چالش طراحی raytracer
(گام اول)

چکیده

هدف از این چالش، طراحی یک raytracer ساده‌ست که به صورت گام به گام انجام می‌شه. توی گام اول، با ریاضیات مورد نیاز برای بردارها، نقاط و احجام سه‌بعدی آشنا می‌شیم و نگاهی به روش تصویر کردن یه حجم سیمی (wireframe) میندازیم.

برای شرکت توی این چالش، استفاده از هر زبان برنامه‌نویسی مجازه و فقط تصویرهای نهایی‌ای که آخر هر گام به دست میاد و سورس برنامه برای شرکت توی چالش لازمه؛ پیشنهاد می‌کنم تا کدی که نوشتید رو توی گیتهاب یا جای مناسب دیگه با دوستان‌تون به اشتراک بذارید تا اونها هم ازش بتونن استفاده کنن.

۱ یادآوری

قبل از پرداختن به مطلب اصلی، نیاز داریم تا بعضی از مفاهیم ریاضیات برداری و هندسه تحلیلی رو که توی دبیرستان و دانشگاه یاد گرفتیم، یادآوری کنیم. تسلط به این مطالب برای انجام دادن گام‌های این چالش اساسیه.

یه *بردار آزاد*، $\vec{v}$،[۱] یه چهارتایی از مقادیر اسکالر (x_v, y_v, z_v, w_v) ـه، که سه‌تای اول به‌ترتیب اندازه بردار $\vec{v}$ رو روی هر یک از محورهای متناظر در فضا مشخص می‌کنن و چهارمی همیشه مقدارش برابر با یکه.[۲]

طول یه بردار آزاد، $|\vec{v}|$، رو می‌تونیم با کمک رابطه فیثاغورث به دست بیاریم:

$$|\vec{v}| = (x_v^2 + y_v^2 + z_v^2)^{1/2}$$

اگه طول یه بردار آزاد برابر با یک باشه، بهش یه بردار یکه می‌گیم و معمولاً با $\vec{u}$ نشونش می‌دیم. بردار یکه متناظر با هر بردار آزاد دلخواه $\vec{v}$ رو می‌تونیم از این رابطه به دست بیاریم:

$$\vec{u_v} = \frac{\vec{v}}{|\vec{v}|}$$

[۱] بردارهای آزاد رو همیشه با حروف کوچیک انگلیسی نمایش می‌دیم.

[۲] مقدار w_v نمود حقیقی نداره و فقط توی محاسبات ماتریسی کمک‌مون می‌کنه

عملگر *ضرب داخلی* دو بردار آزاد $\vec{v}$ و $\vec{w}$ رو، $\vec{v}\cdot\vec{w}$، به شکل زیر تعریف می‌کنیم. حاصل ضرب داخلی دو بردار آزاد یه مقدار اسکالره که به کمک مثلثات و با دونستن زاویه بین دو بردار، θ، هم قابل محاسبه‌ست:

$$\vec{v}\cdot\vec{w} = x_v \cdot x_w + y_v \cdot y_w + z_v \cdot z_w = |\vec{v}||\vec{w}|\cos\theta$$

یه *بردار ثابت*، $\vec{P}$، برداریه که نقاط ابتدایی و انتهایی مشخصی داره.[۳] به طور مثال اگه بردار $\vec{P}$ دارای نقاط ابتدایی و انتهایی A و B باشه، اون وقت داریم:

$$\vec{P} = (x_B - x_A, y_B - y_A, z_B - z_A, 1)$$

همچنین اگه نقطه ابتدایی یه بردار ثابت روی مبدأ محور مختصات قرار داشته باشه، بهش یه *بردار مکانی* می‌گیم. با این تعریف، هر نقطه توی فضا یه بردار مکانی منحصربه‌فرد داره. بردار یکه و ضرب داخلی بردارها به طریق مشابهی که گفته شد، برای بردارهای ثابت و مکانی هم قابل تعریفه.

علاوه بر تعریف‌های بالا، به بردار یکه‌ای که عمود بر یه صفحه توی فضا باشه، اصطلاحاً *نرمال* اون صفحه می‌گیم و معمولاً اون رو با $\vec{n}$ نمایش می‌دیم.

ماتریس *جابه‌جایی*، M_t، ماتریسیه به شکل زیر، که در صورت ضرب شدن در یک بردار مکانی، اون رو به اندازه بردار $(x_t, y_t, z_t, 1)$ جابه‌جا می‌کنه:

$$M_t = \begin{bmatrix} 1 & 0 & 0 & x_t \\ 0 & 1 & 0 & y_t \\ 0 & 0 & 1 & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

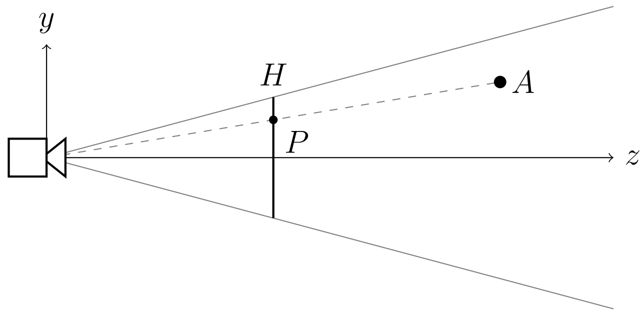
ماتریس چرخش، M_r، ماتریسیه که در صورت ضرب شدن در یک بردار مکانی، اون رو حول یکی از محورهای مختصات به اندازه زاویه θ و در جهت راستگرد دوران می‌ده. بنابراین برای هر کدوم از محورهای مختصات، یه ماتریس چرخش مجزا داریم:

$$M_r^x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

[۳] بردارهای ثابت و مکانی رو همیشه با حروف بزرگ انگلیسی نمایش می‌دیم.



شکل ۱: تصویر کردن نقطه برروی صفحه مجازی

۲ تصویر کردن

علی‌رغم اینکه مغز ما دنیای بیرون رو به صورت سه‌بعدی درک می‌کنه، اما چشم ما دنیای بیرون رو به صورت دوبعدی می‌بینه. این در مورد دریچه لنز یه دوربین هم صادقه. چشم ما، یا دوربین، این عمل رو با تصویر کردن اجسام سه‌بعدی برروی یه صفحه مجازی انجام می‌ده.

در شکل ۱ فرایند تصویر کردن یه نقطه فرضی، A، رو روی صفحه مجازی H می‌بینیم. توی این شکل، به فاصله صفحه H تا دوربین *فاصله کانونی* می‌گیم و با f نشونش می‌دیم، و برای سادگی ارتفاع صفحه H رو همیشه برابر با یک می‌گیریم. با این مفروضات، مختصات تصویر نقطه A بر روی صفحه H به دست میاد:

$$y_P = f\frac{y_A}{z_A}$$

اگه همین محسابات رو توی سه‌بعد برای محور x (عمود بر صفحه) هم انجام بدیم، داریم:

$$x_P = f\frac{x_A}{z_A}$$

یا به شکل کلی‌تر می‌تونیم بنویسیم:

$$\begin{bmatrix} x_P \\ y_P \\ 1 \end{bmatrix} = \frac{1}{z_P} \times \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x_A \\ y_A \\ z_A \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{x_A}{z_A} \\ f\frac{y_A}{z_A} \\ 1 \end{bmatrix}$$

به ماتریس میانی توی این محسابات، *ماتریس پروجکشن* می‌گیم. اما این فرمول فقط زمانی صحت داره که دوربین روی مبدأ و در راستای محورهای مختصات قرار بگیره، اما دوربینی که ما مد نظر داریم، ممکنه در هر نقطه‌ای از فضا و با هر راستای دلخواهی قرار داشته باشه.

برای این منظور دو مختصات مجزا تعریف می‌کنیم. *مختصات جهانی* مختصاتیه که مرجع محسابات ماست. محورهای مختصات جهانی رو با حروف بزرگ X و Y و Z نمایش می‌دیم و به مبدأ مختصات جهانی به اختصار همون مبدأ مختصات می‌گیم. دقت کنید که توی این تعریف، محور Y محوریه که از نظر بصری روبه‌بالاست (شکل ۲).

در مقابل، *مختصات محلی* رو داریم که نقطه مبدأش برروی محل قرارگیری دوربینه. راستای محور z این مختصات رو سمت نگاه دوربین و راستای محور y رو عمود بر اون و رو به بالای دوربین در نظر می‌گیریم و محور x رو به صورت راستگرد و با نسبت به دو محور z و y تعریف می‌کنیم.

با این تعاریف، نیاز داریم تا بردارهای مکانی هر کدوم از نقطه‌ها رو از مختصات جهانی به بردارهای مکانی توی مختصات محلی تبدیل کنیم. به دست آوردن این تبدیل طولانیه و از حوصله این چالش خارجه، برای همین اینجا فقط تبدیل‌های نهایی رو خلاصه می‌کنیم.

فرض کنید بردار مکانی $\vec{P}$ دارای مختصات $(X_P, Y_P, Z_P, 1)$ توی مختصات جهانی باشه و ما یه مختصات محلی داشته باشیم که مبدأش، $\vec{C}$، در مکان $(C_X, C_Y, C_Z, 1)$ نسبت به مختصات جهانی قرار داشته باشه. در این صورت، بردار مکانی محلی $\vec{P_c}$ رو می‌تونیم به شکل زیر به دست بیاریم:

$$\vec{P_c} = M_r^r \times M_r^y \times M_r^p \times M_t \times \vec{P}$$

توی این رابطه M_t ماتریس جابجاییه که با استفاده از برداری مکانی دوربین، $\vec{C}$، و از رابطه زیر به دست میاد:

$$M_t = \begin{bmatrix} 1 & 0 & 0 & -C_X \\ 0 & 1 & 0 & -C_Y \\ 0 & 0 & 1 & -C_Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

و M_rها ماتریس‌های چرخش دوربینن که از روابط زیر به دست میان:

$$M_r^p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_p & \sin\theta_p & 0 \\ 0 & -\sin\theta_p & \cos\theta_p & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

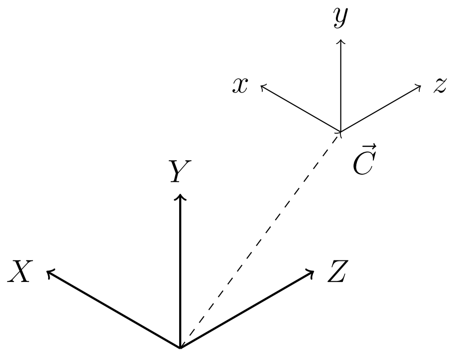
$$M_r^y = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M_r^r = \begin{bmatrix} \cos\theta_r & \sin\theta_r & 0 & 0 \\ -\sin\theta_r & \cos\theta_r & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

توی این تعریف، θ_p و θ_y و θ_r نسبت به وضعیت عادی دوربین، یعنی زمانی که محورهای محلی دوربین با محورهای مختصات جهانی همسوئن، و به‌ترتیب با نسبت با محورهای x، y و z سنجیده می‌شن. در اصطلاح فنی، به این زاویه‌ها به‌ترتیب *زاویه*[۴]، *انحراف*[۵] و *رول*[۶] دوربین هم می‌گیم.

---

[۴] Pitch
[۵] Yaw
[۶] Roll



شکل ۲: مختصات جهانی و مختصات محلی دوربین

شکل ۲ نسبت مختصات جهانی و مختصات محلی دوربین و بردار مکانی دوربین، $\vec{C}$، رو نشون می‌ده. در نهایت، بعد از اینکه مختصات جهانی رو به مختصات محلی دوربین تبدیل کردیم، باید با استفاده از ماتریس پروجکشن، که قبل‌تر در مورد صحبت کردیم، مختصات محلی رو روی صفحه مجازی دوربین تصویر کنیم.

۳ چالش

چالش این گام، نوشتن برنامه‌ایه به زبان و تکنولوژی دلخواه، که با استفاده از روابطی که تا اینجا گفتیم، بتونه یه تصویر سیمی از یه مدل سه‌بعدی و از زاویه دید یه دوربین ایجاد کنه. این برنامه باید اطلاعات مدل و دوربین رو از فایلی متنی بخونه که قالبش رو توی بخش بعدی توضیح دادیم.

برای شرکت توی این چالش باید کدی که نوشتید رو جایی قرار بدید که در دسترس همه باشه، و تصویری رو از خروجی برنامه به ازای تمام دوربین‌ها به اشتراک بذارید.

۴ قالب فایل متنی

فایل متنی مدل شامل سه بخش گره‌ها، وجه‌ها و دوربین‌هاست. بخش گره‌ها که با دستور `VERTICES` شروع می‌شه، شامل یه بردار مکانی در هر خطه که موقعیت یه گره رو مشخص می‌کنه. مثال زیر یه نمونه از بخش گره‌هاست:

```
VERTICES
1.0 0.0 0.0 1.0
1.5 2.1 0.0 1.0
1.0 0.0 3.0 1.0
...
```

دقت کنید که هر بردار شامل ۴ عدد اعشاریه که ۳ تای اول به‌ترتیب معادل X_i، Y_i و Z_i بردار iام نسبت به مختصات جهانی‌ان و عدد آخر همیشه برابر با یکه. توی این بخش هر گره اندیسی داره که برابر با جایگاه ترتیبی اون گره توی فهرست گره‌هاست، و گره اول اندیسش برابر با یکه.

بخش دوم یا بخش وجه‌ها با دستور FACES شروع می‌شه و شامل یک وجه مثلثی به‌ازای هر خطه. مثال زیر یه نمونه از بخش وجه‌هاست:

```
FACES
1 2 3 1.0 0.0 0.0 1.0
1 4 5 0.0 1.0 0.0 1.0
2 5 6 0.0 0.0 1.0 1.0
...
```

تعریف هر وجه شامل ۳ عدد صحیح و ۴ عدد اعشاریه. اعداد صحیح اندیس گره‌هایی‌ان که سه‌گوشه وجه قرار دارن، و ۴ عدد اعشاری بردار نرمال وجه رو مشخص می‌کنن. چینش این سه گره همیشه به صورتیه که به صورت راستگرد نسبت به بردار نرمال در فضا قرار داشته باشن.

بخش نهایی یا بخش دوربین‌ها با دستور CAMERAS شروع می‌شه و شامل تعریف یه دوربین به ازای هر خطه. مثال زیر یه نمونه از بخش دوربین‌هاست:

```
CAMERAS
5.0 6.0 1.0 1.0 30.0 45.0 10.0 0.25 FAR
1.0 0.5 3.0 1.0 20.0 30.0 15.0 0.50 NEAR
...
```

تعریف هر دوربین شامل ۸ عدد اعشاریه، که ۴ عدد اول بردار مکانی دوربین رو نسبت به مختصات جهانی معین می‌کنن. ۳ عدد اعشاری بعدی به‌ترتیب زاویه، انحراف و رول دوربین با واحد درجه‌ان، و عدد هفتم فاصله کانونی دوربینه. در انتهای خط هم یه رشته متشکل از حروف انگلیسی قرار داره که نام منحصربه‌فرد دوربین رو مشخص می‌کنه. دقت داشته باشید که نام دوربین نسبت به بزرگی و کوچیکی حروف حساس نیست.