GYMNÁZIUM JÍROVCOVA 8

MATURITNÍ PRÁCE

Programování grafické aplikace v C++

René Čakan

vedoucí práce: Dr. rer. nat. Michal Kočer

V Českých Budějovicích 14. února 2025 **2025/2026**

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně s vyznačením všech použitých pramenů.

V Českých Budějovicích dne ..... podpis .....

René Čakan

Abstrakt

Klíčová slova

Poděkování

Obsah

Úvod

Část I

Teorie k vývoji hry v OpenGL

1 Programovací jazyky

1.1 Programovací jazyk C

C je středněúrovňový programovací jazyk, tedy jazyk, který je podobou blízko strojovému kódu, ale má už prvky vyššího programovacího jazyka jako jsou funkce, datové struktury nebo to že je strukturovaný. Je kompilovaný a statický, což znamená, že se program musí nejdříve přeložit do strojového kódu a až pak se může spustit. Datové typy jsou známy v čase kompilace, proto všechny proměnné musí být v kódu deklarovány, jelikož vkládání vstupních dat do programu probíhá až při běhu programu. Programuje se v něm strukturovaně a procedurálně, tedy kód se píše pomocí řídících struktur (if, while, for atd.) a pomocí funkcí, které umožňují používat části kódu vícekrát. C nemá automatický správce paměti, takže je potřeba uvolňovat paměť manuálně. C má střídmou standardní knihovnu, která obsahuje základní matematické operace a funkce pro práci s pamětí a soubory, takže jakékoliv složitější datové struktury či funkce si člověk musí naprogramovat sám. Tato strohost a blízkost ke strojovému kódu z C dělá jeden z nejrychlejších programovacích jazyků. [23, 21, 13, 30, 32]

C bylo vytvořeno Dennisem Ritchiem na počátku 70. let 20. století v AT&T Bell Labs Jeho předchůdci byly jazyky ALGOL, CPL, BCPL a B. Jeho prvotním účelem bylo přepsat operační systém UNIX do použitelnějšího jazyka než Assembly a B. Už koncem 70. let bylo C populární, ale nebylo standardizované a vznikalo mnoho různých variant. Na začátku 80. let tedy American National Standards Institute (ANSI) zahájil práci na formální standardizované verzi. Tu dokončili v roce 1989 a je známa pod jménem C89. V průběhu let vycházely další verze, které jazyk zlepšovaly a modernizovali. Nejdůležitější verze byly C99, C11 a C17. Norma C23 byla nedávno schválena a teď se implementuje do kompilátorů. V současnosti mezi nejpoužívanější kompilátory patří GCC, Clang a MSVC. Jelikož bylo C velice populární, ovlivnilo řadu jiných programovacích jazyků, jako C, C#, Java, Rust, Go atd. [4, 16, 23, 21]

C je univerzální programovací jazyk, má tedy širokou škálu využití. Jeho první využití

bylo k napsání UNIXu, který později ovlivnil operační systémy jako Linux, macOS, iOS a Android. Používá se v programování softwaru s omezenou pamětí a výkonem, jako je firmware aut či v zařízeních chytrých domácností. Dále se využívá pro tvoření kompilátorů a interpreterů jako je GCC nebo interpreter Pythonu. Také jsou v něm napsané systémové databáze MySQL a Oracle Database. Kvůli jeho rychlosti jsou v něm napsané knihovny pro jiné programovací jazyky jako je NumPy, OpenGL či GLFW. [4, 23, 44]

Jednoduchý program, který načte ze vstupu počet čísel, která chce uživatel setřídit. Následně daná čísla načte a vytiskne je seřazená:

```c
#include <stdio.h>
#include <stdlib.h>

int compare(const void *a, const void *b)
{
    return (*(int *)a - *(int *)b);
}

int main(void)
{
    int sizeOfArray;
    scanf("%d", &sizeOfArray);
    int *array = malloc(sizeOfArray *sizeof(int));

    for(int i = 0; i < sizeOfArray; ++i)
        scanf("%d", &array[i]);

    qsort(array, sizeOfArray, sizeof(int), compare);

    for(int i = 0; i < sizeOfArray; ++i)
        printf("%d\n", array[i]);

    free(array);
    return 0;
}
```

Zdrojový kód 1.1: sort_n_numbers.c

1.2 Programovací jazyk C++

Programovací jazyk C++ je v mnoha ohledech podobný jazyku C. Je stejně jako C středně-úrovňový, kompilovaný, statický, má datové typy známé v době kompilace a nemá au-

tomatický správce paměti. V C++ se také programuje strukturovaně a procedurálně, ale na rozdíl od C, také umožňuje programovat objektově. Objektové programování umožňuje používat objekty, které jsou instance tříd. Tyto třídy umožňují dědičnost, polymorfismus a zapouzdření, což dělá kód přehlednější a usnadňuje budoucí rozšiřování a debuggování. Dalším rozdílem je standardní knihovna, kterou má C++ rozsáhlejší. Obsahuje nové kontejnery jako vector, map, a priority_queue, které jsou tvořeny pokročilejšími datovými strukturami jako binární vyhledávací strom nebo heap. Dále obsahuje nové algoritmy, například sort, find nebo count. Kvůli velké podobnosti C a C++ se často může C kód používat v C++, ale není tomu tak vždy. Například tento kód:

```
1 int class(int new, int bool);
```

Zdrojový kód 1.2: incompatibility_example.c

V C tento kód vytvoří funkci class, která vrací int a má dva parametry new a bool. V C++ jsou ale class, new a bool klíčová slova, která nelze použít v názvu proměnných a funkcí. Pokud chce programátor napsat C kód, který se bude jednoduše v C++ programech, doporučuje se programovat v C tak, aby daný C kód byl podmnožinou C++. [43, 42]

C++ bylo vytvořeno Bjarnem Stroustrupem v roce 1979 v AT&T Bell Labs. Před vytvořením C++ pracoval Stroustrup s programovacím jazykem Simula 67, který byl objektově orientovaný a sloužil primárně k vytváření simulací. Stroustupovi přišlo objektově orientované programování velmi užitečné, ale Simula 67 byl příliš pomalý pro větší projekty. Rozhodl se vytvořit nadmnožinu jazyka C, která by umožňovala objektově orientované programování a zároveň si zachoval rychlost C, s názvem C with Classes. V roce 1982 byl Stroustrup se stavem C with Classes zklamán. Nepřišlo mu, že oproti C přináší významné zlepšení a rozhodl se jazyk dále vylepšovat nad rámec objektově orientovaného programováni. V roce 1983 byl jazyk přejmenován z C with Classes na C++. Dále bylo C++ v roce 1985 oficiálně vydáno a začalo se používat komerčně. V roce 1998 byla vydána první standardizovaná verze s jménem C++98. Další významné verze, které jazyk modernizovaly a přidávaly mu nové funkce, byly C++03, C++11, C++14, C++17, C++20 a nejnovější verze C++23. C++ se kompiluje pomocí stejných kompilátorů jako C, tedy GCC, Clang a MSVC. [42, 41, 3, 47]

C++ je stejně jako C univerzální programovací jazyk, a využívá se v široké škále odvětví. První využití je ve videoherním průmyslu. V C++ jsou napsané populární hrací enginy jako Unity nebo Unreal Engine. Dále se v něm vytváří aplikace jako Photoshop nebo Blen-

der. Využívá se v částech operačních systémů jako Apple macOS nebo Microsoft Windows OS. Dále se využívá při vytváření internetových prohlížečů, například Firefox nebo Google Chrome. C++ se využívá i ve vědě, například v CERNu nebo v NASA. [43, 14, 40]

Program se stejnou funkcí jako z kapitoli o C, ale napsán v C++

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

int main()
{
    int sizeOfArray;
    std::cin >> sizeOfArray;
    std::vector<int> arr(sizeOfArray);

    for(int i = 0; i < sizeOfArray; ++i)
        std::cin >> arr[i];

    std::sort(arr.begin(), arr.end());

    for(int i : arr)
        std::cout << i << '\n';

    return 0;
}
```

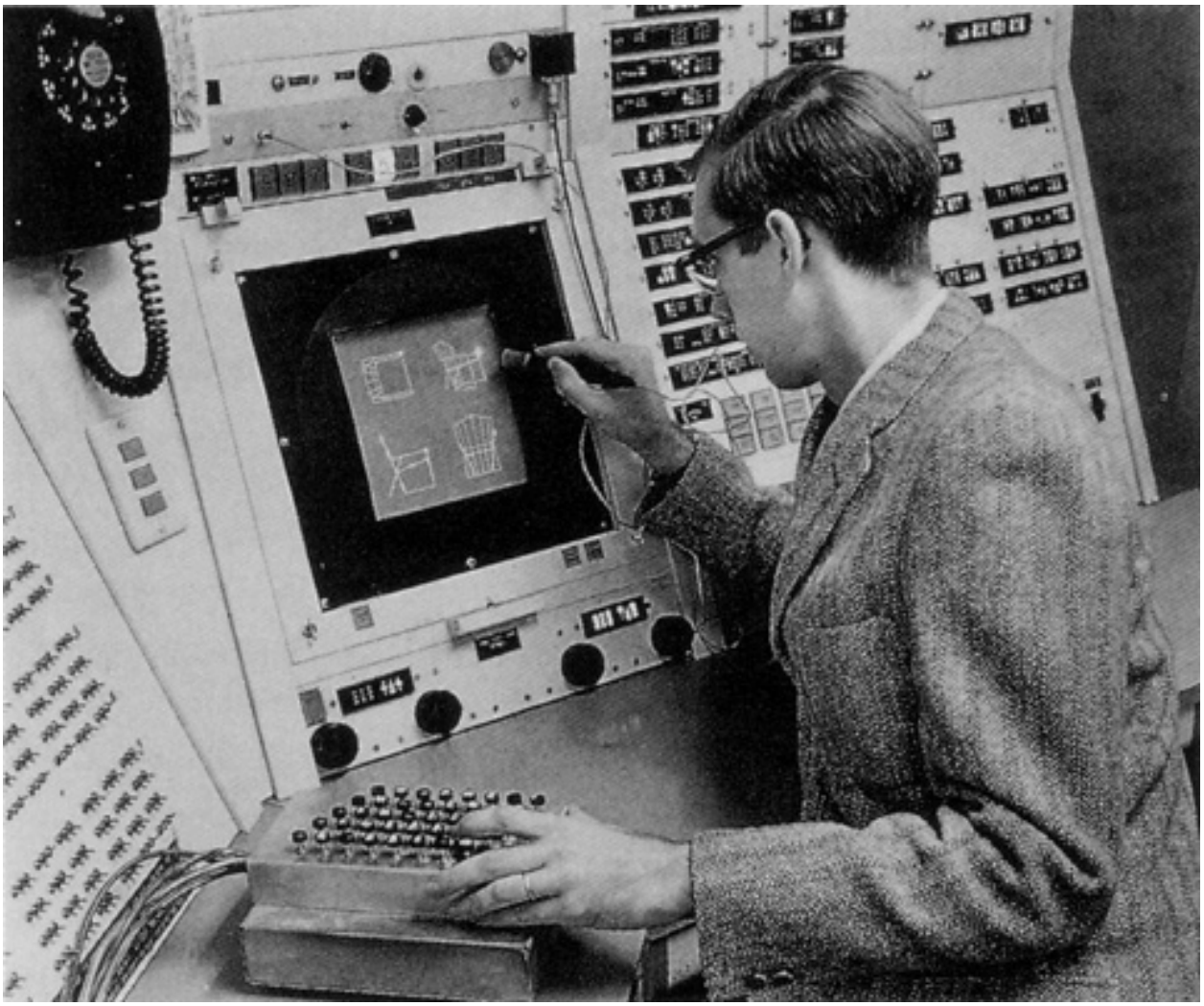
Zdrojový kód 1.3: sort_n_numbers.cpp

2 Počitačová grafika

2.1 Historie počítačové grafiky

Není úplně jasné, který počítač jako první využíval počítačovou grafiku, ale začnu počítačem Small-Scale Experimental Machine (SSEM). Tento počítač byl vytvořen v roce 1948 na Manchesterské univerzitě a jeho tvůrci byli Frederic C. Williams, Tom Kilburn a Geoff Tootill. Tento počítač byl první počítač s uloženým programem, tedy počítač, který měl svůj program uložen ve stejné paměti jako data, se kterými počítač pracoval. Data uchovával na katodové trubici (CRT), které se později začalo říkat Williamsova trubice. Tato trubice si dokázala pamatovat až 2048 bitů, které byly uchovávány jako elektrické náboje. Součástí trubice byl i displej, který na svém fosforovém povrchu promítal oblasti s nábojem. Takto vznikl první počítač s digitálním displejem. [35, 31]

Dalším zajímavým počítačem byl Whirlwind. Americký Office of Naval Research a U.S. Air Force chtěli vytvořit počítač, ve kterém by dokázal běžet letecký simulátor. Tak tedy v roce 1947 začal Jay Forrester pracovat v laboratořích MIT na projektu Whirlwind. Při práci vyvinul Forrester paměť s náhodným přístupem (RAM), tvořenou magnetickými jádrami, skrz které proudil koincidenční proud. Počítač byl dokončen v roce 1951. Využíval CRT na zobrazování výsledků podobně jako SSEM a dokázal na svém displeji řešit rovnice, později i simulovat karetní hru blackjack. Následně se projekt Whirlwind stal součástí projektu Semi-Automatic Ground Environment (SAGE). Ten měl za úkol vytvořit počítačový systém, který by pomocí radarů dokázal odhalovat letadla a řídit obranné síly proti případným letadlům. Projekt SAGE byl jeden z prvních systémů, které využívaly interaktivní ovládání pomocí klávesnice či speciálního světelného pera. Pokud bylo perem namířeno na ikonu letadla, zachytilo světlo z displeje a počítač zobrazil informace o daném letadle jako rychlost a směr jeho letu. Další počítač, který navazoval na projekt Whirlwind a SAGE, byl počítač TX-2, který byl vytvořen Wesem Clarkem na MIT. Ten byl na rozdíl od Whirlwindu tranzisto-



Obrázek 2.1: Skatchpad Ivana Sutherlanda [20]

rový. Zajímavým projektem, který tento počítač umožnil, byl Sketchpad vytvořen Ivanem Sutherlandem. Tento program byl první, který umožnil interaktivně kreslit na obrazovku. Obrazovka byla velká 7x7 palců s rastrem 1024x1024 bodů a psalo se na ní perem, které zachytávalo světlo z obrazovky. Poloha pera se poté poslala do počítače a na daném místě se vybarvil bod. Druhou rukou ovládal uživatel box s přibližně 40 tlačítky, které měly funkce jako mazaní, zoomování či ukládání. Tím vznikl první počítač s interaktivní počítačovou grafikou. [35, 11, 25, 1, 20]

S postupným vývojem počítačů se začaly vytvářet i grafické algoritmy. Jedním z nich je Bresenhamův algoritmus. Vytvořil ho Jack Elton Bresenham v roce 1962 v International Business Machines Corporation (IBM). Tento algoritmus se používá ke kreslení úsečky mezi dvěma body. Jelikož počítačová obrazovka je rozdělena na pixely, pokud daná úsečka není vodorovná ani svislá, nelze ji vykreslit přesně. Algoritmus determinuje, jaký z dvojice pixelů

se více blíží funkci požadované úsečky a tento pixel vybarví. Dalším zajímavým algoritmem je ray casting. První obrázek byl vytvořen v roce 1968. Tento algoritmus funguje tak, že nejdříve uživatel definuje objekty, které chce mít ve své scéně jako matematické rovnice. Pro každý objekt se definuje model osvětlení, což udává jakou má barvu a jak se od něj světlo odráží. Poté se z pohledu kamery vyšle na každý pixel na obrazovce paprsek. Ten putuje, dokud se nezastaví o nějaký námi vytvořený objekt. Následně se z každého místa zastavení vyšle stínový paprsek do zdroje světla. Pokud cestou tento paprsek potká nějaký jiný objekt, daný objekt blokuje světlo, takže tento bod bude ve stínu. [35, 26, 29]

V průběhu 70. let se vyvíjely a vylepšovaly renderovací algoritmy. Jedním z problémů, který bylo potřeba vyřešit, byl hidden surface determination problem. Renderování částí objektů, které nejsou z pohledu kamery vidět zbytečně zatěžuje CPU. Tento problém řeší více rozdílných algoritmů, jako Z-buffer algorithm, Painterś algorithm nebo Binary Space Partitioning. Nejpopulárnější z nich je Z-buffer algoritmus vytvořen Wolfgangem Straßerem v roce 1974. Depth buffer uchovává nejmenší dosud naleznutou hloubku a pokud se přidá nový fragment na daný pixel, algoritmus porovná obě hloubky a pokud je blíže kameře, depth buffer se přepíše na novou hloubku. Další významný pokrok byl v oblasti shadingu. Doposud se využíval k barvení flat shading, takže každý polygon měl svojí vlastní barvu. V roce 1971 vytvořil Henri Gouraud gouraud shading. Tato metoda ukládá barvu polygonů jen do vrcholu daného polygonu, a následně se vypočítává barva každého pixelu váženým poměrem podle vzdálenosti od daného vrcholu. Dalším z důležitých techink je Phong shading, kterou vytvořil Bui Tuong Phong v roce 1974. Por každý pixel se počítá, jak na daný pixel dopadá světlo a jak se od něj odráží. To se následně aplikuje pro výhcozí barvu polygonu. Tato technika je velice náročná a mohla se plně začít využívat až s rovojem GPU. Pokrok se udělal i v realističnosti povrchů. Pokud by povrch měl mýt nějaké výstupky čí hrbolky, museli by býti dodány v popisu daného polygonu. Tento problém řeší technika zvaná Bump Mapping, vytvořena Jamesem Blinnem v roce 1978. Vytvoří se texturová mapa, která určuje kde je vrcholek a kde prohlubeň. Poté se ve fázi počítaní světla místa s vrcholekm udělají světlejší a místa s prohlubní tmavější, což dodává efekt hrbolatého povrchu na uplně rovném polygonu. [46, 28, 27, 8, 15]

V 80. letech za začaly vytvářet první počítače, které byly dostupné pro veřejnost a začala se využívat GPU . Počítačová grafika se začala využívat ve filmovém průmyslu. V roce 1982 vyšel film Tron, který první masivně využíval CGI. Posun se udělal i v modelování vytvořením matematického modelu Non-Uniform Rational B-Spline (NURBS). Tento model

je schope z bodů a uzlu vytvořit 3D křivku nebo plochu. Další důležitý model byl Cook-Torrancův, který popisuje bidirectional reflective distribution function (BRDF). Tento model je podobný Phong shadingu, ale narozdíl od něj dodržuje fyzikální korektnost. Odražené světlo nikdy nemůže přesahovat světlo dopadající, címž se vyrendrovaná věc zdá realističtější. Dalším důležitým algoritmem je radiosity algororithm. Tento lgoritmus počítá světlo polygonu součtem světla kolik vyzařuje a kolik světla přijmá od ostatních polygonů. Narodíl ale od starších algoritmu všechny polygony také nějaké světlo odrážejí, což je dáno povrchem polygonu. Toto odražené světlo se také počítá do celkového osvícení ostatních polygonů. Významné zlepšení proběhlo u Ray Casting algoritmu. Paprsek se místo toho, aby se o polygon zastavil, odrazí a algoritmus se znovu rekurzivně spustí pro tento paprsek. Toto zlepšení umožňuje vidět v odraze jednoho polygonu polygon druhý. [2, 24, 37, 33, 7]

V průběhu 90. let se významě vylepšovaly domácí počítače, které dokázali spustit i složitější programy a hry. 3D grafika se stala populárnější a vznikal pro ni i software jako Blender nebo 3D Studio. Nevznikli žádné nové revoluční renderovací metody, ale díky zlepšení výkonu počítačů se mohly začít využívat metody dříve vytvořené, které byly přílič náročné pro tehdejší počítače. Velký pokrok zaznamenal filmový průmysl. Terminátor 2 použil CGI na realistického humanoidního robota. V roce 1995 vydal Pixar první celovečerní plně počítačově animovaný film toy strory. 90. léta jsou označována jako zlatý věk videoher. První z důležitých her je Wolfenstein 3D. Tato hra využívá ray-casting, takže není 3D, ale vytváří iluzi 3D prostoru. Další důležitou hrou je Doom. Tato hra byla 2.5D, tedy kombinovala prvky 2D a 3D. Zdi a podlaha byly 3D modely, ale příšery byly 2D sprity, kterych bylo vice a ukazovaly se podle toho pod jakým úhlem se hráč na příšeru koukal. První plně 3D hra bula Quake. Quake měl pokročilý client-server multiplayer, takže bylo možno hrát ne jen na LAN, ale i přes internet. 3D modely umožnili vývojářům přidat do hry 3. rozměr a tím i více patrové mapy či létející projektyly a nepřátele. [5, 12, 9, 19, 17]

V novém tisíciletí se zlepšovalo vše, co se doposud vynalezlo. Renderovací algoritmy mohli býti více realisticé kvůli zlepšení GPU.

Společně s vývojem počítačové grafiky začaly vznikat i první videohry. Hra Spacewar! byla jedna z prvních, která se rozšířila po laboratoř9ch na amerických univerzitách. Hru vytvořil na MIT Steve Russell a jeho přátelé v roce 1962. Hra byla vytvořena pro dva hráče, každý ovládal svou loď a jeho cílem bylo sestřelit loď protihráče. Lodě byly ovládány speciálním boxem, který se dá považovat za předchůdce moderních herních kontrolerů. Hra běžela na počítači PDP-1, který namísto rasterové grafiky využíval grafiku vektorovou. Věci vykreslené

na osciloskopu byly definované matematickou funkcí. Díky tomu, že vektorová grafika nebyla tolik náročná na CPU jako rasterová, mohla ve hře být hvězda, která svojí gravitací působila na lodě. [39, 6, 45]

pridej pong

2.2 Architektura grafických karet

2.2.1 CPU vs. GPU

Tradičně běžela většina apklikací sekvenčně, které běží na procesorech, které mají centrální výpočetní jednotku a provádí instrukce jednu po druhé CPU. V průběhu 20. století se výkon CPU významě zlepšoval až na bilion operací za sekundu, čímž se mohli zlepšévat grafické aplikace a využívat náročnější funkce. Na přelomu tisíceletí se ale vývoj začal zpomalovat, kvůli problému se spotřebou energie a odvodem tepla. Proto výrobci začali vytvářet CPU s více jádry, ale ani to nebylo dostatečně výkonné na složíté výpočti, a proto se už sedmdesátých a osmdesátých letech začali vznikat počítače které nepracovali sekvenčně, ale paralelně. V devadesátých se začali vytvářet mikroprocesory, které se soustředili na paralelní výpočty GPU. V průběhu let se počet vláken z jednotek dostal na tisíce, což vytvořilo prostor pro inovaci v grafice. [22]

Prvně bylo GPU pouze v počítačích specializovaných pro 3D hry a vizualizace, ale postupně se stala součástí každého počítače. GPU a CPU se používá v různých případech kvůli rozdílné architektuře. CPU má jednotky až desítky jader a umí dělat jeden krok extrémě rychle kvůli sekvenčnímu zapojení. Velká část čipu je určena pro cachi, tedý malu pamět, kam se ukládají data která budou pravděpodobně v budoucnisti potřeba, takže není potřeba komunikovat neustále s RAM. Část čipu je také pro control logiku, která dokáže instrukce z jednoho vlákna dělat paralelně nebo na jiném sekvenčním pořadí, ale zachovat sekvenčnost celého procesu. CPU má také menší propustnost paměti než GPU, proto cache-friendly programy jsou výrazně rychlejší. Narozdíl od toho GPU má stovky až tisíce jader, která jsou ješte typicky rozdělena na 32 nebo 64 vláken. Kontrolní logika a cache jsou na jádro menší a jednodužší, takže velmi pomalu reaguje na události a po dokončení operaí se většinou data vrací zpět do VRAM, což je díky velké propustnosti i pro velmi velký objem dat rychlé. [22]

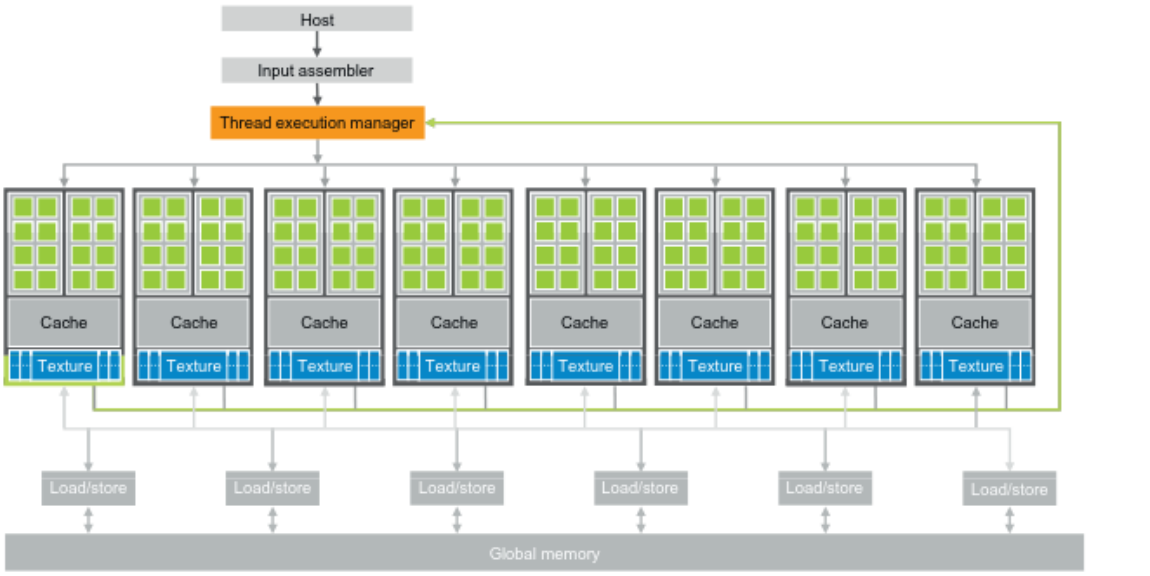


FIGURE 1.2
Architecture of a CUDA-capable GPU.

Obrázek 2.2: Architektura GPU podporující CUDA [22]

2.2.2 Paralelní architektura GPU

Do roku 2007 se GPU pouzival prevazne pro renderovani, ale v roce 2007 prisla NVIDIA s programovacim modelem pro paralelni vypocty CUDA. Tento model umoznil spousteni aplikaci na CPU a GPU, takze narocnejsi pocetni operace mohly bezet na GPU, a tim zvysit vykon. Pro ucel vysvetleni programovacich modelu pro paralelni vypocty budu vyuzivat model CUDA, ale existuji i jine, podobne fungujici modely jako OpenCL nebo SYCL.

2.3 Grafická pipeline

2.4 Transformace a lineární algebra

3 OpengGL

3.1 Historie OpenGL

3.2 OpenGL pipeline

3.3 Shadery

3.4 Textury

4 to do

historie compileru C celá část hirstorie počítačové grafiky je scuffed, chybí tam části o gpu, a nejspis to bude potreba zkratit.

Část II

Vývoj hry v OpenGL

5 Grafika a zvuk

5.1 Aseprite

5.2 Bosca Ceoil

6 Použité knihovny

6.1 GLAD

6.2 GLFW

6.3 stb_image

7 Herní scény

7.1 Scéna hlavního menu

7.2 Scéna pozastavené hry

7.3 Scéna hry

8 Herní mechaniky

8.1 Generování objektů

8.2 Pohyb hráče

8.3 Detekce kolize

Závěr

Bibliografie

1. *A Critical History of Computer Graphics and Animation* [Online]. [B.r.]. Dostupné také z: https://web.archive.org/web/20070405181508/http://accad.osu.edu/%7Ewaynec/history/lesson2.html. [citováno 2025-11-07].
2. AKENINE-MÖLLER, Tomas et al. *Real-Time Rendering*. 4th. A K Peters/CRC Press, 2018.
3. ALBATROSS. *History of C++* [Online]. [B.r.]. Dostupné také z: https://cplusplus.com/info/history/. [citováno 2025-10-17].
4. BANAHAN, Mike; BRADY, Declan; DORAN, Mark. *The C Book: Featuring the ANSI C Standart(Instruction Set)*. 2nd. Addison-Wesley, 1991. ISBN 9780201544336.
5. BHATTACHEJEE, Souktik. *A timeline of 3D softwares* [Online]. [B.r.]. Dostupné také z: https://www.re-thinkingthefuture.com/career-advice/a2944-a-timeline-of-3d-softwares/. [citováno 2025-12-02].
6. BRITANNICA. *vector graphics* [Online]. 2025. Dostupné také z: https://www.britannica.com/technology/vector-graphics/additional-info#history. [citováno 2025-11-12].
7. CCGOMENZ. *Radiosity Algorithm* [Online]. [B.r.]. Dostupné také z: https://ccgomezn.github.io/vc/docs/workshops/rendering/radiosity. [citováno 2025-11-17].
8. DIGITAL ARTS, Copenhagen Academy of. *WHAT IS PHONG SHADING? (GUIDE WITH EXAMPLES)* [Online]. [B.r.]. Dostupné také z: https://cada-edu.com/guides/what-is-phong-shading. [citováno 2025-11-15].
9. DOUGC. *What was the deal with 2.5D games like DOOM* [Online]. 2002. Dostupné také z: https://boards.straightdope.com/t/what-was-the-deal-with-2-5d-games-like-doom/105737/3. [citováno 2025-12-02].

10. ENGEL, Wolfgang. *The History of the GPU - New Developments*. 1st. Springer International Publishing, 2023. ISBN 9783031140464.

11. ENGINEERING; WIKI, Technology History. *Milestones:Whirlwind Computer, 1944-59* [Online]. 2024. Dostupné také z: `https://ethw.org/Milestones:Whirlwind_Computer,_1944-59`. [citováno 2025-11-06].

12. FRONT, Wolf. *Development History* [Online]. [B.r.]. Dostupné také z: `https://wolfenstein3d.nl/development-history/`. [citováno 2025-12-02].

13. GEEKSFORGEEKS. *Features of C* [Online]. 2025. Dostupné také z: `https://www.geeksforgeeks.org/c/features-of-c-programming-language/`. [citováno 2025-10-12].

14. GEEKSFORGEEKS. *Top 25 C++ Applications inf Real World[2025]* [Online]. 2025. Dostupné také z: `https://www.geeksforgeeks.org/blogs/top-applications-of-cpp-in-real-world/`. [citováno 2025-10-22].

15. GLOSSARY, Computer Graphics. *Bump Mapping - Definition & Detailed Explanation - Computer Graphics Glossary Terms* [Online]. 2025. Dostupné také z: `https://pcpartsgeek.com/bump-mapping/`. [citováno 2025-11-15].

16. HARBISON, Samuel P.; STEELE, Guy L. *C, a Reference Manual*. 5th. Prentice-Hall, 2002. ISBN 9780130895929.

17. HICKMAN, Zachary. *Quake Engine Analysis* [Online]. [B.r.]. Dostupné také z: `https://zhickman.com/analysisfinal.pdf`. [citováno 2025-12-02].

18. HUGHES, John F. et al. *Computer Graphics: Principles and Practice*. 3rd. Addison-Wesley Professional, 2013.

19. JPIOLHO. *[2021 Re-release] How multiplayer works* [Online]. 2021. Dostupné také z: `https://steamcommunity.com/sharedfiles/filedetails/?id=2624343184`. [citováno 2025-12-02].

20. KAY, Alan. *Vision & Reality of Hypertext and Graphical User Interfaces* [Online]. [B.r.]. Dostupné také z: `https://mprove.de/visionreality/text/3.1.2_sketchpad.html`. [citováno 2025-11-07].

21. KERNIGHAN, Brian W.; RITCHIE, Dennis M. *The C Programming Language*. 2nd. Pearson, 1988. ISBN 9780131103627.

22. KIRK, David B.; HWU, Wen-mei W. *Programming Massively Parallel Processors: A Hands-on Approach*. 3rd. Morgan Kaufmann, 2016.

23. KOCHAN, Stephen G. *Programming in C*. 4th. Pearson Education, 2015. ISBN 9780321776419

24. KONOW, David. *Putting the Original Tron's Special Effects Together* [Online]. 2015. Dostupné také z: `https://web.archive.org/web/20180512181521/http://www.tested.com/art/movies/520562-putting-original-trons-special-effects-together/`. [citováno 2025-11-16].

25. LABORATORY, Lincoln. *SAGE: SEMI-AUTOMATIC GROUND ENVIRONMENT AIR DEFENSE SYSTEM* [Online]. [B.r.]. Dostupné také z: `https://www.ll.mit.edu/about/history/sage-semi-automatic-ground-environment-air-defense-system`. [citováno 2025-11-06].

26. LABS, BCA. *Vision & Reality of Hypertext and Graphical User Interfaces* [Online]. [B.r.]. Dostupné také z: `https://bcalabs.org/subject/bresenhams-line-algorithm-in-computer-graphics`. [citováno 2025-11-11].

27. LEE, Sarah. *Mastering Gouraud Shading Techniques* [Online]. 2025. Dostupné také z: `https://www.numberanalytics.com/blog/mastering-gouraud-shading-techniques`. [citováno 2025-11-15].

28. LOGAN. *Z-Buffer Algorithm* [Online]. 2025. Dostupné také z: `https://www.onlycode.in/z-buffer-algorithm/`. [citováno 2025-11-15].

29. LONDON, Imperial College. *Ray Tracing* [Online]. [B.r.]. Dostupné také z: `https://www.doc.ic.ac.uk/~bkainz/graphics/notes/GraphicsNotes1011.pdf`. [citováno 2025-11-12].

30. MORTENSEN, Peter. *What's the difference between a low-level, midlevel and hight-level language* [Online]. 2023. Dostupné také z: `https://stackoverflow.com/questions/3468068/whats-the-difference-between-a-low-level-midlevel-and-high-level-language`. [citováno 2025-10-13].

31. NAPPER, Brian. *The Manchester Small Scale Experimental Machine – "The Baby"* [Online]. 1999. Dostupné také z: `https://web.archive.org/web/20120604211339/http://www.computer50.org/mark1/new.baby.html`. [citováno 2025-11-05].

32. NOLLE, Tom. *What is structured programming(modular programming)* [Online]. 2023. Dostupné také z: https://www.techtarget.com/searchsoftwarequality/definition/structured-programming-modular-programming. [citováno 2025-10-14].

33. OPENGL, Learn. *Theory, BRDF* [Online]. [B.r.]. Dostupné také z: https://learnopengl.com/PBR/Theory. [citováno 2025-11-16].

34. PEDDIE, Jon. *The History of the GPU - Eras and Environment*. 2nd. Springer Nature, 2023. ISBN 9783031135811.

35. PEDDIE, Jon. *The History of the GPU - New Developments*. 1st. Springer International Publishing, 2023. ISBN 9783031140464.

36. PEDDIE, Jon. *The History of the GPU - Steps to invention*. 1st. Springer International Publishing, 2023. ISBN 9783031109676.

37. RHINOCEROS. *What are NURBS* [Online]. [B.r.]. Dostupné také z: https://www.rhino3d.com/features/nurbs/. [citováno 2025-11-16].

38. SEVO, Daniel. *HISTORY OF COMPUTER GRAPHICS 1990-99* [Online]. [B.r.]. Dostupné také z: https://www.danielsevo.com/hocg/hocg_1990.htm. [citováno 2025-12-03].

39. SMITH, Ryan P. *How the First Popular Video Game Kicked Off Generations of Virtual Adventure* [Online]. 2018. Dostupné také z: https://www.smithsonianmag.com/smithsonian-institution/how-first-popular-video-game-kicked-off-generations-virtual-adventure-180971020/. [citováno 2025-11-12].

40. SRUTHY. *What Is C++ Used For? Top 12 Real-World Applications And Uses Of C++* [Online]. 2025. Dostupné také z: https://www.softwaretestinghelp.com/cpp-applications/. [citováno 2025-10-22].

41. STROUSTRUP, Bjarne. *Design and Evolution of C++, The*. 1st. Addison-Wesley Professional, 1994. ISBN 9780201543308.

42. STROUSTRUP, Bjarne. *Programming Principles and Practice Using C++*. 2nd. Pearson Education, 2014. ISBN 9780321992789.

43. STROUSTRUP, Bjarne. *The C++ Programming Language*. 4th. Pearson Education, 2013. ISBN 9780321563842.

44. TECH, WsCube. *Top Applications of C programming* [Online]. 2024. Dostupné také z: https://www.wscubetech.com/resources/c-programming/applications. [citováno 2025-10-09].

45. TILLEY, Thomas. *Spacewar! Controllers* [Online]. 2015. Dostupné také z: https://tomtilley.net/projects/spacewar/. [citováno 2025-11-12].

46. TODAY, Everything Explained. *Hidden-surface determination explained* [Online]. [B.r.]. Dostupné také z: https://everything.explained.today/Hidden-surface_determinatio [citováno 2025-11-15].

47. VOLLE, Adam. *C++ computer language* [Online]. 2025. Dostupné také z: https://www.britannica.com/technology/C-computer-language. [citováno 2025-10-17].

Zkratky

CPU Central processing unit. 9

CRT Cathode ray tube. 7, 12

GPU Graphics processing unit. 9

RAM Random access memory. 7

SSEM Small-Scale Experimental Machine. 7

Seznam obrázků

Seznam tabulek

Přílohy

A Fotky z pokusů

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

B Příloha další