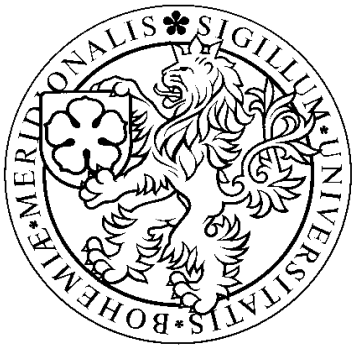


Příprava prezentace počítačové fyziky

oblast počítačová grafika, zpracování obrazu a integrální transformace

Diplomová práce

Bc. Martin Švarc

Jihočeská univerzita
Pedagogická fakulta
Katedra fyziky

Vedoucí práce: RNDr. Petr Bartoš, Ph.D.

České Budějovice 2009

Prohlášení:

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě - v úpravě vzniklé vypuštěním vyznačených částí archivovaných pedagogickou fakultou elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách.

V Českých Chalupách dne 11.11. 2009 Martin Švarc

.....

Poděkování

Rád bych zde poděkoval RNDr. Petrovi Bartošovi, Ph.D. za odborné vedení a podporu při tvorbě této práce a rodině za trpělivost.

Anotace

Cílem této práce je seznámit srozumitelnou a populární formou laickou veřejnost, a především budoucí studenty vysokých škol, s oborem počítačové fyziky. Součástí samotného vypracování je obsáhlejší prezentace a k ní tento doprovodný text v tištěné formě.

Obsahuje tyto části: Historie a vývoj výpočetní techniky, Historie a vývoj počítačových jazyků, Významné osobnosti počítačové fyziky, Počítačová grafika, Zpracování obrazu a Integrální transformace.

Annotation

The goal of this work is to introduce a subject of computer physics to the general public and especially to the prospective university students in an intelligible and popular way. A comprehensive presentation accompanied by this explanatory text is a part of the thesis.

The presentation is divided into these chapters: History and development of computer technology, History and development of the computer (programming) languages, Personalities of the computer physics, Computer graphics, Image processing and integral transformation.

Obsah

Úvod

Počítačová fyzika představuje řešení fyzikálních problémů za použití výpočetní techniky. I když se jedná o relativně mladý pojem, náznaky či „prapůvodní principy“ tohoto fyzikálního oboru by se daly nalézt již v počátcích vývoje lidského druhu.

Na začátku éry výpočetní techniky představovala počítačová fyzika v podstatě jakékoli využití počítače pro účely práce fyziků. Dnes se do jisté míry orientuje především na oblasti simulací a modelování reálných i čistě teoretických dějů. Nezávisí přitom z jakého oboru je přímo studovaný fyzikální proces, rozhodující je metodika, s níž přistupujeme k jeho řešení.

Její vznik nemůžeme zařadit do přesného času. Stejně tak nemůžeme vymezit ani její hranice. Bývá zařazována mezi dva základní směry fyziky, kterými jsou teoretická a experimentální fyzika, protože využívá matematického aparátu pro návrhy a simulace reálných procesů či experimentů. Podle jiného názoru je zase pouze poddisciplínou teoretické fyziky, protože v principu neustále jen řeší výpočty.

Oba pohledy mají svým způsobem pravdu, i když osobně se přikláním spíše k tomu prvnímu, protože díky silnému výpočetnímu aparátu můžeme získat vcelku přesný popis reálného světa.

Touto prací bych chtěl trochu přiblížit okolnosti, které postupně vedly ke vzniku počítačové fyziky. V posledních dvou kapitolách jsou pak uvedeny i některé její principy. Součástí vypracování je obsáhlejší prezentace, která je přiložena na médiu, následující text by měl blíže objasnit její obsah.

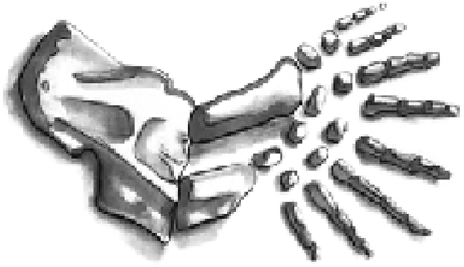
1 Historie a vývoj výpočetní techniky

Historie výpočetní techniky není nijak pevně vymezená. Od počátku se spíše jednalo o pomůcky, které měly člověku ulehčit jeho matematické úkony. Do dnešního stavu se tak dostala přes jednoduché počítání na prstech, počítadla, programovatelné tkalcovské stroje až k dnešním sofistikovaným technologiím, bez kterých by byl život pro většinu lidí na Zemi velice krušný a nemohl by vzniknout tak krásný a zajímavý obor, jakým počítačová fyzika bezesporu je.

1.1 Kosti zvířat - doba kamenná - paleolit

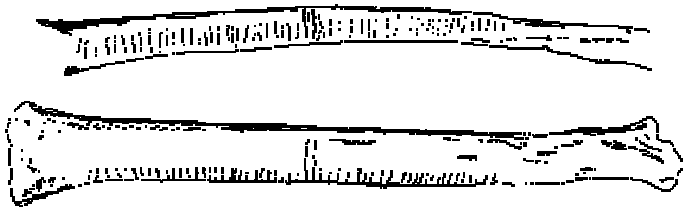
Pokud bychom se ohlédli do nejhlubší historie výpočetní techniky, museli bychom dohlédnout až do hlubin pravěku, kdy se mezi našimi prapředky objevili první „matematikové“. Početní výkony těchto jedinců však byly velice omezené, proto ke složitějším početním úkonům používali speciální pomůcky z kostí zvířat, na nichž dělali zářezy. Na prstech běžně počítaly pouze do dvou.

Zřejmě nejstarší nalezenou výpočetní pomůckou je kost paviána nalezená v jeskyni pohoří Lemombo v jižní Africe, její stáří se odhaduje na 35 000 let. Další pozůstatky přibližně stejného stáří byly nalezeny i na Sibiři či ve Francii.



Obr. 1.1: Jedna z pravěkých početních pomůcek - ruka dinosaura (Tetrapoda).

Další zajímavou početní pomůckou je lýtková kost z mladého vlka s 55 zářezy, která byla objevena 19. 8. 1936 v Dolních Věstonicích Karlem Absolonem. Ve své době byla považována za nejstarší výpočetní pomůcku na světě, její stáří je odhadováno přibližně na 25 000 let. [1]



Obr. 1.2: Kost vlka z Dolních Věstonic. [1]

1.2 Abakus - mladší doba kamenná - neolit

Abakus je již značně pokročilejší pomůckou, určenou i pro složitější výpočty, a stojí tak na půli cesty mezi počítáním na prstech a mechanickými kalkulátory. Poprvé se objevil zřejmě před 5000 lety př.n.l. na Blízkém východě. Je to zařízení určené pro početní operace typu sčítání, odčítání, násobení a dělení.

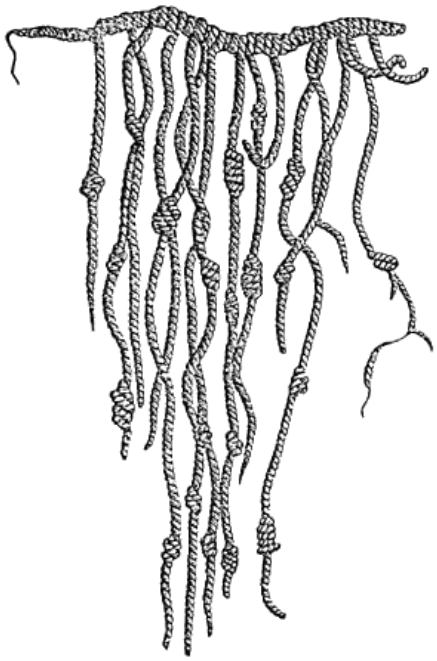
Jeho původní latinský název *abacus* je odvozen z od řeckého slova *abax* či *abakon* (ve smyslu tabulka či deska pokrytá prachem), nebo semitského slova *abq* (ve smyslu písek). [2]

Původně se jednalo o ploché kameny s vrstvou písku, ve které byly rýhy reprezentující jednotlivé řády. Dalšími variantou byla deska s počtářskými kameny, později destička se žlábky a nakonec i dnes nejpoužívanější verze - rám s kuličkami na tyčkách.

Abacus se používal a dodnes používá v různých provedeních po celém světě, převážně však na východě - viz přehledová tabulka 1.1. Existuje i verze pro nevidomé studenty v kombinaci s použitím tzv. Nemeth kódu.

Civilizace	Název abakusu	Název v původním jazyce
Aztékové, Mayové	Nepohualtzitzin	*provázkové písmo Kipu*
Čína	Suan pan	算盘, 算盤
Japonsko	Soroban	算盤, そろばん
Rusko	Sčot	Счёты
Korea	Chisanbop	지산법

Tab. 1.1: Přehledová tabulka variant abakusu.



Obr. 1.3: Nepohualtzitzin s provázkovým písmem Kipu.[14]

1.3 Mechanická kalkulačka - středověk

1.3.1 Leonardo da Vinci

První úspěšnou snahu o navržení mechanického kalkulátoru můžeme dle dochovaných záznamů přisoudit vynálezci Leonardu da Vinci a to na základě náčrtků a poznámek, obsažených v dílech Codex atlanticus a Codex Madrid.

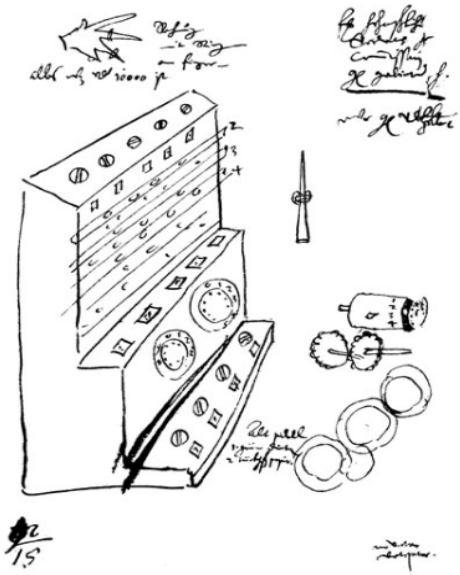
Funkčnost modelu byla ověřena v roce 1968, kdy byla Dr. Robertem Guatelim vytvořena replika zařízení za podpory společnosti IBM. Ta se stala jedním z exponátů výstavy IBM. Časem byla ale stažena kvůli četným diskusím týkajících se zřejmě neúplných poznámek da Vinciho a vlastní invence a představivosti dr. Guateliho. [3]

1.3.2 Wilhelm Schickard

První skutečně funkční model počítacího stroje sestrojil Wilhelm Schickard, a to v roce 1623. Toto zařízení bylo schopné vykonávat operace typu sčítání, odčítání, násobení, dělení, zároveň pracovalo s plovoucí desetinnou čárkou. Bohužel, ani jeden ze tří vyrobených kusů se nedochoval. Nízký počet vyrobených kusů byl dán nesmírnou technologickou a konstrukční náročností výroby vzhledem ke své době. Jedním z kritizovaných prvků byla právě nepřesnost zoubkování.

Shickard si dopisoval s řadou osvícenců své doby, mimo jiné s Johannesem Keplerem, který tento kalkulátor využíval pro své astronomické výpočty. [4]

Podle dochovaných návrhů byla v roce 1960 sestavena replika zařízení, která byla plně funkční. Jeden z exemplářů je k vidění např. i v Německém muzeu v Mnichově. Na počest prvenství byla v roce 1973 v Německu vydána i známka s počítacím strojem.



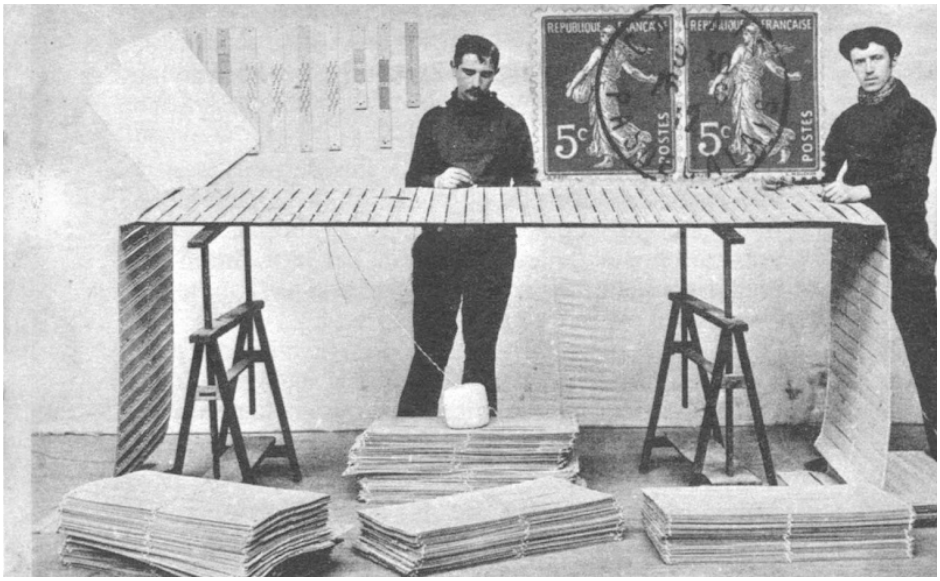
Obr. 1.4: Původní Schickardův nákres počítacího stroje. [4]

1.4 Analytický stroj - moderní dějiny

1.4.1 Joseph Marie Jacquard

Další etapu ve vývoji představovaly analytické stroje. Největším rozdílem oproti mechanickým kalkulačkám byla možnost programování. Prvním strojem této koncepce byl automatizovaný tkalcovský stav z roku 1805, jehož chod se ovládal pomocí děrných štítků. Autorem byl Francouz Joseph Marie Jacquard. Vyřešil tak tkaní složitě vzorovaných tkanin jednou osobou s možností tkát nezávisle velké vzory co nejjednodušším způsobem.

Napoleon vyplatil vynálezci za žakardu, jak se tento stroj začal nazývat, tři tisíce franků vládní odměny. Princip automatizace spočíval v tom, že Jacquard rozkreslil geometrické vzory na čtverečkovaný papír a ten rozřezal na pásy. Tam, kde měla procházet jehla s nití, vyrazil dírku. Pásy pak byly upevněny na osmistěnný otočný hranol. Stroj vytkával kilometry tkanin podle programu z papíru. [5]

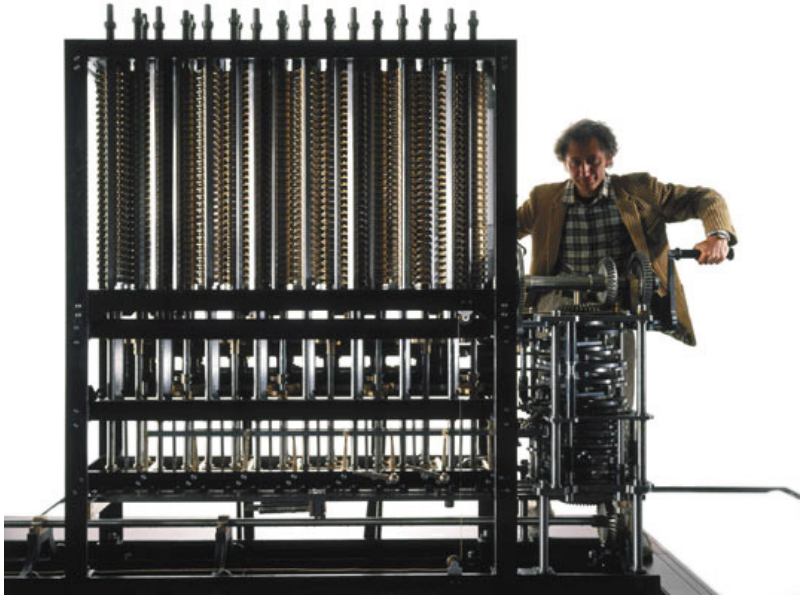


Obr. 1.5: Kartony pro programování žakáry. [6]

1.4.2 Charles Babbage

Jméno tohoto britského matematika je neodmyslitelně spjato s analytickým strojem. Ještě předtím ale začal vytvářet **Diferenční stroj**. Na ten dostal v roce 1822 od britské vlády dotace. Stroj o velikosti větší místnosti měl pracovat s 20 cifernými čísly, s diferencemi 6. řádu. V roce 1842 byl ale Babbageův projekt zrušen, vzhledem k obrovským nákladům a i faktu, že Babbage se více věnoval svému novému projektu - Analytickému stroji. Přesto byly ale díky jeho návrhům vytvořeny dalšími konstruktéry jiné jednodušší diferenční stroje. [7]

V roce 1991 pak byla postavena replika tohoto stroje. Stroj se pohání klikou, je vysoký přes 2 metry, dlouhý 4 metry a váží několik tun.



Obr. 1.6: Replika Diferenčního stroje. [8]

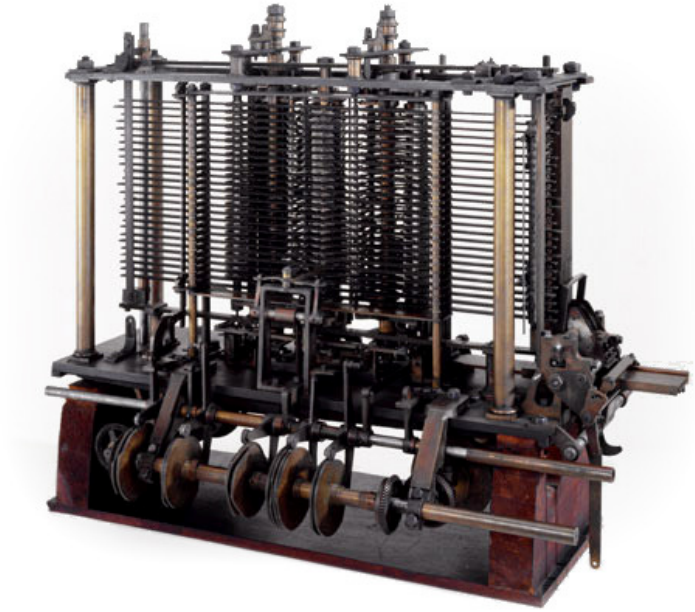
V roce 1834 začal navrhovat Babbage svůj **Analytický stroj**, v podstatě mechanický počítač. Svojí koncepcí to byl zároveň i první turing-kompletním počítačem (možná emulace stroje změnou programu bez nutnosti fyzické přestavby). Koncept stroje obsahoval mimo jiné i sklad (paměť), mlýn (procesor), děrné štítky (vstupní jednotku). Měl pracovat s padesátimístnými čísly s pevnou desetinnou čárkou, pohon mechanických částí měl obstarat parní stroj.

Babbage však narazil na několik překážek, tou nejpodstatnější byly finance. Tento fakt se pokoušel společně s Adou Lovelace kompenzovat sázkami v dostizích. Společně došli k „dokonalému“ systému, který jim měl zaručit výhru. Ada byla náruživým hazardérem a Babbage zase nutně potřeboval peníze na svůj stroj. Systém ale nefungoval. Lovelace prosázela rodinné jmění, Babbage ztratil důvěru vlády a tedy i zbývající finanční podporu a sotva unikl totálnímu bankrotu. Stroj, který měl být tvořen 50 000 součástkami, tedy nikdy nedokončil. Nepodařilo se to ani jeho synovi.

V roce 1960 byla postavena anglickými nadšenci zjednodušená replika tohoto stroje. K údivu všech, stroj spolehlivě fungoval a pracoval rychlostí, kterou vypočítal Babbage. Repliky obou zmíněných Babbageových strojů jsou v současnosti vystaveny v londýnském Science museum včetně několika původních prototypů.

Augusta Ada Byron King, hraběnka z Lovelace, dcera lorda Gordona Byrona, bývá považována za prvního programátora. Na její počest byl v roce 1979

pojmenován programovací jazyk Ada. Podle některých materiálů byla ale až asi čtvrtým programátorem v pořadí. - potom jí tedy „předběhli“ Babbage, jeho asistent a pravděpodobně i Babbageovi synové. [9]



Obr. 1.7: Replika Analytického stroje. [10]

1.5 Počítače nulté generace

Nultá generace využívala původní mechanické principy při použití prvních elektronických součástek, jednalo se tedy o elektromechanické stroje. Jelikož se jednalo o začátky výpočetní techniky, konstruované stroje byly prototypy bez sériové výroby. Základními součástkami byly relé či jejich kombinace s elektronkami. Kmitočet procesorových jednotek dosahoval maximálně 100 Hz. [11] Vývoj počítačů byl následně značně ovlivněn jejich využitím během druhé světové války.

1.5.1 Z1, Z2, Z3 - Konrad Zuse, Německo

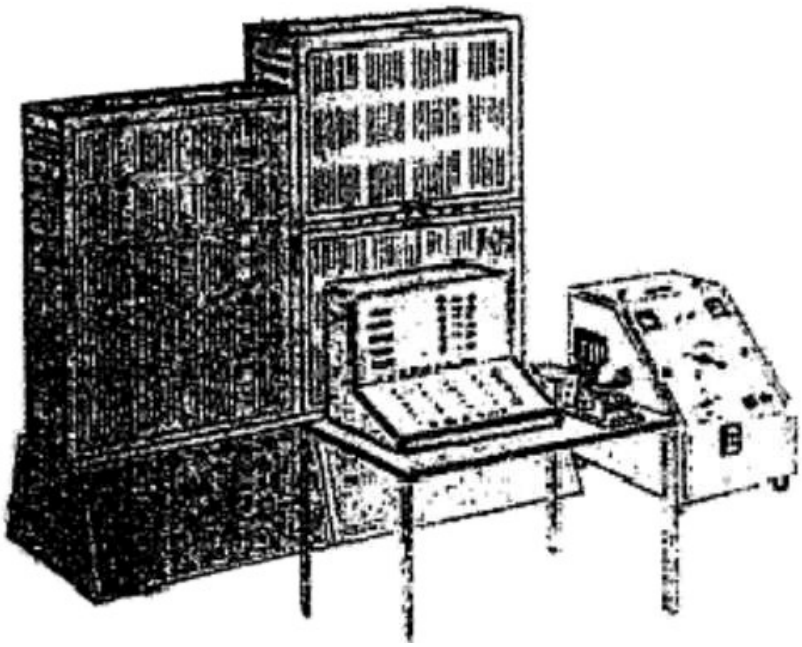
Tento německý inženýr byl autorem počítačů této generace. Prvním z nich byl počítač **Z1**, na kterém Zuse pracoval mezi lety 1934 až 1938. Do mechanické kolíčkové paměti bylo možné uložit 16 čísel, počítač ale nebyl příliš spolehlivý. Jelikož Zuse nebyl seznámen s principy, které uplatnil Babbage (v té době spíše upadly v zapomnění), nepracoval stroj ani s podmíněnými skoky.

Dalším typem byl **Z2**. Ten obsahoval stejný typ mechanické paměti jako Z1, ale na stavbu aritmetické a řídící jednotky bylo použito 800 starých relé z telefonní společnosti. Bohužel, fotografie i plány Z2 byly zničeny při

spojeneckých náletech během druhé světové války. Nicméně na Z2 si Zuse ověřil spolehlivost relé, které následovně použil pro svůj další typ počítače. [12]

Nejznámějším Zuseovým počinem je počítač řady **Z3**, první prakticky použitelný počítač na světě. Za ten později v roce 1964 obdržel Werner i von Siemensovu cenu, nejvyšší ocenění za technické vědy v Německu. Z3 prováděl 50 aritmetických operací za minutu v pohyblivé desetinné čárce. Do paměti se dalo uložit 64 čísel po 22 bitech, údaje se zadávaly pomocí klávesnice. Stroj se skládal z 2600 telefonních relé, stále nepodporoval instrukce podmíněného skoku. K programování se používal tzv. freiburský kód, z dnešního pohledu předchůdce assembleru. Z3 se používal ke složitým výpočtům charakteristik balistických raket V2. Ve válce nalezl využití i konec, byl zničen při náletu v roce 1944 v Berlíně, stejně tak i jeho fotografie a veškerá dokumentace.

Určitou zajímavostí bylo, že některé kombinace podmínkových a operačních znaků byly neprobádané a o některých se vědělo, že mohou být dokonce pro počítač nebezpečné. [13]



Obr. 1.8: Původní návrh Z3 Konrada Zuse. [12]

1.5.2 Mark I

I na druhé straně válečného konfliktu se zapojily počítače do války. Tím prvním byl ASCC (Automatic Sequence Controlled Calculator - automatický sekvenčně řízený počítač), známý pod civilním označením Mark I. Projekt pod vedením Howarda Aikena, financovaný firmou IBM, byl dokončen v roce 1943.

Počítač, či spíše monstrum, byl dlouhý 16 metrů při váze 4,5 tun. Sestaven byl ze 765 000 součástek a 800 km vodičů. Hnací jednotkou byla hřídel 4,5 kW

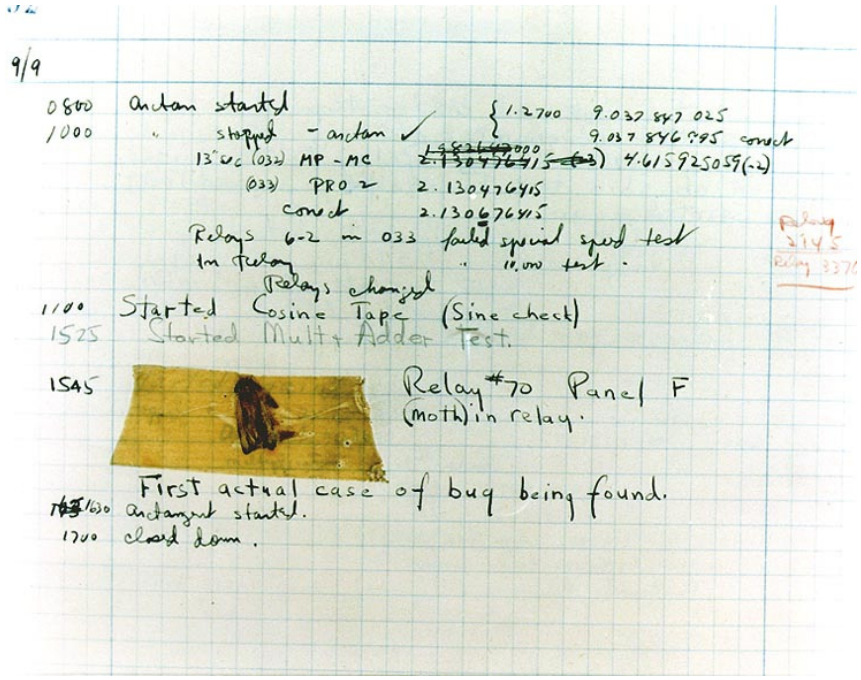
elektromotoru. Čím rychleji se hřídel točila, tím rychleji počítač pracoval. Díky tomu bylo zapotřebí mohutného chlazení a tedy i zesílenou přípojku elektřiny.

Mark I dovedl sečíst dvě čísla za 0,3 s, vynásobit je za 6 s a vypočítat např. hodnotu sinus daného úhlu během jedné minuty. [11] Používal se pro výpočty balistických tabulek.

1.5.3 Mark II

Další Aikenův projekt, čistě reléový, byl dokončen roku 1947. Místo původních elektromechanických relé bylo použito 13 000 vysokorychlostních elektromagnetických. Pracovat se dalo s čísly v desítkové soustavě, převod do dvojkové obstarávaly 4 relé. Součet dvou čísel trval průměrně 0,125 s a násobení 0,25 s. Paměť pojala 100 čísel při deseti platných číslicích. Počítač používalo americké námořnictvo.

Zvláštností je, že u tohoto počítače došlo k prvnímu zdokumentování počítačové chyby. Byla odhalena mezi dvěma body na relé # 70, panelu F.



Obr. 1.9: Dokumentace první odhalené chyby na počítači. [31]

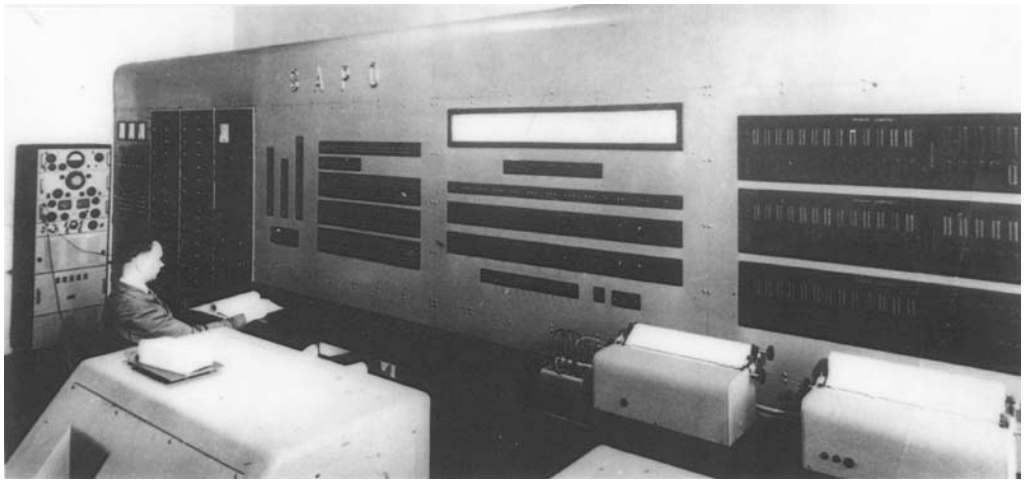
1.5.4 SAPO

V roce 1957 byl uveden do provozu SAPO (Samočinný POčítač), první počítač vyrobený v Československu. Zasloužil se o to tým Výzkumného ústavu matematických strojů pod vedením prof. Svobody a Dr. Oblonského.

Jeho koncepce byla sice technicky zastaralá, přesto obsahoval několik unikátních prvků. Jednalo se vlastně o tři stejné počítače, které pracovaly paralelně.

Výsledky z jednotlivých počítačů se mezi sebou porovnávaly a hlasováním se rozhodlo o konečném výsledku. Alespoň dva stejné výsledky znamenaly správný výsledek, jinak se operace opakovala.

Počítač zabíral celý sál, byl složen ze 7000 relé a 400 elektronek. V provozu byl pouze 3 roky, poté shořel. Příčinou bylo vzplanutí louže mazacího oleje od jiskry mezi reléovými kontakty.



Obr. 1.10: SAPO, první československý počítač. [15]

1.6 Počítače první generace

Přes svojí poruchovost se ukázaly počítače během války jako vysoce užitečné stroje, proto se do nich začalo více investovat. Základními součástkami v klopných obvodech se staly elektronky. Od relé se ustupovalo kvůli jejich rychlosti a nespolehlivosti. Charakteristický byl diskrétní režim práce, který se ale kvůli pomalému zadávání vstupních dat lidskou obsluhou ukázal později jako neefektivní. Počítače dosahovaly obřích rozměrů, obrovské spotřeby energie a stále byly relativně pomalé.

1.6.1 ENIAC

V roce 1944 byl v Pensylvániil uveden do provozu první elektronkový počítač - ENIAC (Electronic Numerical Integrator And Computer). Za jeho vývojem byli především John W. Mauchly, John Presper Eckert a John Von Neumann.

Jednalo se o monstrum složené z 17 648 elektronek, 10 000 kondenzátorů, 70 000 rezistorů, 1500 relé, vše umístěné ve 40 skříních. Zabíral plochu 150 m^2, chlazen byl dvěma leteckými motory a vážil zhruba 30 tun. Při spotřebě 150 kW (v té době jako valná část osvětlení Filadelfie) dosahoval taktovací frekvence 100 Hz,

střední doba mezi dvěma poruchami činila 80 minut. V průměru dokázal sečíst 5000 čísel za sekundu, počítal dekadicky.

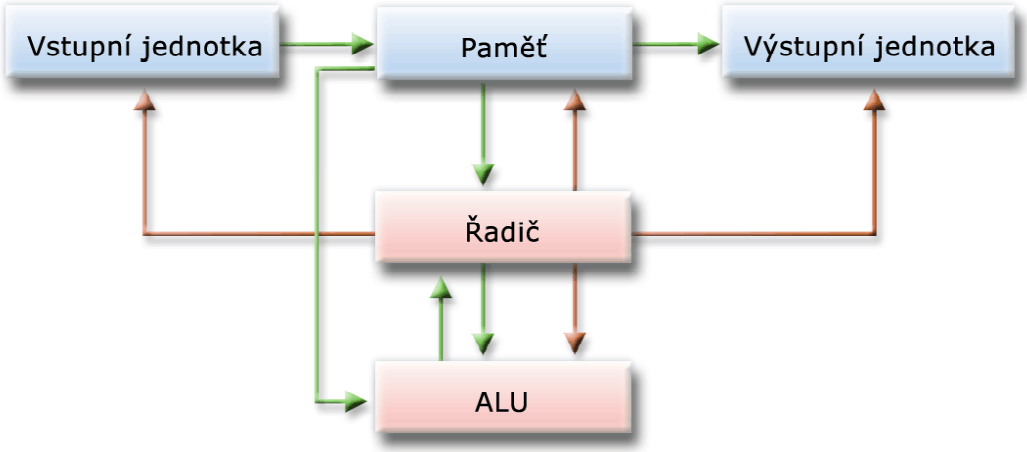
Jeho existence byla veřejnosti do roku 1946 přísně utajovaná, počítač sloužil pro balistický výzkum, jeho činnost byla ukončena v roce 1955. [16]

1.6.2 MANIAC

Po vzájemných neshodách s oběma dalšími tvůrci ENIACu ohledně koncepce počítačů uvádí J. V. Neumann do provozu MANIAC (Mathematical Analyser Numerical Integrator And Computer). Byl použit pro vývoj vodíkové bomby v atomové laboratoři v Los Alamos.

1.6.3 Manchester Mark

U tohoto počítače z roku 1948 se objevilo několik významných technických novinek. Tou první byla nová, tzv. **Von Neumannova architektura** s programem uloženým v paměti. Další novinkou byly paměťové obrazovky, známé dnes třeba z osciloskopů. Na Marku jich bylo hned 6, na každou z nich se dalo uložit 2048 bitů, celkem tedy 12 kB informací. Alan Turing pro tento počítač vymyslel jednoduchou formu jazyka adres - **Assembler**. [16]



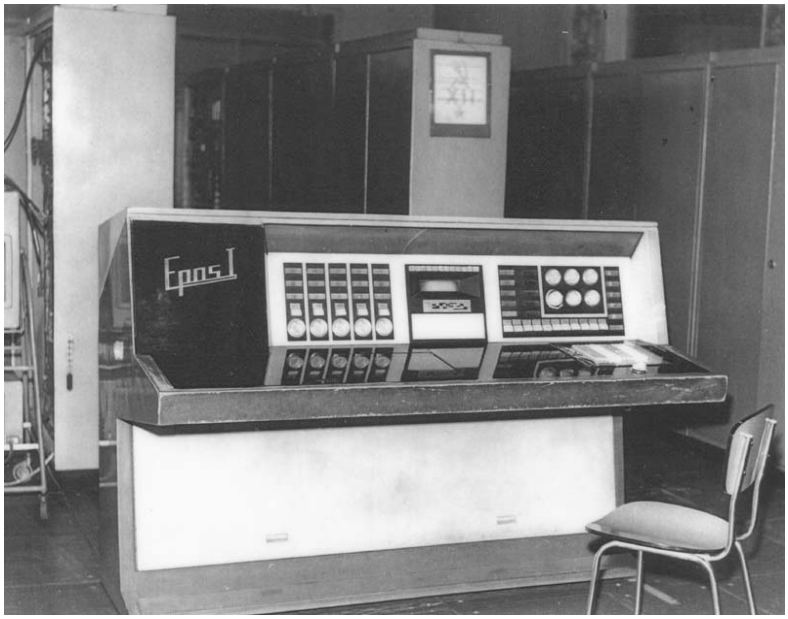
Obr. 1.11: Von Neumannova architektura.

1.6.4 Epos 1

Po úspěchu počítače SAPO se tým prof. Svobody soustředil na nový projekt, tím byl Epos 1. Byl určen především pro zpracování hromadných dat. Obsahoval několik originálních řešení. Jedním z nich bylo např. multiprogramování, český vynález. Díky tomu mohl počítač zpracovávat až pět programů současně.

Projekt původně počítal se sériovou výrobou, proto struktura počítače byla modulární. Jednalo se o základní jednotku tvořenou vlastním základním počítačem

a pak různé přídavné vstupní a výstupní jednotky. Prof. Svoboda však v této době emigroval do USA. Upravený konečný průmyslový vzor pak obsahoval 8000 elektronek a příkon počítače byl 300 kW. Původní koncept obsahoval 3400 elektronek a příkon činil 80 kW. Vzhledem ke střední době mezi dvěma chybami (asi 80 minut) se nedočkal velké sériové výroby. [17]



Obr. 1.12: Epos 1 - Elektronický POčítač Střední.

1.7 Počítače druhé generace

Tato generace počítačů byla osazena novou základní součástkou - tranzistory. Ty byly objeveny v roce 1947 v Bellových laboratořích.

Ačkoliv už první typy tranzistorů byly výrazně menší, energeticky úspornější a odolnější než elektronky, v počítačích se začaly používat až o řadu let později. Hlavním důvodem byla nechuť výrobců elektronek podporovat technologii konkurence. V této době se také objevily první programovací jazyky (Fortran, Cobol, Algol).

1.7.1 Tradic

Počítač Tradic (TRAnsistor DIgital Computer) byl prvním plně tranzistorovým počítačem. Do provozu byl uveden v Bellových laboratořích roku 1954.

Skládal se zhruba z 800 tranzistorů a byl 20krát rychlejší (milión logických operací za vteřinu) než jeho předchůdci. Ke svojí činnosti potřeboval pouze 160 W, což bylo více než 1000krát méně než jeho elektronkoví předchůdci. Navíc

dramaticky vzrostla i spolehlivost, během dvou měsíců se při nepřetržitém provozu vyskytly pouze dvě chyby. Vzhledem k tomu, že byl dostatečně malý a lehký, byl použit ve strategickém bombardéru dlouhého doletu Boeingu B-52 Stratofortress. Další počítače této generace byly o něco větší se spotřebou zhruba do 25 kW.

Obr. 1.13: Tradic vážil pouze 200 kg. [18]

1.7.2 DP 100

Jednalo se o první komerčně úspěšný počítač v ČSSR, na jeho výrobě se podílel Výzkumný ústav matematických strojů a podnik Aritma. Byl určen ke zpracování hromadných dat, jako řídící prvek děrnoštítkových výpočetních soustav.

Konstruktéři se zaměřili na jednoduchost a spolehlivost stroje, což se ukázalo jako správný záměr. V roce 1967 se začal sériově vyrábět. Během následujících let se prodalo zhruba 200 kusů, což činilo asi 1/8 trhu s počítači v tehdejším východním bloku.

1.8 Počítače třetí generace

Postupné zmenšování součástek vedlo v roce 1958 ke vzniku integrovaného obvodu. Ten se pak stal i základním prvkem nové generace počítačů. Na malou destičku křemíku (čipu) o ploše několika čtverečních milimetrů se podařilo dostat zpočátku desítky, později i miliony mikroskopických tranzistorů a dalších základních součástek.

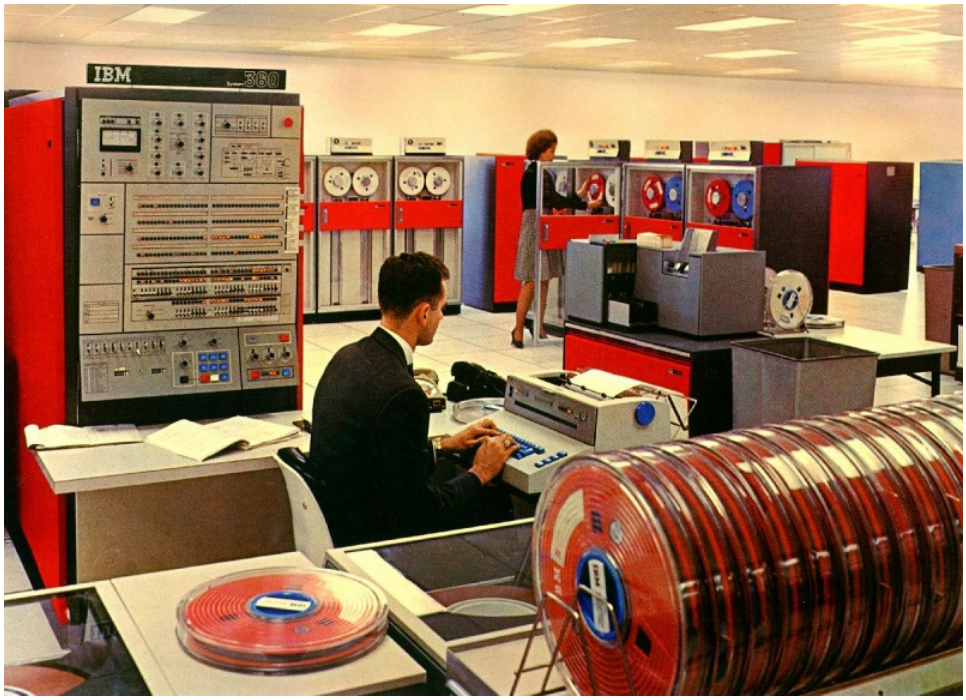
V této době se zároveň vzniklo multiprogramování a multitasking. Kromě sálových počítačů se začaly objevovat i stroje více se již blížící svou velikosti dnešním osobním počítačům. Vzniká i první PC (Personal Computer).

V roce 1965 byl definován tzv. **Moorův zákon**, nebo spíše empiricky vysledované pravidlo. Říká, že hustota tranzistorů na čipech se každých 12 - 18 měsíců zdvojnásobí. Platí dodnes.

1.8.1 System/360

Tento sálový počítač od IBM z roku 1964 přišel s novou, komerčně dostupnou modulární koncepcí. Tento krok byl v IT oblasti doslova revoluční, protože do této doby byly všechny monolitické počítače de facto vzájemně nekompatibilní.

V rámci této řady bylo k dispozici 5 procesorů při 19 kombinacích výkonu, rychlosti i paměti. To znamenalo razantní snížení ceny a dostupnost i mimo vědeckou a vojenskou sféru. Jednalo se o velice úspěšnou řadu a na trhu zůstala až do roku 1977.



Obr. 1.14: Sálový počítač IBM System/360. [19]

1.8.2 Tesla 200

Tento sálový počítač byl uveden do výroby roku 1969. Projekt byl vypracován ve spolupráci s francouzskou firmou BELL-GE na základě jejího počítače Bull Gamma 140. Tvůrci se inspirovali západním modelem System/360, tomu pak odpovídal vzhled zařízení i jeho parametry.

Tímto počítačem byla v průběhu let 1970-74 vybavena většina výpočetních středisek vysokých škol. [16]

1.9 Počítače čtvrté generace

Rokem 1968 začala éra současných počítačů, vytvořením mikroprocesorů (integrovaných obvodů velmi velké integrace - VLSI). Vzdálenost mezi jednotlivými součástkami se zmenšila na 0,001 mm. Malý čip tak měl stejný výkon jako např. sálový Eniac z roku 1944.

Během sedmdesátých let době se z marketingových důvodů začíná používat termín „generace počítačů“ a následovalo i zde použité postupné zařazení výpočetních pomůcek. I když pokud bychom chtěli být skutečně exaktní, když mluvíme o počítači - počítač ve skutečnosti nic nepočítá, ale porovnává.

1.9.1 IBM PC (model 5150)

První počítače pro širokou veřejnost byly konstruovány kvůli „lidovější“ ceně jako stavebnice. Dostupné tak byly spíše zdatným nadšencům. Tento fakt změnil 12. srpna 1981 právě **IBM PC** s operačním systémem tehdy ještě neznámé firmy Microsoft. Od firmy IBM to byl poměrně velký risk, protože stroje podobných parametrů, velikosti a především nízké ceny (od $ 1600) se do té doby prostě nevyráběly.

Tento stroj ve své době oslňoval své uživatele úctyhodnými parametry. Pracoval s procesorem Intel 8088 při frekvenci 4,77 MHz, 16 kB RAM s možností rozšíření až na 256 kB. Neobsahoval pevný disk, který byl příliš velký, standardní součástí však byla disketová mechanika. Monitor byl monochromatický, alfanumerický.

Termín PC (Personal Computer) je svým původem známý především v souvislosti s tímto modelem. Ve skutečnosti byl ale použit už v roce 1973 u počítače **Xerox Alto**, což byl jakýsi předchůdce osobních počítačů. Nejednalo se však o komerční produkt, několik tisíc exemplářů bylo využíváno výpočetními středisky a vysokými školami. IBM PC jej později zcela zastínil a stal se etalonem osobních počítačů na dlouhá léta.

Obr. 1.15: IBM PC z roku 1981. [20]

1.10 Počítače páté generace

Jedná se spíše o počítače let budoucích. V současnosti se k nim přibližují pouze ty nejvýkonnější superpočítače. Předpokládá se, že by měly umět pracovat s informacemi a ne pouze daty. Ke zpracování by měly sloužit stovky či tisíce paralelně zapojených procesorů. Základem by měly být biologické systémy na bázi neuronových sítí využívající výpočetního výkonu kyseliny DNA či systémy jim podobné. V každém případě se již nebude jednat o Von Neumannovskou architekturu.

Jedním z typických představitelů této generace by mohl být třeba HAL9000 z filmu 2001: Vesmírná odysea od Arthura C. Clarka. Jeho míra lidství se bohužel projevila i psychickou poruchou, což se negativně projevilo na dočasném životě posádky vesmírné lodi. Snad se jeho skuteční nástupci vyvarují podobných chyb.

1.11 Superpočítače

Nejvýkonnější počítače označujeme termínem superpočítače. Hranice pro takové označení není pevně stanovená, zpravidla se ale uvádí minimálně desetinásobný výkon oproti běžně dostupným počítačům. Takový výkon je nutný pro náročné aplikace jako např. modelování počasí, jaderných výbuchů, genomiky, ale i katalogizaci internetových stránek v internetu či hraní šachu. Klasickým

způsobem pro vytvoření superpočítače je realizace díky vysokorychlostnímu síťovému propojení mezi velkým množstvím procesorů.

Výkon se počítá ve flopech (FLoating-point OPerations per Second), čili počtu operací v plovoucí řádové desetinné čárce za sekundu. Samotný výpočet se realizuje pomocí světově uznávaného benchmarku LinPack a jeho modifikované verzi. Naprostá většina superpočítačů využívá UNIX a OS Linux.

1.11.1 Amálka

Je opět nejvýkonnějším českým superpočítačem. Byla jím do března 2009, kdy ji krátce předběhl Altix ICE 8200, ale v listopadu 2009 byl její výkon navýšen na 6,38 TeraFlops. Výkonem odpovídá 650 běžným kancelářským stolním počítačům, zvládne zpracovat více než šest bilionů početních operací za sekundu. Je tak mimo pořadí v žebříčku nejvýkonnějších počítačů Top500.

Amálka stojí za úplně prvním kinetickým modelem magnetického pole Merkuru, studiem bezesrážkového slunečního plazmatu a pomáhá v americkém projektu ARTEMIS. Průměrná doba řešení úlohy je zhruba 6 dní. Provozovatelem je Ústav fyziky atmosféry Akademie věd ČR. [50]

1.11.2 The Cray XT5 Jaguar

Je k listopadu 2009 nejvýkonnějším superpočítačem dle žebříčku Top500. To je seznam 500 nejvýkonnějších počítačů světa. Jeho výkon je 1,756 PetaFlops, obsahuje 224 162 procesorových jader, 300 TB RAM, uložiště 10 PB. Jeho příkon je téměř 7 MW. Svým okamžitým výkonem je více než 275krát výkonnější, než česká Amálka.

Obr. 1.16: Jaguar, nejvýkonnější superpočítač - listopad 2009. [21]

1.11.3 Internetové gridy

Fenomén počítačových sítí nabízí další možnost, překračující možnosti klasických superpočítačů. Gridy vznikají sesíťováním nejen klasických osobních počítačů, ale třeba i herních konzolí PlayStation.

Nejznámějším počítačovým gridem bylo hledání mimozemských civilizací pomocí spořiče displeje v projektu Seti@Home. V současnosti je nejvýkonnějším gridem projekt Folding@Home, který se zabývá biochemickými projekty. Jedná se např. o výzkum léků na rakovinu či modelování proteinových řetězců. Výkon tohoto gridu (k listopadu 2009) je 4,3 PetaFlops, čili téměř 2,5krát rychlejší než Jaguar.

1.11.4 Lidský mozek

Nabízí se určitě i porovnání nejvýkonnější výpočetní techniky s lidským mozkem. S tím přišel Dharmendra Modha, ředitel oddělení rozpoznávacích systémů společnosti IBM [22]. Mozek má podle něj výkon 38 Pflops při úložné kapacitě 3584 TB. Výkonově jsme tedy na tom téměř 9krát lépe než grid Folding@Home a téměř 6000krát než česká Amálka. Je nutné si ale uvědomit, že běžný člověk využívá pouze 4 % mozkové kapacity, géniové jako Einstein. Neumann a další i okolo 7 %.

Využití mozkové kapacity se stále mírně zvyšuje, plného využití kapacity ale není nikdy možné dosáhnout. Mozek řídí i velkou část složitých dějů v našem těle, které nejsme schopni ovlivnit - metabolismus, dýchání apod.

Současné superpočítače jsou lepší v jednotlivých výpočtech, mozek je ale výrazně lepší ve své komplexnosti. Jeho výzkum se pak stává inspirací pro neuronové sítě, které se tak více přibližují k principům fungování mozku.

A na konec ještě jedno důležité porovnání - mozek váží zhruba 1,4 kg a zabírá objem 1950 cm^3, nejvýkonnější superpočítače zabírají celé budovy s příkonem řádově v megawattech.

2 Historie a vývoj počítačových jazyků

I když si to neuvědomujeme, i my do jisté míry využíváme dvojkové soustavy. Ta je reprezentována přítomností a nepřítomností signálu v některých nervových vláknech. Mozek pak interpretuje tento signál do pro nás srozumitelnější a jednodušší podoby.

Podobně fungují počítače ve strojovém kódu. Jeho instrukce převádíme ve vhodných počítačových jazycích do pro nás jednodušší podoby. Jedna instrukce počítačového jazyka pak v sobě obsahuje více instrukcí strojového kódu. Tato skutečnost nám pak pomáhá výrazně rychleji definovat naše představy. Softwarové prostředky se vyvíjely v součinnosti s hardwarem, tomu pak odpovídá i stejné označení jednotlivých generací.

Pro ilustraci syntaxe v jednotlivých generacích programovacích jazyků je u všech typů znázorněn kód programu „Hello World!“ (Ahoj světe!).

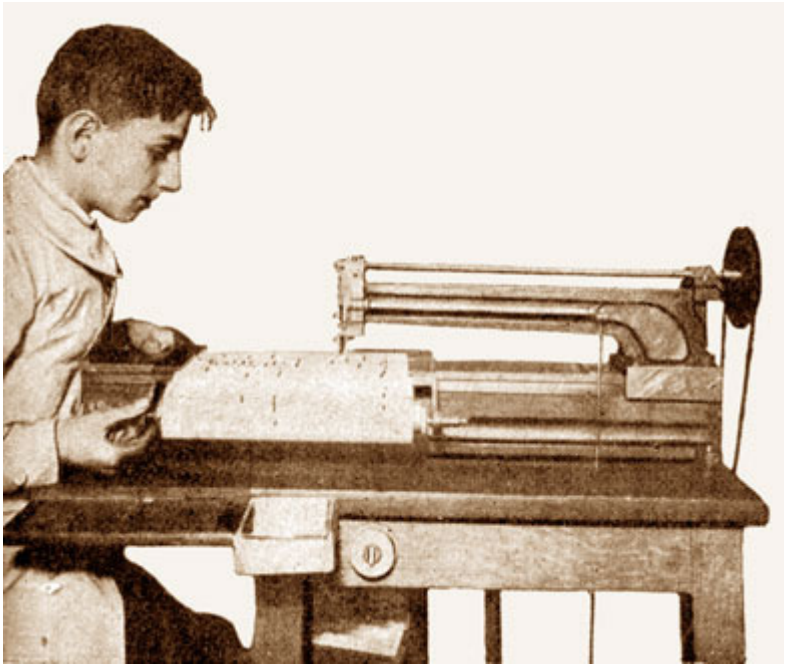
2.1 Počítačové jazyky 0. generace

Principem bylo v podstatě využití principu dvojkové soustavy, kdy signál byl dán přítomností či nepřítomností nějakého mechanického elementu. Tím mohly být hřeby na válci, ozubená kola nebo třeba „děrovaný papír“, obdoba pozdějších děrných štítků. Tohoto principu programování tak využívaly všechny mechanické a elektromechanické stroje.

Způsob programování byl jednoduchý na pochopení. Na druhou stranu, složitější stroje narážely na technologické omezení své doby. Mechanické součástky nebylo možno vyrobit s dostatečnou přesností a nebyl k dispozici ani dostatečně kvalitní materiál. Rozbití jedné součástky představovalo dlouhé hledání závady a rozebrání složitého stroje.

Za prvního programátora bývá označována hraběnka Ada Lovelace, i když se o tom později objevily určité spekulace - viz závěr kapitoly 1.4 Analytický stroj. Za první skutečný program je považován její návrh na výpočet Bernoulliho čísel. Zároveň zavedla princip programové smyčky, algoritmus pojmenovala na počest perského matematika Al-Chorezmiho, který objevil princip této metody již okolo roku 820.

Jen pro zajímavost, slovo **algoritmus** vychází právě ze jména tohoto slavného matematika a otce algebry. Latinizací a tedy zkomolením se došlo ke jménu Al-Gorizmi a později Algoritmí. Ve středověkých skriptech se pak objevovala formule „Dixit Algorizmi“, která byla zárukou spolehlivosti, jasnosti a zároveň argumentem, o kterém se nedalo pochybovat. [23]



Obr. 2.1: Programování v roce 1914. [24]

2.2 Počítačové jazyky 1. generace

Během této doby začaly vznikat na prototypech počítačů jednoúčelové programy zaměřené na matematické a fyzikální výpočty. Programovacím jazykem byl přímo strojový kód počítače. To teoreticky znamená maximálně výkonně ale zároveň i velice nepřehledně a pomalu, pokud se jedná o dvojkový kód. Většinou se proto přecházelo na osmičkový či šestnáctkový kód, které jsou pro člověka přehlednější.

Programátor tak zapisoval přímo číselné kódy, odpovídající jednotlivým instrukcím (příkazům) procesoru. Kódy byly umístěny na pevně daných adresách v paměti, každá instrukce měla jeden nebo více bytů. Počítač byl vybaven prostředky k převodu binárních čísel na osmičková (šestnáctková) a naopak. [25]

Programátoři měli oproti dnešku těžkou práci. Veškeré programy byly závislé na konkrétním procesoru. Doba programování byla na dnešní poměry neúměrně dlouhá a velmi dlouhý byl i samotný kód programu. I když se nepoužívala dvojková soustava, kód byl nepřehledný a často tak docházelo k chybám, které se obtížně hledaly.

Strojový kód označujeme jako strojově závislý programovací jazyk nižší úrovně. Běžně se v něm dnes již neprogramuje.

```
00000000: 27 42 45 47 49 4E 27 0D|0A 20 20 20 27 43 4F 4D
00000010: 4D 45 4E 54 27 20 48 65|6C 6C 6F 20 57 6F 72 6C
00000020: 64 20 69 6E 20 41 6C 67|6F 6C 20 36 30 3B 0D 0A
00000030: 20 20 20 20 4F 55 54 50|55 54 28 34 2C 27 28 27
00000040: 27 28 27 48 65 6C 6C 6F|20 57 6F 72 6C 64 21 27
00000050: 29 27 2C 2F 27 29 27 29|0D 0A 27 45 4E 44 27
```

Obr. 2.2: „Hello world!" - ve strojovém kódu, přeloženo z: Algol 60.

2.3 Počítačové jazyky 2. generace

V průběhu 50. let minulého století se přešlo na jazyk symbolických adres (assembly language). Jednotlivé instrukce a konstanty dostaly odpovídající symbolické názvy. Pro překlad instrukcí do strojového kódu byl určen assembler (kompilátor, překladač). Pozdější verze assembleru již obsahovaly i nástroje pro ladění programu. V kódu pak odpovídá každému řádku odpovídající instrukce ve strojovém kódu.

Zásadní výhodou bylo to, že si stačilo zapamatovat pouze jednoduché názvy místo složitých číselných kódů. Tím se zefektivnila rychlost programování i snížilo množství chyb. Zároveň se zmenšila velikost programů, což se hodilo, protože tehdejší operační paměti nevynikaly svou velikostí.

Nevýhodou zůstala závislost programů na typu procesoru. Každá rodina procesorů má svůj specifický jazyk symbolických adres, kvůli odlišnému způsobu rozdělování paměti. Zároveň jedna instrukce v assembleru odpovídala jedné instrukci strojového kódu, proto i tento jazyk nazýváme jazykem nižší úrovně. Používá se často i v současnosti - většinou k programování hardwaru vyžadující činnost v reálném času nebo pro jednoúčelové systémy.

Před nástupem další generace počítačových jazyků se ještě objevily tzv. autokódy. Byly jakýmsi přechodem mezi oběma generacemi. Byly stále vázány na typ procesoru, ale jedné jejich instrukci odpovídalo více instrukcí ve strojovém kódu. V současnosti se nepoužívají.

```
mov ax,cs
mov ds,ax
mov ah,9
mov dx, offset Hello
int 21h
xor ax,ax
int 21h

Hello:
  db "Hello World!",13,10,"$"
```

Obr. 2.3: „Hello world!" - Assembler-Intel. [27]

2.4 Počítačové jazyky 3. generace

V průběhu 60. let došlo k zásadní změně. Poprvé se objevily první vyšší programovací jazyky. Tímto označením je míněno, že jedné jejich instrukci odpovídá více instrukcí strojového kódu. Tyto jazyky nebyly strojově závislé, byly závislé pouze na operačním systému. Podporovaly metody strukturovaného programování, byly členěny do několika autonomních funkčních celků - modulů. Všechny tyto výhody znamenaly výrazné zjednodušení práce a mnohem větší dostupnost pro různé oblasti programování.

Jazyky této generace jsou v současnosti nejrozšířenější, jsou to např. BASIC, Pascal, jazyky typu C, Java, Python, Perl, Fortran … . Jejich pozdější verze již většinou uplatňují objektově orientovaný přístup a patří tak k hybridním OOP jazykům - viz následující generace.

2.4.1 Fortran

Tento programovací jazyk byl prvním své generace. Jeho název je akronymem slov FORmula TRANslator. Byl vytvořen v roce 1954 týmem společnosti IBM jako efektivnější alternativa k jazyku symbolických adres pro programování sálového počítače IBM 704. Byl určen pro vědecké výpočty a numerické aplikace.

Krátce po svém uvedení se stal leaderem mezi ostatními programovacími jazyky, především kvůli jeho efektivnosti a snadné pochopitelnosti. Postupem doby se jazyk vylepšoval až do dnešní podoby, kdy přibyla podpora pro datová pole, objektově orientované programování či genetické programování. Poslední verzí je zatím Fortran 2008.

Fortran je zároveň jedním z nejpoužívanějších programovacích jazyků pro účely počítačové fyziky. Je velmi oblíben např. v Japonsku.

```
      PROGRAM HELLO
      WRITE (*,100)
      STOP
100   FORMAT (' Hello World! ' /)
      END
```

Obr. 2.4: „Hello world!" - Fortran. [27]

2.4.2 Algol

Je dalším z prvních vyšších programovacích jazyků, byl uveden v roce 1958. Jeho název je tvořen akronymem slov ALGOrithmic Language. Byl navržen téměř výhradně na řešení matematických algoritmů. Byly v něm poprvé použity

konstrukce if/then/else, příkaz for, přepínač, volání procedur jménem nebo hodnotou atd..

Samotný jazyk se nikdy příliš nerozšířil, stal se ale významnou inspirací pro řadu dalších slavnějších nástupců, jako např. Pascal.

```
'BEGIN'
     'COMMENT' Hello World in Algol 60;
      OUTPUT(4,'('''('Hello World!')',/')')
'END'
```

Obr. 2.5: „Hello world!" - Algol-60. [27]

2.4.3 Cobol

Tento programovací jazyk, uvedený v roce 1960, byl určený na oblast zpracování hromadných dat v ekonomické oblasti. Název je akronymem slov Common Business Oriented Language. Syntaxe jazyka byla vytvořena tak, aby byl jazyk použitelný i pro uživatele s omezenými matematickými schopnostmi.

Brzy po svém uvedení se stal prakticky celosvětově univerzálním prostředkem k tvorbě programů pro zpracování ekonomických dat. Náklonnost uživatelů si získal způsobem zápisu kódu, který byl velmi blízký běžné angličtině. Cobol představuje podle několika průzkumů nejméně polovinu ze všech obchodních a finančních aplikací a to především pokud jde o skutečně velké firmy.

Zajímavostí je, že tvar zápisu byl odvozen z formátu děrného štítku. Cobol totiž původně vznikl z jazyku FLOW-MATIC, který vytvořila slavná programátorka Grace Murray. Tato dáma byla dokonce jedním ze tří programátorů prvního číslicového počítače MARK 1, který vlastnila armáda. Za svoji práci obdržela celou řadu ocenění. Do důchodu odešla s hodností kontradmirála a po její smrti byl na její počest pojmenován nový torpédoborec americké armády. [26]

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO.
ENVIRONMENT DIVISION.
DATA DIVISION.
PROCEDURE DIVISION.
MAIN SECTION.
DISPLAY "Hello World!"
STOP RUN.
```

Obr. 2.6: „Hello world!" - Cobol. [27]

2.4.4 C

C je programovací jazyk, který byl původně vyvinut pro potřeby operačního systému Unix a to v roce 1972. Unix byl původně napsán v assembleru, ale kvůli lepší přenositelnosti bylo potřeba zvolit vyšší programovací jazyk. Fortran se ukázal pro tyto potřeby jako nepoužitelný a proto byl vytvořen jazyk B. Dalšími úpravami pak vznikl z jazyku B jazyk C, který byl výrazně rychlejší.

C bylo navrženo především pro praktické a efektivní programování. Nevyžaduje následnou runtime podporu, na rozdíl např. od Javy. Překládá se přímo do strojového kódu.

Jazyk C se těšil značné oblibě. Postupem doby se proto začaly objevovat jeho odnože. Tou první byl jazyk C++, uvedený v roce 1983. Ten není s původním C zcela kompatibilní. Od klasického C se lišil především objektovým přístupem, implementací generického a procedurálního programování. C++ je také mnohem striktnější v ohledu pravidel pro přetypování datových typů. Dalším odvozeným jazykem je Objective-C z roku 1986, který je zcela kompatibilní s původním C. Všechny tyto typy se stále používají. S ohledem na fyzikální využití je jazyk C oblíben a upřednostňován především v USA. Značné oblibě se celosvětově těší i v ostatních oblastech programování.

Nepřímým potomkem jazyku C je i C# z roku 2002. Ten byl vyvinut firmou Microsoft. Jedná se o objektově orientovaný jazyk založený na C++ a Javě. Cílem bylo především konkurovat Javě.

```
#include <stdio.h>

main()
{
  for(;;)
      {
          printf ("Hello World!\n");
      }
}
```

Obr. 2.7: „Hello world!" - C.

2.5 Počítačové jazyky 3,5. generace

Pod tímto označením se skrývají objektově orientované programovací jazyky (OOP). Nejde o změněnou syntaxi, ale o způsob programování. Snažíme se uplatnit principy z reálného světa při řešení problémů. Ty jsou reprezentovány objekty - prvky modelované reality reprezentované počítačovým kódem, které se následně sdružují do větších celků - tříd. Základem jsou tři pojmy - **zapouzdření**, **dědičnost** a **polymorfismus**.

Zapouzdření nám zaručuje konzistenci objektů. **Dědičnost** nám umožňuje organizovat objekty stromovým způsobem. Můžeme tak sdílet určité vlastností, díky tomu je kód přehlednější a kratší. V reálu tomu odpovídá např. stav, kdy máme různé druhy ptáků. Všechny druhy mají křídla, zobák, peří, něco konzumují atd., ale vzájemně liší se jednotlivými „parametry“ těchto vlastností. **Polymorfismus** pak umožňuje objektům volání metody s různou implementací (přístupem). Typickým příkladem využití polymorfismu je přepínač switch.

Prvním objektově orientovaným jazykem byl v roce 1969 Smalltalk. OOP jazyky dělíme na tzv. čisté a hybridní. Čisté OOP jazyky nepřipouštějí jiný programovací model, kromě Smalltalku k nim patří např. Ruby, Eiffel, Java nebo C#. U hybridních OOP jazyků máme na výběr, jakým způsobem přistoupíme k řešení problému. Patří k nim např. C++, Python, Object Pascal, Ada a další.

V počítačové fyzice se OOP jazyky často používají pro oblast analýzy obrazu vysoké úrovně a tam, kde máme vysoce specifické požadavky.

```
#include <iostream.h>

main()
{
    cout << "Hello World!" << endl;
    return 0;
}
```

Obr. 2.8: „Hello world!“ - C++. [27]

2.6 Počítačové jazyky 4. generace

Předchozí generace prog. jazyků i přes své postupné zjednodušení stále vyžadovaly značné znalosti v oblasti programování. V komerční i vědecké sféře ale vznikla potřeba programování i pro další profese, kde byly zásadní zcela jiné znalosti. V průběhu 80. let tedy vznikla 4. generace prog. jazyků, kde odpadla potřeba umět programovat, ale pouze se interaktivně vytváří požadovaný výsledek, který je často následně vizualizován.

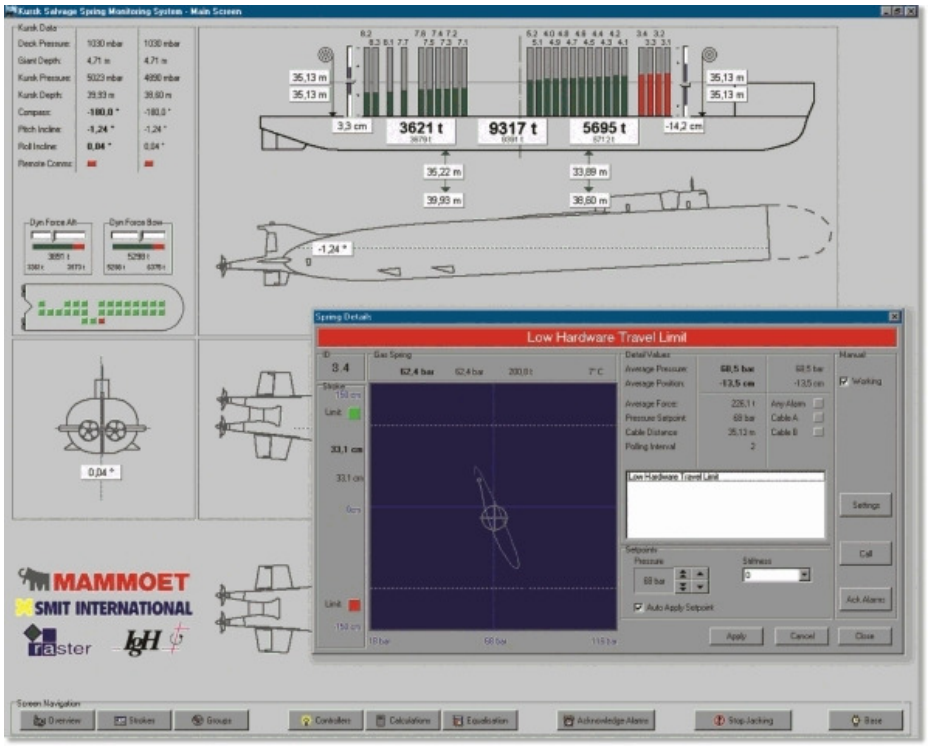
Programovací jazyky této generace jsou orientovány pro konkrétní účel, K těm nejznámějším patří HTML a odvozené jazyky, PHP, Microsoft Access, **Maple** či LabVIEW. Z hlediska počítačové fyziky jsou pro nás důležité jazyky určené přímo pro řešení fyzikálních problémů. Těmi jsou v současnosti především **MATLAB** a specializované systémy **FLUENT** či **COMSOL**.

2.6.1 MATLAB

MATLAB (MATrix LABoratory) byl uveden na trh v roce 1985 firmou The MathWorks. Původní systém byl vyvinutý pro přístup k matematickým knihovnám (EISPACK, LINPACK). V současnosti bývá označován za světový standard v oblasti vědeckotechnických výpočtů. Jde zároveň o program i jazyk v něm použitý. Vzhledem k prostředkům, které zahrnuje, jde asi o nejčastěji používaný programovací systém v počítačové fyzice.

Zadávání příkazů je interaktivní a program je tak přístupný téměř každému. Uživatel zadává příkazy a ty se následně podle definovaných podmínek provedou. Řada příkazů je ale dostupná přes grafické rozhraní, takže není potřeba si pamatovat všechny příkazy.

Výhodou je i propojení MATLABu s dalšími programovacími jazyky. Můžeme přímo použít objekty vytvořené v Javě, můžeme tedy i využívat její knihovny. Navíc lze použít i moduly psané ve Fortranu či C. S MATLABem lze připojit i řadu různých programů, např. u nových verzí Adobe Photoshopu lze zobrazovat přes příkazy v MATLABu grafiku apod.



Obr. 2.9: Aplikace pro řízení a monitorování záchranné operace potopené jaderné ponorky Kursk realizované v Matlabu. [28]

Silnou stránkou MATLABu jsou především toolboxy neboli knihovny funkcí. Ty rozšiřují program pro použití v příslušných vědních i technických oborech. Jejich výčet by byl velice dlouhý, spíše by byl problém najít obor, v němž by se MATLAB nedal využít. Můžeme sem zahrnout oblast fyziky, automatizace, finanční modelování, teorii signálů, měření atd. Tento systém byl využit i např. pro simulace k vyzvednutí potopené jaderné ponorky Kursk, operace pak proběhla úspěšně. Dalšími známými příklady byl řízený pád orbitální stanice Mir, který byl nejprve simulován právě v MATLABu, nebo návrh dvoukolového vozítka Segway.

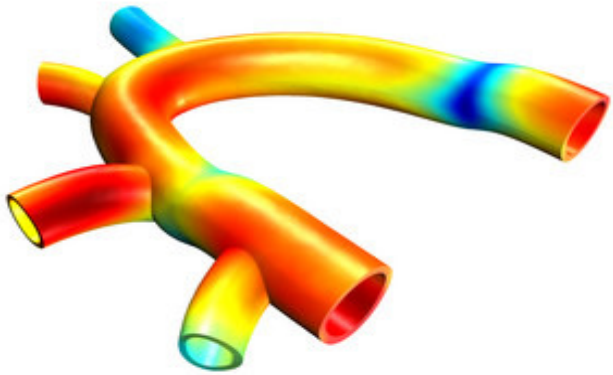
Díky tomu všemu není MATLAB zadarmo. Cena všech toolboxů a dalších doplňků do MATLABu vyjde na cca 7 000 000 Kč bez DPH. Několik toolboxů samotných převyšuje částku 1 000 000 Kč bez DPH. Existují i GNU alternativy jako GNU Octave nebo FreeMat, ovšem s omezenějšími možnostmi.

```
disp('Hello World!');
```

Obr. 2.10: „Hello world!" - Matlab.

2.6.2 COMSOL Multiphysics

Tento program se specializuje na simulaci a modelování fyzikálních procesů, které se dají popsat parciálními diferenciálními rovnicemi s následným řešením metodou konečných prvků. Do samotného řešení lze zahrnout i více fyzikálních vlivů.



Obr. 2.11: Model deformování cévy při průtoku krve u malého dítěte. [29]

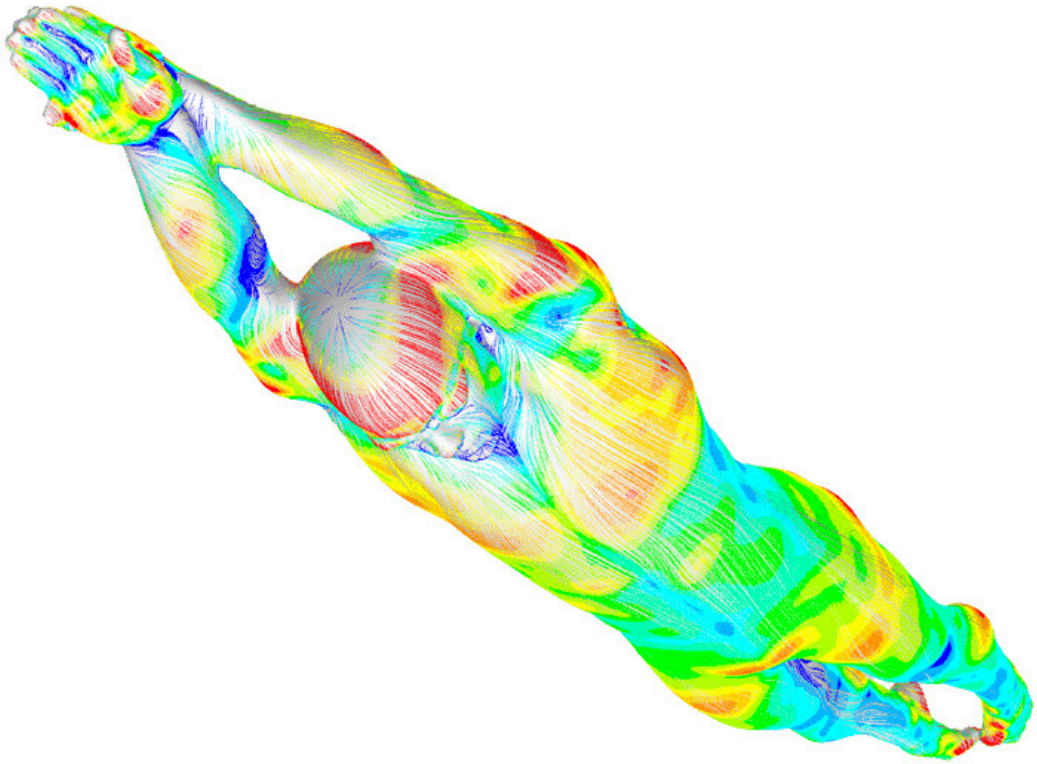
COMSOL nabízí výborné vyzualizační prostředky statických i dynamických dějů. K dispozici jsou opět moduly podle profesního zaměření.

Program nabízí výbornou kompatibilitu s MATLABem. Podobně jako MATLAB používá vlastní jazyk.

Je vhodný k řešení úloh z pružnosti a pevnosti, přestupu tepla, elektromagnetismu, dynamiky tekutin, průmyslové chemie, geologie, akustiky, a dalších. [30]

2.6.3 FLUENT

FLUENT je CFD (Computational Fluid Dynamics) komplexní systém pro numerické výpočty v oblasti dynamiky tekutin. Při výpočtu se využívá principu metody konečných objemů. Modelovat můžeme např. chování systémů při proudění, přenos tepla či hmoty, včetně chemických reakcí a spalování, případně jakékoli úlohy s pohybujícími se oblastmi.



Obr. 2.12: Simulace proudění vody ve FLUENTu - výsledkem projektu byly plavky Speedo, díky nimž padly téměř všechny plavecké rekordy na Olympijských hrách Peking 2008. [31]

Pro systém jsou vstupem geometrická data získaná např. z CAD systémů, materiálové vlastnosti, termodynamické vztahy a další veličiny a podmínky vedoucí k řešení. Při samotné práci můžeme využít kód z jiných programovacích jazyků, např. C++.

O kvalitě tohoto produktu hovoří i téměř 50% podíl na celosvětovém trhu s CFD softwarem.

2.7 Počítačové jazyky 5. generace

V předchozích generacích bylo nutné specifikovat posloupnost jednotlivých kroků, které mají být provedeny, abychom dosáhli požadovaného výsledku. Člověk má ale často má nějaký cíl a je mu celkem jedno, jak se k němu v rámci určitých mezích dojde. Podobným způsobem, a tedy i teoreticky blíže lidské mentalitě, fungují jazyky 5. generace. Programátor nadefinuje základní objekty, pravidla a omezení a popíše kritéria požadovaného řešení. Je pak na počítači, aby dosáhl co nejefektivnějšího způsobu k dosažení požadovaného řešení..

Počátky této generace jsou v roce 1979, kdy japonská vláda zahájila výzkum zaměřený na počítače páté generace. Cílem bylo „posunout společnost více žádoucím směrem“. Výsledkem měl být počítač (umělá inteligence), který by uměl mimo jiné i komunikovat s lidmi. Pilířem projektu bylo i použití odpovídajícího programovacího jazyku, tím byl Prolog. Počítač sice světlo světa nespatřil, přesto tento projekt obohatil svět informatiky o cenné poznatky a právě Prolog, zástupce logického programování. Název Prologu je akronymem z francouzského PROgrammation en LOGique. Jazyk byl vytvořen už v roce 1972, ale větší pozornosti se dočkal až díky zmíněnému japonskému projektu.

Další variantou této generace je funkcionální programování. Základem je princip rozepsání programu do rekurzivní funkce, ta opakovaně před svým dokončením volá sama sebe s novou sadou parametrů. Nejznámějším zástupcem je Lisp, který se využívá i v AudoCADu, dále např. Haskell, Miranda, ML či Scheme.

Největším omezením těchto jazyků je jejich výpočetní náročnost.

```
write('Hello world!'),nl.
```

Obr. 2.13: „Hello world!“ - Prolog. [27]

```
(defun helloworld ()
   (print "Hello World!")
 )
```

Obr. 2.14: „Hello world!“ - Lisp. [27]

3 Významné osobnosti počítačové fyziky

Zmínil jsem zde již vývoj části počítačové fyziky, která se věnovala výpočetní technice. Druhá a neméně obsáhlá část tohoto oboru se týká fyziky. Řada základních technik byla zavedena slavnými fyziky již před vynálezem funkčního počítače. Podstata počítačové fyziky ale není v použití počítačů. Ty jsou pro nás jen výkonným prostředkem pro použití numerických metod. Jejich nejdůležitější principy byly objeveny již ve středověku. Objeviteli a průkopníky byly často jedny z nejslavnějších osobností fyziky vůbec, často ale zasahovali i do dalších vědních disciplín.

3.1 Isaac Newton

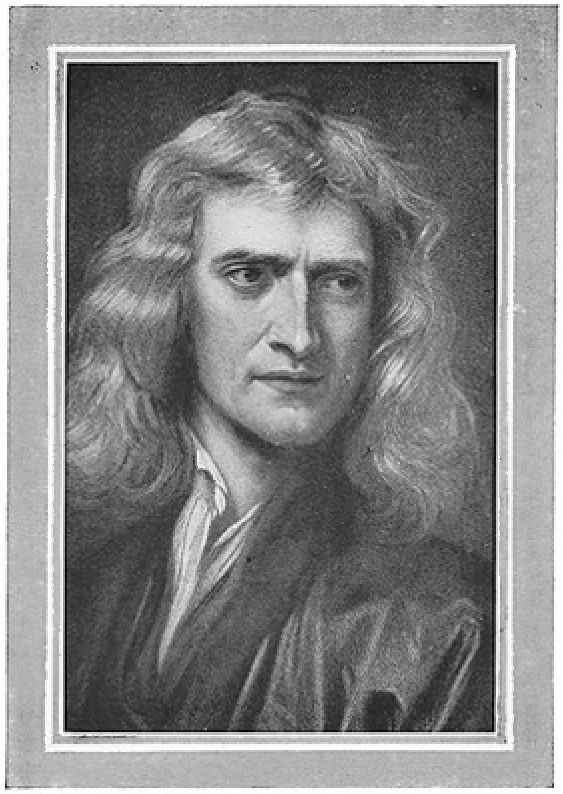
Tento slavný fyzik se narodil 4. ledna roku 1643 ve vesnici Woolsthorpe (asi 200 km severně od Londýna). Newtonův otec zemřel krátce před jeho narozením, takže se o něj starala pouze matka. Po třech letech se ale znovu vdala a Isaaca zanechala v Woolsthorpu. Předškolní věk přežil malý a neduživý Isaac jen s největšími obtížemi. Tesknil totiž po matce natolik, že se několikrát ocitl na pokraji duševního zhroucení.

Ve škole v ničem příliš nevynikal a byl často nemocný. Časem si našel zálibu ve vytváření modelů vodních mlýnků a hodin všech typů. Ty pak montoval se souhlasem majitelů na různá stavení. Zachovaly se i více než 100 let, protože byly neobyčejně přesné.

Zájem o mechaniku se v Newtonovi probudil na střední škole. V roce 1661 přestoupil na Cambridge na Trinity College, kde se ocitl pod vedením Isaaca Barrowa, známého fyzika a matematika. Jako chudý student to neměl lehké, byl v pozici subsizara, tedy v pozici žáka/sluhy. Vše se ale změnilo, když vešlo ve známost jeho **zobecnění binomické věty v řadu**, použitelné pro libovolné exponenty. Tím získal rázem vážnost na celé univerzitě a stal se scholarem, tedy získával stipendium a nemusel tak posluhovat bohatějším žákům.

V 22 letech získal hodnost bakaláře svobodných umění. V té době vypuká v Anglii morová nákaza, takže univerzity byly uzavřeny a učitelé i studenti se rozutekli na venkov, aby si zachránili život. O rok později objevil **derivace funkcí**. Pochopil důležitost integrálů jako „obrácených derivací“ a s pomocí těchto veličin dovedl analyticky zjišťovat **tečny křivek a křivosti čar**. Pro nebeskou mechaniku vytušil správný **tvar gravitačního zákona** i jeho důsledky pro astronomii a v optice si vytkl za cíl zlepšit astronomické dalekohledy. Při stavbě optických přístrojů objevil **chromatickou a otvorovou vadu** (aperturní aberaci). Zároveň poznal, že **tvarem ploch u čoček** lze ovlivnit a tedy zmenšit právě otvorovou vadu.

Během této doby, kdy pobýval na statku a objevoval, pochází i známá historka s jablkem. Jablko mu sice nespadlo na hlavu, ale při pozorování úplňku si Newton všiml jablka na stromě (pietně udržovaném do roku 1820) a uvědomil si, že dosah zemské přitažlivosti nemá omezení, takže síla udržující Měsíc na kruhové dráze kolem Země by mohla být totožná se silou působící pád jablka na Zem. Provedl tedy příslušný výpočet pro pohyb Měsíce kolem Země a došel k závěru, že obíhající těleso je přitahováno silou nepřímo úměrnou druhé mocnině vzdálenosti obou těles. [32]



Obr. 3.1: Portrét sira Isaaca Newtona od Godfreye Knellera (1689). [34]

V roce 1668 se stal magistrem umění a o rok později profesorem matematiky. Na universitě v Cambridge zůstal dalších 40 let. Díky svému revolučnímu dalekohledu s kulovým zrcadlem se stal členem Královské společnosti. Později založil emanační, **korpuskulární teorii světla**, podle níž je světlo jakýsi proud elementárních světelných částic lišících se podle barvy tvarem i velikostí. Zároveň byl ale odpůrcem vlnové teorie, i když ne totálním jako někteří další. Důkazem toho je, že vypočetl vlnovou délku světla.

Newton přednášel především optiku. Zachovala se zpráva, že Newtonovy přednášky byly nudné, studenti jim nerozuměli a nejednou se prý stalo, že Newton přišel do prázdné posluchárny. Přednášky z optiky detailně popisovaly experimenty

a přístroje, podstatnou část tvořily výpočty s dlouhými geometrickými důkazy. Experimenty se tehdy posluchačům nepředváděly a zůstalo jen mluvené slovo. [33]

Newtonovy zásluhy v mechanice a v teoretické fyzice se pojí s jeho největším dílem a snad nejvýznamnějším dílem v dějinách vůbec. Tím je jeho dílo: „**Matematické základy přírodní filozofie**“. Newtonova kniha podává soustavný a na svou dobu úplný systém dynamiky hmotných bodů, tuhých těles a tekutin, tj. kapalin a plynů. Veškeré dosavadní poznatky jsou v jejím rámci jen jednoduchými příklady, nad něž přináší Newtonovo dílo nepřehledně velké množství vlastních, úplně nových výsledků, a to vše na nové, vyšší úrovni a ve spojení se zcela novými matematickými ideami. Newtonův spis obsahuje v prvním svazku **mechaniku bodů** a **tuhého tělesa**, ve druhém **hydromechaniku** a ve třetím **mechanický a astronomický obraz kosmu**, bez vzorců – určen širším vrstvám.

Po vydání tohoto díla Newton ochabuje a začalo se projevovat jeho podivínství. Svým přátelům a známým osobnostem píše nesmyslné a urážlivé dopisy a vzápětí jejich obsah odvolává. Za svoje zásluhy byl královnou Annou jmenován rytířem, byl asi vůbec prvním vědcem, kterému se dostalo takového uznání.

Známým se stal i nesmiřitelný spor mezi Newtonem a Gottfriedem Leibnizem. Vedli jej o **prvenství objevu diferenciálního a integrálního počtu**. Angličané stranili Newtonovi, kontinentální Evropa zase Leibnizovi. Nakonec se ale projevily výhody Leibnizovy koncepce a symboliky, a proto na něj navázaly další osobnosti matematiky i fyziky.

Newton také často vydal svá díla veřejnosti až po mnoha letech, do té doby ležely ve stolu. Také než aby polemizoval se svými odpůrci, počkal raději, než umřeli. Jednou z těchto odkládaných knih byla **Optical Lectures** z roku 1671. Tu měl téměř dokončenou, ale po sporech s fyzikem Robertem Hookem se rozhodl do jeho smrti nepublikovat nic z optiky. Tato kniha vyšla až po Newtonově smrti v roce 1729. Stěžejní dílo **Optics** publikoval po Hookově smrti v roce 1704. [32]

Později byl jmenován velmistrem královské mincovny v Londýnském Toweru. Nebylo to kvůli jeho schopnostem, ale kvůli jeho neteři, která se zalíbila tehdejšímu ministru financí. Přesto se proslavil bojem proti penězokazům a zavedl definici nové měny guiney. Během svého života publikoval v prvním a zároveň nejslavnějším vědeckém časopisu světa, ve Philosophical Translations. Umírá 31. března 1727 v Londýně.

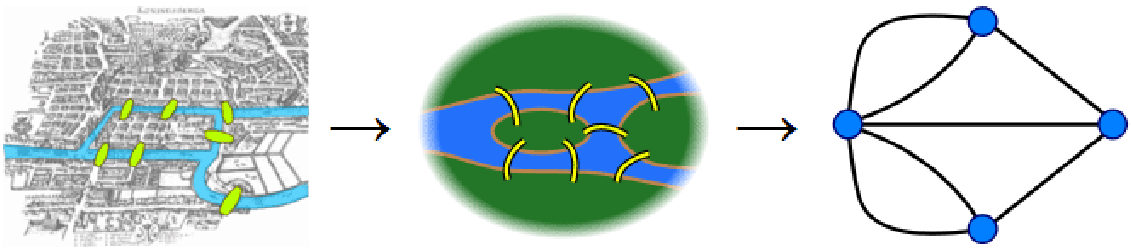
3.2 Leonhard Euler

Je považován za nejlepšího matematika 18. století a jednoho z nejlepších matematiků vůbec. Narodil se 15. dubna 1707 ve švýcarské Basileji v rodině kalvinistického duchovního Paula Eulera, který studoval u slavného Jakuba

Bernoulliho. Otec mu předurčil stejnou dráhu, ale také chtěl, aby syn měl všeobecné vzdělání. Matematika v rodině Eulerových nebyla ničím neznámým., často ji probírali s rodinným přítelem Jakubem Bernoullim.

Už v třinácti letech začal Leonhard studovat na basilejské univerzitě, kde navštěvoval přednášky Johanna Bernoulliho. Ten brzy poznal Leonhardovo nadání, a tak mu věnoval individuální konzultace. Vždy přes týden mu zadal k samostudiu komplikované matematické práce a v sobotu je s ním probíral. Učení mu šlo lehce, měl totiž fenomenální paměť a velmi dobře rozvinuté logické uvažování. V pouhých šestnácti letech latinsky přednášel na téma „Newtonova a Descartesova filozofie“, za což získal hodnost magistra umění. Univerzitní studia ukončil v devatenácti letech.

V roce 1727 odcestoval do Ruska, jazyku se naučil rychle. Jako matematik nemohl zastávat nabídnutý post fyziologa v oddělení pro medicínu, proto se zapsal na studia medicíny v Basileji. O rok později se stal vedoucím matematického oddělení v petrohradské Akademii věd. V Petrohradě se poprvé oženil. Vybral si podle sebe tu nejvhodnější z 11 předvedených nevěst, později s ní měl 13 dětí, 5 přežilo dětství. V roce 1735 oslepl na jedno oko. Prý se to stalo po 3 dnech a nocích usilovné práci na astronomických výpočtech, jiní matematici by na to potřebovali několik měsíců. V roce 1736 vyřešil úlohu, jak projít přes sedm mostů v Königsbergu a vrátit se do jednoho místa. Tím se stal zakladatelem **teorie grafů**.



Obr. 3.2: Slavná matematická úloha - Sedm mostů města Královce. Cílem bylo zjistit jak přejít všechny mosty jen jednou a vrátit se do výchozího místa. Euler dokázal, že to není možné, protože výsledný graf neodpovídá tzv. eulerovskému grafu. [51]

Kvůli narůstajícím nepokojům se přesunul v roce 1741 do Berlína. Strávil zde 25 let, během té doby vytvořil na 380 prací. Z této doby pochází jeho nejvýznamnější objev - **vlnová rovnice**. Díky použití Newtonovy mechaniky vyřešil parciální diferenciální rovnici popisující změnu tvaru houslové struny v čase i prostoru. O pár let později vyřešil rovnici i pro strunu s pevným okrajem - pro buben. Euler navíc uměl efektivně sdělit své poznatky i laické veřejnosti, jeho matematické práce byly vysoce ceněny a těšily se velké pozornosti.

Eulera doprovázely potíže s očima. Fridrich II. Veliký, pruský král, jej nazýval kyklopem. Později dostal šedý zákal i do druhého oka a téměř na něj

neviděl. Přesto se jeho produktivita spíše zvýšila - díky jeho neuvěřitelným matematickým schopnostem, dokonalé paměti a diktování zápisků písaři.

Díky příznivější situaci se vrátil v roce 177l do Ruska, kde strávil zbytek života. O rok později se úspěšně podrobil operaci šedého zákalu jednoho oka. Díky částečnému zraku opět intenzivně pracoval, až oslepl úplně. Konec života strávil v kruhu svých osmatřiceti vnoučat, která učil matematice a geometrii. Umřel na mozkovou mrtvici 18. září 1783 při hře s vnukem. [35]

Jeho dílo nemá v matematice obdoby. Vytvořil celkem 868 prací bez ohledu na svůj zdravotní stav. Jako první použil pojmu „**imaginární číslo**“, zavedl **dvojrozměrný integrál**, symboliku pro označení např. $f(x), \pi, \infty, \int, \sum$. Položil **základy mechaniky tuhých těles** a **hydrodynamiky**, např. zformuloval **diferenciální rovnici popisující pohyb ideální vazké tekutiny** a mnoho dalšího.

Obr. 3.3: Portrét Leonharda Eulera, jednoho z nejlepších matematiků. [36]

S odkazy na jeho dílo se setkáváme prakticky v celé matematice. Je mimochodem připomínán i v luteránském kalendáři svatých. Laplace o něm napsal: „Čtěte Eulera, čtěte Eulera, je to učitel nás všech.“

3.3 Erwin Shrödinger

Rakouský fyzik Erwin Rudolf Josef Alexander Schrödinger se narodil 12. srpna 1887 v Erdbergu. Nechodil do základní školy, protože měl soukromého vychovatele až do věku 10 let. Později nastoupil na gymnázium, kde měl nejradši matematiku a fyziku, ale nerad se učil cokoli nazpaměť, raději se snažil vše chápat. V 23 let získal doktorát s disertační prací na téma „O vedení elektřiny po povrchu izolantu ve vlhkém vzduchu“. Později byl povolán na frontu, kde sloužil u dělostřelectva.

Od roku 1921 působil v pozici profesora teoretické fyziky v Zürichu. Věnoval se teorii barevného vidění, struktuře atomu a kvantové statistice. Ve fyzikálním světě se tehdy řešily tři zásadní problémy: teorie relativity, atomové jádro a kvantová hypotéza. Trochu jasno do této problematiky vnesl de Broglie - s myšlenkou de Broglieho vln, doprovázející pohybující se částice. Schrödingerův nadřízený Peter Debye svému podřízeného požádal, aby se tímto tématem zabýval a vypracoval přednášku. Schrödingerovi se příliš nechtělo, ale Debye využil svého vlivu a sdělil mu, že přednášku udělat prostě musí.

Schrödinger předpokládal, že každou částici provází de Broglieho vlna, která může obíhat kolem jádra jen tehdy, je-li obvod dráhy celistvým násobkem vlnové délky. Tím bylo vysvětleno, proč nemůže elektron obíhat po libovolné dráze. Pro takovou vlnu musí platit nějaká vlnová rovnice, která bude relativistická a kvantová. Téměř na počkání rovnici vytvořil (dnes se nazývá Klein, Gordon, Fokova), avšak nedávala výsledky shodné s pozorováním spektra atomu vodíku. Formuloval i nerelativistickou kvantovou aproximaci (dnes se nazývá **Schrödingerova rovnice**), která vedla ke skvělým výsledkům.

Přednáška byla pečlivě vyslechnuta a pak ji ředitel ústavu zhodnotil: „Člověče, vždyť vy jste našel nový princip vlnové mechaniky“. Vše publikoval v roce 1926 v sérii článků uveřejněných v Annalen der Physik. Za tyto práce byl společně s Paulem Diracem v roce 1933 vyznamenán Nobelovou cenou za fyziku a ta mu pomohla získat místo v zahraniční v Magdalen College v Oxfordu. [37]

Schrödinger je mimo jiné i autorem známého myšlenkového experimentu z roku 1935 nazývaného „**Paradox Schrödingerovy kočky**“. Chtěl jím poukázat na rozdílnost kvantového a makroskopického světa na polomrtvé kočce. Kočka je zavřená v krabici a hrozí jí smrt, pokud se uvolní kyanovodík. To se stane, pokud se zaregistruje rozpad radioaktivního nuklidu. Podle principů kvantové mechaniky se ale nuklid, který není pozorován, nachází v superpozici. Existuje tedy zároveň jakoby v obou stavech - „rozpadlý“ i „nerozpadlý“. Z toho ale vyplývá, že na tomto jevu je závislá celá soustava a tedy i kočka je polomrtvá, dokud se sami nepřesvědčíme. Pokud to uděláme, v našem makroskopickém světě uvidíme vždy kočku živou nebo mrtvou, ne polomrtvou.

Obr. 3.4: Erwin Schrödinger v roce 1927. [38]

Mezi jeho další práce patří **matematická teorie barvy**, ve které objasnil zákony míchání barev a položil tak **základy kolorimetrie**. Zabýval se také **statistickou termodynamikou**, měrnou **tepelnou kapacitou látek** a **spektroskopií**. Mimo samotné fyziky se zabýval i historií a filozofií fyziky a vědy obecně. Jeho díla ovlivnila řadu vědců zabývajících se nejen fyzikou, ale i biologií. Umřel 4. ledna 1961 ve Vídni.

3.4 John Ludwig von Neumann

Narodil se 28. prosince 1903 jako János Neumann v rodině maďarského bankéře a právníka v Rakousku-Uhersku. Jeho rodina, židovského původu, byla velmi bohatá a žili v rodinném činžáku. Jeho genialita se projevovala už odmalička. V šesti letech mluvil se svým otcem plynně starořečtinou, ještě předtím se učil francouzštinu a němčinu. V osmi letech uměl z hlavy dělit osmiciferná čísla a znal zpaměti stránky telefonního seznamu. Když mu bylo 10 let, jeho otec koupil šlechtický titul, od té doby jim v rodině přibylo „von“. Od dvanácti let se mu soukromě věnoval nejlepší profesor matematiky v Budapešti. Malý János velice rád četl a po celý život si vše pamatoval díky svojí dokonalé paměti. Jeho otec dokonce koupil ve výprodeji i celou knihovnu nějakého panství a János všechny knihy přečetl.

V 17 letech publikoval za pomoci učitele matematiky svůj první vědecký článek. Na přání otce začal poté studovat chemické inženýrství. Studium pro něj bylo snadné, takže nedocházel na přednášky a ve volném čase udělal doktorskou práci z matematiky. V 22 letech přešel do Göttingenu v Německu, kde byla matematika na vysoké úrovni, jako nejmladší asistující profesor v historii. Zabýval se kvantovou teorií a teorií neuronových sítí. V roce 1928 se celosvětově proslavil jako spolutvůrce **matematické teorie her**, která se dodnes využívá v ekonomice, politice a dalších oblastech. V roce 1930 odjel na pozvání Princeton Univerzity do USA. O tři roky se spolu s Albertem Einsteinem stal zakládajícím členem nového oddělení matematiky v Institut for Advanced Stud. Princeton se stal světovým centrem matematiky a teoretické fyziky.

Později se dvakrát oženil, často ale navštěvoval noční podniky. Mimo to rád jezdil rychlými automobily a v jeho domě se často oslavovalo a pilo. Přesto ale ustavičně pracoval. V roce 1937 vydal článek o ekonomii, který se stal základem **moderní teorie růstu**.

Obr. 3.5: John Ludwig von Neumann. [39]

Neumanna přitahoval militarizmus. Během 2. světové války se zapojil do projektu Manhattan. Ten se zabýval výrobou jaderné zbraně. Ostatní vědci měli problémy s výpočtem množství explosivního materiálu. Neumann problém vyřešil téměř ihned **čočkou pro implozi** - bomba pak byla svržena na Nagasaki. Byla dvakrát účinnější, než předchozí bomba, svržená o tři dny dříve na Hirošimu. Klíčovou roli při simulacích štěpné reakce a později i vodíkové bomby sehrála **metoda Monte Carlo**, jejíž rozvoj je rovněž spojován s Neumannem.

Později se podílel právě na vývoji **vodíkové bomby** (tisíckrát ničivější než jaderná), která měla být podle něj preventivně použita proti Sovětskému svazu. Na výpočty této bomby již nestačil a použil počítač. Tyto stroje ho uchvátily a později se stal průkopníkem jejich nové, tzv. **von Neumannovy koncepce** (viz obrázek 1.11 na str. 16). To byl vlastně jen „vedlejší produkt“ při vývoji bomby. Zabýval se i binárním systém a zavedl **bit**, jako jednotku paměti počítače.

Mezi jeho objevy patří i teorie seberozmnožovacích **celulárních automatů**, které sloužily k časové idealizaci fyzikálních systémů v matematice, teorii systémů, teoretické biologii a dalších oblastech. Zajímal se i o **předpověď a řízení počasí**. Uvědomil si problém **globálního oteplování** kvůli emisím kysličníku uhličitého. Obával se, že začernění ledu a sněhu na pólech by se mohlo stát zkázonosnou zbraní. Vytvořil **matematická pravidla pro konstrukci robotů** se schopností sebezdokonalování a reprodukce, zformoval **kvantovou mechaniku**.

V roce 1955 onemocněl rakovinou a umřel 8. února 1957. Jeho záběr ve všech disciplínách byl neuvěřitelný, stejně jako Neumann sám. Za jeho života koloval mezi vědci tento vtip: „Matematikové dokazují, co se jim dokázat podaří; von Neumann dokazuje, co chce“.

4 Počítačová grafika

Termín „počítačová grafika“ byl poprvé použit v roce 1960 designérem firmy Boeing, Wiliamem Fetterem, když vytvořil trojrozměrný model lidského těla kvůli zefektivnění uspořádání kokpitů letadel. Z technického hlediska jde o obor informatiky, který je i jednou z důležitých oblastí počítačové fyziky.

Obrazová informace je pro nás velice důležitá, protože lidské smysly jsou přednostně orientovány právě na zrak, který nám dokáže zprostředkovat asi 90 % vjemů. Díky vizualizaci tak získáváme umělé objekty z tvarů a barev, které nám mnohem lépe pomáhají pochopit získaná data oproti nepřehledným tabulkám. Cílem počítačové grafiky je zobrazení numerických dat - ty jsme získali již nějakou předchozí činností (např. měřením, simulacemi či modelováním). V počítačové fyzice neustále pracujeme s velkým množstvím dat a složitými modely, je proto více než žádoucí tyto data vhodně prezentovat.

Při práci s počítačovou grafikou je pro nás základem odpovídající hardware, software a především znalost související problematiky.

4.1 Grafický hardware

K počítači je nutné připojit takové výstupní zařízení, které nám bude vhodně interpretovat signál přes grafickou kartu z našich dat do výsledného obrazu. Taková zařízení můžeme rozdělit na více druhů:

1) Zařízení poskytující obraz v trvalé formě
   - tiskárny, zapisovače a plottery
2) Zařízení poskytující výstup zobrazovací jednotkou
   a) alfanumerické - zobrazují pouze text
   b) vektorové - obraz složen z vektorů - plynulé úsečky
   c) rastrové - obraz složen z pixelů (CRT, LED, OLED, …)
   d) speciální zařízení - 3D, holograf. zařízení

4.1.1 Zařízení poskytující obraz v trvalé formě

Zapisovače jsou přístroje určené k zápisu různých procesních nebo elektrických signálů v závislosti na čase. Popisovaným médiem bývá většinou papír. Ten se pohybuje pod perem, které zapisuje data. Výsledkem je pak graf. Mohou být jednokanálové nebo vícekanálové (více per). Důležitým parametrem je rychlost posuvu. Setkáme se s nimi spíše v profesně zaměřeném průmyslu. Nejznámější příklad je asi záznam seismografu nebo EKG.

Klasické perové **plottery** vykreslují obraz pomocí tužky nebo pera. Pro velkoplošný tisk se často používají inkoustové nebo laserové plottery, které jsou schopny tisknout vektorovou i rastrovou grafiku. Další variantou jsou řezací plottery s nožem místo pera. Původně se plottery používaly pouze na technické výkresy z CAD aplikací.

Tiskárny jsou v současnosti běžná zařízení. Je třeba se zaměřit spíše na technologii a parametry. Pokud požadujeme přesné barvy, je potřeba provést stejně jako u plotterů kalibraci barev pomocí kalibrační sondy a odpovídajících terčů. Kvalita výstupu je dána parametrem dpi (dot per inch). Jde o množství obrazových bodů na jeden palec. Pokud volíme větší dpi, je potřeba zvolit odpovídající gramáž papíru.

4.1.2 Zařízení poskytující výstup zobrazovací jednotkou

Tyto zařízení nám poskytují měnící se obrazový výstup v závislosti na použité technologii. **Alfanumerické displeje** zobrazují pouze alfanumerické znaky, pro naše účely s poněkud pokročilejší grafikou jsou tedy krajně nevhodné. Setkáme se s nimi většinou v jednodušších zařízeních nebo v průmyslu.

Vektorové displeje jsou založeny na vykreslení paprsku mezi počátečním a koncovým bodem. Jejich předchůdcem byly paměťové obrazovky, které se dosud používají u osciloskopů. Vývoj výpočetní techniky se ale ubíral spíše vykreslováním textu a později barevné grafiky, kde oproti bitmapě není vektorové zobrazení příliš efektivní. Určitou dobu se používaly ve výherních automatech. V roce 1982 byla dokonce vytvořena i jediná herní konzole Vectrex s vektorovým displejem, nebyla ale komerčně úspěšná. S vektorovými displeji se běžně už asi vůbec nesetkáme, používají se pouze ve specializovaných případech.

Obr. 4.1: Jediná herní konzole s vektorovým displejem - Vectrex (1982). [40]

Rastrové displeje jsou v současnosti nejrozšířenější, bez ohledu na použitou technologii. Jejich zobrazovací plochu označujeme jako rastr, jde o pravidelnou mřížku složenou z obrazových bodů (pixelů – z angl. picture element). Základním parametrem je tedy rozlišení (rozměr rastru).

Nevýhodou oproti vektorovým displejům jsou velké nároky na kapacitu paměti pro uložení rastru (framebuffer), která musí být i dostatečně rychlá kvůli obnově obrazu. Obrovskou výhodou je ale zobrazení prakticky libovolně složité grafiky a snazší programová tvorba.

Z technologického hlediska patří k rastrovým displejům několik typů: CRT monitory, LCD displeje, plasmové displeje, elektroluminiscenční zobrazovače (ve speciálních aplikacích) a kromě klasických displejů i dataprojektory.

Pro počítačovou grafiku se kvůli postranním lištám v aplikacích častěji používají širokoúhlé displeje např. s poměrem 16:10, případně sestava dvou displejů většinou s poměrem stran 4:3. Pro profesionální práci s bitmapovou grafikou se používají displeje se širším rozsahem gamutu, kalibrace by měla být samozřejmostí. Notebooky nejsou pro takovou činnost příliš vhodné, protože mají zpravidla pouze 6 bitové displeje, oplývají tedy značně omezeným gamutem a v závislosti na úhlu pohledu se často mění kontrast a světlost. Ani osvětlení pak zpravidla neodpovídá požadované intenzitě a teplotě světla, jinak stanovené pro naše podmínky na 5184 K. Pro účely počítačové fyziky máme ale zpravidla výrazně menší nároky. Nutný je spíše výkon hardwaru pro složité výpočty, simulace a modelování.

Formát	Rozměr v pixelech	Celkový počet pixelů	Poměr stran
QVGA	320 × 240	76 800	4 : 3
VGA	640 × 480	307 200	4 : 3
SVGA	800 × 600	480 000	4 : 3
XGA	1024 × 768	786 432	4 : 3
SXGA	1280 × 1024	1 310 720	5 : 4
SXGA+	1400 × 1050	1 470 000	4 : 3
WXGA+	1440 × 900	1 296 000	8 : 5
UXGA	1600 × 1200	1 920 000	4 : 3
QXGA	2048 × 1536	3 145 728	4 : 3
...			
WHUXGA	7680 × 4800	36 864 000	16:10

Tab. 4.1.: Přehledová tabulka nejčastěji používaných rozlišení.

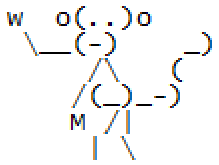
Budoucností snad blízkou jsou takové prozatím **speciální přístroje**, které budou simulovat realitu, tak jak jí vnímáme my, tedy 3D. Jejich vývoj je ale značně složitý a drahý, zatím se s nimi běžně nesetkáme.

4.2 Grafický software

Prvním programem, který využíval grafických možností počítače, byl vektorový Sketchpad z roku 1963 od známého průkopníka počítačové grafiky Ivana Sutherlanda. Šlo v podstatě o předchůdce dnešních CAD systémů s využitím světelného pera.

Během 80. let ale vektorové terminály postupně zanikly s nástupem grafických karet. Začaly se tak uplatňovat systémy pracující s bitmapou. Specializované systémy dokázaly už tehdy pracovat s jednoduchou grafikou, rozhodně ale nebyly dostupné pro běžného uživatele.

Běžně dostupné systémy práci v plnohodnotném grafickém režimu neumožňovaly. Využívalo se tedy tzv. **pseudografiky**, kde se využívalo méně užívaných alfanumerických znaků z ASCII tabulky (znaky 0 až 127). Dodnes se zachovalo používání tzv. emotikon. Tento způsob využíval např. Fortran, Pascal. [41]



Obr. 4.2: Ukázka pseudografiky.

Semigrafikou pak označujeme využití dalších znaků, které rozšířily původní ASCII tabulku, tedy znaky 128 až 255. Mezi pozdější rozšiřující znaky patří i česká diakritika či další znaky patřící do různých abeced, to už však záleží na konkrétním kódování. Semigrafické symboly oživily tehdy chudé grafické možnosti, patří k nim např.: ▒▓█ (stínování), ┌ ┐ └ ┘ ┼ ╬ (různé typy rámečků) atd.

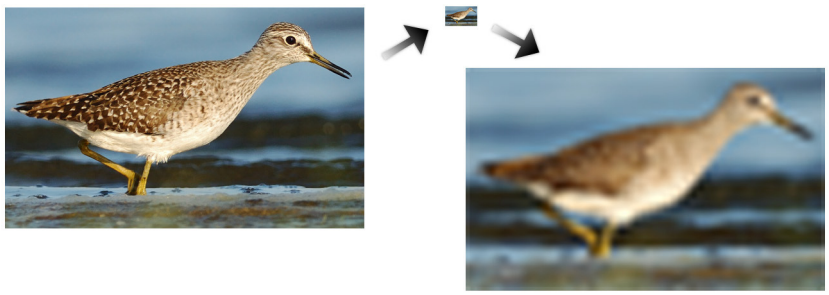
Později se využívalo ještě **spritů**, jednoduchého vykreslování rastrových obrázků na pozadí. Používaly se zhruba do roku 1996. Spritů se často využívalo u počítačových her - např. Doom a Wolfstein. Mimo her se běžně používaly v CAD systémech a matematických programech. V současnosti se s nimi můžeme setkat v teletextu či jednodušší počítačové grafice, počítá se s nimi i v kaskádových stylech CSS3.

Po širším použití grafických karet se s těmito způsoby zobrazování pro naše účely už tak často nesetkáme. V současnosti nám programovací jazyky nabízejí široké možnosti zobrazení ať už formou jednoduchého příkazu pro zobrazení obyčejného bodu, změny barvy apod. Na druhé straně máme k dispozici i silné programové prostředky pro vytváření kvalitních 2D i 3D grafických výstupů (MATLAB, COMSOL, FLUENT, Origin). [41]

4.2.1 Rastrová grafika

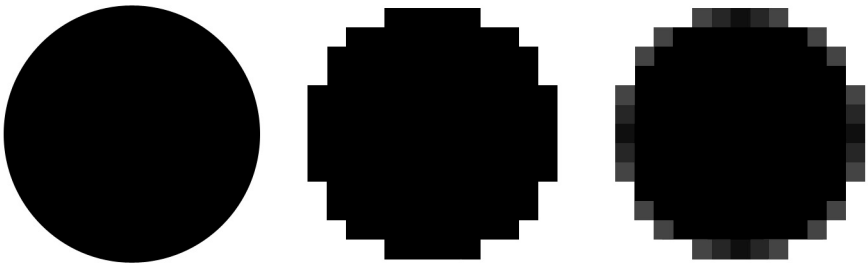
Jedním ze dvou způsobů uložení grafické informace je pomocí rastru (bitmapy). I při použití vektorů jsme ale ve většině případů nuceni kvůli technologii hardwaru pracovat s rastrem.

Všechny obrazové elementy rastru se skládají z bodů (pixelů). Obecnou výhodou je možnost snadné manipulace s libovolnou částí obrazu, větší než jeden pixel, což vede k ztrátě dat vůči originálu. Díky zobrazení v mřížce ale vzniká celá řada problémů. Kromě vyšších nároků na místo v paměti nemůžeme měnit velikost obrazu beze ztráty informace (vede ke zhoršení kvality). Zároveň vznikají problémy s ukončením křivek a celkovou „kostrbatostí“ nerovných částí obrazu.



Obr. 4.3: Zpětná změna velikosti bitmapy - ze šířky 560px na 50px a zpět na 560px.

Na obrázku 4.3 si můžeme všimnout výrazné ztráty detailů v obraze při zpětné změně velikosti. Zmenšením snímku došlo k redukci obrazové informaci a zpětně resamplovaný snímek se tak jeví jako rozmazaný. Z menšího rastrového obrázku nikdy nedostaneme stejně kvalitní snímek, jakým byl nezmenšený originál. Můžeme tak říct, že entropie systému (míra neuspořádanosti) vzrostla.



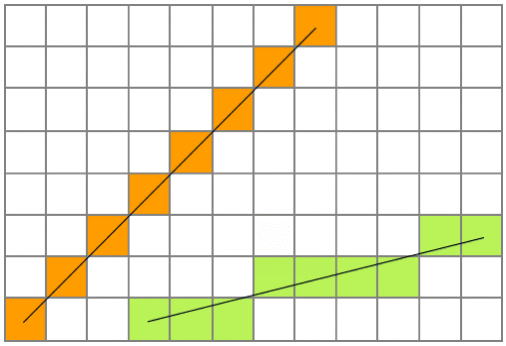
Obr. 4.4: První kolečko je náš ideál (vektor). Prostřední je rasterizované, pro názornost byl použit hrubý rastr. Poslední kolečko využívá optického klamu, vyhlazením se redukuje efekt hran.

Na obrázku 4.4 je s přehnaným efektem zobrazen efekt rasterizace původního ideálního vektorového kruhu. Míru tohoto efektu můžeme zmírnit dvěma způsoby. První možnost představuje zjemnění rastru, tedy zvýšení počtu pixelů ve stejné ploše. Druhý způsob využívá nedokonalosti lidského zraku.

Zmenšuje se kontrast bodu na rozhraní elementu vůči okolí v poměru se sousedními pixely. Využívá se v rámci možností obou způsobů. Efektu rasterizace je možné si všimnout např. při porovnání nekvalitního CRT a kvalitního LCD monitoru u drobných objektů (typicky písma).

Kreslení úsečky v rastru

Zatímco u fotografie máme předlohu předem danou, samotné vytváření grafiky není až tak jednoduché, aniž bych nějak zlehčoval proces fotografování. To si můžeme ukázat už na nejjednodušším prvku složeném z pixelů, kterým je úsečka. Při úhlech 0°, ±45°a ±90° uvidíme pravidelné rozložení pixelů. Při jiných úhlech nám ovšem vzniknou nepravidelné schody. Míra „schodovitosti“ pak záleží na sklonu úsečky a použitém algoritmu pro vykreslení.



Obr. 4.5: DDA algoritmus pro kreslení úsečky.

DDA (Digital Differential Analyzer) algoritmus je založen na postupném přírůstku k oběma souřadnicím. Na jeho principu byl mimo jiné vytvořen i engine počítačové hry Wolfstein 3D.

Pro názornost si zde uvedeme princip tohoto algoritmu.

Vycházíme z rovnice pro úsečku mezi počátečním (x_1,y_1) a koncovým bodem (x_2,y_2):

$$y = kx + q \tag{1}$$

Přírůstky v osách:

$$\Delta x = x_2 - x_1 \quad a \quad \Delta y = y_2 - y_1 \tag{2}$$

Směrnice je pak dána vztahem:

$$k = \frac{\Delta y}{\Delta x} \tag{3}$$

Dosazením dostaneme posunutí po ose y:

$$q = \frac{x_2 y_1 - x_1 y_2}{x_2 - x_1} \tag{4}$$

Proměnnou x budeme postupně inkrementovat a vypočítáme si hodnotu proměnné y. Tak budeme postupovat až do chvíle, kdy $x = x_2$.

První problém, na který pak asi narazíme, je vynechávání bodů úsečky. To se bude týkat těch případů, kdy $|k > 1|$. Jinak řečeno, pro tento případ nám podle předchozích vzorců vychází příliš málo bodů. Řešení je jednoduché, prostě obrátíme proměnné - dosazovat budeme proměnnou y a počítat budeme hodnoty x.

Výpočet souřadnice pro $|k > 1|$*, jinak řečeno když* $|\Delta y| > |\Delta x|$*:*

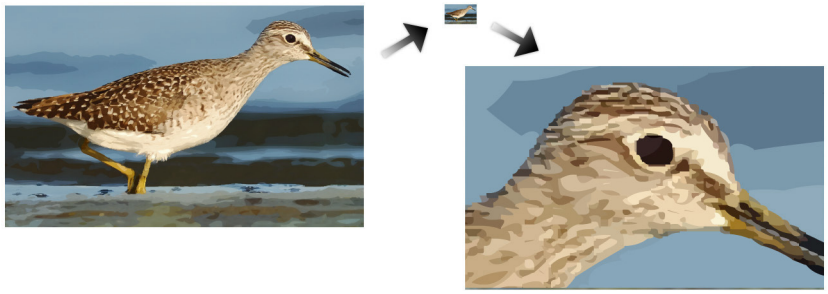
$$x = \frac{y - q}{k} \tag{5}$$

Základní nevýhodou algoritmu je práce v oboru reálných čísel - to je v případě celého počtu pixelů v rastru pro nás naprosto zbytečné. S tímto faktem pak souvisí i relativní pomalost algoritmu. Na druhou stranu je velice jednoduchý a snadno se implementuje. Existuje ale více variant, které jsou i efektivnější. Mimo úsečku můžeme vykreslit samozřejmě i další prvky.

V praxi se častěji používá např. Bresenhamův algoritmus, který používá celočíselnou aritmetiku, což zjednodušuje výpočet, a je i výrazně rychlejší.

4.2.2 Vektorová grafika

Představuje druhý základní způsob uložení obrazové informace. Jejím základem jsou jednoduché geometrické objekty (body, křivky a polygony), z kterých se celý obraz skládá. Všechny tyto objekty jsou definovány matematicky, lze pak pracovat i odděleně s každým z nich.



Obr. 4.6: Změna velikosti vektorové grafiky s výřezem - beze ztráty kvality.

Z toho všeho vyplývají základní výhody i nevýhody. Obraz nevyniká takovým množstvím detailů jako u rastrové grafiky, je vizuálně plošší. Celkově

však většinou zabírá výrazně méně místa v paměti. Pokud můžeme libovolně zvětšovat bez újmy na kvalitě dílčí objekty, z kterých se obraz skládá, můžeme to samé udělat i s celým obrazem. To vše je neocenitelné u celé řady případů - od nenáročné 2D grafiky třeba až k 3D modelování povrchů (např. terénu u map) s následným renderováním či vizualizací různých objektů či přímo procesů při modelování.

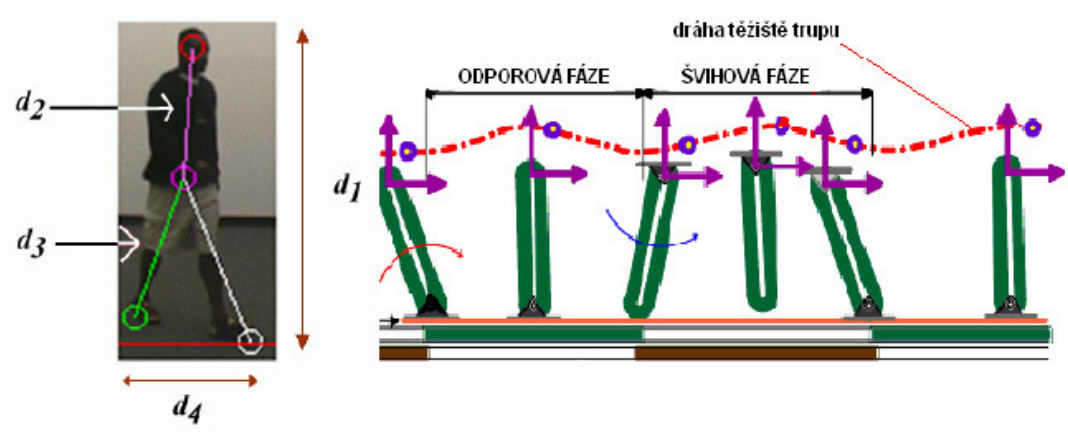
5 Zpracování obrazu

Obraz často vnímáme jako prostředek k vizualizaci dat. V řadě oborů pro nás ale naopak představuje zdroj informací, které bychom jinak jen těžko získali.

Klasickým příkladem může být astronomie. Pomocí pozemských dalekohledů a satelitů na oběžné dráze snímáme části vesmíru a doufáme, že něco objevíme. Množství takto získaných dat je ale natolik obrovské, že není v lidských možnostech bádat pečlivě naším zrakem nad každým snímkem, mnohdy by ani náš zrak oproti technice nestačil. Filtrováním a dalšími fázemi zpracování obrazu jsme schopni efektivně vyselektovat užitečné informace .

Jinde je zapotřebí identifikovat vozidlo podle poznávací značky. Kamera tedy pořídí snímek. Pomocí transformace obrazu, filtrů a dalších metod zpracování obrazu se mohou znaky převést do elektronické podoby a posléze identifikovat vozidlo v příslušné databázi. Obdobné metody (zne)užívají i roboti při prolomení CAPTCHA ochrany, která se často používá u formulářů v ochraně před spamem.

Jiným příkladem je jednoznačná identifikace osoby - např. podle stylu chůze, která je charakteristická pro každého z nás, rysů tváře, otisků prstů apod. Z původního obrazu se vyselektují znaky jednoznačně určující daného jedince a převedou se vhodně do databáze.

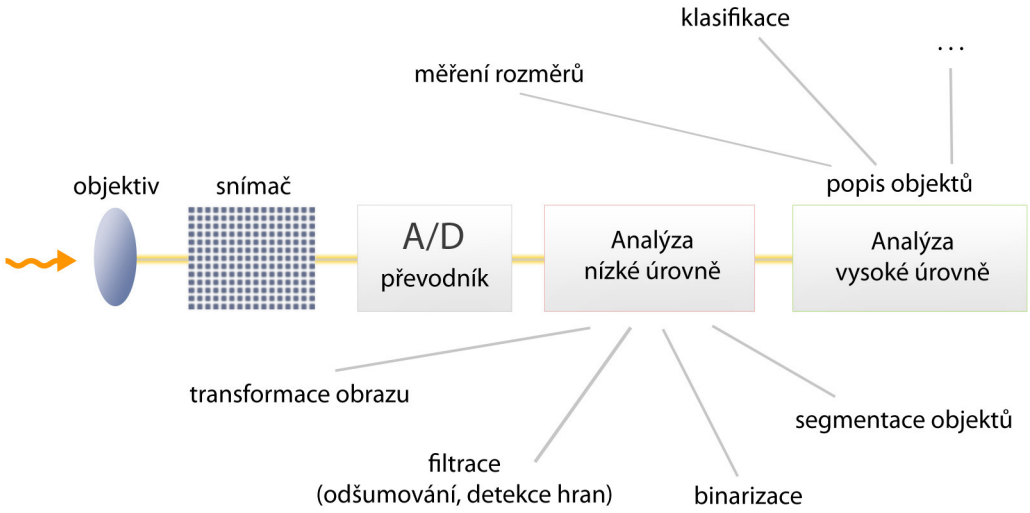


Obr. 5.1: Biometrická identifikace osob podle chůze na základě vytváření dráhy těžiště trupu. [42]

V řadě úloh budeme často využívat stejné algoritmy, i když u nich určitě změníme parametry. Takové postupy zahrnujeme většinou do tzv. **analýzy nízké úrovně**. V této fázi tedy zpracováváme obraz pomocí stejných prostředků, které jsou obsaženy v celé řadě dostupných aplikací. Obejdeme se tedy i bez hlubších znalosti programování.

V některých případech mohou být ale naše požadavky natolik specifické, že nám nezbude nic jiného, než že si naprogramovat vlastní prostředky. Týká se to především těch úloh, kdy se snažíme porozumět samotnému obsahu obrazu. Tento

proces nazýváme **analýzou vysoké úrovně**. Týká se to většinou složitých úloh, proto se snažíme úlohu v rámci možností ještě co nejvíce zjednodušit.



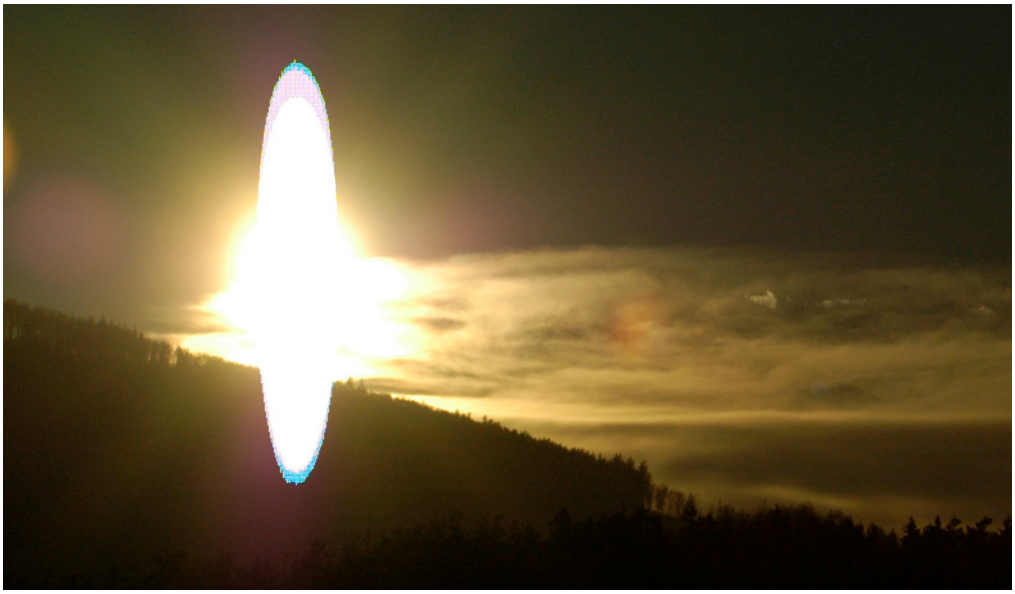
Obr. 5.2: Řetězec snímání a zpracování obrazu.

5.1 Snímání obrazu

Snímání obrazu je prvním procesem pro následnou práci s obrazem. Jde v podstatě o formu měření. Měříme tedy nějaký analogový signál, který převádíme na elektrický. Nemusí přitom ale jít vždy o optický signál. Pokud se trochu posuneme ve frekvenčním spektru, snímat můžeme třeba i radiové vlny, rentgenové nebo tepelné záření.

Pokud zvolíme vhodný způsob, můžeme si výrazně ulehčit další práci, případně získat kvalitnější data. Příkladem může být snímání sluneční koróny při zatmění slunce. Digitální čipy mají příliš nízký dynamický rozsah a dojde tak k přetečení A/D převodníků, kontrast je v tomto případě skutečně extrémní. Ve snímku se pak objeví plochy s přepaly, tedy místa bez informace. Nejlepší volbou bude v tomto případě barevný negativ, který pak naskenujeme. Neúspěšný pokus fotografování s extrémním kontrastem je uveden na následujícím obrázku 5.3.

Výsledek našeho snažení můžeme obecně ovlivnit např. kvalitním osvětlením scény, difuzéry, filtry korigujícími vliv barevných vad, polarizačními filtry pro redukci odlesků apod. U každého snímání jsou do jisté míry zastoupeny chyby objektivů a šum. Např. při snímání optického signálu přímo digitálním snímačem můžeme míru šumu ovlivnit osvětlením scény, velikostí buněk snímače a i teplotou, při které snímáme.



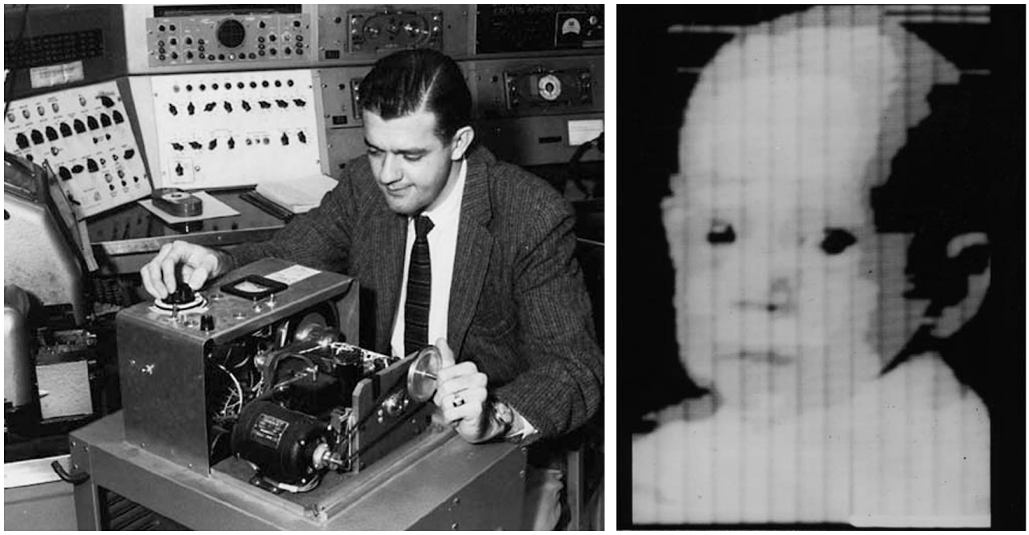
Obr. 5.3: Pro podobné snímky není CCD snímač vhodnou volbou. Zde došlo k přetečení náboje do sousedních pixelů, tento efekt nazýváme blooming. Vzniklý útvar postrádá informace, obsahuje pouze čistě bílou barvu.

5.2 Digitalizace

Abychom mohli s obrazem dále pracovat v počítači, je potřeba jej digitalizovat. Převádíme tak spojitý elektrický signál do binární podoby, výsledkem je matice bodů. Při vzorkování je důležité vzít v potaz velikost nejmenších detailů, které chceme analyzovat. Vzorkovací frekvence by měla být minimálně dvojnásobná oproti nejvyšší frekvenci obsažené ve vzorkovaném signálu, to nazýváme jako Nyquistův teorém.

Vzorkování si můžeme představit jako „rozsekání“ spojitého intervalu na intervaly, kterým přiřadíme hodnoty bitů. Pokud budeme pracovat jen s malým množstvím kvantizačních úrovní (intervalů), dostaneme značně zkreslený signál. Pokud ale zvolíme příliš velkou přesnost, tedy velké množství kvantizačních úrovní, výsledkem může být zbytečně velká zátěž na paměť a procesor počítače, zvlášť při zpracování obrazu v reálném čase.

Vytvořenou matici popisujeme obrazovou funkcí *f*. Ta má běžně dva parametry *f(x, y)*, jde tedy dvourozměrný obraz. V některých případech je zapotřebí více parametrů, např. *f(x, y, z)* pro třírozměrný prostor nebo třeba *f(x, y, t)* pro snímání dvourozměrného obrazu v čase atd.



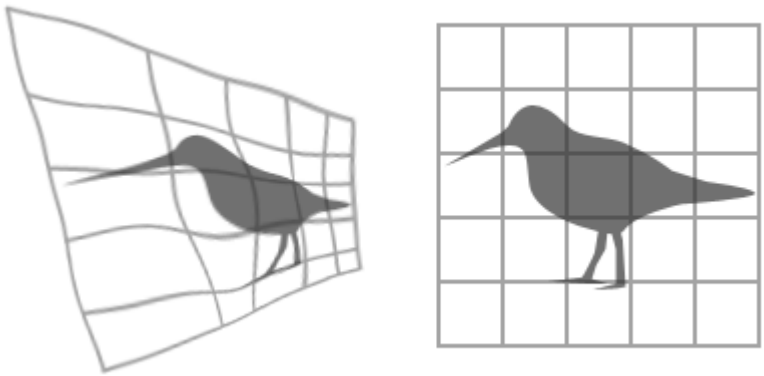
Obr. 5.4: Vlevo je první zařízení pro digitalizaci obrazu - bubnový skener. Vpravo pak první digitalizovaná fotografie (z roku 1957). Přes svojí technickou nedokonalost se právem dostala mezi 100 fotografií, které změnily svět (Life magazine). [43]

5.3 Analýza nízké úrovně

Představuje kvalitativní analýzu vstupních dat, je nutná pro vyšší úroveň zpracování obrazu. Většinu požadavků splní řada volně dostupných i komerčních programů, obsahující nejčastěji používané algoritmy pro zpracování obrazu.

5.3.1 Geometrické transformace

Snímání obrazu neprobíhá vždy za optimálních podmínek, často se proto setkáme s deformacemi. Typicky se jedná o případy, kdy není obraz v rovině s obrazovým snímačem, nebo když je část obrazu výrazně blíž snímači oproti zbytku. V řadě případů jsme schopni eliminovat zkreslení už při samotném snímání, to ale nejde vždy - např. při snímání zakřiveného povrchu Země družicí.



Obr. 5.5: Ilustrace geometrické transformace.

Podobné případy řešíme geometrickou transformací. Její princip spočívá ve vynásobení matice prvků transformační funkcí, tu lze vypočítat či empiricky odvodit. Pro vyhledání této funkce se často využívá i několika známých bodů, které lze přiřadit ke stejnému objektu ve zdrojovém i výstupním obrazu. Dalším způsobem je využití přímo testovací pravoúhlé mřížky, podle míry jejího zakřivení pak sestavíme transformační funkci pro výsledný obraz.

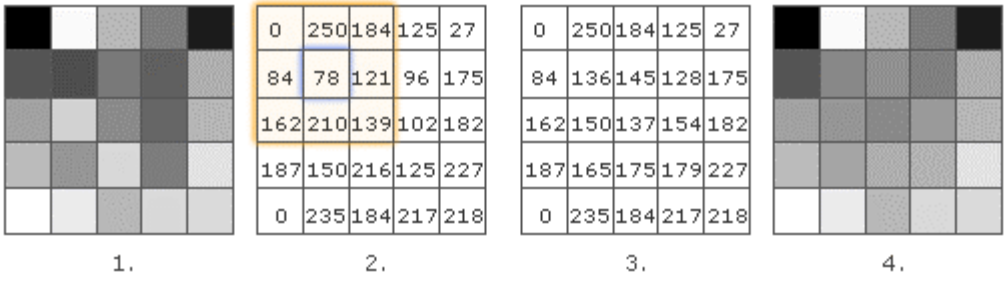
5.3.2 Filtrace

Při zpracování obrazu většinou potřebujeme pracovat jen s jeho určitou částí. Úkolem filtrace je tedy zvýraznění nebo naopak potlačení určitých informací v obrazu. Tím ale přicházíme o možnost vrátit obraz do identicky původního stavu, proto je potřeba dobře zvážit míru filtrace, vždy přijdeme o nějaké informace. Nejčastějším cílem je redukce šumu, případně vyhlazení obrazu, detekce hran, zvýraznění kontrastu atd.

Filtr je definován jako matice pixelů s lichým počtem řádků a sloupců, kterou nazýváme maska nebo kernel. Její rozměr je většinou 2×2, 3×3 nebo 5×5 pixelů. Maska postupně projíždí matici originálních dat a statisticky postupně hodnotí pixely, které zabírá. Podle toho pak určí novou hodnotu vybraného pixelu. Důležité je, že se neustále projíždí originální matice, vyhodnocené pixely se ukládají do nové matice.

Pokud budeme upravovat např. pixel uprostřed matice 3×3, dočkáme se toho, že nebudou filtrovány krajní body originální matice. Proto se užívají i asymetrické rotující masky, u kterých se určuje hodnota pixelu např. v rohu. Na každém místě se pak otočí dokola a průměrováním všech těchto stavů se určí nové hodnoty.

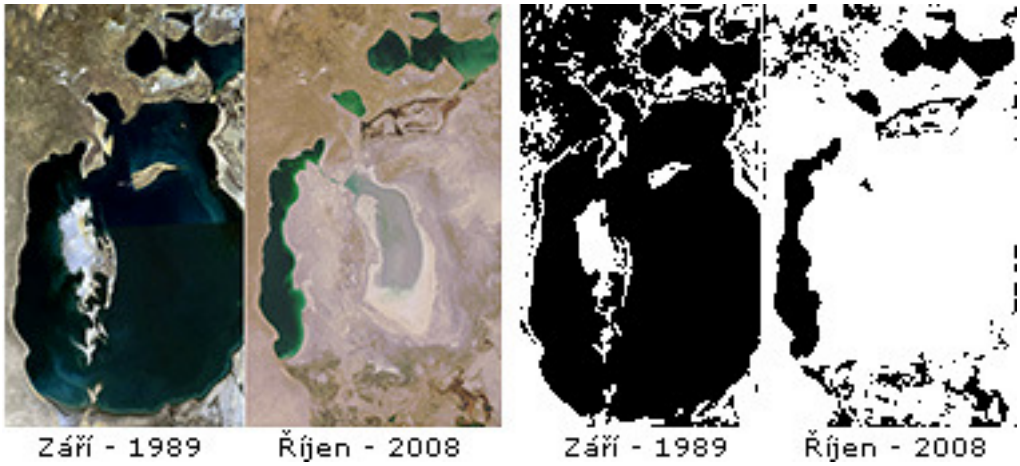
Jak jsem psal výše, filtrace je sice nesmírně užitečným ale i nevratným procesem vedoucím k degradaci dat, proto bychom měli vždy pracovat s kopií originálu. Podobné zásady bychom se měli řídit i při ostatních operacích s daty.



Obr. 5.6: Ukázka filtrace maskou 3×3 prostým průměrováním pixelu uprostřed.
1. Původní obraz, pro zjednodušení jsem použil šedou paletu.
2. Původní obraz - zobrazení barev číselným kódem.
3. Filtrovaný obraz - zobrazení barev číselným kódem.
4. Filtrovaný obraz - obraz je hladší, ale jednoduchý typ masky nezasáhl okraje obrazu.

5.3.3 Binarizace (prahování)

Binarizace představuje nejstarší, nejjednodušší, nejčastěji používaný a zároveň nejrychlejší způsob segmentace obrazu – lze jej používat i v reálném čase. K dispozici máme většinou obraz barevný nebo v odstínech šedé. Naším cílem je převedení takové škály odstínů pouze do dvou barev - černé a bílé, abychom mohli s obrazem dále pracovat. Základní myšlenka je tedy velice jednoduchá, praxe ale může být o něco složitější.



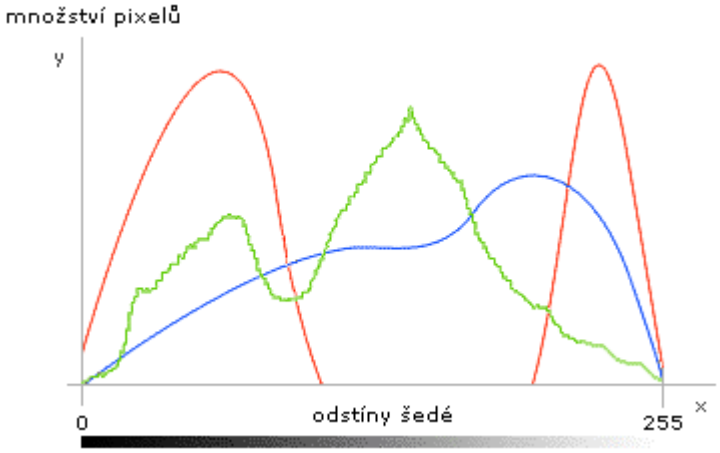
Obr. 5.7: Vysychání Aralského jezera. Horní plochy byly příliš tmavé, takže je potřeba je ručně odstranit – což lze snadno, protože jsou oddělené od okolí. Díky binarizaci pak můžeme v tomto případě např. rychle určit úbytek vody v ploše. [44]

Problémem je, jak zvolit práh, od kterého bude vše černé či bílé, tedy co bude ještě spadat do pozadí či objektů. Pomocným nástrojem je pro nás histogram. Ten vyjadřuje rozložení odstínů vzhledem k množství pixelů v obraze. Reálné snímky mají pozvolné přechody jasů, proto je obtížné automatizovat proces, co ještě spadá do objektů a co do pozadí. Jsme tedy nuceni vykonávat tuto činnost většinou sami, na základě našich zkušeností.

Na obr. 5.8 jsou uvedeny tři různé případy histogramů. Červená křivka ilustruje ideální případ. Máme k dispozici dva píky, které jsou dokonale odděleny. Širší pík zastupuje tmavé pixely, užší zase výrazně světlé. Práh můžeme zvolit kdekoli v prázdné oblasti mezi nimi. Binarizace nebude zatížena žádnou chybou. Studiovými podmínkami bychom snadno docílit takového stavu. Pozornější čtenář perfekcionista si může povšimnout, že oba píky nejsou možná ve všech ohledech zcela ideální. Několik pixelů bude pravděpodobně mimo dynamický rozsah snímače, to nemá ale v tomto případě samozřejmě žádný vliv.

Zelená křivka pochází z reálné fotografie. Vidíme, že obsahuje dvě maxima, nejsou ale zcela jasně oddělená a navazují na sebe plynulým přechodem. Určité chybě se při určení prahové hodnoty nevyhneme. Řešením může být změna podmínek při snímání.

Modrá křivka pro nás představuje zákeřnou modelovou situaci. Maxima zde jsou, ale jen velice těžko rozeznatelná. Binarizace by v tomto případě zcela pozbývala smyslu. Opět záleží na situaci, zda by se nedaly změnit podmínky snímání.



Obr. 5.8: Ilustrace 3 různých histogramů.

Často je zapotřebí použít jiných metod segmentace, mimo problém zjištění prahu je někdy obtížné oddělit objekty od nežádoucí informace, protože nejsou jasově rozlišitelné. Pak může být nutný lidský zásah, jak je to vidět na obr. 5.7.

5.3.4 Rozpoznání objektů

Cílem tohoto postupu je získat informaci o počtu objektů, z kterých se objekt skládá a ke každému objektu přiřadit seznam pixelů tvořících tento objekt. Taková informace je již postačující pro následnou analýzu vysoké úrovně. [41]

Nejjednodušší algoritmus je založen na postupném procházení řádků a sloupců binarizovaného obrazu. Princip algoritmu je takový, že se nejprve vynulují počítadla a začnou se postupně procházet pixely v obrazu. V okamžiku, kdy narazíme na první buňku nového objektu, inkrementujeme stav počítadla. U každé další nalezené buňky zjišťujeme, zda není tato buňka součástí objektu, který jsme již zaznamenali. Kontrola probíhá tím způsobem, že se zjišťuje přítomnost buněk o buňku zpět v osách x a y. Pokud má tedy buňka ve zpětném směru souseda, patří do zaznamenaného objektu, stav počítadla objektů se nezmění, ale změní se počet buněk v rámci tohoto objektu a zaznamenají její souřadnice. Algoritmus je tedy sice jednoduchý, ale i pomalejší a náročný na paměť, zvlášť u větších obrazů.

5.4 Analýza vysoké úrovně

Celý proces zpracování obrazu od snímání až po vyhodnocení do jisté míry napodobuje vnímání okolí naším zrakem. Analýzu vysoké úrovni bychom mohli jednoznačně přirovnat ke zpracování obrazových informací naším mozkem. Touto analýzou se tedy snažíme porozumět obsahu obrazu, přisoudit vzájemné vztahy mezi objekty nebo je popsat i samostatně. Je ale poněkud neschůdné mít natolik výkonný a variabilní výpočetní systém, jakým je lidský mozek. Nemůžeme už použít obecné algoritmy jako v případě analýzy nízké úrovně. Analýza vysoké úrovně slouží k vyhodnocení dat z různých oborů (astrofyzice, geologii, medicíně, strojírenství atd.), proto budeme výrazně méně používat obecné algoritmy a těžištěm řešení bude psaní vlastních algoritmů, řešící konkrétní úlohu.

Vstupem jsou pro nás zpracované informace z nízké úrovně. Máme tedy korigovaná geometrická zkreslení, redukovaný šum, spočítané objekty a v jejich rámci i počet pixelů se souřadnicemi. Obraz je téměř vždy binarizovaný.

Snažíme se najít taková kritéria, která nám pomohou od sebe odlišit samotné objekty i celé obrazy. Obraz tedy vyhodnocujeme z hlediska velikosti a tvaru objektů v rovině i prostoru, poloze objektů a míře pokrytí obrazu objekty. Závislosti, které nám popisují tato kritéria, nazýváme morfologické charakteristiky (z řeckého *morfé* - tvar). To mohou být funkce nebo tzv. příznaky (vlastnosti obrazu v podobě číselné hodnoty). K vlastnostem, podle kterých pak popisujeme obraz patří především:

1. Informace o celém obrazu
2. Informace o jednotlivých objektech
3. Informace o rozložení objektů

5.4.1 Informace o celém obrazu

Většinou je vztahujeme na nějaký fyzikální parametr, který se mění pro různé obrazy. Jsou nejméně detailní, protože popisují obraz jako celek, někdy však postačují.

K základním charakteristikám této skupiny patří koncentrace objektů a stupeň pokrytí.

Koncentrace objektů

Výpočet provádíme pomocí počtu objektů (známe z binarizace) a velikosti reálného obrazu nebo známého měřítka. Typicky se používá pro časový vývoj koncentrace objektů, tedy např. v biologii, růstu tenkých vrstev, geografii apod. [41]

Stupeň pokrytí

Vyjadřuje plochu objektů (černá barva) vzhledem k pozadí (bílá barva). Čísla opět známe z binarizace a velikosti obrazu v pixelech.

5.4.2 Informace o jednotlivých objektech

Popisujeme zde vlastnosti jednotlivých objektů. Ke všem objektům přistupujeme rovnocenně. Výsledky vyhodnocujeme statisticky. Vylučujeme ale přitom objekty, které by nám zkreslily výsledky, např. jen části objektů na krajích.

Na výběr máme více metod, je tedy nutné zvolit tu nejvhodnější. Uvedu zde ve zkratce rozdělení efektivních poloměrů a rozdělení tvarových faktorů.

Rozdělení efektivních poloměrů

Z hlediska popisu tvaru nepravidelných objektů využíváme porovnání s ideálním tvarem – tím je kruh. Vyjdou nám tak dvě charakteristiky. První se bude týkat efektivních poloměrů, pracujeme s objekty, jako by byly dokonale kruhové.

Efektivní poloměry objektů určíme z reálné plochy objektu v rastru, tedy:

$$R_{ef} = \sqrt{\frac{S_{objektu}}{\pi}} \tag{1}$$

Rozdělení efektivních poloměrů

Výpočtem efektivních poloměrů se nám rýsuje další charakteristika. Tou je odchylka objektu našeho přesného kruhu, nazýváme ji **tvarovým faktorem**.

Princip je jednoduchý, porovnáme plochu a obvod objektu. Čím bude objekt nepravidelnější, tím bude mít větší obvod a bude se více lišit od ideálního stavu – kruhu. Výpočet jsme jen uzpůsobili tak, aby pro kruh vycházel tvarový faktor rovný jedné. Obvod je ve jmenovateli, čili míra nepravidelnosti bude růst od jedničky k nule. [41]

$$FF = 4\pi \frac{S_{objektu}}{(O_{objektu})^2} \tag{2}$$

5.4.3 Informace o rozložení objektů

Tyto charakteristiky mohou být do určité míry zkreslující. Je důležité si uvědomit, do jaké míry odpovídá dvourozměrné zobrazení původnímu reálnému obrazu. Třeba v astronomii mohou být vzdálenosti v třetím rozměru oproti průmětu

opravdu astronomické, je tedy potřeba uvážit všechny faktory. Typicky se jedná o důležitou charakteristiku pro oblasti s růstem počtu objektů - tedy právě astronomie, biologie, nanášení tenkých vrstev apod.

Použité algoritmy pak dělíme podle charakteru zdrojových objektů - na rozložení bodových a plošných objektů. Některé z algoritmů se dají použít pro obě oblasti.

Radiální distribuční funkce

Jde o metodu původně určenou pouze pro bodové objekty. Po nahrazení plochy objektů jejich těžištěm ji lze ale použít i pro plošné objekty, pokud jejich rozměry budou vzhledem ke vzdálenostem mezi nimi zanedbatelné.

Princip je takový, že postupně bereme od bod zastupující objekt jako střed mezikruží. Poloměr obou kružnic se postupně mění ve zvoleném intervalu, určíme počet objektů v mezikruží. Pak můžeme spočítat radiální distribuční funkci:

$$P(r) \cong \frac{1}{n_0} \cdot \frac{\Delta n}{2\pi r \Delta r} \tag{3}$$

n_0 - koncentrace objektů
Δn - počet objektů v mezikruží
r - poloměr
Δr - interval mezikruží

Vzorec platí relativně přesně pro velká *r*. Poblíž objektu se logicky zvyšuje míra šumu. Můžeme jej ale redukovat opakováním pro větší počet objektů a následným průměrováním.

Vzorec je postaven tak, že při zcela náhodném uspořádání objektů v obrazu bude hodnota radiální distribuční funkce rovna jedné. Místa, kde bude větší soustředěnost objektů ve vzdálenosti *r*, se projeví zvýšením hodnoty a naopak. [41]

6 Integrální transformace

K naplnění našich fyzikálních tužeb je nezbytný i určitý matematický aparát. Integrální transformace patří určitě k jeho obtížnější části. Rozborem názvu dojdeme k tomu, že vzájemně propojují dvě transformované veličiny s využitím integrálního počtu. V oboru přírodních věd jsou těmito veličinami obvykle čas a frekvence. Nejvýznamnějším zástupcem je **Fourierova transformace**.

K matematickým výpočtům používáme vhodné programy. Např. MATLAB obsahuje celou řadu vhodných nástrojů i knihoven, ale existují i specializované softwarové produkty určené pro obtížnější výpočty. Jejich hlavním představitelem pro oblast integrálních transformací je zřejmě program **Maple**, postihující většinu matematických disciplín.

6.1 Fourierova transformace

Jde o matematickou metodu, která nám pomáhá vyjádřit časově závislou funkci popisující obraz do vhodnějšího pojetí v závislosti na frekvenci. Využíváme přitom harmonické funkce, díky kterým můžeme aproximovat většinu periodických signálů. Transformace nám pak umožňuje vybrat ze zdrojového signálu určité části, s kterými pak můžeme dále manipulovat. Můžeme některé části zvýraznit, detekovat hrany a objekty, rekonstruovat obraz, redukovat šum, případně i provést aproximaci určitých fyzikálních jevů, které bychom jinak obtížně řešili, např. zobrazení tenkou čočkou v rovině. V teoretické fyzice se využívá pro řešení parciálních diferenciálních rovnic.

6.1.1 Spojitá Fourierova transformace

Někdy nás zajímá přímo frekvenční spektrum studovaného jevu. Tak tomu bylo i u objevení této transformace baronem Jeanem Baptistem Josephem Fourierem. Dlouho nemohl nalézt řešení pro nestacionární rovnici vedení tepla. Převedením do frekvenčního spektra rovnici vyřešil. Transformaci, která je na jeho počest po něm pojmenovaná, použil mimoděk jako vedlejší nástroj. Protože jde o jednosměrný převod, označujeme ji jako **přímou Fourierovu transformaci**.

Je definována vztahem:

$$F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-j\omega t}\, dt \tag{1}$$

Je nutné, aby byla funkce f(t) po částech spojitá, tedy integrovatelná.
Funkce f(t) pro nás představuje zdroj - spektrum signálu v čase. F(ω) je obrazem této funkce ve frekvenčním spektru (ω = úhlová frekvence).

Komplexní tvar $e^{-j\omega t}$ je souborem harmonických funkcí (sin a cos), kterými je popsán časově závislý signál, lišících se v úhlové frekvenci a fázi.

Častějším postupem je převedení jevu do frekvenčního spektra, čímž se nám rovnice zjednoduší. Snadněji zde dosáhneme potřebných úprav a následně opět převedeme zpět do časové závislosti. Transformace je tedy vlastně obousměrná, celý převod označujeme jako **zpětnou Fourierovu transformaci**. Tímto způsobem např. výrazně snadněji redukujeme šum ve zpracovávaném signálu.

Je definována vztahem:

$$f(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega)\, e^{+j\omega t}\, d\omega \tag{2}$$

6.1.2 Diskrétní Fourierova transformace (DFT)

Při zpracování signálu výpočetní technikou jsme nuceni pracovat jen se vzorky původního signálu, nemůžeme použít předchozí spojité definice. Je tedy nutné zavést jejich diskrétní alternativy. Kromě spočetného množství nabývajících hodnot se liší i periodou (2π).

Konečná přímá diskrétní Fourierova transformace *je definována vztahem:*

$$F_k = \sum_{i=0}^{N-1} f_i\, e^{\frac{-2\pi ijk}{N}} \quad , \tag{3}$$

kde N je počet vzorků; i,k = 0, 1, 2, …, N-1

Konečná zpětná diskrétní Fourierova transformace *je definována vztahem:*

$$f_i = \frac{1}{N} \sum_{k=0}^{N-1} F_k\, e^{\frac{+2\pi ijk}{N}} \tag{4}$$

6.1.3 Rychlá Fourierova transformace (FFT – Fast Fourier Transform)

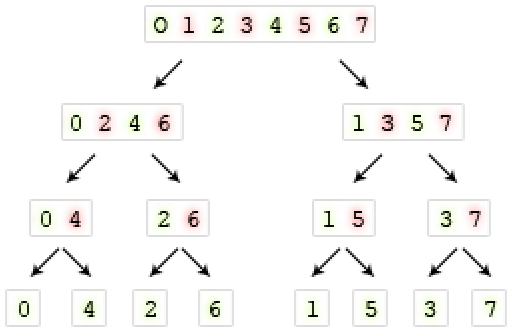
Provádění výpočtu diskrétní Fourierovy transformace je časově velmi náročné. Předchozí vztahy jsou ekvivalentní vyčíslení hodnotě polynomu řešeného Hornerových schématem, což je přibližně N^2 operací. Už pro malý počet vzorků

čas výpočtu neúměrně narůstá. Řešením je takový algoritmus, který minimalizuje násobení, protože na všech výpočetních systémech je násobení časově náročnější než sčítání, obzvlášť v případě komplexních čísel.[45]

V roce 1965 publikovali John Tukey a James Cooley rychlý algoritmus Fourierovy transformace v časopisu Mathematics of Computation, tzv. Cooley-Tukey algoritmus. Oproti původní DFT se výpočet značně urychlil na $N \times (log\ N)$ operací. Pokud bychom tak měli k dispozici např. 10 vzorků, pro DFT by vyšlo 100 operací, s FFT pouze 10. Pro vyšší hodnoty N by byl rozdíl ještě výraznější, u náročnějších úloh jde o uspořené jednotky i desítky hodin. Fourierova transformace tak rázem získala na značné oblibě, které se těší dodnes.

FFT nepředstavuje další vzorce, přináší „pouze“ zefektivnění výpočtu DFT. Princip vychází z rozkladu původní posloupnosti N prvků na několik dílčích posloupnosti, které vyžadují menší počet aritmetických operací DFT. Většinou se kvůli efektivitě volí rozklad posloupnosti na dvě části, tj. podposloupnost sudých a lichých členů, viz obr. 6.1.

Časová úspora tohoto algoritmu oproti DFT může být v rozdílech několika řádů, tj. sekundy proti hodinám či desítkám hodin. Rychlá Fourierova transformace je velmi často používaná, bývá tedy v součástí většiny programovacího softwaru s fyzikálním nebo matematickým zaměřením.



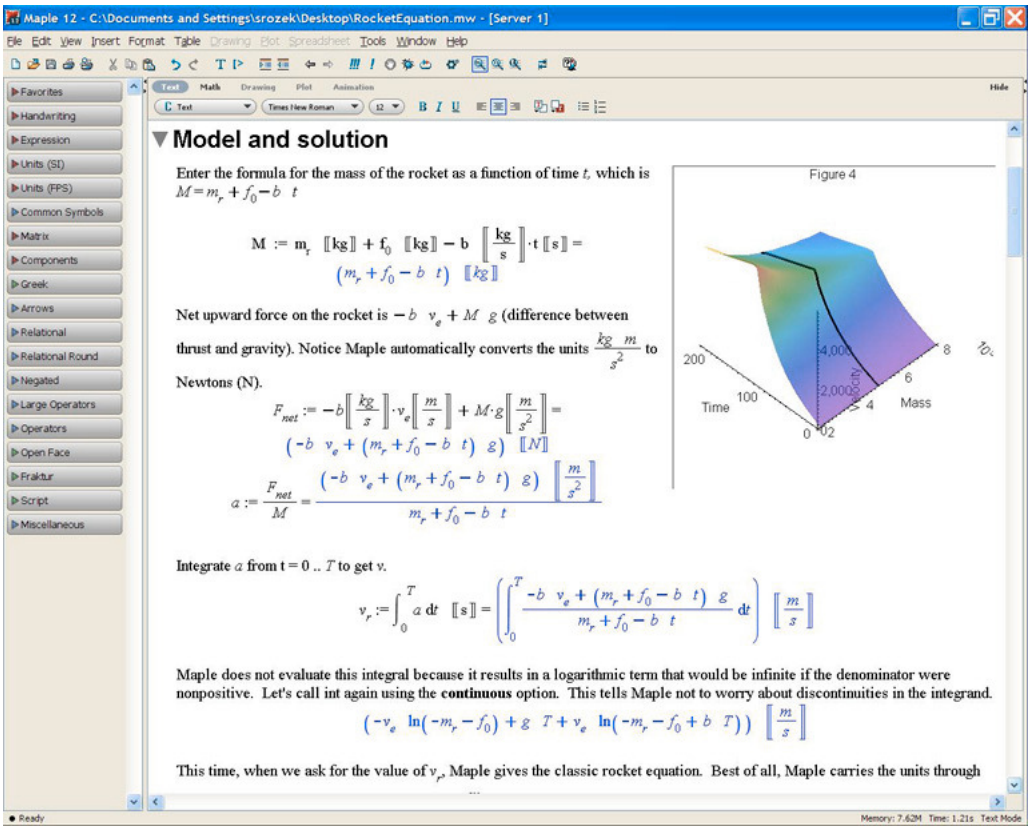
Obr. 6.1: FFT - ukázka rozkladu původní posloupnosti na podposloupnosti.

6.2 Maple

Je komplexní komerční systém počítačové algebry určený pro symbolické výpočty. Tím je myšleno, že můžeme pracovat se symboly místo s čísly. V jednom výpočtu tak můžeme nahradit složitý polynom, matici nebo jiný matematický objekt zástupným symbolem, který použijeme do dalších vztahů. Pracujeme tak bez vlivu na úkor přesnosti při zaokrouhlování, jako je tomu u numerických výpočtů. Zápis je přitom přehledný a přesně v souladu s pravidly algebry. Výpočty

tohoto typu označujeme zkratkou CAS (Computer Algebra System), představující samostatný obor. Přehled dalších systémů počítačové algebry je uveden ve zdroji [46]. [47]

Maple byl vytvořen v roce 1980 univerzitou ve Waterloo, od roku 1988 je vyvíjen firmou Maplesoft Inc. jako komerční produkt. Název vznikl jako akronym z anglických slov MAthematics PLEasure (matematika potěšením). Ve své kategorii je značně oblíbený, podobně kladných hodnocení dosahuje z konkurenčních komerčních systémů snad jen Mathematica.



Obr. 6.2: Maple 13, prostředí prozatím nejnovější verze systému. [48]

Jde o multiplatformní systém, vydávají se verze pro operační systémy Windows, Linux, Solaris a Macintosh. Linux je oproti Windows z hlediska rychlosti výhodnější a to i pokud v něm spustíme přes aplikaci Wine verzi pro Windows. Nárůst výkonu je cca až 20%, podobně jako u MATLABu.

Maple nabízí kompatibilitu s MATLABem, Microsoft Excelem, CAD systémy, po doplnění odpovídajícími toolboxy i s LabVIEW a dalšími. Kód se dá přímo konvertovat i do jiných programovacích jazyků (Fortran, C, Java, Visual Basicu.). I když to je vhodné, není potřeba znát syntaxi jazyku Maple, příkazy jsou dostupné přes grafické rozhraní systému a kontextovou nápovědu.

Systém obsahuje i nástroje k vizualizaci, simulaci a modelování. Můžeme v něm tvořit tzv. maplety. Jde o interaktivní prostředí, které se dá spouštět přímo v Maplu i nezávisle jako samostatná aplikace. Je to dáno prostřednictvím Javy, maplety jsou v podstatě obdobou javovských apletů, díky tomu mohou být přístupné i prostřednictvím internetu.

```
printf("Hello World!");
```

Obr. 6.3: „Hello world!" - Maple. [27]

Z mediálně známějších použití jmenuji alespoň využití Maplu společností Renault, výrobcem automobilů. Cílem bylo snížení hluku a vibrací u dieselového motoru modelu Laguna. Důvodem byl komfort a snížení míry opotřebení součástek. Byl sestaven model motoru, popsaný pěti diferenciálními rovnicemi. Testováním modelu se našlo kritické místo u setrvačníku. Stačila pak jen mírná úprava změnou přívodu vzduchu. Simulace pak ukázala snížení vibrací až o 30% . K vytvoření a řízení modelu stačil kvalifikovanému odborníkovi jeden den.

Závěr

Cílem práce bylo srozumitelně a populárně přiblížit obor počítačové fyziky, což se podle ohlasů myslím podařilo.

V případě prezentace jsem se soustředil především na její celkový dojem a názornost k tématu. Vzhledem k délce prezentace (121 snímků) jsem se snažil vytvořit takovou šablonu, která by svým barevným schématem zbytečně neodváděla pozornost od obsahu, ale přesto byla atraktivní. Pro udržení pozornosti jsem také vložil po určitých intervalech obrázky humornějšího charakteru. Celkově jsem volil takový obsah, který by byl zajímavý a oslovil laickou veřejnost k metodám a prostředkům počítačové fyziky.

V tištěné části práce jsem se často zaměřil na historický vývoj tématu a především na úplné počátky. Myslím, že pochopením důvodů, které vedly k vytvoření hardwaru i softwaru, který jsem zmínil, si čtenář udělá snáze představu o dané problematice, než abych podrobně popisoval její současný stav.

Při vyhledávání informací jsem použil často internetových zdrojů. Rozhodl jsem se pro tuto možnost, protože práce je zaměřena na mladou generaci a potenciální zájemci si tak snadno mohou najít doplňující ověřené informace. Druhým důvodem bylo množství textu, který jsem selektoval a pročítal. Úskalím takové volby je ale větší možnost neobjektivity. Proto jsem se snažil u všech informací pátrat po původních zdrojích a informace, které jsem přímo neznal, jsem porovnal mezi více důvěryhodnými zdroji.

Autorem části prezentace, která se zabývá modelováním, je kolega Bc. Jaroslav Harvalík. Vypracovává ve své diplomové práci další část společného tématu se zaměřením právě na oblast modelování a simulací v počítačové fyzice.

Na přiloženém médiu je k dispozici elektronická verze tištěné části práce a prezentace.

Seznam použitých zdrojů

Všechny uvedené odkazy jsou platné k datu 28. 11. 2009

[1] ► časopis Vesmír 76, 310, 1997/6, článek *Věstonická Vrubovka*
http://www.vesmir.cz/clanek.php3?CID=2861

[2] ► A Brief History of the Abacus
http://www.ee.ryerson.ca/~elf/abacus/history.html

[3] ► The Controversial Replica of Leonardo da Vinci's Adding Machine
http://192.220.96.166/leonardo/leonardo.html

[4] ► Wilhelm Schickard
http://www-history.mcs.st-andrews.ac.uk/Biographies/Schickard.html

[5] ► Do muzea zve pan Jacquard...
http://wp.wpublisher.cz/malenoviny/index.php?ID=1503

[6] ► Laçage des cartons du Jacquard
http://metiers.free.fr/dcanuts/canutsn.html

[7] ► Programátoři bez počítače
http://www.fi.muni.cz/usr/jkucera/pv109/2000/matyasko_referat.html

[8] ► Doron Swade operating Babbage's Difference Engine
http://www.sciencemuseum.org.uk/images/I033/10303328.aspx

[9] ► Augusta Ada Lovelace
http://projects.exeter.ac.uk/babbage/ada.html

[10] ► Babbage's Analytical Engine
http://www.sciencemuseum.org.uk/images/I030/10297676.aspx

[11] ► Příběh počítače
http://www.galaxie.name/index.php?clanek=pribeh-pocitace-1-dil

[12] ► Konrad Zuse's Z1 and Z3 Computers
http://www.epemag.com/zuse/part4a.htm

[13] ► 10 nejvýznamnějších objevů 20. století
http://www.21stoleti.cz/rservice.php?akce=tisk&cisloclanku=2005021809

[14] ► *Meyers Konversationslexikon*; 1888

[15] ► První český samočinný počítač SAPO
http://www.fi.muni.cz/usr/jkucera/pv109/vystavka/xrycka_01-02.html

[16] ► Jaroslav Zelený, Božena Mannová; *Historie výpočetní techniky*; Scientia; 2006; ISBN: 80-86960-04-8

[17] ► Elektronkový počítač EPOS 1
http://www.fi.muni.cz/usr/jkucera/pv109/vystavka/xrycka_01-03.html

[18] ► TRADIC Second Feasibility Computer
http://ed-thelen.org/comp-hist/BRL61-l.html

[19] ► Confessions of an Antediluvian Geek - A personal history of computing
http://docmorbius.net/article0002/Confessions03.html
[20] ► IBM Personal Computer
http://www-03.ibm.com/ibm/history/exhibits/pc25/pc25_PH02.html
[21] ► Oak Ridge National Laboratory
http://www.flickr.com/photos/oakridgelab/3592579200/
[22] ► Dharmendra S. Modha
http://www.almaden.ibm.com/cs/people/dmodha/
[23] ► Dixit Algorizmi
http://edi.fmph.uniba.sk/~winczer/historia/GHWZ/index.html
[24] ► Perforating Pleyela master rolls - Paris, 1914
http://www.pianola.org/history/history_jacquard.cfm
[25] ► Vývoj programování a programovacích jazyků
http://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm
[26] ► Paní Hopper a FLOW-MATIC
http://www.osu.cz/katedry/kip/aktuality/sbornik99/cevela.html
[27] ► The Hello World Collection
http://www.roesler-ac.de/wolfram/hello.htm
[28] ► Simulink a MATLAB pomáhají zachránit jadernou ponorku Kursk
http://www.humusoft.cz/old/pub/matlab/10_03/kursk.htm
[29] ► Blood Vessel
http://www.euro.comsol.com/showroom/gallery/660/
[30] ► COMSOL Multiphysics
http://www.humusoft.cz/produkty/comsol/index.php?lang=cz&p1=1&p2=2
[31] ► The First "Computer Bug
http://www.history.navy.mil/photos/images/h96000/h96566kc.htm
[32] ► Vladimír malíšek; *Isaac Newton, zakladatel teoretické fyziky*;
Prometheus; 1999, ISBN: 80-7196-136-1;
[33] ► F. Koutný; *Newton, Calculus, Minimalizace*; Zlín
http://www.zas.cz/download/newton-predn.pdf
[34] ► Isaac Newton
http://www.marcdatabase.com/~lemur/lemur.com/gallery-of-antiquarian-technology/worthies/
[35] ► Euler Leonhard
http://www.techmania.cz/edutorium/art_vedci.php?key=224
[36] ► Leonhard Euler
http://turnbull.mcs.st-and.ac.uk/history/Mathematicians/Euler.html
[37] ► Schrödinger Erwin
http://www.techmania.cz/edutorium/art_vedci.php?key=62

[38] ► Schrödinger Erwin
http://osulibrary.oregonstate.edu/specialcollections/coll/pauling/bond/pictures/portrait-schrodinger.html
[39] ► http://phil.elte.hu/redei/Utrecht/UtrechtNeumann.html
[40] ► Video Game Consoles (1980-1982)
http://www.thegameconsole.com/videogames80.htm
[41] ► Rudolf Hrach; *Počítačová fyzika II.*, PF UJEP Ústí nad Labem; 2003

[42] ► Radomír Ščurek; *Biometrické metody identifikace osob v bezpečnosti praxi*; VŠB TU Ostrava; 2008
[43] ► Fiftieth Anniversary of First Digital Image Marked
http://www.nist.gov/public_affairs/techbeat/tb2007_0524.htm
[44] ► Aral Sea
http://www.nasa.gov
[45] ► skripta *Fourierova transformace;* Olomouc; 2003
http://apfyz.upol.cz/ucebnice/down/mini/fourtrans.pdf
[46] ► SAC Systems Listing
http://www.symbolicnet.org/systems/
[47] ► Jiří Hřebíček, Jan Kohout; *Úvod do systému Maple ;*FI Masarykova univerzita v Brně; 2004
[48] ► http://www.maplesoft.com/
[49] ► Leading car manufacturer Renault solves unwanted engine noise and vibration using Maple
http://www.maplesoft.com/company/publications/articles/view.aspx?SID=19216
[50] ► Tisková zpráva Sprinx Systems - superpočítač Amálka
http://hpc.sprinx.cz/amalka.aspx
[51] ► Seven Bridges of Königsberg
http://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

Seznam doporučených zdrojů

Abacus

- http://www.tux.org/~bagleyd/AbacusGB.html
- http://www.fi.muni.cz/usr/jkucera/pv109/xdavidov.html
- http://jaromirmatucha.ic.cz/historie_pc/abakus.html
- http://webhome.idirect.com/~totton/abacus/pages.htm
- http://www.geocities.com/maya_site/nepohualtzintzin/nepohualtzintzin.htm

Analytický stroj

- http://ei.cs.vt.edu/~history/Babbage.html
- http://www.fourmilab.ch/babbage/sketch.html

Počítače

- http://www.epemag.com/zuse/
- http://www.computer50.org/mark1/MM1.html

Superpočítače

- http://www.top500.org/
- http://folding.stanford.edu/

Počítačové jazyky

- http://www.fi.muni.cz/usr/jkucera/pv109/2002/xkriz1.htm
- http://www.root.cz/
- http://www.linuxsoft.cz
- http://www.humusoft.cz

Významné osobnosti počítačové fyziky

- http://www.zas.cz/download/newton-predn.pdf
- http://plus.maths.org/issue42/features/wilson/
- http://natura.eri.cz/natura/2003/7/20030704.html
- http://www.britskelisty.cz/9912/19991228i.html

Počítačová grafika

- http://www.root.cz/serialy/graficke-karty-a-graficke-akceleratory/
- http://www.grafika.cz/art/sazba/clanek850964718.html
- Ira Greenberg; *Processing: Creative Coding and Computational Art*; (2007); ISBN 159059617X.
- Sochor, Žára, Beneš; *Algoritmy počítačové grafiky; ČVUT; 1996; ISBN: 80-01-01406-1*

Zpracování obrazu

- přednáška Prof. Miloslava Druckmüllera - video http://www.hvezdarna.cz/prednes/271105.avi -
- Zpracování signálu a obrazu, Václav Hlaváč, Miloš Sedláček, ČVUT, 1999 http://cmp.felk.cvut.cz/~hlavac/HlavacTeachPresentCz.htm
- http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi?ID_tema=67&stranka=publ_tema
- Obrazové segmentační techniky http://www.fit.vutbr.cz/~spanel/segmentace/.cs.iso-8859-2

Integrální charakteristiky

- Hušek, P. Pyrih et al.; *Fourierova transformace*; Matematicko-fyzikální fakulta, Univerzita Karlova v Praze http://matematika.cuni.cz/dl/analyza/37-fou/lekce37-fou-pmax.pdf