به نام خدا

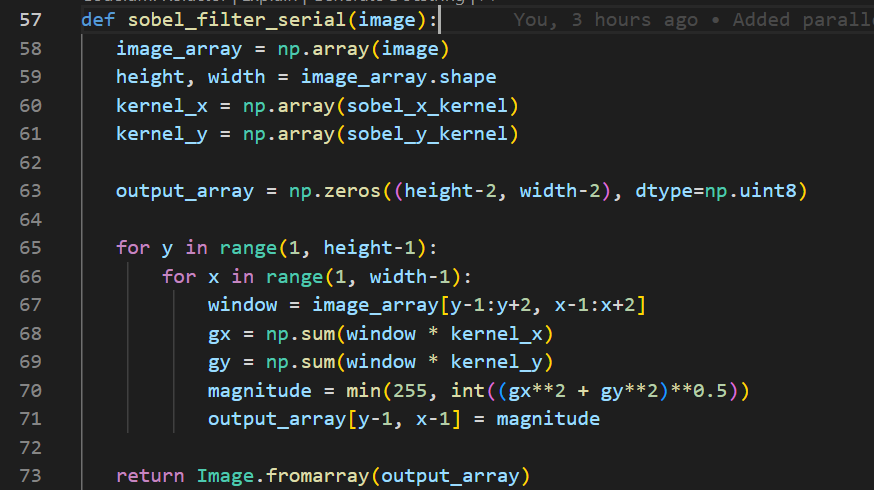
تمرین کامپیوتری پنجم- CUDA

فربد عظیم محسنی 810100187

امیرحسین راحتی 810100144

سوال اول

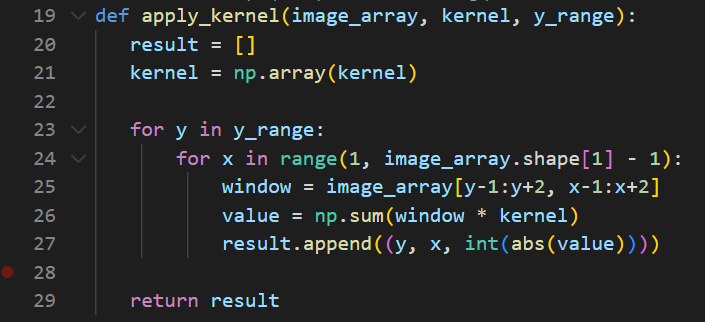
در این قسمت ، ابتدا یک کد پایتون برای بخش سریال آماده میکنیم .

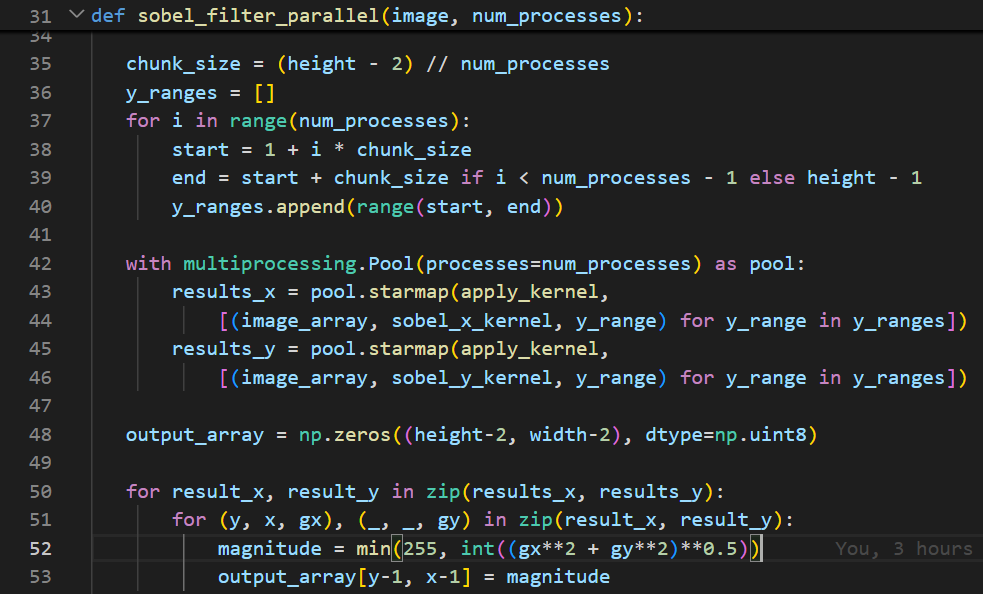


این قطعه کد ، عکس را به وسیله numpy در کرنل ضرب میکند . سپس آرایه حاصل را جمع میکند و در خانه مربوطه میریزد .

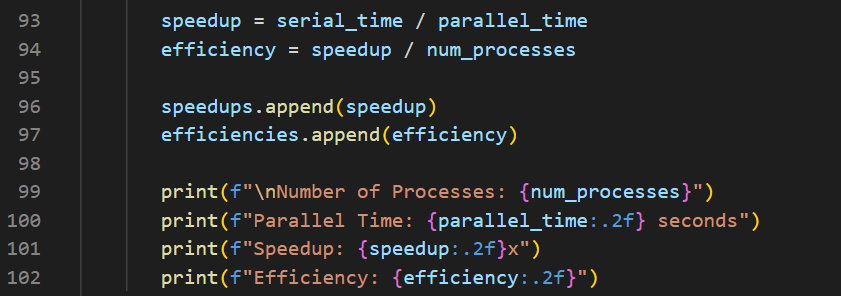
در بخش موازی هم از multiprocessing استفاده میکنیم و به ازای هر قسمت یک پردازه اجرا میکنیم که عملیات را انجام دهد . همچنین این تابع بخش محاسبه ایندکس ها برای دادن به کرنل را انجام میدهد.

یک کرنل هم داریم که بر اساس پارامتر های داده شده به آن محاسبات را انجام میدهد.

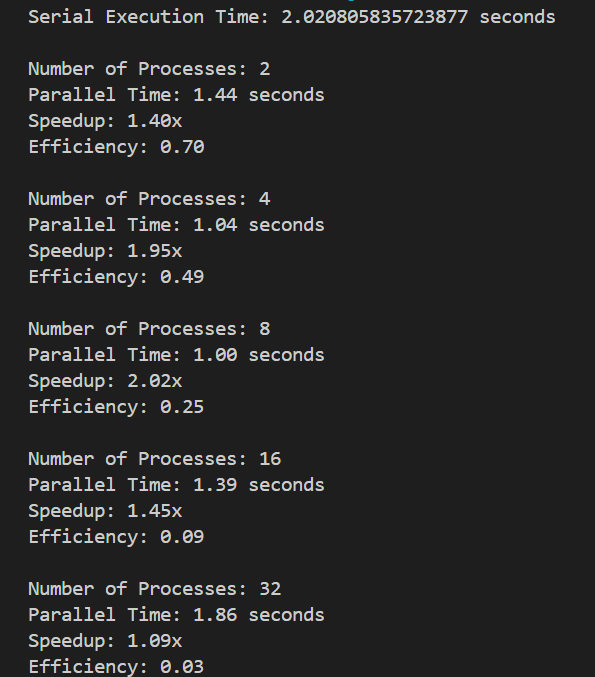


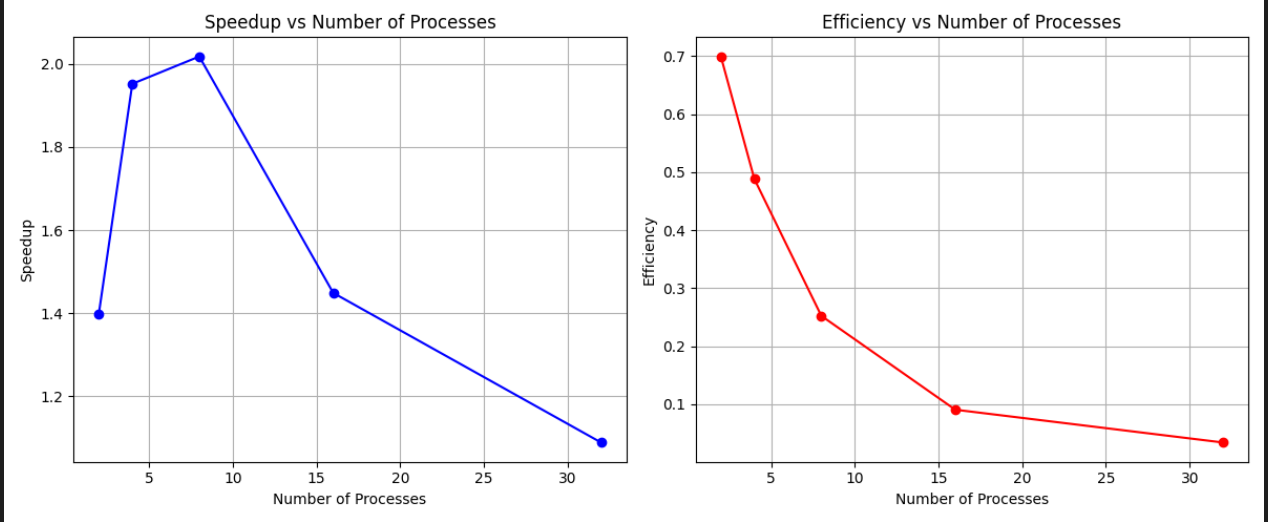


همچنین پارامتر های speedup و efficiency به صورت زیر محاسبه میشود .



در نهایت خروجی برای تعداد پردازه های مختلف به صورت زیر خواهد بود :



همچنین خروجی به صورت نمودار به شکل زیر است : 

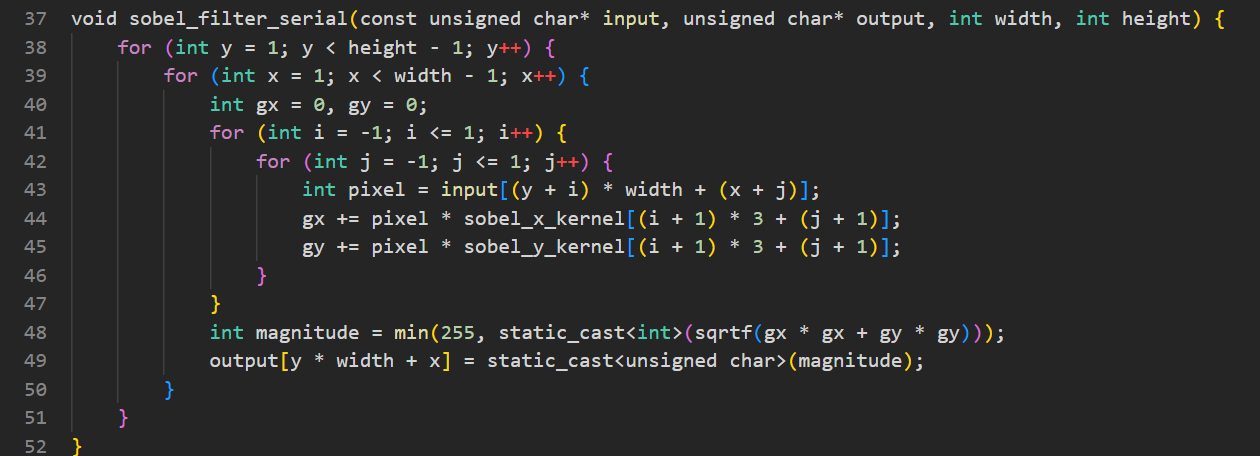
مشاهده میشود که با افزایش تعداد پردازه ها از 2 به 4 و سپس 8 ، speedup افزایش پیدا میکند. ولی با رسیدن به تعداد 16 و 32 ، این مقدار کاهش می یابد و عملکرد بدتر میشود . این نشان میدهد که هزینه ایجاد پردازه و احتمالا سوییچ بین پردازه ها ، هزینه بیشتری ایجاد میکند که از بهره وری موازی سازی بیشتر است .

تصویر خروجی در حالت موازی :



سوال دوم

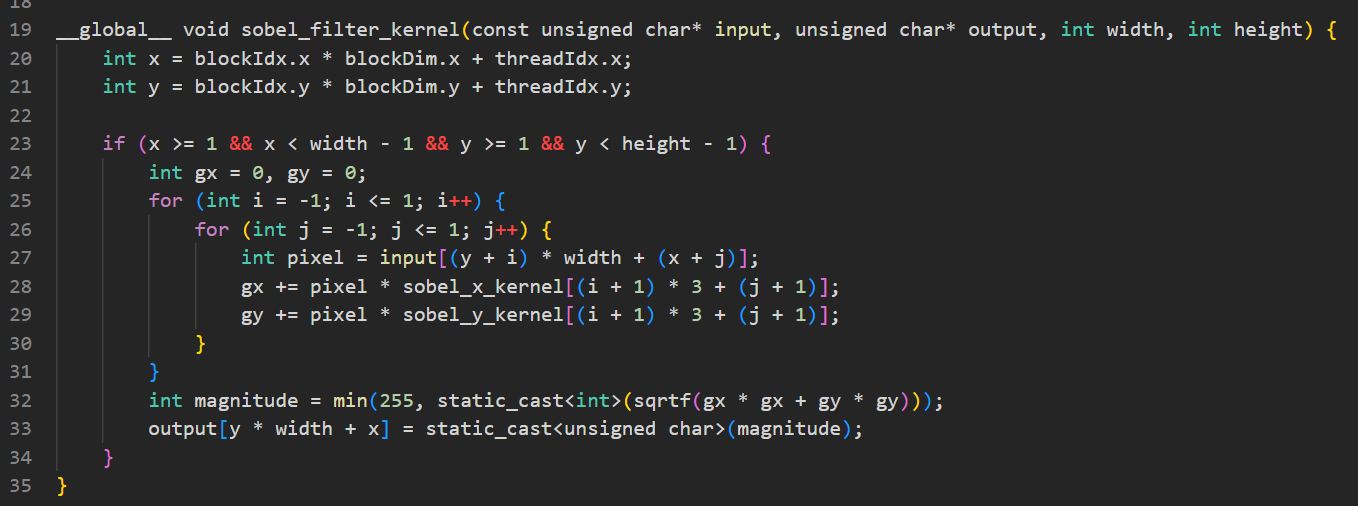
در این قسمت ، موازی سازی و پردازش عکس با cuda انجام میگیرد . برای این کار ، یک قطعه کد سریال بوسیله openCV ایجاد میکنیم که به صورت سریال عملیات را انجام میدهد :



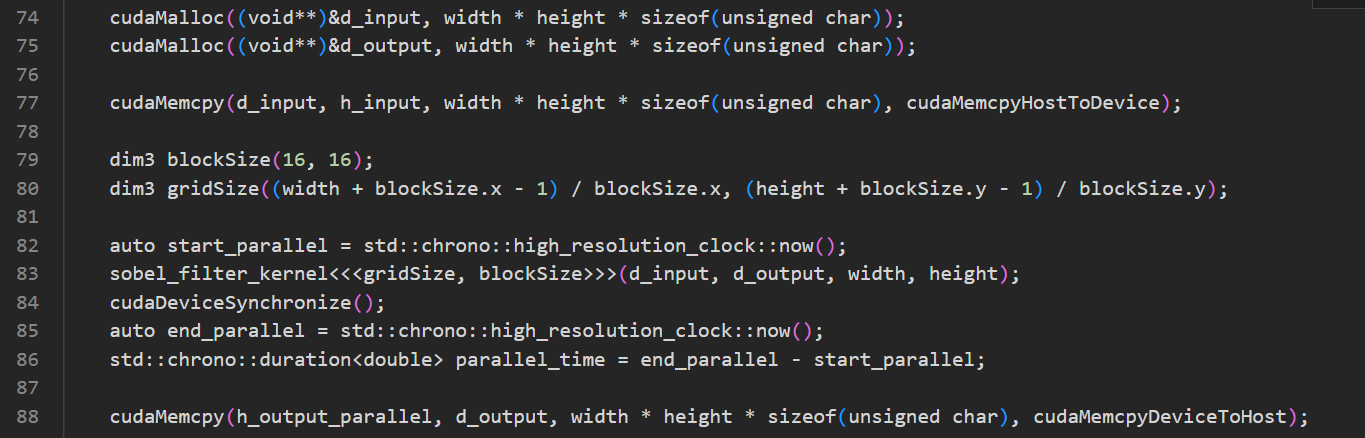
مشابه قطعه کد پایتون قسمت قبل ، این کد سریال هم کرنل را در عکس ضرب می کند و نتیجه را بازنویسی می کند .

در بخش موازی ، یک کرنل داریم که توسط cuda core ها اجرا خواهد شد . این کرنل به صورت زیر است :

بر اساس شماره ایندکس ترد ، روی عکس محاسبات را انجام میدهد و بر روی پیکسلی از عکس که توسط ایندکس ترد محاسبه شده می نویسد :

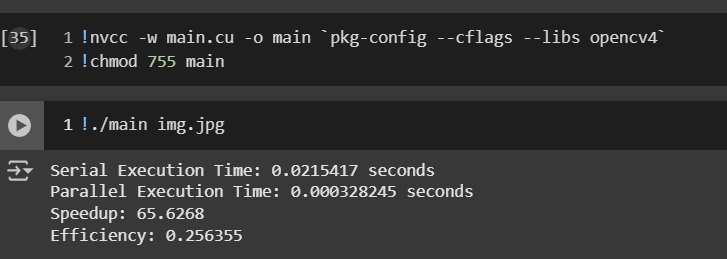


در تابع main هم ابتدا عکس را میخوانیم و سپس حافظه را در سمت device تخصیص می دهیم و عکس را از host به device انتقال میدهیم تا محاسبه شروع شود .



بعد از اجرا و بازگرداندن عکس به هاست ، حافظه ها را free میکنیم .

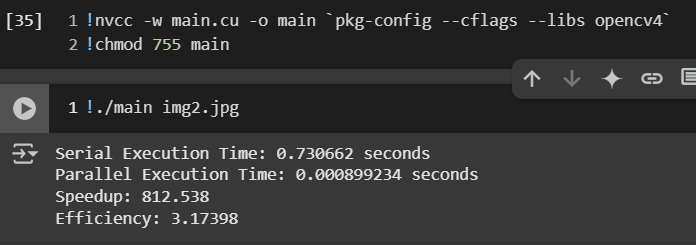
در این قسمت برای اجرا از محیط colab و کارت گرافیک T4 استفاده میکنیم . در نهایت خروجی برای یک عکس کم حجم به صورت زیر است :



تصویر خروجی حالت موازی :



همچنین برای یک عکس بزرگتر با ابعاد 3480\*2160 چون از تعداد core های بیشتری استفاده می شود ، خروجی speedup به شدت بیشتر میشود :



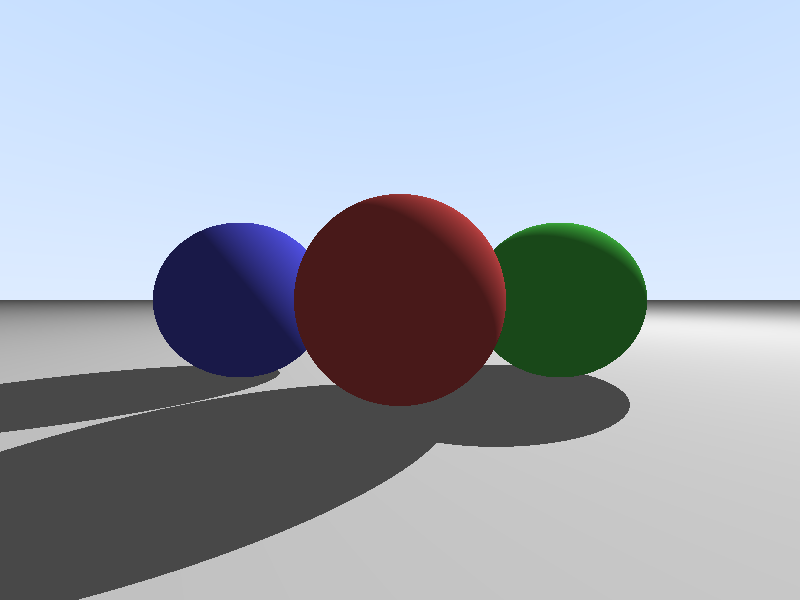
در این حالت دوم از منابع استفاده خیلی بهتری میشود و تعداد بیشتری از هسته ها درگیر میشوند و بیکار نمی مانند.   
تصویر خروجی برای این حالت به صورت زیر است :

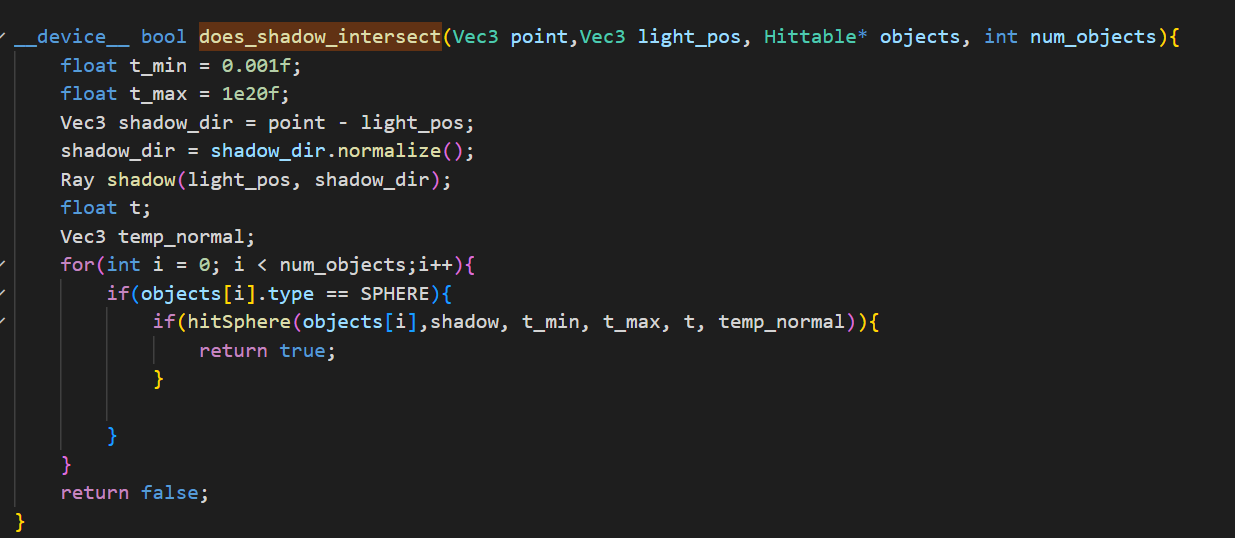


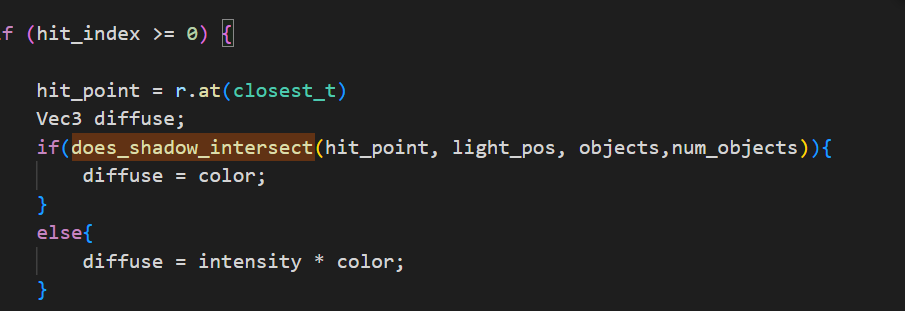
سوال سوم

یکی از ساده ترین روش هایی که برای shadow casting وجود دارد این است که برای هر نقطه از plane که یک ray به آن برخورد می کند به این صورت وجود سایه را حساب کنیم که از مبدا نور یک ray دیگر به سمت آن نقطه می فرستیم اگر در راه تا رسیدن به آن نقطه با جسمی برخورد کرد یعنی که نور دیگر به آن نقطه مورد نظر نمی رسد و باید روی آن نقطه سایه بیفتد. سایه به این صورت اعمال می شود که می توان از درجه diffuse آن کم می شود.

در ساده ترین حالت اگر بخواهیم چیزی که در بالا گفته شد را پیاده سازی کنیم به این شکل ممکن می شود:

همان طور که می بینید که سایه ایجاد می شود. این سایه به قدری غیر طبیعی است چرا که انتظار داریم که یک سری سایه های نرم هم ببینیم مخصوصا زیر کره ها باید خیلی تاریک تر باشد، اما طبق بررسی که من کردم پیاده سازی soft shadow سخت است و ما در این پروژه این کار را دیگر انجام ندادیم.

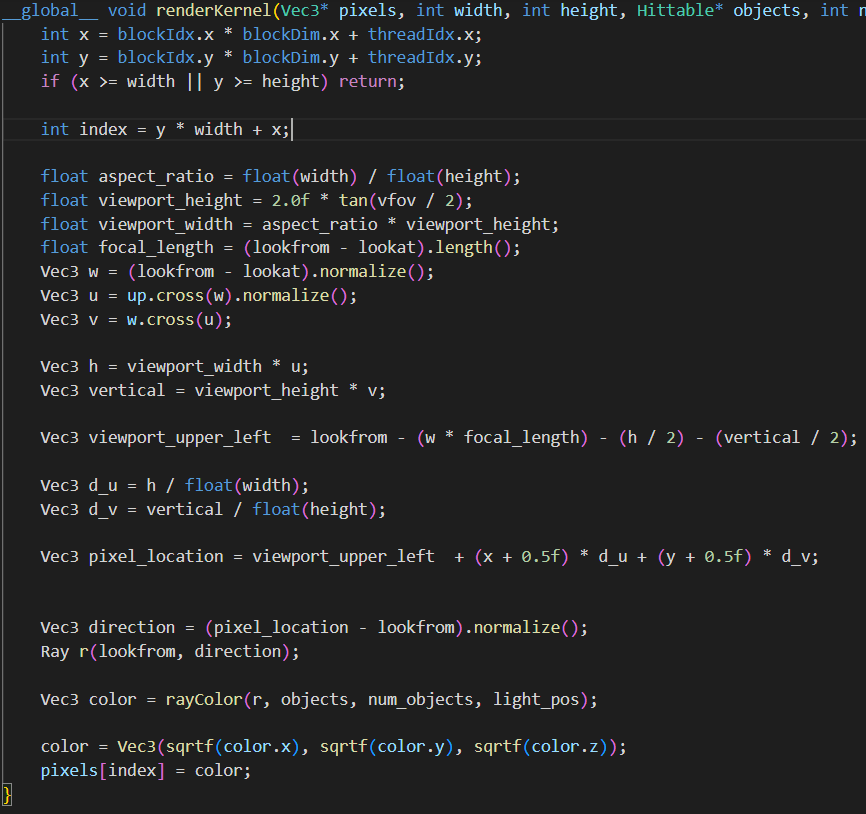


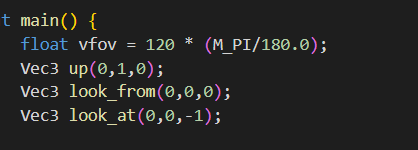
کد چک کردن سایه به این شکل کار میکند که از نقطه مورد نظر که میخواهیم ببینیم روی آن سایه افتاده است یا نه به سمت نور یک ray میسازیم و چک میکنیم که این ray آیا در راه با کره ها برخورد کرده است یا نه.

اگر برخورد کرده باشد ما diffuse که شدت نور است را در آن نقطه کم میکنیم که تاریک تر به نظر برسد.

بخش امتیازی

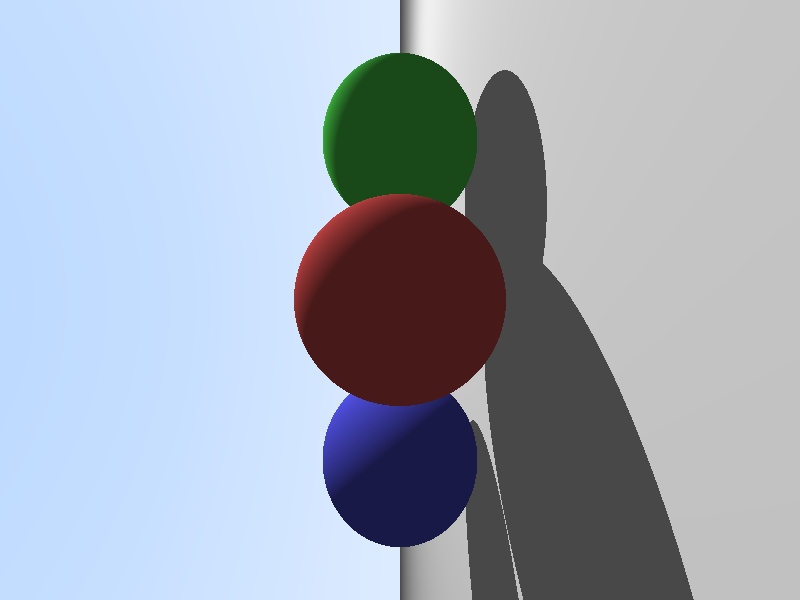
طبق بررسی ها از منابع مختلف که در پایین ذکر شده است با توجه به پارامتر های ورودی میتوان u و v و w را با فرمول های زیر به دست آورد:

کد به شکل زیر با استفاده از کتاب Ray Tracing in One Weekend به شکل زیر کامل شده است:

پارامتر vfov مشخص میکند که زاویه عمودی دوربین تا چه مقدار باشد و محدوده دید دوربین را مشخص می کند. پارامتر های دوربین در ابتدا به شکل زیر مشخص شده اند:

در نهایت با مقادیر مختلف fov رندر عکس به این شکل تغییر می کند:



می توان پارامتر up که محور جهت دوربین را مشخص می کند عوض کرد:

و خب میتوان نقطه مبدا دوربین و همچنین جهت نگاه آن را نیز تغییر داد.



شکل های بالا با جهات مختلفی از دوربین گرفته شده اند.

منابع:

[Ray Tracing in One Weekend](https://raytracing.github.io/books/RayTracingInOneWeekend.html#positionablecamera)

[Vertical and horizontal camera FOV angles](https://b3d.interplanety.org/en/vertical-and-horizontal-camera-fov-angles/)