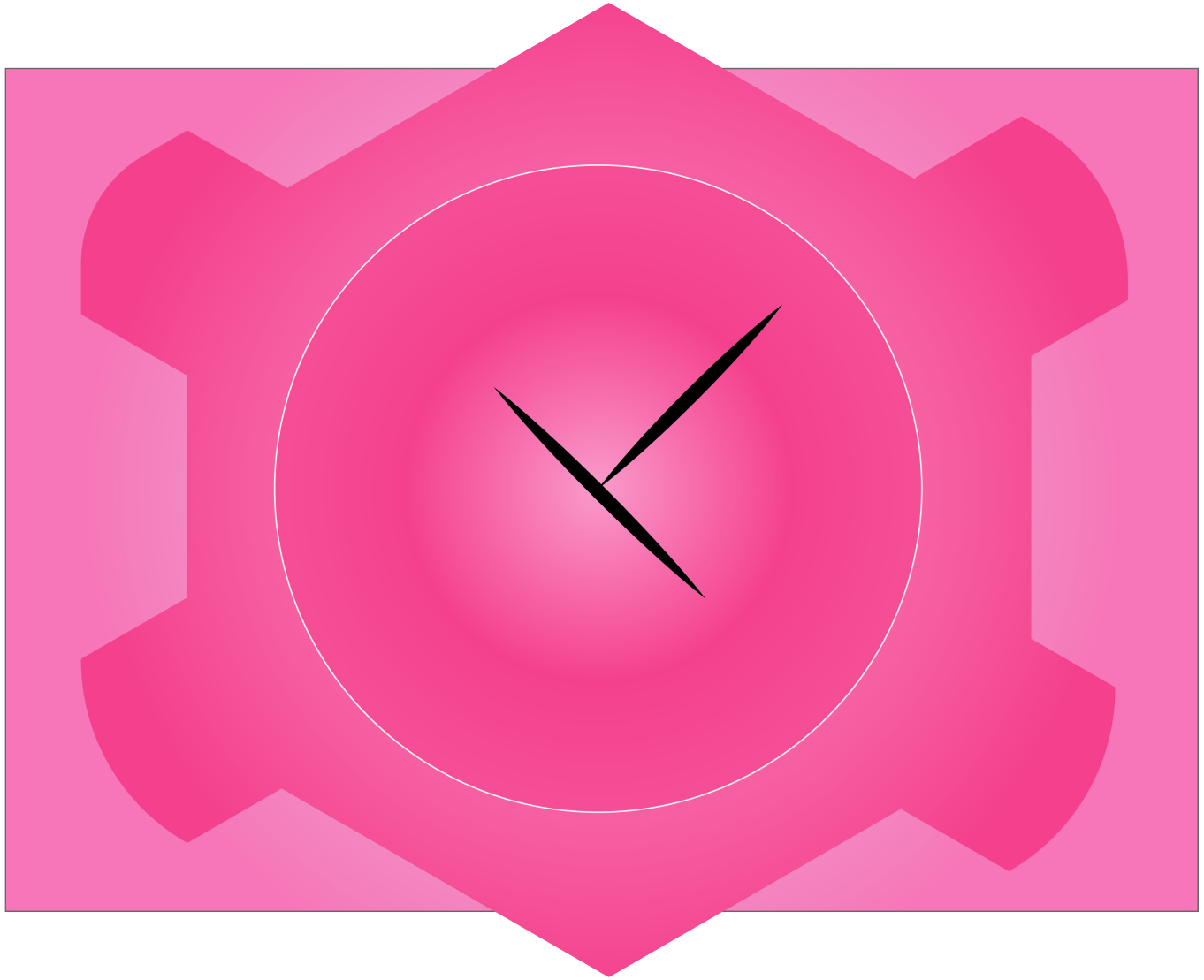
بازی سازی و برنامه نویسی خلاقانه پایه

چوبک بیدپا

):



[۱] Passion Project

[۲] با بیشتر شدن تعداد خریدارها، قیمت کاهش میابد.

قراردادهای کتاب

۱. بخشهای کتاب: این کتاب به دو بخش برنامه نویسی خلاقانه[۳]، و بازی سازی تقسیم شده است.

۲. نکته: نکته های 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 پررنگ: وقتی لازم است روی کلمه ای تاکید کنم، یا کلمه جدید است و قبلا استفاده نشده، از **متن پررنگ** استفاده خواهد کرد.

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

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

[۳] Creative Programming

فهرست مطالب

فصل ۱

چند کلمه با خواننده

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

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

این کتاب نه تنها بر پایه ی عقیده به آزادی اطلاعات[۵] مجانی است، بلکه یکی از دلایل مجانی بودن آن اینست که تمام آن مال من نیست، بلکه، حدود ۳۰% این کتاب، ترجمه ی آموزشهای اینترنت، با اجازه از صاحبان آنهاست. ۲۰% این کتاب، از داکیومنتشنهای رسمی برداشته شده و ۵۰ درصد باقی را خودم نوشته ام.

شاید برایتان سوال باشد چرا این کتاب را نگاشت کرده ام. دلیل اصلی آن اینست که دلیلی داشته باشم تا برنامه نویسی را ادامه دهم. بعضی ها پروژه مینویسند، بعضی ها کتاب مینویسند. من در لفافه ی کتاب، پروژه مینوسم. تمام پروژه های کتاب اریجینال بوده، و فایلهایی که همراه کتاب خریده اید، کار من هستند.

دلیل دیگری که این کتاب را نوشته ام، اینست که کتاب های بازی سازی به زبان فارسی کم هستند، و کمتر کسی در ایران از برنامه نویسی خلاقانه خبر دارد. سعی من درینست که با نوشتن در مورد این دو دیسیپلین دوست داشتنی، فرهنگ آنها را در کشور اشاعه بدهم.

---

[۱] GNU

[۲] Linux

[۳] Mozilla

[۴] Firefox

[۵] Freedom of Information

از سابقه ام در برنامه نویسی و بازی سازی بگویم. من از شانزده سالگی 25 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 اسکی مانند بلک جک نیز نوشته ام. من زبانهای سی پلاس پلاس، پایتان، و سی را میدانم و با زبان اسکریپت نویسی چندین نرم افزار آشنایی دارم. درضمن نمره ی تافلم در ۱۷ سالگی ۹۵ بوده پس به ترجمه ام اعتماد کنید.

سابقه ی من در برنامه نویسی خلاقانه کمتر است. دو سال پیش بود که با نرم افزار افتر افکتس[۶] آشنا شدم و به صرافت نوشتن پلاگین برایش افتادم، و طی این امر، با کتابخانه ی Cinder برای سی پلاس پلاس آشنا شدم. و از آنجا بود که با زبان Processing و شیدر ها آشنا گردیدم. الان تسلط کافی برای آموزش پایه ی شیدرها و زبانها و کتابخانه های برنامه نویسی خلاقانه دارم.

بگذارید در مورد چارچوب کتاب کمی صحبت کنم. در این کتاب، دو بخش داریم، برنامه نویسی خلاقانه، و بازی سازی که به دو بخش Asset و برنامه نویسی تقسیم میشود. در بخش اَسِت سعی شده با استفاده از برنامه های 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 کتابخانه ی Arcade پایتان، کتابخانه ی SFML سی پلاس پلاس، و انجین Godot آموزش داده خواهد شد. اگر قرصت 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 وظیفه ی خود خواننده است که این دو را از قبل یاد داشته باشد، اما فقط در حد یادآوری، در مورد این دو حرف خواهم زد.

من تازه نوشتن این کتاب را شروع کرده بودم و همین الان برای من موهبتی بوده است، چون باعث شد لِیتِک را یاد بگیرم و مطمئنم در طول کتاب، من بیشتر از شما خواهم آموخت! و آیا اینچنین نیست که همه، برای یادگیری، درس میدهند؟ هر کلمه ای که یاد میدهی، دو کلمه می آموزی. و این یک موهبت است.

در آخر، در این دنیای پر هیر و گیر، اگر پَشِنی دارید که به شما آرامش میدهد، نیکوست. و اگر این کتاب برای پیدا کردن این پَشِن کمک میکند، خوشحالم.

چوبک بیدپا مشهد ـ ۱۳۹۷

---

[۶] After Effects

فصل ۲

نگاهی کوتاه به ریاضی لازمه

۱.۲ توابع

یک تابع[۱] **به صورت زیر نشان داده میشود:**

$$y = f(x)$$

وظیفه ی یک تابع، تغییر عدد داده شده بر اساس قوانین داده شده است. به این قانون، تابع میگویید. مثلا تابع $f(x) = x^2$ که به آن تابع مربع می گویند، وظیفه اش بردن عدد به توان دو است. به عکس تابع، تابع **معکوس**[۲] میگویند و به صورت زیر نشان داده میشود:

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

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

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

[۱] Function
[۲] Inverse

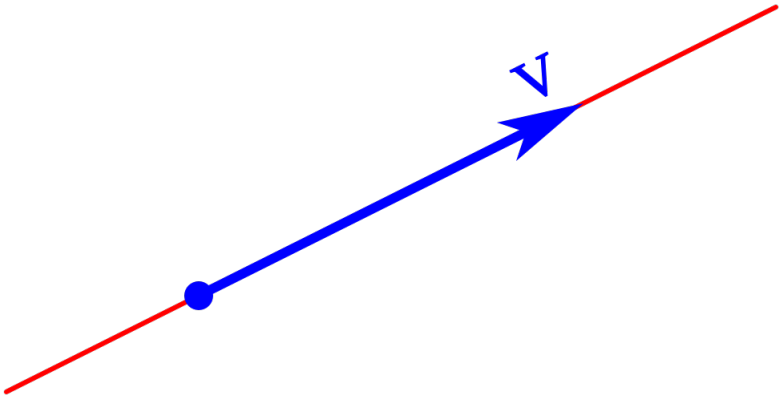
$$(\frac{f}{g})(x) = \frac{f(x)}{g(x)}$$

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

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

۲.۲ بردارها

اگر فضای دوبعدی را به دو بخش نقاط افقی و نقاط عمودی تقسیم کنیم، **بردار**[۵] خطی است که چهار نقطه را به هم وصل میکند.



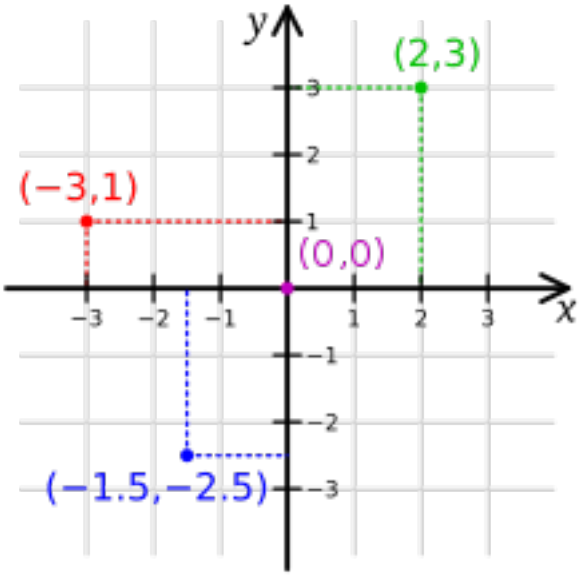
بردار را به صورت زیر نشان میدهند:

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

صفحه ی مختصات را به این صورت میکشیم:

---

[۳] Domain
[۴] Range
[۵] Vector



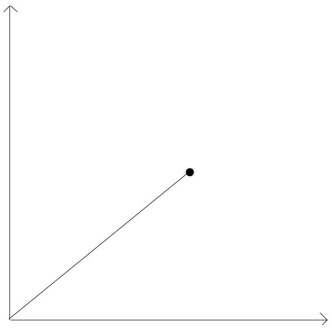
که به آن **دستگاه مختصات دکارتی**[۶] میگویند. در دستگاه مختصات دکارتی، دو محور X و Y به ترتیب محور افقی و عمودی ما را تشکیل می دهند. ما مختصات یک نقطه را در پرانتز به صورت (X, Y) نشان میدهیم. همانطور که گفته شد، خطی که دو نقطه را به هم وط کند، بردار نام دارد.

در جبر خطی، بردار را به صورت:

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

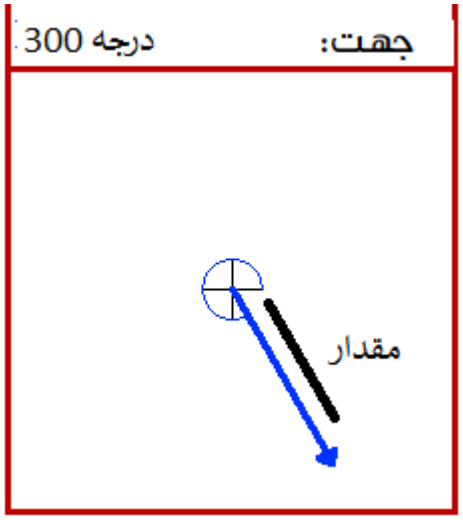
نشان داده میشود و نقطه ی اول آن، مسقط الرأس دستگاه مختصات یعنی $(0, 0)$ میباشد.

[۶] Carthesian Coordinate System



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. بردار ها 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:

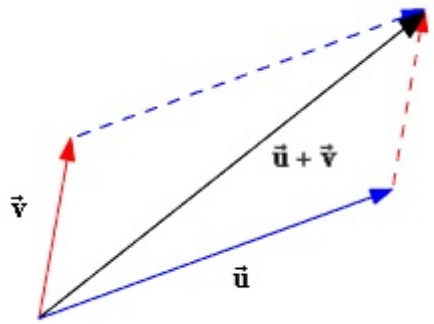


برای به دست آوردن مقدار بردار ازین فرمول استفاده میکنیم:

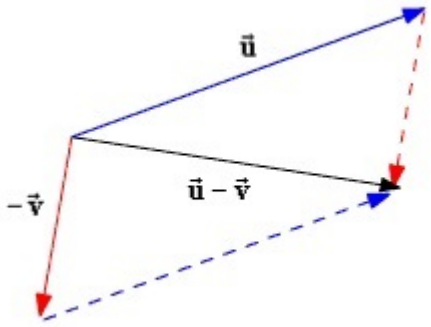
$$|\vec{V}| = \sqrt{x^2 + y^2}$$

دو بردار را میتوان به صورت زیر جمع کرد:

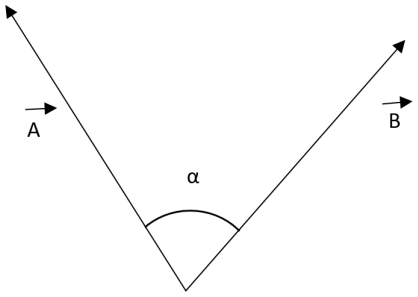
[۷] Direction
[۸] Magnitude



و به این صورت تفریق کرد:



اما دو نوع ضرب برداری داریم. **ضرب نقطه ای**[۹] و **ضرب صلیبی**[۱۰]. قبل ازین که پیش بروید، قسمت مثلثات 2.3 را بخوانید. فرض کنید دو بردار به صورت زیر هستند:



[۹] Dot Product
[۱۰] Cross Product

ضرب نقطه ای به صورت زیر تعریف میشود:

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

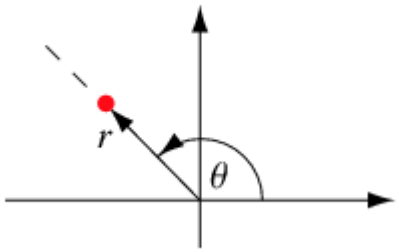
و ضرب صلیبی ازین فرمول استفاده میکنیم.

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

که $\vec{n}$ برداری واحد عمود بر دو بردار است. برای به دست آوردن $\vec{n}$ کافیست از انگستان وسط، اشاره، و شصت خود استفاده کنید. انگشت شصت شما، همواره بردار واحد عمود است، که مضربی از ضرب صلیبی دو بردار می باشد.

۳.۲ مثلثات

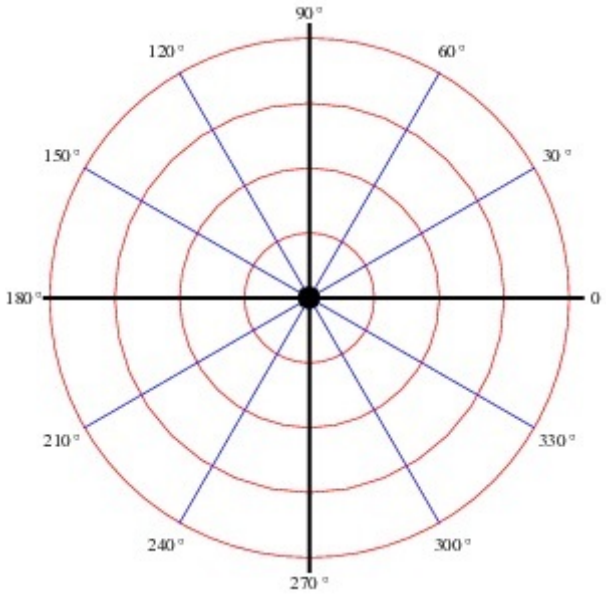
مثلثات بحثیست پیپیده. و من نیز ریاضیدان نیستم پس به کمی در مورد این مبحث قناعت میکنیم. قبل از هرچیزی، بگذارید در مورد **دستگاه مختصات قطبی**[۱۱] حرف بزنم. دستگاه مختصات قطبی، مانند دستگاه مختصات دکارتی، دارای دو محور عمودی و افقی است. اما در این دستگاه مختصات، ما یک نقطه را، عوض X و Y توسط یک زاویه α و یک بردار شعاع $\vec{r}$ نشان میدهیم:



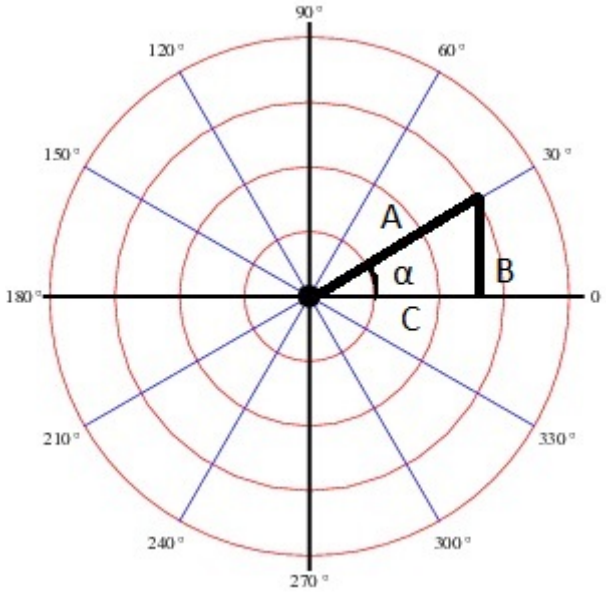
در برنامه نویسی خلاقانه، دستگاه مختصات قطبی کاربردهای زیادی دارد. اما در کامپیوتر، **پیکسلها**[۱۲] در دستگاه مختصات دکارتی قرار دارند. حلال مشکلات ما، مثلثات است. یک دایره ی واحد را در دستگاه مختصات قطبی کنید که شعاعش ۱ میباشد:

[۱۱] Polar Coordinate System

[۱۲] در مورد پیکسلها به وفور حرف خواهم زد.



اگر زاویه ی $30°$ را انتخاب کرده و یک مثلث قائم الزاویه دور آن بکشیم:



سینوس و کسینوس زاویه ۳۰ درجه که اینجا α نامیده میشود، به صورت زیر تعریف میگردد:

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

و:

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

همچنین تانژانت و کتانژانت به صورت زیر تعریف میشوند:

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

و

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

درجه، تنها واحد اندازه گیری زاویه نیست. واحد دیگر، **رادیان**[۱۳] نام دارد. یک زاویه در رادیان، بین 0 و2Π قرار دارد. ارزش Π حدود 3.1415926535 است. ما برای کار با پیکسلها، ارقام اعشار بیشتر ازین نیز نیازمندیم. برای تبدیل درجه به رادیان:

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

، و بالعکس:

$$nrad \times \frac{180}{\pi}$$

توابع مثلثاتی توسط **هویتهای مثلثاتی**[۱۴] 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 سرفصهلایی هستند که شما برای این کتاب لازم دارید. توجه کنید، این کتاب، نه برنامه نویسی خلاقانه و بازی سازی کلی.

---

[۱۳] Radians

[۱۴] Trigonometric Identities

۴.۲ ماتریسها

به آرایه هایی از اعداد که به صورت n سطر و m ستون به نمایش در می آیند.، **ماتریس**[۱۵] میگویند. یک ماتریس را به این صورت نشان میدهند:

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

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

$$M_{m,n} \pm N_{m,n} = \begin{bmatrix} m_{1,1} \pm n_{1,1} & m_{1,2} \pm n_{1,2} & \cdots & m_{1,n} \pm n_{1,n} \\ m_{2,1} \pm n_{2,1} & m_{2,2} \pm n_{2,2} & \cdots & m_{2,n} \pm n_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} \pm n_{n,1} & m_{n,2} \pm n_{n,2} & \cdots & m_{n,n} \pm n_{n,n} \end{bmatrix}$$

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.

$$M_{1,n} \times N_{m,1} = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,m} \end{bmatrix} \times \begin{bmatrix} n_{1,1} \\ n_{2,1} \\ \vdots \\ n_{n,1} \end{bmatrix} = (m_{1,1} \times n_{1,1}) + (m_{1,2} \times n_{2,1}) + \cdots + (m_{1,n} + n_{m,1})$$

یکی دیگر از عملیتهای ماتریسی، **دترمینان**[۱۶] است. برای احتساب دترمینان ماتریسهای بزرگتر از 3×3الگوریتمهای زیادی مانند **دیکامپوزیشن**[۱۷] وجود دارد که خود آن توسط افراد مختلفی در طول سالها بهسازی گشته اند، اما راه ساده ای برای به دست آوردن دترمینان 2×2 وجود دارد که به شرح زیر است:

---
[۱۵] Matrix
[۱۶] Determinant
[۱۷] Decomposition

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

$$|A| = AD - BC$$

به I_n ماتریس **هویت** [۱۸] میگویند و مثلا I_3 به صورت زیر تعری میشود:

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

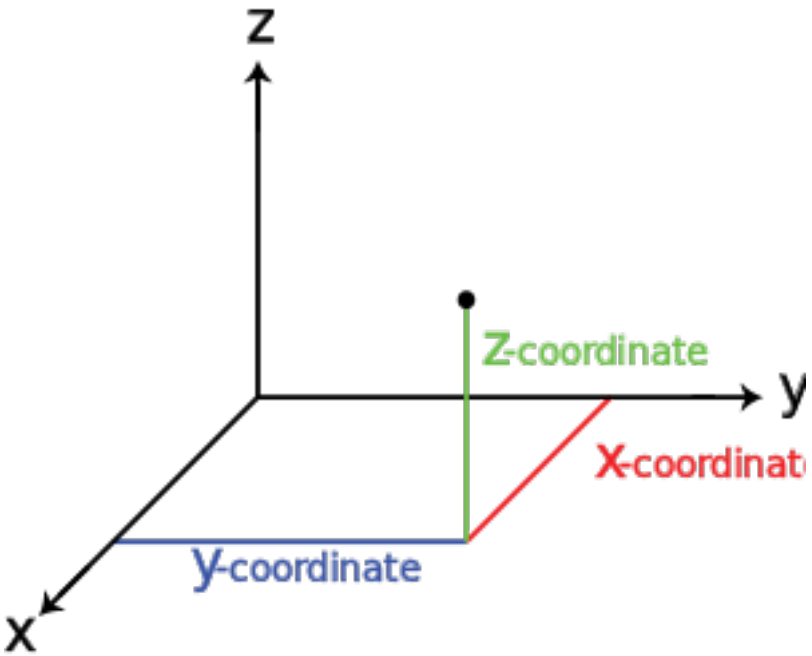
ما در برنامه نویسی خلاقانه و بازی سازی از ماتریسها استفاده های زیادی خواهیم برد.

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 اظافه کردن یک بردار اضافه که نام آن Z است به دستگاه سه بعدی دست پیدا کنیم. این دستگاه را به صورت $\mathbf{R^3}$ نشان میدهند و در تصویر زیر میتوانید محور Z را مشاهده کنید:

---

[۱۸] Identity



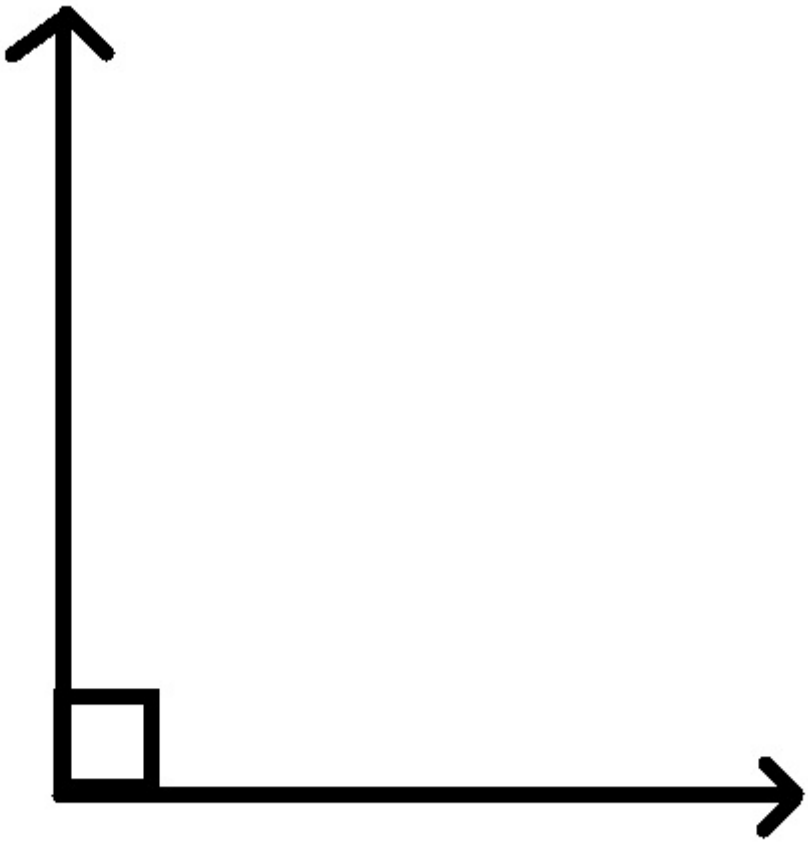
توجه کنید که در بعضی از نرم افزارها جای Y با Z عوض میشود.

یک بردار را در فضای R^3 به صورت زیر نشان میدهیم:

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

همه ی قوانین R^2 برای R^3 برقرار است. مثلا به بردار واحد محور Z، $\vec{k}$ میگویند. غرض از این بخش، اینست که **قائمیت**[۱۹] در فضای سه بعدی را معرفی کنم. زیرا برای بازیهای دو و نیم بعدی، دوربین باید قائم بر فضای R^3 باشد.

[۱۹] Orthogonality



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

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

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

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

و $cos(90) = 0$ پس:

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

۶.۲ پایان فصل ریاضی

کلید یادگرفتن ریاضی یک چیز است: ^{n}تمرین! یادتان نرود که حافظه، چیزیست فرَار، و هر لحظه ممکن است همین چیزهای کمی که از بنده ی حفیر 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 حافظه ی شما رخت برمیبندند. تمرین کنید، نوت برداری کنید، و یادتان نرود که روم را در یک روز نساخته اند. 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 **پَرَدایم**[۲] برنامه نویسی **فانکشنال**[۳] و **شیء گرا**[۴]. و بعد **پیچیدگی زمانی**[۵] را توضیح خواهیم داد. بعد از آن، نگاهی می اندازیم به **سینتکسِ**[۶] زبان **پایتان**[۷] و C++.

۱.۳ برنامه نویسی فانکشنال

از بین تمام روشها، یا به عبارتی، پَرَدایمهای برنامه نویسی، برنامه نوسی 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 **لیسپ**[۹] همه فانکشنال هستند.

با توابع در فصل ریاضی آشنا شدیم. توابع کامپیوتری نیز با توابع ریاضی فرق زیادی ندارد، همه ی آنها یک ماشین هستند که ورودی را به خروجی تبدیل میکنند. یک تابع، مجموعه ای از **دستورات**[۱۰] است که **پارامتر**[۱۱] داده شده را با تغییرات، باز میگردانند. این تغییرات میتواند

---

[۱] Programming
[۲] Paradigm
[۳] Functional
[۴] Object Oriented Programming
[۵] Time Complexity
[۶] به دستور زبانی یک زبان برنامه نویسی Syntax گفته میشود.
[۷] Python
[۸] FORTRAN
[۹] Lisp
[۱۰] Instructions
[۱۱] Parameter

عملیاتهای جمع و تفری، ضرب و تقسیم، باقیمانده، و یا تغییر نوع پارامتر مثلا از عدد صحیح به عدد حقیقی، و یا هرچیز دیگری باشد. برای اجرای تابع، آنرا **میخوانیم**[۱۲] و به پارامتری که به آن میدهیم، **آرگومان**[۱۳] میگوییم.

اینستراکشن سِت زیر را در نظر بگیرید:

۱. عدد n را بگیر.

۲. برای n بار، n را ضربدر $n-1$ کن.

۳. جواب را برگردان.

به این تابع، تابع **فاکتوریل**[۱۴] 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 نمی گردانند. به این نوع از توابع **وویید**[۱۵] میگویند. بعضی از زبانها، **تایپ ثابت**[۱۶] هستند و باید نوع ارزشهای باز گرداننده را مشخص کرد. C++ یکی ازین نوع زبانهاست. بعضی از زبانها **تایپ دینامیک**[۱۷] 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 پَرَدایم شیء گرا در بخش بعد صحبت خواهیم کرد. هرزبان مقداری تابع از پیش تغیین شده دارد، اما بقیه ی تابع ها 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. همه ی زبانها دارای **کتابخانه**[۱۸] هایی هستند که شامل توابع و کلاسها 3.2 و دیگر کدهایی هستند که به برنامه نویس کمک میکنند خود را تکرار نکند.

خود را تکرار نکنید.[۱۹]

---

[۱۲] Call
[۱۳] Argument
[۱۴] Factorial
[۱۵] Void
[۱۶] Statically Typed
[۱۷] Dynamically Typed
[۱۸] Library
[۱۹] DRY - Don’t Repeat Yourself

قانون پلاتینیوم برنامه نویسی، اینست. هرگز چیزی که در یک کتابخانه موجود است را ننویسید. مثلا عوض اینکه در زبان **جاوا**[۲۰] عوض نوشتن صدها خط کد برای به دست آوردن یک تابع ماتریس، میتوان از کتابخانه ی JAMA استفاده کرد.

۲.۳ برنامه نویسی شیء گرا

[۲۰] Java