**Umělá inteligence (AI)**  
Umělá inteligence (Artificial Intelligence, AI) je obor informatiky zaměřený na tvorbu softwaru a systémů, které napodobují lidské myšlení, učení a rozhodování. AI lze rozdělit na úzkou (Narrow AI), která je specializovaná na konkrétní úlohy (např. rozpoznávání obličejů), a obecnou (General AI), která by teoreticky mohla dosahovat stejných schopností jako člověk.

**Aplikace umělé inteligence ve formě LLM (Large Language Model)**  
LLM, neboli velké jazykové modely, jsou typem umělé inteligence určeným pro zpracování přirozeného jazyka. Jsou trénovány na obrovském množství textových dat a dokáží generovat souvislý a smysluplný text, odpovídat na otázky, překládat jazyky a analyzovat textové vstupy.

**Struktura neuronové sítě pro LLM a historie jejího vzniku**  
LLM jsou založeny na hlubokých neuronových sítích, konkrétně na architektuře transformátorů (Transformer), kterou představila společnost Google v roce 2017 v článku *Attention Is All You Need*. Transformátory využívají mechanismus *self-attention*, který jim umožňuje efektivně analyzovat dlouhé texty a chápat kontext slov.

**Zástupci softwaru umělé inteligence a oblasti jejich použití**

1. ChatGPT (OpenAI) – jazykový model pro generování textu, odpovídání na otázky, psaní esejí apod.
2. DALL·E (OpenAI) – model pro generování obrázků na základě textového popisu.
3. DeepL – pokročilý překladatelský nástroj využívající AI pro kvalitní překlady.
4. AlphaFold (DeepMind) – AI model pro predikci struktury proteinů v biologii.
5. Tesla Autopilot – systém umělé inteligence pro autonomní řízení vozidel.

**Využití AI v oblasti programování**  
AI může pomáhat programátorům například generováním kódu, opravou chyb nebo vysvětlením algoritmů.

**Příklad zadání: *Vytvoř jednoduchý program v Pythonu, který zjistí, zda je číslo prvočíslem.***

Generovaný kód:

def je\_prvocislo(n):

if n < 2:

return False

for i in range(2, int(n\*\*0.5) + 1):

if n % i == 0:

return False

return True

cislo = int(input("Zadej číslo: "))

print("Prvočíslo" if je\_prvocislo(cislo) else "Není prvočíslo")

* **Posouzení kódu:** Generovaný kód je správný, efektivní a používá optimalizaci při hledání dělitelů pomocí odmocniny z *n*. Program korektně zjistí, zda je zadané číslo prvočíslem, a jeho výstup je intuitivní pro uživatele.
* Umělá inteligence tedy usnadňuje programování tím, že pomáhá s generováním kódu, jeho optimalizací a detekcí chyb.

**VCS (Version Control System)** - Systém správy verzí VCS (Version Control System) je systém pro správu verzí souborů a projektů. Umožňuje sledování změn v kódu, spolupráci více vývojářů a návrat k předchozím verzím souborů. Díky VCS mohou vývojáři efektivně pracovat na různých částech projektu současně, minimalizovat riziko ztráty dat a analyzovat historické změny v kódu.

**Git** – nejpoužívanější VCS Git je distribuovaný systém správy verzí, který uchovává historii změn projektu a umožňuje paralelní vývoj. Pracuje s lokálním i vzdáleným repozitářem, což usnadňuje spolupráci mezi vývojáři. V porovnání s centralizovanými systémy, jako je SVN - (*Apache Subversion je systém pro správu a verzování zdrojových kódů, náhrada za starší CVS*), Git poskytuje větší flexibilitu a efektivitu při správě verzí.

**Způsob verzování v Gitu**

1. **Inicializace repozitáře** – pomocí git init se vytvoří nový repozitář, který uchovává historii změn.
2. **Přidání změn do staging area** – příkaz git add <soubor> přesune změny do dočasné oblasti (staging area), kde čekají na commit.
3. **Commit** – příkaz git commit -m "Popis změny" uloží změny do historie verzí s popisem změny.
4. **Vytvoření větve** (**branch**) – git branch <název\_větve> vytvoří novou větev pro paralelní vývoj, což umožňuje pracovat na více funkcích současně.
5. **Přepnutí mezi větvemi** – git checkout <název\_větve> nebo modernější git switch <název\_větve> umožňuje přepínání mezi větvemi.
6. **Sloučení větví** (**merge**) – git merge <název\_větve> spojí změny z jedné větve do druhé a vytvoří novou verzi.

**Klíčové pojmy**

1. **Staging area** – dočasné úložiště změn před jejich potvrzením (commitnutím). Umožňuje vývojářům připravit změny předtím, než se stanou trvalou součástí historie.
2. **Commit** – uložená verze projektu s unikátním identifikátorem. Každý commit je zachován v historii, což umožňuje kdykoliv se vrátit ke starší verzi.
3. **Repozitář** – úložiště obsahující historii změn projektu. Může být lokální nebo vzdálené a umožňuje spolupráci více vývojářů.
4. **Branch** (**větev**) – oddělená linie vývoje umožňující experimentování bez ovlivnění hlavní větve (main nebo master).
5. **Merge** – sloučení větví, při kterém se změny jedné větve začlení do druhé. Lze provádět automaticky, pokud neexistují konflikty, nebo ručně v případě kolize změn.

**Konflikt při merge** – nastává, když dvě změny upraví stejný řádek kódu. Git nemůže rozhodnout, kterou verzi použít, a je potřeba ruční zásah vývojáře. Konflikty se typicky řeší v editorech, které umožňují porovnání a úpravu změn (**VSCode**).

**Výhody používání Gitu**

1. **Zálohování a obnova** – možnost vrátit se ke starším verzím souborů.
2. **Paralelní vývoj** – umožňuje vývojářům pracovat na různých funkcích odděleně.
3. **Lepší spolupráce** – usnadňuje týmovou spolupráci prostřednictvím vzdálených repozitářů (např. GitHub, GitLab, Bitbucket).
4. **Efektivní správa kódu** – přehledná historie commitů a možnost sledování změn.

**3D Grafika na PC** – Základní principy konstrukce a simulace 3D grafika na počítači se zabývá vytvářením a zobrazováním trojrozměrných objektů. Hlavní principy konstrukce zahrnují:

1. **Modelování** – tvorba 3D objektů pomocí polygonů, křivek nebo voxelů.
2. **Texturování** – aplikace obrazových map na povrch objektů pro realistický vzhled.
3. **Osvětlení** **a** **stínování** – simulace světelných zdrojů a jejich interakce s objekty.
4. **Renderování** – převod 3D scény do 2D obrazu s použitím výpočetních algoritmů.
5. **Animace** – pohyb objektů, deformace a fyzikální simulace.

Používaný software Pro práci s 3D grafikou se využívají různé programy, které se liší svou složitostí a zaměřením:

1. **Blender** – open-source nástroj pro modelování, animaci a renderování.
2. **Autodesk** **Maya** – profesionální software používaný v herním a filmovém průmyslu.
3. **3ds** **Max** – široce používaný ve vizualizaci a architektuře.
4. **Cinema** **4D** – oblíbený ve filmové a reklamní produkci.
5. **ZBrush** – specializovaný na digitální sochařství.
6. **Unity** **a** **Unreal** **Engine** – využívané pro vývoj her a interaktivních aplikací.

**Druhy 3D grafiky**

1. **Polygonální modelování** – vytváření objektů pomocí sítí trojúhelníků.
2. **Procedurální** **generování** – využívání algoritmů k tvorbě obsahu.
3. **Voxelová** **grafika** – reprezentace objektů jako objemových pixelů (voxely).
4. **NURBS** **a** **Bézierovy** **křivky** – hladké plochy využívané v průmyslovém designu.

**Výhody a nevýhody 3D grafiky**

**Výhody:**

* Možnost realistické vizualizace.
* Široké využití v hrách, filmech, architektuře a vědeckých simulacích.
* Interaktivita a dynamické úpravy.

**Nevýhody:**

* Vysoké nároky na hardware.
* Složitost tvorby realistických modelů a animací.
* Dlouhé výpočetní časy při renderování složitých scén.
* 3D grafika hraje klíčovou roli v mnoha odvětvích a neustále se vyvíjí díky pokroku v hardware i software.