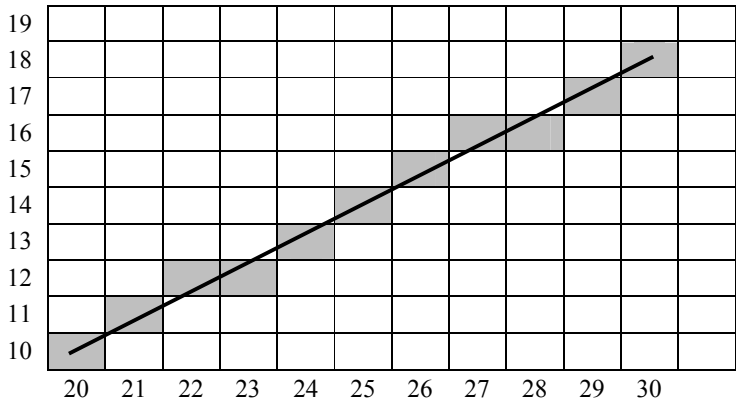
$if\ P_k<0 \ \rightarrow\ P_{k+1}= P_k+2\Delta y= 16$

$if\ P_k>=0 \rightarrow\ P_{k+1}= P_k+2\Delta y- 2\Delta x= -4$

به منظور تعیین سایر نقاط در مسیر خط با شروع از (*20, 10*) = (x_0 , y_0) جدولی به صورت زیر تشکیل داده ایم :

k	P_k	(x_k+1, y_k+1)	k	P_k	(x_k+1, y_k+1)
0	6	(21, 11)	5	6	(26, 15)
1	2	(22, 12)	6	2	(27, 16)
2	-2	(23, 12)	7	-2	(28, 16)
3	14	(24, 13)	8	14	(29, 17)
4	10	(25, 14)	9	10	(30, 18)

نقاطی که در راستای مسیر خط روشن می شوند در شکل ۲-۱۰ نشان داده شده است .



شکل ۲-۱۰- موقعیت پیکسل هایی که در راستای مسیر خط از نقطه (۱۰و۲۰) تا (۱۸و۳۰) بر اساس الگوریتم برسنهام روشن می شود.

در ادامه برنامه‌ای به نام *LineBresenham* به زبان *C++* مشاهده می کنید که برای رسم یک خط با شیبی در محدوده $0 < m < 1$ در صفحه نمایش بکار می رود.

مختصات نقاط شروع و پایان به این برنامه ارسال می شوند و پیکسل ها از نقطه سمت چپ به نقطه پایانی سمت راست ترسیم می شوند (*(x_k+1, y_k) OR (x_k+1, y_k+1)*).

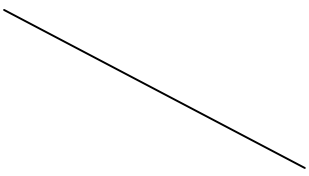
برای ترسیم خطوطی که دارای شیب بزرگتر از یک می باشند، نقش محورهای x, y تعویض می شود. بدین مفهوم که گام ها در راستای محور y به اندازه یک واحد بوده و مقادیر x برای یافتن نزدیکترین پیکسل به مسیر واقعی خط، محاسبه می گردد (*(x_k, y_k+1) OR $(x_k, +1, y_k+1)$*). در صورتی که شیب خط مثبت بوده و نقطه شروع سمت راست باشد، در هر گام از راست به چپ ، هر دو مقدار x, y کاهش می یابند (*(x_k-1, y_k) OR (x_k-1, y_k-1)*). برای مقادیر شیب منفی، همین روال قابل استفاده خواهد بود با این تفاوت که یکی از محورها افزایش و دیگری کاهش می یابد (*(x_k+1, y_k) OR (x_k+1, y_k-1)*). برای حالات خاص خط عمودی کامل $(\Delta x = 0)$، خط افقی کامل $(\Delta y = 0)$ و خط با شیب ۴۵ درجه $(\Delta x = \Delta y)$ نیازی به پردازش جداگانه نبوده و نقاط می توانند مستقیما به بافر فریم بارگذاری گردند.

ترسیم خط توسط الگوریتم *Bresenham*

از این الگوریتم جهت رسم خط با شیب کمتر از یک ($M<1$)، و همچنین برای رسم خطوطی که نقطه آغاز، از نقطه پایان بزرگتر است، استفاده می شود.

```
/*********************************/
/* Author: Mohsen Shakiba
/*********************************/
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<stdlib.h>

void main (void)
{
int xa,ya,xb,yb,dx,dy,p,x,y,Xend;
int driver,mode;
driver=DETECT;
mode=0;
initgraph(&driver,&mode,"");
//**************************************

cout<<"***** PLOT BRESENHAM ******\n\n";
cout<<"Please Enter FOUR Number
(Xa,Ya,Xb,Yb):\n";
cin>>xa>>ya>>xb>>yb;

dx=abs(xb-xa);
dy=abs(yb-ya);
p=2*dy-dx;          //P0 Is Gaam
 if(xa<xb)
    {x=xa;y=ya;Xend=xb;}
  else
    {x=xb;y=yb;Xend=xa;}
putpixel(x,y,15);
cout<<"Xa="<<xa<<" Ya="<<ya<<" Xb="<<xb<<"
Yb=" <<yb<<" Dx="<<dx<<"
Dy="<<dy<<endl<<" P(Gam)="<<p <<"
X0="<<x<<" Y0="<<y<<" Xend="<<Xend;
//////////////////////Loop//////////////////////////
while (x<Xend)
{
   if (p<0)
   { x++;
     p=p+2*dy;
   }
   else
   { x++;
     y++;
     p=p+2*(dy-dx);
   }
   putpixel(x,y,15);
}
getch();
restorecrtmode();
}
```

```
******************************** PLOT BRESENHAM ****************************
Please Enter FOUR Number (Xa,Ya,Xb,Yb):
200
200
400
300
Xa=200 Ya=200 Xb=400 Yb=300 Dx=200 Dy=100 P(Gam)=0 X0=200 Y0=200 Xend=400
```



۳.۲.۲.۲. الگوریتم ترسیم دایره

دایره و یا کـــمانی از آن، از جمله اجـــزائی است که به وفور در تصاویر و ترسـیمات گـرافیکی به چـشم می خورند. از همین رو در اغلب بسته های گرافیکی، روالی برای ترسیم دایره و یا بخشی از آن وجود دارد. اغلب یک روال می تواند هم برای ترسیم دایره و هم برای ترسیم بخشی از آن به صورت کمان و یا حتی منحنی های بیضی شکل مورد استفاده قرار گیرد.

- **ویژگی های دایره**

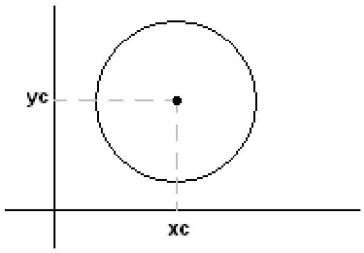
دایره مجموعه ای از نقاط است که فاصله‌ی آنها از مرکز مختصات (x_c , y_c) برابر مقدار ثابت r (شعاع) است. شکل (۲-۱۱). این رابطه در محور مختصات دکارتی به صورت زیر قابل تعریف است :

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

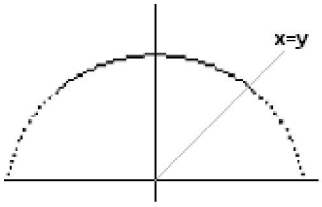
با استفاده از رابطه فوق می توان مختصات نقاطی را که در راستای محیط دایره هستند بدست آورد. بدین ترتیب که با گام هایی در راستای محور x به اندازه یک واحد از نقطه x_c-r تا x_c+r مقادیر y متناسب با هر x را از طریق رابطه زیر به دست آورد.

$$y = y_c \pm \sqrt{r^2 - (x_c - x)^2}$$

اما این روش، شیوه خوبی برای ترسیم دایره نیست. مشکل اول، محاسبات زیاد برای یافتن مختصات هر نقطه در مسیر دایره است. از طرفی، فضای بین موقعیت پیکسل های ترسیمی بصورت یکنواخت توزیع نمی گردد. (شکل ۲-۱۲ را مشاهده کنید).



شکل ۲-۱۱- دایره‌ای به مرکز (x_c, y_c) و شعاع r



شکل ۲-۱۲- عدم یکنواختی فواصل در نیمی از دایره به مرکز (0, 0)

با جابجا کردن نقش x , y با یکدیگر (یعنی پیشروی در راستای y برای یافتن مقادیر متناظر x) در نقاطی که شیب دایره بزرگتر از یک می گردد، می توان این فضای نامتراکم را تکمیل نمود ولی این کار موجب افزایش بار محاسباتی روال ترسیم می گردد.

راه دیگر برطرف نمودن نقص مربوط به عدم تناسب فواصل در شکل ۲-۱۲، محاسبه موقعیت نقاط مرزی با استفاده از مختصات قطبی براساس r و θ می باشد. نقاط روی دایره براساس روابط قطبی بصورت زیر تعیین می گردند.

$$x = x_c + r\,Cos\theta$$
$$y = y_c + r\,Sin\theta$$

در صورتی که اندازه گام ها به صورت ثابت، زاویه مشخصی تعیین گردد، نقاط مرزی دایره از یکنواختی لازم برخوردار خواهند بود. انتخاب اندازه قدم ها برای زاویه θ، بستگی به کاربرد و رسانه نمایشی دارد. فواصل بزرگتر موجب گسستگی می گردد که این گسستگی می تواند با خطوط مستقیم پر شود. در صورتی که بخواهیم در یک نمایش پویشی (Raster) محدوده پیوسته‌تری داشته باشیم، می توانیم اندازه گام ها را $\frac{1}{r}$ انتخاب کنیم. بدین ترتیب به ازاء هر نقطه یک پیکسل وجود خواهد داشت.

با نگاهی به ساختمان ظاهری دایره می توان میزان عملیات محاسباتی را کاهش داد. شکل دایره در هر یک از چهار ربع آن یکسان است. با علم بر این نکته می توان با ترسیم نقاط مرزی در یک ربع آن، با قرینه یابی نسبت به محور x ها و y ها و مرکز مختصات، سایر نقاط واقع بر روی مسیر دایره را به دست آورد. (شکل ۲-۱۴). حتی با نگاه دقیق تر به شکل می توان قرینه بودن دایره نسبت به خط x = y را نیز مشاهده نمود. بدین ترتیب با محاسبه نقاط واقع در $\frac{1}{8}$ محیط دایره، می توان سایر نقاط را با یک عمل قرینه یابی ریاضی ساده به دست آورد.

الگوریتم برسنهام با کمی تغییر برای یافتن نقاط واقع در مسیر دایره قابل پیاده سازی است. بدین مفهوم که با ساختن پارامترهای تصمیم می توان نزدیکترین پیکسل ها در مسیر دایره را در هر گام تعیین نمود. معادله دکارتی دایره غیر خطی بود و برای یافتن فاصله پیکسل ها از مسیر دایره نیاز به یک محاسبه جذرگیری دارد. الگوریتم برسنهام دایره برای اجتناب از محاسبه ریشه دوم، فاکتور تصمیم تعریف می کند.

روش کار را میتوان به صورت خلاصه بدین ترتیب شرح داد که فاصله مستقیم نقطه میانی بین دو پیکسل از مسیر دایره مورد بررسی قرار می گیرد تا تعیین شود کدام پیکسل روشن می گردد. این روش برای ترسیم دوایری با شعاع صحیح بسیار ساده است. شرح جزئیات محاسبات را در ادامه می خوانید.

۱.۳.۲.۲.۲. الگوریتم ترسیم دایره به روش ، نقطه میانی (*Midpoint*)

روش کار بسیار شبیه ترسیم خط در صفحه مشبک پویش است. بدین ترتیب که پیشروی با گام هایی به اندازه واحد پیکسل تعیین می شود که موقعیت کدام پیکسل به مسیر واقعی گذر دایره نزدیکتر است. فرض کنید با شعاع معـــلوم r و به مرکز (x_c , y_c)، قرار است در یک صفحه پویشی با بهترین تقریب، پیکسل های واقع در مسیر تعیین گردند. به منظور سهولت محاسبات ریاضی، دایره مزبور درمرکز (0, 0) ترسیم می گردد و سپس تفاوت مکان x_c , y_c را به منظور انتقال دایره به موقعیت اصلی اعمال می کنیم. بخشی از این دایره واقع بین محور عمودی $x = 0$ الی خط $x = y$ ترسیم می گردد (یک هشتم کل دایره). این خطوط دارای شیب صفر و ۱- هستند. بعد از تعیین نقاط در این بخش، با قرینه یابی، دایره بصورت کامل ترسیم می گردد. مانند قبل از سمت چپ به راست و با گام های واحد، نزدیکترین پیکسل به مسیر دایره تعیین می گردد.

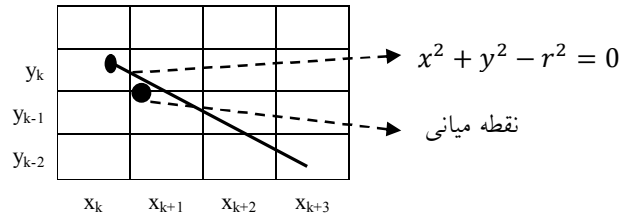
برای بکار گیری روش نقطه میانی، تابع دایره به صورت زیر تعریف می شود:

$$F_{circle}(x,y) = x^2 + y^2 - r^2$$

می دانیم که هر نقطه ای مانند (x, y) به شرطی بر روی مرز دایره‌ای به شعاع r قرار دارد، که در رابطه فوق صدق کند یعنی $F_{circle}(x,y) = 0$.حال اگر نقطه‌ای در یکی از نقاط درونی دایره باشد، مقدار تابع دایره منفی می گردد. مختصات نقاط خارج از مرز دایره نیز موجب مثبت شدن رابطه تابع فوق می گردد. از همین رو خواهیم داشت:

$$F_{circle}(x,y) \begin{cases} < 0 & \text{برای نقاط درون دایره} \\ = 0 & \text{برای نقاط واقع بر روی مرز دایره} \\ > 0 & \text{برای نقاط خارج دایره} \end{cases}$$

تست فوق می تواند در هر گام برای تعیین این که در نقطه میانی پیکسل خارج و یا داخل دایره است، مورد استفاده قرار گیرد. از همین رو تابع دایره در الگوریتم نقطه میانی، پارامتر تصمیم می باشد.



شکل

در شکل ۲-۱۳ نقطه میانی بین دو پیکسل در موقعیت موردی x_k+1 نشان داده شده است. فرض کنید پیکسل (x_k, y_k) روشن شده و در گام بعدی می خواهیم تعیین کنیم که کدامیک از پیکسل های (x_{k+1}, y_k) و یا (x_{k+1}, y_{k-1}) به مسیر دایره نزدیکتر است. پارامتر تصمیم براساس معادله دایره در نقطه میانی بین این دو پیکسل به صورت زیر می باشد:

$$P_k = F_{Circle}\left(x_k + 1, y_k - \frac{1}{2}\right) = (x_k + 1)^2 + (y_k - \frac{1}{2})^2 - r^2$$

$(y_k, y_{k-1}) \Longrightarrow y_k - \frac{1}{2}$ نقطه میانی

در صورتی که $P_k<0$ باشد، این نقطه میانی درون دایره بوده و پیکسل موجود در خط پویش y_k به خط مرزی دایره نزدیکتر می باشد. در غیر این صورت نقطه میانی در خارج دایره است. به عبارت دیگر، باید پیکسل موجود در خط پویش y_k-1 انتخاب شود.

می بینید که پارامتر تصمیم با به توان رساندن و جمع و تفریق قابل محاسبه است. حال سعی خواهیم داشت پارامتر تصمیم مرحله بعد را در موقعیت $x_{k+1} + 1 = x_k + 2$ به صورت بازگشتی بدست آوریم. از همین رو:

$$P_{k+1} = f_{circle}\left(x_{k+1} + 1, y_{k+1} - \frac{1}{2}\right) = [(x_k + 1) + 1]^2 + (y_{k+1} - \frac{1}{2})^2 - r^2$$

$$P_{k+1} = P_k + 2(x_k + 1) + (y_{k+1}^2 - y_k^2) - (y_{k+1} - y_k) + 1$$

می دانیم مقدار y_{k+1} بسته به علامت P_k برابر یکی از مقادیر y_k یا y_{k-1} خواهد بود. از این رو برای به دست آوردن P_{k+1} مقدار اضافه شونده با منفی بودن P_k برابر $2X_{k+1}+1$ و یا $2X_{k+1}+1-2Y_{k+1}$ خواهد بود. معادل عبارت $2X_{k+1}$ و $2Y_{k+1}$ را می توان به صورت زیر نوشت:

$$2x_{k+1} = 2x_k + 2$$
$$2y_{k+1} = 2y_k - 2$$

با شروع از نقطه $(0, r)$، دو مقدار فوق به ترتیب دارای مقادیر صفر و $2r$ خواهند بود. مقادیر متوالی می تواند با افزودن عدد ۲ به مقدار قبلی $2x$ و کاهش ۲ از مقدار قبلی $2y$ بدست آید. مقدار اولیه پارامتر تصمیم می تواند با ارزیابی تابع دایره در موقعیت شروع بدست آید که برابر است با: $(x_0, y_0) = (0, r)$

$$P_0 = F_{Circle}\left(1, r - \frac{1}{2}\right) = 1 + (r - \frac{1}{2})^2 - r^2 = \frac{5}{4} - r$$

در صورتی که مقدار شعاع، عدد صحیحی باشد با رند کردن P_0 به سادگی می توان از تقریب زیر استفاده کرد:

$P_0 = 1 - r$ (برای مقادیر صحیح r)

روش نقطه میانی برای محاسبه موقعیت پیکسل در راستای مرز دایره، همانند الگوریتم ترسیم خط برسنهام، فرض می کند که پارامترهای دایره در یک صفحه با مختصات صحیح تعیین می گردند. مراحل ترسیم دایره براساس الگوریتم نقطه میانی به صورت زیر است:

الگوریتم ترسیم دایره به روش نقطه میانی

۱. **مختصات مرکز دایره (x_c,y_c) و شعاع r را دریافت کن. دایره‌ای به شعاع فوق در مرکز محور مختصات دارای نقطه شروع زیر خواهد بود:** $(x_0,y_0) = (0, r)$

۲. **مقدار اولیه پارامتر تصمیم را از رابطه زیر بدست آورید:** $P_0= 1-r$

۳. **به ازای هر موقعیت x_k، با شروع از k=0، آزمون زیر را انجام دهید:**

اگر $P_k<0$ باشد، نقطه بعدی (x_{k+1}, y_k) خواهد بود و $P_{k+1}= P_k+ 2x_{k+1}+1$ می باشد، و

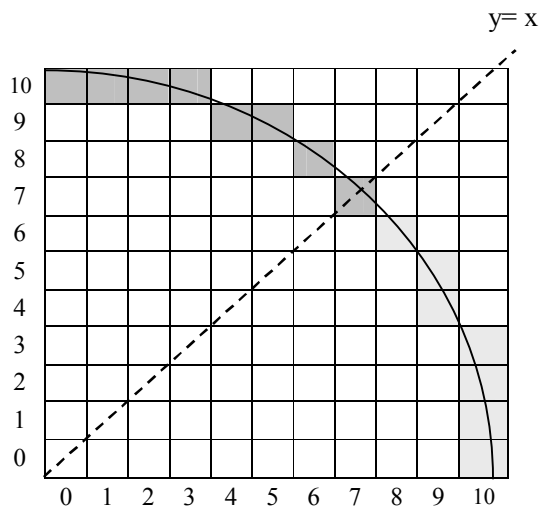
اگر $P_k>=0$ باشد، نقطه بعدی در راستای دایره (x_{k+1}, y_{k-1}) بوده و $P_{k+1}= P_k+ 2x_{k+1}+1-2y_{k+1}$ و می دانیم $2x_{k+1}= 2x_k+ 2$ و $2y_{k+1}= 2y_k- 2$.

۴. **موقعیت سایر نقاط در هفت ناحیه بعدی را نیز تعیین کنید.**

۵. **هر نقطه بدست آمده را با قرار دادن در رابطه زیر، به موقعیت اصلی دایره با مرکز (x_c, y_c) انتقال دهید:** $x=x+x_c$ $y=y+y_c$

۶. **مراحل ۳ الی ۵ را تا وقتی $x>=y$ برقرار است، تکرار کنید.**

شکل ۲-۱۴ نمونه ای از ترسیم را نشان می دهد. حال اجازه دهید مراحل نقطه یابی این ترسیم را به صورت گام به گام با یکدیگر دنبال کنیم.



شکل ۲-۱۴- موقعیت پیکسل‌های ترسیم شده در مسیر دایره‌ای با شعاع r= 10 با استفاده از الگوریتم نقطه یابی

شعاع دایره r= 10 اختیار شده است و قرار است مسیر دایره از محور x= 0 الی خط x= y تعیین گردد. اولین پارامتر تصمیم به صورت زیر بدست می آید:

$P_0 = 1-r = -9$

در صورتی که مرکز دایره را مبداء مختصات فرض کنیم نقطه شروع $(x_0 , y_0) = (0, 10)$ خواهد بود. جملات افزایشی اولیه برای محاسبه پارامترهای تصمیم عبارتند از:

$2x_0 = 0 \ , \ 2y_0 = 20$

پارامتر های تصمیم متوالی و موقعیت پیکسل ها در راستای مسیر دایره به صورت زیر خواهد بود:

K	P_k	(x_{k+1}, y_{k+1})	$2x_{k+1}$	$2y_{k+1}$
0	-9	(1,10)	2	20
1	-6	(2, 10)	4	20
2	-1	(3, 10)	6	20
3	6	(4, 9)	8	18
4	-3	(5, 9)	10	18
5	8	(6, 8)	12	16
6	5	(7,7)	14	14

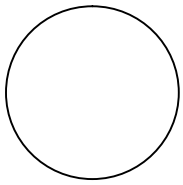
در ادامه برنامه‌ای به نام *Circle* به زبان *C++* مشاهده می کنید که با الگوریتم نقطه میانی در یک صفحه نمایش پویشی، دایره ای را ترسیم می کند.

ترسیم دایره به روش نقطه میانی

```
/*********************************/
/* Author: Mohsen Shakiba
/*********************************/
#include<iostream.h>
#include<conio.h>
#include<graphics.h>

void  main (void)
{
int xcenter,ycenter,radius,x,y,p;
int driver,mode;
driver=DETECT;
mode=0;
initgraph(&driver,&mode,"");
//****************************************
******

cout<<"************ PLOT CIRCLE
*************\n\n";
cout<<"Please Enter Three Number: (Xcenter &
Ycenter & Radius):\n";
cin>>xcenter>>ycenter>>radius;
x=0; y=radius; p=1-radius;
cout<<"Xcenter="<<xcenter<<"Ycenter="<<ycenter<
<" Radius="<<radius<<"X="<<x<<"Y="<<y<<"
P="<<p;
```

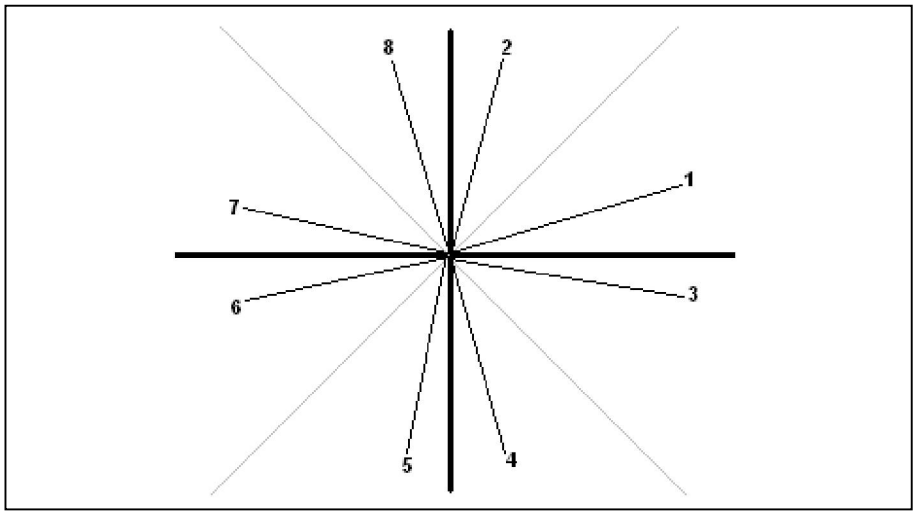
```
while(x<=y)
{
putpixel(xcenter+x,ycenter+y,15);
putpixel(xcenter-x,ycenter+y,15);
putpixel(xcenter+x,ycenter-y,15);
putpixel(xcenter-x,ycenter-y,15);

putpixel(xcenter+y,ycenter+x,1);
putpixel(xcenter-y,ycenter+x,1);
putpixel(xcenter+y,ycenter-x,1);
putpixel(xcenter-y,ycenter-x,1);

          if (p<0)
           {x++;
           p=p+(2*x+1);}
          else
           {x++;
           y--;
           p=p+(2*(x-y)+1);
            }
}
getch();
restorecrtmode();
}
```

```
********************************** PLOT CIRCLE
**********************************
Please Enter Three Number: (Xcenter & Ycenter & Radius):
200
200
50
Xcenter=200 Ycenter=200 Radius=50  X=0 Y=50 P=-49
```

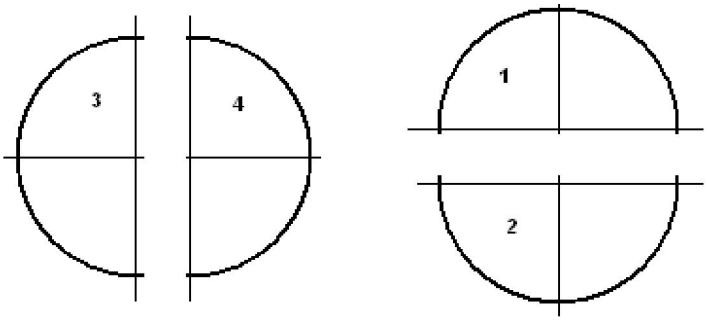


تمرین

۱. با استفاده از الگوریتم *DDA* و *Bresenham* خطوط شماره ۲ تا ۸ را رسم کنید.



۲. با استفاده از الگوریتم *Cirlce* نیم دایره‌های زیر را رسم کنید.



محسن شکیبا فخر

مراجع

محمدرضا حیدری نژاد- گرافیک کامپیوتری

عین الله جعفر نژاد قمی- برنامه نویسی به زبان C (کتاب جامع توربو C و C++)