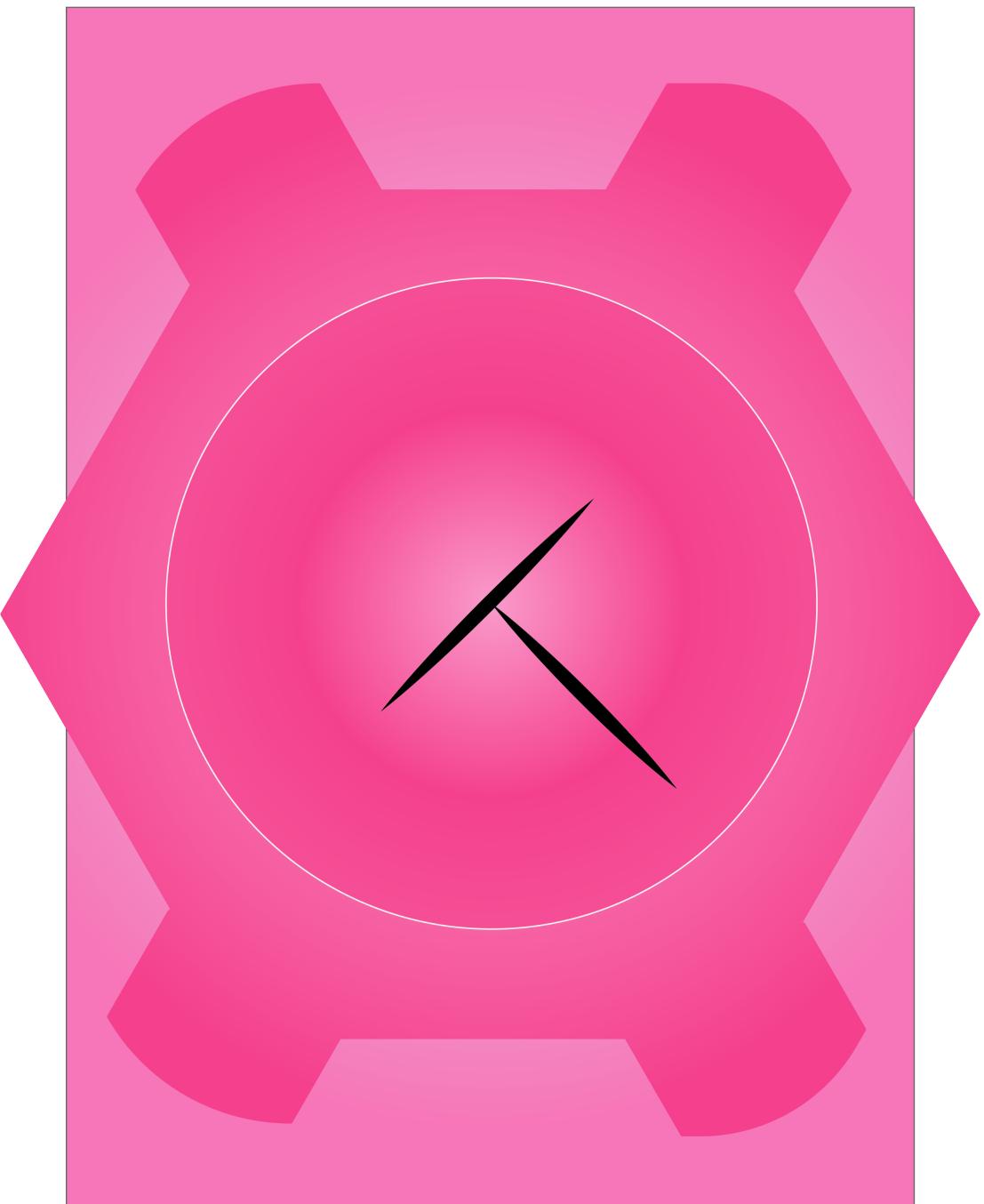


بازی سازی و برنامه نویسی گرافیکی پایه

چوبک بیدپا

•



عرضه شده تحت لیسانس MIT

نویسنده: چوبک بیدپا

سال عرضه: ۱۳۹۷

کاملا رایگان

جهت برقراری ارتباط با چوبک بیدپا از ایمیل
chubakbidpaa@riseup.net استفاده نمایید.

به خاطر اینکه بخش قابل توجهی ازین کتاب، از آموزش‌هایی که تحت لیسانس GPL و MIT، Creative Commons و آموزش این کتاب، به شرط نام بردن نویسنده یعنی شخص حقیقی چوبک بیدپا، آزاد می‌باشد.

توجه داشته باشید که فایل‌هایی که همراه کتاب به فروش گذاشته شده اند، غیرقابل تکثیر بوده، و آپلود آنها توسط شخص حقیقی چوبک بیدپا قابل قبول می‌باشد.

تحت قوانین Transference شما جهت استفاده از بخش‌های این کتاب که توسط افراد دیگر نگاشته شده اند احتیاجی به اجازه از آنها ندارید.

در آخر، قابل توجه باشد که این کتاب یک پروژه‌ی اشتیاقی^۱ می‌باشد، و نه پروژه‌ای که برای به دست آوردن پول نوشته شده است. برای همین، مرام را به جای آورید و آنرا در جای دیگر آپلود نکنید.

لطفا فایل‌هایی که همراه کتاب خریده اید نیز جایی کپی نکنید. قیمت فایل‌ها با توجه به استطاعت خوانندگان، با الگوریتمی پیچیده^۲ تعیین شده است. برای همین همه میتوانند آن را بخوردند. آپلود فایل‌های کتاب در جای دیگر، پایرسی حساب می‌شود و از لحاظ اخلاقی، کاریست نپسندیده.

اما تکثیر خود کتاب با ذکر منبع آزاد است.

Passion Project^۱

^۲ در ضمن، در صورت بیشتر شدن تعداد خریدها، قیمت کاهش می‌ابد.

قراردادهای کتاب

۱. بخش‌های کتاب: این کتاب به دو بخش برنامه نویسی گرافیکی، و بازی سازی تقسیم شده است.

۲. نکته: نکته‌های خاص کتاب در به این صورت مشخص شده اند:

نکته اینجاست که....

۳. استفاده از فایلها: اگر فایلی لازم باشد، نام و آدرس آن در فایل زیپ دانلود شده نوشته خواهد شد.

۴. یو آر الها به صورت <https://google.com> نوشته خواهند شد.

۵. متن پرنگ: وقتی لازم است روی کلمه‌ای تاکید کنم، یا کلمه جدید است و قبل از استفاده نشده، از متن پرنگ استفاده خواهم کرد.

۶. دکمه‌ها به صورت آیکون تعریف شده اند. یادتان باشد که دکمه‌ی نشانگر دکمه‌ی شیفت میباشد. دکمه‌ی بالا میباشد.

۷. کد: کدهایی لازمه به صورت زیر نوشته خواهند شد.

```
for i:=maxint to 0 do
begin
{ do nothing }
end;
Write('Case insensitive '');
```


خلاصه‌ی کتاب

سه فایل همراه این کتاب به فروش گذاشته شده است:

- **codes.zip**: این فایل، حاوی کدهایی میباشد که از فصل ۴ ببعد شماره گذاری شده، و نوشته میشوند.
- **assets.zip**: این فایل حاوی فایل‌های لازمه برای ساخت بازی میباشد.
- **tutorials.zip**: این فایل حاوی آموزش‌های ویدئویی همراه کتاب میباشند.

فهرست مطالب

ح

فهرست مطالب

۱

۱ چند کلمه با خواننده

۳
۳
۴
۸
۱۱
۱۳
۱۴
۱۵

۲ نگاهی کوتاه به ریاضی لازمه
۱.۲ توابع
۲.۲ بردارها
۳.۲ مثلثات
۴.۲ ماتریسها
۵.۲ قائمیت در فضای سه بعدی
۶.۲ دنباله ها و سری ها
۷.۲ پایان فصل ریاضی

۱۷
۱۷
۱۹
۲۰
۲۰
۳۸
۵۴
۵۵

۳ نگاهی کوتاه به برنامه نویسی
۱.۳ برنامه نویسی فانکشنال
۲.۳ برنامه نویسی شیء گرا
۳.۳ کتابخانه ها
۴.۳ پایتان
۵.۳ C++
۶.۳ سیستمهای کنترل نسخه
۷.۳ پایان فصل برنامه نویسی

۵۷
۵۷
۶۲

۴ مفهومات پایه‌ی گرافیک
۱.۴ مانیتورهای کامپیوتر
۲.۴ صفحه‌ی شترنجی و المان تصویری (پیکسل)

ح

فهرست مطالب

٦٤	رسترسازی	٣.٤
٦٦	سایه زنهای ترکشی	٤.٤
٦٨	رابطهای گرافیکی	٥.٤
٧٠	فریم بافر و وی رم	٦.٤
٧١	پایان فصل مفهومات گرافیکی	٧.٤
٧٣	الگوریتمهای رنگ، رسم و منحنیها	
٧٣	رنگ در کامپیووتر	١.٥
٧٧	کتابخانه‌ی تصویری پایتان	٢.٥
٧٩	خط راست	٣.٥
٨٦	ترسیم دایره	٤.٥

فصل ۱

چند کلمه با خواننده

«آزادی خود را گرامی بدارید، و گرنه آنرا از دست میدهید.» امروزه، جبهه های مختلفی هستند که بر آزادی اطلاعات عقیده دارند. یکی از آنها نهاد گنو^۱ است که کرنل سیستم عامل لینکس^۲ را در دست دارد. دیگری مازیلاست^۳، که مرورگر فایرفاکس^۴ را منتشر کرده است.

من به شخصه معتقدم آزادی اطلاعات از آزادی بیان مهمتر است، چون اگر اطلاعات را برای خود نگه داریم، کمتر کسی راههای اشاعه‌ی آزادی بیان را یاد خواهد گرفت، یا اصلاً خواهد دانست که آزادی بیان چه هست.

این کتاب نه تنها بر پایه‌ی عقیده به آزادی اطلاعات^۵ مجازی است، بلکه یکی از دلایل مجازی بودن آن اینست که تمام آن مال من نیست، بلکه، حدود ۳۰٪ این کتاب، ترجمه‌ی آموزش‌های اینترنت، با اجازه از صاحبان آنهاست. ۲۰٪ این کتاب، از داکیومنتشن‌های رسمی برداشته شده و ۵۰٪ درصد باقی را خودم نوشته‌ام.

شاید برایتان سوال باشد چرا این کتاب را نگاشت کرده‌ام. دلیل اصلی آن اینست که دلیلی داشته باشم تا برنامه نویسی را ادامه دهم. بعضی‌ها پروژه مینویسند، بعضی‌ها کتاب مینویسند. من در لفافه‌ی کتاب، پروژه مینوسم. تمام پروژه‌های کتاب اریجینال بوده، و فایلهایی که همراه کتاب خریده‌اید، کار من هستند.

دلیل دیگری که این کتاب را نوشته‌ام، اینست که کتاب‌های بازی سازی به زبان فارسی کم هستند، و کمتر کسی در ایران از برنامه نویسی گرافیکی به صورت حرفه‌ای و پولساز خبر دارد. سعی من درینست که با نوشتمن در مورد این دو دیسیپلین

GNU^۶

Linux^۷

Mozilla^۸

Firefox^۹

Freedom of Information^{۱۰}

فصل ۱. چند کلمه با خواننده

دوست داشتنی، فرهنگ آنها را در کشور اشاعه بدهم. از سابقه ام در برنامه نویسی و بازی سازی بگویم. من از شانزده سالگی^۶ کم و بیش در برنامه نویسی، و گهگاهی ساخت بازی، فعال بوده ام. مانند خیلی ها از نرم افزار Game Maker کارم را شروع کردم و با آن چندین بازی مانند تتریس، بریک اوت و... ساختم. من چندین بازی تحت اسکریپت نویسی نیز نوشته ام. من زبانهای C++، پایتان، و C را میدانم و با زبان اسکریپت نویسی چندین نرم افزار آشنایی دارم. به علوم نرم مانند ادبیات انگلیسی آشنایی آکادمیک دارم و در حال حاضر دانشجوی برنامه نویسی ام.

سابقه‌ی من در برنامه نویسی گرافیکی کمتر است. دو سال پیش بود که با نرم افزار افتر افکتس^۷ آشنا شدم و به صرافت نوشتتن پلاگین برایش افتادم، و طی این امر، با کتابخانه‌ی Cinder برای C++ آشنا شدم. و از آنجا بود که با زبان Processing و شیدرها آشنا گردیدم. الان تسلط کافی برای آموزش پایه‌ی شیدرها و زبانها و کتابخانه‌های برنامه نویسی گرافیکی دارم.

بگذارید در مورد چارچوب کتاب کمی صحبت کنم. در این کتاب، دو بخش داریم، برنامه نویسی گرافیکی، و بازی سازی که به دو بخش Asset و برنامه نویسی تقسیم میشود. در بخش آسیت سعی شده با استفاده از برنامه‌های مختلف، ساخت اسپرایت، تایل، اسپرایت شیت، تایل شیت، عکس پس زمینه، مدل سازی سه بعدی، و تکسچر و متریال را آموزش دهم. در بخش برنامه نویسی کتابخانه‌ی Arcade پایتان، کتابخانه‌ی SFML سی پلاس پلاس، و انجین Godot^۸ آموزش داده خواهد شد. اگر فرصت شد، آموزشی کوتاه برای ساخت انجین خودتان را خواهم نوشت.

قبل از هرچیزی دو چیز باید یادآوری شود: برنامه نویسی، و ریاضی. من زیاد در مورد این دو کانسپت حرف نمیزنم، چون وظیفه‌ی خود خواننده است که این دو را از قبل یاد داشته باشد، اما فقط در حد یادآوری، در مورد این دو حرف خواهم زد.

در آخر، در این دنیای پر هیر و گیر، اگر عشقی^۹ به چیزی دارید که به شما آرامش میدهد، نیکوست. و اگر این کتاب برای پیدا کردن این عشق کمک میکند، خوشحالم.

چوبک بیدپا مشهد - ۱۳۹۷

^۶الان بیست و پنج ساله ام.

^۷After Effects

^۸تلفظ این انجین، گدو میباشد.

^۹Passion

فصل ۲

نگاهی کوتاه به ریاضی لازمه

۱.۲ توابع

یک تابع^۱ به صورت زیر نشان داده میشود:

$$y = f(x)$$

وظیفه‌ی یک تابع، تغییر عدد داده شده بر اساس قوانین داده شده است. به این قانون، تابع میگوییم. مثلاً تابع $f(x) = x^2$ که به آن تابع مربع می‌گویند، وظیفه اش بردن عدد به توان دو است. به عکس تابع، تابع معکوس^۲ میگویند و به صورت زیر نشان داده میشود:

$$y = f^{-1}(x)$$

مثلاً معکوس تابع مربع، تابع ریشه دو یعنی $f(x) = \sqrt{x}$ میباشد. میتوان دو تابع را با هم به صورت $(f \circ g)(x)$ ترکیب کرد که به آن تابع مرکب میگویند. از دیگر عملیاتها عبارت است از:

$$(f + g)(x) = f(x) + g(x)$$

$$(f - g)(x) = f(x) - g(x)$$

$$(f \times g)(x) = f(x) \times g(x)$$

Function^۱
Inverse^۲

فصل ۲. نگاهی کوتاه به ریاضی لازمه

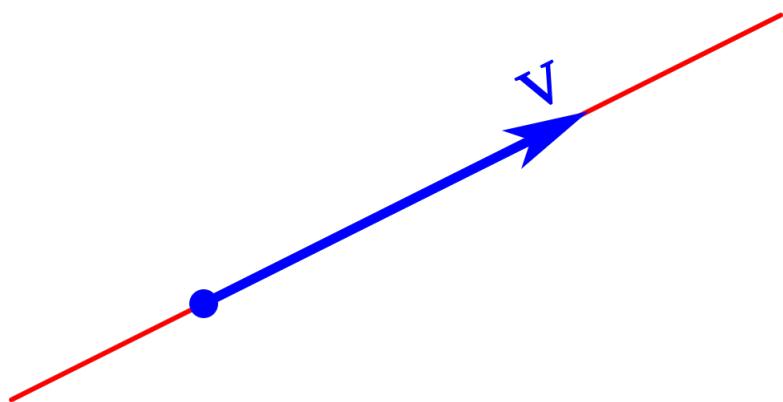
$$\left(\frac{f}{g}\right)(x) = \frac{f(x)}{g(x)}$$

به تمام اعدادی که تابع میپذیرد، **دامنه**^۳، و تمام اعدادی که تابع خارج میکند، **برد**^۴ خوانده میشود. دامنه‌ی یک تابع را ما تعیین میکنیم، اما برد آن را خود تابع تعیین میکند.

در آخر، بگذارید بگوییم که تابع مانند یک ماشین است. ورودی آن x و خروجی آن $f(x)$ است. در برنامه نویسی از توابع استفاده‌ی زیادی میشود. در بخش برنامه نویسی خواهید خواند.

۲.۲ بردارها

اگر فضای دو بعدی را به دو بخش نقاط افقی و نقاط عمودی تقسیم کنیم، **بردار**^۵ خطی است که ۲ نقطه را به هم وصل میکند.

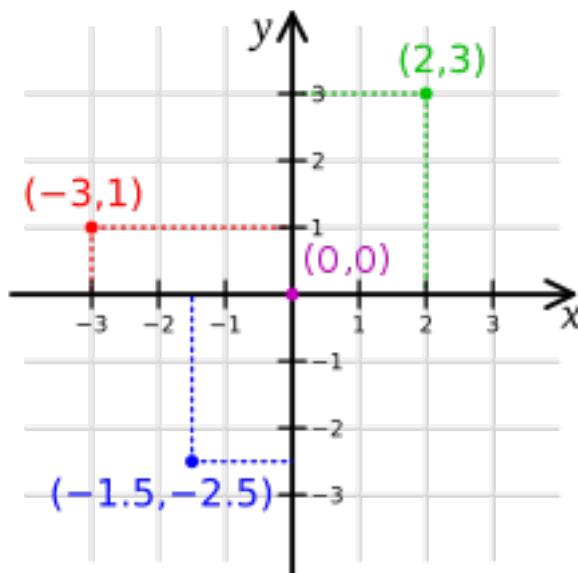


بردار را به صورت زیر نشان میدهند:

$$\vec{V} = (x_2, y_2) - (x_1, y_1)$$

صفحه‌ی مختصات را به این صورت میکشیم:

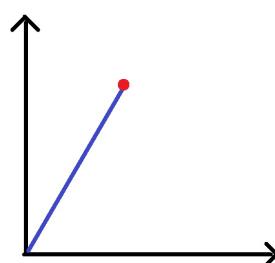
Domain^۳
Range^۴
Vector^۵



که به آن **دستگاه مختصات دکارتی**^۶ میگویند. در دستگاه مختصات دکارتی، دو محور X و Y به ترتیب محور افقی و عمودی ما را تشکیل می‌دهند. ما مختصات یک نقطه را در پرانتز به صورت (X, Y) نشان میدهیم. همانطور که گفته شد، خطی که دو نقطه را به هم وصل کند، بردار نام دارد.
در جبر خطی، بردار به صورت:

$$\vec{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$$

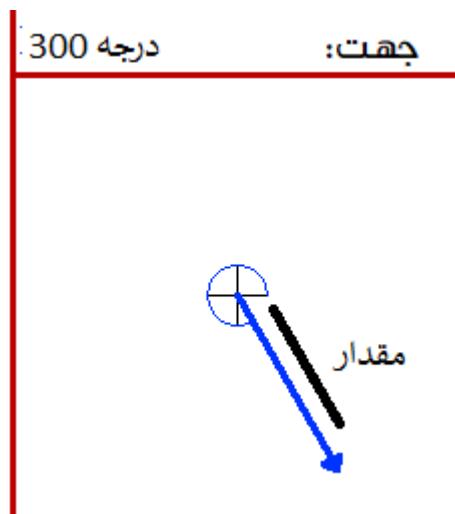
نشان داده میشود و نقطه‌ی اول آن، مسقط الرأس دستگاه مختصات یعنی $(0, 0)$ میباشد.



فصل ۲. نگاهی کوتاه به ریاضی لازمه

به برداری که مختصات افقی، یا عمودی آن، یک باشد بردار واحد میگویند. بردار واحد افقی را \vec{i} و بردار افقی عمودی را \vec{j} میگویند. بردارها را میتوان به صورت مضربی از بردار واحد نشان داد مثلا بردار $\vec{V} = \begin{pmatrix} X \\ Y \end{pmatrix}$ را میتوان به صورت $X\vec{i} + Y\vec{j}$ نشان داد.

یک بردار دارای دو خصیصه می باشد. **جهت**^۷ و **مقدار**^۸. که به صورت زیر نشان داده میشود:

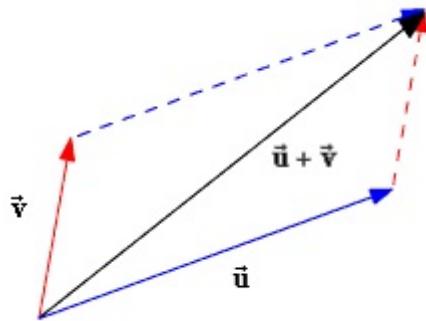


برای به دست آوردن مقدار بردار ازین فرمول استفاده میکنیم:

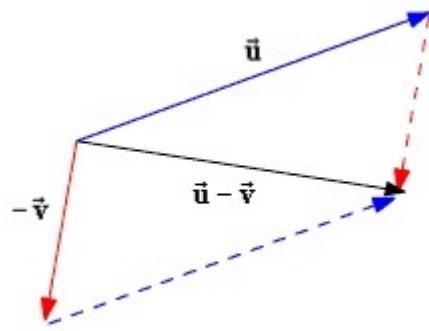
$$|\vec{V}| = \sqrt{x^2 + y^2}$$

دو بردار را میتوان به صورت زیر جمع کرد:

Direction^۷
Magnitude^۸

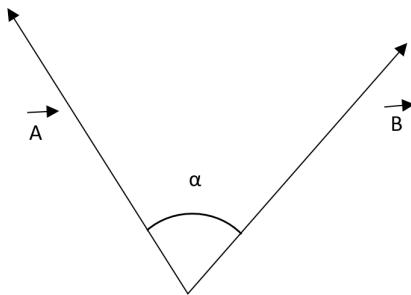


و به این صورت تفريقي کرد:



اما دو نوع ضرب برداری داریم. ضرب نقطه ای^۹ و ضرب صلیبی^{۱۰}. قبل ازین که پیش بروید، قسمت مثلثات ۲.۳ را بخوانید. فرض کنید دو بردار به صورت زیر هستند:

Dot Product^۹
Cross Product^{۱۰}



ضرب نقطه‌ای به صورت زیر تعریف می‌شود:

$$\vec{A} \cdot \vec{B} = |A||B| \cos \alpha$$

و ضرب صلیبی ازین فرمول استفاده می‌کنیم.

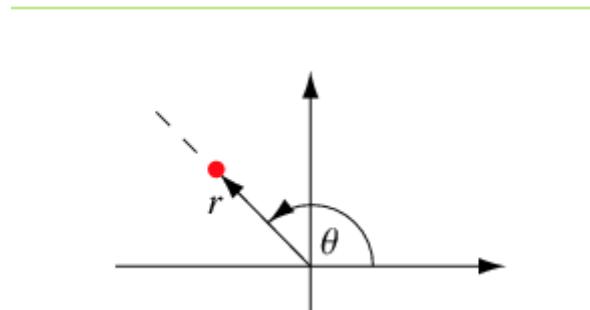
$$\vec{A} \times \vec{B} = |A||B| \sin \alpha \vec{n}$$

که \vec{n} برداری پایه^{۱۱} عمود بر دو بردار است. برای به دست آوردن \vec{n} کافیست از انگشتان وسط، اشاره، و شست خود استفاده کنید. انگشت شست شما، همواره بردار پایه‌ی عمود است، که مضربی از ضرب صلیبی دو بردار می‌باشد.

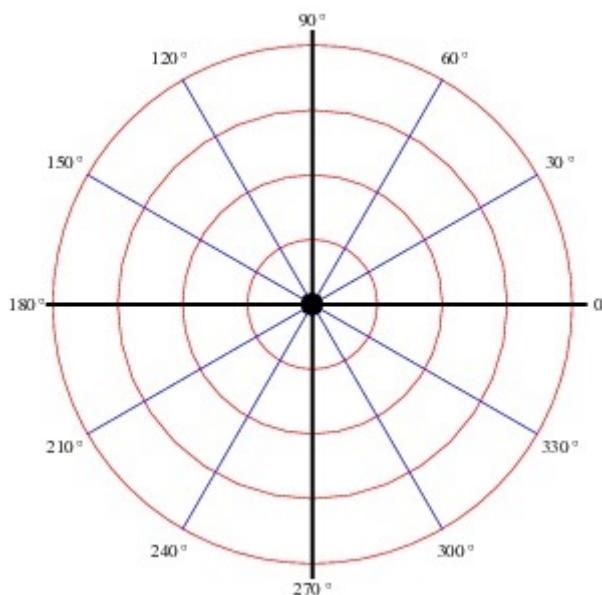
۳.۲ مثلثات

مثلثات بحییست پیپیده. و من نیز ریاضیدان نیستم پس به کمی در مورد این مبحث قناعت می‌کنیم. قبل از هرچیزی، بگذارید در مورد دستگاه مختصات قطبی^{۱۲} حرف بزنم. دستگاه مختصات قطبی، مانند دستگاه مختصات دکارتی، دارای دو محور عمودی و افقی است. اما در این دستگاه مختصات، ما یک نقطه را، عوض X و Y توسط یک زاویه α و یک بردار شعاع \vec{r} نشان میدهیم:

Basis Vector^{۱۱}
Polar Coordinate System^{۱۲}



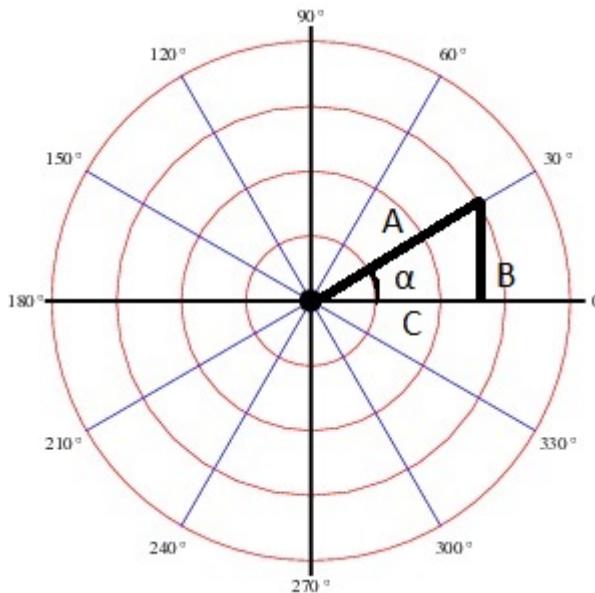
در برنامه نویسی گرافیکی، دستگاه مختصات قطبی کاربردهای زیادی دارد. اما در کامپیوتر، پیکسلها^{۱۳} در دستگاه مختصات دکارتی قرار دارند. حل مشکلات ما، مثلثات است. یک دایرهٔ واحد را در دستگاه مختصات قطبی کنید که شعاعش ۱ میباشد:



اگر زاویهٔ 30° را انتخاب کرده و یک مثلث قائم الزاویه دور آن بکشیم:

^{۱۳}در مورد پیکسلها به وفور حرف خواهم زد.

فصل ۲. نگاهی کوتاه به ریاضی لازمه



سینوس و کسینوس زاویه 30° درجه که اینجا α نامیده میشود، به صورت زیر تعریف میگردد:

$$\sin \alpha = \frac{\text{مجاور}}{\text{وتر}}$$

و:

$$\cos \alpha = \frac{\text{مجاور}}{\text{وتر}}$$

همچنین تانژانت و کتانژانت به صورت زیر تعریف میشوند:

$$\tan \alpha = \frac{\sin \alpha}{\cos \alpha}$$

۹

$$\cot \alpha = \frac{\cos \alpha}{\sin \alpha}$$

درجه، تنها واحد اندازه گیری زاویه نیست. واحد دیگر، رادیان^{۱۴} نام دارد. یک زاویه در رادیان، بین 0 و 2π قرار دارد. ارزش π حدود 3.1415926535 است. ما برای کار با پیکسلها، ارقام اعشار بیشتر ازین نیز نیازمندیم. برای تبدیل درجه به رادیان:

$$n^\circ \times \frac{\pi}{180}$$

Radians^{۱۴}

و بالعکس:

$$n\text{rad} \times \frac{180}{\pi}$$

تواجع مثلثاتی توسط **هویتهای مثلثاتی**^{۱۵} به هم ربط داده میشوند. بعضی ازین هویتها عبارتند از:

$$\sin^2 \alpha + \cos^2 \alpha = 1$$

$$\sin(-\alpha) = -\sin \alpha$$

$$\cos(-\alpha) = \cos(\alpha)$$

$$\sin(\alpha \pm \beta) = \sin \alpha \cos \beta \pm \cos \alpha \sin \beta$$

$$\cos(\alpha \pm \beta) = \cos \alpha \cos \beta \pm \sin \alpha \sin \beta$$

اینها تقریبا تمام سرفصلهایی هستند که شما برای این کتاب لازم دارید. توجه کنید، این کتاب، نه برنامه نویسی گرافیکی و بازی سازی کلی.

۴.۲ ماتریسها

به آرایه هایی از اعداد که به صورت n سطر و m ستون به نمایش در می آیند، **ماتریس**^{۱۶} میگویند. یک ماتریس را به این صورت نشان میدهند:

$$A_{m,n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}$$

در ساخت بازیهای کامپیوتری و برنامه نویسی گرافیکی ما بیشتر نیاز به ماتریسهای 2×2 ، 3×3 و 4×4 داریم. جمع و تفریق ماتریسها به صورت همسان انجام میشود:

$$A_{m,n} \pm B_{m,n} = \begin{bmatrix} a_{1,1} \pm b_{1,1} & a_{1,2} \pm b_{1,2} & \cdots & a_{1,n} \pm b_{1,n} \\ a_{2,1} \pm b_{2,1} & a_{2,2} \pm b_{2,2} & \cdots & a_{2,n} \pm b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} \pm b_{m,1} & a_{m,2} \pm b_{m,2} & \cdots & a_{m,n} \pm b_{m,n} \end{bmatrix}$$

ضرب ماتریسها به این روش صورت می‌پذیرد که، هر سطر با یک ستون. پس تا سطرها و ستونهای دو ماتریس با هم مساوی نباشند، ضرب صورت نمی‌پذیرد. مثلاً ضرب دو ماتریس 2×2 به صورت زیر است:

$$A_{2,2}B_{2,2} = \begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} & a_{1,1}b_{2,1} + a_{1,2}b_{2,2} \\ a_{2,1}b_{1,1} + a_{2,2}b_{2,1} & a_{2,1}b_{2,1} + a_{2,2}b_{2,2} \end{bmatrix}$$

یکی دیگر از عملیتهای ماتریسی، **دترمینان**^{۱۷} است. برای احتساب دترمینان ماتریس‌های بزرگتر از 3×3 الگوریتمهای زیادی مانند **دیکامپوزیشن**^{۱۸} وجود دارد که خود آن توسط افراد مختلفی در طول سالها بهسازی گشته است، اما راه ساده‌ای برای به دست آوردن دترمینان 2×2 وجود دارد که به شرح زیر است:

$$A = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$|A| = AD - BC$$

به I_n ماتریس **هویت**^{۱۹} می‌گویند و مثلاً I_3 به صورت زیر تعری می‌شود:

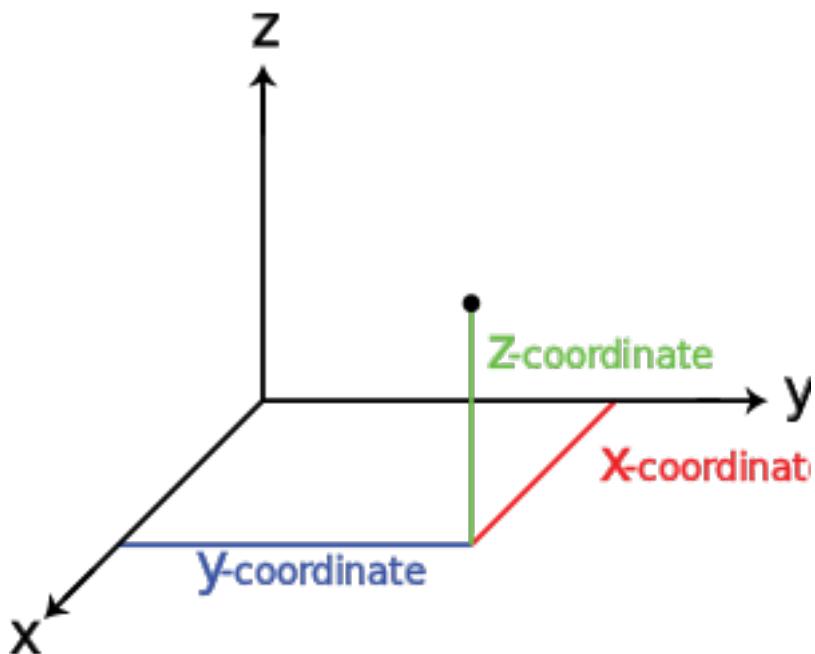
$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ما در برنامه نویسی گرافیکی و بازی سازی از ماتریسها استفاده های زیادی خواهیم برد.

Determinant^{۱۷}
Decomposition^{۱۸}
Identity^{۱۹}

۵.۲ قائمیت در فضای سه بعدی

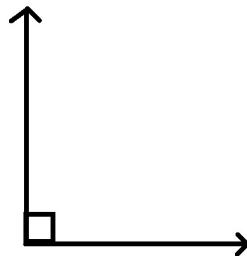
ما در بخش بردار دیدیم که دستگاه مختصات دکارتی شامل دو محور عمودی و افقی است. اما همیشه اینگونه نیست، بلکه، میتوان با اضافه کردن یک بردار اضافه که نام آن \vec{Z} است به دستگاه سه بعدی دست پیدا کنیم. این دستگاه را به صورت \mathbb{R}^3 نشان میدهند و در تصویر زیر میتوانید محور \vec{Z} را مشاهده کنید:



توجه کنید که در بعضی از نرم افزارها جای Y با Z عوض میشود. یک بردار را در فضای \mathbb{R}^3 به صورت زیر نشان میدهیم:

$$\vec{V} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

همه ای قوانین \mathbb{R}^2 برای \mathbb{R}^3 برقرار است. مثلاً به بردار واحد محور Z ، \vec{k} میگویند. غرض از این بخش، اینست که **قائمیت**^۲ در فضای سه بعدی را معرفی کنم. زیرا برای بازیهای دو و نیم بعدی، دوربین باید قائم بر فضای \mathbb{R}^3 باشد.



در کل، دو بردار وقتی بر هم قائمند که حاصلضرب نقطه ای آندو، صفر باشد:

$$\vec{A} \cdot \vec{B} = 0 \text{ اگر و تنها اگر } \vec{A} \text{ و } \vec{B} \text{ قائم است.}$$

اگر فرمول ضرب نقطه ای یادتان باشد، و در اینترنت کسینوس ۹۰ را خوانده باشید، میدانید که:

$$\vec{A} \cdot \vec{B} = |A||B| \cos \alpha \quad \text{و } \cos(90) = 0 \text{ پس:}$$

$$|A||B| \cos(90) = 0$$

۶.۲ دنباله ها و سری ها

دنباله^{۲۱} تابعیست به مانند زیر:

$$f : \mathbb{N} \longrightarrow \mathbf{A}$$

که دامنه‌ی آن، اعداد طبیعی است. یک عضو در یک دنباله میتواند بینهایت تکرار شود. مثلًا دنباله‌ی زیر:

$$\frac{1}{2}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{n}{n+1},$$

یک دنباله‌ی نامتناهی^{۲۲} کراندار^{۲۳} میباشد. اگر بر دنباله، به عدد خاص نزدیک شود، کراندار و در در صورتی که به عددی خاص نزدیک نشود، بیکران است. عضو

Sequence^{۲۱}
Infinite^{۲۲}
Convergent^{۲۳}

۷ از دنباله‌ی A را a_n تعریف می‌کنیم و به آن **ایندکس^{۲۴}** n می‌گوییم. دنباله‌ی **فیبوناچی^{۲۵}** را به صورت زیر تعریف می‌کنیم:

$$1, 1, 2, 3, 4, 8, 13, 21, F_{n-1} + F_{n-2},$$

که یک دنباله‌ی نامتناهی بیکران است.

سری^{۲۶} توضیحاتی است که تعیین کننده‌ی بعلاوه کردن بینهایت یک سری اعداد است. برای نشان دادن سریها، از علامت جمع یعنی **سیگما^{۲۷}** استفاده می‌کنیم:

$$\sum_{n=1}^{\infty} 2^n$$

این جمله به ما می‌گوید: اول، ۷ را ۱ بگیر. بعد برای ۱ تا بینهایت، ۲ را به توان n برسان. حتما لازم نیست تا بینهایت باشد، یا اینکه از ۱ شروع بشود.

$$\sum_{n=3}^8 n = 3 + 4 + 5 + 6 + 7 + 8$$

علامت ∞ بینهایت نام دارد. بینهایت، عددی آنقدر بزرگ است که نمی‌توانیم آنرا بشماریم.

۷.۲ پایان فصل ریاضی

کلید یادگرفتن ریاضی یک چیز است: n تمرین! یادتان نرود که حافظه، چیزیست فرآر، و هر لحظه ممکن است همین چیزهای کمی که از بندۀی حقیر آموخته اید، که مطمئنم برای بیشتر شما یک یادآوری ساده و کوتاه بوده، و برای خیل عظیمی از شما فوت آب بوده، و فقط یک لیست است، سریع از حافظه‌ی شما رخت بر می‌بندند. تمرین کنید، نوت برداری کنید، و یادتان نرود که روم را در یک روز نساخته‌اند. این ضرب المثال را چندین بار در طول کتاب تکرار خواهم کرد. یادگیری طول می‌کشد. و یادتان نرود که هیچکس استعداد چیزی را ندارد، و همه چیز با تمرین میسر می‌شود. در فصل بعد، در مورد برنامه نویسی، زبان پایتان و C++ حرف خواهیم زد.

فصل ۳

نگاهی کوتاه به برنامه نویسی

درین فصل نگاهی کوتاه می اندازیم به برنامه نویسی^۱. ابتدا به دو پردایم^۲ برنامه نویسی فانکشنال^۳ و شیء گرا^۴. بعد از آن، نگاهی می اندازیم به سینتکس^۵ زبان پایتان^۶ و C++^۷.

۱.۳ برنامه نویسی فانکشنال

از بین تمام روشها، یا به عبارتی، پردایمهای برنامه نویسی، برنامه نویسی فانکشنال یا تابعی ساده ترین، و پر مصرف ترین آنهاست. اکثر اشخاصی که برنامه نویسی را شروع میکنند، از برنامه نویسی فانکشنال شروع میکنند. زبانهای قدیمی مانند فورترن^۸ و لیسب^۹ همه فانکشنال هستند.

با توابع در فصل ریاضی آشنا شدیم. توابع کامپیوتری نیز با توابع ریاضی فرق زیادی ندارد، همه‌ی آنها یک ماشین هستند که ورودی را به خروجی تبدیل میکنند. یک تابع، مجموعه‌ای از دستورات^{۱۰} است که پارامتر^{۱۱} داده شده را با تغییرات، باز میگردانند. این تغییرات میتواند عملیاتهای جمع و تفریق، ضرب و تقسیم، باقیمانده،

Programming^۱
Paradigm^۲
Functional^۳
Object Oriented Programming^۴
به دستور زبانی یک زبان برنامه نویسی Syntax گفته میشود.^۵
Python^۶
FORTRAN^۷
Lisp^۸
Instructions^۹
Parameter^{۱۰}.

فصل ۳. نگاهی کوتاه به برنامه نویسی

و یا تغییر نوع پارامتر مثلا از عدد صحیح به عدد حقیقی، و یا هرچیز دیگری باشد. برای اجرای تابع، آنرا **میخوانیم**^{۱۱} و به پارامتری که به آن میدهیم، **آرگومان**^{۱۲} میگوییم. اینستراکشن سِت زیر را در نظر بگیرید:

۱. عدد n را بگیر.
۲. برای n بار، n را ضربدر $1 - n$ کن.
۳. جواب را برگردان.

به این تابع، تابع **فاکتوریل**^{۱۳} میگویند. توابع زیادی هستند، پیچیده و ساده، مهم اینجاست که از آنها درست استفاده کنید. بعضی از توابع، پارامتر قبول نمیکنند. بعضی از توابع، ارزشی را باز نمی‌گردانند. به این نوع از توابع **ووید**^{۱۴} میگویند. بعضی از زبانها، **تایپ ثابت**^{۱۵} هستند و باید نوع ارزش‌های باز گرداننده را مشخص کرد. C++ یکی ازین نوع زبانهاست. بعضی از زبانها **تایپ دینامیک**^{۱۶} هستند و لازم نیست نوع ارزش بازگرداننده را در آنها مشخص کرد. پایتان یکی ازین زبانهاست. هردو زبان پایتان و C++ هم فانکشنال هستند، هم شیء گرا. در مورد پردایم شیء گرا در بخش بعد صحبت خواهیم کرد. هر زبان مقداری تابع از پیش شده دارد، اما بقیه‌ی تابع‌ها را خودتان باید تعیین کنید. اینکه در چه زمانی باید تابع تعیین کرد، قانون طلایی اینست که هرگاه دیدید عملی را دارید بیشتر از یک بار انجام میدهید، وقت تعیین کردن یک تابع است. همه‌ی زبانها دارای **کتابخانه**^{۱۷} هایی هستند که شامل توابع و کلاسها ۳.۲ و دیگر کدهایی هستند که به برنامه نویس کمک میکنند خود را تکرار نکند.

خود را تکرار نکنید.^{۱۸}

قانون پلاتینیوم برنامه نویسی، اینست. هرگز چیزی که در یک کتابخانه موجود است را ننویسید. مثلاً عوض اینکه در زبان **جاوا**^{۱۹} عوض نوشتن صدها خط کد برای به دست آوردن یک تابع ماتریس، میتوان از کتابخانه‌ی JAMA استفاده کرد.

Call ^{۱۱}
Argument ^{۱۲}
Factorial ^{۱۳}
Void ^{۱۴}
Statically Typed ^{۱۵}
Dynamically Typed ^{۱۶}
Library ^{۱۷}
DRY - Don't Repeat Yourself ^{۱۸}
Java ^{۱۹}

۲.۳ برنامه نویسی شیء گرا

برنامه نویسی شیء گرا بر پایه‌ی کانسپت کلاس^{۲۰} می‌جرخد.

گفتیم توابع مانند ماشینهایی هستند که اطلاعات را از حالتی به حالت دیگر تغییر میدهند. اگر تابع، ماشین است، کلاس، یک خیابان پر از ماشین است که در آن هزاران ماشین وجود دارد، و همچنین چندده هزار عابر پیاده که سوار ماشین می‌شوند. درین شبیه، به ماشین **اسلوب**^{۲۱} و به عابر پیاده **خواص**^{۲۲} می‌گویند. اگر ایده‌ی کلاس، یعنی یک خیابان پر از عابر پیاده و ماشین را داشته باشیم، با آن میتوانیم هزاران هزار خیابان بسازیم. به هر خیابانی که ما می‌سازیم، **شیء**^{۲۳} می‌گویند. مثلا خیابان ولی‌عصر، یک شیء خیابان است. در یک کتابخانه مانند کتابخانه‌ی JAMA که از آن نام برده‌یم، یک کلاس به نام ماتریس وجود دارد و این کلاس چندین اسلوب و چندین خواص دارد. یکی از آن اسلوبها، ارزش سطر و ستون داده شده را به کاربر برمی‌گردد.

بعضی از متدها و خواصها، خصوصی‌اند، یعنی جز سازنده‌ی کلاس خیابان، کسی اجازه‌ی عوض کردن آن را ندارد. اما بعضی از اسلوبها و متدها قابل تغییرند. یک کلاس میتواند فرزند یک کلاس دیگر باشد. در بعضی از زبانها، یک کلاس میتواند فرزند چندین کلاس باشد. C++ یکی ازین زبانهاست.

فرض کنیم یک کلاس داریم به نام مدرسه. برای ساختن یک شیء مدرسه از آن، باید به آن ارزش‌هایی مانند آدرس مدرسه، تعداد کلاسها، اینکه مدرسه بوفه داشته باشد یا نه، نام مدرسه، اینکه دبیرستان است یا ابتدایی، و... را بدهیم تا از آنها، خواص مدرسه را تعیین کند. این کار توسط اسلوب سازنده^{۲۴} انجام می‌شود. و یا به سادگی میخواهیم یک مدرسه‌ی قدیمی را بکوییم و خراب کنیم. برای این کار از اسلوبی به نام **خراب کننده**^{۲۵} استفاده می‌کنیم. یگ کلاس میتواند چنین سازنده داشته باشد، ولی فقط یک خراب کننده میتواند داشته باشد.

در بخش بعدی در مورد کتابخانه‌ها صحبت خواهیم کرد.

Classes ^{۲۰}
Method ^{۲۱}
Properties ^{۲۲}
Object ^{۲۳}
Constructor ^{۲۴}
Destructor ^{۲۵}

۳.۳ کتابخانه ها

به مجموعه توابع و کلاس‌های از قبل آماده شده، کتابخانه می‌گویند.

هر کدی که در صورت اجرا، عملیات خاصی انجام نداده، و به کدهای دیگر برای اجرا وابسته باشد، کتابخانه نام میگیرد. اکثر زبانها برای کتابخانه های خود دارای یک دیتابیس هستند، که زبان پایتان جزء آنهاست، اما بعضی از زبانها برای کتابخانه های خود دیتابیس ندارد، مانند C++. دلیل آن اینست که اکثر کتابخانه های C++، متن بسته و یولی هستند.

کتابخانه ها میتوانند به صورت فایل متند، یا فایل باینری^{۲۶} عرضه شوند. کتابخانه های پایتان متند، و کتابخانه های C++ باینری هستند. گاها کتابخانه هایی به صورت متند نیز عرضه میشوند.

پاپستان ۴۰۳

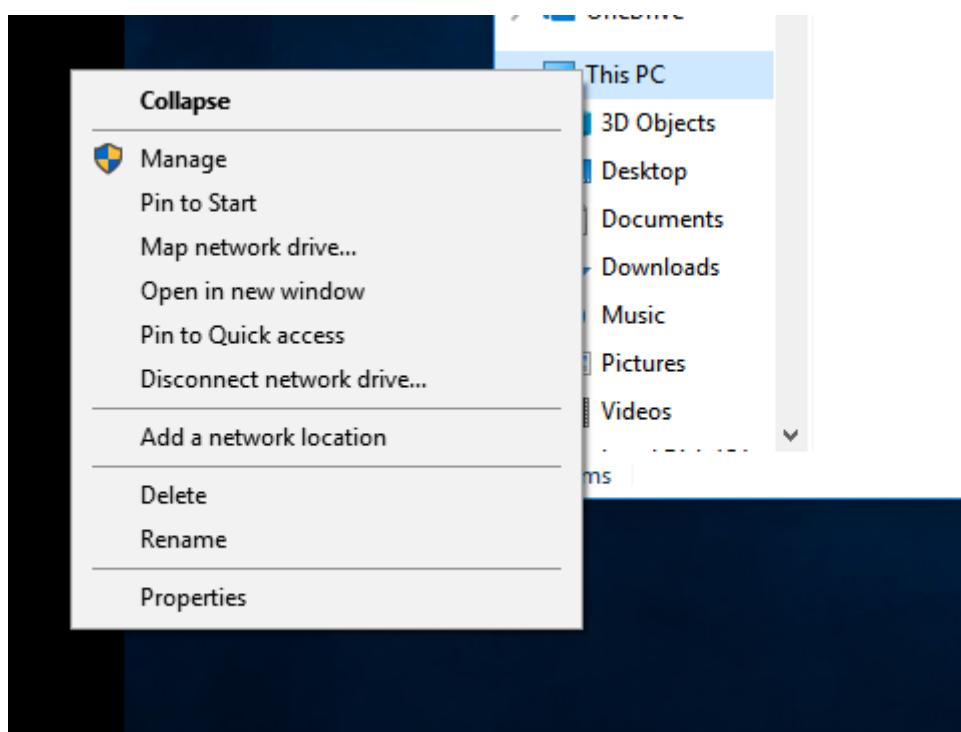
زبان پاپیтан در سال ۱۹۹۹ برای بار اول عرضه شد و در هنگام نوشتن این کتاب، در ۳.۷.۱ به سر میبرد. درین فصل، فقط قطره‌ای از دریای این زبان را آموزش میدهیم. برای آموزش بهتر زبان، به کتاب مخصوص مراجعه کنید.

پایتان زبانی کاملاً مدرن، قابل انعطاف، یکدست، و جذاب و ساده میباشد که برای از اتوماسیون گرفته تا بازی سازی، کاربرد دارد. اینستراکشن های پایتان **ترجمه**^{۲۷} میشوند، یعنی لازم نیست که از قبل به زبان اسمنبلی یا ماشین در بیایند، بلکه، در حین اجرا به زبانهایی مثل C یا Java ترجمه میشوند و بعد خط به خط اجرا میشوند. عرضه ی اصلی پایتان که ما از آن استفاده میکنیم، از C++ استفاده میکند.

نصب پاپتان

۱. به این صفحه بروید و پایتان را دانلود کنید: <https://www.python.org/downloads/>
 ۲. آنرا نصب کنید.
 ۳. از فایل اکسیلور مانند زیر روی Properties کلیک کنید:

Binary Interpret



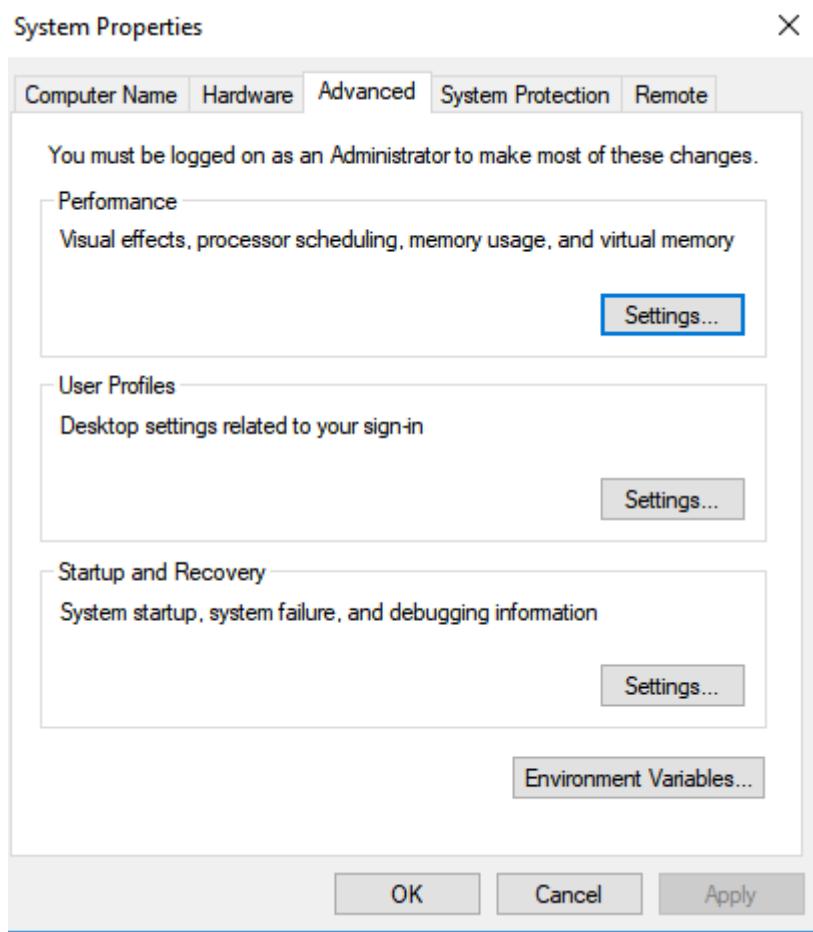
۴. روی Advanced System Settings کلیک کنید.

← → ⏪ ⏪ Control Panel

Control Panel Home

-  [Device Manager](#)
-  [Remote settings](#)
-  [System protection](#)
-  [Advanced system settings](#)

۵. روی گزینه Environment Variables کلیک کنید.



۶. روی Path کلیک کنید.

Variable	Value
OPENCV_DIR	C:\opencv\build\x64\vc15
OS	Windows_NT
Path	I:\AE SDK\Cuda\bin;I:\AE SDK\Cuda\libnvvp;C:\Program Files (x86)\...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 60 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6

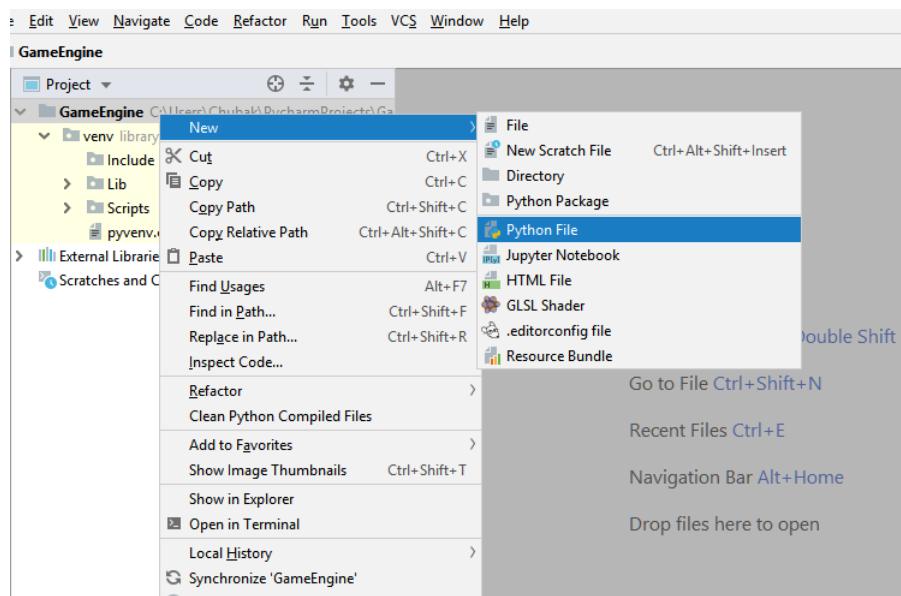
۷. دو گزینه‌ی زیر را به آن اضافه کنید.

C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
C:\Program Files\Microsoft SQL Server\130\Tools\Bin\
D:\Program Files\MiKTeX 2.9\miktex\bin\x64\
C:\Users\Chubak\AppData\Local\Programs\Python\Python37
C:\Users\Chubak\AppData\Local\Programs\Python\Python37\Scripts
C:\opencv\build\x64\vc15\bin
C:\Program Files\dotnet\
D:\SFML\thor-v2.0-msvc2015\bin
C:\Program Files\CMake\bin
D:\SFML\SFML\lib
D:\SFML\Thor\lib
D:\GLFW\lib

برای نوشتن کد پایتان احتیاج به یک **محیط گسترش مجتمع**^{۲۸} دارد. من PyCharm را پیشنهاد میکنم که کاملاً مجانیست. آنرا میتوانید از [Charm Community](https://www.jetbrains.com/pycharm/download/#section=windows) دانلود کنید.

فصل ۳. نگاهی کوتاه به برنامه نویسی

بعد از باز کردن پایی چارم و باز کردن یک پروژه‌ی جدید، از طریق زیر یک فایل پایتان بسازید:



حال وقت نوشتن اولین کد ماست. بنویسید:

```
new_variable = "A Portal to the World of Python"
print(new_variable)
```

را بزنید. در پایین صفحه، کد شما اجرا می‌شود.

میتوانید از محیط‌های گسترش دیگر نیز استفاده کنید. مانند PyDev، Beans IDE و در ضمن Cloud9 IDE Spyder، Microsoft Visual Studio دارد.

متغیرها

در بخش قبل، `new_variable` یک متغیر^{۲۹} است. ممتغیرها مانند ظروفی هستند که میتوان در آنها هر چیزی ریخت. چون پایтан زبان تایپ امن^{۳۰} نیست، میتوان هر نوع دیتایی را داخل یک متغیر جا داد.

Variable^{۲۹}
Type Safe^{۳۰}

```
var = True #boolean
var = "String" #string
var = 1 #integer
var = 1.0 #float
var = Class() #Class
```

همانطور که میبینید، متغیرهای پایتان هر نوع دیتایی^{۳۱} را قبول میکنند، استرینگ، عدد صحیح، فلوت و کلس. بگذارید تمام این انواع دیتا را توضیح دهم.

- به متغیرهایی که دو حالت راست^{۳۲} و غلط^{۳۳} دارند، متغیر بولی^{۳۴} میگویند.
- به تکه های متنی که از الفباء تشکیل شده، رشته یا String میگویند. به متنی که به یک استرینگ میدهیم، لیترال^{۳۵} میگویند.
- به اعداد صحیح بدون اعشار اینتجر^{۳۶} یا به کوتاه int میگویند.
- به عدد حقیقی با اعشار فلوت^{۳۷} میگویند
- از هر کلاسی می توان یک متغیر ساخت. درین صورت، نام کلاس، نوع متغیر میشود.

اینها فقط چند نوع از متغیرهای پایتان هستند. همانطور که گفته شد، لازم نیست نوع متغیر را مشخص کنید چون پایتان، تایپ امن نیست. اما لازم است برای تعیین نوع آن، آنرا مقداردهی اولیه^{۳۸} نمایید. اما اگر بخواهیم چندین نوع دیتا، یا متغیر، را در یک متغیر نگاه داریم چه؟ بخش ۴.۳ درین مورد صحبت خواهد کرد.

^{۳۱} دیتا تایپ
^{۳۲} True
^{۳۳} False
^{۳۴} Boolean
^{۳۵} Literal
^{۳۶} Integer
^{۳۷} Float
^{۳۸} Initialize

لیست، تاپل، دیکشنری

برای نگاه داشتن چندین نوع دیتا، یا چندین متغیر، در یک مکان، از **لیست**^{۳۹} استفاده میکنیم. لیستها به صورت زیر مقداردهی اولیه میشوند:

```
a_list = []
a_list = [1, 2, 3, 4]
a_list = ["Hello World", "Goodbye World"]
a_list = [1, 2, 3, 4, "Hello", "World"]
```

یک لیست وقتی مقداردهی اولیه شد، نباید با علامت مساوی، به آن مقدار اضافه کرد. بلکه، باید از `list.append` استفاده کرد:

```
a_list = [1]
print(a_list)
a_list.append(2)
print(a_list)
```

را بزنید تا نتیجه را ببینید. در زیر چند اسلوب لیست را میخوانید.

- `list.reverse`: لیست را برعکس یا به عیارت دیگر، معکوس میکند.
- `list.copy`: لیست را در لیست دیگر کپی میکند.
- `list.pop`: در ایندکس^{۴۰} داده شده، عضو را پاک میکند.
- `list.sort`: لیست را بر اساس الگوریتم **Merge Sort** مرتب میکند.
- `len(list)`: سایز لیست را به دست می اورد.

یک لیست، میتواند لیستهای دیگری نیز در بر بگیرد:

```
multi_dimensional_list = [[], [], []]
multi_dimensional_list.append([])
```

Lists^{۳۹}
- به شماره‌ی عضو ایندکس میگویند. Index^{۴۰}.

برای به دست آوردن عضو خاصی از لیست، از ایندکس آن استفاده میکنیم.

```
my_list = [2, 4, 6, 8, 10]
print(my_list[0])
```

ایندکسها از **۰** شروع میشوند تا سایز لیست منهای یک ادامه دارند. برای بدست آوردن چندین عضو از لیست، از علامت دو نقطه استفاده میکنیم:

```
my_list = [2, 4, 6, 8, 10]
print(my_list[0:3])
```

در کل، لیستها بهترین روش برای نگه داشتن دیتاهای زیاد هستند. اما برای دیتای کم و **غیر جهشی**^{۴۱}، از **تاپل**^{۴۲} استفاده میکنیم. تاپلها میتوانند به اندازه ی لیست دیتا نگه دارند، اما نمیتوان از آنها دیتا کم و زیاد کرد. تاپلها بدین صورت تعریف مقداردهی اولیه میشوند:

```
my_tuple = (R, G, B)
print(my_tuple[0:1])
```

مثلا برای رنگ یا موقعیت یک فرگمنت در یک تصویر رستر از تاپل استفاده میشود. مانند لیستها، میتوان از `len()` برای بدست آوردن سایز تاپل استفاده کرد. دیکشنریها نیز مانند لیستها، برای نگه داشتن مقدار زیادی دیتا استفاده میشود. اما در دیکشنری، ایندکسها، عوض شماره، دارای نام هستند. دیکشنریها همتا به **همتا**^{۴۳} هستند. برای مقداردهی اولیه ی یک دیکشنری، از تفريعی زیر استفاده میکنیم:

```
my_dict = {"Name" : "Chubak",
           "Last Name" : "Bidpaa"}
print(my_dict["Name"])
```

در تمام زبانهای برنامه نویسی، زدن  وسط خط کد اشکالی ندارد.

مهمترین اسلوب دیکشنری dictionary.items() میباشد که آیتم های دیکشنری را بر میگرداند. در بخش لوب در موردش صحبت خواهیم کرد صحبت از لوب شد، وقت آن است که در مورد **بیانیه های ^{۴۴}** پایتان صحبت کنیم.

بیانیه های شرطی

بیانیه های شرطی ^{۴۵}، بخشهایی از پایتان هستند که به کد اجازه ی اجرا، یا در صورت عدم اجازه، اجرای کد دیگری را میدهند.

کلمات کلیدی ^{۴۶} که ما برای شرط گذاشتن روی جریان اجرای برنامه استفاده میکنیم، **if** و **elif** و **else** هستند. همیشه لازم نیست از دو تای دوم استفاده کرد، اما پیشنهاد میشود اگر مستلزم است، حتما از آنها استفاده کنید. کد زیر را ببینید:

```
pi = 3.14
r = 10
area= pi*r*r

if area > 20:
    print("The area is greater than 20.")
else:
    print("The area is not greater than 20.")
```

این کد، مساحت یک دایره با شعاع ۱۰ را حساب میکند و اگر این مساحت، بیشتر از ۲۰ است، میگوید مساحت بیشتر از ۲۰ است، و گرنه میگوید مساحت بیشتر از ۲۰ نیست. به همین سادگی، ما **جریان ^{۴۷}** اجرای کد را تغییر دادیم.

پای چارم خودش اینکار را میکند، اما بین اول خط کلمه ی **if** و **else** و اصطلاح ^a

شرط، باید چهار فاصله باشد.Expression^a

برای شرط گذاشتن، از آپریتور^{۴۸} هایی مانند < استفاده میکنیم. به اعدادی که آپریتور روی آنها تاثیر میگذارد، آپرند^{۴۹} میگویند.
آپریتور های شرطی پایتان به شرح زیرند:

بعلاوه	+
منها	-
ضرب	*
تقسیم	/
باقیمانده	%
بزرگتر	<
کوچکتر	>
بزرگتر مساوی	=<
کوچکتر مساوی	=>
مساوی	==
نامساوی	!=
و	and
یا	or
نیست	!

جدول ۱.۳: آپریتورهای پایتان

آپریتورهای دیگری نیز داریم مانند آپریتورهای **Bitwise** ولی الان به کار ما نمی آیند.
میتوانید از کلمه‌ی کلیدی **if** **Else** که مخفف **elif** است برای افزایش شرط استفاده کنید:

```
if area > 20:
    print("The area is bigger than 20.")
```

Operator^{۴۸}
Operand^{۴۹}

فصل ۳. نگاهی کوتاه به برنامه نویسی

```
elif area < 15:
    print("The area is less than 15")
elif area < 10:
    print("The area is less than 10")
else:
    print("The area is not bigger than 20.")
```

اما این فقط تنها بیانیه‌ی شرطی پایتان نیست. دو بیانیه‌ی شرطی دیگر داریم، که با `if` فرق زیادی دارند.

شرط، میتواند یک متغیر بولی باشد. مثلا `bool = 1 < 10`. این متغیر، راست میباشد چون ۱ کوچکتر از ۱۰ است. (True)

بیانیه‌های چرخشی شرطی

بیانیه‌های چرخشی شرطی^۵ بیانیه‌هایی هستند که تا شرط برقرار است، یک اصطلاح یا بیانیه را به صورت نا محدود^۶ اجرا میکنند. گاهی این چرخش، بینهایت است. اما اکثر اوقات، شرط تمام شده و غلط (False) میشود. وقتی شرط، غلط میشود، چرخش تمام شده و بیانیه‌ی بعدی اجرا میشود. همچنین میتوانیم خودمان جریان چرخش را کنترل کرده، و به میل خود چرخش را تکرار کرده و یا بشکنیم. دو کلمه‌ی کلیدی برای اینکار استفاده میشود، `for` و `while`. اولی مصارف دیگری هم دارد که به آن میپردازیم. اما بگذارید اول به `while` بپردازیم. سینتکس آن اینگونه است:

```
i = 0

while i < 50:
    print(str(i))
    i += 1
```

ابتدا ما به متغیر `i` عدد ۰ را میدهیم. بعد میگوییم تا این متغیر، از ۵۰ کوچکتر از، ارزش متغیر را روی صفحه نمایش بده و در هر بازتکرار^۷، ارزش ۱ را به متغیر اضافه

میکنیم. وقتی متغیر به ۵۰ رسید، چرخش تمام میشود و به بیانیه‌ی بعدی میرسد.

تابع `print()` نمیتواند جز استرینگ لیترال و استرینگ، چیز دیگری را در صفحه به نمایش بگذارد. با استفاده از تابع `str()` متغیرهای عددی را به استرینگ لیترال تبدیل میکنیم.

میتوانید با استفاده از کلمه‌ی کلیدی `and` یک شرط دیگر اضافه کنید:

```
i = 0

while i < 50 and i < 25:
    print(str(i))
    i += 1
```

اینگونه، فقط در صورتی متغیر روی صفحه پرینت میشود که بین ۲۵ و ۵۰ باشد. `or` هم دو یا چند شرط را در صورتی اجرا میکند که یکی از آنها، راست باشد. کلمه‌ی کلیدی بعدی که داریم، `for` میباشد. این کلمه بیشتر برای دسترسی به لیست، دیکشنری و تاپل به کار میرود. اما برای چرخش برای `n` بار از سینتکس زیر استفاده میکنیم:

```
for i in range(n):
    print(str(i))
```

تابع `range(m, n)` یک لیست قابل بازتکرار بین `m` و `n` ایجاد میکند. اگر پارامتر اول را به آن ندهیم، یک لیست قابل بازتکرار بین ۰ و `n` ایجاد میکند. و متغیر ارزش آ را در هر بازتکرار، بر اساس لیست ساخته شده مشخص میکند. این بیانیه‌ی چرخشی شرطی نیست، بلکه بیانیه‌ی چرخشی بازتکراریست. در بخش بعد، از کلمه‌ی کلیدی `for` برای دسترسی به اعضای لیست، تاپل، و دیکشنری استفاده میکنیم.

دسترسی به لیست، دیکشنری و تاپل

برای دسترسی به اعضای یک لیست، تاپل، دیکشنری، یا هر شیء قابل بازتکرار^{۵۳} دیگری، از `for` استفاده میکنیم. مثال برای لیست اینگونه است:

فصل ۳. نگاهی کوتاه به برنامه نویسی

```
one_dim_list = [1, 2, 3]
two_dim_list = [[1, 2, 3], [4, 5, 6]]

for i in one_dim_list:
    print(i)

for list in two_dim_list:
    for i in list:
        print(i)
```

همانطور که مشاهده میکنید، ما با استفاده از کلمه `i` کلیدی `in` توانستیم به اعضای لیست یک بعدی `one_dim_list` دسترسی پیدا کنیم و آنها را روی صفحه پرینت کنیم. سپس، ما با استفاده از یک بیانیه `ای` ^{۵۴} توانستیم یک لیست دو بعدی را روی صفحه پرینت بگیریم.

تمام بیانیه های چرخشی را میتوان لانه کرد، اما اگر که اشتباهی صورت بگیرد، سریزی پشته^۵ صورت میپذیرد. پشته بخشی از زبان C است که کامپایلر پایتان در آن نوشته شده است. ۵.۳ را بخوانید.

Stack Overflow^a

ما میتوانیم با استفاده از دو کلمه `break` و `continue` بر جریان چرخشمان تاثیر بگذاریم.

```
n = 0
while True:
    n += 1

    if (n > 20):
        break;
```

Nested Statement^{۵۴}

این کد، همیشه صحیح است، پس همواره اجرا میشود. اما اگر متغیر ما، بیشتر از ۲۰ شود، زنجیر میشکند و چرخش پایان میابد. `continue` نیز مانند `break` است، فقط حلقه را نمیشکند، بلکه کاری میکند که حلقه دوباره بازتکرار شود.

تابع در پایتان

در بخش پردایم فانکشنال، در مورد توابع در برنامه نویسی صحبت کردیم. در پایتان، تابع بلوکه ای از کد است که با خواندن آن، یک یا یک امر صورت میپذیرد، یا یک ارزش باز گردانده میشود، یا هردو. پایتان دارای ۶۸ تابع از پیش ساخته شده است، و ما خودمان میتوانیم تا هرچقدر لازم داریم، تابع بسازیم. برای اینکار، از کلمه `ی` کلیدی `def` استفاده میکنیم:

```
def first_functions():
    pi = 3.14
    r = 10
    area = pi * r * r

    print(area)

def second_function(r):
    pi = 3.14
    area = pi * r * r

    return area
```

همانطور که مشاهده میکنید، تابع اولی، نه پارامتر قبول میکند، نه ارزشی را باز می گرداند. اما یک عملیات پرینت انجام میدهد. به این نوع توابع، همانطور که گفتیم، ووید میگویند.

تابع دوم یک تابع فلوت است، چون یک ارزش فلوت باز می گرداند. و ابتدا شعاع دایره را به عنوان پارامتر میپذیرد. یک تابع را میتوان در خودش بخواند. به این امر تابع بازگشتی^{۵۵} میگویند. مثلا برای بدست آوردن فاکتوریل یک عدد:

فصل ۳. نگاهی کوتاه به برنامه نویسی

```
def factorial(n):
    if n == 1:
        return n
    else:
        return n*factorial(n - 1)
```

اگر شرایط بازگشت در تابع محبی نباشد همانطور که در بخش قبل گفتیم، سرریزی پشته صورت میگیرد. در مورد پشته و هرم در بخش C++ صحبت خواهیم کرد. تا این حد بدانید که پایتان، مدیریت حافظه^۵ را خودش انجام میدهد و نیازی به این کار توسط شما نیست. در C++ ما متغیرهایی داریم که به آدرس یک متغیر دیگر در RAM اشاره دارند، اما در پایتان، ما همچین چیزی نداریم.

توجه داشته باشید که اسکوپ^a متغیرها در تابع، مخصوص خودشان است. اگر متغیری را در یک تابع تعیین کردید، نمیتوانید آنرا بیرون از تابع استفاده کنید. و همچنین اگر متغیری را داخل یک بلاک بیانیه‌ی شرطی یا چرخش شرطی تعیین کردید، آن متغیر، بیرون از آن بلاک، ارزشی ندارد.

Scope^a

فایل در پایتان

برای باز کردن یک فایل در پایتان، از روش زیر استفاده میکنیم:

```
f = open(filename, mode)
```

فایل حتما باید در فolder اسکریپتی که داریم مینویسم باشد. نام فایل یک استرینگ لیترال است پس باید داخل علامت نقل قول قوی باشد. حالت باز کردن فایل بستگی به عملیاتی که میخواهیم روی فایل انجام دهیم دارد. در جدول زیر، حالات باز کردن فایلها را میبینید:

باز کردن یک فایل متنی برای خواندن	'r'
باز کردن فایل متنی برای نوشت	'w'
حالت ساخت فایل. اگر فایل وجود دارد، عملیات شکست میخورد.	'x'
باز کردن فایل برای اضافه کردن متن به فایل.	'a'
باز کردن در حالت متنی.	't'
باز کردن در حالت باینری. برای هر فایلی جز فایل متنی.	'b'

جدول ۲.۳: حالت‌های باز کردن فایل در پایتان

میتوانید با علامت بعلاوه، حالت‌ها را با هم ترکیب کنید مثلاً `open("scene.jpg", 'w+b')` برای بستن فایل از اسلوب `file.close()` استفاده کنید. نوشتن روی فایلها به صورت زیر انجام می‌پذیرد:

```
with open("test.txt", 'w', encoding = 'utf-8') as f:
    f.write("my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")
```

کلاس‌های پایتان

در مورد کلاس‌ها در بخش پردایم شیء گرا حرف زدیم. در پایتان، کلاس مجموعه‌ای از توابع و متغیرهاست که به آنها اسلوب و خواص می‌گوییم. یک کلاس میتواند فرزند یک کلاس دیگر باشد. آنگاه کلاس فرزند تمام متدها و خواصهای کلاس مادر را به ارث می‌برد. یک کلاس به صورت زیر درست می‌شود:

```
class Rectangle:
    def __init__(self, color, filled, width, length):
        self.__color = color
        self.__filled = filled
        self.__width = width
        self.__length = length

    def get_color(self):
```

```

        return self.__color

    def set_color(self, color):
        return self.__color = color

    def is_filled(self):
        self.__filled

    def set_filled(self, filled):
        return self.__filled

    def get_area():
        return self.__width * self.__length

```

کلاس `rectangle` دارای یک اسلوب سازنده به نام `__int__` است که چهار خواص مستطیل را تعیین میکند. و دارای پنج اسلوب دیگر است که هر کدام یک عمل متفاوت انجام میدهند.

برای درست کردن یک فرزند از کلاس مستطیل، کافیست به صورت (۵۷) `class Square(Rectangle)` عمل کنیم. یکی از کارهایی که میشود در برنامه نویسی شی گرا کرد، **چند ریخت** کردن است. که عبارتست از تعیین یک متدهای فرزند که نام یک متدهای کلاس مادر را دارد، ولی خواصش متفاوت است.

ماژولها و پکیجهای پایتان

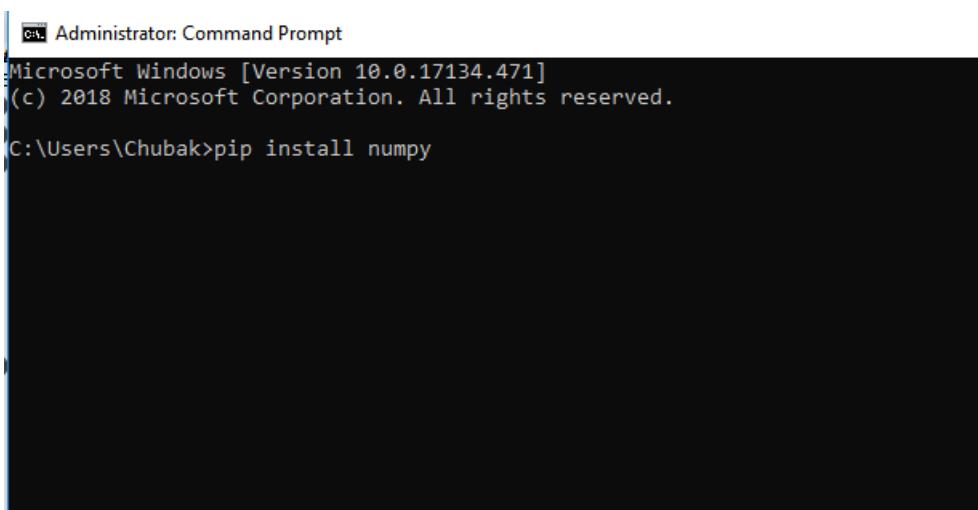
پایتان، به خاطر پکیجهای (یا همان کتابخانه ها) که برایش عرضه میشود، نامدار است. پایتان بدون پکیج، یک زبان متوسط است، مانند، `Ruby`, `Ada`, `Lua`, `Perl`، و... اما در به خاطر پکیجهایی که برای این زبان در طول سالها عرضه شده، و سادگی نصب این پکیجها، همه برای کارهای کوچک و بزرگ به پایتان روی می اورند. (۵۸) فلسفه ای نرم افزار مدیریت پکیج پایتان، یعنی `pip` — «خود را تکرار نکنید» است. برای نصب یک پکیج روی پایتان، کافیست:

. ۱. `CMD` را باز کنید.

۲. بنویسید `.pip install package-name`

۳.  را بزنید.

۴. پکیج روی سیستم شما نصب خواهد شد.



وقتی پکیج روی کامپیوترمان نصب شد، میتوانیم با فرمانهای زیر، در اول یا هرجای فایل اسکریپت، پکیج را **وارد^{۵۹}** اسکریپتمان کنیم.

```
import numpy
from numpy import class
import numpy as math
```

چندین نوع وارد کردن وجود دارد. اولی، همانطور که میبینید، وارد کردن کل پکیج است. وقتی اینکار انجام شد. میتوانیم با استفاده از فرمان `numpy.method()` اسلوبها و خواصهای مورد نظر خود را استفاده نماییم. روش دوم، وارد کردن کلاسی خاص با استفاده از فرمان `from x import y` است. و فرمان سوم، نام پکیج مورد نظر را تغییر میدهد.

هر پکیج راهنمای خود را دارد که در اینترنت پیدا میشود. در این کتاب، وقتی از یک پکیج نام میبریم، دیگر نمیگوییم آنرا چگونه نصب کنید. وظیفه‌ی خودتان است که پکیج را نصب کرده و وارد نرم افزار کنید.

پایان بخش پایتان

با پایان رسیدن بخش پایتان، لازم است یادآوری کنم که من فقط پوسته‌ی پایتان را خراش دادم. اگر میخواهید بیشتر بدانید، کتابهای متعددی برای اینکار وجود دارند.

C++ ۵.۳

زبان C++ در اوایل دهه‌ی هشتاد توسط دکتر بی‌یارنه استراستروپ، مهندس دانمارکی، ابداع شد. فرق C++ با C در برنامه نویسی شیء گر است. این زبان، عوض پایتان، کامپایل^{۶۰} میشود. یعنی، توسط یک نرم افزار به نام کامپایلر که به زبان C نوشته شده، به زبان اسembلی تبدیل میشود و سپس سیستم عامل آنرا به زبان ماشین تبدیل کرده و آنرا اجرا میکند.

کامپایلرهای زیادی برای C++ وجود دارند. دو تا از بزرگترین کامپایلرهای GCC روی لینکس و Microsoft Visual C++ روی ویندوز است. البته روی ویندوز میتوان از Cygwin یا MinGW هم استفاده کرد.

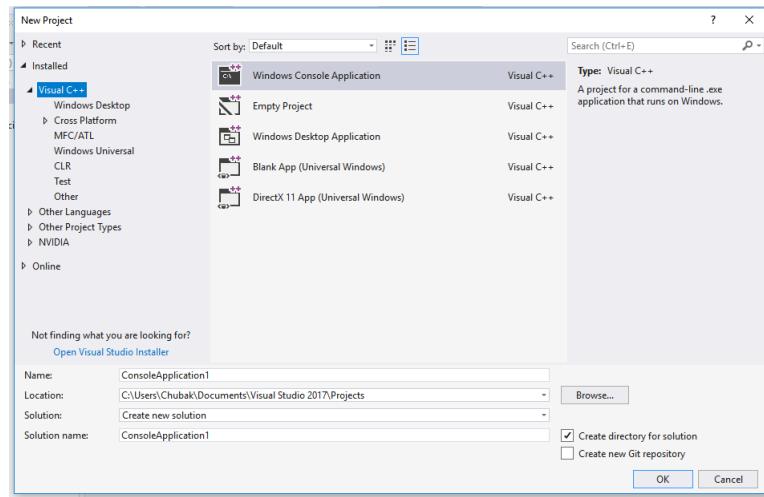
محیط گسترش مجتمع روی ویندوز، برای C++ زیاد است. اما ما از Microsoft Visual Studio استفاده میکنیم. پیشنهاد نمیکنم این نرم افزار را به صورت پایریت شده از بازار بخرید، بلکه، پیشنهاد نمیکنم نسخه‌ی مجانی Community آنرا از سایت مایکروسافت دانلود کرده و آنرا نصب کنید.

<https://visualstudio.microsoft.com/vs/community/>

یادتان باشد هنگام نصب، Visual C++ Tools را نصب کنید. و گرنه از منوی Tools میتوانید پکیج را نصب کنید. یکی از نیکوییهای ویژوال استودیو، NuGet Pack-Manager است که به شما اجازه میدهد فایلهای سری^{۶۱} کتابخانه‌ها را دانلود کنید. اما ما خودمان فایلهای سری را دانلود کرده و اضافه^{۶۲} میکنیم. به معرفی C++ بپردازیم. چون خیلی از چیزها تکرار از بخش پایتان میباشد، این بخش بسیار کوتاه خواهد بود.

برای شروع یک پروژه‌ی جدید از بخش File -> New Project یک پروژه‌ی Console Application بسازید.

Compile^{۶۳}
Header^{۶۴}
Include^{۶۵}



سینتکس

یک برنامه‌ی C++ به صورت زیر نوشته می‌شود:

```
#include "pch.h"
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}
```

کلمه‌ی کلیدی `#include` فایلهای سری برنامه را تعیین می‌کنند. فایلهای سری، اعلامیات^{۶۳} توابع، متغیرها، و کلاس‌های برنامه هستند. تعیینات^{۶۴} برنامه در فایلهای سورس برنامه قرار دارند. هر فایل سری با پسوند `.h`. یک فایل سورس با همان نام با پسوند `.cpp` دارد که حاوی تعیینات برنامه است.

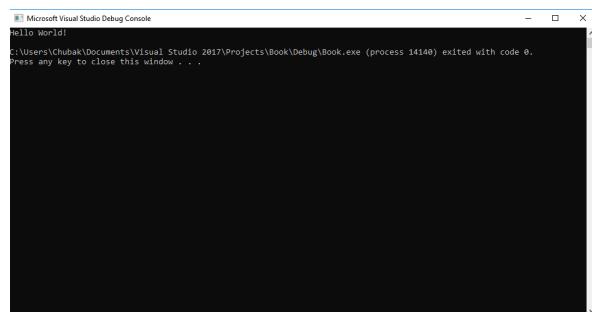
تابع `main` تابع اصلی برنامه است. اگر یک برنامه، تابع اصلی نداشته باشد، کتابخانه به حساب می‌آید. در C++ برخلاف پایتان نمی‌توانیم یک کد را خارج از تابع به اجرا درآوریم، و تابع اصلی برای همین کار است.

متد `std::cout` (تلفظش سی-او-ت می‌باشد) یکی از اسلوبهای کتابخانه‌ی **STL**^{۶۵} است.

Declaration^{۶۳}
Definitions^{۶۴}
Standard Template Library^{۶۵}

فصل ۳. نگاهی کوتاه به برنامه نویسی

میباشد. std که با دو تا دونقطه از اسلوب سی-اوست جدا شده، **فضانام**^{۶۶} اسلوب میباشد. فضانامها مجمعه ای از اسلوبها هستند که مسمای آنان، کتابخانه شان است. std مسمای اسلوبهای کتابخانه **STL** است. از علامت >> تعجب نکنید، این یک آپریتور بیت وایز میباشد. اما درینجا معنی "خروج" میدهد. با زدن **ctrl + F5** خروجی نرم افزار را خواهید دید.



تابع معکوس سی-اوست، **std::cin** (سی-این) میباشد. سی-این یک متغیر را گرفته، و به از کیبورد ارزش گرفته و به آن مقداردهی میکند.

```
int main()
{
    int variable = 0;
    std::cin >> variable;
    std::cout << variable << std::endl;
}
```

کد بالا، توسط سی-این، یک عدد به متغیر میدهد و بعد آنرا به نمایش میگذارد. که حرف آخرش L کوچک است، خط جدید ایجاد میکند. همانطور که توجه کرده اید، متغیرها در C++، نوع دارند چون C++ زبان تایپ امن است. زیر زیر چند گونه از انواع دیتا در این زبان را مشاهده میکنید.

```
//integer
unsigned int;
signed int;
```

```

unsigned short;
signed short;

unsigned long long;
signed long long;

unsigned long;
signed long;

//decimal
float;
double;
long double;

//text
char;
wchar_t;

//rest
bool;
void;

```

تمام اعداد صحیح دو حالت دارند، signed و unsigned. همانطور که بر می آید، اگر یک عدد صحیح signed باشد، میتواند اعداد منفی نیست دریافت کند. اما اگر عدد صحیح unsigned باشد، بین ۰ و مکسیمم اعداد صحیح پذیرفته شده توسط پراسسور کامپیوتر است. مکسیمم short، یک عدد بسیار کوچکتر از مکسیمم int بوده، و مکسیمم long و long int بسیار از مکسیمم int بیشترند. اما long و long، با استفاده از **حقه های اتاق نشیمن**^{۶۷} مکسیمم بیشتری دارند. مکسیمم واقعی اعدادی که یک کامپیوتر دریافت میکند، توسط واحد پراسسور مرکزی^{۶۸} تعیین میشود. مثلا یک پراسسور ^{۳۲} بیتی میتواند بین ۰، ۱۴۷، ۴۸۳، ۶۴۷ و ۶۴۷، ۴۸۳، ۱۴۷ عدد صحیح نگه دارد که مساوی ^{۲^{۳۱}} میباشد. یک پراسسور

فصل ۳. نگاهی کوتاه به برنامه نویسی

بیتی^{۶۳} ۲ مکسیمم عدد نگه میدارد. علت اینکه ۳۱ و ۶۳ است اینست که، یکی از بیتها برای علامت منفی یا مثبت عدد نگه داشته میشود. اما معنی ۳۲ بیت و ۶۴ بیت چیست؟ در بخش ۵.۳ خواهید خواند. در C++، متغیر int، ۳۲ بیتی است. اما در کتابخانه‌ی استاندارد، int ۶۴ بیتی نیز یافت میشود.

متغیرهایی که عدد اعشار میپذیرند، سه نوعند. فلوت، که در پایتان با آن آشنا شدیم، دابل^{۶۹} که دقت^{۷۰} آن دو برابر فلوت است، و دابل طولانی، که دقتش چندین برابر دابل معمولی است.

شاید برایتان سوال باشد که از فلوت استفاده کنید یا از دابل. جواب، در احتیاجات شماست. مثلا اگر π را حساب کنیم، و آنرا یک بار در فلوت قرار بدهیم، یکی بار در دابل، یک بار در لانگ دابل:

```
//float:  
3.1415927410125732421875  
//double:  
3.141592653589793115997963468544185161590576171875  
//long double:  
3.1415926535897932385128089594061862044327426701784133911132812
```

و انواع کاراکتر در C++ هستند. اولی، فقط کاراکترهای اسکی^{۷۱} دومی کاراکترهای یونیکد^{۷۲} را قبول میکند. میتوان خود کاراکتر را به این متغیر داد، یا شماره‌ی آن در یونیکد یا اسکی را. در C++ مانند پایتان، نوع استرینگ نداریم. اما در کتابخانه‌ی استاندارد استرینگ داریم که std::string نام دارد. برای استفاده از بخش استرینگ کتابخانه‌ی استاندارد مانند زیر عمل میکنیم:

```
#include <iostream>  
#include <string>
```

Double^{۶۹}
Percision^{۷۰}

ASCII^{۷۱} - کیبورد استاندارد آمریکا
Unicode^{۷۲} - کاراکترستی که تمام کاراکترهای دنیا، حتی خط میخی پارسی، را در بر دارد.

```
int main()
{
    std::string myString = "Text";
}
```

با تایپ بولی نیز آشنا هستید. تایپ ووید، یعنی تایپ خالی. کاربرد آن، کم است. هر نوع تایپ را می‌شود ترکیب کرد و **ثابته**^{۷۳} ساخت. ثابته‌ها، برعکس متغیرها، هرگز تغییر نمی‌کنند. برای ساخت ثابته از نوع فلوت به صورت زیر عمل می‌کنیم:

```
const float pi = 3.1415
```

متغیرهای اشاره‌ای و مرجعی

مسلمان شما تابحال وقتی خواستید فایلی را دانلود کنید، به سایز آن فایل نگاه کرده اید. مثلا ۱۰ مگابایت، ۲۰ گیگابایت، و یا سایز هاردیسک اکسترنال شما، مثلا ۱ ترابایت. هر بایت، مختص از ۸ بیت است. هر بیت، یک اینستراکشن به پراسسور است: ۰ یا ۱. هر بایت، یک عدد **دودویی**^{۷۴} است و هر بیت، یک رقم آن عدد است. اعداد دودویی یا باینری، عوض ارقام ۰ تا ۹، از ارقام ۰ و ۱ تشکیل شده‌اند. همچنین می‌تواند یک بایت را به صورت **شانزده‌شانزده‌ی**^{۷۵} نشان داد. ارقام شانزده‌شانزده‌ی از ۰ تا ۱۶ هستند. اما ما ارقام شانزده‌شانزده‌ی را با حروف الفبا نشان میدهیم. مثلا FF مساوی ۲۵۵ است.

از اعداد شانزده‌شانزده‌ی بگذریم و به اعداد دودویی بپردازیم. یک کامپیوتر، اینگونه عمل می‌کند:

۱. ابتدا، سیستم عامل، دستورات را به صورت ۰ و ۱ به رم می‌فرسد.
۲. پراسسور، بسته به **ساعت**^{۷۶} خود، در بازی‌های زمانی ثابت، این دستورات را از رم وارد **باس**^{۷۷} خود می‌کند.
۳. دستورات باینری وارد **دروازه‌های منطقی**^{۷۸} می‌شوند.

Constant^{۷۹}
Binary^{۷۹}
Hexadecimal^{۷۸}
Clock^{۷۹}
Bus^{۷۷}
Logic Gates^{۷۸}

۴. دستورات به اطلاعات تبدیل شده، و به دستگاههای خروجی داده میشوند.

اطلاعات در رم، با یک آدرس حافظه ای^{۷۹} هستند. آدرس حافظه، در پایه ی شانزده شانزدهی نوشته میشود. این آدرس حافظه ای در پراسسورهای اولیه فقط ۸ بیت بود، و با گذر زمان، بیشتر شد. اکثر پراسسورهای امروزی ۶۴ بیت آدرس حافظه دارند. اما بیشتر ازین هم میشود. مثل پراسسور ۲ Playstation ۱۲۸ بیت آدرس حافظه ای دارد.

در C++, حافظه به دو بخش تقسیم میشود: پشته^{۸۰} و هرم^{۸۱}. پشته، توسط پراسسور کنترل میشود و اگر سایز آن از حدی بیشتر شود، سرریز^{۸۲} میشود. هرم، دینامیک است و توسط کاربر کنترل میشود. متغیرها را باید دستی از پشته به هرم برد.

و اما متغیرهای اشاره ای^{۸۳}. متغیرهای اشاره ای، متغیرهایی هستند که به آدرس حافظه ی یک متغیر دیگر اشاره دارند و اینگونه درست میشوند:

```
#include "pch.h"
#include <iostream>

int main()
{
    int i = rand();
    int *ip = &i;
    std::cout << "'i' is: " << i << "; " <<
    "The memory address of it is" << ip << std::endl <<
    "And by adding * to ip we 'dereference' it like so: " << *ip;
}
```

خروجی این نرم افزار، اینست:

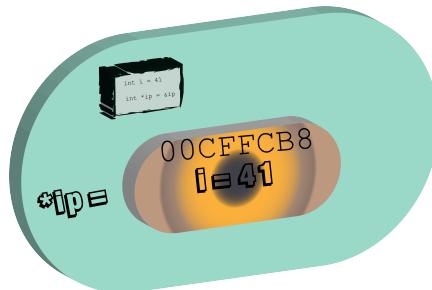
Memory Address ^{۷۹}
Stack ^{۸۰}
Heap ^{۸۱}
Overflow ^{۸۲}
Pointers ^{۸۳}

```
'i' is: 41; The memory address of it is 00CFFCB8
And by adding * to ip we 'dereference' it like so: 41
```

بگذارید این کد را مرحله به مرحله توضیح دهم:

۱. ابتدا، ما، یک متغیر به نام `i` درست میکنیم و یک عدد رندوم به آن میدهیم.
۲. سپس، ما یک متغیر اشاره ای به نام `ip` درست میکنیم. برای اینکه متغیر اشاره ای درست کنیم، از علامت ستاره ^{۸۴} استفاده میکنیم. سپس با علامت امپرسند ^{۸۵} آدرس `i` را به آن میدهیم.
۳. سپس به کامپایلر میگوییم که اول، متغیر را پرینت کن. بعد، ارزش متغیر اشاره ای را پرینت کن، که آدرس متغیر اصلی در حافظه است. سپس، متغیر اشاره ای را دیریفرنس ^{۸۶} کن. یعنی، ارزشی که در آدرس حافظه ای که به آن اشاره میکنی ر پرینت کن.

عکس زیر، گویای همه چیز است.



شکل ۱.۳: آدرس حافظه و دیریفرنس کردن

به `&i` اشاره ی مرجعی ^{۸۷} به آ میگوییم. کلا برای اشاره مرجعی به هر متغیری، از علامت امپرسند استفاده میکنیم. در بخش‌های بعد، مصرف آنرا خواهید دید.

Asterisk ^{۸۴}
Ampersand ^{۸۵}
Dereference ^{۸۶}
Reference ^{۸۷}

بیانیه های شرطی و چرخشی

بیانیه های شرطی و چرخشی، در C++ مانند همتایانشان در پایتان هستند. در زیر تتها به سینتکس شان بسنده میکنیم:

```
int i = rand();
if (i > 20)
{
    std::cout << "i is greater than 20.";
}
else if (i > 30)
{
    std::cout << "i is greater than 30.";
}
else
{
    std::cout << "i is smaller than 20 and 30.";
}
{
while (i > 0)
{
    std::cout << i << std::endl;
    i -= 0;
}

for (int h = i; h > 0; h--)
{
    std::cout << h << std::endl;
}
```

یک کار را انجام میدهند، فقط، اینگونه است که ابتدا بازتکرارکننده^{۸۸} یک مقدار میگیرد، بعد شرط میگذاریم، و بعد میگوییم چقدر از مقدار کم، اضافه،

ضرب یا تقسیم کن. به هر کدام ازین بیانیه‌هایی که در چرخش `for` داریم، **اصطلاح^{۸۹}** می‌گوییم. هر اصطلاح، یک بیانیه است، اما هر بیانیه‌ای، یک اصطلاح نیست.

درینجا، ا در اسکوپ خارجی تابعی که کد را در آن اجرا می‌کنیم، قرار دارد. به اینگونه متغیرها، **جهانی^{۹۰}** می‌گوییم. یک متغیر می‌تواند نسبت به کل کد، جهانی باشد. برای اینکار، کافیست متغیر را بیرون تابع اصلی تعیین، یا مقداردهی کنید. معمولاً ثابت‌های نسبت به همه چیز، جهانی هستند.

Global^a

اما C++ دارای یک بیانیه‌ی چرخشی دیگر به نام **do...while** هستیم:

```
do
{
    i %= 10;
    i - 1;
}
while (i > 0)
```

فرق `do...while` با `while` اینست که شرط درست در `while` در صورتی که شرط درست باشد، یک بیانیه را اجرا می‌کند. اما در `do...while` بیانیه‌ای که در برگت `do` قرار دارد، یک بار اجرا می‌شود تا شاید اگر لازم بود، شرط برقرار شود.

آرایه‌ها، بردارها، نقشه‌ها

در C++ برای نگه داشتن چندین متغیر معمولی یا اشاره‌ای، چندین نوع لیست محبی شده که پایه‌ی همه‌ی آنها آرایه^{۹۰} است. آرایه، یک گروه از دیتای هم نوع است. اندازه‌ی یک آرایه، از قبل تعیین شده است و نمی‌تواند کاهش یا افزایش یابد. اما آرایه، جهشی بوده و می‌توان اعضای آنرا توسط ایندکس داده شده، تعیین کرد.

```
int integerArray[3];
int integerArray[] = { 1, 2, 3 };
```

Expression^{۸۹}
Array^{۹۰}

فصل ۳. نگاهی کوتاه به برنامه نویسی

```
int integerArray[3] = { 0 };

std::cout << integerArray[0];

int myArray[20];
for (int i = 0; i < 20; i++)
{
    myArray[i] = rand();
}
```

۱. ابتدا ما یک آرایه با ۳ عضو میسازیم. این آرایه، خالی است.
۲. سپس ما یک آرایه با ۲۰ عضو ساخته به و به آن اعضای ۱، ۲ و ۳ را میدهیم.
۳. سپس یک آرایه‌ی ۲۰ عضوی میسازیم که همه‌ی اعضای آن ۰ است.
۴. در مرحله‌ی بعد، یک آرایه‌ی ۲۰ عضو ساخته، و توسط بیانیه‌ی چرخشی `for`، به هر عضو آن یک عدد تصادفی میدهیم. یادتان باشد که ایندکس اعضا از ۰ شروع میشود و تا سایز منهای یک ادامه دارد.

اما نمیتوان به آرایه عضوی اضافه و کم کرد. پس برای اینکار از چه استفاده کنیم؟ جواب، استفاده از یک **لیست زنجیره‌ای**^{۹۱} به نام **بردار**^{۹۲} میباشد. بردار، یکی از بخش‌های کتابخانه‌ی استاندارد است. برای استفاده از بردار به صورت زیر عمل میکنیم:

```
#include "pch.h"
#include <iostream>
#include <vector>

int main()
{
    std::vector<int> myVector;
```

```

for (int i = 0; i < 20; i++)
{
    myVector.push_back(rand());
}

for (auto &i : myVector)
{
    std::cout << i << std::endl;
}
return 0;
}

```

۱. ابتدا با فرمان `#include`، فایل‌های کتابخانه را وارد کدمان می‌کنیم.
۲. سپس، یک بردار با نوع `int` می‌سازیم.
۳. بعد از آن، بردارمان را برای ۲۰ بار یک عدد تصادفی واردش می‌کنیم. اسلوب `push_back()` برای اینکار است.
۴. سپس با استفاده از یک چرخش `for`^{۹۳} تمام اعداد را روی صفحه پرینت می‌کنیم.

در پایتان دیدیم که اگر بخواهیم یک لیست داشته باشیم که همتا به همتاست، باید از دیکشنری استفاده کنیم. اما در C++ از یک نقشه‌ی آمیزشی^{۹۴} به نام نقشه^{۹۵} استفاده می‌کنیم. دو نوع نقشه داریم، **ترتیبی** و **غیرترتیبی**. ما از نقشه‌ی غیرترتیبی استفاده می‌کنیم.

```

#include "pch.h"
#include <iostream>
#include <map>

```

```

int main()
{
    std::map<int, char> myCharMap;
    char string[] = "Hello World!";

    for (int i = 0; i < 12; i++)
    {
        myCharMap[i] = string[i];
    }

    for (auto &i : myCharMap)
    {
        std::cout << i.first << " : " << i.second << std::endl;
    }

    return 0;
}

```

۱. ابتدا، فایل سری نقشه را وارد میکنیم.
۲. سپس، یک نقشه میسازیم که ایندکسی `int` و ارزشش `char` باشد.
۳. سپس یک آرایه‌ی کاراکتری میسازیم و به آن یک متن میدهیم.
۴. سپس یک چرخش به اندازی سایز آرایه کاراکتری میسازیم و به هر ایندکس نقشه، یک کاراکتر از آرایه را میدهیم.
۵. در آخر، با استفاده از خواصهای `first` و `second` در یک `for` برداری، ایندکسها و ارزشها را پرینت میکنیم.

`C++`، تاپل نیز دارد. و همچنین چندین نوع دیتای دیگر. برای اطلاعات بیشتر در مورد هر چیزی از این زبان، میتوانید از اینترنت کمک بگیرید.

توابع در C++

توابع در C++ یک نوع برگشت دارند، و چندین پارامتر با انواع مختلف می‌پذیرند. تابع main یک تابع int است چون یک عدد صحیح باز می‌گرداند. توابع در دو مرحله ساخته می‌شوند، اعلامیه و تعیینیه. اکثر اوقات، اعلامیه در فایلهای سری انجام می‌پذیرد و تعیینیه در فایلهای سورس. اما اعلامیه همواره لازم نیست، و می‌توان بدون اعلام کردن یک تابع، آن را تعیین کرد. تعیین کردن یک تابع اینگونه صورت می‌پذیرد:

```
int numberOfDigits(int num)
{
    std::vector<int> digits;

    do
    {
        digits.push_back(num % 10);
        num /= 10;
    } while (num > 0);

    return digits.size();
}
```

این تابع، تعداد ارقام یک عدد صحیح را باز می‌گرداند.

در تابع قبلی، از یک **الگوریتم**^a استفاده کردیم. به یک سری دستور که باهم، یک کار خاص را انجام میدهند، الگوریتم می‌گویند. زبان C++ دارای یک کتابخانه به نام `<algorithm>` است که الگوریتمهای لازمه برای کار روی لیستها را به ما میدهد. در بخش‌های بعد از الگوریتمهای گرافیک کامپیوتری، برنامه نویسی گرافیکی، و بازی سازی حرف خواهیم زد.

Algorithm^a

اگر نوع دیتایی که تابع باز می‌گرداند، با نوع دیتایی که اول تابع تعیین کرده

باشد، یکی نباشد، **خطای زمان کامپایل**^{۹۶} میگیرید. یکی دیگر از انواع خطا، **خطای زمان اجرا**^{۹۷} میباشد. برای جلوگیری از این نوع خطای زمان اجرا، از **Try...Catch** و **throw** استفاده میکنیم. به این منوال، **کنترل استثنایات**^{۹۸} میگویند.

```
int division(int a, int b)
{
    if (b == 0)
    {
        throw "Division by Zero!"
    }

    return a / b;
}

int main()
{
    int a = 10;
    int b = 0;

    try
    {
        std::cout << a / b;;
    }
    catch (const std::exception& e)
    {
        std::cerr << e;
    }
}
```

Compile Time Error^{۹۶}

Run-Time Error^{۹۷}

Exception Handling^{۹۸}

سی-ار یا `std::cerr` بخشی از کتابخانه `iostream` است که وظیفه‌ی آن نمایش استثنای انداده شده توسط برنامه است.

کلاس‌های C++

کلاس‌ها، مهمترین بخش C++ هستند. اصلاً این زبان از اول برای برنامه نویسی شی‌عه گرا درست شد. یک کلاس به صورت زیر است:

```
class Ancestor
{
public:
    Ancestor() = default;
    Ancestor(int x, int y, int z) { x = x; y = y; z = z };

    int returnX() { return x };

    int doSomething() override;

private:
    int x;
    int y;
    int z;

};

class Descendent : Ancestor
{
    int doSomething() { return 2*2 };
};
```

در بخش پایتان در مورد اسلوبها و خواصها حرف زدیم. آیا میتوانید اسلوبها و خواصهای کلاس `Ancestor` را پیدا کنید؟ یکی از تابعهای `Ancestor` با کلمه `ی` کلیدی `override` مشخص شده است. اینکار برای اینست که میخواهیم کلاس‌های فرزند این کلاس، این تابع را تعیین کنند. به این کار، همانطور که در بخش پایتان گفتیم چندریختی گری یا پولی مورفیسم میگوییم.

خیلی کم اتفاق می‌افتد که یک اسلوب را داخل همان کلاس تعیین کنیم. معمولاً کلاسها را در فایل سری اعلام، و اسلوبهایش را در فایل سورس تعیین میکنیم. در طول کتاب به مراتب این کار را انجام خواهیم داد.

پایان بخش C++

من اصلاً پیشنهاد نمیکنم که به این توضیحات کم قناعت کنید. حتماً یک کتاب بخرید و آنرا بخوانید، و یا از منابع اینترنتی استفاده کنید. هرکار میکنید، حتماً مطالعه ای زبانی خود را گسترش دهید.

۶.۳ سیستمهای کنترل نسخه

سیستمهای کنترل نسخه^{۹۹} سیستمهایی هستند که با استفاده از آنها میتوانید کدهای خود را ارگانیزه کرده و در اینترنت یا کامپیوتر خود دخیره کنید. معروفترین سیستمهای کنترل ورژن، `Git` و `SVN` هستند.

سایت `Github` که بزرگترین سایت `Git` میباشد، میتواند یک `ریپازیتوری`^{۱۰۰} برای فایلهای متنی شما درست کند و با سیستم کنترل ورژن `Git`، فایلهای `Commit` و `Push` کند. کافیست نرم افزار دسکتاب گیتهاپ را دانلود کرده، یک `ریپازیتوری` و `Commit` بسازید، کدهای خود را در آن قرار بدهید و چند وقت یک بار، `Pull` کنید. توضیحات بیشتر را میتوانید از خود سایت بیابید. من در حالی که این کتاب را بدون `Push` کنید. یک سیستم کنترل ورژن را پیشنهاد نمیکنم. من در حالی که این کتاب را مینویسم، دارم هرگاهی تغییراتم را گردآوری کرده، و کامیت، و پوش میکنم. `ریپازیتوری` این کتاب را میتوانید در لینک زیر بیابید.

<https://github.com/Chubek/Book>

۷.۳ پایان فصل برنامه نویسی

خوب، فصل برنامه نویسی هم به پایان رسید. همانطور که بارها در طول فصل گفتم، این به مثابه‌ی این نیست که برنامه نویسی را یاد گرفته باشید. یادگرفتن برنامه نویسی سالها وقت میبرد. اما تا وقت هست، وقت تمرین هست. تمرین کنید تا یاد بگیرید. و در حین تمرین، یاد خواهید گرفت. هرگز ناامید نشوید چون هربار که زمین بخورید، دوبار بلند خواهید شد. کدهای خود را به اشتراک گذاشته و از سوال پرسیدن نترسید.

میتوانید با استفاده از سایتهايی مثل Code Wars و Leetcode به چالشهاي برنامه نویسی دست پیدا کنید تا برنامه نویسيتان قويتر شود.

فصل ۴

مفهومات پایه‌ی گرافیک

۱.۴ مانیتورهای کامپیوتر

امروزه در بازار، چندین نوع مانیتور وجود دارد. LED، OLED، LCD، و همه‌ی آنها دارای پنلهای مختلفی هستند. اما همه‌ی اینها **واژه‌های باب روز**^۱ هستند. مثلاً اپل، برای مانیتورهای خود از کلمه‌ی Retina استفاده میکند. در عین حال، تمام مانیتورهای امروزی، و مانیتورهای قدیمی، از یک تکنولوژی استفاده میکنند و آن تکنولوژی **صفحه شترنج**^۲ است. اما صفحه شترنج چیست؟ بگذارید ابتدا تکنولوژی مانیتورهای قدیمی و جدید را بررسی کنیم.

مانیتورهای اشعه‌ی کاتدی

مانیتورهای اشعه کاتدی^۳ با تلوزیونهای اشعه کاتدی فرقی ندارند. تنها فرقشان درینست که، آسیلیتور^۴ مانیتور، از آرایه‌ی **ویدئوگرافیکی**^۵ دستور میگیرد، اما آسیلیتور تلوزیونهای اشعه کاتدی، از امواج الکترومغناطیس دستور میگیرند.

مانیتورهای جدید، از رابط دیجیتالی **ویدئو**^۶ و رابط چندرسانه‌ای با تعیین بالا^۷

Buzzwords^۱
Raster^۲
Cathode Ray Tube^۳
Ocillator^۴
VGA^۵

عرض آرایه‌ی ویدئوگرافیکی استفاده میکنند.

DVI^a

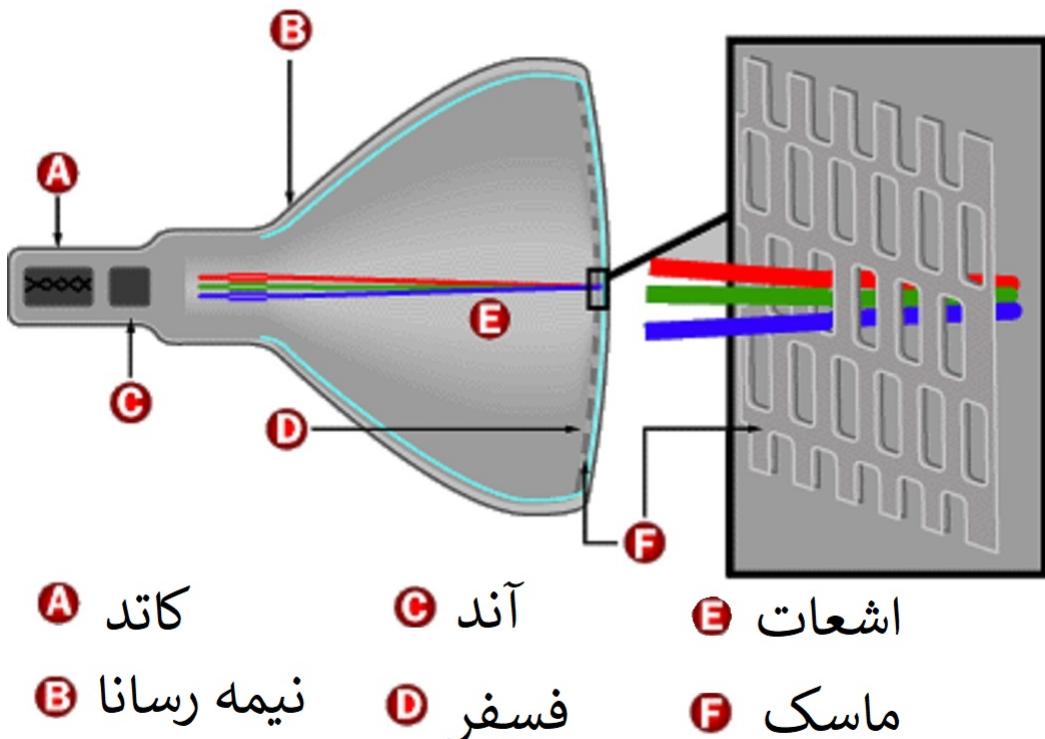
HDMI^b

همانطور که در دبیرستان آموختیم، یک اتم دارای چندین الکترون است که بار منفی دارند. با گرفتن یک الکترون از یک اتم، یک **یون**^۶ مثبت و با اضافه کردن یک الکترون به اتم، یک **یون منفی** درست میکنیم. مثلا H^- اتم هیدروژن، که یک الکترون بیشتر ندارد، با دارا شدن دو الکترون تبدیل به H^- و با یونیزه شدن مثبت، تبدیل به H^+ میشود. به **یون منفی آنیون**^۷ و به **یون مثبت، کاتیون**^۸ میگویند.

به ماده‌ای که یک سر آن رسانا و سر دیگر آن نارسانا، نیمه رسانا، یا **خلاء**^۹ است، **الکترود**^{۱۰} میگویند. یک مدار الکتریکی یا الکترونیکی، معمولاً از جفتهای الکترود به نام **آند**^{۱۱} و **کاتد**^{۱۲} استفاده میکند. کاتیونها به سمت آند، و آنیونها به سمت آنود حرکت میکنند.

در یک مانیتور اشعه کاتدی، از یک کاتد، سه **تفنگ الکترونی**^{۱۳} که هر کدام مسئول یک رنگ قرمز، سبز و آبی هستند، الکترونها را به صورت نور به سمت یک آند و سپس یک پوشش نیمه رسانا شلیک میکنند و پس نور پس از گذشت از یک **ماسک سایه**^{۱۴}، به یک صفحه‌ی پوشیده از **فسفر**^{۱۵} Ph میرسند و بر اساس دستورات آرایه‌ی ویدئوگرافیکی، تصویر صفحه شترنجی را تشکیل میدهند.

Ion ^۶
Anion ^۷
Cation ^۸
Vacuum ^۹
Electrode ^{۱۰}
Anode ^{۱۱}
Cathode ^{۱۲}
Electron Gun ^{۱۳}
Mask Shadow ^{۱۴}
Phosphorus ^{۱۵}



شکل ۱.۴: مکانیزم سی آر تی

شاید برایتان سوال باشد که الکترون که یک ذره^a است، چگونه تبدیل به نور میشود؟ وقتی جسم داغ میشود بعضی از الکترونها به مدار بالاتر میجهند که در حقیقت جایشان آنجا نیست و لذا ناپایدارند. موقعی که این الکترون تحریک شده دوباره به مدار خود بر میگردد یک فوتون^b نور آزاد می شود.

Particle^a
Photon^b

در قدیم، مانیتورهای تک رنگه^a، فقط رنگ سبز را نمایش میدادند چون رنگ اصلی Ph سبز است

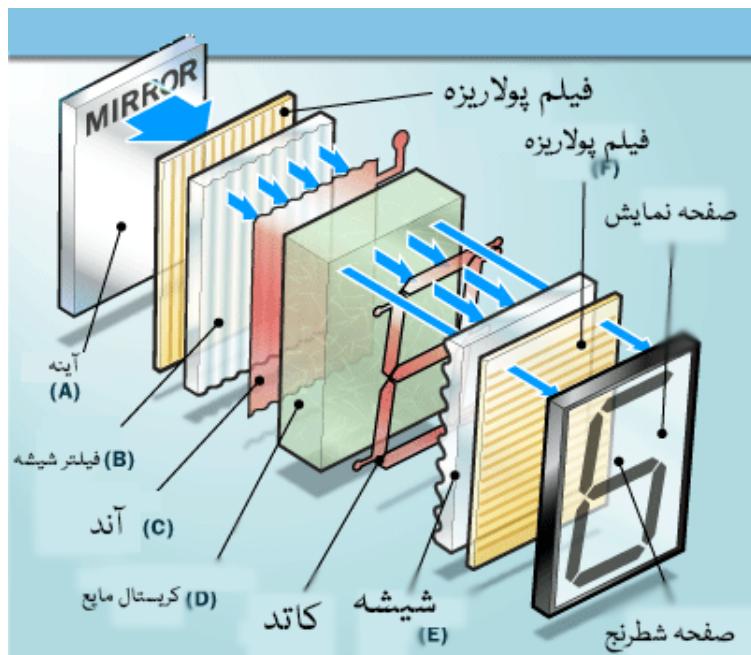
Monochrome^a

فصل ۴. مفهومات پایه‌ی گرافیک

در بخش ۲.۴ به بررسی اینکه آرایه‌ی ویدئوگرافیکی، چگونه صفحه‌ی مانیتور را دستکاری می‌کند، حرف خواهیم زد. اما قبل از آن، به بررسی مانیتورهای کریستال مایع^{۱۶} بپردازیم.

مانیتورهای کریستال مایع

مانیتورهای کریستال مایع در قدیم فقط روی ماشین حسابها و ساعتهای دیجیتال یافت می‌شدند چون فقط می‌توانستند یک آرایه‌ی هشت عضوه‌ی نشان دهند که با آن فقط می‌شد اعداد و به صورت محدود، چند حروف الفبا نشان داد. به این نمایشگر **ماژول کریستال مایع**^{۱۷} گفته می‌شود، و هنوز کاربرد دارد. برای یاد گرفتن اینکه یک مانیتور کریستال مایع چگونه کار می‌کند، باید بفهمیم یک ماژول کریستال مایع چگونه کار می‌کند.



شکل ۲.۴: چگونگی کارکرد ماژول کریستال مایع

همانطور که می‌بینید، خود کریستال مایع بین یک آند و یک کاتد قرار دارد که شکل آند، مانند شکل ۸ می‌باشد. بسته به دستورات باینری، این شکل تبدیل به ۰

LCD^{۱۶}

LCD Module^{۱۷}

۴.۱. مانیتورهای کامپیوتر

۶۱

۹، ۸، ۷، ۶، ۵، ۴، ۳، ۲، ۱ میشوند. ماشین حسابهای قدیمی تگزاس اینسترامنتس، دارای ۸ ماظول کریستال مایع بودند.^{۱۸} توجه دارید که نور، یک بار قبل از بازتاب، و یک بار قبل از خروج، پولاریزه^{۱۹} میشود. در دو عکس زیر، اثر فیلتر پولاریزه‌ی دایره‌ای را روی یک دوربین DSLR خواهید دید. عکاس هردو عکس خودم هستم.^{۲۰}



شکل ۴.۳: یک عکس گرفته شده بدون فیلتر پولاریزه



شکل ۴.۴: یک عکس گرفته شده با فیلتر پولاریزه

مانیتورهای کریستال مایع امروزی نیز مانند همین عمل میکنند، اما آند آنها، یک الکترود ایندیم-قلح-اکسید^{۲۱} است که به صورت In_2O_5Sn نمایش داده میشود. این آند، میتواند رنگهای مختلفی به خود بگیرد. آرایه‌ی ویدئوگرافیکی، ۲۵۶ رنگ قرمز، ۲۵۶ رنگ سبز، و ۲۵۶ رنگ آبی به این آند میدهد و این آند، با ترکیب آنها،

Polarize^{۱۸}
من عاشق عکاسی هستم و یک دوربین T3i کن نیز دارم. میخواهم روزی از دوربینم برای ساخت تکسچر استفاده کنم.
Indium Tin Oxide^{۲۰}.

$16777216 = 256^3$ رنگ نمایش دهد. هر نگ همچنین دارای شدت خود است که در مورد آن صحبت خواهیم کرد.

کنسول دستی **Game and Watch** شرکت نینتندو از اولین کنسولهای دستی ای بود که از یک مانیتور کریستال مایع استفاده میکرد. اولین کنسول دستی ای که از مانیتور کریستال مایع استفاده میکرد که از پشت آینه، نور میتاباند، **PSP** سونی بود. امروزه مانیتورهای کریستال مایع همه از پشت نور میتابانند. مانیتورهای LED و پلاسما هم مکانیسمی شبیه به همین دارند. اما ما، بحثمان سخت افزار نیست، بلکه بحثمان تصویری است که روی صفحه‌ی مانیتور نقش میبندد. مانیتورهای کریستال مدرن همه صفحه‌ی شترنجی‌اند، یا به عبارتی، Raster. بگذارید در بخش بعد، کامل توضیح بدهم.

۲.۴ صفحه‌ی شترنجی و المان تصویری (پیکسل)

وقتی من نوجوان بودم، سالها قبل ازینکه مردم با دوربینهای موبایل خود شروع به سلفی گرفتن کنند، دوربینهای دیجیتال تازه وارد بازار شده بودند. برندهای متفاوت، باعث رقابتی سخت بین سازندگان دوربین شده بود تا به مردم بباورانند که کیفیت دوربین آنها، از بقیه بهتر است. اگر کیفیت لنز، کُدک فایلها، قابلیت ضبط ویدئو، کوچک بودن دوربین، و قیمت آن، پیش غذای رقابت دوربینها بود، **مگاپیکسل^{۲۱}** شیرینی خامه‌ای بعد از غذا بود.

«سلام آقا، یک دوربین میخواستم... چی دارید؟»

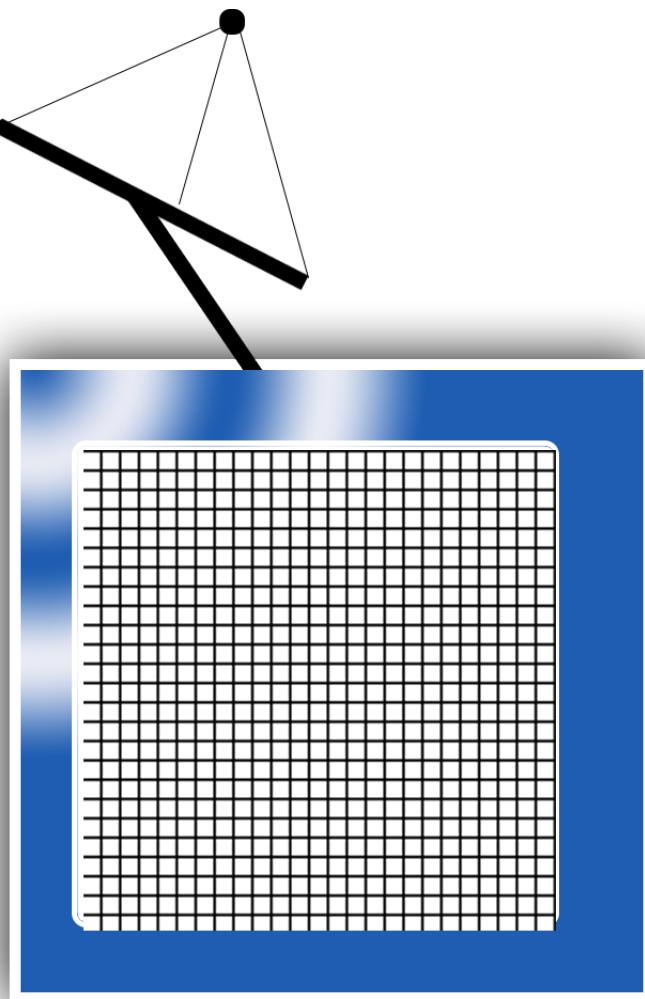
«مینولتا بدون آینه، قابلیت ذخیره‌ی تصاویر به صورت خام، لنز قابل تعویض، قابلیت تغییر بالانس سفیدی^{۲۲} — ایزو تا ۳۲۰۰، مود منوال...»

«حاجی، اینا رو بیخیال، مگاپیکسلش چنده؟»

و اینگونه بود که، بر اساس گفته‌های ریچارد داوکینز، **مگاپیکسل**، یک **میم^{۲۳}** شد. اما پیکسل چیست؟

پیکسل کلمه‌ای قدیمی است که از زمان تلوزیونهای سیاه و سفید وجود داشته، و **Picture Element** یا المان تصویری معنی میدهد. هر صفحه‌یا زیرصفحه، دارای طول×عرض پیکسل است.

۲.۴. صفحه‌ی شطرنجی و المان تصویری (پیکسل)

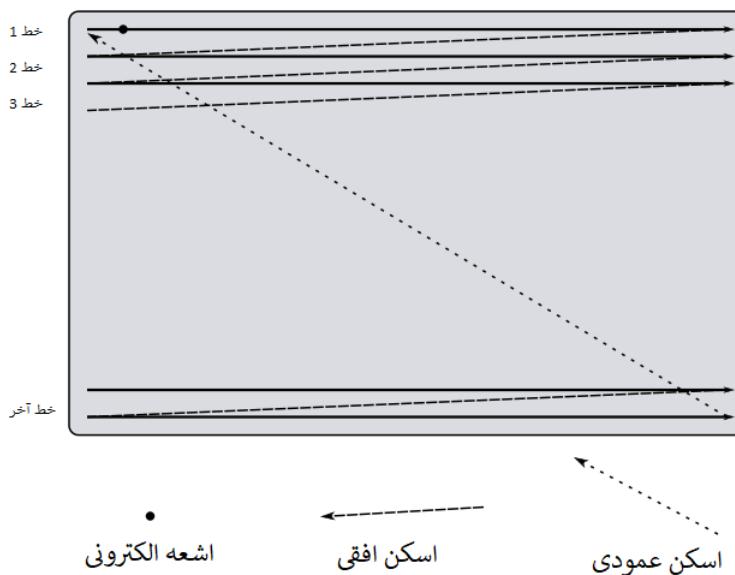


شکل ۵.۴: هر صفحه یا زیرصفحه طول \times عرض پیکسل دارد. به این مجموعه پیکسل، رستر می‌گویند.

اما صفحه‌ی شطرنجی یا رستر چیست؟ به مجموعه پیکسلهای یک صفحه یا زیرصفحه، که توسط اسکن رستر یا رسترایزیشن درست شده، صفحه‌ی شطرنجی یا رستر می‌گویند. ما ازین بعد از کلمه‌ی رستر استفاده خواهیم کرد.

تلوزیونهای قدیمی، تقریبا در هر ۱۰۰ میلی ثانیه، که به آن آهنگ تازه سازی ^{۲۴} می‌گویند، امواج الکترومغناطیسی که توسط آنتن ^{۲۵} دریافت می‌شود را تبدیل به پیکسل

میگرد. تلوزیونهای قدیمی در صفحه‌ی خود ۳۰۷۲۰۰ پیکسل داشتند. ۶۸۰ پیکسل افقی و ۴۸۰ پیکسل عمودی. و پیکسل آنها مانند پیکسل اکثر صفحه‌های مدرن، مربع نبود بلکه $11:10$ نسبت^{۲۶} داشت. این پیکسلها به صورت زیگزاگی پر میشدند. به این امر، اسکن رستری^{۲۷} میگفتند.



اسکنرهای مدرن از همین تکنیک برای رستر سازی^{۲۸} دیتای خامی که اسکن میشوند، استفاده میکنند. اما رستر سازی چیست؟ در بخش بعد در مورد آن صحبت میکنیم.

۳.۴ رستر سازی

گفتیم که، به آرایه‌ای از پیکسلها، که تعداد آنها، اندازی طول تصویر ضربر اندازی عرض تصویر است، رستر میگوییم. و همچنین گفتیم این پیکسها، به صورت زیگزاگی، در بازه زمانی خاص که همانطور که گفته شده به آن آهنگ تازه سازی میگویند، بازسازی میشوند. آهنگ تازه سازی یک فرکانس^{۲۹} است. فرکانس، تعداد کارهای

Aspect Ratio^{۲۶}

Raster Scan^{۲۷}

Rasterization^{۲۸}

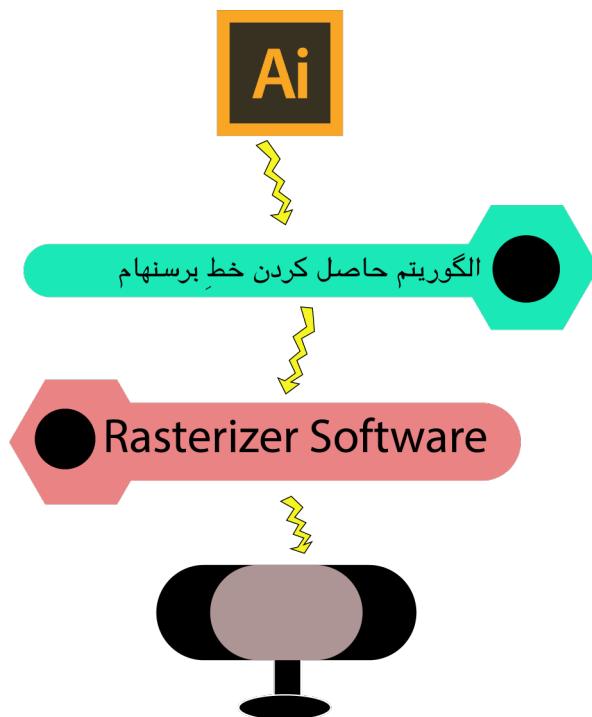
Frequency^{۲۹}

انجام شده در یک بازه است. واحد فرکانس، هرتز^{۳۰} میباشد. مانیتورهای اشعه کاتدی ای هستند که آهنگ تازه سازی شان ۱۲۰ مگاهرتز است. اما آهنگ تازه سازی اکثر تلوزیونها و مانیتورهای کریستال مایع امروزی، ۶۰ مگاهرتز میباشد.

پس ۶۰ بار در هر ثانیه، اسکن رسترنی صورت میپذیرد. اما برای اینکه اسکن رسترنی صورت بپذیرد، تمام اطلاعات تصویری داده شده به مانیتور، باید رسترنی باشد. چون مانیتورها همه رسترنی هستند و نمیوانند تصویری که رسترنیست را، نمایش دهند. به پرسه‌ی تبدیل تصاویری که با فرمول دو بعدی یا سه بعدی در نرم افزارهای مختلف به دست آمده‌اند، رسترسازی میگویند. به نرم افزاری که فرمولهای برداری در R^2 را در محیط دو بعدی، تبدیل به رسترنی کند، رسترایزر^{۳۱} و به نرم افزاری که اشکال سه بعدی را تبدیل به رسترنی کند، رندرر^{۳۲} میگویند. یک رندرر، یک خط لوله‌ی گرافیکی^{۳۳} میباشد که رسترایزر، جزئی از آن میباشد. در بخش ۵.۴ در مورد این خط لوله بیشتر خواهید خواند.

مثلا، نرم افزار Adobe Illustrator، یا آلتیرنیتو مجانی آن، Ink Escape — فرمولهایی که در بخش‌های بعدی بررسی خواهیم کرد، مانند الگوریتمهای به دست آوردن خط در فضای دو بعدی را، تند تند تبدیل به مجموعه‌ای از پیکسل کرده، و در صفحه نمایش میدهد.

Hertz^{۳۰}
Rasterizer^{۳۱}
Renderer^{۳۲}
Graphics Pipeline^{۳۳}



شکل ۶.۴: رسترایز آدوبی ایلوستریتور

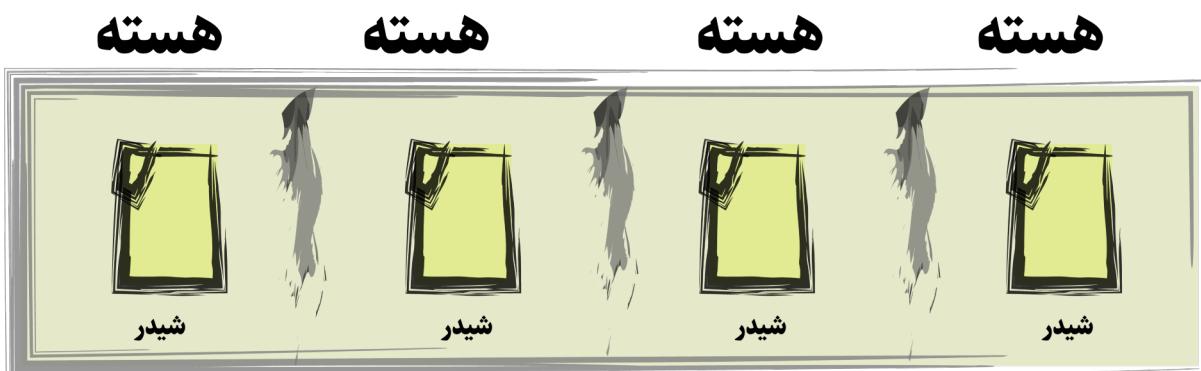
اما میتوان سرنوشت یک پیکسل را مستقیماً به دست گرفت. برای اینکار کافیست که از یک ادیتور رستر مانند **GIMP** یا **Adobe Photoshop** استفاده کرد. این گونه نرم افزارها، روی هر پیکسل عکس یا تصویری که خودمان درست کرده ایم، تاثیر میگذارند. در ویدئوهایی که همراه کتاب عرضه شده اند، من به شما یاد میدهم تا با استفاده از اداره کردن پیکسلهای یک تصویر، یک پیکسل آرت^{۳۴} برای بازیهای خود بسازید.

شاید برایتان سوال باشد که رسترایز، از کجا میفهمد رنگ پیکسل چیست؟ یا اینکه موقعیتش کجاست؟ در بخش بعدی درین مورد صحبت خواهیم کرد.

۴.۴ سایه زنهای ترکشی

کامپیوترهای امروزی همگی از کامپیوترهای **IBM/۲** نشئت میگیرند. و اینگونه کامپیوترها، در اواسط دهه ۹۰، صاحب پراسسور گرافیکی^{۳۵} شدند. قبل از آن گرافیک در

پراسیسور مرکزی سیر روند اجرایی خود را طی میکرد.
 یک پراسیسور گرافیکی، دارای چند صد هسته^{۳۶} بوده که هر کدام شان چند ده رشته نخ^{۳۷} دارند. این گونه، پراسیسور گرافیکی هزاران برنامه را در یک آن اجرا میکند.
 به این برنامه هایی که در هسته های پراسیسور گرافیکی اجرا میشوند، سایه زن^{۳۸} یا شیدر میگویند.



پراسیسور گرافیکی

شکل ۷.۴: تمثیل شیدرها

چندین نوع سایه زن داریم. به سایه زنی که وظیفه اش، نگاه داشتن رنگ، موقعیت، بیلیورد^{۳۹}، پنجه و... میباشد، سایه زن ترکشی^{۴۰} میگوییم. در مورد سایه زن های ترکشی در طول کتاب صحبت خواهیم کرد. هر رابط گرافیکی^{۴۱} مانند دایرکت تری دی^{۴۲} و اوپن جی ال^{۴۳} زبان خود را برای سایه زنی دارد. زبان سایه زنی اوپن جی ال، GLSL نام دارد که با آن آشنا خواهیم شد. ما، ابتدا با استفاده از کتابخانه ای پراسیینگ^{۴۴} سایه زنها را اجرا، و سپس از مورد رابط گرافیکی اوپن جی ال یاد خواهیم

Core ^{۴۶}
Thread ^{۴۷}
Shader ^{۴۸}
Billboard ^{۴۹}
Fragment Shader ^{۴۰}
Graphics API ^{۴۱}
۳D Direct ^{۴۲}
OpenGL ^{۴۳}
Processing ^{۴۴}

گرفت. اینگونه، آماده خواهیم بود تا سایه زنهای نقطه‌ای^{۴۵} و سایه زنهای هندسی^{۴۶} را که برای ساخت اشکال سه بعدی لازمند، به راحتی یاد بگیریم.

۴.۵ رابطه‌ای گرافیکی

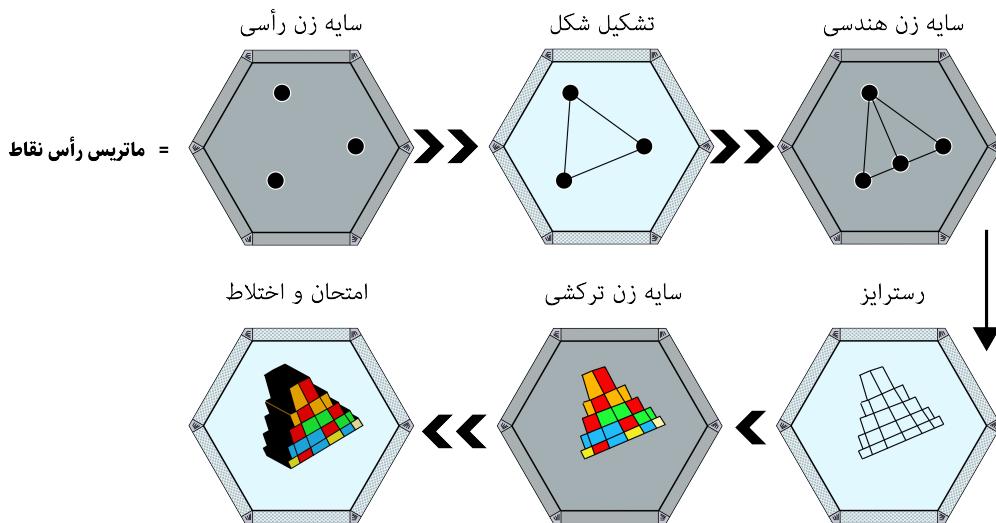
برای هر کار گرافیکی ای در کامپیوتر که به پراسسور گرافیکی مرتبط است، احتیاج به یک رابط برنامه نویسی کاربردی^{۴۷} مخصوص گرافیک داریم. رابط نرم افزاری مجموعه ای از روتینها^{۴۸}، ساختمانهای داده^{۴۹} و کلا، ابزاریست که مسهل برنامه نویسی هستند. گاهی اوقات، این مسهلهای فقط در حد اعلامیه هستند، اما گاهها تعیینیات آنها نیز به صورت یک کتابخانه‌ی دینامیک^{۵۰} که همگی شما به صورت فایلهای DLL. با آنها آشناشی دارید، عرضه می‌شوند. بعضی از روابط نرم افزاری پایین دست^{۵۱} بوده، یعنی مستقیماً با سخت افزار ارتباط برقرار می‌کنند، و بعضی از آنها بالا دست^{۵۲} بوده — یعنی با سیستم عامل ارتباط برقرار می‌کنند.

دو رابط گرافیکی دایرکت تری دی و اوپن جی ال روابط گرافیکی پایین دست بود که اولی توسط مایکروسافت، و دومی توسط گروه Kronos کنترل می‌شوند. در حالی که این کتاب نوشته می‌شود دایرکت تری دی ۱۲ روی ویندوز ۱۰ بیرون آمد، اوپن جی ال ۴.۶.۰ به سر می‌برد. دایرکت تری دی را خود مایکروسافت، تعیین کرده و میتوانید فایلهای دینامیک آنرا در پوشه‌ی Windows\System32 بیابید. و اما اوپن جی ال توسط هر سازنده‌ی کارت گرافیک، مثل AMD Intel، nVidia تعیین می‌شود و همراه درایور کارت گرافیک عرضه می‌شود.

همانطور که خواهیم آموخت، تمام کتابخانه‌های دینامیک ویندوز در پوشه‌ی System32 قرار دارد. اما میتوان در متغیرهای محیطی، پوشه‌ای که کتابخانه در آن قرار دارد را اضافه‌ی متغیر Path کرده و اینگونه دیگر لازم نیست آنرا در System32 کپی کنید.

Vertex Shaders ^{۴۸}
Geometry Shaders ^{۴۶}
Application Programming Interface ^{۴۷}
Routines ^{۴۸}
Data Structures ^{۴۹}
Dynamic Library ^{۵۰}
Low-level ^{۵۱}
High-level ^{۵۲}

اوپن جی ال برای نمایش اشکال سه بعدی روی صفحه از یا همانطور که گفتیم، رندر کردن آنها از خط لوله‌ی گرافیکی ای استفاده میکند که در زیر میبینید:



شکل ۸.۴: خط لوله گرافیکی اوپن جی ال

— گروه Kronos همچنین یک رابط نرم افزاری دیگر اعلام میدارد که **OpenCL** با C — نام دارد. این رابط نرم افزاری به کاربر اجازه میدهد سایه زنها را روی پراسسور مرکزی اجرا کند. به این سایه زنها **مغزک**^{۵۳} میگویند. شرکتهای زیادی با استفاده ازین رابط نرم افزاری، خط لوله‌ها وابسته به پراسسور مرکزی خود را میسازند و به عنوان رندرر برای پکیجهای سه بعدی عرضه میدارند. همچنین گاهی میتوان با غیرفعال کردن بخشی از خط لوله، به اثراتی خاص دست یافت. مثلا اثر **Cel Shading** که با دستکاری بخش اختلاط به دست می‌آید.



شکل ۹.۴: سایه زنی سلولی

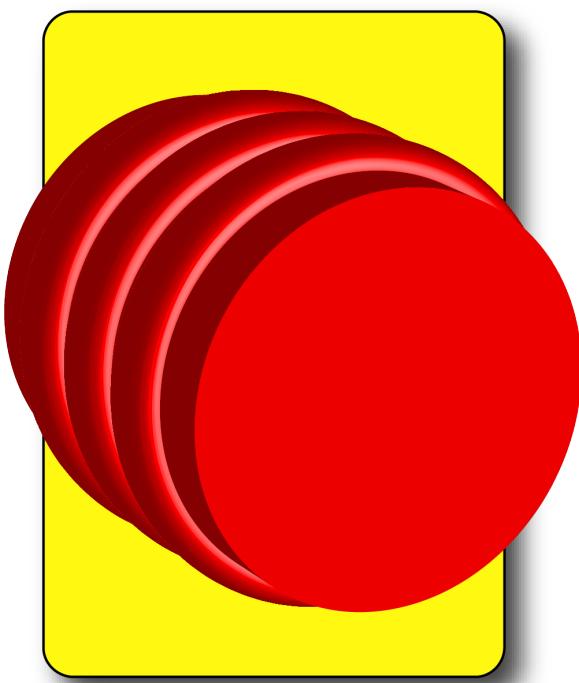
۶.۴ فریم بافر و رم



شکل ۱۰.۴ : FX ۵۲۰۰ GeForce

در تصویر بالا، اولین کارت گرافیک مرا می‌بینید. آن زمان، این کارت گرافیک با اینکه حافظه‌ی ویدئویی بالایی داشت، باز هم ضعیف بود چون کلاک پر اسپورش کم بود. خیلی از مردم گمان می‌کنند حافظه‌ی گرافیکی، یا **VRAM** تعیین کننده‌ی قدرت کارت گرافیک است. در حالی که حافظه‌ی گرافیکی تنها نگه دارنده‌ی مرحله‌ی آخر خط لوله‌ی گرافیکی، یعنی **فریم بافر**^{۵۴} است.

هنگامی که سایه زن ترکشی کارش تمام شده، و اختلاط صورت می‌پذیرد، تمام فریم‌هایی که رسترایز شده‌اند، وارد حافظه‌ی ویدئویی می‌شوند. به این مرحله، همانطور که گفتیم، فریم بافر می‌گویند.



شکل ۱۱.۴: تمثیل فریم بافر

در کامپیوترهای قدیمی فریم بافر جزئی از رم بود. اما امروزه وی رم جزئی از کارت گرافیک است. فریم بافر، فاصله‌ی بین پراسسور گرافیکی و پراسسور مرکزی میباشد. در مورد فریم بافر بیشتر صحبت خواهیم نمود.

۷.۴ پایان فصل مفهومات گرافیکی

به پایان فصل مفهومات گرافیکی رسیدیم. مفاهیمی که آموختیم، پایه بود و تنها به عنوان یک سکوی پرتاب حساب میشود. در فصلهای بعدی در مورد اوپن جی ال و سایه زنها خواهیم آموخت. قبل از جهش به فصل بعد، سعی کنید تمام مفاهیمی که آموختیم را در اینترنت جست و جو کرده، و بیشتر در مورد آنها بیاموزید. در بخش ۵ در مورد الگوریتمهای رسم و منحنیهای گرافیکی خواهید خواند.

فصل ۵

الگوریتمهای رنگ، رسم و منحنیها

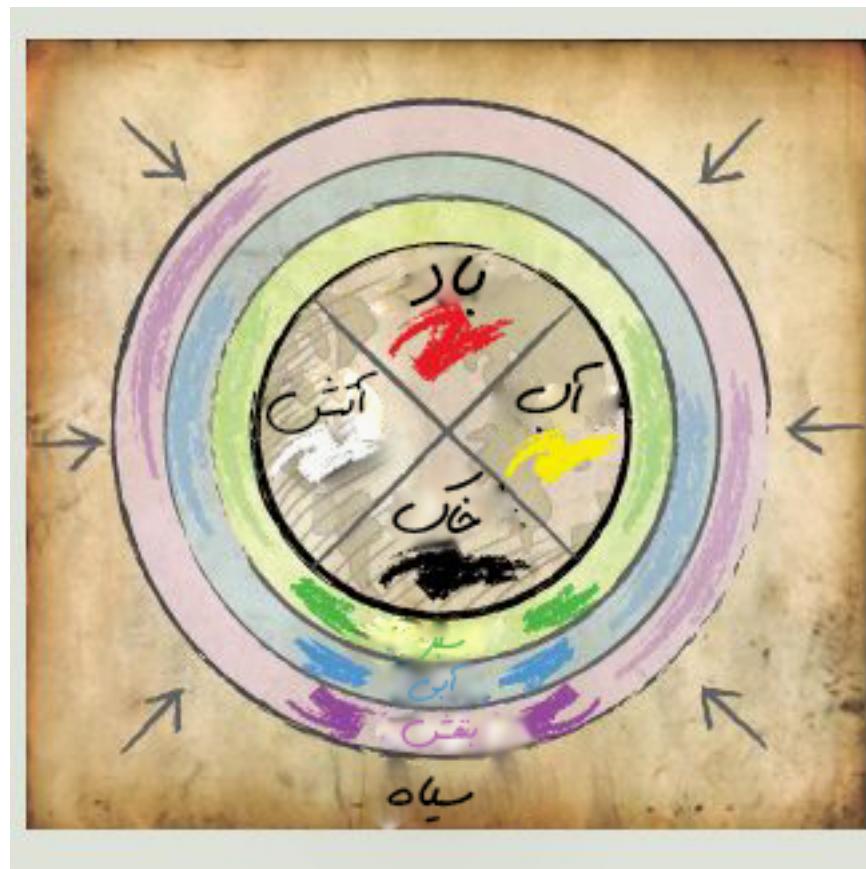
در این فصل، در مورد الگوریتمهای رسم اشکال هندسی در صفحه‌ی رستر، رنگ، و رسترايز کردن ساده، حرف خواهیم زد. کدهای این فصل نام گذاری شده‌اند و در فایل **codes.zip** که از سایت خریده اید، حاضر و آماده‌ی کپی کردن هستند.

اکثر کدهایی که در این کتاب می‌بینید نوشته‌ی خود من هستند، و خواهش‌ها وقتی انها را جای دیگری پست می‌کنید، از من نام ببرید و لینک کتاب را اضافه کنید.

اما قبل از آن، برای تست الگوریتم‌ایمان، احتیاج به یک پکیج رستر داریم. ما برای این امر از پکیج رستر **Pillow** پایتان استفاده مینیم. اما قبل از آن بگذارید رنگ در کامپیوتر را توضیح بدهم.

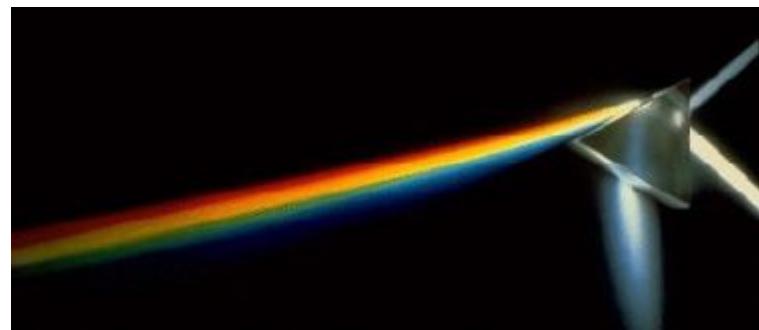
۱.۵ رنگ در کامپیوتر

گفتم سایه زن‌های ترکشی، به پیکسلها، رنگ میدهند. اما این رنگ چگونه حساب می‌شود؟ برای اینکار باید به تاریخ بازگردیم. زمانی که فلاسفه‌ی یونیه مانند ارسسطو باور داشتند رنگ، موهبتی از خدایان است و از چهار عنصر آتش، خاک، باد، و آب تشکیل شده است.



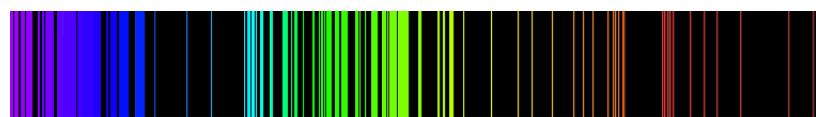
شکل ۱.۵: تصور ارسسطو از رنگ

در سال ۱۶۶۶ آیزک نیوتون نشان داد که میتوان با رد کردن رنگ از بین دو منشور^۱ یک طیف رنگی — که مساوی رنگهای رنگین کمان است — به دست آورد.



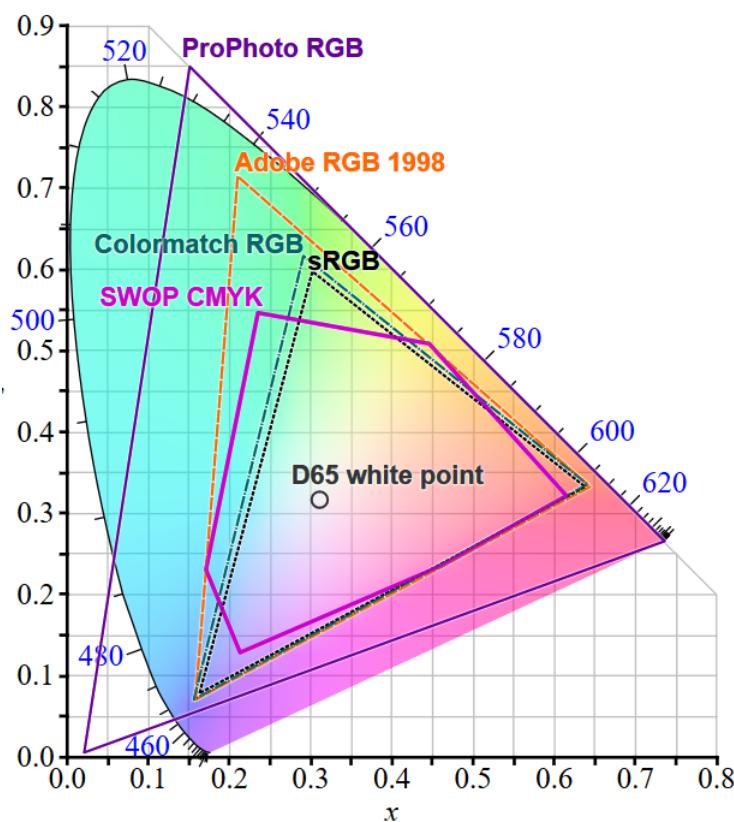
شکل ۲.۵: طیف رنگی

میتوان با داغ کردن یک اتم هیدروژن، نور ساطح شده از آن را با همین تکنیک، شکست. به این نور شکسته شده، **طیف نشری خطی**^۳ هیدروژن، یا هر اتم دیگری، میگویند.



شکل ۳.۵: طیف نشری خطی آهن

در دنیای واقعی، رنگها از قرمز، آبی، و زرد تشکیل شده اند. اما در مانیتورهای امروزی، این سه رنگ، قرمز، سبز، و آبی میباشد. از زمان خروجی آرائه‌ی ویدئوگرافیکی تا به الان، **فضای رنگی**^۴ ای که سایه زنها استفاده میکردند، فضای "قرمز سبز آبی" یا RGB میباشد.

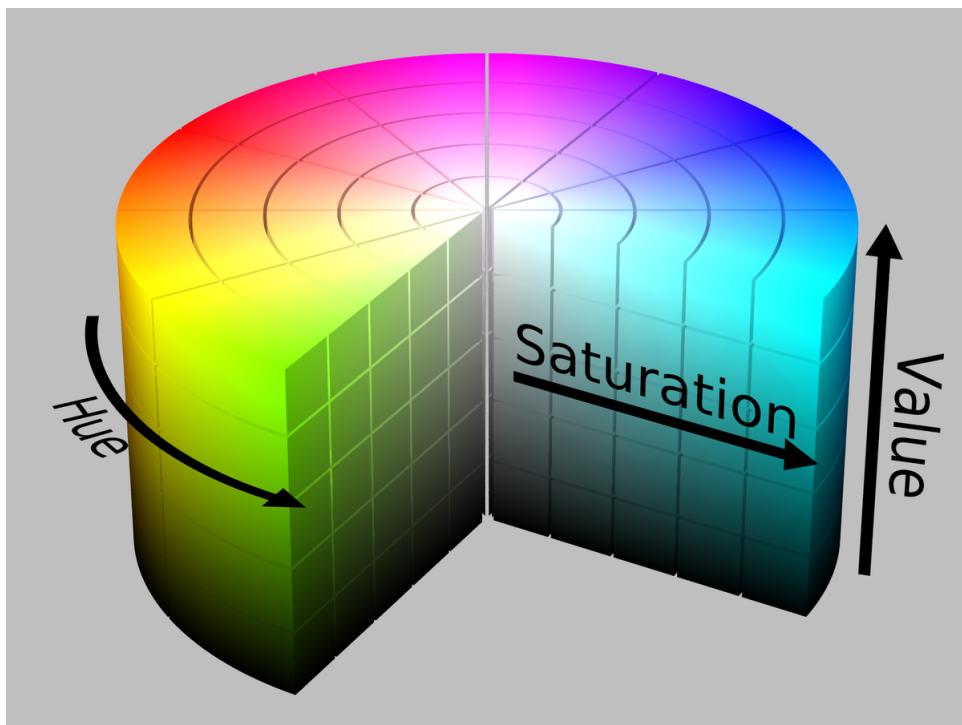


شکل ۴.۵: فضای قرمز، سبز، آبی در دستگاه مختصات دکارتی

همانطور که مشاهده میکنید، RGB یک فضای مثلث تا نیم دایره مانند، بسته به ورژن آن، در صفحه‌ی مختصات دکارتی میباشد. هر رنگ در فضای RGB به صورت یک تاپل سه عددی (r, g, b) نمایش داده میشود. هر کدام ازین سه، بین 0 تا 255 قرار دارد. میتوانید با آنابالای 16 میلیون رنگ بسازید. اگر این اعداد را نرمال سازی^۴ کنیم بین 0 و 1 میشوند. و سایه زن‌ها ازین اعداد استفاده میکنند.

در کامپیوتر جز RGB — که در فضای مختصات دو بعدی قرار دارد — ما میتوانیم از مدل‌های رنگ و فضاهای رنگ دیگر نیز استفاده کنیم. یکی ازین مدلها و فضاهای، ته رنگ، اشباع، ارزش^۵ یا HSV نام دارد که یک استوانه در دستگاه مختصات قطبی میباشد.

Normalize^۶
Hue, Saturation, Value^۷



شکل ۵.۵: فضای رنگ، اشباع، ارزش در دستگاه مختصات قطبی

رنگ‌های این فضا به صورت یک تاپل (h, s, v) نمایش داده شده، و هر کدامشان اعداد درصدی بین ۰ و ۱۰۰ هستند. هم RGB و HSV یک عضو چهارم به نام **alfa** دارند که نشان دهندهٔ پررنگی یا کم‌رنگی رنگ است. می‌توان با استفاده از پکیج colorsys پایتان، که به صورت معمول روی همهٔ عرضه‌های اینترپرتر نصب است، رنگها را از بین فضاهای مختلف تبدیل کرد. بعدها ازین تکنیک استفاده خواهیم نمود.

۲.۵ کتابخانه‌ی تصویری پایتان

کتابخانه‌ی تصویری پایتان در قدیم PIL نام داشت و همراه پکیج‌های اصلی عرضه می‌شد. اما، از ورژن^۳ ببعد، آپدیت این پکیج قطع گردید و ما هم اکنون برای استفاده از آن مجبوریم از پکیج Pillow که یک کد کاغذ کادویی^۶ از PIL می‌باشد استفاده

نماییم.

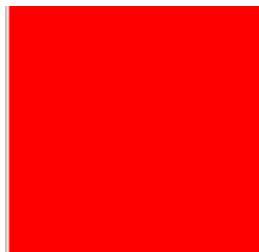
همانطور که آموزش دادیم، با استفاده از پیپ – pip – پکیج Pillow را نصب کنید. سپس به حالت زیر عمل کنید:

```
from PIL import Image

img = Image.new('RGB', (32, 32), color= 'red')
img.save('pil_color.png')
```

Listing 5.1: PIL-Intro.py

اگر عکسی که ایجاد میشود (در فolder کد) را باز کنید، عکس زیر را خواهید دید:



شکل ۶.۵: تصویر ۳۲ در ۳۲ پیکسل رستر ما

اینگونه یک تصویر ۳۲ در ۳۲ ایجاد میکنیم. میتوانید سایز، و رنگ را تغییر بدهیم.

```
img = Image.new('RGB', (64, 64), color= (23, 11, 24))
img.save('pil_ChangedColor.png'))
```

Listing 5.2: PIL-ChangeColor.py

فایل جدید را باز کرده، و متوجه خواهید شد که یک تصویر ۶۴ در ۶۴ خواهید دید. میتوان اطلاعات پیکسل خاص را تعیین کرد، و یا اطلاعات پیکسل خاص را از رستر گرفت.

```
pixels = img.load()
```

```

print(pixels[32, 32]) #get
pixels[32, 32] = (255, 255, 255) #set
print(pixels[32, 32]) #get again

img.save('pil_GetSet.png')

```

Listing 5.3: PIL-GetSet.py

این یک تصویر مشکی با یک نقطه‌ی سفید در وسطش تولید می‌کند. همانطور که می‌بینید کافیست نقطه‌ای که می‌خواهیم را بین آکولاد قرار داده و رنگی که می‌خواهیم به صورت تاپل سه تایی به پیکسل میدهیم. حال ما می‌خواهیم با این رسترایزر ساده، الگوریتمهای رسم اشکال هندسی و منحنیها را بررسی تماییم.

۳.۵ خط راست

در بخش بردارها آموختیم که یک دو نقطه در فضای مختصات دکارتی، وقتی به هم وصل می‌شوند، یک بردار تشکل میدهند:

$$\vec{V} = (x_2, y_2) - (x_1, y_1)$$

این فرمول را می‌توان به صورت تابع خط راست به حساب دیفرانسیل آورد:

$$y = mx + b$$

که در آن، y خط، m شیب خط، x متغیر ما، و b عرض از مبدأ^۷ می‌باشد. پس ما برای رسم خط در فضای دو بعدی، ک هم‌مجموعه پیکسلهای عکسمان می‌باشد، احتیاج به دو نقطه، و یک شیب داریم.

به m مشتق^۸ تابع خط راست می‌گویند. در توابعی با توان بالاتر متغیر، مشتق، شیب خط مماس^۹ را مشخص می‌کند.

Derivative^a

y-intercept^v

Tangent^b

الگوریتم آنالیزگر دیجیتال تفاضلی

الگوریتم آنالیزگر دیجیتال تفاضلی^۸ یک الگوریتم ساده لوحانه^۹ برای بدست آوردن خط در فضای دیجیتال میباشد. چرا به این الگوریتم، ساده لوحانه میگوییم؟ چون کند بوده، و ساده لوحانه است که فکر کنیم، با اولین نتیجه ای که به به ذهنمان میرسد، بهترین نتیجه است! این الگوریتم اینگونه تعریف میشود:

۱. دو نقطه‌ی ش (X_1, Y_1) و (X_0, Y_0) را در نظر بیرید.

۲. تفاضل بین دو نقطه را حساب کنید.

$$dx = X_1 - X_0$$

$$dy = Y_1 - Y_0$$

۳. اگر dx از dy بزرگتر بود، آنگاه گام^{۱۰} بیشتری در محور افقی لازم دارید. در صورت هکس این ماجرا، ما گاهای بیشتری در محور عمودی لازم داریم. ازینجا بعد، وارد بخشی میشویم که به آن الگوریتم ساده لوحانه میگویند. اول بگذارید ادامه بدهیم.

۴. افزایشگر^{۱۱} برای هم X و هم Y را حساب کنید.

۵. پیکسل را در صفحه رسم کنید.

متوجه نشیدید؟ عیب ندارد. قصدم این بود که با مثال، این را آموزش دهم. پس بگذارید مثال را بیاورم. این اولین کد پیچیده‌ی کتاب است، نگران نباشید، ازین بعد کدهای ما ازین پیچیده تر خواهند شد!

Digital Differential Analyzer^۸

Naive^۹

Step^{۱۰}

Increment^{۱۱}

۳.۵ خط راست

۸۱

```
from PIL import Image

img = Image.new('RGB', (64, 64), color= (0, 0, 0)) #I
pixels = img.load()

coord_1 = (2, 5) #II
coord_2 = (56, 32) #III

dx = coord_2[0] - coord_1[0] #IV
dy = coord_2[1] - coord_1[1] #V

Steps = 0

abs_dx = abs(dx) #VI
abs_dy = abs(dy)

if abs_dx > abs_dy:
    Steps = abs_dx #VII
else:
    Steps = abs_dy

Xincrement = dx / float(Steps) #VIII
Yincrement = dy / float(Steps)

x, y = coord_1[0], coord_1[1]

for i in range(Steps):
    x = x + Xincrement
    y = y + Yincrement

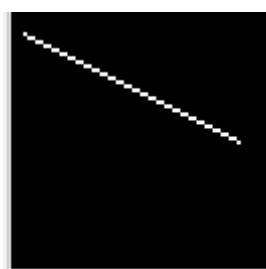
    pixels[int(x), int(y)] = (255, 255, 255) #IX
```



```
img.save('Line-DDA.png')
```

Listing 5.4: Line-DDA.py

- (I) ساخت عکس و رستر
 - (II) مختصات نقطه‌ی یک
 - (III) مختصات نقطه‌ی دو
 - (IV) تفاضل افقی
 - (V) تفاضل عمودی
 - (VI) مثبت سازی (قدر مطلق) تفاضلات
 - (VII) تعیین گامها
 - (VIII) تعیین افزایشگر
 - (IX) سیاه کردن پیسکل در موقعیت
- حاصل ازین کد، این تصویر است:



شکل ۷.۵: خط DDA

مشکل این الگوریتم، اینست که تقسیم اعشاری استفاده می‌کند. و این باعث می‌شود سرعت نرم افزار کم شود. برای همین به این الگوریتم، ساده‌لواحانه می‌گویند. دو الگوریتم بهتر برای احتساب خط وجود دارد که به بررسی آنها می‌پردازیم.

الگوریتم تحصیل خط برسنهم

جک التن برسنهم^{۱۲} متخصص شرکت IBM در سال ۱۹۶۲، برای رسم خط در رستر، یک الگوریتم ساخت که تا به امروز استفاده میشود.

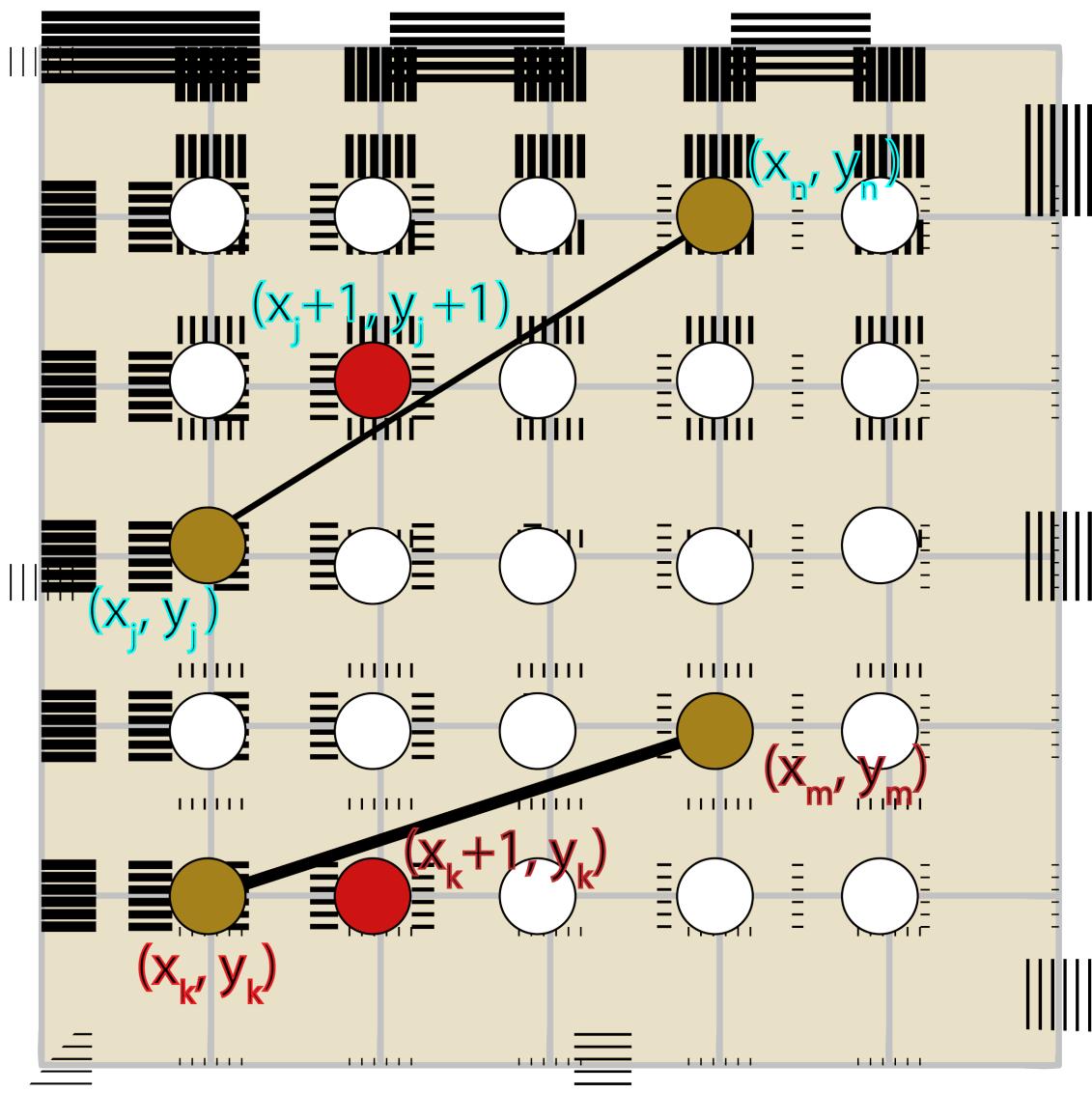


شکل ۸.۵: یک عکس قدیمی از برسنهم

برتری این الگوریتم نسبت به الگوریتم DDA اینست که، این الگوریتم، کاملا بر پایه ای اعداد صحیح مثبت برپاست. و نه اعداد حقیقی که اعشار دارند.

۱. به ما ابتدا و انتهای خط را میدهند. بگذارید به ابتدای خط (x_k, y_k) و به انتهای خط (x_m, y_m) بگوییم.

۲. به ازای هر پیکسلی که بین را ماست، باید تصمیم بگیریم که به سمت $(x_k + 1, y_k)$ رفته، یا به سمت $(x_k + 1, y_k + 1)$.



شکل ۹.۵: خط با الگوریتم برسنهم

این الگوریتم را اینگونه در پایتان مینویسیم:

```
from PIL import Image

img = Image.new('RGB', (64, 64), color= (0, 0, 0))
pixels = img.load()
```

```

coord_1 = (2, 5)
coord_2 = (56, 32)

dx = coord_2[0] - coord_1[0]
dy = coord_2[1] - coord_1[1]

m = 2 * dy #I

Slope_error = m - dx #II

y = coord_1[1]
for x in range(coord_1[0], coord_2[0] + 1):

pixels[x, y] = (255, 255, 255) #III

Slope_error = Slope_error + m #IV

if (Slope_error >= 0):
y += 1 #V
Slope_error = Slope_error - (2 * dx) #VI

img.save('Line-Bresenham.png')

```

Listing 5.5: Line-Bresenham.py

بخشی از کد که با کد قبلی مشترک است را مورد بررسی قرار نخواهیم داد.

(I) به دست آوردن شیب

(II) به دست آوردن ضریب خطای شیب

(III) سیاه کردن پیکسل.

(IV) به دست آوردن ضریب خطای جدید.

(V) اضافه کردن ۱ به ،^a به شرط اینکه ضریب خطای شبیب بیشتر از ۰ باشد.

(VI) حساب کردن دوباره ای ضریب خطای و از بین بردن اعشار آن.

میتوانید برای فهم بهتر کدها، آنها را **دیباگ**^a کنید. برای اینکار یا از ابزار دیباگ محیط گسترشتای استفاده کنید، یا اینجا و آنجا از تابع `print()` برای نشان دادن ارقامی که نمیدانید در روی صفحه استفاده کنید.

Debug^a

گفتیم روش برسنهم هنوز کاربرد دارد. اما کاربرد آن بیشتر در پلاترها و صنعت چاپ است. نرم افزارهایی مثل فتوشاپ از **ضدپلگی**^{۱۳} استفاده میکنند. هنوز وقت آن نرسیده که در مورد ضدپلگی صحبت کنیم. در اواخر بخش اول کتاب در مورد آن صحبت خواهیم کرد.

۴.۵ ترسیم دایره

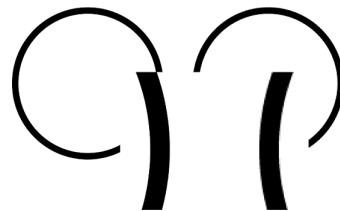
در فضای رستر، نمیتوانیم یک دایره ای صحیح و پراز انحنای داشته باشیم. اما میتوانیم با حقه هایی خاص، یک دایره در فضای رستر رسم نماییم. دو الگوریتم برای اینکار وجود دارد، الگوریتم دایره ای برسنهم، و الگوریتم دایره ای میان نقطه ای. ابتدا به الگوریتم برسنهم میپردازیم.

الگوریتم دایره ای برسنهم

در بخش قبل در مورد برسنهم و اینکه چگونع میتوان از الگوریتم رسم خط وی استفاده کرد ف صحبت کردیم. اما درین بخش میخواهیم در مورد الگوریتم رسم دایره ای وی صحبت کنیم.

ابتدا با استفاده از یک دایره ای کامل، مفهومش را توضیح داد. اما یک دایره ای کامل در مانیتور یک **پادآمیزه**^{۱۴} میباشد. در عین حال، چون الگوریتم رسترایزشن نرم افزار ایلستریور از همه ای الگوریتمها قویتر است، از ایلستریتیر برای نمایش یک دایره ای کامل در برابر یک دایره که قرار است رسم کنیم، استفاده خواهیم کرد.

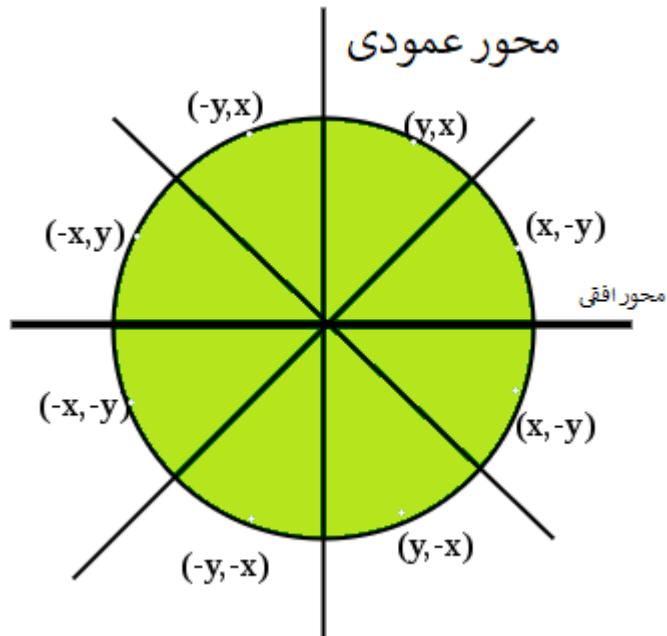
Antialiasing^{۱۳}
Oxymoron^{۱۴}



شکل ۱۰.۵: یک دایره‌ی برابری در برداری یک دایره‌ی رستراایز شده

همانطور که مشاهده میکنید، لبه‌های یک دایره‌ی رستراایز شده پله‌ای میباشد. همانطور که گفتیم، الگوریتمهای برای ضد پلگب وجود دارند که بعدا در موردش بحث خواهیم کرد.

در دایره‌ی برسنهم، ما احتیاج به هشت نقطه اصلی داریمو و ازین هشت نقطه‌ی اصلی، تا شزایطمان برقرار است، یک بار به سمت شرق، و یک بار به سمت جنوب شرق، رفته و آنقدر اینکار را تکرار میکنیم تا دایره‌ی کشیده شود. هشت نقطه‌ی اولیه بدین قرارند:



شکل ۱۱.۵: هشت نقطه‌ی لازمه