STŘEDNÍ PRŮMYSLOVÁ ŠKOLA
MLADÁ BOLESLAV

# ROČNÍKOVÁ PRÁCE

Ondřej Fíla

Mladá Boleslav 2025

STŘEDNÍ PRŮMYSLOVÁ ŠKOLA
MLADÁ BOLESLAV

# Pac-Man

**Autor: Ondřej Fíla**
**Studijní obor: 18-20-M/01 Informační technologie**
**Vedoucí práce: Jan Till**

Mladá Boleslav 2025

# Obsah

**Prohlášení**

Prohlašuji, že jsem svou ročníkovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

Nemám závažný důvod proti zpřístupňování této ročníkové práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Mladé Boleslavi dne ..... podpis: .....

# 1 Úvod

Smyslem této práce bylo vytvořit věrnou repliku známé videohry Pac-Man, která vznikla v 80. letech 20. století a stala se jednou z nejpopulárnějších her na arkádových automatech. Název vznikl na základě japonského slova *paku-paku* označující zvuk rychlého konzumování jídla. Tento zvuk symbolizuje chování hlavní postavy, která se v průběhu hry věnuje právě této činnosti.

Cílem je kompletně sesbírat veškeré jídlo rozptýlené po všech herních mapách, přičemž hráč musí dbát na to, aby se vyhnul duchům, kteří představují hlavní hrozbu v plnění tohoto náročného úkolu. Pokud duchové chytí Pac-Mana, hráč ztrácí život. Ztratí-li všechny své životy, znamená to konec hry. Tímto způsobem hra kombinuje prvky strategie a předvídání, kde hráč musí během okamžiku zvolit směr za účelem vyhnutí se duchům.

Hra byla vytvořena pomocí programovacího jazyka TypeScript, jenž byl vybrán díky jeho statickému typování usnadňující detekci chyb již během vývoje. Rozšíření skriptovacího jazyka JavaScript o moderní syntaxi a důkladnější správu datových typů umožňuje psaní přehlednějšího a lépe udržovatelného kódu. Použití jazyka TypeScript si vyžádalo použít Visual Studio Code, který automaticky rozpoznává typescriptové soubory bez nutnosti žádné nadbytečné konfigurace. Dalším důvodem pro volbu tohoto editoru je jeho bezplatná dostupnost a efektivní podpora vývoje různých typů projektů, obzvlášť webových aplikací.

Pac-Man se stal tématem této práce především kvůli svému zásadnímu významu v historii videoher. Zároveň se jedná o skvělý projekt, v němž lze aplikovat mnoho matematických algoritmů. Obrázky, s výjimkou úvodní obrazovky, byly vytvořeny pomocí grafického editoru s názvem Piskel. Hudba hrající na pozadí byla následně vygenerována prostřednictvím známého programu Suno využívajícího prvky umělé inteligence.

# 2 Obsah práce

## 2.1 Použité technologie

### 2.1.1 TypeScript

TypeScript je open-source programovací jazyk vytvořený a spravovaný společností Microsoft, který byl poprvé představen 1. října 2012. Jedná se o nadstavbu skriptovacího jazyka JavaScript, která přidává statické typování, čímž zvyšuje bezpečnost a předvídatelnost kódu. Kromě toho rozšiřuje tento jazyk o prvky známé z objektově orientovaného programování (OOP), jako jsou rozhraní (interfaces), modifikátory přístupu (access modifiers) nebo abstraktní třídy. Díky statickému typování TypeScript umožňuje odhalit chyby, nebo potenciální hrozby již v průběhu vývoje, což výrazně usnadňuje ladění programu. Stejně jako JavaScript podporuje širokou škálu knihoven a frameworků, které poskytují předdefinované moduly a pomocné funkce pro usnadnění vývoje. TypeScript se kompiluje do standardního JavaScriptu, který je poté možné spustit v jakémkoli prostředí podporujícím JavaScript, například v prohlížeči, nebo pomocí softwaru Node.js. Tento výsledný javascriptový kód lze následně propojit s HTML.[1]

### 2.1.2 HTML

HyperText Markup Language (HTML) je značkovací jazyk, který společně s kaskádovými styly CSS a skriptovacím jazykem JavaScript tvoří základní stavební kámen pro vývoj webových stránek. Poprvé byl představen v roce 1993 a během posledních třiceti let prošel řadou aktualizací, přičemž oficiálně nejnovější a nejpoužívanější verzí je HTML5 z roku 2014. Hlavním úkolem HTML je vytvořit

[1] *TypeScript [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-25]. Dostupné z: https://cs.wikipedia.org/wiki/TypeScript.*

strukturu samotné stránky. K tomu slouží HTML elementy, které jsou tvořeny pomocí HTML tagů. Tyto tagy symbolizují, zda daným elementem je odstavec, nadpis, odkaz, obrázek, nebo něco jiného.

Je-li uvnitř HTML tagu například písmeno *p*, jedná se o odstavec, z anglického slova *paragraph*. Poslední dobou však existuje mnoho technologií, které na jazyce HTML staví. Mezi ně patří například React, Vue, Angular.[2]

### 2.1.3 CSS

Cascading Style Sheets (CSS) je jazyk sloužící k definování vzhledu webových stránek. Poprvé jej v roce 1994 navrhl norský technolog Håkon Wium Lie, který pracoval v Evropské organizaci pro jaderný výzkum společně s vynálezcem HTML, Tim Berners-Lee. Před příchodem kaskádových stylů existovaly šablony stylů, které byly zabudované v jednotlivých prohlížečích, ale jejich využití bylo značně omezené. První standardizovaná verze s názvem CSS1 vznikla o dva roky později. Příchodem různých frameworků pro jazyk CSS došlo k výraznému zjednodušení responzivního designu, mezi které patří například Bootstrap. Pravidla pro vzhled webové stránky lze implementovat přímo v HTML, pomocí speciálního tagu *<style>*, nebo externě prostřednictvím samostatného souboru s příponou .css napojeného na HTML.[3]

[2] *HTML [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-25]. Dostupné z: https://en.wikipedia.org/wiki/HTML.*

3 *CSS [online]. Britannica, 2025 [cit. 2025-03-25]. Dostupné z: https://www.britannica.com/technology/CSS-programming-language.*

### 2.1.4 Visual Studio Code

Visual Studio Code je populární integrované vývojové prostředí (IDE) vyvinuté společností Microsoft. Tento editor umožňuje práci s různými programovacími jazyky, jako je TypeScript, což z něj činí ideální nástroj pro vývoj webových aplikací. Kromě toho nabízí možnost rozšíření pro další jazyky a runtime moduly, včetně C++, C#, Java, Python, PHP, .NET, Go a mnoho dalších. Visual Studio Code je dostupný na několika platformách, mezi ně patří Windows, MacOS, Linux. Mezi jeho klíčové funkce patří integrované ladění, kontrola verzí prostřednictvím systému Git, inteligentní dokončování kódu a tvorba uživatelského rozhraní.[4]

### 2.1.5 Piskel

Piskel je webová aplikace umožňující uživateli libovolně malovat různé obrázky a animace. Tato aplikace byla naprogramována prostřednictvím JavaScriptu a její zdrojový kód je dostupný na platformě GitHub. Díky intuitivnímu ovládání a bezplatnému použití se stává velmi oblíbeným nástrojem pro tvorbu pixelových obrázků. Prostřednictvím tohoto editoru byly vytvořeny grafické návrhy pro duchy a speciální schopnosti v mém projektu. Zároveň je Piskel podporován mnoha prohlížeči, mezi ně patří Google Chrome, Microsoft Edge, Mozilla Firefox a Internet Explorer 11. Editor je poskytován pod licencí Apache License 2.0, která umožňuje volné použití a distribuci vytvořených materiálů, pokud je zachováno oznámení o licenci.[5]

[4] *Visual Studio Code [online]. Microsoft, [2015] [cit. 2025-03-25]. Dostupné z: https://visualstudio.microsoft.com/cs/#vscode-section.*

[5] *Piskel [online]. GitHub, 2008 [cit. 2025-03-27]. Dostupné z: https://github.com/piskelapp/piskel.*

### 2.1.6 Suno

Suno je program využívající umělou inteligenci, který umožňuje uživatelům generovat hudbu podle zvoleného žánru. Tento editor byl uveden na trh v roce 2023 a rychle se stal populárním nástrojem pro vytváření hudebního doprovodu. Uživatelé mohou do textového pole zadat vlastní text, vybrat žánr a Suno následně vygeneruje plnohodnotnou píseň. Program nabízí obrovský rozsah možností pro tvorbu hudby, čímž usnadňuje tvorbu originálních skladeb i pro lidi bez hudebních předpokladů. Dále také umožňuje zvolit více než jeden žánr současně, což výrazně rozšiřuje tvůrčí možnosti. Program navíc chrání autorská práva tím, že uživatelům neumožňuje upravovat písně, které jsou chráněny právními předpisy nebo vlastníky práv. Z tohoto důvodu byl vybrán jako ideální nástroj pro tvorbu hudby v tomto projektu.[6]

### 2.1.7 JSON

JavaScript Object Notation (JSON) je populární datový formát založený na syntaxi javascriptových objektů. Jedná se o textovou podobu, která je snadno čitelná pro člověka a zároveň zpracovatelná počítačem. JSON se stal jedním z nejrozšířenějších formátů pro výměnu dat, zejména v oblasti vývoje softwaru, kde hraje klíčovou roli při komunikaci mezi klientskými a serverovými službami. JSON je rovněž podporován nerelačními databázemi, které nevyužívají tradiční tabulkovou strukturu. Kompatibilita s mnoha programovacími jazyky z něj činí univerzální nástroj pro práci s daty v moderních aplikacích.[7]

[6] *Suno [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-27]. Dostupné z: https://en.wikipedia.org/wiki/Suno_AI.*

[7] *JSON [online]. Oracle, 2024 [cit. 2025-03-30]. Dostupné z: https://www.oracle.com/database/what-is-json/.*

## 2.2 TypeScript konfigurace

Před prací na celém projektu bylo třeba vytvořit inicializační soubor *tsconfig.json*. Tento soubor slouží k definování nastavení programovacího jazyka TypeScript pro daný projekt. Soubor lze vygenerovat zadáním příkazu *tsc –init* do příkazového řádku. Mezi jeho výhody patří automatická kompilace všech existujících typescriptových souborů, nastavení striktního hlídání datových typů, ale také nastavení zdrojových adresářů pro javascriptové a typescriptové soubory. Většina těchto nastavení jsou předem nadefinovaná a uživatel je tedy nemusí přidávat manuálně.

```
{
  "compilerOptions": {
  "outDir": "./res/js", // Kde se nachází zkompilované js soubory
  "rootDir": "./res/ts", // Kde se nachází ts soubory
  "target": "ES2022", // Verze JavaScriptu, do které se TypeScript kompiluje
  "module": "ESNext", // Použitý systém modulů
  "esModuleInterop": true, // Lepší kompatibilita s moduly CommonJS
  "strict": true, // Povolení všech kontrol datových typů
  },
}
```

Obrázek **1**: Ukázka nastavení tsconfig.json [vlastní zdroj]

## 2.3 Herní mapa

Herní mapa byla původně vytvořena jako dvourozměrné pole, kde každý prvek představuje určitý objekt ve hře. Každý z těchto objektů na mapě je reprezentován číslem určující jeho účel. Číslo jedna symbolizuje zeď, která představuje pevnou bariéru znemožňující průchod nejen Pac-Manovi, ale i duchům. Číslo dvě označuje jídlo, které je rozprostřeno po celé mapě a jedná se o hráčův hlavní úkol jej sesbírat dříve, než přijde o všechny životy. Číslo tři představuje prázdné místo, které je nedostupné všem herním postavám. Toto číslo bylo použito k odlišení využité speciální schopnosti od nedostupného místa.

Další jednociferná sudá čísla následující po čísle tři slouží k definování speciální schopnosti v základním stavu. Když je speciální schopnost aktivována, odpovídající prvek na mapě se inkrementuje, což signalizuje, že speciální schopnost na daném poli byla využita. Tímto úkonem má každá speciální schopnost své unikátní liché číslo, díky němuž lze rozlišit různé schopnosti mezi sebou. Po restartování hry se číslo na příslušném poli sníží o číslo jedna, čímž navrátí speciální schopnosti zpět.

Všechny mapy použité v mém projektu mají stejný rozměr, konkrétně dvacet tři řádků a dvacet jedna sloupců. Tento jednotný formát zajišťuje jednotnost mezi různými úrovněmi. Z počátku byla herní mapa uložena přímo v hlavním souboru *script.ts* jako konstanta *map.* Tento způsob řešení by však postupem času vedl k nepřehlednosti hlavního souboru, protože přidání většího množství map by soubor zahltilo několika nadbytečnými řádky.

Ideálním řešením tohoto problému bylo využití technologie JSON, kde byl vytvořen speciální soubor *data.json*, do kterého byly informace o mapách uloženy. Kromě samotných dat o mapách soubor zahrnuje také komentáře vysvětlující nejen symboliku čísel používaných pro různé objekty ve hře, ale i informace o každé úrovni. Přechodem na formát JSON byla zajištěna jednodušší správa map, přehlednost kódu a snadné přidávání nových úrovní.

### 2.3.1 Nahrání informací do hlavního souboru

Pro práci s daty uloženými ve formátu JSON bylo třeba v TypeScriptu tato data načíst, aby bylo možné s nimi manipulovat. Před nahráním dat ze souboru bylo zapotřebí deklarovat proměnnou *currentMap*, která bude obsahovat nahrané informace. Poněvadž všechny mapy mají stejný rozměr, tak tato proměnná symbolizuje prázdné dvourozměrné pole s třiadvaceti prázdnými jednorozměrnými poli.

Poté bylo možné implementovat funkci *loadData*. Tato funkce je asynchronní, protože nahrání dat ze souboru může být časově náročný úkon, na který musíme počkat před spuštěním hry. Data jsou získávána pomocí funkce *fetch*, která načítá obsah souboru z určené cesty. Jedná se o úkon, který může trvat delší dobu a je třeba na něj počkat, k tomu je využito klíčové slovo *await*.

Jakmile se data úspěšně načtou, uloží se do konstanty *data*, která uchovává tato data převedená do podoby JSON. Na tento úkon je také třeba počkat, proto se zde opět nachází slovo *await*. Do proměnné *currentMap* se tato data následně uloží, přičemž je vybrána mapa odpovídající aktuální úrovni hráče. Úroveň hráče je uložena jako celé číslo, kde počáteční hodnotou je číslo jedna.

Jelikož se jedná o práci s poli, kde první prvek v poli má index s hodnotou nula, toto číslo muselo být odečteno číslem jedna, aby odpovídalo správnému indexu v poli. Při práci s objektem JSON byl vybrán klíč currentLevel, ve kterém jsou obsaženy informace o aktuální mapě.

```
const loadData = async (): Promise<void> => {
     const file: Response = await fetch("../res/data/data.json");
     const data = await file.json();
     currentMap = data[pacman.currentLevel - 1].currentLevel;
};
```

Zdrojový kód č. 1

### 2.3.2 Vykreslení mapy

Pro vizuální reprezentaci herní mapy byl využit HTML element *<canvas>*, který byl do dokumentu přidán prostřednictvím TypeScriptu. Rozměry tohoto plátna byly nastaveny tak, aby každý blok na mapě měl rozměry čtverce 24x24, což bylo nezbytné vzhledem k tomu, že žádná mapa neobsahuje stejné množství řádků a sloupců.

Dále bylo nutné vytvořit štětec uložený v konstantě *ctx*, jenž umožňuje uživateli malovat na plátno (*canvas*). Zároveň byl nastaven tak, aby malování bylo možné pouze ve dvou dimenzích.

Po vytvoření těchto dvou základních komponent bylo třeba implementovat několik funkcí, které zajišťují vykreslení jednotlivých prvků na mapě, včetně zdí, jídla a speciálních schopností. Po načtení mapy pomocí funkce *loadData* do proměnné *currentMap* byly v rámci dvou cyklů procházeny všechny prvky ve dvourozměrném poli.

Dva cykly byly použity, protože se jedná o dvourozměrné pole. První cyklus prochází řádky pole, zatímco druhý cyklus má na starost samotné prvky v daném řádku. Na základě hodnoty, která určuje jednotlivé objekty na mapě, byly následně vykresleny příslušné objekty, jako je jídlo, zeď nebo speciální schopnosti pro Pac-Mana.

Vykreslování mapy a herních objektů probíhá v herní smyčce, která se volá šedesátkrát za sekundu, což zajišťuje plynulost hry. V této herní smyčce zároveň probíhá vykreslování duchů, Pac-Mana a zpracovává jejich chování.

### 2.3.3 Teleport

Portál v mém projektu je implementován pouze v první mapě, která byla vytvořena na základě originální hry. Princip pohybu přes teleport nejen u Pac-Mana, ale i duchů, spočívá v tom, že pokud se jeden z krajních bodů v daném směru Pac-Mana nebo duchů ocitne na kraji herní mapy a Pac-Man nebo duch se pohybují směrem k vchodu do teleportu, nastaví se jeho souřadnice na východ teleportu. Současně se zachová směr pohybu, kterým do teleportu vnikli, a pokračují v pohybu tímto směrem. Tato logika je ve třídě *Pacman* uložena v metodě *teleportPacman* a ve

třídě *Ghost* uložena v metodě *teleportGhost.* Obě tyto metody se nachází před kontrolou kolizí v metodě *wallCollision*.

## 2.4 Pac-Man

### 2.4.1 Vykreslování

Pac-Man byl vykreslen pomocí štětce *ctx* do elementu *<canvas>* prostřednictvím metody *drawPacman* ve třídě *Pacman.* Tato metoda obsahuje vnitřní funkci, která slouží jako šablona pro vykreslení hlavní postavy. Mezi její parametry patří barva ve formě textového řetězce, souřadnice ve dvourozměrném prostoru (x a y), rozměry Pac-Mana (šířka a výška), rotace v radiánech, která byla vyhodnocena pomocí ternárních operátorů v daném řádku na základě aktuálního směru, počáteční a koncový úhel (také v radiánech) a parametr *counterclockwise* určující vykreslování proti směru hodinových ručiček. Tento poslední argument typu *boolean* byl klíčový pro správné vykreslení těla Pac-Mana a jeho charakteristicky otevřené pusy.

Před implementací animace byl Pac-Man vykreslen pouze jako statický objekt, bez jakékoli opakující se animace. Toto vykreslování bylo provedeno pomocí pevně daných koncových úhlů, které vymezovaly část definující tělo a část definující pusu.

Vhodným řešením pro vytvoření efektu otevírání a zavírání pusy bylo využití goniometrické funkce sinus, která osciluje mezi hodnotami –1 a 1. Jelikož bylo zapotřebí funkci nabývající pouze kladných hodnot včetně nuly, byla aplikována absolutní hodnota. Výsledek této funkce v daném okamžiku byl vynásoben číslem 0.003 (určující rychlost animace), následně odečten od čísla 1 a na závěr vynásoben maximálním úhlem, přes který se pusa nemůže otevřít.

### 2.4.2 Pohyb

Ovládání bylo, stejně jako ve většině her, přizpůsobeno čtyřem základním klávesám: W, A, S, D, případně šipkám nahoru, doleva, doprava a dolů. V praxi to bylo implementováno pomocí *document.addEventListene*r, který "naslouchá" eventům na dané stránce. Mezi tyto eventy patří například kliknutí myši, držení myši, ale také stisknutí klávesy, které bylo pro tento úkol nezbytné.

Konkrétní stisknutá klávesa byla uložena do proměnné *key*. Každá klávesa má své specifické číslo, které lze získat pomocí *event.keyCode*. Na základě této hodnoty byl určen směr pohybu a uložen do vlastnosti *desiredDirection* ve třídě *Pacman*. Tato vlastnost sloužila i k opravení většího problému, který bude zanedlouho zmíněn. Na základě tohoto směru se jeho souřadnice x, nebo y zvyšovala, nebo snižovala.

Při vývoji však nastával problém v tom, že když uživatel dané tlačítko držel, tak se Pac-Man nejen pohyboval daným směrem, ale dokázal i zrychlovat. Tato chyba byla opravena zavedením proměnné *keyHeld*, která uchovávala pouze jeden směr. Proměnná funguje jako množina směrů, což znamená, že pokud už v ní daný směr je uložen, nedochází k jeho opakovanému přidávání. Jakmile hráč klávesu uvolní, dojde k odstranění příslušného směru z proměnné *keyHeld*. Tímto způsobem došlo k odstranění přidávání duplicitních záznamů o daném směru, a proto bylo také odstraněno zrychlování Pac-Mana.

Dalším problémem, který se vyskytl v průběhu vypracovávání projektu byla změna směru mezi zdmi. Protože když uživatel stiskl klávesu před křižovatkou, kdy je možné změnit směr, Pac-Man narazil do zdi a přestal se pohybovat. Tato chyba zabrala hodně času k opravení. Řešením byla implementace metody tryNewDirection.

Vyžadovaný směr (*desiredDirection*) je ve výchozím stavu *null*, protože na počátku hry má hráč striktně daný směr doprava, před jeho změnou. Pokud hráč stiskne klávesu, tak hodnota *desiredDirection* se změní. Pokud hráč tedy stiskne klávesu, uloží se do proměnné *previousDirection* současný směr, kterým se Pac-Man pohyboval. Současný směr, kterým se Pac-Man bude pohybovat nadále, bude nahrazen směrem vyžadovaným hráčem a Pac-Man se tímto směrem pohne. Pokud by však došlo ke kolizi se zdí, Pac-Man se zastaví a nastaví svůj současný směr na předchozí. Jinak vrátí požadovaný směr na hodnotu *null*.

Celkový pohyb probíhá v metodě *movement*. Tato metoda spočívá v tom, že Pac-Man zkusí nový směr změněný uživatelem (*tryNewDirection*) a pohne se tím směrem. Dojde-li ke kolizi se zdí, bude nadále pokračovat ve svém původním směru, dokud nebude možné provést změnu.

### 2.4.3 Kolize se zdí

Kolize Pac-Mana se zdmi byla kontrolována pomocí čtyř základních metod, které získávaly souřadnice rohových bodů v dvoudimenzionálním prostoru. Souřadnice *this.posX* a *this.posY* udávají souřadnice středu.

Protože Pac-Man má pevně dané rozměry *this.size.width* a *this.size.height* (šířku a výšku), bylo možné určit souřadnice jeho rohů sčítáním, nebo odečítáním poloviční hodnoty těchto rozměrů od příslušné souřadnice. Konkrétně byla ke každé souřadnici x přičtena nebo odečtena polovina šířky a k souřadnici y polovina výšky.

Protože tato kontrola kolizí se zdí se následně projevila jako příliš striktní, což neumožňovalo Pac-Manovi se pohybovat mezi uličkami, použitím hodnot +1 nebo -1, které sloužily jako menší posun v souřadnicích, byl problém vyřešen.

Samotná kontrola se zdí se prováděla použitím metody *wallCollision*, kde se kontrolovala poloha krajního bodu vůči herní mapě. Pokud tento krajní bod se nacházel ve stejném poli jako zeď, došlo k detekci kolize.

### 2.4.4 Zpracování jídla

Princip sbírání jídla po herní mapě byl v mém projektu implementován podobně jako detekce kolizí se zdí, avšak s použitím odlišných metod. Pro detekci kolize s jídlem byly vytvořeny metody, které získávají souřadnice pusy v závislosti na daném směru. Tyto souřadnice jsou odlišné od těch, které slouží pro detekci kolizí s překážkami, protože pusa se nenachází na krajích Pac-Mana, ale mezi těmito krajními body.

Kontrola kolize s jídlem se neprovádí na všech čtyřech bodech, ale pouze na bodě v aktuálním směru pohybu. V případě, že Pac-Man se ocitne na políčku, kde se nachází volné jídlo, přepíše hodnotu ve dvourozměrném poli z čísla 2 na 0. Toto číslo v mém projektu symbolizuje prázdné místo, po restartování hry, nebo jejím úspěšném dokončení hráč může hru opakovat, tehdy se číslo 0 vrátí zpět na číslo 2. Tímto způsobem se jídlo vrátí zpět do hry.

Jakmile Pac-Man dané jídlo sní, zvýší se mu skóre o 1, v originální hře se skóre navyšuje o 10. Kontrola toho, jestli se Pac-Man nachází na políčku s jídlem probíhá pomocí proměnné *foodEaten*, která je v defaultním stavu *false*. Pokud se Pac-Man dostane na políčko s jídlem, hodnota této proměnné se změní na *true*. Jakmile tato proměnná *foodEaten* nastavena na *true*, zvýší skóre o 1 a vypíše do HTML nadpisu *pacmanScore*. Ihned po provedení této akce se hodnota vrátí zpět na *false*.

### 2.4.5 Speciální schopnosti

Jako v originální hře, tak i v mém projektu Pac-Man disponuje několika speciálními schopnostmi, které mu na krátkou dobu poskytují výhodu oproti duchům. Implementace těchto schopností probíhá obdobně jako zpracování běžného jídla. Liší se pouze číslem dané speciální schopnosti a jídla.

Jedna z nich je ikona ducha, která po jejím sebrání umožní Pac-Manovi po dobu pěti sekund duchy vrátit se zpátky na původní pozici. Tato mechanika je provedena metodou *switchGhostsIntoFrightenedMode* ve třídě *Pacman*. Tato metoda obsahuje vnitřní funkci, která má na starost změnu chování a vizuální podoby všech duchů na mapě. Konkrétně se jedná o metodu *setFrightenedMode ve třídě Ghost*.

Duchové se v tomto režimu pohybují pomaleji a nejsou schopni Pac-Mana ohrozit, což hráči poskytuje příležitost je chytit. Pokud Pac-Man ducha chytí, je tento nepřítel navrácen na svou výchozí pozici ze začátku hry, zároveň mu počínaje tímto úkonem končí možnost jej chytit znovu a může ohrožovat hráče V opačném případě se všechny postavy vracejí na své původní pozice a hra pokračuje.

Další speciální schopností je bonusový život, který hráči přidá jeden život. Tato mechanika může být klíčová, zejména pokud hráč ztratil všechny zbývající pokusy. Poslední speciální schopností je sebrání třešně (*cherry*), která hráči přidává 100 bodů ke skóre. Po ukončení hry je pak celkové skóre vyhodnoceno a zobrazeno hráči.

## 2.5 Duchové

### 2.5.1 Vykreslování

Vykreslování duchů, stejně jako Pac-Mana, bylo realizováno prostřednictvím štětce *ctx* v HTML elementu *<canvas>*. Hlavním rozdílem bylo to, že zatímco Pac-Man

byl vykreslován programově, grafické návrhy duchů byly předem připravené jako samostatné obrázky, což usnadnilo jejich vykreslování ve hře.

Při zobrazování duchů došlo k několika problémům. Jeden z nich se týkal chybového hlášení prohlížeče, že daný obrázek neexistuje, nebo že je v takzvaném *broken state*. První chyba byla způsobena nesprávnou cestou k obrázku vzhledem k souboru obsahujícího typescriptový kód, nebo nesprávné příponě souboru. Druhá chyba týkající se *broken state* obrázku, trvala delší dobu na opravení.

Řešením této chyby bylo zavedení vlastnosti *this.imageLoaded* do abstraktní třídy *GhostTemplate*, od které třída *Ghost* dědí. V defaultním stavu tato vlastnost má hodnotu *false*. Na načtení obrázku, který je uložen ve vlastnosti *this.image,* se tato vlastnost změní na *true,* a dojde k vykreslení obrázku pomocí *ctx.drawImage*. Kontrola načtení stránky se prováděla pomocí *this.image.onload*.

Dalším problémem bylo vykreslování duchů v závislosti na směru jejich pohybu. Ze základu měl každý duch pouze jeden obrázek, který se vykresloval nezávisle na tom, v jakém směru se pohybuje. Všechny cesty obrázků pro duchy byly uloženy v jednorozměrném poli *this.imagePaths*. Toto pole však muselo být rozšířeno do dvourozměrného pole.

Jednorozměrných polí v *this.imagePaths* bylo nyní pět. Čtyři slouží pro jednotlivé duchy, a to poslední páté pole je obrázek, který slouží pro režim, kdy Pac-Man může duchy chytat. Každý duch má svou hodnotu *this.imageIndex*, která získává obrázky z pole *this.imagePaths*.

Problémem nyní však byl druhý index, který se mění v závislosti na směru. Tento problém byl vyřešen použitím nové vlastnosti *this.imageIndexDirection*, který posuzuje jeho hodnotu pro jednotlivého ducha na základě aktuálního směru. Toto

posuzování bylo implementováno ve třídě *Ghost* vytvořením metody *switchImageByDirection*, kde se vyhodnocuje druhý index na základě směru.

Pokud se duch nenachází v režimu zranitelnosti, tak se tento index posuzuje na základě směru. Pokud se v něm ocitá, má fixně nastavený druhý index, který získává obrázek zranitelného ducha, který se vyskytuje i v originální hře.

### 2.5.2 Souřadnice

Stejně jako u získávání informací o mapě ze souboru JSON, i souřadnice jednotlivých duchů jsou uloženy v JSON souboru. Z počátku vývoje, kdy hra obsahovala pouze jednu mapu, měl každý duch pouze jednu počáteční pozici, přidáním dalších map musely být tyto počáteční souřadnice přidány v závislosti na dané mapě. Na tyto počáteční souřadnice se duchové vrací, chytí-li je duch v režimu zranitelnosti.

Pro načítání těchto informací byla vytvořena asynchronní funkce *loadGhostPositions*. Její logika je obdobná jako při načítání mapy, liší se pouze v souboru, ze kterého jsou data získávána. Data ze souboru *ghostPositions.json* jsou převedena do formátu JSON a následně je podle aktuální hráčovy úrovně *pacman.currentLevel* zvolena odpovídající sada souřadnic. Konkrétně se používá index *pacman.currentLevel - 1*, aby bylo možné správně přistupovat k prvku v poli.

Při zpracování objektu JSON byl vybrán klíč odpovídající názvu ducha (*blinky*, *pinky*, *inky*, *clyde*), který obsahuje informace o jeho souřadnicích. Poté jsou z daného objektu získány hodnoty *posX* a *posY*, které určují umístění ducha na mapě. Tyto souřadnice jsou následně vynásobeny šířkou, nebo výškou jednoho bloku mapy, čímž se převedou do skutečných souřadnic na mapě. Pro zajištění správného formátu čísel je výsledek konvertován pomocí třídy *Number*. Tímto způsobem došlo k snazší rozšiřitelnosti hry bez nutnosti větších úprav kódu.

### 2.5.3 Algoritmus pohybu

V originální hře má každý duch odlišnou logiku pohybu. V tomto projektu však všichni duchové sdílejí stejný pohybový algoritmus. Pro správnou implementaci bylo nutné vytvořit funkci, která efektivně prochází grafy a hledá optimální cestu k cíli. K tomuto účelu byl zvolen A* algoritmus, jenž patří mezi nejpoužívanější metody pro hledání nejkratší cesty.

A* algoritmus úzce souvisí s Dijkstrovým algoritmem, který se využívá například v routovacím protokolu OSPF v oblasti počítačových sítí. Na rozdíl od Dijkstrova algoritmu, A* pracuje s heuristickou funkcí, která odhaduje zbývající vzdálenost mezi aktuálním a cílovým uzlem. Díky této vlastnosti je A* algoritmus rychlejší, efektivnější a flexibilnější. Jeho logiku v tomto projektu zajišťuje metoda *findPathToPacman* ve třídě *Ghost.*

Před samotnou implementací bylo nutné vytvořit metodu *buildGraph*, která slouží jako vstupní graf pro A* algoritmus. Tato metoda prochází celou mapu pomocí dvou cyklů. Pokud se na určitém poli nachází číslo 1, signalizující zeď, přiřadí se tomuto poli nedosažitelná vzdálenost (*Infinity*), čímž se zajistí, že duchové nebudou moci tímto místem procházet. V opačném případě je poli přiřazena metrika 1, označující vzdálenost 1 od cíle. Tyto hodnoty se následně uloží do dvourozměrného pole *graph*, které metoda *buildGraph* vrací.

Dalším krokem bylo vytvoření metody pro výpočet heuristiky. V tomto projektu byla zvolena Manhattanova vzdálenost, která je ideální pro pohyb ve čtvercové mřížce se čtyřmi možnými směry. Heuristická funkce vypočítává součet rozdílů souřadnic dvou bodů v absolutní hodnotě.

Implementace A* algoritmu probíhá následovně. Nejprve jsou definovány startovní pozice ducha a cílové souřadnice pozice Pac-Mana na mapě. Pomocí

metody *buildGraph* se získá graf, který se uloží do konstanty *graph*. Dále jsou vytvořeny dvě důležité datové struktury: *openList*, obsahující uzly, které duch již prozkoumal a *closedList*, obsahující cesty, které byly definitivně vyhodnoceny. Konstanta *cameFrom* uchovává pozice ducha, ze kterých přišel. Konstanty *gScore* a *fScore* slouží k uchování vzdálenostní hodnoty. *gScore* ukládá skutečnou vzdálenost od startu, zatímco *fScore* představuje odhadovanou vzdálenost k cíli na základě Manhattanovy heuristiky.

Po inicializaci je startovní uzel nastaven na hodnotu 0, protože představuje výchozí bod. *fScore* startovního uzlu se rovná odhadované vzdálenosti k Pac-Manovi. Algoritmus poté pracuje v cyklu, dokud se v *openList* nacházejí uzly k prozkoumání. V každé iteraci se odebere uzel s nejnižší hodnotou *fScore* a nastaví se jako aktuální. *openList* je následně seřazen podle výhodnosti cest.

Pokud dojde k neúspěšnému pokusu o získání hodnoty *fScore* pro určité souřadnice, použije se operátor *??* (*nullish coalescing operator*), který v případě hodnoty *null* nebo *undefined* nastaví výsledek jako nedosažitelný (*Infinity*). Pokud jsou dvě cesty stejně výhodné, upřednostní se uzel v aktuálním směru pohybu ducha.

Jestliže duch není ve *frightened* módu a dojde ke kolizi s Pac-Manem, začne se zpětně rekonstruovat cesta pomocí pole *path*, které uchovává jednotlivé kroky. V opačném případě se aktuální uzel přidá do *closedList*, čímž se označí jako již prozkoumaný. Poté se vyhodnotí sousední uzly. Pokud nový uzel není mimo mapu, není neprůchozí a dosud nebyl prozkoumán, vypočítá se jeho dočasná hodnota *gScore*. Jestliže je nově nalezená cesta kratší než dříve uložená, dojde k aktualizaci hodnot a nový uzel se přidá do *openList*.

Pokud se duch nachází ve *frightened* módu, algoritmus prochází všechny uzly v *closedList* a hledá ten nejvzdálenější od Pac-Mana. Jakmile je nalezen, uloží se cesta

k tomuto bodu. Nakonec algoritmus vrátí nalezenou cestu nebo hodnotu *null*, pokud žádná cesta neexistuje.

Jednalo se o příliš obtížný úkol. Původní koncept pohybu duchů byl myšlen tak, že budou měnit směry v závislosti na Pac-Manově poloze. Například pokud by se Pac-Man vyskytoval vpravo, tak by změnili směr doprava. Avšak toto řešení nefungovalo, poněvadž duchové nebrali v potaz zeď, která jim bránila v tomto směru a neustále do zdi naráželi a vraceli se zpět, což neumožňovalo souvislé pronásledování. Zároveň tímto řešením nedokázali hledat alternativní průchody, které by jim umožňovaly se k Pac-Manovi dostat. Implementace tohoto algoritmu byla velmi náročná, mnohokrát duch nedělal to, co by měl, a někdy se dokonce nedokázal pohybovat vůbec, proto byl pro můj projekt částečně vytvořen za pomoci umělé inteligence.[8]

### 2.5.4 Pohyb

Po zavedení A* algoritmu do projektu bylo nezbytné zpracovat samotný pohyb. Metoda *moveTowards* získá cestu z metody *findPathToPacman*, která je uložena v poli *path*. Pokud pole není prázdné, nastaví další krok pohybu indexem 1, protože na indexu 0 je pozice samotného ducha. Metoda následně vypočítá cílové souřadnice (*targetX* a *targetY*), které odpovídají skutečným souřadnicím, nikoliv souřadnicím v *grid* systému. Zároveň jsou zohledněny rozměry Pac-Mana.

Dalším krokem je výpočet rozdílu mezi souřadnicemi Pac-Mana a ducha. Na základě tohoto rozdílu duch určuje směr pohybu pomocí metody *moveGhost*. Tato metoda upraví souřadnice ducha podle směru, kterým se pohybuje. Současně si duch uchovává předchozí krok, což mu umožňuje vrátit se zpět, pokud narazí do zdi. Pokud

[8] *ChatGPT [online]. 3. 4. 2025. Dostupné z: https://chat.openai.com.*

je duch vzdálen více než pět bloků od Pac-Mana, náhodně změní směr pomocí metody *randomDirection*.

Tato metoda využívá pomocnou metodu *getAvailableDirections*, která kontroluje, zda se v sousedním poli nenachází zeď. Pokud zeď není přítomna, daný směr se uloží do pole *ghostDirections*, které se následně vrátí. Dále metoda kontroluje, zda duch není mimo mapu. Metoda *randomDirection* získá směry z *getAvailableDirections* a filtruje je za účelem minimalizace opakovaného pohybu ze strany na stranu. Pokud je možné změnit směr, duch upřednostní jiný směr než opačný. Pokud není jiná možnost, duch se může pohybovat i opačně.

Celkový pohyb je řízen metodou *ghostMovement, kdy se nejprve* kontroluje, zda je duch zarovnaný v mřížce (*alignedX* a *alignedY*). Pokud je duch ve vzdálenosti menší nebo rovné pěti blokům od Pac-Mana, zavolá se metoda *moveTowards*. Pokud je duch dále, náhodně mění směr pomocí metody *randomDirection*. Nakonec se pomocí metody *moveGhost* provede samotný pohyb ducha, a zároveň se v metodě ghostMovement zpracovává kolize s Pac-Manem pomocí metody *handleCollisionWithPacman*, která je podrobněji vysvětlena v kapitole 2.5.6.

### 2.5.5 Režimy

Režimy Pac-Manovi značí aktuální chování ducha ve hře. V původní verzi hry existují tři hlavní režimy: *chase*, *scatte*r a *frightened*. Během režimu *chase* duchové aktivně pronásledují Pac-Mana po mapě a snaží se jej chytit. V režimu *scatter* se každý duch přesouvá do svého předem určeného rohu mapy. *Frightened* režim nastává ve chvíli, kdy Pac-Man sebere speciální schopnost – v této implementaci je tento stav na mapě znázorněn ikonou bílého ducha.

V rámci tohoto projektu byly implementovány pouze dva režimy – *chase* a *frightened*. Logika režimu *chase* spočívá v tom, že pokud se duch nachází ve

vzdálenosti pěti bloků od Pac-Mana, začne jej aktivně pronásledovat pomocí A* algoritmu popsaného v předchozí kapitole. Pokud je vzdálenost větší, duch se pohybuje náhodným směrem. Ověření vzdálenosti zajišťuje metoda *isPacmanNear* ve třídě *Ghost*, která počítá vzdálenost mezi dvěma body v rovině pomocí vzorce.

V režimu *frightened* duch hledá na mapě nejvzdálenější možný bod od aktuální pozice Pac-Mana a snaží se k němu po dobu pěti vteřin dostat. V ten moment opačně mění směr. Ovšem 1,25 sekundy před koncem se promění v bílého ducha, čímž dávají hráči znamení, že se brzy vrátí do *chase* režimu. Přepnutí režimů probíhá pomocí metod *setChaseMode* a *setFrightenedMode*. Metoda *setChaseMode* nejprve ověří, zda duch již není v režimu *chase*, a pokud ano, nic nemění. Totéž platí i pro metodu *setFrightenedMode*.

Každý duch má přiřazený svůj *imageIndex*, který určuje správné vykreslení příslušného obrázku. Přepnutí režimu aktualizuje také hodnotu *this.mode* na *chase*, čímž se duchové po skončení *frightened* režimu vrátí do běžného chování. Zároveň se obnoví jejich rychlost, která byla v režimu *frightened* snížena. Tato změna probíhá v metodě *setFrightenedMode*, která nastaví *this.imageIndex* na hodnotu 4, odpovídající obrázku vystrašeného ducha, a *this.imageIndexDirection* na 0, protože tento obrázek nemění vzhled podle směru pohybu. Navíc nastaví režim *this.mode* na *frightened*, sníží vzdálenost (*this.distance*) pohybu na hodnotu 1 a spustí časovač, který po pěti vteřinách opět přepne režim na *chase* prostřednictvím metody *setChaseMode*.

Speciální schopnosti, které aktivují režim *frightened*, byly rozmístěny tak, aby Pac-Man nemohl v průběhu pěti vteřin sebrat další a prodloužit tím dobu trvání tohoto režimu. Díky tomuto řešení nebylo potřeba implementovat složitější logiku pro skládání efektů více schopností.

### 2.5.6 Kolize s Pac-Manem

Stejně jako Pac-Man řeší kolize se zdí, tak duch řeší kolize s Pac-Manem. Tyto kolize jsou důležité pro vyhodnocování toho, zda duch Pac-Mana chytil.

Kolize ducha s Pac-Manem záleží především na současném režimu daného ducha. Nachází-li se v režimu *frightened*, Pac-Man jej chytil a duch se vrací zpět na své výchozí souřadnice. Pokud je však duch v režimu chase, sebere Pac-Manovi život a nejen duchové, ale i Pac-Man se vrací zpět na své počáteční souřadnice.

Tuto logiku řeší metoda *isPacmanCatched*, která kontroluje, zda se Pac-Man s duchem nepřekrývají na základě jejich souřadnic. Pokud se překrývají, metoda vrátí *true*, jinak vrátí *false*.

Samotné zpracování kolizí s Pac-Manem probíhá v metodě *handleCollisionWithPacman*, která vyhodnocuje dané chycení. Je-li duch ve *frightened* modu, hráči se přičte 200 bodů a daný duch se vrátí zpět na svoje počáteční souřadnice. Pokud však je duch v chase modu, hráč ztrácí život a všechny herní postavy se vrací na své počáteční souřadnice zpět.

Při programování kolizí s Pac-Manem došlo snad k jedinému problému, že byla detekována kolize, přestože duch a Pac-Man byli od sebe vzdálení 1 blok, tato komplikace byla odstraněna opravením metod získávajících souřadnice rohových bodů pomocí čísla sloužícího jako menší posun.

## 2.6 Hudba

Hudební doprovod použitý v projektu byl vygenerován prostřednictvím editoru Suno. Před použitím daného editoru bylo třeba vytvořit si účet pro získání 50

kreditů zdarma, které slouží k vytváření vlastních písní. Tyto kredity se dennodenně automaticky vracejí.

Po úspěšném přihlášení byl vytvořen *workspace* s libovolným názvem. V textovém poli nacházejícím se možnosti nabídky žánru písně, která byla vytvořena, byly nejčastěji používány následující žánry: *Latin Pop*, *EDM*, *House*, *Reggaeton*, *Dance hall*, *Tropical house*, *Trap*, *Pop*.

Všechny písně byly vytvořeny pomocí verze 3.5, která disponuje lepší akustikou a délkou písní maximálně 4 minuty. Po vytvoření několika desítek písní bylo vybráno 5 nejlepších dle vlastního uvážení.

Pokud hráč nedohraje úroveň do té doby, než skončí audio v pozadí, nic se neděje, protože toto audio je v nekonečné smyčce. Každá cesta k souboru byla uložena do pole *musicArray* a ve funkci *playMusic* byla vybrán hudební doprovod pro příslušnou úroveň. Tato funkce se volá při začátku nové úrovně nebo při restartu celé hry. Pokud hráč nedohraje úroveň do té doby, než skončí audio v pozadí, nic se neděje, protože toto audio je v nekonečné smyčce.

## 2.7 Grafické návrhy

Grafické návrhy použité v projektu byly vytvořeny prostřednictvím webového editoru Piskel. Na začátku vytváření vlastních návrhů, Piskel vytvoří plátno v základním rozlišení 32x32 pixelů. Na levé straně tohoto editoru se paleta několika nástrojů pro malování.

Pro vytvoření návrhu duchů a ikon představující speciální schopnosti byl převážně použit nástroj *pen tool*, kterým byl manuálně namalován pixel po pixelu.

Zpočátku byly návrhy namalovány v rozlišení 14x14 pixelů pro snazší malování, ale díky možnosti nastavení rozměrů obrázku pro export byly zvoleny rozměry 195x195. Zároveň lze při exportu zvolit také formát souboru, například *.png* nebo *.jpg* podle potřeby projektu. Obrázek na pozadí menu byl jediný grafický návrh stažený z internetu.[9]

## 2.8 Úvodní menu a instrukce

### 2.8.1 Font

Pac-Man je videohra, která vyžaduje styl písma z doby arkádových automatů. Pro tento účel byl zvolen font *PublicPixel*, který je navržen pro pixel art projekty a retro hry. Tento font je dostupný pod licencí *CC0*, což znamená, že je zdarma a může být volně používán, upravován a distribuován a díky těmto benefitům je ideálním řešením pro školní projekty, kde je důležité dodržet autorská práva.[10]

### 2.8.2 Design

Grafické zpracování herního menu společně s instrukcemi bylo realizováno s využitím technologií HTML a CSS. Samotná struktura menu byla vytvořena pomocí HTML a vizuální podoba jednotlivých prvků byla vytvořena prostřednictvím kaskádových stylů CSS.

Zobrazování a skrývání menu po kliknutí na příslušné tlačítko bylo implementováno prostřednictvím programovacího jazyka TypeScript. Pozadí menu je

[9] *PAC-MAN [online]. 2022 [cit. 2025-04-05]. Dostupné z: https://www.alza.cz/gaming/pac-man-historie-hry*

[10] *OPEN GAME ART. Public Pixel Font [online]. 29. března 2022 [cit. 2025-04-06]. Dostupné z: https://opengameart.org/content/public-pixel-font*

tvořeno obrázkem, který pokrývá celou plochu stránky a vytváří dojem tematického prostředí.

Textové prvky na úvodním menu jsou zvýrazněny zlatou barvou, která v kombinaci s animovaným stínem dodává menu výrazný a dynamický vzhled. Tlačítka jednotlivých možností menu jsou stylizována pomocí poloprůhledného tmavého pozadí a rámečku stejné zlaté barvy, čímž vizuálně zapadají do zbytku rozhraní.

Instrukce pro uživatele byly vytvořeny jako element překrývající menu po stisknutí tlačítka *instructions*. Tento element na levé straně obsahuje v angličtině seznámení uživatele se hrou a na pravé straně se nachází ukázka samotné hry.

### 2.8.3 Responzivita

Protože tento projekt není realizován pouze na počítače, bylo zapotřebí jej přizpůsobit i rozměrům pro tablet a mobilní zařízení. V oblasti kaskádových stylů CSS na tohle existují *media queries*, které umožňují definovat specifická pravidla na základě šířky zobrazovacího zařízení.

Konkrétně byly použity rozměry šířky 576, 768 a 1024 pixelů, což jsou nejčastější body zlomu při vývoji responzivních webů. V těchto bodech zlomu byly poupraveny vlastnosti vybraných HTML prvků, například zmenšení velikosti písma, úprava šířky elementu <*div*>, nebo ke změně vnitřního uspořádání jednotlivých prvků tak, aby se obsah lépe přizpůsobil menší obrazovce.

# 3 Závěr

V průběhu vývoje tohoto projektu se naskytlo mnoho komplikací. Většina z nich byla opravena během krátké chvíle, některé však zabraly spoustu času. Nejobtížnější byla implementace chování duchů, jež byla vyřešena zavedením algoritmu procházejícího grafem. Při návrhu této logiky bylo nutné zohlednit vzdálenost ducha vůči hráči.

Na práci bylo projeveno úsilí pracovat pravidelně, což se s menšími přestávkami dařilo. Významná část byla věnována testování, optimalizaci a odstraňování chyb, což napomohlo k lepšímu pochopení daného programovacího jazyka.

Tento projekt zároveň představoval skvělou příležitost implementovat znalosti získané za tři roky studia. Práce na projektu pomohla k prohloubení dovedností v oblasti programování, algoritmizace, ale i práce s grafickými editory. Získané zkušenosti jsou považovány za cenný základ pro další studium nebo profesní růst v oblasti informačních technologií.

Poděkování patří mému vedoucímu Janu Tillovi za průběžnou konzultaci se zpracováním nejen praktické, ale i písemné části. Jeho rady a zpětná vazba přispěly ke zlepšení kvality výsledného projektu.

# 4 Přílohy

## 4.1 Seznam obrázků



## 4.2 Zdroje

TypeScript [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-25]. Dostupné z: https://cs.wikipedia.org/wiki/TypeScript.

HTML [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-25]. Dostupné z: https://en.wikipedia.org/wiki/HTML.

CSS [online]. Britannica, 2025 [cit. 2025-03-25]. Dostupné z: https://www.britannica.com/technology/CSS-programming-language.

Visual Studio Code [online]. Microsoft, [2015] [cit. 2025-03-25]. Dostupné z: https://visualstudio.microsoft.com/cs/#vscode-section.

Piskel [online]. GitHub, 2008 [cit. 2025-03-27]. Dostupné z: https://github.com/piskelapp/piskel.

Suno [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2025-03-27]. Dostupné z: https://en.wikipedia.org/wiki/Suno_AI.

JSON [online]. Oracle, 2024 [cit. 2025-03-30]. Dostupné z: https://www.oracle.com/database/what-is-json/.

ChatGPT [online]. 3. 4. 2025. Dostupné z: https://chat.openai.com.

PAC-MAN [online]. 2022 [cit. 2025-04-05]. Dostupné z: https://www.alza.cz/gaming/pac-man-historie-hry.

OPEN GAME ART**.** Public Pixel Font [online]. 2022 [cit. 2025-04-06]. Dostupné z: https://opengameart.org/content/public-pixel-font.