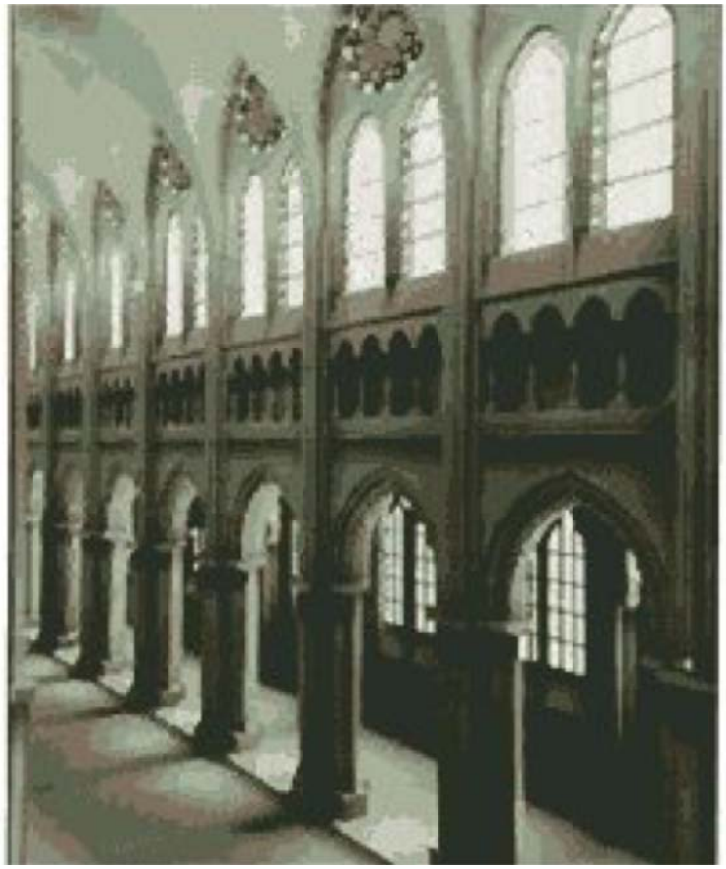
گرافیک کامپیوتری ۱

Computer Graphic ۱



فهرست مطالب

فصل اول

بررسی اجمالی گرافیک کامپیوتری

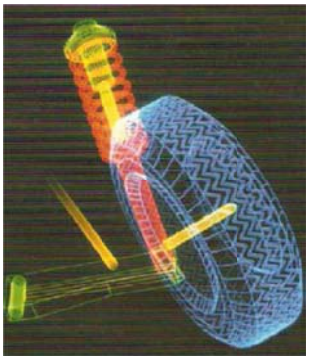
A Survey Of Computer Graphics

کاربردهای گرافیک های کامپیوتری

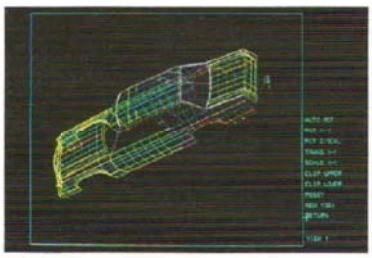
۱.۱ **طراحی به کمک کامپیوتر(Computer-Aided Design)**؛ مهمترین استفاده از گرافیک کامپیوتری در طراحی یا CAD ، به خصوص در رشته های مهندسی و سیستم های معماری می باشد . اما امروزه تقریبا تمامی محصولات بوسیله کامپیوتر طراحی می شود. به طور کلی ، CAD به عنوان یک مرجع مهم برای طراحی به حساب می آید ، که به طور معمول در طراحی ساختمان ها، اتومبیل، هواپیما، زیر دریایی، فضاپیما ،کامپیوتر، پارچه، و بسیاری از محصولات دیگر مورد استفاده قرار می گیرد.

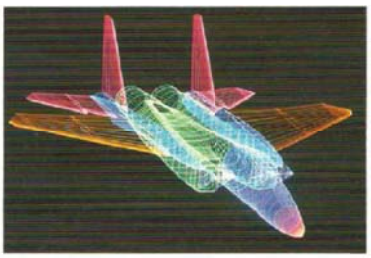
خصوصیات ابزارهای گرافیکی عبارتند از:

۱- مدل سیمی[۱]: یعنی علاوه بر اینکه ما می توانیم یک شی را رسم کنیم بتوانیم اجزای درونی آنرا نیز ببینیم. این خصوصیت به طراح کمک می کند تا بلافاصله نتیجه تغییر برخی از پارامتر ها را برای تمامی اجزای شی مشاهده نماید .در شکل زیر یک مدل سیمی رنگی از یک چرخ اتومبیل دیده می شود.



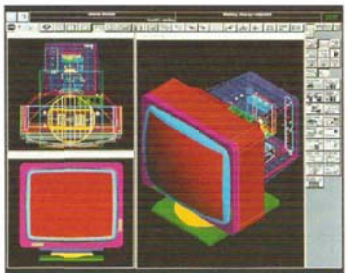
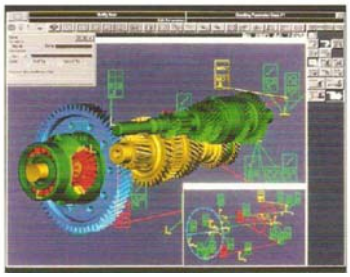
همچنین در شکلهای فوق نمونه های دیگری از مدل های سیمی طراحی شده توسط کامپیوتر برای یک هواپیما و اتومبیل را مشاهده می کنید.



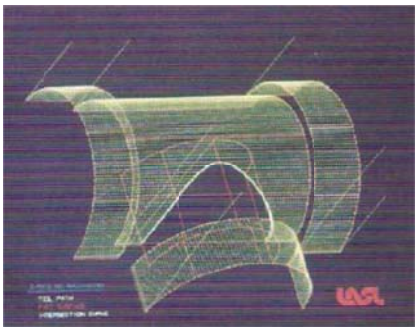


۲- چند دید[۲]؛ برخی از نرم افزارهای طراحی این امکان را به طراح می دهد تا علاوه بر رسم شی آنرا از زوایای مختلف ببیند. در شکلهای زیر نمونه های از طراحی انجام شده توسط کامپیوتر برای دو شی کامپیوتر و چرخ دنده ارائه شده است که امکان مشاهده اشیاء را از زوایای مختلف توسط کامپیوتر فراهم شده است.

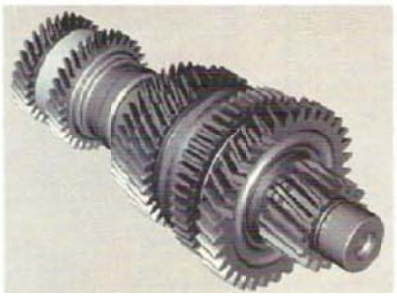
۱ : Wire Frame
۲ : Multi Window

۳- لایه بندی[۱]: امکانی است که برخی از نرم افزارهای طراحی در اختیار طراحان قرار می دهد تا یک شی را به کمک لایه ها ایجاد نماید . این امکان موجب می شود تا یک شی بزرگ و پیچیده به لایه های بسیار کوچکتر و ساده تر شکسته شود . که طراح با طراحی هر کدام از این لایه ها و در کنار هم قرار دادن آنها شی اصلی را طراحی کند . که یکی از نرم افزارهای کاربردی معروف در این زمینه نرم افزار فتوشاپ می باشد. در شکل زیر نمونه ای از لایه های طراحی شده برای یک شی سیلندررا مشاهده می کنید.



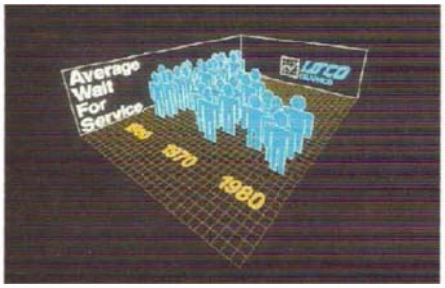
۴- نمایش دادن ونورپردازی[۲]: بعد از اینکه یک شی به کمک یک نرم افزار طراحی می شود. می بایست این شی واقعی به نظر برسد برای اینکار باید سطوح شی مورد نظر نورپردازی و سایه پردازی شود. با استفاده از این خصوصیت جنبه های واقعی شی تقویت می شود تا شی نمود واقعیت پیدا کند بنابراین هرچقدر که ابزار طراحی قوی تر باشد طراح می تواند یک شی نزدیکتر به واقعیت را طراحی نماید. در شکل زیر نمونه های از این خصوصیت را مشاهده می کنید.

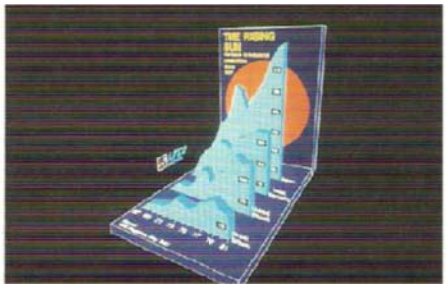



۱: Layout
۲: Lighting And Rendering

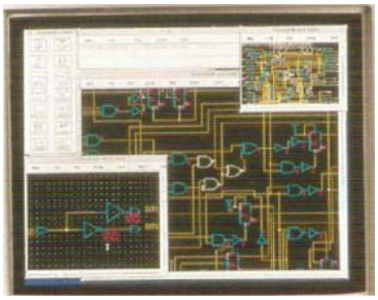
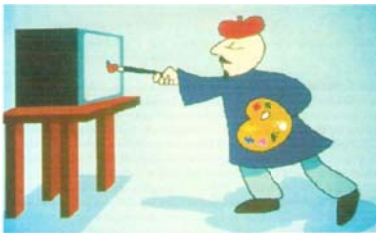

۵- گرافیک تعاملی[۱] ؛ با استفاده از این خصوصیت طراح یک ارتباط کاملا دو طرفه و تعاملی با نرم افزار طراحی دارد و قادر خواهد بود بلافاصله نتیجه اعمال خود را بر روی شی مورد طراحی ببیند.

۱.۲ **ارائه گرافیکی(Presentation Graphics)**؛ یکی دیگر از کاربردهای اصلی گرافیک کامپیوتری استفاده از آن در ارائه می باشد. نرم افزارهای مورد استفاده در این زمینه به ما کمک می کنند تا یکسری اسلاید را در زمینه های مختلف را آماده کنیم تا در زمان ارائه توسط یک پروژکتور نمایش دهیم.ارائه گرافیکی اغلب برای ارائه در زمینه های خلاصه وضعیت های مالی ،آماری ، ریاضیاتی، داده های اقتصادی، گزارشات تحقیقی ، گزارشات مدیریرتی ،بولتن های مورد استفاده مشتریان و ... می باشد. ابزارهای ارائه گرافیکی همچنین دارای امکان تبدیل داده به یکسری از نموداری های ، میله ای ، خطی ، دایره ای و ... به صورت دو بعدی و ۳ بعدی می باشند.همچنین برای جذاب تر کردن ارائه می توانند افکت های متنوعی را بر روی یک اسلاید قرار دهند. در اشکال زیر نمونه های از نمودار های طراحی شده را مشاهده می کنید.



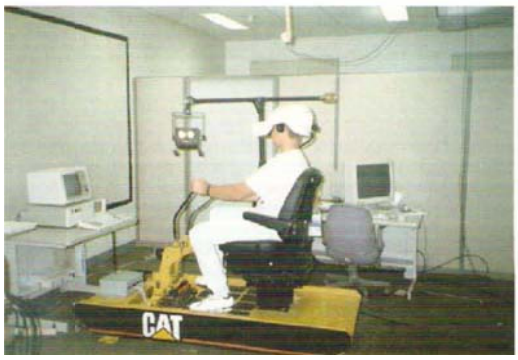


۱.۳ **در هنر نقاشی** ؛ رسم توابع ریاضی، اشکال الکترونیکی ، سرگرمی، تبدیل تدریجی یک شکل به شکل دیگر[۲]

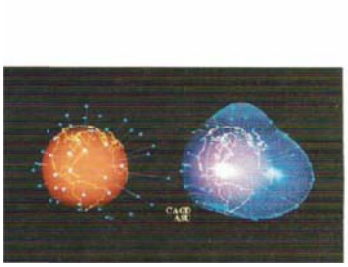



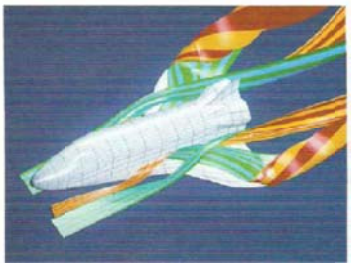
۱ : Interactive Graphic
۲: Morphing

۱.٤ **آموزش** ؛ دیگر کاربرد مهم ابزارهای گرافیکی در آموزش می باشد. برخی از نرم افزارهای طراحی محیطی را فراهم می کنند تا با استفاده از آن فراگیران فرآیند یادگیری را توسط یک محیط مشابه در دنیای واقعی دنبال کنند . به عنوان مثال آموزش رانندگی ، خلبانی ، قرارگیری در جو و ... در شکل زیر نمونه های از آنها را مشاهده می کنید.



۱.٥ **مصور سازی داده** [۱]؛ تبدیل اطلاعات خام به یک تصویر که در شکل زیر نمونه های از آن را مشاهد می کنید.





۱.٦ **پردازش تصویر**[۲]: در پردازش تصویر هدف این است تا اطلاعاتی را از تصویر استخراج کنیم. برخی از نرم افزارهای گرافیکی این امکان را به استفاده کنندگان تا یکسری اطلاعاتی که شاید استخراج آن توسط انسان غیر ممکن و یا سخت و زمان گیر می باشد را استخراج نمایند. غالب کاربردهای پردازش تصویر در تشخیص پزشکی ،صنعت، تجارت و ... می باشد.در شکل زیر استخراج اطلاعات یک بارکد را مشاهده می کنید.

۱:Data Visualization
۲ : Image Processing

۱.۷ **رابط گرافیکی کاربر** [۱]؛ یکی دیگر از کاربردهای ابزارهای گرافیکی طراحی رابط گرافیکی کاربر است که در شکل زیر نمونه ای از آن را مشاهد ه می کنید. در بسیاری از ابزارهای مهندسی نرم افزار امکانی جهت طراحی فرمهای ارتباط با کاربر تعبیه شده است که دارای اشیاء مختلفی مانند Image,TextBox, Combobox,Listbox,... می باشد که برنامه نویس یا طراح می تواند با سلیقه خود فرمهای ارتباط با طراح را ایجاد نماید.

۱ : Graphic User Interface

فصل دوم

مروری بر سیستمهای گرافیکی

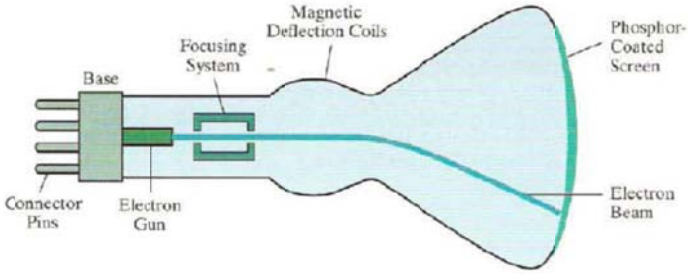
Overview of Graphics Systems

امروزه با توجه به شناخت گسترده ای که از قدرت و کاربردهای گرافیک کامپیوتری در تمام زمینه ها وجود دارد ، طیف وسیعی از سخت افزارها و سیستم های نرم افزاری گرافیکی در حال حاضر در دسترس است. گرافیک کامپیوتری امکان سه بعدی سازی در فضای دو بعدی سازی را در بسیاری از نرم افزارهای کاربردی فراهم کرده است . در حال حاضر از گرافیک کامپیوتری در بسیاری از دستگاههای ورودی تعاملی مانند ماشین حساب ها ، رایانه های شخصی و ... استفاده می شود. همچنین برای کاربردهای پیچیده از سیستم های گرافیکی خاص و پیچیده ای استفاده می شود. در این فصل سعی می کنیم که این سیستم های گرافیکی را بررسی نماییم.

نمایشگرهای ویدوئویی CRT **(**Cathode Ray Tube**):**

یکی از ابتدایی ترین وسایل خروجی در سیستمهای گرافیکی ؛ نمایشگرهای ویدوئویی می باشد. که غالب آنها بر اساس تکنولوژی اشعه کاتدی می باشد. اما تکنولوژیهای دیگری در بازار وجود دارد که در این قسمت سعی می شود. این تکنولوژیها را بررسی و با هم مقایسه کنیم .

یک مانیتور CRT قدیمی از یک لوله شبیه یک بطری شیشه ای بزرگ استفاده می کند. ۳ تفنگ الکترونی برای هر رنگ قرمز،سبز وآبی در سمت باریک آن قرار دارند آنها الکترونها را به سمت صفحه بزرگ مسطحی که در برابر تماشاگر قرار دارد شلیک می کنند. روش تولید رنگ در مانیتورهای CRT به صورت Shadow Mask می باشد. شکل زیر عملیات پایه ای را در یک سیستم نمایشگر CRT نشان می دهد. همانطور که مشخص شده است اشعه کاتدی تابیده شده از تفنگ الکترونی شلیک می شود بر روی یک نقطه فسفری خاص از نمایشگر ویدوئویی با استفاده از سیستم متمرکز کننده [۱]متمرکز می شود. فسفرهای موجود بر روی نمایشگر به اشعه کاتدی حساس می باشند که با این تابش فسفرها برای مدتی کوتاه نورانی می شوند . بنابراین برای اینکه همیشه در یک نقطه خاص یک رنگ خاص به چشم شما برسد می بایست این عمل سریعا تکرار شود. تعداد دفعاتی که این عمل در یک ثانیه انجام می شود را نرخ دوباره سازی [۲] نامیده می شود و بسته به کیفیت فسفرها متفاوت خواهد بود.

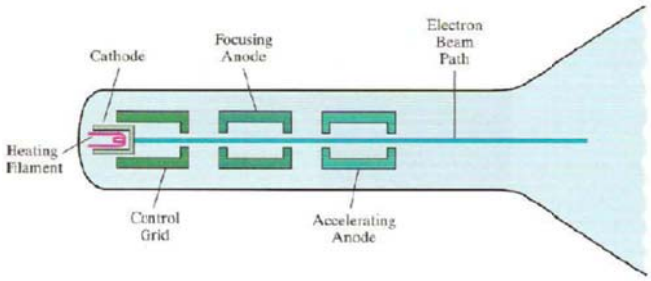


همانطور که در شکل زیر مشخص شده است اجزای اصلی یک تفنگ الکترونی در لامپ تصویر نمایشگر CRT اشعه کاتدی و یک شبکه کنترلی می باشد.که حرارت از طریق یک سیم پیچ به اشعه کاتدی اعمال و در نتیجه

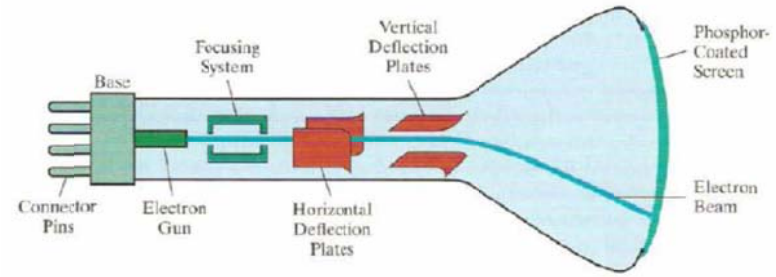
۱: Focusing System

۲ : Refresh Rate

موجب گرم شدن و یونیزشدن مثبت آن می شود. این در حالی است که در داخل لوله لامپ تصویر یونهای منفی به صورت آزادانه وجود دارند. که همین امر موجب ایجاد یک میدان مغناطیسی و هدایت اشعه کاتدی به سمت یک نقطه خاص از نمایشگر که با فسفر پوشانده شده است می شود. همچنین وظیفه شبکه کنترل این است که شدت تابش الکترون ها را برای اشعه الکترونی که از آن عبور میکند را کنترل نماید .



همچنین برای هدایت اشعه کاتدی به سمت یک مختصات خاص مانند شکل زیر دو صفحه مغناطیسی عمودی و دو صفحه مغناطیسی افقی وجود دارد . صفحات مغناطیسی افقی برای هدایت اشعه کاتدی در امتداد افق و یا تعیین ستون و صفحات مغناطیسی عمودی برای تعیین سطر مختصات در نظر گرفته شده اند.



قدرت تفکیک یا وضوح تصویر [1]: به حداکثر تعداد نقاطی که می تواند بدون همپوشانی در یک نمایشگر ویدوئویی CRT نمایش داده شود گفته می شود . همچنین همجواری بیش از حد نقاط به یکدیگر باعث همپوشانی و عدم تشخیص نقاط می گردد. شدت نقاط دارای یک توزیع گوسی است. و نقاط زمانی قابل تشخیص هستند که فاصله مراکز آنها از یکدیگر بیشتر از ۶۰٪ حداکثر قطر تابش باشد.

شکل الف: یک نقطه روشن شده که دارای توزیع گوسی است

شکل ب: دو نقطه روشن شده همجوار

قطر تابش یا قطر هر نقطه (پیکسل) ؛ قطر باریکه الکترونی و سطح انرژی آن گفته می شود . بنابراین، رزولوشن یا قدرت تفکیک به نوع فسفر، شدت تابش و سیستم متمرکز کننده و منحرف کننده بستگی دارد. وضوح تصویر هر نمایشگری از لحاظ سخت افزاری محدود است. بنابراین هرگونه تغییر رزولوشن به صورت نرم افزاری پایینتر از محدوده رزولوشن سخت افزاری می باشد. بنابراین تغییر رزولوشن به صورت نرم افزاری می تواند مثلا پیکسل های سخت افزاری به صورت یک درمیان و دو در میان و ... و یا دوپیکسل به صورت یک پیکسل نشان داده شوند

[1] : Resolution

نرخ نمایش یا نسبت تصویر [1]:

نسبت نقاط عمودی به افقی را می نامند . که برای رسم خطوط عمودی و افقی به آن نیاز است. لازم به ذکر است که نقاط افقی نسبت به عمودی پیوستگی بیشتری دارند ؛ یعنی برای رسم یک خط یک سانتی متری افقی می بایست تعداد پیکسل بیشتری نسبت به رسم یک خط یک سانتی متری عمودی روشن شود.

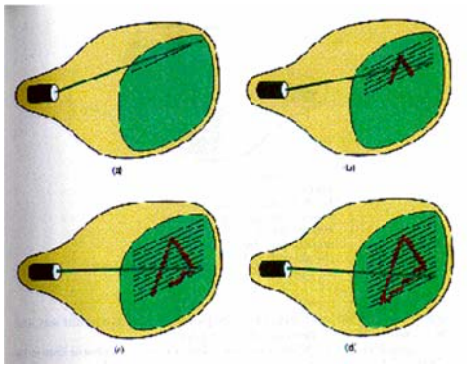
$$\text{نسبت تصویر} = \frac{\text{تعداد نقاط راستای در عمودی}}{\text{تعداد نقاط راستای در افقی}}$$

انواع نمایشگر CRT :

- رستری [2]
- برداری [3]

صفحه نمایش رستری :

مانند تلویزیون:



همانطور که در شکل فوق مشاهده می کنید . در اینگونه نمایشگرها برای اینکه تصویر به صورت ماندگار در ذهن نقش ببندد می بایست هر لحظه تصویر دوباره سازی [4] شود. بنابراین طبق این روش کل صفحه نمایش به صورت پیکسل به پیکسل از سمت چپ بالای تصویر و ردیف به ردیف دوباره سازی کرد که همانگونه که قبلا هم گفته شده است بسته به کیفیت فسفرها این عمل می تواند با تعداد بیشتر- در فواصل زمانی کوتاهتر- و یا کمتر – در فواصل زمانی طولانی تر انجام شود. اما معمولا به طور متوسط ۶۰ الی ۸۰ بار در ثانیه این عمل انجام می شود. که همین امر سبب می شود تا تصویر کاملا در ذهن نقش ببندد.عمل دوباره سازی تصویر دارای چند مولفه می باشد که در ادامه آنها را بررسی خواهیم کرد.

---

[1] : Aspect Ratio
[2] : Raster- Scan Display
[3] : Vector-Scan Display
[4] : Refresh

مولفه های دوباره سازی تصویردر نمایشگرهای رستر:

- تعداد فریم هایی که در یک ثانیه دوباره سازی [۱]می شود. این مولفه مشخص می کند که در یک ثانیه چند قاب صفحه نمایش دوباره سازی می شود.
- تعداد خطوطی که در یک ثانیه دوباره سازی [۲]می شود. این مولفه مشخص می کند که در یک ثانیه چند خط در صفحه نمایش دوباره سازی می شود.

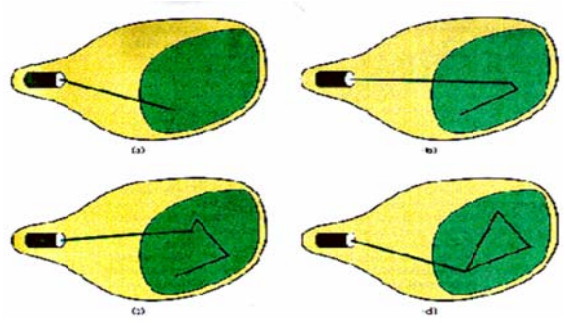
$$Resolution = Row * Col$$
$$Horizontal\ Scan\ Rate = Vertical\ Scan\ Rate * Row$$

- تعداد نقاطی که در یک ثانیه دوباره سازی [۳]می شود. این مولفه مشخص می کند که در یک ثانیه چند پیکسل دوباره سازی می شود.

$$Band\ Width\ Scan\ Rate = Resolution * Vertical\ Scan\ Rate$$

- مدت زمانی که لازم است تا اشعه کاتدی از نقطه پایانی یک خط مجددا به نقطه ابتدایی خط بعد برسد[۴]. این مولفه در واقع فاصله زمانی بین روشن شدن آخرین پیکسل یک خط تا روشن شدن اولین پیکسل خط بعدی می باشد
- مدت زمانی که لازم است تا اشعه کاتدی، از پیکسل پایانی یک فریم مجددا به اولین پیکسل یک فریم برسد[۵]. این مولفه در واقع فاصله زمانی بین روشن شدن آخرین پیکسل یک فریم تا روشن اولین پیکسل فریم بعدی می باشد.

نمایشگرهای برداری [۶]:



عملکرد نمایشگرهای وکتوری شبیه عملکرد وسایلی مانند پلاتر و عملکرد نمایشگرهای رستری مانند پرینتر است. تازه سازی در این نمایشگرها برداری به تعداد خطوطی که روی صفحه نمایش می یابد بستگی دارد.همچنین خود تصویر هم به صورت مجموعه ای خطوط در حافظه نگهداری می شود. بنابراین برا ی تازه سازی تصویر، سیستم هربار باید به صورت

---

[۱] : Vertical Scan Rate
[۲] : Horizontal Scan Rate
[۳] : Bandwidth Scan Rate
[۴] : Horizontal Retrace
[۵] : Vertical Retrace
[۶] : Vector-Scan Display

دوره ای این اطلاعات را از بافر برداشته و آنها را ترسیم کند.و این کار ۳۰ تا ۶۰ بار در ثانیه انجام می شود.به عبارت دیگر برخلاف روش رستر که کل نمایشگر دوباره سازی می شود فقط پیکسل های دارای مقدار معتبر برای تصویر دوباره سازی می شود .

مزایا و معایب صفحه نمایش برداری:

- مناسب برای رسم تصاویر به صورت مجموعه خطوط، مثل تجهیزات آزمایشگاهی مانند اسیلوسکوپ
- وضوح بالاتر نسبت به نمایشگرهای رستری
- تصاویر با تن رنگی پیوسته را به خوبی نمایش نمی دهند.

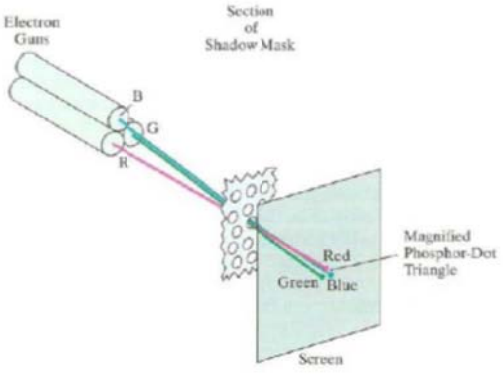
روشهای تولید رنگ در نمایشگر ها :

روش نفوذ اشعه [۱] :

این روش تولید رنگ اغلب در مانیتورهای برداری کاربرد دارد. و سه لایه فسفری قرمز آبی سبز روی صفحه ای که جنس آن آلیاژی از یک فلز خاص می باشد وجود دارند . رنگ نمایش داده شده به فاصله پرتو الکترونی به لایه های فسفری بستگی دارد. در واقع اشعه کاتدی به این سه صفحه تابیده می شود و شدت هر اشعه موجب می شود که از این صفحات عبور بیشتری داشته باشد.در نتیجه عبور اشعه از این صفحات باعث تولید رنگ مورد نظر می شود

روش ماسک سایه [۲] :

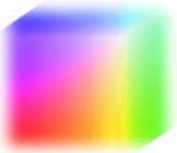
این روش در ابتدای فصل توضیح داده شده است و اغلب در نمایشگرهای رستری کاربرد دارد . لازم به ذکر است که هر پیکسل دارای ۳ زیرپیکسل قرمز آبی سبز است. و در اینگونه نمایشگرها ۳ تفنگ الکترونی مجزا برای نورهای رنگی قرمز سبز آبی تعبیه شده است. که با شدت های متفاوت تابش رنگهای متنوعی ایجاد می شود .



[۱] : Beam Penetration
[۲] : Shadow Mask

اگر برای هر رنگ ۸ بیت در نظر بگیریم:

انواع قرمز: $۲^{۸}$ = ۲۵۶ حالت



انواع سبز: $۲^{۸}$ = ۲۵۶ حالت

انواع آبی: $۲^{۸}$ = ۲۵۶ حالت

بنابراین کل حالتها: ۲۵۶*۲۵۶*۲۵۶ = ۱۶۷۷۷۲۱۶ رنگ خواهد شد.

اگر برای هر رنگ فقط ۱ بیت در نظر بگیریم:

انواع قرمز: $۲^{۱}$ = ۲ حالت (روشن یا خاموش)

انواع سبز: $۲^{۱}$ = ۲ حالت

انواع آبی: $۲^{۱}$ = ۲ حالت

بنابراین کل حالتها: ۲*۲*۲= ۸ رنگ می باشد.

روش های افزایش سرعت دوباره سازی:

۱- افزایش عرض گذرگاه کارت گرافیک که باعث می شود در یک ارجاع تعداد بیشتری از اطلاعات خوانده شود.
۲- افزایش تعداد Frame Buffer های روی کارت گرافیک مثلا اضافه کردن یک فریم بافر دیگر برای خواندن و یا نوشتن ، در زمانی که فقط یک فریم بافر هم برای خواندن و هم برای نوشتن وجود دارد. بنابراین می توان سرعت دوباره سازی را دو برابر کرد.
۳- استفاده از Look up Table : با استفاده از این جدول زمانی که تصویر تغییر می کند به جای ذخیره مجدد تصویر ؛ فقط نقاطی از تصویر را ذخیره می کنیم که تغییر کرده اند . برای پیاده سازی این جدول از حافظه ای بنام Associated Memory می باشدکه فقط شماره سطر و شماره ستون و رنگ تغییر یافته در آن ذخیره می شود.
۴- ترکیب کردن اطلاعات تصویر فعلی با اطلاعات تصویری که جدیدا به وجود می آید .

روشهای افزایش حافظه تصویر (Memory Saving) یا روشهای ذخیره سازی اطلاعات یک تصویر در حافظه:

۱- Run length Encoding: در این روش به جای ذخیره تک تک نقاط تصویر آنرا به صورت خطوطی در نظر می گیریم و فقط نقاط ابتدا و انتهای خطوط را ذخیره می کنیم.
۲- Frame Encoding: در این روش به جای ذخیره تک تک نقاط ، تصویر را به کادرهایی مشخص تقسیم بندی کرده و فقط دو نقطه با مختصات زیر ذخیره می شود:
- گوشه سمت چپ بالای کادر
- گوشه سمت راست پایین کادر.

نمایشگرهای صفحه تخت[1]:

مشخصات و ویژگی های عمده :

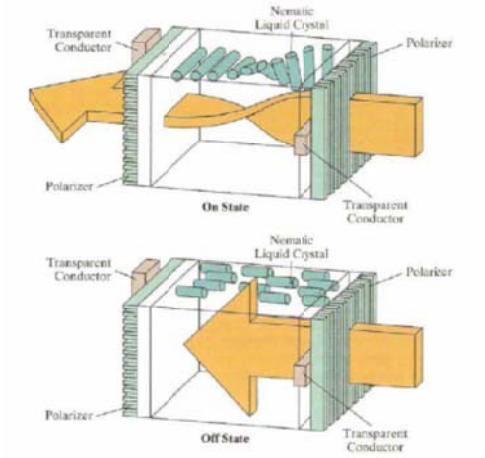
۱- از نمایشگرهای CRT نازک ترند.
۲- حجم ،وزن و توان مصرفی آنها نیز کمتر است
۳- موارد استفاده آنها عبارتند از ,TV,Monitor ، ماشین حساب ، لپ تاپ

انواع نمایشگرهای صفحه تخت:

۱- نمایشگرهای تابنده[2]: انرژی الکتریکی را به نور تبدیل می کند. مانند Plasma Panel , Light Emitting Diodes
۲- نمایشگرهای که در داخل آنها یک منبع نور وجود دارد . و یا از نور خورشید و سایر منابع نور برای تشکیل قالبهای گرافیکی استفاده می کند. مانند Liquid Crystal Display

نمایشگر های LCD:

عملکرد اینگونه مانیتورها در شکل زیر آورده شده است. قابل ملاحظه است که در اینگونه نمایشگرها دو Polarizer وجود دارد که آنها نسبت به هم نود درجه اختلاف دارند . نور ابتدا از Polarizer اول وارد می شود که بسته به شدت یا انرژی نور، الکترود ها نیز می چرخند ؛ این چرخش الکترودها باعث می شود که نور نیز بچرخد و از Polarizer دوم عبور کند. بنابراین هرچقدر شدت نور بیشتر باشد چرخش نور نیز بیشتر شده و در نهایت عبور بیشتری خواهد داشت.



لازم به ذکر است که در مانیتورهای LCD سیاه و سفید به ازای هر پیکسل یک الکترود وجود دارد اما در LCD های رنگی به ازای هر پیکسل ۳ الکترود وجود دارد. همچنین به خاطر اینکه در مانیتور های LCD فسفر وجود ندارد عمل دوباره سازی تصویر صورت نمی گیرد و فقط زمانی دوباره سازی داریم که تصویر تغییر پیدا کند.

[1] : Flat Panel Display
[2] :Emissive Display

مانیتورهای پلاسما:

ساختار اینگونه مانیتورها شبیه مانیتورهای LCD می باشد اما به جای کریستال مایع از مخلوطی از گازها بنام پلاسما استفاده می شود.همچنین به ازای هر پیکسل ۳ فسفر RGB وجود دارد. بسته به تولید ولتاژ و شدت نور ، پلاسما تحریک می شود و باعث می شود تا اشعه ماورای بنفش از خود تولید کند. این اشعه با شدت و انرژی خود به فسفر ها تابیده می شود سپس رنگ تولید شده و به چشم ما می رسد .

مقایسه نمایشگرهای پلاسما با LCD:

- زاویه دید پلاسما از LCD بیشتر است که در این بین بیشترین زاویه دید مربوط به نمایشگرهای CRT می باشد.
- در پلاسما به جای کریستال مایع از سولفید روی به علاوه منگنز استفاده می شود.
- LCD توان مصرفی بیشتری و تولید رنگ دلخواه و مطلوب در LCD سخت تر از پلاسما می باشد.

فصل سوم

ترسیم اشکال پایه ای گرافیکی

Graphics Output primitives

ترسیم نقطه:

کوچکترین مولفه یک تصویر و یا شکل هندسی یک پیکسل می باشد بنابراین برای ترسیم تمامی اشکال هندسی می بایست مجموعه ای از این پیکسل ها در کنار یکدیگر قرار داد تا یک تصویر مشخص به وجود آید. در نمایشگر های مختلف ساختار پیکسل ها متفاوت می باشد. هر پیکسل خود نیز از مولفه های کوچکتری تشکیل شده است که نحوه چیدمان آنها پیکسل های متفاوتی می تواند بسازد. اما به طور معمول ساختار یک پیکسل به شکل یک مستطیل عمودی می باشد. از سایر ساختارهای پیکسل می توان به ساختار های مثلثی ، مربعی و یا شش ضلعی نیز اشاره کرد. که هر کدام از آنها مزایا و معایب خود را دارند.

لازم به ذکر است که در اکثر نرم افزارهایی که برای ترسیم اشکال گرافیکی از دستورات گرافیکی استفاده می کنند. یک دستور به شکل زیر برای ترسیم یک پیکسل وجود دارد:

SetPixel(X,Y,Color)

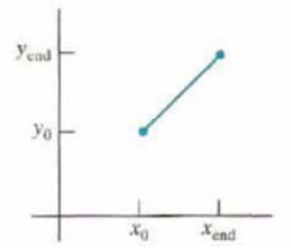
بنابراین برای ترسیم یک پیکسل ما نیاز به یک مختصات یعنی (X,Y) و یک رنگ داریم.

ترسیم خط

الگوریتم های مختلفی برای ترسیم یک خط وجود دارد که همگی آنها تعمیم یافته معادله رسم خط در دستگاه معادلات ریاضی می باشد بنابراین ما در اینجا ابتدا روش رسم خط با استفاده از معادله آنرا تجربه می کنیم و سپس روشهای توسعه یافته دیگر را بررسی خواهیم کرد.

ترسیم خط با استفاده از معادله رسم خط یا روش مستقیم:

ما برای رسم یک خط نیاز به دو خواهیم داشت تا با مجموعه ای از نقاط با شیب خاص از نقطه اول تا نقطه دوم یک خط را بسازیم .



بنابراین معادله رسم خط به صورت زیر می باشد:

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

در این معادله پارامتر m شیب خط و پارامتر b عرض از مبدا را نشان می دهد که از معادله زیر قابل محاسبه می باشد:

$$m = \frac{y_{end} - y_0}{x_{end} - x_0}$$
$$b = y_0 - m \cdot x_0$$

با توجه به توضیحات فوق و با استفاده از معادله رسم خط به راحتی قادریم با نقطه یابی یک خط ترسیم کنیم.

مثال: با استفاده از معادله رسم خط یک از نقطه ۲۰و۱۰ الی ۵۰و۳۰ ترسیم کنید.

$$m = \frac{\Delta y}{\Delta x} = \frac{y۲ - y۱}{x۲ - x۱} => \frac{۵۰ - ۲۰}{۳۰ - ۱۰} = \frac{۳۰}{۲۰} = ۱.۵$$

$$b = y - mx => b = y۱ - mx۱ = ۲۰ - ۱.۵ * ۱۰ = ۵$$

مرحله k ام	X	Y
۰	۱۰	Y=۱.۵*۱۰+۵=۲۰
۱	۱۱	۲۱.۵
۲	۱۲	۲۳
۳	۱۳	۲٤.۵
....	.....	.....
۲۰	۳۰	Y=۱.۵*۳۰+۵=۵۰

بنابراین با توجه به مطالب فوق شبه کد زیر برای رسم یک خط به روش مستقیم می تواند به صورت زیر باشد:

```
Linedirect(int x۱,int y۱,int x۲,int y۲,int color)
{
  Float m,b;
  Int x,y;
  M=(y۲-y۱)/(x۲-x۱);
  B=y۱-m*x۱;
  For(x=x۱;x<x۲;x++)
  {
    Y=m*x+b;
    Setpixel(x,y,color)
   }
}
```

اما اشکالات این روش این است که به علت وجود عملیات ریاضی زیاد بسیار کند عمل می کند . و خط ترسیم شده توسط این روش یک خط بریده بریده می باشد و Flat نیست (بدلیل عملیات گرد کردن). بنابراین گفته این الگوریتم روش مناسبی برای رسم خط در فضای نمایشگر های کامپیوتر نیست. به همین دلایل سعی می شود از روشهای دیگر استفاده شود.

الگوریتم رسم خط به روش(Digital Differential Analyzer) DDA:

یکی از مشکلات الگوریتم مستقیم این است که در هر مرحله بدون توجه به شیب خط یک واحد مقدار x افزایش پیدا می کرد و مقدار Y محاسبه می شد. همین موضوع موجب می شود که خط بریده بریده ترسیم شود. بنابراین یکی از الگوریتم هایی که بر اساس شیب خط عمل می کند. الگوریتم DDA می باشد. در این الگوریتم ابتدا شیب خط محاسبه می شود. سپس با توجه به شیب محاسبه شده میزان تغییرات نقاط در جهت محور x(Δx) و همچنین میزان تغییرات در جهت محور y(Δy) بدست می آید . بنابر همین موضوع اگر Δy بزرگتر از Δx باشد آنگاه شیب بزرگتر از یک خواهد شد . در این حالت در هر مرحله یک واحد به y اضافه می شود و مقدار y به صورت زیر محاسبه می شود:

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

همچنین اگر Δx بزرگتر از Δy باشد آنگاه شیب خط کوچکتر از یک خواهد شد . در این حالت در هر مرحله یک واحد به x اضافه می شود و مقدار y به صورت زیر محاسبه می شود:

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

بنابراین عملکرد این الگوریتم به طور خلاصه تر به شرح زیر است:

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

حالت اول: در صورتیکه شیب خط مقدار مثبت و کمتر از یک باشد در این صورت در هر مرحله مقدار Δx را یک واحد افزایش می دهیم و سپس مقدار y را از فرمول ۱ محاسبه می کنیم.

حالت دوم: در صورتیکه شیب خط مقدار مثبت و بیشتر از یک باشد در این صورت در هر مرحله مقدار Δy را یک واحد افزایش می دهیم و سپس مقدار x را از فرمول ۲ محاسبه می کنیم.

از آنجا که مقدار m می تواند هر عدد حقیقی باشد بنابراین نیاز به گرد کردن عدد حاصل داریم. الگوریتم DDA الگوریتم سریعی برای پیدا کردن پیکسل ها است که مستقیما از معادله خط استفاده می کند. اما به دلیل عملیات پیاپی در گرد کردن می تواند منجر به خطاهای چشمگیری در خطوط طولانی شود. همچنین عملیات گرد کردن و کار با اعداد اعشاری بسیار وقت گیر است. رویه اگوریتم DDA به صورت زیر است:

```
#include "device.h"

#define ROUND(a) ((int)(a+0.5))

void lineDDA (int xa, int ya, int xb, int yb)
{
  int dx = xb - xa, dy = yb - ya, steps, k;
  float xIncrement, yIncrement, x = xa, y = ya;

  if (abs (dx) > abs (dy)) steps = abs (dx);
  else steps = abs  dy);
  xIncrement = dx / (float) steps;
  yIncrement = dy / (float) steps;

  setPixel (ROUND(x), ROUND(y));
  for (k=0; k<steps; k++) {
    x += xIncrement;
    y += yIncrement;
    setPixel (ROUND(x), ROUND(y));
  }
}
```

لازم به ذکر است که در صورتی که شیب خط منفی باشد. تمامی واحد های افزایش برای x,y منفی خواهد شد. و یا به عبارتی دیگر به جای افزایش کاهش صورت می گیرد.

مثال: الگوریتم DDA را برای یک خط با مشخصات روبرو دنبال کنید؟ (از نقطه ۲۰و۱۰ الی ۵۰و۳۰)

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1} => \frac{۵۰ - ۲۰}{۳۰ - ۱۰} = \frac{۳۰}{۲۰} = ۱.۵ > ۱$$

حالت دوم رخ داده است بنابراین در هر مرحله مقدار y یک واحد افزایش می یابد و مقدار x طبق فرمول زیر محاسبه می شود:

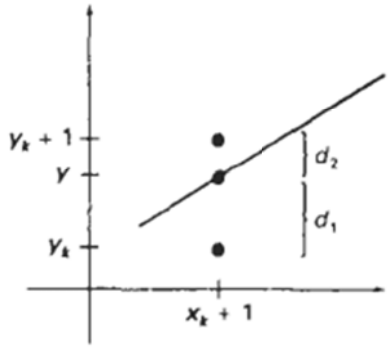
$$x_{k+1} = x_k + \frac{1}{m}$$

ام kمرحله	X	Y
۰	۱۰	Y=۲۰
۱	۱۰+۰.۶۶=۱۰.۶=۱۱	۲۱
۲	۱۱+۰.۶۶=۱۱.۶۶=۱۲	۲۲
۳	۱۲+۰.۶۶=۱۲.۶۶=۱۳	۲۳
....	.....	.....
.....	.....	.....

رسم خط با استفاده از الگوریتم برسنهاوم: Bresenham**:**

الگوریتمی سریع ، محاسبات بر روی اعداد صحیح، قابل اصلاح برای رسم دایره و یا سایر منحنی ها:

در این الگوریتم برای یک حالت مانند شکل زیر تصمیم گیری بر روی این مساله است که بعد رسم اولین نقطه کدام یک از نقاط ترسیم شود. به همین منظور فاصله نقاط قابل نمایش بر روی صفحه نمایش (دستگاه مختصات صفحه نمایش)با معادله خط اصلی در دستگاه مختصات ریاضی محاسبه می شود و نقطه ای روشن می شود که فاصله نزدیکتری با خط دارد. و یا به بیانی دیگر مقادیر صحیح در دو حالت را محاسبه کرده بنابراین دو نقطه وجود خواهد داشت که فاصله آنها تا خط اصلی یا نقطه واقعی معادله خط در دستگاه مختصات ریاضی محاسبه می شود هر کدام از دو حالت که به این نقطه نزدیکتر باشند آنرا روشن می شود.



برای هنگامی که تغییرات x از تغییرات y شدیدتر است یعنی

$|m|<1$

در هر مرحله برای خطی با مقدار شیب فوق همیشه یک واحد x افزایش می یابد و برای افزایش یافتن و یا ثابت ماندن y نسبت به مرحله قبل تصمیم گیری می شود.

It may plot the point (x+۱,y), or:

It may plot the point (x+۱,y+۱).

1. Input the two line endpoints and store the left endpoint in (x_0, y_0).
2. Load (x_0, y_0) into the frame buffer; that is, plot the first point.
3. Calculate constants Δx, Δy, $2\Delta y$, and $2\Delta y - 2\Delta x$, and obtain the starting value for the decision parameter as

$$p_0 = 2\Delta y - \Delta x$$

4. At each x_k along the line, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point to plot is $(x_k + 1, y_k)$ and

$$p_{k+1} = p_k + 2\Delta y$$

Otherwise, the next point to plot is $(x_k + 1, y_k + 1)$ and

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

5. Repeat step 4 Δx times.

برای هنگامی که تغییرات y از تغییرات x شدیدتر است یعنی

|m|>۱

در هر مرحله برای خطی با مقدار شیب فوق همیشه یک واحد yافزایش می یابد و برای افزایش یافتن و یا ثابت ماندن x نسبت به مرحله قبل تصمیم گیری می شود:

It may plot the point (x,y+۱), or:

It may plot the point (x+۱,y+۱).

برای استفاده از الگوریتم فوق فقط کافی است جای x, y با هم جا به جا شوند

همانطور که مشخص است مشکل این الگوریتم این است که به شیب خط بستگی دارد .

مثال:

الگوریتم رسم خط به روش برسنهاوم را برای یک خط با مشخصات p۱(۲۰,۱۰),p۲(۳۰,۱۸)

نکته : اگر در حالت اول مقدار X نقطه اول(p۱) بزرگتر از نقطه دوم باشد جای دو نقطه را با هم تعویض می کنیم و محاسبات را با دو نقطه جدید انجام می دهیم. وهمچنین در حالت دوم اگر مقدار Y نقطه اول (p۱) بزرگتر از نقطه دوم باشد جای دو نقطه را با هم تعویض می کنیم.

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

M=(y۲-y۱)/(x۲-x۱)=۰.۸

$$p_0 = 2\Delta y - \Delta x$$
$$= 6$$

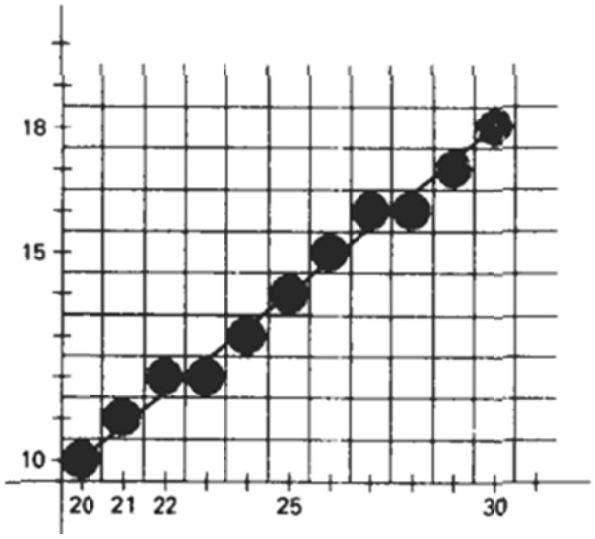
$$2\Delta y = 16, \qquad 2\Delta y - 2\Delta x = -4$$

Plot(۲۰,۱۰) نقطه اول روشن می شود

بقیه نقاط عبارتند از:

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)

خط ترسیمی مانند زیر است:



کد رسم خط برسنهاوم:

```
void Bresenham(int x۱, int y۱, int x۲, int y۲)
    {
      int slope;
      int dx, dy, incE, incNE, d, x, y;
      if (x۱ > x۲)
      {
        Bresenham(x۲, y۲, x۱, y۱);
        return;
      }
      dx = x۲ - x۱;
      dy = y۲ - y۱;
      if (dy < ۰)
      {
        slope = -۱;
        dy = -dy;
      }
      else
      {
        slope = ۱;
      }
      incE = ۲ * dy;
      incNE = ۲ * dy - ۲ * dx;
      d = ۲ * dy - dx;
      y = y۱;
     for (x = x۱; x <= x۲; x++)
      {
        putpixel(x, y);
        if (d <= ۰)
        {
          d += incE;
        }
        else
        {
          d += incNE;
          y += slope;
        }
      }
    }
```

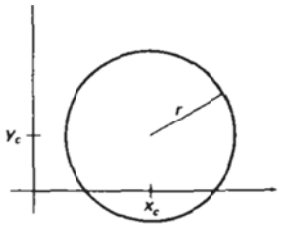
رسم دایره (circle)

دایره مجموعه ای از نقاط است که فاصله آنها تا یک نقطه خاص به نام مرکز به یک اندازه است یعنی r

- الگوریتم رسم خط به روش مستقیم (معادله دکارتی و معادله زاویه یا قطبی)
- الگوریتم رسم خط به روش Midpoint

الگوریتم رسم دایره به روش مستقیم:

معادله دکارتی دایره به صورت زیر می باشد:

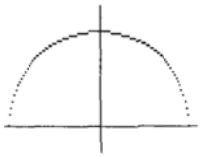


$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

حال اگر معادله فوق را بر اساس X بنویسیم و مقدار X از xc-r تا xc+r تغییر دهیم و y را حساب کنیم داریم:

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

بنابراین اگر یک دایره با معادله فوق ترسیم کنیم شکل ترسیمی مانند زیر خواهد بود:



مثال: الگوریتم رسم دایره با استفاده از فرمول ضمنی را برای یک دایره با مشخصات زیر دنبال کنید؟

مرکز دایره : ۲۰,۳۰ و شعاع:۱۰

X	Y
۲۰ – ۱۰=۱۰	$۳۰ \pm \sqrt{۱۰^۲ - (۱۰ - ۲۰)^۲} = ۳۰ \pm \sqrt{۱۰۰ - ۱۰۰} = ۳۰$
۱۱	$۳۰ \pm \sqrt{۱۰^۲ - (۱۱ - ۲۰)^۲} = ۳۰ \pm \sqrt{۱۰۰ - ۸۱} = ۳۰ \pm ۴.۳۵ = ۳۴.۳۵$ و ۲۵.۶۵
……	
۳۰+۱۰=۳۰	$۳۰ \pm \sqrt{۱۰^۲ - (۳۰ - ۲۰)^۲} = ۳۰ \pm \sqrt{۱۰۰ - ۱۰۰} = ۳۰ \pm ۰ = ۳۰$

مشکل استفاده از این الگوریتم این است که محاسبات آن به دلیل استفاده از ضرب و عمل جذر سنگین است و دایره ترسیم شده توسط آن یک دایره Flat نیست(مانند شکل فوق)

رسم دایره با استفاده از معادله زاویه به شکل زیر است:

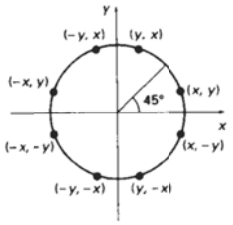
$$x = x_c + r\cos\theta$$
$$y = y_c + r\sin\theta$$

برای استفاده از الگوریتم مقدار θ می بایست بین ۰ تا ۳۶۰ بر حسب رادیان تغییر کند و در هر بار تغییر مقدار جدید X,Y محاسبه شود که در این حالت بهترین مقدار تغییر برای θ ، ۱/r می باشد.

مثال: مثال قبلی را با استفاده از الگوریتم زوایه دنبال کنید:

X	Y	θ
$x = x_c + r\cos\theta$	$y = y_c + r\sin\theta$	
X=۲۰+۱۰ *cos ۰=۳۰	Y=۳۰+۱۰*sin ۰=۳۰	۰
X=۲۰+۱۰ *cos ۰.۱=۲۹.۹	Y=۳۰+۱۰*sin ۰.۱=۳۰.۰۰۱	۰+۱/r=۰+۰.۱=۰.۱
X=۲۰+۱۰*cos ۰.۲=	Y=۳۰+۱۰*sin ۰.۲=	۰.۱+۰.۱=۰.۲
.....		

الگوریتمهای رسم دایره دارای قرینه هشت می باشد یعنی با داشتن یک نقطه می توان هشت نقطه دیگر را محاسبه کرد مانند شکل زیر:



دایره ترسیمی توسط این روش یک دایره Flat می باشد اما به دلیل عملیات سنگین مثلثاتی بسیار کند عمل می کند.

رسم دایره به روش Midpoint:

اساس این الگوریتم این است که در هر مرحله نزدیکترین نقطه به مسیر و محدود مرزی دایره انتخاب می شود.

همچنین در این روش فرض بر این است که مرکز دایره مبدا مختصات است و بعد از محاسبه مختصات نقطه مورد نظر برمحدود دایره مختصات مرکز به آن افزوده می شود. بنابراین :

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

که در آن شرایط زیر ممکن است رخ دهد:

$$f_{circle}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the circle boundary} \\ = 0, & \text{if } (x, y) \text{ is on the circle boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the circle boundary} \end{cases}$$

همانطور که در شکل زیر مشخص است دو پیکسل به عنوان کاندید وجود دارد:

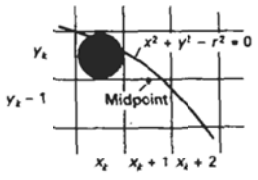


Figure 3-15
Midpoint between candidate pixels at sampling position x_k+1 along a circular path.

بعد از ترسیم نقطه (x_k, y_k)، برای ترسیم نقطه بعدی دو انتخاب وجود دارد:

۱ـ $(x_k + 1, y_k)$

۲ـ $(x_k + 1, y_k - 1)$

که برای انجام این تصمیم گیری تابع دایره را برای نقطه میانی این دو پیکسل محاسبه می شود یعنی :

$$p_k = f_{\text{circle}}\left(x_k + 1, y_k - \frac{1}{2}\right)$$

محاسبه می شود بعد از آن اگر این مقدار مثبت باشد پیکسل پایینی یعنی $(x_k + 1, y_k - 1)$ و در غیر اینصورت پیکسل بالایی یعنی $(x_k + 1, y_k)$ انتخاب می شود. برای ادامه مراحل ، از فرمول فوق یک معادله بازگشتی ساخته می شود و هر مرحله طبق آن عمل می کند. الگوریتم کلی آن به شکل زیر است:

Midpoint Circle Algorithm

1. Input radius r and circle center (x_c, y_c), and obtain the first point on the circumference of a circle centered on the origin as

$$(x_0, y_0) = (0, r)$$

2. Calculate the initial value of the decision parameter as

$$p_0 = \frac{5}{4} - r$$

3. At each x_k position, starting at $k = 0$, perform the following test: If $p_k < 0$, the next point along the circle centered on (0, 0) is (x_{k+1}, y_k) and

$$p_{k+1} = p_k + 2x_{k+1} + 1$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p_{k+1} = p_k + 2x_{k+1} + 1 - 2y_{k+1}$$

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

4. Determine symmetry points in the other seven octants.
5. Move each calculated pixel position (x, y) onto the circular path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, \qquad y = y + y_c$$

6. Repeat steps 3 through 5 until $x \geq y$.

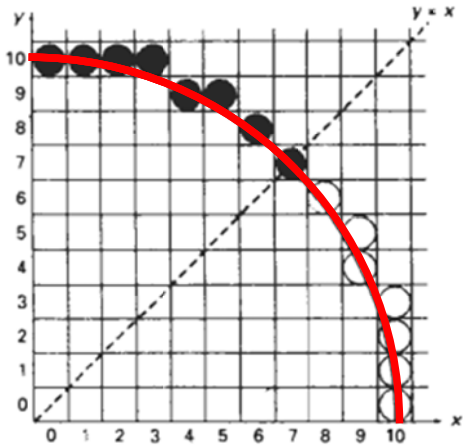
مثال : الگوریتم رسم دایره به روش Midpointرا برای یک دایره به شعاع ۱۰ و به مرکز (۰،۰) دنبال کنید؟

نقطه شروع را (۱۰،۰) در نظر می گیریم.

$$p_0 = 1 - r = -9$$

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

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



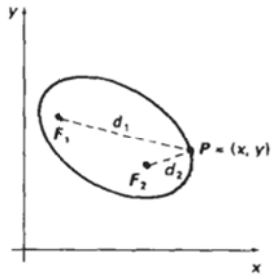
همانطور که در شکل و الگوریتم فوق مشخص است دایره برای یک هشتم رسم می شود. یعنی از x=۰ تا x=y

```
#include "device.h"

void circleMidpoint (int xCenter, int yCenter, int radius)
{
  int x = 0;
  int y = radius;
  int p = 1 - radius;
  void circlePlotPoints (int, int, int, int);

  /* Plot first set of points */
  circlePlotPoints (xCenter, yCenter, x, y);

  while (x < y) {
    x++;
    if (p < 0)
      p += 2 * x + 1;
    else {
      y--;
      p += 2 * (x - y) + 1;
    }
    circlePlotPoints (xCenter, yCenter, x, y);
  }
}

void circlePlotPoints (int xCenter, int yCenter, int x, int y)
{
  setPixel (xCenter + x, yCenter + y);
  setPixel (xCenter - x, yCenter + y);
  setPixel (xCenter + x, yCenter - y);
  setPixel (xCenter - x, yCenter - y);
  setPixel (xCenter + y, yCenter + x);
  setPixel (xCenter - y, yCenter + x);
  setPixel (xCenter + y, yCenter - x);
  setPixel (xCenter - y, yCenter - x);
}
```

الگوریتمهای رسم بیضی:

یک تعریف مقدماتی : بیضی یک دایره کشیده شده می باشد. بنابراین می توان آنرا با اصلاح الگوریتم دایره رسم کرد.

اما بیضی به صورت مجموعه ای از نقاط تعریف می شود که مجموع فاصله آنها از دو نقطه ثابت به نام کانون ها یک مقدار ثابت می شود.



بنابر شکل فوق خواهیم داشت:

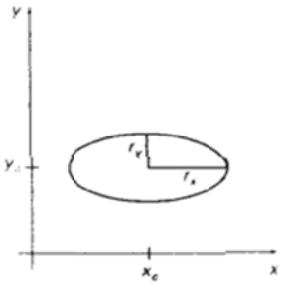
$$d_1 + d_2 = \text{constant}$$

$$\mathbf{F}_1 = (x_1, y_1)$$

$$\mathbf{F}_2 = (x_2, y_2)$$

$$\sqrt{(x - x_1)^2 + (y - y_1)^2} + \sqrt{(x - x_2)^2 + (y - y_2)^2} = \text{constant}$$

اما یک بیضی ساده و استاندارد مانند شکل زیر است که شامل دو شعاع در امتداد X, Y می باشد که الگوریتمهای آتی همگی بر اساس این معادله بیضی بنا نهاده شده است.



که در مختصات دکارتی خواهیم داشت:

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

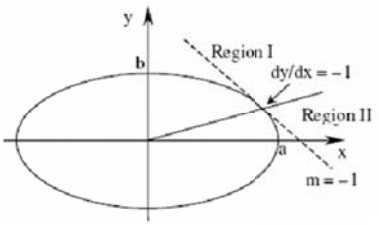
همچنین در مختصات قطبی داریم:

$$x = x_c + r_x \cos\theta$$
$$y = y_c + r_y \sin\theta$$

همچنین می توان با استفاده از تقارن مقداری از پیچیدگی محاسبات را کاهش داد اما بر خلاف دایره بیضی تقارن ۴ دارد و نه ۸ یعنی در بهترین تقارن بیضی را می توان برای یک چهارم ترسیم کرد.

الگوریتم رسم بیضی به روش Midpoint

این الگوریتم توسعه یافته برسنهام می باشد که مزیت آن، استفاده از عمل جمع در حلقه می باشد که در نهایت منجر به پیاده سازی آسان و سریع شده است منحنی بیضی برای در ربع اول به دو ناحیه تقسیم می شود که در ناحیه اول شیب منحنی بیشتر از ۱- است و در ناحیه دوم شیب منحنی کمتر از ۱- می باشد . که در هر ناحیه محاسبات متفاوتی ارائه می شود



فرض می کنیم بیضی در مرکز مختصات قرار دارد و بعد از توسعه الگوریتم فقط با یک جابه جایی بیضی را به مرکز اصلی خودش منتقل می کنیم بنابراین اگر:

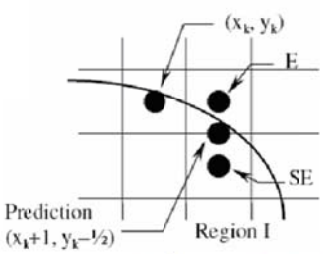
$$(x_c, y_c) = (0, 0)$$

$$f_{\text{ellipse}}(x, y) = r_y^2 x^2 + r_x^2 y^2 - r_x^2 r_y^2$$

وبرای هر نقطه X,Y که در معادله فوق قرار می گیرد شرایط زیر ممکن است رخ دهد:

$$f_{\text{ellipse}}(x, y) \begin{cases} < 0, & \text{if } (x, y) \text{ is inside the ellipse boundary} \\ = 0, & \text{if } (x, y) \text{ is on the ellipse boundary} \\ > 0, & \text{if } (x, y) \text{ is outside the ellipse boundary} \end{cases}$$

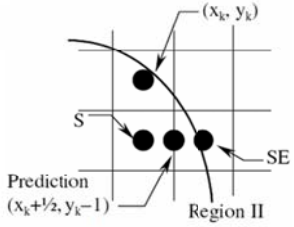
در ناحیه اول که رابطه $dy/dx > -1$ برقرار است :



در هر مرحله مؤلفه x یک واحد افزایش می یابد. به عبارتی داریم : $x_{k+1} = x_k + 1$

داریم $y_{k+1} = y_k$ هر گاه پیکسل E انتخاب شود و داریم $y_{k+1} = y_k - 1$ هر گاه پیکسل SE انتخاب شود. برای اینکه بتوانیم بین انتخاب دو پیکسل S و SE تصمیم بگیریم، نقطه میانی دو پیکسل کاندید را در معادله قرار میدهیم. تابع

در ناحیه دوم که $dy/dx < -1$ تمام محاسبات مانند ناحیه اول است، با این تفاوت که در ناحیه دوم مقدار y در هر 1 واحد کاهش می یابد.



در هر مرحله مؤلفه y یک واحد کاهش می یابد. به عبارتی داریم : $y_{k+1} = y_k - 1$

اگر S انتخاب شود داریم : $x_{k+1} = x_k$ و اگر SE انتخاب شود داریم : $x_{k+1} = x_k + 1$.

برای تصمیم گیریهای فوق الگوریتم MidPoint به ما کمک می کند که به صورت زیر است:

1. Input r_x, r_y, and ellipse center (x_c, y_c), and obtain the first point on an ellipse centered on the origin as

$$(x_0, y_0) = (0, r_y)$$

2. Calculate the initial value of the decision parameter in region 1 as

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2$$

3. At each x_k position in region 1, starting at $k = 0$, perform the following test: If $p1_k < 0$, the next point along the ellipse centered on (0, 0) is (x_{k+1}, y_k) and

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

with

$$2r_y^2 x_{k+1} = 2r_y^2 x_k + 2r_y^2, \qquad 2r_x^2 y_{k+1} = 2r_x^2 y_k - 2r_x^2$$

and continue until $2r_y^2 x \geq 2r_x^2 y$.

4. Calculate the initial value of the decision parameter in region 2 using the last point (x_0, y_0) calculated in region 1 as

$$p2_0 = r_y^2 \left(x_0 + \frac{1}{2}\right)^2 + r_x^2 (y_0 - 1)^2 - r_x^2 r_y^2$$

5. At each y_k position in region 2, starting at $k = 0$, perform the following test: If $p2_k > 0$, the next point along the ellipse centered on (0, 0) is $(x_k, y_k - 1)$ and

$$p2_{k+1} = p2_k - 2r_x^2 y_{k+1} + r_x^2$$

Otherwise, the next point along the circle is $(x_k + 1, y_k - 1)$ and

$$p2_{k+1} = p2_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_x^2$$

using the same incremental calculations for x and y as in region 1.

6. Determine symmetry points in the other three quadrants.
7. Move each calculated pixel position (x, y) onto the elliptical path centered on (x_c, y_c) and plot the coordinate values:

$$x = x + x_c, \qquad y = y + y_c$$

8. Repeat the steps for region 1 until $2r_y^2 x \geq 2r_x^2 y$.

مثال: الگوریتم رسم بیضی به روش Midpoint را برای یک بیضی با مشخصات rx=۸,ry=۶ و به مرکز (۰,۰) دنبال کنید؟

$2r_y^2 x = 0$ (with increment $2r_y^2 = 72$)

$2r_x^2 y = 2r_x^2 r_y$ (with increment $-2r_x^2 = -128$)

(x۰,y۰)=(۰,ry)=(۰,۶)

$$p1_0 = r_y^2 - r_x^2 r_y + \frac{1}{4} r_x^2 = -332$$

=> ۳۶-۶۴*۶+۰.۲۵*۶۴=۳۶-۳۸۴+۱۶=۵۲-۳۸۴=-۳۳۲

k	$p1_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	-332	(1, 6)	72	768
1	-224	(2, 6)	144	768
2	-44	(3, 6)	216	768
3	208	(4, 5)	288	640
4	-108	(5, 5)	360	640
5	288	(6, 4)	432	512
6	244	(7, 3)	504	384

For k=۰: p۱k=-۳۳۲<۰ so:

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} + r_y^2$$

=> -۳۳۲+۷۲+۳۶=-۳۳۲+۱۰۸=-۲۲۴

For k=٣: p١k=٢٠٨>٠ so:

$$p1_{k+1} = p1_k + 2r_y^2 x_{k+1} - 2r_x^2 y_{k+1} + r_y^2$$

=>٢٠٨+٢٨٨-٦٤٠+٣٦=-١٠٨

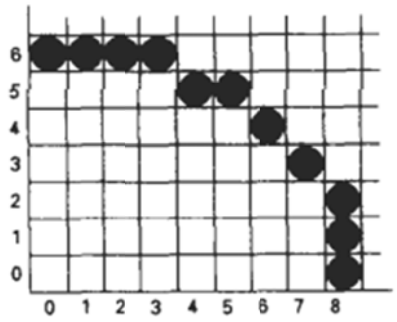
We now move out of region 1, since $2r_y^2 x > 2r_x^2 y$.

For region 2, the initial point is $(x_0, y_0) = (7, 3)$ and the initial decision parameter is

$$p2_0 = f\left(7 + \frac{1}{2}, 2\right) = -151$$

P٢٠=٣٦*٧.٥*٧.٥+٦٤*٢*٢-٦٤*٣٦=٢٠٢٥+٢٥٦-٢٣٠٤=-٢٣

k	$p2_k$	(x_{k+1}, y_{k+1})	$2r_y^2 x_{k+1}$	$2r_x^2 y_{k+1}$
0	−151	(8, 2)	576	256
1	233	(8, 1)	576	128
2	745	(8, 0)	—	—



فصل چهارم

تبدیلات هندسی

تبدیلات هندسی عبارتند از تغییر در موقعیت یا شکل و یا اندازه اشکال و تصاویر. تبدیلات هندسی اولیه عبارتند از : انتقال، دوران و تغییر مقیاس. سایر تبدیلاتی که به اشکال اعمال میشوند عبارتند از : انعکاس و تبدیلات Shear .

تبدیلات اولیه :

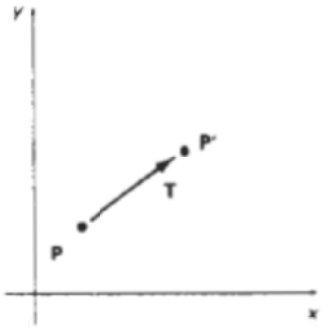
در اینجا به تبدیلات هندسی پایه میپردازیم و در بخش بعدی بیان میکنیم چگونه این تبدیلات را به صورت بیان ریاضی از ماتریسهای ساده تر پیاده سازی کنیم تا بتوان ترکیبی از این تبدیلات را محاسبه کرد.

انتقال :

انتقال یعنی حرکت دادن یک شئ در امتداد یک خط راست از نقطه ای به نقطه دیگر. برای انتقال یک نقطه در مختصات دو بعدی کافیست مختصات بردار انتقال (t_x, t_y) را به مختصات آن نقطه اضافه کنیم.

$$x' = x + t_x \quad , \quad y' = y + t_y$$

شکل زیر انتقال نقطه توسط بردار انتقال نشان میدهد :



معادلات بالا را میتوان به صورت ماتریسی نیز بیان کرد:

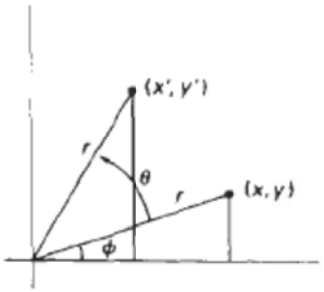
$$p = \begin{bmatrix} x \\ y \end{bmatrix} \ , \ p' = \begin{bmatrix} x' \\ y' \end{bmatrix} \ , \ t = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$p' = p + t$$

واضح است که انتقال نوعی تبدیل هندسی سخت است یعنی شکل را تغییر نمیدهد، تنها مکان هرنقطه از تصویر را توسط یک مقدار ثابت تغییر میدهد.

دوران :

دوران در مختصات دو بعدی یعنی انتقال شئ روی یک مسیر دایره ای در صفحه xy . برای انجام عمل دوران به دو پارامتر نیاز داریم: اول زاویه چرخش و دوم نقطه محوری. ابتدا معادلات دوران یک نقطه حول مبدأ مختصات را بدست می آوریم. شکل زیر دوران نقطه حول مبدأ مختصات را نشان میدهد.



$$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$
$$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

همچنین مختصات اولیه نقطه در مختصات قطبی برابر است با :

$$x = r\cos\phi \quad , \quad y = r\sin\phi$$

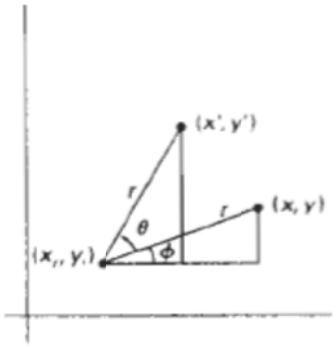
با جایگزینی معادلات بالا داریم:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$

مانند قبل میتوان این معادلات را بر حسب ماتریسها بدست آورد:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \ , \ p' = \begin{bmatrix} x' \\ y' \end{bmatrix} \ , \ \mathrm{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$
$$p' = R.p$$

برای دوران حول نقطه غیرمبدأ از فرمول های زیر استفاده میکنیم:



$$x' = x_1 + (x - x_1)\cos\theta - (y - y_1)\sin\theta$$
$$y' = y_1 + (x - x_1)\sin\theta + (y - y_1)\cos\theta$$

نکته 1 : جرخش در جهت عکس حرکت عقربه های ساعت است .

نکته 2 : واضح است که دوران یک تبدیل سخت است..

تغییر مقیاس:

تغییر مقیاس تبدیلی است که اندازه تصویر را تغییر میدهد. برای تغییر اندازه از ضریب مقیاس استفاده میشود.

$$x' = x \cdot s_x \quad , \quad y' = y \cdot s_y$$

که در آن s_x تغییر اندازه روی محور x و s_y تغییر اندازه روی محور y میباشد. این معادلات را میتوان به صورت معادلات زیر نوشت:

$$p = \begin{bmatrix} x \\ y \end{bmatrix} \ , \ p' = \begin{bmatrix} x' \\ y' \end{bmatrix} \ , \ \mathrm{S} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \qquad \Rightarrow p' = S.p$$

میتوان هر مقدار مثبتی را به ضریب مقیاس نسبت داد. مقادیر کمتر از یک موجب کوچک شدن و مقادیر بیشتر از یک موجب بزرگتر شدن جسم می شود. اگر به هر دو مؤلفه ی ضریب مقیاس، مقدار یک داده شود، در این صورت شکل تغییر اندازه نخواهد داد.

نمایش ماتریسی و مختصات همگن:

در برنامه های گرافیکی معمولاً از ترکیبی از تبدیلات متوالی استفاده میشود. دراین بخش ما فرمولهای ماتریسی بخش قبل را دوباره بازنویسی میکنیم تا فرایند چند تبدیل متوالی را راحتتر انجام دهیم. در بخش پیش دیدیم که هر یک از تبدیلات اولیه را میتوان به صورت زیر بیان کرد :

$$P' = M_1.P + M_2$$

که در آن P مختصات نقطه اولیه، P' مختصات نقطه نهایی، M_1 یک ماتریس دو در دو که میتواند یک ماتریس ضریب مقیاس و یا یک ماتریس دوران باشد و در نهایت M_2 برابر با یک ماتریس یک در دو است که یک ماتریس انتقال است. اگر به جای استفاده از ماتریس دو در دو از یک ماتریس سه در سه استفاده کنیم، میتوانیم تمام تبدیلات را بصورت ضرب بیان کنیم. برای این منظور به جای استفاده از مختصات دوبعدی (x, y) از مختصات سه بعدی $(x, y, 1)$ استفاده میکنیم. استفاده از مختصات سه بعدی این امکان را میدهد تا فرمولهای تبدیلات را دوباره و تنها بر اساس ضرب بیان کنیم. حال دوباره فرمولهای تبدیلات هندسی را بر اساس ضرب ماتریسهای سه در سه بیان میکنیم:

انتقال :

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad , \quad P' = T(t_x, t_y) \cdot P$$

دوران:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad , \quad P' = R(\theta) \cdot P$$

تغییر مقیاس:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad , \quad P' = S(s_x, s_y) \cdot P$$

تبدیلات مرکب

با استفاده از نمایش ماتریسی که در بخش قبل ارائه شد، میتوان برای هر ترکیبی از تبدیلات یک ماتریس تبدیل مرکب بنویسیم. ماتریس حاصل را معمولاً ماتریس مرکب یا ماتریس الحاق می نامند. برای بدست آوردن ماتریس تبدیل مرکب، ماتریسهای تبدیل را از آخر به اول در هم ضرب میکنیم.

اگر دو تبدیل انتقال T_1 و T_2 به یک نقطه اعمال شود، ماتریس تبدیل مرکب به صورت زیر بدست می آید.

$$\begin{bmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{bmatrix}$$

به همین ترتیب برای دو دوران متوالی $R(\theta_1)$ و $R(\theta_2)$ داریم :

$$R_{total} = R(\theta_2).R(\theta_1) = R(\theta_1+\theta_2)$$

و در نهایت برای دو تغییر مقیاس متوالی با ضریب S_1 و S_2 داریم :

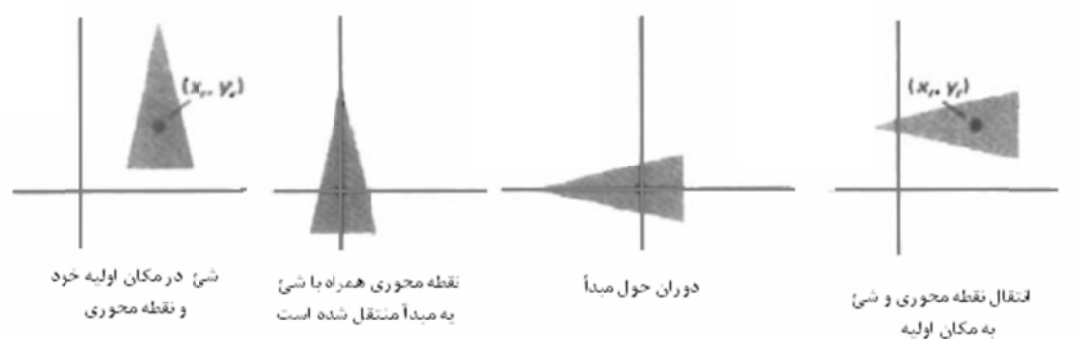
$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1}.s_{x2} & 0 & 0 \\ 0 & s_{y1}.s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

دوران حول نقطه دلخواه :

برای دوران حول نقطه دلخواه مراحل زیر را دنبال میکنیم:

1- شکل را همراه با نقطه محوری به مبدأ انتقال میدهیم.
2- شکل را حول مبدأ مختصات دوران میدهیم .
3- شکل را به مکان اولیه خود باز میگردانیم.

این انتقالات در شکل زیر نشان داده شده است(از چپ به راست).



بنابراین ابتدا یک انتقال در امتداد بردار $(-x_c, -y_c)$ سپس یک دوران به اندازه θ و در نهایت یک انتقال دیگر در امتداد بردار (x_c, y_c). لذا ماتریس ترکیب به صورت زیر بدست می آید.

$$\begin{bmatrix}1&0&x_r\\0&1&y_r\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}1&0&-x_r\\0&1&-y_r\\0&0&1\end{bmatrix}=\begin{bmatrix}\cos\theta&-\sin\theta&x_r(1-\cos\theta)+y_r\sin\theta\\\sin\theta&\cos\theta&y_r(1-\cos\theta)-x_r\sin\theta\\0&0&1\end{bmatrix}$$

برای عمل تغییر مقیاس نیز کافیست که نقطه را به مبدأ مختصات انتقال و سپس عمل تغییر مقیاس را انجام دهیم.

تبدیلات سه بعدی :

انتقال : در مختصات سه بعدی، یک نقطه از موقعیت $P(x,y,z)$ به موقعیت $P'(x',y',z')$ توسط ماتریس زیر انتقال می یابد.

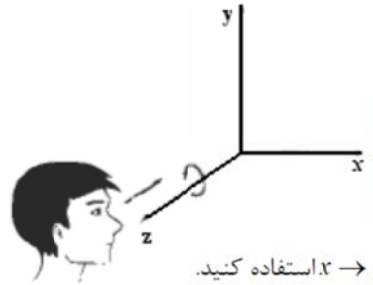
$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}1&0&0&t_x\\0&1&0&t_y\\0&0&1&t_z\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\Rightarrow\begin{cases}x=x+t_x\\y=y+t_y\\z=z+t_z\end{cases}$$

تغییر اندازه : ماتریس تغییر اندازه را میتوان به صورت زیر نوشت :

$$\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}S_x&0&0&0\\0&S_y&0&0\\0&0&S_z&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}\Rightarrow\begin{cases}x=x.S_x\\y=y.S_y\\z=z.S_z\end{cases}$$

دوران : برای تولید یک تبدیل دوران باید محور دوران و زاویه چرخش را مشخص کنیم. در سه بعدی دوران را میتوان حول هر خط موجود در فضا انجام داد اما در اینجا ما تنها دوران حول محورهای مختصات را بررسی می کنیم.

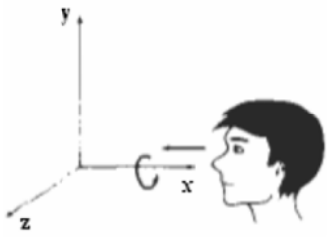
دوران حول محور z :



$$\begin{cases}x'=x\cos\theta-y\sin\theta\\y'=x\sin\theta+y\cos\theta\\z'=z\end{cases}\Rightarrow\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}\cos\theta&-\sin\theta&0&0\\\sin\theta&\cos\theta&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

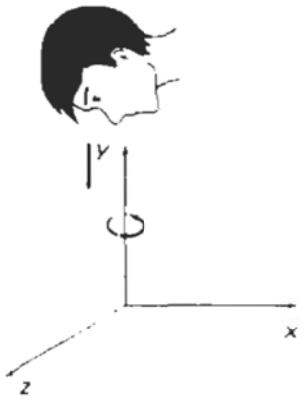
برای بدست آوردن فرمول دوران حول سایر محورها از جایگذاری $x\to y\to z\to x$ استفاده کنید.

دوران حول محور x :



$$\begin{cases}x'=x\\y'=y\cos\theta-z\sin\theta\\z'=y\sin\theta+z\cos\theta\end{cases}\Rightarrow\begin{bmatrix}x'\\y'\\z'\\1\end{bmatrix}=\begin{bmatrix}1&0&0&0\\0&\cos\theta&-\sin\theta&0\\0&\sin\theta&\cos\theta&0\\0&0&0&1\end{bmatrix}\cdot\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

دوران حول محور y :

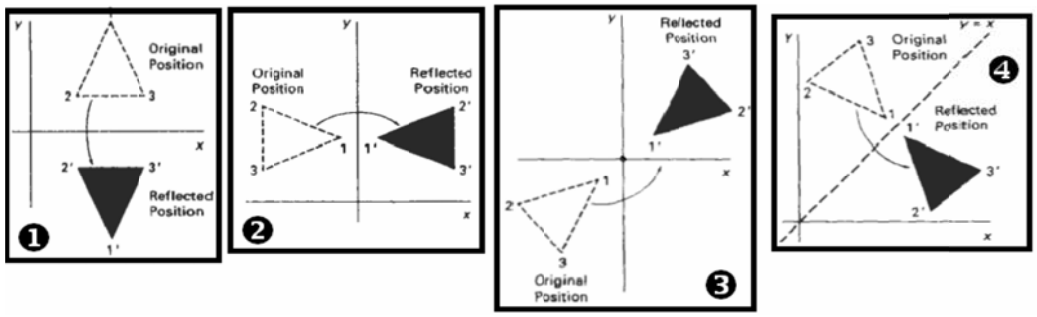


$$\begin{cases} x' = z\cos\theta - x\cos\theta \\ y' = y \\ z' = z\cos\theta - x\sin\theta \end{cases} \Rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

سایر تبدیلات :

علاوه بر تبدیلات اولیه که بیان شدند، تبدیلات مفید دیگری نیز وجود دارند که مهمترین آنها عبارتند از : بازتاب و تبدیلات Shear.

برای عمل بازتاب یک محور بازتاب نیاز داریم . در دو بعدی این محور میتواند هر خطی در صفحه مختصات باشد. ما در این جا بازتاب نسبت به محورهای اصلی و محورهای فرعی بیان میکنیم.

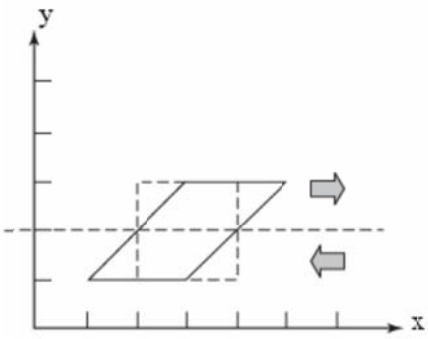


معادلات مربوط به بازتاب اشکال بالا عبارتند از :

$$\textbf{1}\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{2}\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{3}\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \textbf{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

تبدیلات Shear :

تبدیلات Shear تصویر را در جهتهای مشخص شیفت میدهند. در واقع این تبدیلات نقطه را به نسبت فاصله از یک خط و موازی با خط حرکت میدهند. نقاط واقع بر روی خط شیفت داده نمیشود و نقاطی که در طرف مقابل اند در جهت مخالف شیفت میابند. تبدیلات Shear شئ را دگرگون میکنند اما با این حال موازی بودن خطوط را حفظ میکنند. شکل روبرو یک نمونه از این تبدیل را نشان میدهد که با مقیاس 1 و نسبت به خط $y = 2$ انجام شده است.



معمولاً شیفت Shear در جهت محور x یا y انجام میشود .

شیفت Shear در جهت محور x : ماتریس زیر نقاط را با مقیاس sh_x و به نسبت فاصله از خط $y = y_{ref}$ و موازی با آن شیفت میدهد.

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x + sh_x (y - y_{ref}) \\ y' = y \end{cases}$$

شیفت Shear در جهت محور y : ماتریس زیر نقاط را با مقیاس sh_y و به نسبت فاصله از خط $x = x_{ref}$ و موازی با آن شیفت میدهد.

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{cases} x' = x \\ y' = y + sh_y (x - x_{ref}) \end{cases}$$

تبدیلات Affine :

تبدیلاتی هستند که در آنها موازی بودن خطوط حفظ میشود ولی طول خطوط تغییر نمی یابند. معروفترین این تبدیلات عبارتند از : انتقال، دوران و تغییر اندازه. برای این سه تبدیل زاویه بین خطوط بعد از تبدیل تغییر نخواهد کرد.

بخش عملی گرافیک کامپیوتری ۱

شامل:

- مقدمه
- تعریف واقعیت مجازی
- تاریخچهٔ واقعیت مجازی
- چهار عنصر کلیدی واقعیت مجازی
- نمونه های واقعیت مجازی
- توسعهٔ واقعیت مجازی
- نمونه های به‌کارگیری تکنیک‌های واقعیت مجازی

مقدمه

واقعیت مجازی؛ به دنبال گسترش جایگاه خود و اکتساب سهمی است که در آینده می تواند داشته باشد ، حرکت می‌کند. امروزه فناوری اطلاعات روز به روز در حال گسترش می باشد فناوری اطلاعات در واقع یک هجوم است. درچنین دنیایی رقابت در این عرصه بسیار بسیار فشرده و تنگاتنگ می باشد. و واقعیت مجازی ابزاری است برای پیشی گرفتن از رقبا. واقعیت مجازی؛ درواقع یک مزیت رقابتی در دنیای امروز و آینده محسوب می شود. همچنین واقعیت مجازی سعی دارد تا استفاده از خودش را عمومیت بیشتری ببخشد

رسانه‌های ارتباطات انسانی Human Communication Media

- نقاشی‌های غار
- زبان نوشتاری
- مجسمه سازی
- ارتباطات آوایی
- کتاب‌های دست‌نویس
- کتاب‌های چاپی
- روزنامه (اطلاعات دوره‌ای)
- عکاسی
- صداهای ضبط شده
- تلگراف
- تلفن
- رادیو
- تلویزیون
- رایانه‌های دیجیتال
- رایانه‌های گرافیکی
- بازی‌های ویدیویی
- کنفرانس تصویری
- اینترنت
- نمایش تعاملی و حسی کامپیوتر (VR)
- World Wide Web

تفاوت اصلی واقعیت مجازی با سایر رسانه ها در تعاملی بودن آن است

تعریف واقعیت مجازی

- Virtual: وجود ماهیت یا اثر چیزی بدون وجود حقیقی آن
- Reality: حالت یا ظرفیت واقعی بودن
- **Webster defines:**
  - Virtual: *being in essence of effect, but not in fact*
  - Reality: *something that exists independently of ideas concerning it*
- تعریف واقعیت مجازی، همان چیزی است که اکنون در ذهن شماست.
- درواقع کلمه ی واقعیت مجازی در سال ۱۹۳۸توسط یک نمایشنامه نویس و شاعر فرانسوی بکار برده شد. وی معتقد بود که تئاتر یک واقعیت مجازی است.
- در یک مفهوم عام شبیه سازی فضای سه بعدی توسط تکنیکهای نرم افزاری در فضایی که ذاتا دو بعدی است؛ به گونه ای که کاربر می تواند به غیر از مشاهده این فضا ، در آن سیر کرده و در محیط تغییراتی را ایجاد نمایند. برای درک بهتر می توان صفحه شطرنجی که قابلیت چرخاندن و جابه جا کردن مهره ها را دارد

تاریخچهٔ واقعیت مجازی

- ۱۹۲۹

  After several years of flight training,

  Edward Link develops a simple mechanical *flight simulator* to train a pilot at a stationary (indoor) location. The trainee can learn to fly and navigate using instruments via instrument replicas in the cockpit of the *Link Trainer.*

- Edward Link توانست پرواز مکانیکی در یک محیط داخلی شبیه سازی کند.
- ۱۹۶۸

  In the paper "A head-mounted three dimensional display," Ivan Sutherland describes his development of a tracked stereoscopic head-mounted display at Harvard University. The display uses miniature CRTs with optics to present separate images to each eye, and is interfaced to mechanical and ultrasonic trackers.

- Ivan Sutherland اعلام کرد در دانشگاه هاروارد ابزاری را ساخته است که بر روی سر قرار می‌گیرد و امکان پیگیری را در اختیار می‌گذارد. (نخستین دنیای مجازی)
- ۱۹۹۰

  W-Industries launches the first public venue VR system, coining it *Virtuality*. It is a dual-player VR arcade system that includes an HMD, hand-held prop, and ring platform for each participant. The initial game -- *Dactyl Nightmare* -- involves two players in a simple multi-level world where they attempt to shoot one another. In ۱۹۹۳, W-Industries changes their name to Virtuality PLC, and in ۱۹۹۷ they sell their assets as part of filing for Chapter ۱۱ Bankruptcy.

  - از سال ۱۹۹۰: شروع پیشرفت در واقعیت مجازی

- ۱۹۹۸

Disney opens the first of their DisneyQuest family arcade centers which feature numerous VR attractions using both HMD, and projection-based visual displays.

- **از سال ۱۹۹۸: عمومی شدن کاربردهای واقعیت مجازی**

انعطاف‌پذیری واقعیت مجازی

- **اکثر رسانه‌ها به انعطاف‌پذیری واقعیت مجازی نیستند.**
  - **موسیقی در زمان وجود دارد.**
  - **نقاشی در فضا و مکان وجود دارد.**
- **واقعیت مجازی در زمان و فضا توأمان موجود است.**
- **در واقعیت مجازی کاربر می‌تواند چگونگی عبور از زمان و فضا را کنترل نماید.**

عناصر اصلی واقعیت مجازی

- **دنیای مجازی** – Virtual World
- **غوطه وری** – Immersion
- **بازخورد حسی** – Sensory Feedback
- **تعامل** – Interactivity

دنیای مجازی (Virtual World)

- **یک رسانه؛ در ذهن بوجود آورندهٔ آن؛ امکان انتقال به دیگری یا به اشتراک گذاشتن آن**
- **فضایی تخیلی که (اغلب) با یک رسانه آشکار شده است.**
- **توصیفی از مجموعه‌ای از اشیاء در یک فضا و قواعد و روابطی که آن اجزاء را اداره می‌کنند.**

غوطه‌وری (Immersion)

- **غوطه‌وری: احساس بودن در یک محیط؛ می‌تواند کاملاً در حالت ذهنی باشد یا می‌تواند با ابزار فیزیکی تجهیز شده باشد؛ غوطه‌وری فیزیکی با خواصی از واقعیت مجازی شناخته می‌شود؛ غوطه‌وری ذهنی هدف بیشتر سازندگان رسانه‌ها است.**
- **غوطه‌وری ذهنی: احساس کاملاً درگیر شدن؛ دوری از ناباوری؛ پیچیدگی**
- **غوطه‌وری فیزیکی: ورود همراه با جسم (بدن) به درون رسانه؛ محرک غیرواقعی از حس‌های جسمانی با استفاده از تکنولوژی؛ این دلالت بر این نمی‌کند که همهٔ حس‌ها(ی پنج‌گانه) یا همهٔ اجزاء بدن غوطه‌ور شوند.**

بازخورد حسی
(Sensory Feedback)

- تأثیرگذاری
- تأثیرپذیری
- هر کنشی واکنشی به همراه دارد و بازخوردی که به لحاظ آن کنش یا واکنش به‌دست می‌آید
- چشاندن طعم واقعیت به یک سیستم، نیازمند گنجاندن مکانیزم‌هایی است که این بازخوردها را منتقل کند
- سیستم‌های واقعیت مجازی بازخورد حسی مستقیمی به شرکت‌کنندگان می‌دهد که برپایهٔ موقعیت فیزیکی آنها خواهد بود

تعامل (Interactivity)

- کنش با کاربر : تعامل
- مفهومی آشنا در دنیای کامپیوتر
- مثال فیلم در مقابل بازی رایانه ای
- سیستم بدون تعامل: کارایی پایین؛ تکرارپذیری کم؛ خسته کننده

نمونهٔ Active Worlds

- ابزارهایی را جهت ایجاد یک محیط مجازی تعاملی فراهم می کند
- بخشها و اماکن مختلفی وجود دارد.

نمونهٔ Active Worlds(ادامه)

- @mart نمونهٔ دنیایی است که نشان می دهد که چگونه ابزاری می تواند جهت ایجاد یک فروشگاه مجازی سه بعدی به کار رود .

نمونهٔ ADVIRT

- فروش‌گاه‌های سازگار با واقعیت مجازیADaptive VIrtual Reality sTore
- این نمونه برای فروشگاه‌های بزرگ که انواع مختلفی از محصولات را ارائه می‌کنند طراحی شده است
- یک فروشگاه واقعیت مجازی را با استفاده از سابقهٔ مشتریانی که از آن استفاده کرده‌اند، اختصاصی می‌کند
- با درنظر گرفتن سلائق مشتری، از بعد روانی، نزدیکی محیط به واقعیت را برای وی ترسیم می‌کند

نمونهٔ CyberTown

- CyberTown، یک محیط مجازی روی اینترنت
- شعار: شهروندی در عصر مجازی – در سال ۲۰۹۲ میلادی
- در این شهر مجازی، استادیوم، اماکن تفریحی، بیمارستان و مراکز درمانی، کتابخانه، دانشگاه و سایر اماکن وجود دارد
- بخشی نیز جهت امور تجاری و فروشگاه وجود دارد ؛ بخش CyberTown Shopping
- جنبه‌های گرافیکی بسیار قوی است
- مشابه Active Worlds ولی با مظاهر تخیلی

ابزارهای تولید دنیای مجازی

- زبان مدلسازی واقعیت مجازی:
- VRML : Virtual Reality Modeling Language
- X۳D: جانشین VRML باتعریف و انتشار اطلاعات سه بعدی بر روی وب
- نرم‌افزارهای گرافیکی
- نرم افزارهای نمایش دنیای مجازی

Augmented Reality

- نوعی از واقعیت مجازی با اطلاعات اضافی
- یک رسانه نامحسوس که در دنیای فیزیکی احساس می‌شود.
- محیط مجازی مخلوط با فضای حقیقی

مثال: میز و تلفن واقعی، دو صندلی و لامپ مجازی اشیاء مجازی ۳ بعدی هستند شرایط آنها همسو با شرای اشیاء واقعی

پیوست

کدهای VRML

کد VRML مربوط به فضا سازی یک محیط کوهستانی

```
#VRML V۲.۰ utf۸
Background {
   skyColor [
      ۰.۰ ۰.۲ ۰.۷,
      ۰.۰ ۰.۵ ۱.۰,
      ۱.۰ ۱.۰ ۱.۰
   ]
   skyAngle [ ۱.۳۰۹, ۱.۵۷۱ ]
   groundColor [
      ۰.۱ ۰.۱۰ ۰.۰,
      ۰.٤ ۰.۲۵ ۰.۲,
      ۰.٦ ۰.٦۰ ۰.٦,
   ]
   groundAngle [ ۱.۳۰۹, ۱.۵۷۱ ]
   frontUrl "mountns.png"
   backUrl  "mountns.png"
   leftUrl  "mountns.png"
   rightUrl "mountns.png"
}
```

کد VRML مربوط ترسیم اشیای پایه ای:

```
#VRML V۲.۰ utf۸
Shape {
   appearance Appearance {
      material Material {}
       #texture ImageTexture { url "wood.jpg" }

   }
   geometry Box {
      size ۲.۰ ۲.۰ ۲.۰
   }


}
Shape       {
            appearance  Appearance {
                        material Material {
                                    diffuseColor .۳۵ .۱ .۱
                                    specularColor .۳۷ .۰٦ .۰٦
                                    emissiveColor .۱ .۰۳ .۰۳
                                    ambientIntensity .۰۲۳۳
                                    shininess .۰٦
                        }
            }
            geometry Sphere{
                        radius ۱.٤
            }
```

```
}
Shape {
        appearance Appearance {

                material Material {}

        }
        geometry  Cone {
        bottomRadius ١ height ٥
        }
}
```

کد VRML **مربوط ترسیم یک قوطی نوشابه:**

```
Shape {
   appearance Appearance {
        material Material { }
        texture ImageTexture { url "cantop.jpg" }

   }
   geometry Cylinder {
        height  ٢.٧
        side    FALSE
        bottom  FALSE
   }
}
Shape {
   appearance Appearance {
        material Material { }
        texture ImageTexture { url "canbot.jpg" }
   }
   geometry Cylinder {
        height  ٢.٧
        side    FALSE
        top     FALSE
   }
}
Shape {
   appearance Appearance {
        material Material { }
        texture ImageTexture { url "canlabel.jpg" }
   }
   geometry Cylinder {
        height  ٢.٨
        top     FALSE
        bottom  FALSE
   }
}
```

کد VRML **ترسیم یک صندلی:**

```
#VRML V٢.٠ utf٨
Transform {
   translation ٠.٠ ٠.٥ ٠.٠
```

```
  children [
        Shape {
          appearance DEF Brown Appearance {
              material Material {
                diffuseColor ٠.٦ ٠.٣٥ ٠.٠
              }
          }
          geometry Box {
              size ٠.٣٩ ٠.٠٣ ٠.٤١
          }
        }
  ]
}

#
#  Chair legs
#
Transform {
  translation ٠.١٥٧٥ ٠.٢٤٨٥ ٠.١٥٧٥
  children [
        DEF Leg Shape {
          appearance USE Brown
          geometry Box {
              size ٠.٠٣ ٠.٤٩٧ ٠.٠٣
          }
        }
  ]
}
Transform {
  translation -٠.١٥٧٥ ٠.٢٤٨٥ ٠.١٥٧٥
  children [ USE Leg ]
}
Transform {
  translation -٠.١٥٧٥ ٠.٢٤٨٥ -٠.١٥٧٥
  children [ USE Leg ]
}
Transform {
  translation ٠.١٥٧٥ ٠.٢٤٨٥ -٠.١٥٧٥
  children [ USE Leg ]
}

#
#  Chair back
#
Transform {
  translation ٠.١٨٧٥ ٠.٥ ٠.٠
  rotation ٠.٠ ٠.٠ ١.٠ -٠.١٧
  children [
        Transform {
          translation ٠.٠ ٠.٥٤ ٠.٠
          children [
              Shape {
                appearance USE Brown
                geometry Box {
                      size ٠.٠٦ ٠.١٧ ٠.٤٣
                }
              }
          ]
        }

        Transform {
          translation ٠.٠ ٠.٢٢٧٥ ٠.٠
          children [
              DEF BackPole Shape {
                appearance USE Brown
                geometry Box {
                      size ٠.٠٢ ٠.٤٥٥ ٠.٠٢
                }
              }
          ]
        }
        Transform {
          translation ٠.٠ ٠.٢٢٧٥ -٠.٠٨٣
          children [ USE BackPole ]
        }
```

```
            Transform {
              translation ۰.۰ ۰.۲۲۷۵ ۰.۰۸۳
              children [ USE BackPole ]
            }


            Transform {
              translation ۰.۰ ۰.۲۲۷۵ -۰.۱۶۶
              children [ USE BackPole ]
            }

            Transform {
              translation ۰.۰ ۰.۲۲۷۵ ۰.۱۶۶
              children [ USE BackPole ]
            }
  ]
}
```

کد طراحی یک ستون:

```
#VRML V۲.۰ utf۸
 Transform {
        translation ۰.۰ ۰.۰ ۰.۰
        children [
           Shape {
              appearance Appearance {
                 material DEF MarbleMaterial Material {
                    ambientIntensity ۰.٤
                 }
                 texture DEF MarbleImage ImageTexture {
                    url "marble_g.jpg"
                 }
                 textureTransform TextureTransform {
                    translation ۰.۲ ۰.۲
                    rotation ۰.۱
                    scale ۱.۰ ۰.۱
                 }
              }
              geometry Box { size ۱.۱ ۰.۳ ۱.۱ }
           }
        ]
     }

 Transform {
        translation ۰.۰ ۰.۲ ۰.۰
        children [
           Shape {
              appearance Appearance {
                 material USE MarbleMaterial
                 texture USE MarbleImage
                 textureTransform TextureTransform {
                    translation ۰.۲ ۰.۳
                    rotation ۰.٥
                    scale ۱.۰ ۰.۲
                 }
              }
              geometry Box {
                 size ۱.۰ ۰.۱ ۱.۰
```

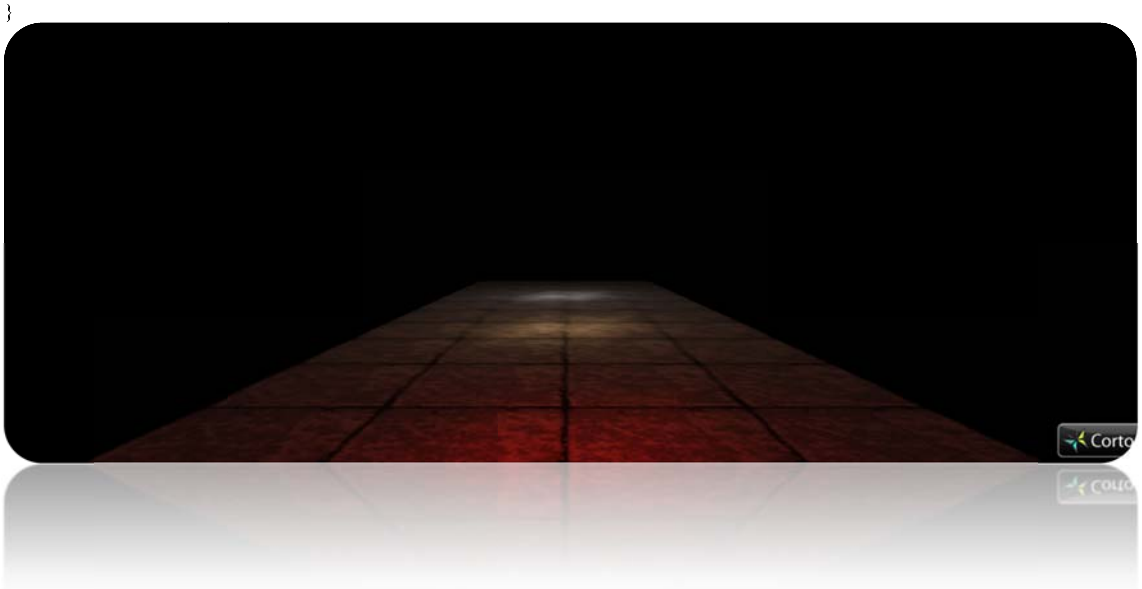
```
            }
          }
        ]
      }


  Transform {
        translation ٠.٠ ٠.٣ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.٢ ٠.٣
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box {
              size ٠.٩ ٠.١ ٠.٩
            }
          }
        ]
      }

  Transform {
        translation ٠.٠ ٠.٤ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.٢ ٠.٣
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box {
              size ٠.٨ ٠.١ ٠.٨
            }
          }
        ]
      }


  Transform {
        translation ٠.٠ ٢.٥ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
            }
            geometry Cylinder {
              height ٤.٢
              radius ٠.٤
              bottom FALSE
              top FALSE
            }
          }
        ]
      }


  Transform {
        translation ٠.٠ ٤.٨٥ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.٢ ٠.٣
```

```
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box {
              size ١.٠ ٠.١ ١.٠
            }
          }
        ]
      }
          Transform {
        translation ٠.٠ ٤.٧٥ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.٢ ٠.٣
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box {
              size ٠.٩ ٠.١ ٠.٩
            }
          }
        ]
      }
    Transform {
        translation ٠.٠ ٤.٦٥ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.٢ ٠.٣
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box {
              size ٠.٨ ٠.١ ٠.٨
            }
          }
        ]
      }

Transform {
        translation ٠.٠ ٥.٠٥ ٠.٠
        children [
          Shape {
            appearance Appearance {
              material USE MarbleMaterial
              texture USE MarbleImage
              textureTransform TextureTransform {
                translation ٠.١ ٠.٣
                rotation ٠.٥
                scale ١.٠ ٠.٢
              }
            }
            geometry Box { size ١.١ ٠.٣ ١.١ }
          }
        ]
      }
```

کد طراحی کف برای یک محیط باستانی:

```
#VRML V۲.۰ utf۸
DEF Entry Viewpoint {
   position ۰.۰ ۱.٦ ۱۰.۰
   description "Entry view"
}
NavigationInfo {
   type [ "WALK", "ANY" ]
   headlight FALSE
   speed ٤.۰
}
Background {
   skyColor [ ۰.۰ ۰.۰ ۰.۰ ]
}
PointLight {
   color ۱.۰ ۱.۰ ۱.۰
   ambientIntensity ۰.۰
   intensity ۱.۰
   location ۰.۰ ۰.٥ -٦.۰
}
PointLight {
   color ۱.۰ ۰.۸ ۰.٥
   ambientIntensity ۰.۰
   intensity ۱.۰
   location ۰.۰ ۰.٥ ۰.۰
}
PointLight {
   color ۰.۸ ۰.۰ ۰.۰
   ambientIntensity ۰.۰
   intensity ۱.۰
   location ۰.۰ ۰.٥ ٦.۰
}
Transform {
   translation -۳.٥ ۰.۰ -۱۲.۰
children Shape  {
      appearance Appearance {
         material Material { ambientIntensity ۰.٥ }
         texture ImageTexture { url "stone۲.jpg" }
         textureTransform TextureTransform { scale ٤.۰ ۱٥.۰ }
      }
      geometry ElevationGrid {
         xDimension ۸
         zDimension ۱٥
         xSpacing ۱.۰
         zSpacing ۲.۰
         height [
            ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
            ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
            ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
```

```
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
        ]
        solid FALSE
      }
    }
}
```



کد طراحی یک محیط باستانی:

```
#VRML V٢.٠ utf٨
DEF Entry Viewpoint {
   position ٠.٠ ١.٦ ١٠.٠
   description "Entry view"
}
NavigationInfo {
   type [ "WALK", "ANY" ]
   headlight FALSE
   speed ٤.٠
}
Background {
   skyColor [ ٠.٠ ٠.٠ ٠.٠ ]
}
PointLight {
   color ١.٠ ١.٠ ١.٠
   ambientIntensity ٠.٠
   intensity ١.٠
   location ٠.٠ ٠.٥ -٦.٠
}
PointLight {
   color ١.٠ ٠.٨ ٠.٥
   ambientIntensity ٠.٠
   intensity ١.٠
   location ٠.٠ ٠.٥ ٠.٠
}
PointLight {
   color ٠.٨ ٠.٠ ٠.٠
   ambientIntensity ٠.٠
   intensity ١.٠
   location ٠.٠ ٠.٥ ٦.٠
}
DEF ColumnRow Transform {
```

```
   translation -١.٥ ٠.٠ ٠.٠
   children [

            DEF Column Transform {
               scale ٠.٥ ٠.٧ ٠.٥
               children Inline { url "untitled٥.wrl" }
            }


      Transform { translation ٠.٠ ٠.٠ -٩.٠ children USE Column }
      Transform { translation ٠.٠ ٠.٠ -٦.٠ children USE Column }
      Transform { translation ٠.٠ ٠.٠ -٣.٠ children USE Column }
      Transform { translation ٠.٠ ٠.٠  ٣.٠ children USE Column }
      Transform { translation ٠.٠ ٠.٠  ٦.٠ children USE Column }
      Transform { translation ٠.٠ ٠.٠  ٩.٠ children USE Column }


                    Transform { translation ٣.٠ ٠.٠  ٩.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠ -٣.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠ -٦.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠  ٣.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠  ٦.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠ -٩.٠ children USE Column }
                    Transform { translation ٣.٠ ٠.٠  ٠.٠ children USE Column }
   ]
}
Transform {
   translation -٣.٥ ٠.٠ -١٢.٠
children Shape  {
      appearance Appearance {
         material Material { ambientIntensity ٠.٥ }
         texture ImageTexture { url "stone٢.jpg" }
         textureTransform TextureTransform { scale ٤.٠ ١٥.٠ }
      }
      geometry ElevationGrid {
         xDimension ٨
         zDimension ١٥
         xSpacing ١.٠
         zSpacing ٢.٠
         height [
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
            ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠ ٠.٠,
         ]
         solid FALSE
      }
   }

}
Transform {
   translation -٣.٥ ٣.٦ -١٢.٠

children Shape  {
      appearance Appearance {
         material Material { ambientIntensity ٠.٥ }
         texture ImageTexture { url "block٠١.gif" }
         textureTransform TextureTransform { scale ٤.٠ ١٥.٠ }
      }
      geometry ElevationGrid {
         xDimension ٨
         zDimension ١٥
         xSpacing ١.٠
         zSpacing ٢.٠
```

```
            height [
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
               ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰ ۰.۰,
            ]
            solid FALSE
         }
      }

}

Transform {
            translation ۰.۰ ۱.۵ -۹.۰
            children [
               Shape {
                  appearance Appearance {
                           texture ImageTexture { url "window.jpg" }

                  }
                  geometry Box { size ۱ ۱ ۰.۰۱ }
               }
            ]
         }
```

کد طراحی یک باغچه:

```
#VRML V۲.۰ utf۸
Transform {
   translation ۰ ۰ ۰
children Shape  {
      appearance Appearance {
         material Material {
                              }
         texture ImageTexture { url "classic_red.gif"  }
         textureTransform TextureTransform {  scale ۲ ۱
                              }
      }
      geometry Box {
```

```
                        size ١ .٥ .١
                        }
    }

}
Transform {
   translation ٠ ٠ .٩

children Shape  {
       appearance Appearance {
          material Material {

                          }
          texture ImageTexture { url "classic_red.gif"  }
          textureTransform TextureTransform {  scale ٢ ١
                          }
       }
       geometry Box {
                        size ١ .٥ .١
                        }
    }

}
Transform {
   translation .٥ ٠ .٤٥

children Shape  {
       appearance Appearance {
          material Material {


                          }
          texture ImageTexture { url "classic_red.gif"  }
          textureTransform TextureTransform {  scale ٢ ١
                          }
       }
       geometry Box {
                        size .١ .٥ ١
                        }
    }

}
Transform {
   translation -.٥ ٠ .٤٥

children Shape  {
       appearance Appearance {
          material Material {


                          }
          texture ImageTexture { url "classic_red.gif"  }
          textureTransform TextureTransform {  scale ٢ ١
                          }
       }
       geometry Box {
                        size .١ .٥ ١
                        }
    }

}
Transform {
   translation ٠ ٠ .٠٣ .٤٥

children Shape  {
       appearance Appearance {
          material Material {


                          }
          texture ImageTexture { url "turf.gif"  }
          textureTransform TextureTransform {  scale ٢ ١
                          }
       }
       geometry Box {
```

```
                        size ١ .٥ .٨
                        }
    }

}
Transform {
    translation  -.١ ١.٣ ٠.٥
    children [
            Billboard {

                children [
                        Shape {
                            appearance Appearance {
                                    # No material, use emissive texturing
                                    texture ImageTexture { url "tree١.png" }
                            }
                            geometry Box {size  ٢ ٢ .٠٠١}

                            }

                ]
            }
    ]
}
```

پروژه طراحی یک خیابان برای یک شهر مجازی مانند زیر: