

# Interaktivní web

Textová hra za použití webových nástrojů:  
HTML, CSS, JavaScriptu

**Vojtěch Lančarič**



Střední průmyslová škola a Gymnázium Na Třebešíně

MATURITNÍ PRÁCE S OBHAJOBOU

Březen 2023

Vedoucí: Mgr. Jan Souhrada

## **Poděkování**

Děkuji vedoucímu Mgr. Janu Souhradovi za vedení, pomoc a cenné rady při realizaci Maturitní práce.

## **Čestné prohlášení**

Čestně prohlašuji, že jsem tuto práci vypracoval samostatně s použitím literatury a zdrojů uvedených v seznamu použité literatury.

## **Abstrakt**

Tato práce se zabývá návrhem a realizací textové hry napsané za použití webových nástrojů - HTML, CSS a JavaScriptu. Hra byla inspirována zkázkou raketoplánu Columbia, která se odehrála v roce 2003. HTML a CSS byly použity pro grafický vzhled statického webového dokumentu, zatímco JavaScript je použit jako hlavní programovací jazyk. Zpracovává příkazy od uživatele a jejich základě dynamicky upravuje dokument. Dále se v teoretické části pojednává verzovacích systémech a značkovacím jazyku  $\text{\LaTeX}$ , neboť byly využity při tvorbě práce. Byla vytvořena textová hra, odehrávající se na palubě fiktivní rakety Shumaker-Levi 9. Hra se odehrává ve 4 místnostech a je koncipována do jedné hodiny hraní.

## **Klíčová slova**

HTML, CSS, JavaScript, textová hra

## **Anotation**

In this paper we will discuss implementation of text adventure game. The game is inspired by space shuttle Columbia disaster, which occurred in 2003. The paper also focuses on software technologies description used for development. There are HTML and CSS for static webpage content and its design. Then there is a JavaScript handling user's requests and dynamically modifies the webpage. Theoretical section also covers another software technologies such as version control systems and  $\text{\LaTeX}$  and also introduces the concept of object-oriented programming. An alternative to JavaScript as the main programming language for the text adventure is mentioned in section TADS3.

The text adventure, which takes place on board the fictional rocket Shumaker-Levi 9, has been developed. The game includes 4 rooms and is designed to be completed in less than an hour.

## **Keywords**

HTML, CSS, JavaScript, texture adventure

# Obsah

<b>1</b>	<b>Úvod</b>	<b>4</b>
1.1	Cíle . . . . .	4
<b>2</b>	<b>Teoretická část</b>	<b>5</b>
2.1	Textové hry . . . . .	5
2.1.1	Textové hry vs. adventury . . . . .	5
2.1.2	Historie . . . . .	5
2.1.3	České textové hry . . . . .	5
2.2	HTML . . . . .	6
2.2.1	Historie . . . . .	6
2.2.2	Specifikace . . . . .	6
2.3	CSS . . . . .	7
2.3.1	Specifikace . . . . .	7
2.4	JavaScript . . . . .	9
2.4.1	Vlastnosti . . . . .	9
2.4.2	Historie . . . . .	10
2.5	Objektově orientované programování . . . . .	11
2.6	Verzovací systémy . . . . .	12
2.6.1	Centralizované systémy . . . . .	12
2.6.2	Distribuované systémy . . . . .	12
2.6.3	Git . . . . .	12
2.6.4	Vlastnosti . . . . .	12
2.6.5	Použití v projektu . . . . .	13
2.6.6	Github . . . . .	13
2.7	TADS 3 . . . . .	13
2.8	L <sup>A</sup> T <sub>E</sub> X . . . . .	14
2.8.1	Historie . . . . .	14
2.8.2	WYSIWYG editory . . . . .	14
2.8.3	Příklady . . . . .	14
2.8.4	Použití v projektu . . . . .	15
<b>3</b>	<b>Praktická část</b>	<b>16</b>
3.1	Námět . . . . .	16
3.2	Engine . . . . .	16
3.3	Parsování vstupu . . . . .	17
3.4	Příkazy . . . . .	17
3.5	Děj . . . . .	18
<b>4</b>	<b>Závěr</b>	<b>19</b>

# 1 Úvod

Textové hry jsou jedním z prvních odvětví počítačových her, které se začalo rozvíjet s rozšířením osobních počítačů v sedmdesátých letech minulého století. Jedím z důvodů takto brzkého rozmachu jsou nízké nároky na výpočetní výkon a relativní nenáročnost vývoje oproti ostatním počítačovým hrám, videohram. Proto byl jejich provoz možný i na velmi málo výkonných sálových počítačích bez videokonzole. Později na osobních počítačích, již s videokonzolemi. [4] Od té doby je téměř kompletně nahradilo odvětví videoher. Přesto se však v dnešní době věnuje vývoji textových her řada nadšenců, primárně na nekomerční úrovni, kteří spoléhají na hlavní kouzlo textových her, tedy lidskou představivost. [5]

Pro toto téma jsem se rozhodl, protože se zajímám o informatiku a programování, a chtěl jsem se blíže seznámit a zdokonalit ve vývoji webových stránek. Jejich podrobná znalost je při téměř jakémkoli komerčním vývoji softwaru esenciální.

## 1.1 Cíle

Cílem této práce je vytvořit textovou hru, provozovanou ve webovém prohlížeči, s využitím dnes nejčastěji používaných webových technologií. Zejména se jedná o HTML, CSS, JavaScript, jejichž podrobnému popisu budou věnovány následující kapitoly. K realizaci však budou nepřímo využity i jiné nástroje, jako verzovací program git a  $\text{\LaTeX}$ , kterým se také budeme věnovat.

## 2 Teoretická část

Na následujících stranách budou popsány jednotlivé technologie, které byly použity pro vývoj projektu. Teoretickou část lze rozdělit na popis technologií, které byly použity přímo pro vývoj hry, tedy [HTML](#), [CSS](#), [JavaScript](#), a technologií, které byly použity pro tvorbu dokumentace - [L<sup>A</sup>T<sub>E</sub>X](#). Dále jsou do teoretické části zařazeny kapitoly [Objektově orientované programování](#), [Verzovací systémy](#), [TADS 3](#) a [Textové hry](#), neboť jsou s tématem či prací také spjatý.

### 2.1 Textové hry

Textová hra je počítačová hra, která pro interakci s uživatelem používá text, přesněji řečeno množinu standardně kódovaných znaků, historicky například pomocí ascií tabulky. Jiným typem počítačových her, dnes převažujících, jsou videohry, které naopak využívají vektorovou nebo bitmapovou grafiku. [5] Alternativně lze v žánru textových her doplnit text obrázky, zvuky, případně jinými vjemy. Charakterizující prvek textových her ale zůstává slovní popis herního prostředí. V některých případech, tzv. grafických textových her, rozhoduje hráč o vývoji hry kliknutím na jedno z nabízených tlačítek.[7]

#### 2.1.1 Textové hry vs. adventury

Základním znakem adventur je, že se hráč snaží splnit úkoly, které mu hra zadá. Není podmínkou textový režim - existuje velké množství grafických adventur. [6]

#### 2.1.2 Historie

První zdokumentované textové hry se začaly objevovat v 60. letech minulého století. Byly provozovány na sálových počítačích, tzv. mainframech s přídatným výstupem realizovaným pomocí dálnopisu, který tiskl výstup hry na papír. V 60. letech sice již existovaly videoterminály, ale jejich větší rozšíření bylo omezeno vysokou cenou. [4, 5]

Příklady takových her jsou The Oregon Trail (1971), Lunar Lander (1969), Dungeon (1977). [4]

V sedmdesátých letech videoterminály nahradily dálnopisy v rozhraní sálových počítačů (mainframů). Stalo se tak díky zlevňování technologií. Dalšímu rozvoji textových her napomohl rozmach osobních počítačů. Většina her té doby byla inspirována hrou Dungeons & Dragons nebo světem v dílech J.R.R. Tolkienu. [1]

#### 2.1.3 České textové hry

Jedním z nejznámějších českých vývojářů textových her a adventur je František Fuka, člen programátorské skupiny Golden Triangle, která vyvinula množství textových her. Fuka je spoluautorem například her Belegost, Podraz 3, Indiana Jones. [3]

Textovým hrám se věnuje množství webových stránek. Archivem českých a slovenských textových her je například webová stránka [textovky.cz](http://textovky.cz), kde je možné zahrát si historické textové hry a adventury, převážně českých a slovenských autorů. [2]

## 2.2 HTML

Hypertext Markup Language je značkovací jazyk určený pro zobrazení webovým prohlížečem, používaný pro tvorbu statických webových stránek. Spolu s CSS a JavaScriptem patří k základním technologiím používaným při vývoji webových stránek. Tak jako i jiné značkovací jazyky, je i HTML dokument formátován značkami, tzv. tagy, které předdefinovaným způsobem formátují dokument do výsledné podoby. Tagy jsou uzavírány do špičatých závorek `< p >`. [9] V současnosti k tomu slouží kolem 110 formátovacích tagů. Neurčitost je dána rozdílnými způsoby počítání, k nimž se dostaneme v podkapitole Specifikace. [10]

### 2.2.1 Historie

Autorem jazyka je Tim Berners-Lee, který jej vyvinul za účelem zjednodušení výroby vědeckých dokumentů, které byly psány v jazycích TeX, PostScript a SGML (Standard Generalized Markup Language). SGML se stal pro HTML předlohou. Samotné HTML publikoval Tim Berners-Lee v roce 1990, spolu s webovým protokolem HTTP (hyper text transfer protocol), sloužícím k přenosu HTML stránky po internetu. 26. února 1991 představil Berners-Lee také první webový prohlížeč, který nazval WorldWideWeb. Verze HTML 0.9 - 1.2 vznikly v letech 1991 - 1993, avšak nepodporovaly grafické rozhraní - bylo možné zobrazovat pouze standardní znaky. Dále byla v listopadu 1995 vydána verze 2.0, ve které byla přidána podpora grafiky a interaktivních formulářů. Následovaly verze 3.0 v roce 1997, 4.0 v roce 1999. Zatím poslední verze jazyka - HTML 5.3, byla vydána 22. prosince 2022. HTML je tedy stále aktivně vyvíjeno. [9]

### 2.2.2 Specifikace

HTML je strukturováno množinou tagů, které mohou mít vlastnosti (atributy). Tag se skládá z názvu tagu, který je uzavřen mezi špičatými závorkami. Tagy jsou obvykle párové, přičemž počáteční značka se shoduje s koncovou. Koncová má navíc před názvem lomítko. Tag ovlivňuje veškerý obsah mezi značkami, což mohou být i další tagy. Atributy tagu se zapisují za název první značky a dále určují vlastnosti tagu. V ukázce níže je tag, který slouží pro vytvoření odkazu. Jeho atributy jsou v tomto případě href, který udává, kam se má uživatel dostat po kliknutí na odkaz, a target, jenž definuje, že se má odkaz otevřít v novém okně prohlížeče. [9]

```
<a href="https://discord.com/" target="_blank">
| Po kliknutí sem se dostanete na Discord
</a>
```

Obrázek 1: Otevření odkazu na nové kartě

Nepárové tagy mají pouze jednu značku - postrádají koncovou, a nemají žádný obsah. Příkladem buď tag img, jenž do dokumentu přidá obrázek "obrazek.jpg".

```

```

Obrázek 2: Tag img

Počet HTML tagů se v jednotlivých verzích liší. HTML 1.0 definovalo 22 značek. Jejich počet se v 3.2 zvýšil na 70, a dále v HTML 5.2 je jich definováno 111. Počítáme-li k nim i tagy XHTML specifikace, dostaneme 133. [11]

## 2.3 CSS

CSS (zkratka z anglického Cascading Style Sheet, v češtině kaskádové styly) je jazyk definující způsob zobrazení elementů na HTML, XHTML nebo XML stránce. Hlavní myšlenkou je oddělit obsah a strukturu dokumentu od jeho vizuální podoby, jako barvy, rozložení a fontů. Toto oddělení také umožňuje použít stejný vizuální vzhled pro více HTML stránek. Autorem myšlenky oddělit obsah od vzhledu je norský programátor Håkon Wium Lie. Nyní je publikovaný pod organizací W3C.

### 2.3.1 Specifikace

*„Definice kaskádových stylů sestává z několika pravidel. Každé pravidlo obsahuje selektor a blok deklarací. Každý blok deklarací pak obsahuje deklarace oddělené středníky; a každá deklarace sestává z identifikátoru vlastnosti, následuje dvojtečka : a hodnota vlastnosti. Nepovinně ještě může následovat označení !important, které zvýší sílu deklarace.” [12]*

#### Selektory

Pravidla jsou na HTML elementy uplatňována podle selektorů, a základní můžeme rozdělit následujícím způsobem.

- Selektor pro elementy jednoho typu. Například pro nadpis třetí úrovně `< h3 >`. Zahrneme-li toto pravidlo do HTML, všechny nadpisy 3. úrovně budou modrou

```
h3{
  color: blue;
  size: 10px;
  text-decoration: underline;
}
```

Obrázek 3: Selektor tagu h3

barvou, velikosti 10 pixelů a podtržené.

- Elementy s konkrétním atributem - názvem třídy nebo id.
  - Podle atributu id, které musí být unikátní v celém dokumentu. Pravidla s tímto selektorem se použijí pro nejvýše jeden element a zapisují se *idname*.
  - Podle atributu class. Ten nemusí být unikátní. Zapisují se *.classname*.

```
.jmeno_tridy, #nazev_id{
  font-style: italic;
  size: 1cm;
}
```

Obrázek 4: Selektor třídy a id

Pravidla mohou mít více selektorů oddělených čárkou. Pro všechny elementy s atributem *class = "jmenotridy"* a unikátní element s atributem *id = "nazev\_id"*, bude text psán kurzivou o velikost 1 cm.



- Elementy v závislosti na jejich umístění v hierarchii dokumentu. Například pravidla se selektorem *body > div* budou aplikována pouze na elementy *div*, které jsou přímým potomkem elementu *body*. Neplatil by tedy pro *div* uzavřený v elementu *p*, uzavřeném v elementu *body*.

Výše vyjmenované sektory však nejsou kompletní. Existuje množství dalších selektorů. [13]

## Zahrnutí do HTML

Existují tři možnosti, kde zapisovat CSS pravidla, aby byla zahrnuta do HTML.

### Inline zápis

Tento zápis je připsán ke konkrétnímu elementu vedle zápisu jeho atributů. Tento zápis se

```
<div style="border-style: dotted; border-radius: 5px;">
  Tento text bude ohraničen tečkovanou hranicí.
  Rohy budou zakulacený.
</div>
```

Obrázek 5: Inline zápis css stylů

nedoporučuje používat, jelikož potlačuje výhody CSS - oddělení obsahu a vizuální podoby.

### Zápis v hlavičce

CSS pravidla jsou zapsána v hlavičce HTML dokumentu do tagu `< head >`. Nevýhodou je aplikace pravidel pouze na jeden konkrétní dokument.

```
<style type="text/css">
  .class_name{
    border-radius: 2cm;
    border-color: aquamarine;
    border-style: dashed;
  }
</style>
```

Obrázek 6: Zápis css v hlavičce html

### Externí CSS soubor

Nečastěji používaným způsobem je deklarace pravidel v externím souboru. Ten může být s HTML dokumentem provázán tagem `< link >`. Atribut *href* udává relativní cestu

```
<head>
  <link rel='stylesheet' href='kaskadove_styly.css' type='text/css'>
</head>
```

Obrázek 7: Načtení css do html z externího souboru

k souboru, atribut *type* říká prohlížeči, aby daný soubor interpretoval jako CSS. Další možností je připojení externího souboru pomocí http hlavičky. Poslední možností, jak připojit externí soubor je tagem `< style >` s pravidlem *@important*.

## 2.4 JavaScript

JavaScript je programovací jazyk, který je společně s HTML a CSS jednou z hlavních technologií používaných pro tvorbu webových stránek, nicméně jeho použití se neomezuje pouze na webové stránky. U webových stránek se používá pro zajištění interaktivního designu a interakci s uživatelem. V roce 2022 98% webových stránek používalo JavaScript na straně klienta. Syntaxe JavaScriptu patří do rodiny jazyků C/C++/Java. Navzdory podobnému jménu jsou JavaScript a Java odlišné jazyky a název Java je v případě JavaScriptu zahrnut pouze z marketingových důvodů.

### 2.4.1 Vlastnosti

Na anglické wikipedii jsou vyjmenovány následující vlastnosti:

*"JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMAScript standard. It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM)."*<sup>[15]</sup>

Nyní budou některé vlastnosti vysvětleny. Objektivě orientovanému programování je věnována samostatná kapitola [Objektivě orientované programování](#).

- high-level (vysokoúrovňový programovací jazyk) - tyto jazyky disponují silnou abstrakcí v tom, jak je zacházeno s hardwarem počítače. Opakem jsou nízkourovňové jazyky, které jsou naopak s hardwarem provázány velmi úzce a je například možné přistupovat k jednotlivým adresám v paměti počítače.
- just-in-time compiled (JIT) - způsob vykonávání zdrojového kódu, kdy nejsou zdrojové kódy překládány do strojového kódu před spuštěním programu, nýbrž až za běhu programu, často z takzvaného bytekódu, již kompilovaného zdrojového kódu. Obvykle dosahuje většího výkonu než interpretované jazyky a zároveň kombinuje jejich výhody (např. optimalizace možné až za běhu programu).
- dynamic typing (dynamicky typovaný) - vlastnost programovacích jazyků, kdy je typ proměnné vázán na hodnotu. Není tedy potřeba při deklaraci proměnné definovat její typ - ten je určen podle proměnné.
- first-class functions - programovací jazyky s touto schopností nerozlišují mezi funkcemi a běžnými proměnnými. Je v nich tedy možné například jako parametr funkce určit jinou funkci.
- event-driven (událostmi řízený) - programovací paradigma, kdy jsou funkce spouštěny na základě nějakého podnětu, události. Příkladem může být zmáčknutí tlačítka od uživatele.
- functional (funkcionální) - programovací styl vycházející z lambda kalkulu. Hlavní myšlenkou je vyhodnocování vstupního výrazu pomocí jeho zjednodušování až na elementární výrazy. Zjednodušování typicky probíhá formou rekurze.
- imperative (imperativní programování) - podporuje řešení problému krok za krokem. Typickými prvky imperativního programování je přiřazení do proměnné, cykly, podmíněné skoky.

- multi-paradigm (multiparadigmatický) - jazyk umožňující zapojení více programovacích technik. V případě JavaScriptu jsou tyto techniky například imperativní programování, funkcionální programování, řízení událostmi, atd.
- API (application programming interface) - knihovna funkcí, které abstrahují jednodušší příkazy do složitějších celků, které jsou však jednodušší pro konkrétní použití. [14, 15]

## 2.4.2 Historie

Do roku 1995 byly webové prohlížeče schopné zobrazovat pouze statický obsah, bez možnosti jakkoli zobrazit pohyblivé prvky. To se rozhodla změnit společnost Netscape, která se začala ubírat dvěma cestami. Ve spolupráci se společností Sun Microsystems začala pracovat na zakomponování existujícího jazyka Java do jejich webového prohlížeče Navigator a také pověřila Brendana Eichu zakomponováním jazyku Scheme do Navigatoru. Nadřízení Eichu trvali na tom, aby syntaxe jazyka připomínala Javu, proto Eich upustil od zakomponování Scheme do prohlížeče a místo toho napsal během 10 dnů vlastní jazyk. Nejprve pojmenován Mocha, v září 1995 přejmenován na LiveScript, a nakonec v prosinci 1995, při oficiálním vydání, přejmenován na JavaScript. Souběžně také vyvinul pro JavaScript interpret.

V roce 1995 vznikl také prohlížeč Internet Explorer, jenž byl vyvinut společností Microsoft. Reverzním inženýrstvím Navigatoru vyvinul do Exploreru vlastní skriptovací jazyk JScript, jenž byl s JavaScriptem pouze částečně kompatibilní. To představovalo značné nesnáze pro webové vývojáře, kteří museli dělat kompromisy, aby webové stránky vypadaly obstojně v obou majoritních prohlížečích - Internet Exploreru i Navigatoru.

Tato dualita ve skriptovacích jazycích pokračovala až do roku 2004. Během té doby rostla popularita Exploreru až tak, že v roce 2000 zaujímal 95% podíl na trhu, což v podstatě učinilo JScript standardním jazykem webového programování. V roce 2004 vyšel pod organizací Mozilla, následovníkem Nescapu, open-source prohlížeč Firefox. Opětovnému nárůstu popularity JavaScriptu pomohl Jessie James Garret, když publikoval článek o nových webových technologiích založených na JavaScriptu a hlavně jejich souhrnnou filozofii - Ajax (Asynchronous JavaScript and XML). Ten popisoval asynchronní řízení událostí, například načítání dat na pozadí, což se pozitivně projevilo například v rychlosti načítání stránek. Na tento rozvoj navázalo mnoho open-source knihoven zjednodušujících psaní webových aplikací jako Prototype(2005) nebo JQuery(2006). Rozšířit JavaScript mimo prohlížeč se povedlo také díky Ryanu Dahlvi, který napsal prostředí Node.js (2009), které umožňuje spouštět JavaScript, například na straně serveru. [14, 15]

V současnosti je JavaScript jediným skriptovacím jazykem na straně uživatele, který je webovými prohlížeči podporován, což z něj dělá jeden z nejrozšířenějších programovacích jazyků. [14]

## 2.5 Objektově orientované programování

Objektově orientované programování (OOP) je programovací paradigma (styl, technika), založené na konceptu objektů, jež uchovávají data a kód, který by se měl vztahovat k datům objektu. Kód, jež je ve standardním smyslu funkcí, se v kontextu objektu nazývá metoda. Určitým protipólem k OOP je programování imperativní (procedurální). Základními pojmy OOP jsou:

- Třída - šablona, podle které se následně vytváří objekt. Všechny objekty dané třídy ponese stejné proměnné (stejný datový typ, pokud je rozlišován, a názvy proměnných), ale mohou v nich mít uložené jiné hodnoty.
- Objekt - proměnná, která obsahuje další proměnné, tak jak byly nadeklarovány ve třídě. Prakticky pomáhá k abstrakci jednoduchých datových typů do větších, člověku přístupnějších, datových celků. Nejprve je vytvořena třída *Auto*, která ob-

```
class Auto{  
    //v konstrukturu předáme novému objektu parametry  
    constructor( znacka, rok_vyroby){  
        this.znacka = znacka;  
        this.rok_vyroby = rok_vyroby;  
    }  
  
    getZnacka(){  
        return this.znacka;  
    }  
  
    getRokVyroby(){  
        return this.rok_vyroby;  
    }  
}  
  
//ve funkci vyrobíme objekt typu Auto jménem Audi  
function vyrobAuto(){  
    var Audi = Auto("audi", 2015);  
}
```

Obrázek 8: Rozdíly mezi objektem a třídou

sahuje proměnné *znacka*, *rok\_vyroby*. Poté je vytvořen samotný objekt *Audi*, který v sobě uchovává všechny proměnné, k nimž lze přistupovat. (implementace v JavaScriptu)

- Zapouzdření - k datům objektu je typicky zakázáno přistupovat jinak, než přes jeho rozhraní - funkce ( v kontextu objektů nazývané metody). Tyto metody se obvykle nazývají *get* a *set*.
- Dědičnost - pro možnost větší abstrakce slouží dědičnost. Objekt může být potomkem jiného objektu. Tehdy přebírá jeho vlastnosti - data a metody. Tyto vlastnosti jsou dále součástí objektu a lze k nim přistupovat běžným způsobem.
- Polymorfismus - umožňuje různým objektům, které mají společného předka, aby byla zavolána metoda přes tohoto předka. Implementace dané funkce se v potomcích typicky liší. Dalším druhem polymorfismu (parametrický) je volání funkce pro různé datové typy. Příklad z jazyka *c++*, ve kterém to umožňuje použití šablon. [18]

```

#include <iostream>

template <class TypA, class TypB>
int secti(TypA A, TypB B){
    return A+B;
}

int main(){
    int a = 1, b = 2;

    //zavolej šablonu funkce secti s typy int, int
    std::cout<<secti<int, int>(a, b)<<std::endl;

    return 0;
}

```

Obrázek 9: Příklad parametrického polymorfismu.

## 2.6 Verzovací systémy

Verzovací systémy jsou nástroje pro uchovávání změn libovolných souborů v historii. Dále jsou používány k usnadnění vývoje nejčastěji softwarových projektů při týmovém vývoji. Je tedy možné kdykoli dohledat autora libovolné změny a stav souboru k danému datu. Verzovací systémy jsou nejčastěji používány pro zaznamenávání změn ve zdrojových souborech programů. Je však možné je použít k verzování libovolných dat.

Pro vývoj tohoto projektu byl využit verzovací systém git, který bude popsán níže.

Verzovací systémy lze dělit podle způsobu distribuce dat mezi vývojáři.

### 2.6.1 Centralizované systémy

První možností je centralizovaný systém, který pracuje s jedním hlavním centrálním úložištěm, k němuž uživatelé (vývojáři) přistupují a upravují jej. Většina operací s repozitářem vyžaduje připojení k onomu serveru.

### 2.6.2 Distribuované systémy

Oproti centralizovaným verzovacím systémům neexistuje centrální repozitář. Naproti tomu vlastní každý uživatel vlastní kopii projektu. Změny provedené uživatelem jsou poté sdíleny s ostatními typicky pomocí smlouvaného serveru. Tento server je však možné kdykoli zaměnit s libovolným jiným.

### 2.6.3 Git

Git je příkladem distribuovaného verzovacího systému. Je vyvíjen od roku 2005, kdy projekt započal Linus Torvalds, hlavní vývojář linuxového jádra, aby jej mohl udržovat. Předtím byl pro správu linuxového jádra používán verzovací systém BitKeeper, tehdy proprietární, dnes již open-source. [19] Git je používán pro správu malých projektů, včetně tohoto, tak i pro rozsáhlé projekty, jako například [blender](#) nebo již zmíněné [linuxové jádro](#).

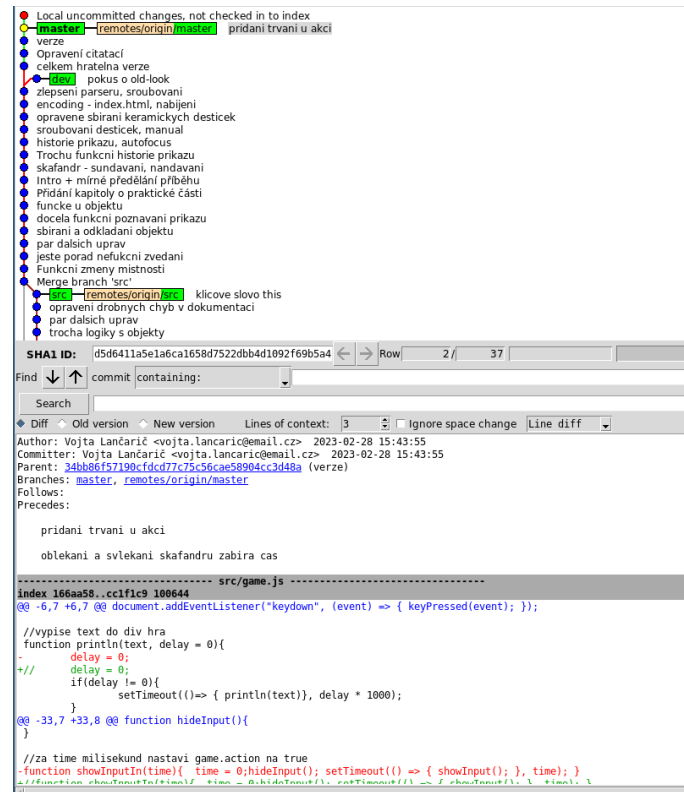
### 2.6.4 Vlastnosti

Následující požadavky byly klíčové při vývoji gitu a nadále se těmito vlastnostmi vyznačuje:

- Git významně podporuje paralelizaci vývoje v tzv. větvích (branch). Díky tomu lze pracovat na jednotlivých změnách např. v programu paralelně, aniž by došlo k

promítnutí změny do hlavní verze programu. Až po otestování funkcionality může být přidána do hlavní větve programu (tzv. merge).

- Vyznačuje se vysokou efektivitou při práci se soubory, a proto je použitelný i u velkých projektů, aniž by operace trvaly nepřiměřeně dlouho.
- Vysoká ochrana proti nechtěnému poškození souborů.



Obrázek 10: Ukázka historie projektu v gitu

### 2.6.5 Použití v projektu

Při vývoji tohoto projektu byl git použit jednak pro zaznamenávání provedených změn, užitečné zejména při samotném vývoji hry - psaní kódu, a také pro sdílení práce s vedoucím prostřednictvím webové stránky Github. Na [teto](#) webové adrese na nechází zdrojové kódy k práci.

### 2.6.6 Github

Github je webová služba umožňující sdílet a uchovávat gitové repozitáře (projekty), buďto jako soukromé (private), nebo veřejné (public). Uživatelům s oprávněním umožňuje komentovat jednotlivé commity a klonovat repozitáře pro vlastní potřebu.

## 2.7 TADS 3

Alternativou k použití JavaScriptu jako hlavního jazyka pro vývoj textové hry je TADS 3 (Text Adventure Development System). Tento jazyk byl vyvinut speciálně pro tvorbu textových her. Již sám o sobě podporuje typické operace pro textové hry jako přesuny mezi místnostmi, braní předmětů, atd. Z anglického originálu je navíc přeložen do češtiny.

Česká verze podporuje automatické skloňování podstatných jmen (předmětů), o což se tak nemusí starat uživatel. Předcházející verze TADS 1 a TADS 2 nebyly pod licencí, která by umožňovala volné rozšiřování jazyka a byly vyvíjeny komerčně. [23, 24]

## 2.8 L<sup>A</sup>T<sub>E</sub>X

L<sup>A</sup>T<sub>E</sub>X je rozšíření jazyku TeX, určenému k sázení dokumentů. TeX je, podobně jako například HTML, značkovací jazyk - formátování výsledného dokumentu je zajištěno množinou značek, které předdefinovaným způsobem ovlivňují vzhled vzniklého dokumentu. Svou filozofií, určení k vytváření technických a matematických dokumentů, podporuje zápis matematických vzorců, chemických sloučenin i dalších technických specifikací.

### 2.8.1 Historie

TeX byl vytvořen stanfordským profesorem Donaldem Knuthem, počítačovým vědcem. První verzi vydal v roce 1978. Motivací ke vzniku byla nedostatečná kvalita tehdejších sazečských programů na Stanfordské univerzitě a s ní spojená nízká kvalita školních publikací. Na jeho práci navázal Leslie Lamport, jenž vytvořil sadu maker k TeXu, nazývaná L<sup>A</sup>T<sub>E</sub>X. Důvodem byla značná složitost sázení textu v samotném TeXu. L<sup>A</sup>T<sub>E</sub>X svou koncepcí sázení značně zjednodušuje a činí jej dostupnější.

### 2.8.2 WYSIWYG editory

Jistou alternativou k vytváření dokumentů skrze značkovací jazyky jsou WYSIWYG editory. Poněkud exoticky znějící zkratka pochází z anglického: what you see is what you get. V těchto editorech je upravovaný dokument zobrazován stejně jako výsledný dokument. Typickým příkladem WYSIWYG editorů mohou být programy z kancelářského balíčku Microsoft Office - Word, Excel, Power Point nebo programy LibreOffice.

### 2.8.3 Příklady

Ukázkový příkladem podpory matematické notace nám poslouží následující zápis matic.

$$\left( \begin{array}{cc|cc} c & -c & 1 & 0 \\ c & c & 0 & 1 \end{array} \right) \sim \left( \begin{array}{cc|cc} 1 & -1 & \frac{1}{c} & 0 \\ 1 & 1 & 0 & \frac{1}{c} \end{array} \right)$$

Tohoto výsledku je docíleno jednoduchou sérií značek:

```

$$
\begin{pmatrix}[cc|cc]
  c & -c & 1 & 0 \\
  c & c & 0 & 1
\end{pmatrix}
\sim
\begin{pmatrix}[cc|cc]
  1 & -1 & \frac{1}{c} & 0 \\
  1 & 1 & 0 & \frac{1}{c}
\end{pmatrix}
$$
```

Obrázek 11: Vytvoření matice v L<sup>A</sup>T<sub>E</sub>X

#### 2.8.4 Použití v projektu

V tomto projektu byl použit  $\text{\LaTeX}$  spolu mnoha balíčky, jenž slouží k dalšímu rozšíření jazyka. Jedním ze zajímavých balíčků je balíček babel s parametrem czech, který se stará o typografická pravidla jednotlivých jazyků. V tomto případě Čestiny, kdy se sám stará mimo jiné o správné odsazování jednoslabičných spojek, dělení slov a zarovnávání odstavců, které se v Čestině liší od anglického standardu.

Pro překlad TeXu byl použit konzolový program pdf $\text{\LaTeX}$ .



## 3 Praktická část

Hra je z velké části napsána v JavaScriptu, jenž dynamicky upravuje obsah html stránky. Ta je dále upravena pomocí css, aby připomínala starou herní konzoli.

### 3.1 Námět

Hra se odehrává na palubě fiktivní rakety Shumaker-Levi 9, která je při startu neznámým způsobem poškozena, nicméně pokračuje dále v plánovaném letu. Komunikace hráče, jediného člena posádky, s řídicím střediskem je přerušena, a hráč se tak ocitá v neznámém způsobem poškozené raketě a je nucen situaci vyřešit.

Původní inspirací pro hru byla zkáza raketoplánu Columbia, která se odehrála 1. února 2003 při přistání. Příčinou havárie se ukázal být kousek hmoty, který se uvolnil při startu z externí palivové nádrže a udělal díru do křídla raketoplánu. Závada se projevila až při konci mise při přistání, kdy se do poškozeného křídla dostal horký vzduch, který poškodil statiku křídla, a raketoplán se rozpadl. [25]

Kvůli hře jednoho hráče byl raketoplán nahrazen raketou - rakeoplán je koncipován pro více osob. Děj hry je v blíže nespecifikovaných časových a místních podmínkách.

### 3.2 Engine

Samotnou hru zle rozdělit z hlediska funkčních celků na engine, který zajišťuje fungování hry - přijímání příkazů, vyhodnocování, reakce na nesprávné příkazy, chybové hlášky, atd., a samotný popis děje. V enginu jsou implementovány pouze základní příkazy, které obecně fungují na větší množství objektů nebo hráče. Tím jsou myšleny přesuny mezi místnostmi, zvedání předmětů, nápovědy k lokalitám a předmětům, a také jejich popisy. Naproti tomu příkazy vztahující se k jednomu předmětu, například *čti* jsou přesměrovány na konkrétní objekty (v případě příkazu *čti* je to manuál), které je pak zpracují dle potřeby. To umožňuje snadnou a přehlednou rozšiřitelnost hry pomocí krátkých funkcí.

```
//word1 je slovo od uzivatele
longestSubstring(word1, word2){
    var l1 = word1.length;
    var l2 = word2.length;

    var max = 0;
    //pro vsechny zacatky word1
    for(var i = 0; i < l1; i++){
        for(var j = 0; j < l2; j++){
            var pole = [];
            var l = 0; //delka shody
            for(var k = 0; k < Math.min(l1 - i, l2 - j); k++){
                if(word1[i+k] !== word2[j+k]) break;
                l++;
            }
            pole.push(l);
            max = Math.max(l, max);
        }
    }
    //word1 je obraz
    return max;
}
```

Obrázek 12: Funkce na najetí nejdelšího společného řetězce mezi vstupy

Aby byl proces přidávání nových funkcí co nejjednodušší, jsou funkce zapsány v datové struktuře Map pod klíčem s jejich názvem.

Příkaz přijatý od uživatele je nejprve upraven parserem, který najde odpovídající příkaz a poté je stromovou strukturou přesměrováván a zpracováván.

### 3.3 Parsování vstupu

Pro rozpoznávání uživatelem vložených příkazů slouží parser. Je-li vstup přijat od uživatele, parser jej nejprve zbaví diakritiky, poté je textový řetězec rozdělen na jednotlivá slova po mezerách. Tento proces je vykonán při přijetí příkazu. Dále parser zajišťuje poznávání slov - nejprve ve funkci *inputCommand* je příkaz rozdělený po mezerách testován s množinou názvů funkcí (jdi, pomoc, poloz, zvedni, popis, ...). V případě, že parser nerozpozná shodu mezi vstupem od uživatele a funkcemi, jsou dále takto testovány funkce u objektů. Je-li rozpoznána shoda s nějakou funkcí, je tato funkce zavolána. Uvnitř této funkce je opět testována shoda mezi předpokládanými parametry funkce (například příkaz *jdi* očekává jako parametr název místnosti) a vstupem od uživatele. Tímto stromovým systémem je zpracováván celý vstup od uživatele, k čemuž napomáhá datová struktura Map.

```
removePunctuation(text){
  const chars = 'áčďéěíňóřšťůúýž';
  const replace = 'acdeeinorstuuyz';

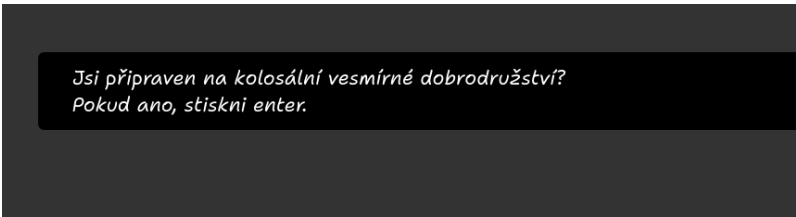
  for(var i = 0; i < chars.length; i++){
    var re = new RegExp(chars[i], 'gi');
    text = text.replace(re, replace[i]);
  }
  return text;
}
```

Obrázek 13: Odstranění diakritiky ze vstupu

### 3.4 Příkazy

Nyní budou stručně popsány příkazy, které jsou implementovány již v enginu, a jejich funkčnost.

- popiš - tento příkaz, je-li zapsán bez parametrů, vypíše popis k aktuální místnosti. Pokud je s parametrem, popíše objekt, který mu byl zadán jako parametr, pokud je v dosahu. Synonymem je také *rozhlédni se, prozkoumej*
- vezmi - příkaz vezmi přijímá parametr názvů objektů, které jsou v okolní místnosti. Pokud dostane jako parametr objekt, který se nenachází v okolní místnosti, vypíše chybovou hlášku. Je možné použít také slova *zdvihni, zvedni, seber*.
- polož - stejně jako příkaz vezmi, přijímá za parametry objekty, které si hráč nese s sebou. Zde je synonymem příkaz *odlož*.
- jdi - validními parametry jsou názvy místností. Zajišťuje přesun do sousedních místností. Dále je možno použít *přesuň se*.

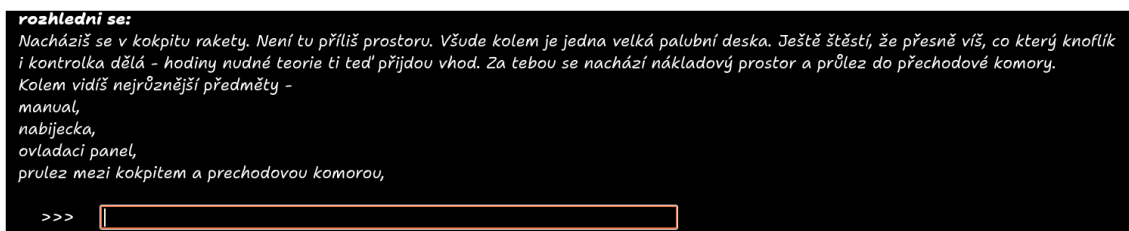


Jsi připraven na kolosální vesmírné dobrodružství?  
Pokud ano, stiskni enter.

Obrázek 14: Po spuštění vítá hráče úvodní obrazovka

### 3.5 Děj

Hra začíná startem rakety Shumaker-Levi 9, na jejíž palubě se hráč, jakožto vycvičený astronaut, ocitá. Při startu je však raketa neznámým způsobem poškozena a hráčovým úkolem je opravit raketu a vrátit se bezpečně na Zemi. Jediným pomocníkem se stává palubní manuál, v němž jsou popsány kroky, které musí hráč učinit, aby například prošel přechodovou komorou do volného vesmíru, kde zjistí rozsah poškození rakety. Manuál ve hře funguje jako jakýsi návod, k němuž by se měl hráč uchýlit v nouzi - neví-li jak postupovat. V manuálu jsou popsány netriviální a neintuitivní operace s předměty, které je potřeba učinit, aby hráč dosáhl pokroku. Příkladem může být samotné opravení závady, která, jak by měl hráč zjistit v průběhu hry, spočívá v poškozeném keramickém plášti rakety. Pro opravení pláště je potřeba vystoupit do vesmíru, kde je nutné přiložit na poškozené místo keramické destičky, které hráč najde v nákladovém prostoru, a následně je přišroubovat aku vrtačkou. Aku vrtačku najde hráč taktéž v nákladovém prostoru. Záludností ovšem je, že je zpočátku nenabitá, což by měl hráč zjistit až ve volném vesmíru, neboť v ostatních prostorech nelze, z důvodu bezpečnosti, aku vrtačku používat. K nabití slouží nabíječka, kterou hráč najde v kokpitu rakety. Tu je potřeba nejprve vzít do ruky, následně zapojit do nabíječky, a posléze nabít aku vrtačku. Nabíjení trvá několik minut. V této době by neměl hráč s nabíječkou nijak manipulovat (brát), neboť tím je proces nabíjení přerušen.



**rozhleďni se:**  
Nacházíš se v kokpitu rakety. Není tu příliš prostoru. Všude kolem je jedna velká palubní deska. Ještě štěstí, že přesně víš, co který knoflík i kontrolka dělá - hodiny nudné teorie ti teď přijdou vhod. Za tebou se nachází nákladový prostor a průlez do přechodové komory.  
Kolem vidíš nejruznější předměty -  
manual,  
nabíječka,  
ovladací panel,  
průlez mezi kokpitem a přechodovou komorou,  
>>>

Obrázek 15: Po zadání příkazu rozhleďni se je vypsán popis okolí spolu s předměty v místnosti.

Hra se odehrává ve čtyřech lokalitách. Nejprve se hráč nachází v kokpitu, posléze objeví nákladový prostor, přechodovou komoru a volný vesmír. Lokality mají význam oddělení. Hráč tedy může používat pouze předměty, které má u sebe nebo se nacházejí ve stejné místnosti jako hráč.

## 4 Závěr

Byla vytvořena funkční textová hra s inspirací tématem zkáza raketoplánu Columbia. Hra se odehrává na palubě fiktivní rakety Shumaker-Levi 9 a ve volném vesmíru, a nabízí hráči kolem 20 příkazů, z nichž některé popisují stejné děje. Hrací čas je velmi rozdílný podle zkušenosti hráče s textovými, či jinými hrami. Obecně lze říci, že by mělo být běžné dohrát hru bez předchozí zkušenosti do jedné hodiny.

Jelikož není JavaScript primárně určen k psaní textových her či adventur, neexistují příliš dobré nástroje, které by jejich vývoj usnadňovaly. Nelze říci, že by se během vývoje hry vyskytly vážnější problémy. Spíše problémy, které byly způsobeny neoptimalitou JavaScriptu jako jazyka pro vývoj textových her, jako například délka kódu.

Pro snazší vývoj by mohl být použit programovací jazyk TADS 3. Dále by samotná hra mohla být libovolným způsobem rozšiřována přidáváním hracích místností, objektů, funkcí, tlačítek, ... Plánovanou funkcionalitou byla nutnost upilovat před použitím aku vrtačky její bit (nástavec pro šroubování různých druhů šroubků) z důvodu nekompatibility bitu a hlavy šroubků. K tomu však nakonec nedošlo, neboť se zřejmě jedná o dosti neintuitivní postup, který by mohl mít nežádoucí vliv na plynulost hraní.

## Seznam obrázků

1	Otevření odkazu na nové kartě . . . . .	6
2	Tag <code>img</code> . . . . .	6
3	Selektor tagu <code>h3</code> . . . . .	7
4	Selektor třídy a <code>id</code> . . . . .	7
5	Inline zápis <code>css</code> stylů . . . . .	8
6	Zápis <code>css</code> v hlavičce <code>html</code> . . . . .	8
7	Načtení <code>css</code> do <code>html</code> z externího souboru . . . . .	8
8	Rozdíly mezi objektem a třídou . . . . .	11
9	Příklad parametrického polymorfismu. . . . .	12
10	Ukázka historie projektu v <code>gitu</code> . . . . .	13
11	Vytvoření matice v <code>L<sup>A</sup>T<sub>E</sub>X</code> . . . . .	14
12	Funkce na najetí nejdelšího společného řetězce mezi vstupy . . . . .	16
13	Odstranění diakritiky ze vstupu . . . . .	17
14	Po spuštění vítá hráče úvodní obrazovka . . . . .	17
15	Po zadání příkazu rozhlédni se je vypsán popis okolí spolu s předměty v místnost. . . . .	18

## Seznam použité literatury a zdrojů informací

- [1] The history of RPGs [online]. May 06, 2021 [cit. 2023-01-04]. Dostupné z: <https://www.pcgamer.com/the-complete-history-of-rpgs>
- [2] Textovky.cz [online]. [cit. 2023-01-04]. Dostupné z: <https://www.textovky.cz>
- [3] František Fuka. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-03-11]. Dostupné z: [https://cs.wikipedia.org/wiki/Franti%C5%A1ek\\_Fuka](https://cs.wikipedia.org/wiki/Franti%C5%A1ek_Fuka)
- [4] The Forgotten World of Teletype Computer Games [online]. [cit. 2023-01-04]. Dostupné z: <https://www.pcmag.com/news/the-forgotten-world-of-teletype-computer-games>
- [5] Textbased game. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-04]. Dostupné z: <https://en.wikipedia.org/wiki/Text-based-game>
- [6] Adventura. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-22]. Dostupné z: <https://cs.wikipedia.org/wiki/Adventura>
- [7] Textová hra. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-07]. Dostupné z: [https://cs.wikipedia.org/wiki/Textov%C3%A1\\_hra](https://cs.wikipedia.org/wiki/Textov%C3%A1_hra)
- [8] HISTORY OF HTML [online]. [cit. 2023-01-06]. Dostupné z: <https://www.bu.edu/lernet/artemis/years/2020/projects/FinalPresentations/HTML/historyofhtml.html>
- [9] HTML. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-06]. Dostupné z: <https://en.wikipedia.org/wiki/HTML>
- [10] On the Difficulty of Counting the Number of HTML Elements [online]. [cit. 2023-01-06]. Dostupné z: <https://meiert.com/en/blog/the-number-of-html-elements/>
- [11] Hypertext Markup Language. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-06]. Dostupné z: [https://cs.wikipedia.org/wiki/Hypertext\\_Markup\\_Language](https://cs.wikipedia.org/wiki/Hypertext_Markup_Language)
- [12] Kaskádové styly. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-22]. Dostupné z: [https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9\\_styl](https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styl)
- [13] CSS2 – selektory, pseudotřídy a pseudoelementy [online]. [cit. 2023-01-06]. Dostupné z: <https://www.interval.cz/clanky/css2-selektory-pseudotridy-a-pseudoelementy/>
- [14] JavaScript. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-06]. Dostupné z: <https://cs.wikipedia.org/wiki/JavaScript>
- [15] JavaScript. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-06]. Dostupné z: <https://en.wikipedia.org/wiki/JavaScript>

- [16] Learn the History of Web Browsers [online]. [cit. 2023-01-08]. Dostupné z: <https://washingtonindependent.com/w/learn-the-history-of-web-browsers/>
- [17] Brendan Eich [online]. [cit. 2023-01-08]. Dostupné z: [https://en.wikipedia.org/wiki/Brendan\\_Eich#Netscape](https://en.wikipedia.org/wiki/Brendan_Eich#Netscape)
- [18] Object-oriented programming. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-01-06]. Dostupné z: [https://en.wikipedia.org/wiki/Object-oriented\\_programming](https://en.wikipedia.org/wiki/Object-oriented_programming)
- [19] Git. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-12-18]. Dostupné z: <https://cs.wikipedia.org/wiki/Git>
- [20] Reference [online]. [cit. 2023-01-06]. Dostupné z: <https://git-scm.com/docs>
- [21] Tex. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-12-19]. Dostupné z: <https://en.wikipedia.org/wiki/TeX>
- [22] Tex. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2022-12-19]. Dostupné z: <https://en.wikipedia.org/wiki/LaTeX>
- [23] TADS. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-02-28]. Dostupné z: <https://en.wikipedia.org/wiki/TADS>
- [24] Tads [online]. [cit. 2023-02-28]. Dostupné z: <http://tads.cz/cs/novinky>
- [25] Columbia (raketoplán). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2023-02-03]. Dostupné z: [https://cs.wikipedia.org/wiki/Columbia\\_\(raketopl%C3%A1n\)](https://cs.wikipedia.org/wiki/Columbia_(raketopl%C3%A1n))