Počítačová grafika

Petr Horáček

Obsah

1 Úvod

Hry a grafické anymace jsou interaktivní filmové sekvence s předeme programovanými reakcemi na události. Každý snímek tohoto filmu je rendrován v reálném čase na základě uživatelské reakce, tak aby tvořil dojem realistického dojmu.

Architektura hry je běžně tvořena třemi hlavními částmi:

- **Inicializace hry**
- **Herní smička**
- **Ukončení hry**

Herní smyčka se následně skládá z následujících:

- **Převzetí uživatelského vstupu**
- **Kontrola interních časovačů**
- **Řízení autonomních herních botů**
- **Úprava herního stavu na základě uživatelského vstupu**
- **Vykreslení nového herního snímku**

2 Grafická karta

Grafická karta, často označovaná jako **GPU** (Graphics Processing Unit), je specializovaná elektronická součástka navržená pro rychlé a efektivní zpracování a vykreslování obrazu.

Zatímco CPU (Central Processing Unit) - hlavní procesor počítače - je skvělý v provádění široké škály úloh sériově (jedna po druhé), není ideální pro úlohy, které vyžadují obrovské množství paralelních výpočtů, jako je tomu u grafiky. Vykreslení jednoho snímku ve 3D hře může vyžadovat miliardy výpočtů pro každý pixel a každý vrchol modelu.

Grafická karta je proto navržena tak, aby tyto specifické grafické výpočty prováděla masivně paralelně. Má tisíce menších, specializovaných jader (na rozdíl od několika málo, ale velmi výkonných jader CPU), která dokážou zpracovávat mnoho grafických úloh současně. To je klíčové pro plynulé zobrazení složitých scén.

2.1 Hlavní komponenty Grafické Karty

- **GPU (Graphics Processing Unit)** - Samotný grafický procesor, který je srdcem karty. Provádí veškeré výpočty související s 2D a 3D grafikou, zpracováním videa, a v moderní době i obecnými výpočty (tzv. GPGPU - General-Purpose computing on GPUs).
- **Video paměť (VRAM - Video Random Access Memory)** - Velmi rychlá paměť, která je vyhrazena pouze pro GPU. Slouží k ukládání dat potřebných pro vykreslování - textur, modelů, shaderů, z-bufferů, framebufferů a dalších informací, ke kterým má GPU okamžitý přístup. Rychlost a kapacita VRAM jsou klíčové pro výkon, zejména ve vysokém rozlišení.
- **Video BIOS (VBIOS)** - Malý firmware, který inicializuje grafickou kartu při startu počítače a obsahuje základní informace o kartě.
- **DAC (Digital-to-Analog Converter) / Video Output Interfaces** - Moderní karty používají digitální výstupy, jako jsou HDMI, DisplayPort nebo DVI-D, které převádějí digitální signál z GPU na formát, kterému rozumí monitor. Starší karty mohly mít i VGA (analogový) výstup.
- **Chladicí systém** - Vzhledem k obrovskému množství tepla generovaného GPU a VRAM při intenzivních operacích, jsou grafické karty vybaveny ro-

bustními chladicími systémy - pasivními chladiči (heatsinky) a aktivními ventilátory.

- **Sběrnice (PCI Express)** - Fyzické rozhraní, které umožňuje grafické kartě komunikovat s ostatními komponentami počítače, zejména s CPU a systémovou pamětí. Nejčastěji se používá standard PCI Express (PCIe).

3 Grafické API

API - Aplication Programming Interface je obecně sada postupů, které přesně popisují přístup a ovládání určité softwarové komponenty. **Grafické API** je programové rozhraní mezi programovým kódem a konkrétní grafickou kartou v počítači, které popisuje jak obecně pracovat s grafickými kartami.

Na trhu existuje velké množství grafických karet různých výrobců a každá grafická karta má určitý způsob ovládání. To znamená, že pro použití dané grafické karty je třeba vytvořit specifický postup řízení v programu, který s touto kartou pracuje. Jakmile by se ale daný program použil na počítači s jinou grafickou kartou, došlo by tomu, že by program nefungoval správně, protože jiná karta vyžaduje jiný specifický způsob ovládání. Daný program tak není přenositelný na jiný stém a vyžaduje speciální hardwarovou konfiguraci. Aby grafické programy byly přenositelné mezi systémy s různými grafickými kartami vzniklo tzv. grafické API, které přesně popisuje jednotný způsob ovládání grafických karet.

Grafické API vytváří neviditelnou mezi-vrstvu mezi grafickou kartou a programem, který s grafickou kartou potřebuje pracovat. V této neviditelné mezi-vrstvě se pak skryje způsob jak se s konkrétní grafickou kartou pracuje a vytvoří se přesně pojmenované příkazy, které mají za úkol provést přesně danou činnost. To ve výsledku znamená, že pokud grafické API zpřístupní seznam operací které budou fungovat na libovolné grafické kartě, je možné z nich sestavit komplexní grafický program a není třeba se starat o to jakým způsobem to grafické API dělá.

3.1 Úkoly grafického API

- **Správa stavu** - nastavení a míchání barev
- **Správa paměti GPU** - Nahrávání vertex dat, textur, shaderů do paměti grafické karty
- **Vykreslování geometrie**
- **Programování shaderů**
- **Správu textur**
- **Nastavení viewporu a projekce** - Definování, jak se 3D scéna mapuje na 2D okno
- **Synchronizaci** - Zajištění správného pořadí operací mezi CPU a GPU

3.2 Příklady grafických API

Existuje více v současné době používaných grafických API. Je to důsledek historického vývoje jak grafického hardwaru tak požadavků uživatelů a konkurence softwarových společností jako je Apple a Microsoft a open source komunity. Každé z těchto grafických API má své pro i proti a jsou optimální pro určité případy použití. Mezi nejčastější grafická API patří:

- **OpenGL - Open Graphics Library** - Starší, ale stále široce používané API od Khronos Group. Je možné jej používat na všech současných platformách - Ms Windows, Apple a GNU Linux.
- **Direct3D** - Grafické API od Microsoftu, exkluzivní pro Windows a Xbox. Je součástí DirectX.
- **Vulkan** - Moderní, nízkoúrovňové, explicitní API od Khronos Group. Dává programátorovi mnohem větší kontrolu nad hardwarem a je navrženo pro moderní vícevláknové procesory a více GPU. Je multiplatformní (Windows, Linux, Android, částečně macOS přes MoltenVK).
- **Metal** - Grafické API od Applu, exkluzivní pro Apple platformy (macOS, iOS, iPadOS, tvOS). Velmi nízkoúrovňové, optimalizované pro Apple hardware.

3.3 Grafické API a herní engine

Grafické API přináší stále vysokou kontrolu nad použitým GPU. To znamená, že většina grafických operací jako je vykreslení základních geometrických objektů nebo texturování není přímočaré a skládá se z určité posloupnosti dílčích příkazů jejichž výsledkem je požadovaná akce. To činí programování grafických aplikací čistě pomocí grafického API zdlouhavé a technicky stále relativně náročné.

Aby vývoj grafických aplikací jako jsou photoshopy nebo hry bylo rychlejší a méně náročné na použití, jsou nad grafickým API postaveny grafické knihovny vyšší úrovně jako jsou například herní enginy. Grafické knihovny a enginy (jako Unity nebo Unreal Engine) výrazně zjednodušují práci s grafikou, protože fungují jako chytrá vrstva abstrakce nad samotnými grafickými API (jako OpenGL nebo Vulkan). Dělají to dvěma hlavními způsoby:

- **Zjednodušují složité operace** - operace, které by bylo nutné provést pomocí mnoha dílčích příkazů grafického API je skryto pod jedním komplexním příkazem grafické knihovny.

- **Abstrahují rozdíly mezi API a platformami** - různá grafická API stejnou věc umožňují realizovat jiným způsobem nebo jinými příkazy. Grafické knihovny umožňují skrýt tyto detaily a stejnou operaci můžou realizovat pomocí různých grafických api a uživateli umožňuje přepínat mezi tím jaké grafické API použít.

4 Okenní systém

Grafické API (jako OpenGL, Vulkan, Direct3D nebo Metal) je sada instrukcí, které umožňují komunikovat s grafickou kartou (GPU). Popisuje, jak provádět grafické výpočty, operace a jak pracovat s grafickou pamětí. Samotné API neřeší, jak se výsledek těchto výpočtů zobrazí na monitoru nebo jak se vytvoří okno, do kterého se má vykreslovat. To je práce pro operační systém a jeho **okenní systém**.

Každý operační systém (Windows, Linux, macOS) má svůj vlastní okenní systém. Ten je zodpovědný za všechno, co je zobrazeno na obrazovce a s čím daný program interaguje:

- **Vytváření a správa oken** - Vytváří okna, ikony, tlačítka a veškeré uživatelské rozhraní.
- **Zpracování uživatelského vstupu** - Reaguje na kliknutí myši, stisky kláves, dotyky.
- **Správa grafických bufferů** - Přiděluje paměťové oblasti (buffery), kam mohou aplikace vykreslovat.
- **Komunikace s ovladači displeje** - Posílá hotové pixely z těchto bufferů na monitor.

4.1 Běžné okenní systémy

- **Win32 API (pro Windows)**
- **X11 (X Window System) nebo Wayland (pro Linux)**
- **Cocoa (pro macOS)**

4.2 Napojení grafického API na nativní okno

Než je možné cokoli zobrazit pomocí grafického API, je třeba nejprve vytvořit spojení mezi konkrétním grafickým API a oknem na daném systému. Tento proces se nazývá napojení nebo integrace (někdy se mluví o Window System Integration - WSI u Vulkanu).

Základní kroky propojení jsou pro všechna grafická API a okenní systémy podobné, liší se jen konkrétní API volání:

- **Vytvoření Nativního Okna** - Nejprve je třeba pomocí funkcí okenního systému vytvořit standardní systémové okno, se kterým může uživatel interagovat.
- **Získání "Vykreslovací Plochy"** - Z okna je třeba získat odkaz na jeho vykreslovací plochu (např. Device Context (HDC) na Windows, Drawable ID na X11 nebo wl_surface na Waylandu). Tato plocha je pro grafické API cílem vykreslování.
- **Nastavení Pixel Formátu** - Systému je třeba sdělit, jaké vlastnosti má mít tato vykreslovací plocha (např. barevná hloubka, podpora dvojitého bufferování, hloubkový a stencil buffer). Tím se připraví buffer, do kterého bude GPU kreslit.
- **Vytvoření Grafického Renderovacího Kontextu** - Na základě vybraného pixel formátu a vykreslovací plochy se ovladač grafické karty požádá o vytvoření grafického renderovacího kontextu. Toto je stavový stroj na GPU, který uchovává všechna aktuální nastavení pro vykreslování.
- **Aktivace Kontextu** - Sdělení operačnímu systému, že daný konkrétní grafický kontext je aktivní pro aktuální vlákno programu. Teprve pak systém propojí volání funkcí grafického API konkrétnímu oknu.
- **Načtení Funkcí API (Loadery)** - Jelikož většina moderních funkcí grafických API není přímo součástí systémové knihovny, je třeba je dynamicky načíst za běhu. Teprve poté je možné volat funkce grafického API.
- **Výměna Bufferů (SwapBuffers)** - Po dokončení vykreslování jednoho snímku do skrytého (back) bufferu, je třeba dát okennímu systému příkaz k prohození tohoto back bufferu s aktuálně zobrazovaným (front) bufferem. Tím se vykreslený snímek zobrazí na monitoru a zabrání se blikání (**Flickering**).

5 OpenGL

6 Vykreslovací řetězec

Vykreslovací řetězec (Graphics pipeline)