مقدمه ای بر گرافیک کامپیوتر

احمد منصوری و **peter shirley**

December 21, 2009

چکیده

همزمان با پیدایش کامپیوتر ها، تلاش ها برای بهره بردن از توان آنها برای ابزار های Visualize و دیگر ابزار ها برای استفاده از این قابلیت در زمینه های نظامی، فیلم و انیمیشن، شبیه سازی و بازی سازی شروع شد، این ابزار ها یا به صورت اختصاصی برای استفاده در صنایع خاص طراحی می شوند یا به صورت عمومی تر برای کاربرد های وسیع تری طراحی و توسعه می یابند. طراحی و پیاده سازی موتور های گرافیکی به صورت کلی دارای پایه ها و دانشی یکسان از نحوه کار پردازنده ها و ریاضیات است.

فهرست مطالب

بخش اول

گرافیک

مقدمه

در بخش اول به معرفی و توضیح قسمت های مربوط به تصویر در پروژه می پردازم، این قسمت از موتور مسئولیت دریافت مدل های سه بعدی و اطلاعات مربوطه(**Textures**, **Normals**, **geometry**, ...) و نمایش آن ها در صفحه را برعهده دارد، در این قسمت ما با استفاده از ریاضیات مدل ها را در فضای سه بعدی شبیه سازی می کنیم.

تمامی ابزار های استفاده شده در این قسمت، در طول فصول و متناسب با بخشی که از آن ها استفاده شده معرفی می شوند.

هر فصل در این بخش مستقیما مربوط به یکی از قسمت های موتور در بخش گرافیک است، ابتدای هر فصل فایل های مربوطه به آن فصل ذکر خواهند شد.

فصل ۱

Renderer

پنجره ها و کانتکست ها

openGL

OpenGL یک ***api*** چند زبانه و کراس پلتفرم است که برای به تصویر کشیدن تصاویر دو بعدی و سه بعدی با استفاده از بردار ها اسفاده می شود، معمولا از ***OpenGL*** برای برقراری ارتباط با واحد پردازش گرافیکی ***GPU*** و بهره بردن از سرعت سخت افزار مخصوص برای رندر استفاده می شود.

همچنین *api* های دیگری نیز برای استفاده از قدرت سخت افزاری و پردازنده گرافیکی وجود دارند *Vulkan* نیز مانند *openGL* چند زبانی و چند سکویی است، *DirectX* به صورت انحصاری توسط مایکروسافت توسعه می یابد و در سیستم عامل ویندوز استفاده می شود ، شرکت **Apple** از *api* اختصاصی خود به نام *Metal* به صورت انحصاری پشتیبانی می کند، دلیل انتخاب*openGL* در این پروژه چندسکویی بودن و ساختار ساده تر برای پروژه های آموزشی در زمینه *real-time rendering* می باشد.

نباید *OpenGL* را با یک کتابخانه*(library)* اشتباه گرفت، **OpenGL** به صورت یک interface و یک قراداد انتزاعی در ورژن های مختلف ارائه می شود که فروشندگان و سازندگان*(vendor)* مختلف باید پیاده سازی ای منطبق با این قرارداد را انجام دهند. پس از نصب درایور مربوط به پردازنده گرافیکی، برنامه نویس قابلیت دسترسی به تابع های مختلف که توسط *vendor* پیاده سازی شده را خواهد داشت. برای استفاده از *OpenGl* نیاز به ابزار های دیگری نیز داریم، ابتدا نیاز داریم که یک *window* و یک *context* تعریف کنیم، برای این کار از **GLFW** استفاده می کنیم.

GLFW

یک کتابخانه برای ساختن *window* و *context* ها برای *openGL, openGL ES, Vulkan* است، این کتابخانه به زبان **C** نوشته شده و *binding* های مختلف آن به زبان های مختلف موجود است، این کتابخانه همچنین توانایی کنترل کردن ورودی های مختلف مثل *keyboard, mouse, joystick* را داراست، ما برای استفاده از *openGL* نیاز به این کتابخانه یا مشابه آن داریم زیرا openGL هیچگونه قابلیت پیشفرضی برای مدیریت window یا context ها یا مدیریت input ندارد. همچنین GLFW یک کتابخانه چندسکویی است و می توانیم آن را در سیستم عامل های مختلف استفاده کنیم، پروژه من نیز چندسکویی است، پس می توانیم از این کتابخانه سبک و چندسکویی استفاده کنیم. windows، پنجره ای است که GLFW به وسیله امکانات فراهم شده در سطح سیستم عامل برای ما فراهم می کند، همچنین یک context را می توانیم به عنوان یک شئ در نظر بگیریم که تمامی اطلاعات openGL را به همراه دارد، اطلاعاتی مانند state و framebuffersها . برای کنترل کردن ورودی ها، glfw از دو روش استفاده می کند، برای ورودی*mouse* از callback function ها استفاده می کند، اما برای ورودی keyboard می توانیم از تابع های کتابخانه استفاده کنیم و به صورت مستقیم ورودی را دریافت کنیم.

حالا که به window و context دسترسی داریم، باید دسترسی به تابع های opengl فراهم کنیم، برای این کار از **GLAD** استفاده می کنیم.

شکل ۱.۱: *glfw logo*

GLAD

کتابخانه ای برای *load* کردن pointer ها به توابع opengl در هنگام runtime. این کتابخانه یکی از کتابخانه هایOpenGL Loading Library است، برای کار با opengl ما حتما باید یکی از این کتابخانه هارا مورداستفاده قرار دهیم تا بتوانیم به توابع opengl دسترسی داشته باشیم، این کتابخانه ها هم ویژگی های Core که توسط opengl مشخص شده را load می کنند و هم ویژگی های extension که توسط Vendor ها به پیاده سازی آن ها از opengl اضافه شده، علاوه بر این دیگر نیازی به اضافه کردن فایل های مربوط به opengl نیست و این فایل ها به صورت خودکار همه موارد را تنظیم می کنند. Glad یک generator است که براساس پارامتر هایی که کاربر انتخاب می کند یک فایل حاوی تمامی تعریف های مربوط به constant و تابع ها و ... به ما ارائه می کند، بعد از دانلود این فایل و اضافه کردن به آن به پروژه از طریق کد زیر می توانیم تمامی opengl function poiter هارا در runtime بارگزاری کنیم.

برنامه ۱.۱: □□□□□ □□□□□□□□ □□□□□□□ □□□□□ □□□□□□□□

```
1  int main()
2  {
3    return 0;
4  }
```

بخش دوم

فیزیک

فصل ۲

Physics

۱.۲ Particles

۲.۲ Rigid Bodies