FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Nástroj pro tvorbu interaktivní fikce
Student:	Zdeněk Havelka
Vedoucí:	Ing. Filip Glazar
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Webové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout a implementovat webové řešení pro tvorbu interaktivní fikce, dále jen gamebook. Webová aplikace by měla minimálně umožňovat vytvoření základního gamebooku pomocí vyvinutého nástroje. Dále musí být možno průběžně ukládat rozpracovaný projekt a jeho exportování do webové prezentace, které bude sloužit jako jeho publikování. Softwarové dílo by mělo zahrnovat i několik základních stylů gamebooků pro uživatele s možnosti jejich snadného rozšíření.

Postupujte dle následujících kroků:

1) Proveďte důkladnou analýzu existujících řešení
2) Specifikujte funkční požadavky
3) Proveďte analýzu a volbu vhodných technologií
4) Navrhněte webovou aplikaci s důrazem na možnost rozšíření
5) Implementujte prototyp aplikace
6) Proveďte testování, a to minimálně tak, že vytvoříte ukázkový gamebook v implementovaném prototypu
7) Připravte prostředí pro zveřejnění vaši aplikace

Elektronicky schválil/a Ing. Jaroslav Kuchař, Ph.D. dne 30. listopadu 2021 v Praze.

Bakalářská práce

NÁSTROJ PRO TVORBU INTERAKTIVNÍ FIKCE

Zdeněk Havelka

Fakulta informačních technologií
Katedra softwarového inženýrství
Vedoucí: Ing. Filip Glazar
12. května 2022

Obsah

Seznam obrázků

Seznam výpisů kódu

Chtěl bych poděkovat v první řadě Ing. Filipu Glazarovi, za jeho vedení, bez nějž by tato práce nikdy nevznikla, a Jiřímu Macháčkovi, se kterým jsme společně zdolávali překážky na cestě k zakončení bakalářského studia.
Dále děkuji Ing. Barboře Havelkové MiM., Bc. Dominice Draesslerové, Mgr. Martinu Šipkovi a Ing. Zdeňku Havelkovi, PhD. za jejich užitečné rady a Vojtěchu Bádalovi, Jakubu Havelkovi, Martině Mihlové a Valentýně Novákové za jejich hodnotnou zpětnou vazbu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2022 .....

Abstrakt

Předmětem této práce je vývoj nástroje pro tvorbu interaktivní fikce, a to od fáze analýzy problému, přes návrh a implementaci, až po testování. V průběhu celého vývoje je kladen důraz na uživatelskou příjemnost a na snadnou rozšiřitelnost funkcionality aplikace. K reprezentaci rozvětveného příběhu interaktivně fiktivního díla je využita teorie grafů.

Výstupem práce je prototyp, který poskytuje dobrou představu, jak by měl kompletní nástroj vypadat, a rozepsaný postup, který by měl být aplikován pro jeho budoucí vývoj. Bude-li tento nástroj dále rozšiřován, má potenciál přispět k vývoji žánru interaktivní fikce a podpořit kreativitu a zájem o informatiku v mladších generacích.

Klíčová slova interaktivní fikce, gamebook, choice-based text adventure, teorie grafů, herní engine, webová aplikace, rozšiřitelnost, UX, Angular

Abstract

The subject of this work is development of a tool for interactive fiction creation, from phase of analysis through design and implementation to testing. During the development user friendliness and extensibility of application are emphasized. Graph theory is used to represent branching story of interactively fictional work.

The result of the work is a prototype, which gives a good idea, of how the tool should look like, once completed, and laid out process of its future development. Shall the tool be further expanded, it has the potential to contribute to the evolution of the interactive fiction genre and to encourage creativity and promote interest in informatics in the younger generation.

Keywords interactive fiction, gamebook, choice-based text adventure, graph theory, game engine, web application, extensibility, UX, Angular

Seznam zkratek

AI	Artificial Inteligence
API	Application Programming Interface
CBTA	Choice-based Text Adventure
CI	Continuous Integration
CSS	Cascading Style Sheets
DFS	Depth-first Search
DOM	Document Object Model
GUI	Graphical User Interface
HF	Hypertext Fiction
HTML	HyperText Markup Language
IF	Interactive Fiction
IFTF	Interactive Fiction Technology Foundation
JS	JavaScript
JSON	JavaScript Object Notation
MIT	Massachusetts Institute of Technology
MPA	Multiple-page Application
MUD	Multi-user Dungeon
PACA	Point And Click Adventure
PBTA	Parser-based Text Adventure
POC	Proof of Concept
RPG	Role-playing Game
SPA	Single-page Application
TADS	Text Adventure Development System
TS	TypeScript
TTRPG	Tabletop Role-playing Game
UI	User Interface
UX	User Experience
VN	Visual Novel
ZIL	Zork Implementation Language

Úvod

Žijeme v době, kdy je vývoj her jednodušší a přístupnější než kdy dřív, jak můžeme vidět například na počtu ročně vydaných nezávislých her na platformě *Steam*, který za posledních deset let vyrostl sedmdesátkrát [1]. Technologické překážky jako nedostatečná paměť nebo práce s 3D grafikou byly dávno překonány, platformy jako *Youtube* nabízí prakticky nekonečnou a bezplatnou knihovnu výukových materiálů a minulostí už je i nutnost, aby byl aspirující herní designér zdatným programátorem — tento nedostatek za něj totiž vykompenzuje herní engine.

Herní engine je nástroj, který zjednodušuje mnohé náročnější aspekty herního vývoje, jako například práci s grafikou a zvukem, fyzikální simulace nebo podporu multiplayeru. Velké herní společnosti obvykle používají vlastní engine šitý na míru, existují však volně dostupné varianty se širokou nabídkou funkcí a rozšíření jako jsou *Unity*, *Unreal Engine* nebo *Godot*.

Zajímavou skupinou herních enginů jsou žánrově specifické enginy. Tyto nástroje se specializují na určitý typ her. Příkladem může být *Puzzlescript* — jednoduchý engine umožňující tvořit hlavolamy nebo *M.U.G.E.N* — nástroj pro tvorbu 2D bojových her pro dva hráče. Tento typ enginu s sebou nese několik specifik. Zejména jde o nesporně jednodušší používání, které z tohoto zúžení zaměření vychází. Tyto nástroje jsou často přístupné na webu bez instalace a uživatel je schopen si rychle osvojit znalosti potřebné k tomu, aby v nich byl schopen tvořit. Na druhou stranu se však netěší takové popularitě jako obecnější enginy, jelikož mají spoustu omezení a nenabízí takovou volnost. Z těchto důvodů se většinou jedná o osobní projekty, kolem kterých často vznikne malá, ale zapálená komunita.

Právě díky této jednoduchosti a schopnosti nadchnout skupinu lidí pro kreativní činnost a pro vývoj her a informatiku obecně, leží v žánrově specializovaných enginech velký potenciál, který se v současnosti nevyužívá tolik, jak by mohl. Žánrově omezený nástroj by mohl být základním stavebním kamenem pro větší projekt, který by vznikal postupným rozšiřováním funkcionality do všech směrů, na němž by se mohla podílet právě kreativní komunita okolo tohoto enginu. Uživatel by tedy mohl začít tvořit s jednoduchostí, kterou nabízí například *Puzzlescript*, ale zároveň by měl možnost vyhnout se omezením, která by mu znemožnila se v herním designu dál rozvíjet. Takovýto nástroj by byl stále úzce svázán se žánrem, na kterém byl vystavěn. Naplňoval by tedy jiné potřeby než všeobecné herní enginy jako *Unity* a v žádném případě by jim nebyl přímou konkurencí. Mohl by však zároveň o daný typ her zvýšit zájem a revitalizovat ho. Otázkou tedy zůstává, jaký herní žánr je pro takovýto projekt nejvhodnější. Je nesporné, že některé varianty mají větší potenciál k růstu než jiné. Například bojová multiplayer 2D hra je mechanicky natolik specifická oblast, že pro velké rozšiřování funkcí není místo. Žánr, který pro tento engine předkládáme jako ideální, je interaktivní fikce.

Důkladnějšímu rozboru tohoto pojmu budeme věnovat samostatnou kapitolu. V této chvíli nám stačí chápat pojem gamebook – tedy příběh, který se větví na základě hráčových/čtenářových rozhodnutí. Gamebooky mají původ v knižní formě, kdy čtenář přečetl část příběhu a následně

mu bylo předloženo několik voleb. Hráč učinil rozhodnutí a pokračoval ve čtení na stránce, na kterou odkazovala daná volba.

Právě tento koncept je spolu s jinými formami interaktivní fikce častým předmětem specializovaných herních enginů. Nástroje jako *Twine* nebo *Choicescript* se těší nemalé popularitě a do hloubky se s nimi také seznámíme ve vlastní kapitole. Stejně jako u ostatních specializovaných herních enginů jsou však jejich možnosti omezeny hranicemi jejich cílových žánrů místo toho, aby na této bázi dále stavěly. Tvrdit, že narativ je základem každé hry by bylo krátkozraké (protipříkladem může být hra *Super Mario Bros.*, kde příběh pouze doplňuje herní mechaniky, nebo *Tetris*, v němž příběh vůbec nefiguruje), to však neznamená, že rozvětvený interaktivní příběh nemůže být jádrem velkého množství her všeho druhu. Jednotlivá rozšíření poté mohou přidat prvky jako soubojový systém, pohyb po mapě nebo různá grafická vylepšení a vykrystalizovat tak v aplikaci umožňující jak rychlé psaní jednoduchého gamebooku, tak tvorbu složitého RPG s propracovaným světem.

Interaktivní fikce nám však přináší ještě jedno další pozitivum a tím je přístupnost pro hráče. Analogicky k počátkům tohoto žánru, můžeme celý gamebook používat a šířit v textové podobě. Nebo lépe — v hypertextové podobě. Jak si popíšeme v kapitole věnované rozboru použitých technologií, forma webové prezentace nám umožní spustitelnost na jakémkoli zařízení s prohlížečem a přímo se tak nabízí k edukativním účelům. Interaktivní fikce má nesporný potenciál ozvláštnit výuku na dálku, která v kontextu posledních let nabyla velké důležitosti, ale není vyloučeno ani využití přímo v hodinách nebo v zájmových kroužcích.

Takovýto nástroj by tedy mohl mít nezanedbatelný dopad na vzbuzování zájmu o informatiku u dětí, doplnit výuku všeho druhu a dát vzniknout komunitě spisovatelů-programátorů a čtenářů-uživatelů, podporující se v kreativní tvorbě rozsáhlého katalogu výukových a zábavních děl. V budoucnu by se zároveň mohl rozrůst v engine umožňující tvořit hry různých žánrů vystavěné okolo interaktivního příběhu dopomoci tak k nastavení vyšších nároků na kvalitu příběhu v herním průmyslu.

Cílem této práce je vyvinout prototyp takového enginu a navrhnout, jak by měl probíhat jeho budoucí vývoj. Vyvíjenou aplikaci pojmenujeme *Vorpal*, podle neologismu použitého Lewisem Carollem v nonsensové básni *Jabberwocky*, jenž byla obsažena v románu *Through the Looking-Glass* [2]. Tento novotvar popisoval ostrý a smrtící meč. Kreativita, s jakou Caroll tvořil nová slova a fantastické světy, je schopnost, kterou bude *Vorpal* u nových autorů podporovat. Zároveň, jestliže je pero mocnější meče, tento název naznačuje, že interaktivní fikce je mocnější pera reprezentujícího běžnou literaturu.

Obrázek 1 Jabberwock a vorpálový meč [3]

Kapitola 1

Interaktivní fikce

Cílem této kapitoly je vysvětlení elementárních pojmů z oblasti interaktivní fikce, rozbor herních podžánrů, které do ní spadají a popis její historie a její aktuální role v herním průmyslu. Dále se kapitola krátce věnuje vztahu mezi narativem a interaktivitou, tedy jak se liší vyprávění příběhu v hře od ostatních médií.

Interaktivní fikce, dále jen IF, je žánrem kombinujícím literární dílo s interaktivitou digitálního světa[1]. Jde o textový narativ, jehož průběh jsou schopni čtenáři ovlivňovat pomocí příkazů, které mohou být buďto explicitně definovány nebo objevovány čtenářem. Konzument IF je tedy kombinací čtenáře a hráče. Jelikož se v této práci zaměříme na význam IF v herním průmyslu, budeme používat primárně termín hráč, nebude-li v daném kontextu vhodnější čtenář. Podobně o jejím producentovi je možné mluvit jako o autorovi, spisovateli, vývojáři nebo designérovi IF. Zde budeme preferovat výraz autor, abychom poukázali na uměleckou hodnotu tohoto žánru a nezdůrazňovali schopnost programování, která by pro práci ve *Vorpalu* neměla být potřebná.

Hlavním specifikem IF je zákonitá rozvětvenost takového díla. Chceme-li mluvit o interaktivitě, musí mít hráčova rozhodnutí dopad na vyprávěný příběh. Mají-li mít tato rozhodnutí dopad, musí v příběhu nutně existovat body, kdy bude na základě tohoto rozhodnutí vybráno mezi několika možnými cestami, kterými se může ubírat. Z této rozvětvenosti plyne mnoho důsledků, mimo jiné jde o větší náročnost tvorby, ale také větší znovuhratelnost. Těmito důsledky se budeme více do hloubky zabývat v poslední části této kapitoly a do jisté míry v kapitole příští, kde probereme aplikaci teorie grafů ke zjednodušení tvorby interaktivního příběhu.

1.1 Subžánry IF

IF má daleko k homogennímu žánru. Přirozeně přebírá žánrové rozdělení běžné literatury na příběhy hororové, detektivní a podobně. K tomu se zde však objevují nové aspekty, dle kterých můžeme díla IF seskupovat. Prvním kritériem je samotné médium, na němž je dílo zaznamenáno – v této práci se věnujeme zejména digitální formě, neměli bychom ovšem opomíjet papírové „gamebooky“, které jsou jakýmsi prarodičem digitální IF tvorby. Dalším prvkem odlišnosti je způsob zadávání hráčových příkazů a stupeň volnosti ve volbě těchto příkazů. Subžánry IF se liší také v abstraktnějších záležitostech – hra může být zaměřená na příběh, objevování fiktivního světa nebo spíše na doprovodné mechaniky jako jsou schopnosti protagonisty nebo hádanky, které hráč řeší pomocí předmětů, jenž v herním světě může získat.

[1] Nebo spíše převážně digitálního světa, jak brzy uvidíme.

Některé hodnoty těchto kritérií se navzájem vylučují[2], jiné je možné navzájem kombinovat a zkoumat jejich nuance. Množství těchto kombinací, častá nejasnost, do které kategorie hru zařadit i spory, zda je to které dílo vůbec zástupcem IF, jsou příčinou neexistence jasně definovaného rozdělení jejích podžánrů [4]. Z tohoto důvodu si zde pouze uvedeme několik termínů, které se v souvislosti s IF často používají a které budeme vnímat jako definici subžánrů IF pro účely této práce.

1.1.1 Gamebook

Gamebook je papírová forma IF, případně může existovat ve formě e-knihy. V tomto případě si však zachovává omezení spojená s knižním médiem — hráč se musí starat o větvení příběhu listováním na správnou stranu a dále o jakékoli případné rozšiřující mechaniky, jako je například zaznamenávání předmětů, které ve hře získá, nebo generování náhody hodem kostkou. Tyto další úrovně interakce může gamebook nabízet navíc k základní možnosti volit směr příběhu, nejsou však nikdy jádrem gamebooku, tím je příběh.

Gamebooky jsou také často označovány jako choose-your-own-adventure knihy podle stejnojmenné série z 80. až 90. let 20. století.

1.1.2 Solo RPG

Solo RPG je také možno považovat za IF, ačkoli se tento žánr vyvíjel poněkud jiným směrem. Jde o odnož klasických TTRPG[3] tedy společenských her, kde jeden hráč vytváří svět a plánuje děj a další hráči zaujímají role postav v tomto světě. Specialita solo RPG je, jak název napovídá, že jde o hru jednoho hráče.

Solo RPG může nabývat mnoha podob, často se však podobá gamebooku. Rozdíl spočívá v herních mechanikách mimo prostý výběr směřování příběhu, které zde mají mnohem větší důležitost. Příkladem RPG, které nabízí solo mód je *Ironsworn*.

1.1.3 Choice-based text adventure

Choice-based text adventure, dále jen CBTA, je digitální obdoba gamebooku. Hráči je předložena část příběhu a několik možností, jak pokračovat dál. Zvolením jedné z těchto možností se příběh posouvá k další části. Stejně jako u gamebooku je možné hru obohatit o doprovodné funkce, větší důležitost však má narativ. Příkladem CBTA je *Wayhaven Chronicles* od společnosti *Choice of Games*.

Engine, který v rámci této práce vznikne, bude umožňovat právě tvorbu CBTA her, není však vyloučené, že budoucí rozšíření umožní přesah z tohoto subžánru například do oblasti parser-based text adventure nebo hypertextové fikce, případně úplně z žánru IF.

Přestože jsme si gamebook definovali jako fyzickou variantu CBTA, v některých případech budeme tyto termíny zaměňovat. Jedná se zejména o kód aplikace nebo v průběhu testování, kdy bude slovo gamebook vhodnější díky větší srozumitelnosti.

1.1.4 Parser-based text adventure

Parser-based text adventure, dále jen PBTA, funguje podobně jako CBTA s tím rozdílem, že hra po každé části příběhu nenabídne hráči možnosti, co dělat dál. Místo toho čeká na jeho

[2] Například si jen těžko představíme hru, která se snaží odpovídat na všechny možné příkazy, co hráče napadnou, sepsanou na papíře.

[3] tabletop role-playing games

textový vstup, který není ničím omezen. Parser[4] následně zkontroluje, zda byl zadán nějaký z jemu známých příkazů nebo synonym k nim.

Tento přístup je zjevně náročnější pro tvůrce, neboť musí počítat s velkým množstvím příkazů a navádět hráče na ty správné. Autor PBTA musí mít dobrý odhad hráčovy psychologie, aby dokázal předvídat jeho možné vstupy a umožnit hře na ně reagovat. Dále musí být schopen tyto vstupy smysluplně seskupovat tak, aby jich jednou odpovědí pokryl co nejvíce. Pokud je PBTA dílo kvalitní, dokáže u hráče navodit pocit, že se opravdu pohybuje po fiktivním světě, jelikož hra reaguje na každý z jeho nápadů. Není-li však dobře rozvrženo, hrozí, že hráče přestane bavit, protože jsou veškeré jeho příkazy zamítány, nebo že nebude vědět, jak dál, neboť nemůže přijít na správnou kombinaci slov. Příkladem PBTA je hra *Zork*, o které si více řekneme, až se budeme věnovat historii IF.

Zvláštním typem PBTA jsou multi-user dungeon hry neboli MUD. Tyto hry fungují jako PBTA, obohacují však tento žánr o hru více hráčů, kteří spolu interagují a dohromady utváří herní příběh a svět. První hra tohoto žánru se jmenovala jednoduše *MUD* [5].

Zajímavým směrem, kterým se PBTA může ubírat v budoucnu, je začlenění umělé inteligence. AI je logickým dalším krokem, který by nahradil předdefinovaný parser a otevřel tak nekonečné možnosti. Prvním jednoduchým příkladem takového postupu je hra *AI Dungeon* vyvinutá Nickem Waltonem, která takto využívá AI ke generování obsahu.

1.1.5 Hypertext fiction

Hypertext fiction, dále jen HF, je další subžánr podobný CBTA - jde o digitální textový příběh, v němž se hráč pohybuje s použitím odkazů. Rozdíl spočívá v tom, že tyto odkazy nejsou nutně na konci dané pasáže a nemusí nutně odpovídat hráčovým rozhodnutím. Často se jedná o detaily prokreslující popisovaný svět a hráč si tak může vybírat, o čem se chce dozvědět víc a co prozkoumat.

Tato drobná změna má za následek, že HF se zaměřuje spíše na objevování fiktivního světa, než na příběh. Často také nabízí možnost vrátit se k předchozí pasáži. Této funkci se CBTA spíše vyhýbá, jelikož vyprávění příběhu neprospívá. Naopak filosofie svobodného objevování jde s možností návratu a prozkoumání jiné cesty ruku v ruce.

Pokud bychom brali v potaz pouze technickou stránku věci, dalo by se říci, že HF je nadmnožinou CBTA – možnosti na konci pasáží jsou zde také odkazy a jakýkoliv nástroj sloužící k tvorbě HF nám zákonitě umožní tvořit i CBTA. Je to však právě tento rozdíl ve filosofii těchto subžánrů, kvůli kterému dává smysl je vnímat odděleně. Příkladem HF je *The Temple of No*.

1.1.6 Na hranici IF a za ní

Jak již bylo řečeno, IF je žánr her, se kterými hráč interaguje textově a které vypráví textový příběh. Použití grafických prvků nutně neznamená, že hra/kniha nespadá pod IF[5], jsou-li však tyto grafické prvky hlavním kanálem, kterým hra s hráčem komunikuje a text má pouze doplňující funkci nebo chybí úplně, nebo pokud hráč používá tyto grafické prvky k interakci, nemluvíme o takovém díle jako o představiteli IF.

Specifickým žánrem je Visual novel, dále jen VN. Jde ve své podstatě o CBTA hru doplněnou statickým nebo jednoduše animovaným obrázkem prostředí a postav ve hře vystupujících. VN pochází z Japonska, zaměřuje se zejména na dialogy a grafika je sice neodmyslitelnou součástí žánru, příběh je však vyprávěn zejména textově. Z tohoto důvodu lze považovat VN za zástupce podžánrů IF. Známým příkladem VN je hororová hra *Doki doki literature club*.

Často jsme schopni objevit mnoho znaků IF v dílech, která její kritéria nesplňují. *Bandersnatch* – epizoda seriálu *Black Mirror* využívá formátu CBTA, pouze nahrazuje textový příběh za

[4] Program, který kontroluje a zpracovává text na vstupu, obvykle rozdělením do částí definovaného významu.
[5] Považme například gamebook s ilustracemi.

video. Podobně fungují hry od společnosti *Telltale*, které sice přidávají doplňující herní mechaniky, ale jádro je stále animovaný video příběh řízený volbami hráče. Tento přístup je náročnější na tvorbu než čistě textová IF, narativ tedy obvykle nebývá tak rozvětvený a různé možnosti se často opět spojí do jedné příběhové větve. Na druhou stranu mají tyto hry potenciál oslovit širší publikum, než čistě textově založené alternativy.

Dalším žánrem, který vznikl rozšířením možností IF je takzvaná Point and click adventura, dále jen PACA. PACA se většinou soustředí spíše na hádanky s použitím inventáře než na příběh. Klasický formát IF doplňuje grafickým znázornění lokací, ve kterých se hráč nachází, a umožňuje mu klikáním interagovat s objekty a postavami v těchto lokacích. Populárními představiteli tohoto žánru jsou hry ze série *Monkey island*, *Grim Fandango* nebo české *Machinarium*.

Přestože čistá IF je na ústupu, její principy hrají roli ve velkém množství moderních her, ať už jde o volbu hráče, která ovlivní příběh, nebo o větvící se dialogy. Dobrým příkladem je hra *Disco Elysium*, která v roce 2019 vyhrála ve čtyřech kategoriích *Game awards* [6]. *Disco Elysium* je RPG, v němž objevuje hráč svět pohybem po mapě a interakcemi s předměty, postavami i sebou samým a snaží se odhalit pachatele vraždy. Jádro hry spočívá ve čtení dialogů a vybírání reakcí hráčem ovládaného protagonisty. *Disco Elysium* nám tedy ukazuje, jak je možné vzít koncept IF a převést jej do modernější podoby.

1.2 Historie IF

Přestože, jak jsme již ukázali, není tvorba IF podmíněna existencí digitální technologie, je tato umělecká disciplína překvapivou novinkou. Jedna z prvních zmínek o větveném narativu se objevila v *The Garden of Forking Paths* z pera Jorge Luis Borgese v roce 1941. Postava jménem Ts'ui Pên zde sepíše knihu, která je zároveň labyrintem – jsou-li její postavy postaveny před nějakou volbu, zvolí si najednou všechny alternativy a vytvoří tak několik různých budoucností [7]. Prozkoumávání teorie paralelních vesmírů se následně zabývají další díla využívající rozvětvený narativ, jako je například seriál *Black Mirror* v epizodě *Bandersnatch*.

1.2.1 Papírová

Přestože tato idea existovala už dříve, první díla považovatelná za gamebook vznikají až v 60. letech 20. století. Francouzská literárně experimentální skupina *Oulipo* přichází se „stromovou literaturou“, jejímž prvním příkladem je krátký příběh *Un conte à votre façon* Raymonda Queneau [8].

Veškerá IF však byla stále silně experimentálního charakteru, až do druhé poloviny 70. let. Dvojice spisovatelů Edward Packard a R. A. Montgomery tehdy znovuobjevili koncept paralelního vyprávění v Packardově knize *Sugarcane Island* a začali vydávat úspěšnou knižní sérii *Choose your own adventure*. Série obsahovala 184 titulů napsaných mezi lety 1979 a 1998, prodala více než 250 milionů výtisků a dostala tak IF do obecného povědomí.

Tato série nastartovala velký zájem o gamebooky, který přetrval až do konce tisíciletí, a dala vzniknout takzvané americké tradici gamebooků. Její výtvory se zaměřovaly na volbu čtenáře a nezahrnovaly prvek náhody. Oproti tomu mladší britská tradice příběh doplňovala o prvky jako jsou schopnosti postav nebo házení kostkou. Populárním příkladem gamebooků britské tradice je sága *Lone wolf*. Éra gamebooků se začala chýlit ke konci v druhé polovině 80. let s rostoucí popularitou počítačových her [9].

Za zmínku stojí též vývoj TTRPG her, které se začaly objevovat kolem 70. let. Nejvýznamnější zástupce tohoto žánru – *Dungeons and Dragons* byl publikován roku 1974 Garym Gygaxem. TTRPGs měly na vývoj gamebooků velký vliv, zvláště pak na tvorbu britské tradice, narozdíl od nich se jim však podařilo přežít expanzi videoherního průmyslu, pravděpodobně díky tomu, že poskytují větší volnost než gamebooky a že narozdíl od gamebooků jsou TTRPG hry společenské.

1.2.2 Digitální

Prvním dílem digitální IF byla hra *Adventure*[6] vyvinutá v roce 1975 Williamem Crowtherem, který se mimo jiné podílel na vzniku sítě *ARPANet* – předchůdci dnešního internetu. *Adventure* fungovala na principu parseru, její součástí byl inventář a zaměřovala se na řešení hádanek a hledání cesty v jeskynním bludišti. Hra se rychle rozšířila po amerických univerzitách a institucích.

Vývoj, který opravdu nastartoval éru IF, však přišel až po vzniku *Adventure*. Bylo jím rozšíření počítačů do domácností. Hry, které dříve byly přístupné pouze na univerzitních mainframech, teď bylo možné hrát z domova. Velkým hráčem v těchto počátcích digitální IF se stala společnost *Infocom*. Její zakladatelé – studenti *MIT* Dave Lebling a Marc Blank, se inspirovali hrou *Adventure* a vytvořili hru *Zork*. *Zork* se v mnoha aspektech podobala svojí předloze. Nabízela ovšem propracovanější příběh a svět i sofistikovanější parser [10]. Lebling s Blankem hru původně naprogramovanou pro mainframe přepracovali pro domácí počítače a setkali se s komerčním úspěchem, který jim umožnil založit *Infocom*. Společnost disponovala vlastním programovacím jazykem *ZIL*[7] specializovaným na IF, který tvorbu her značně zefektivněl. Tato díla se vyznačovala svou literární kvalitou a byla distribuována i v knihkupectvích, zahrnovala různé literární žánry a po vzoru adventure se soustředila zejména na různé hádanky. *Infocom* také přišel s mnohými inovacemi v rámci žánru, jako jsou atributy postavy nebo mapování herního světa. Ke konci 80. let se však dostal do finančních potíží a byl koupen herním gigantem *Activision*, což značilo konec komerčního rozmachu textových adventur [11], [12].

Důležitou hrou v historii IF je *Mystery House* z roku 1980 od manželů Roberty a Kena Williamse, kteří se zasloužili o velký průlom v herním průmyslu, když jako první přidali do textové adventury grafiku. Paměťové omezení tehdejších disket obešli tak, že ukládali pouze souřadnice konců linií, které se pak vykreslovali na obrazovku. Tato inovace se zasloužila za velký úspěch *Mystery House* a byla předzvěstí vzniku VN a směru, kterým se měl herní vývoj ubírat [13].

Dalším zajímavým příkladem IF je hra *The Hitchhiker's Guide to The Galaxy* napsaná podle stejnojmenné knižní předlohy Douglase Adamse. Adams na ní úzce spolupracoval se společností *Infocom* a dostaly se jí nadšené ohlasy v časopise *London Times*. Tato hra se tak stala jedním z nejznámějších příkladů IF [14].

1.2.3 Aktuální stav

V dnešní době se IF stala doménou nezávislých autorů a přežívá tak v této komunitě. Zásadním faktorem, jenž toto přežití umožnil, je množství volně dostupných nástrojů, jako je například populární *Twine* vyvíjený Chrisem Klimasem od roku 2009, který umožňuje vytvářet díla HF a který si důkladně popíšeme v kapitole věnované analýze konkurence. Díky tomuto nástroji vznikla například hra *Howling Dogs* od autorky Porpentine, v níž hlavní postava utíká z ponurého vězení reálného světa, do nespočtu barevných světů virtuální reality, nebo introspektivní *Depression Quest* Zoe Quinn, kde se hráč vžije do postavy trpící klinickou depresí. Mnohá tvorba vznikající s pomocí tohoto nástroje využívá kreativně i pokročilejší funkce jako je audio nebo grafika a oproti ranějším dílům je oproštěna od přehnaného spoléhaní na hádanky.

Technologie umožňující vývoj IF jsou vyhledávány a podporovány neziskovou společností *IFTF* (*Interactive Fiction Technology Foundation*), která také organizuje *Interactive Fiction Competition* - největší soutěž IF. Právě tato podpora umožňuje existenci mnoha projektů, které jsou pro IF stěžejní, ale samy jsou neziskové, jako je například právě *Twine* [15].

[6] označovaná též jako *Colossal Cave Adventure* nebo *Advent*
[7] *Zork* Implementation Language

Komunitní povaha tohoto odvětví je znát také na množství IF her, které jsou volně dostupné z webového prohlížeče[8]. Dalším příkladem může být duben roku 2019, kdy došlo ke zveřejnění kódu všech vydaných titulů společnosti *Infocom* na platformě *Github*, což umožnilo veřejnosti jejich zkoumání, úpravy a možnost se inspirovat.

Důležitým aspektem, který hraje IF do karet, je rozšíření mobilních zařízení. Tato platforma je ideální pro konzumaci krátkých ale působivých zážitků, které může IF poskytovat. Například textová adventura *A Dark Room* byla jeden měsíc nejstahovanější hrou na obchodě *AppStore* pro zařízení od společnosti *Apple* [16].

Zajímavým projektem je již zmiňované *AI Dungeon*, které využívá umělé inteligence pro generování příběhu a hledá tak novou cestu, kterou by se IF mohla ubírat.

Nakonec nezapomeňme zmínit vliv, který má interaktivní fikce na celý herní průmysl. Ať se jedná o hry společnosti *Telltale*, jejichž jádrem je rozvětvený příběh, nebo o roleplayingové hry jako *Disco Elysium*, které staví na dialogu, není pochyb, že principy IF se promítají i mimo omezení tohoto žánru.

1.3 Vyprávění příběhu v interaktivním prostředí

Nyní se na chvíli oprostíme od žánru IF, jelikož ke stejnému oproštění aspiruje v budoucnu náš engine. Z tohoto důvodu má smysl zabývat se příběhem, tedy základní kostrou všech budoucích her vyvinutých ve *Vorpalu*, a jeho místem v interaktivním médiu obecněji.

„*Příběh ve hře je jako příběh v porno filmu: očekává se, že tam bude, ale není tak důležitý.*“, řekl kdysi John Carmack, autor legendární hry *Doom*. Diskuze o významu příběhu ve hrách se vedou již od dob jejich vzniku. První hry byly velmi omezené jak pamětí, tak výkonem. Jejich autoři si tedy museli vybrat, zda se zbaví příběhu[9] nebo všeho ostatního[10]. S postupem času a rostoucími možnostmi herního média začalo docházet ke slučování těchto dvou táborů. Abychom mohli rozhodnout, zda je příběh ve hrách důležitý, musíme nejprve identifikovat, co jim přináší.

V prvé řadě jde o emoce. Příběh má schopnost vyvolávat silné citové zážitky, kterých by hra postavená pouze na mechanikách dosahovala jen stěží. Momenty jako bitva s vlkem Sifem, který byl naším spolubojovníkem v *Dark Souls*, nebo rozhodování, kteří z našich přátel se dostanou na svobodu a kteří zůstanou v očistci Downside ve hře *Pyre*, se svojí silou snadno vyrovnají nejslavnějším momentům filmové historie. Příběh může být dobrým motivačním faktorem, jako v případě Maria, který chce vysvobodit zajatou princeznu. Může také dodávat strukturu a vylepšit tempo hry, například prokládání akčních sekvencí dialogy. Role příběhu v herním průmyslu je tedy nesporná.

1.3.1 Typy příběhů

Fakt, že je vyprávěn v interaktivním médiu, ještě neznamená, že je příběh interaktivní. Příkladem hry s neinteraktivním příběhem může být *Warcraft III*. V této hře je vždy hráči přehrána animovaná sekvence vyprávějící část příběhu a následně je postaven do hratelné mise, kde přebírá otěže mechanická stránka hry a která končí binárním stavem – vítězství/prohra[11].

I hry, jejichž příběh je interaktivní a větví se, dosahují různých úrovní tohoto větvení. Mnohdy má příběh jednu kritickou cestu, kterou musí hráč projít a z níž se pouze krátce odchyluje do příběhových větví, které se s ní po čase opět spojí. Pokud hra obsahuje krátké volitelné příběhy, často může implementovat rozvětvený narativ u nich, aniž by byla ovlivněna kritická cesta. Časté

[8] Mimo jiné jde i o kultovní klasiky jako jsou právě *Adventure*, *Zork* nebo *The Hitchhiker's Guide to The Galaxy*.

[9] jako například v případě hry *Pong*

[10] postup zvolený textovými adventurami

[11] I v této části se objevují dialogy nebo jiné příběhové sekvence, pointou však je, že příběh zůstává stejný, ať už hráč hraje jakkoli.

bývají též hry s několika konci, mezi nimiž je vybráno podle hráčova chování. Tento přístup je vlastně lineárním příběhem s jedním rozvětvením na konci. Pro složitější hry, které nejsou nutně narativně zaměřené, není obvykle výhodné implementovat široce rozvětvený příběh, neboť to znamená, že hráč neuvidí všechen obsah hry, jehož produkce je o mnoho dražší než v případě textových adventur.

Tyto hry však mohou dosáhnout úplně jiného stupně interaktivity – emergentního příběhu[12]. Příběhy tohoto typu nebyly předem navrženy designérem, vznikají totiž interakcí herních systémů s hráčem a navzájem mezi sebou. Jde například o pamětihodnou první noc ve hře *Minecraft*, kdy se hráč vyděšeně ukrývá ve vykopané jámě před monstry, nebo o chaos způsobený divokým tygrem, který napadne tábor hráčových nepřátel ve hře *Farcry*. Podobným způsobem lze dosáhnout i příběhu, který se navenek jeví jako rozvětvený, například použitím systému frakcí, jejichž vztah s hráčem se mění podle jeho akcí a které se k němu chovají různě podle tohoto vztahu.

Speciálním typem je narativ procedurálně generovaný, který využívá například hra *Dwarf Fortress*. Ta vždy náhodně vygeneruje herní svět s celými civilizacemi, historií a rozsáhlou mapou, ve kterém následně hráč prožívá bohaté příběhy.

1.3.2 Výhody interaktivity

Jednou z největších výhod, které mají hry oproti filmům, je vztah jejich konzumenta a protagonisty. V případě filmu sledujeme hlavní postavu. V případě hry se touto postavou staneme. Ovládáme ji, vidíme a prožíváme to, co ona, a postupně si k ní vytvoříme pouto, které je specifické pouze pro toto médium. Začneme promítat vlastní osobnost do této postavy a hlavně vnímat emoce, které vnímá ona.

Ve filmovém průmyslu platí pravidlo „show, don't tell“, které poukazuje na to, že by filmy měly využívat výhodu, kterou mají před literaturou. Herní ekvivalent tohoto pravidla je „play, don't show“ – nestačí, aby hráč události viděl, musí si je zažít. To je to, co hry nabízí. Autentické zážitky prožité v cizí kůži.

Další nezanedbatelnou předností je fakt, že každý průchod hrou je jiný. I když je příběhová interaktivita minimální, mechanická interaktivita stále poskytuje nový zážitek. Hry zde tedy získávají, co se týče opakované konzumace, nádskok nad filmy, které jí musí dosahovat zejména postupným prohlubováním divákova chápání děje. Tato odlišnost každého zážitku se hrou zároveň vede na určitou intimitu, neboť žádný člověk s ní nebude mít stoprocentně stejnou zkušenost jako někdo jiný.

1.3.3 Nevýhody interaktivity

Přestože interaktivita může narativ obohatit, pojí se s jejím využití několik problémů. Jde zejména o případy, kdy jde příběh proti hratelnosti. Častým příkladem této chyby jsou dlouhé animované sekvence, které hráč přeskočí, je-li to možné, nebo frustrovaně přetrpí, pokud to možné není. Rozhodnutí odebrat hráči možnost interagovat by mělo být vždy učiněno velmi obezřetně. Hry jako *Bioshock* nebo *Dead Space* například přenechávají hráči řízení i během důležitých příběhových momentů.

Další problém se pojí s narušením hráčovy imerze v hlavní postavě. Čím více prostoru mu necháme, tím pravděpodobněji udělá hráč něco, co se neshoduje s charakterem postavy, čímž je poškozeno jeho vžití do ní. Tato překážka se také projevuje při neshodě mezi znalostmi hráče a hlavní postavy. Protagonista, který žil v herním světě mnoho let, nebude potřebovat vysvětlovat, kde leží které město, hráč však tyto informace nemá[13].

Při psaní této sekce bylo čerpáno z následujících zdrojů [17], [18].

[12] z anglického emergent story, volně přeložitelné jako samovolně vznikající nebo vynořující se příběh

[13] Z nutnosti řešit tento problém vzniklo herní klišé protagonisty, který trpí amnézií.

Kapitola 2
Teorie grafů

V této kapitole si definujeme základní pojmy teorie grafů a vysvětlíme si, jak ji můžeme aplikovat na interaktivní příběh.

Tvorba nelineárního příběhu s sebou přináší mnohé těžkosti, se kterými se běžný spisovatel nesetkává. Jde zejména o náročnost organizace takového počinu. Autor může zvolit postup vycházející z psaní klasické literatury, tedy že nejprve sepíše příběh pro jednu sérii rozhodnutí a následně se postupně vrací ke každému větvení a píše další paralelní příběhy. Alternativně může zvolit postupovat „záplavově“ a dopisovat vždy za sebou všechny pasáže, které vychází ze stejného předchůdce, než se přesune do větší hloubky příběhu. Případně může použít jakoukoliv variantu mezi těmito dvěma extrémy.

Problém prvního přístupu je, že pasáže, které se liší jen jedním rozhodnutím, jsou často podobné a operují se stejnými prvky narativu. Vracet se tedy do dané části příběhu, rozpomínat se na atmosféru, prostředí, rozpoložení postav a podobně, je pro autora nepochybně náročnější, než popsat několik variant dané situace, dokud jí má v živé paměti. Problém se druhým přístupem je, že autor ztratí smysl posloupnosti popisovaných událostí a snadno dojde k dezorientaci ve spletitém příběhu. Kromě toho musí mít spisovatel v obou případech přehled o velkém množství větví, aby na žádnou nezapomněl, a mít vyřešené číslování pasáží.

Je možné přijít s různými pomůckami, jak si tuto práci ulehčit. Některé větve se mohou spojit zpět do jedné a snížit tak počet pasáží, kterými se autor musí zabývat. Stejného efektu lze docílit vysokým množstvím pasáží, které hru ukončí[1]. Případně je možné členit pasáže do kapitol a napomoct si tak v jejich organizaci a v rozhodování, které pasáži se věnovat dál.

Všechny tyto berličky nám napomohou zorientovat se v rozpracovaném příběhu, pokud však pracujeme na rozsáhlejším díle, zjistíme, že ani s jejich využitím se nevyhneme rychlému nárůstu nepřehlednosti. Musíme tedy zvolit úplně jiný pohled na interaktivní příběh. Tento pohled nám nabízí teorie grafů.

2.1 Neorientovaný graf

Teorie grafů vznikla ve 30. letech 18. století, když matematik Leonhard Euler vyřešil takzvaný „Problém sedmi mostů města Královce“. Zabývá se modelováním reality jako množiny objektů a vztahů mezi těmito objekty. Konstrukt, který tímto popisem získáme, se nazývá graf a definujeme jej následovně:

[1] ať už pozitivně nebo negativně

▶ **Definice 2.1.** *Neorientovaný graf je uspořádaná dvojice (V, E), kde V je neprázdná konečná množina vrcholů a E je množina hran. Hrana je neuspořádaná dvojice různých vrcholů, značíme ji {u, v}.*

Takto definovaný graf můžeme snadno zobrazit pomocí geometrických tvarů představujících vrcholy a spojnic mezi těmito tvary představujících hrany a právě toto zobrazení nám pomůže popasovat se s náročností tvorby IF. Využijeme totiž grafu k popisu rozvětveného příběhu.

Analogicky ke grafu se nelineární narativ skládá ze dvou typů elementů. Z částí příběhu, které jsou hráči vyprávěny – pro tyto části budeme používat termín pasáž, a z možností, pomocí nichž se hráč mezi pasážemi pohybuje - pro ně si stanovíme termín volba. Budeme-li k pasážím přistupovat jako k vrcholům a k volbám jako ke hranám grafu, jsme schopní vytvořit grafické znázornění příběhu a umožnit tak jeho autorovi snadnou orientaci při tvorbě. Aby však zobrazení pomocí grafu odpovídalo našim potřebám, musíme si dodefinovat několik termínů.

2.2 Orientovaný graf

Neorientovaný graf nám umožňuje definovat hrany. Tyto hrany však mají k oběma vrcholům, jež spojují stejný vztah. Pro účely IF by to znamenalo, že pokud hráč učiní rozhodnutí, které ho dovede k určité pasáži, může následně zvolit stejné rozhodnutí, aby se dostal zpět do pasáže předchozí. Abychom však grafem mohli modelovat příběh, potřebujeme, aby hrana z jednoho vrcholu vycházela a do druhého vstupovala. Typ grafu, jehož hrany se takto chovají, nazýváme orientovaný graf:

▶ **Definice 2.2.** *Orientovaný graf G je uspořádaná dvojice (V, E), kde V je neprázdná konečná množina vrcholů a E je množina orientovaných hran. Orientovaná hrana (u, v) ϵ E je uspořádaná dvojice vrcholů u, v ϵ V. Říkáme, že u je předchůdce v a v je následník u.*

2.3 Ohodnocený graf

Vrcholy v běžném grafu jsou prvky určité množiny. Tato definice nám umožňuje, abychom jako vrcholy využili pasáže našeho příběhu. Hrany a volby ovšem tuto možnost neskýtají, neboť hrana námi definovaného grafu je pouze uspořádaná nebo neuspořádaná dvojice vrcholů, kdežto volba s sebou nese navíc jistou informaci, obvykle textového charakteru. Opět tedy rozšíříme naši definici grafu:

▶ **Definice 2.3.** *Mějme libovolnou množinu I. Ohodnocený graf G je uspořádaná trojice (V, E, f), kde V je neprázdná konečná množina vrcholů, E je množina hran – dvojic vrcholů a f: E → I je funkce, která každé hraně přiřadí prvek z množiny I.*

Hovoříme-li tedy o grafu jako o ohodnoceném, znamená to, že ke každé jeho hraně přiřadíme určitý prvek z určité množiny. Obvykle se setkáme s ohodnocenými grafy, které hranám přiřazují prvky z množiny reálných nebo celých čísel, pro naše účely však necháme tuto množinu nespecifikovanou, což nám umožní přiřazovat hranám text. V definici ohodnoceného grafu také nezmiňujeme, zda je hrana dvojicí uspořádanou nebo nikoli, což implikuje existenci jak neorientovaného ohodnoceného grafu tak grafu orientovaného ohodnoceného.

2.4 Multigraf

Ohodnocený orientovaný graf nám umožní reprezentovat libovolný příběh[2]. Z hlediska UX[3] nejde ale o ideální reprezentaci. Uvažme situaci, kdy jedna pasáž nabízí více voleb, které vedou do

[2] Jsme schopni definovat pasáže a volby, ze kterékoli pasáže se můžeme dostat do kterékoli jiné a opakování pasáže můžeme docílit její duplikací.

[3] User Experience – jak je pro uživatele jednoduché a příjemné dosáhnout v aplikaci svého cíle.

stejné pasáže. Pokud bychom chtěli takovýto případ popsat ohodnoceným orientovaným grafem, museli bychom vrchol reprezentující následující pasáž vytvořit vícekrát, jeho definice nám totiž neumožňuje, aby bylo mezi dvěma vrcholy nataženo více hran než jedna[4]. Takové řešení je značně neintuitivní a proto se nám vyplatí povolit více hran mezi stejnými vrcholy. Tuto možnost nám poskytne multigraf:

▶ **Definice 2.4.** *Multigraf G je uspořádaná dvojice (V, E), kde V je neprázdná konečná množina vrcholů a E je multimnožina*[5] *hran. Hrana (u, v) ϵ E je dvojice vrcholů u, v ϵ V.*

Stejně jako definice ohodnoceného grafu nám tato definice neurčuje, zda jde o multigraf orientovaný nebo neorientovaný. Ohodnocený multigraf definujeme analogicky k ohodnocenému grafu přidáním ohodnocující funkce *f*. Struktura, kterou využijeme pro reprezentaci vytvářeného příběhu, bude tedy ohodnoceným orientovaným multigrafem, abychom zachovali veškerou informaci narativu a zároveň se vyhnuli zbytečným složitostem pro jeho tvůrce.

2.5 Stupeň vrcholu

Pasáže nabývají v kontextu příběhu různých funkcí. Když hráč hru poprvé spustí, měla by se mu vždy objevit stejná úvodní pasáž. Tuto pasáž budeme nazývat prolog.

Naopak ve chvíli, kdy se hráč dostane na konec příběhu, kdy už mu nejsou nabízeny další volby[6], je mu zobrazena pasáž, kterou budeme označovat jako epilog. Epilogů může příběh obsahovat více a právě tato skutečnost je častým lákadlem IF i her s rozvětveným příběhem obecně, neboť podporuje znovuhratelnost a experimentování s různými způsoby, jak hru hrát.

V převážné většině IF tvorby ovšem převládají pasáže, které nejsou ani prologem ani epilogem. Tyto pasáže mají minimálně jednoho předchůdce a minimálně jednoho následovníka. Pasáž tohoto typu budeme nazývat běžná pasáž.

Abychom byli schopni formálně zkoumat rozdíly mezi těmito typy pasáží, využijeme dalšího termínu z teorie grafů - stupně vrcholu:

▶ **Definice 2.5.** *Nechť G = (V, E) je graf a v ϵ V jeho vrchol. Stupněm vrcholu v nazveme počet hran grafu G obsahujících vrchol v a označíme ho jako* $deg_G(v)$.

▶ **Definice 2.6.** *Vstupní stupeň vrcholu v v orientovaném grafu G značíme* $deg^+_G(v)$ *a definujeme následovně:* $deg^+_G(v) = |\{u \mid u \epsilon\ V \wedge (u, v) \epsilon\ E\}|$

▶ **Definice 2.7.** *Výstupní stupeň vrcholu v v orientovaném grafu G značíme* $deg^-_G(v)$ *a definujeme následovně:* $deg^-_G(v) = |\{u \mid u \epsilon\ V \wedge (v, u) \epsilon\ E\}|$[7]

S použitím této terminologie můžeme ustanovit prolog jako vrchol v takový, že $deg^+_G(v) = 0$, epilog jako vrchol v takový, že $deg^-_G(v) = 0$, a běžnou pasáž jako vrchol v takový, že $deg^+_G(v) > 0 \wedge deg^-_G(v) > 0$. Zkoumáním vstupních a výstupních stupňů pasáží v grafu můžeme odhalit, nakolik se příběh větví, nakolik se tyto větve opět spojují, nebo které části příběhu nechávají hráči větší možnost volby a které naopak probíhají vždy podobně pouze s drobnými odchylkami.

Zajímavým příkladem je pasáž jejíž $deg^-_G(v) = 1$. Z hlediska větvení a hráčské interakce by byl příběh ovlivněn stejně, kdybychom takovou pasáž spojili s pasáží následující a zobrazili texty obou pasáží najednou. Přesto však využití těchto pasáží povolíme, neboť efekt na hráče se liší od toho, kdyby byly spojeny do většího celku.

Rozdělením jedné dlouhé pasáže do více kratších s jedinou volbou zjednodušíme hráčovi čtení a zároveň si uchováme jeho pozornost tím, že po každé části takového delšího textu vyžadujeme

[4] Více pasáží se stejným textem definice nevylučuje, neboť je můžeme odlišit například pomocí ID.
[5] obdoba množiny, která povoluje vícenásobný výskyt prvků
[6] s případnými rozšiřujícími výjimkami jako je restart nebo krok zpět
[7] V případě multigrafu rozšíříme definice vstupního a výstupního vrcholu tak, aby počítaly velikost multimnožiny namísto množiny.

jeho interakci, byť tato interakce nezahrnuje rozhodnutí mezi více volbami ale pouze potvrzení jediné možné volby. Pasáže s jedinou volbou mohou mít také efekt na hráčovy emoce. Tím, že ho připravíme o možnost určovat, kam jeho příběh směřuje můžeme navodit pocit bezmoci nebo vytvořit rozkol mezi hráčem a postavou, kterou ovládá. Příkladem může být příběh, kdy je hlavní postava zhypnotizována a nemůže ovládat své jednání, když je závislá na návykových látkách a nedokáže již této závislosti odolávat, nebo když hráč protagonistu neovládá přímo, ale například mu jen dává pokyny přes vysílačku, a on je odmítne poslouchat.

2.6 Sled a cesta

Čtením pasáží a výběrem voleb se hráč pohybuje větvemi příběhu. Pomocí stupňů vrcholů jsme schopni zkoumat význam jednotlivých pasáží, abychom však mohli sledovat hráčův postup příběhem, musíme být schopni zkoumat sekvence těchto pasáží. K tomuto účelu si definujeme následující termíny:

▶ **Definice 2.8.** *Sled v_0v_n v grafu G = (V,E) je posloupnost (v_0, e_1, v_1, e_2,..., e_n, v_n) taková, že: $\forall i \in \langle 0;n \rangle$: $v_i \in V$, $\forall i \in \langle 1;n \rangle$: $e_i \in E$ a $\forall i \in \langle 1;n \rangle$: $e_i = (v_{i-1},v_i)$. Počet hran v této posloupnosti nazveme délkou sledu v_0v_n. Vrchol v_0 nazýváme počáteční a vrchol v_n koncový.*

▶ **Definice 2.9.** *Sled v_0v_n v grafu G nazveme cestou v_0v_n v grafu G, pokud platí: $\forall i \in \langle 0;n \rangle$, $\forall j \in \langle 0;n \rangle$: $i \neq j \implies v_i \neq v_j$.*

▶ **Definice 2.10.** *Graf G je souvislý, jestliže v něm pro každé jeho dva vrcholy u, v existuje cesta uv. Jinak je G nesouvislý.*

Snadno si domyslíme, že nesouvislý graf není pro účely popisu příběhu vhodný. Jestliže není možné najít cestu z prologu, do některé jiné pasáže, nejedná se totiž o jeden příběh, ale o více nezávislých příběhů. V takovém případě nemá smysl popisovat je jedním grafem. Abychom však mohli vyžadovat souvislost dokončeného narativního grafu, musíme ji být schopni rozpoznat. Jak vyšetřit souvislost grafu a jaká úskalí v tomto vyšetřování vyplynou s implementací některých rozšíření našeho nástroje, si popíšeme v kapitole 8.

Cesta v grafu je tedy posloupnost vrcholů a hran, jejíž prvky se neopakují. Pro naše účely je však zajímavá i varianta, kdy se vrcholy ve sledu opakují. O takovém sledu tvrdíme, že obsahuje cyklus:

▶ **Definice 2.11.** *Sled v_0v_n v grafu G nazveme cyklem v_0v_n v grafu G, pokud platí: $\forall i \in \langle 1;n \rangle$, $\forall j \in \langle 1;n \rangle$: $i \neq j \implies v_i \neq v_j \wedge v_0 = v_n$.*

Kromě existence cyklů nám naše definice grafu povoluje i používání smyček tedy hran *(u,v)*, pro které platí, že $u = v$. Podobně jako v případě multigrafu je toto počínání motivováno zlepšením UX. Autor může například chtít, aby se hráči zobrazovala stejná pasáž, dokud si vybírá volbu A a donutit ho tak zvolit volbu B, která mu dovolí pokračovat. Bez použití smyček by se taková situace opět musela vyřešit duplikací dané pasáže a natažením stejné volby mezi těmito duplikáty v obou směrech. Bez použití cyklů by pak byla neřešitelná. Při psaní této kapitoly byly využity následující zdroje [19], [20], [21].

Kapitola 3
Funkční požadavky

V následující kapitole si vydefinujeme funkcionalitu, které budeme chtít v rámci této práce dosáhnout. Výsledná aplikace bude prototypem a nebude tedy poskytovat všechny funkce, které by finální řešení poskytovat mělo. Bude na ní však vidět, jakým způsobem by se dalo rozšířit dál. Přesná definice funkčních požadavků nám po dokončení implementace umožní zhodnotit, jak jsme naplnili cíle práce.

Cílem práce je vytvořit prototyp nástroje pro tvorbu choice-based text adventure her. Důležité pro nás budou zejména tři aspekty takového nástroje – snadná tvorba, snadné publikování a snadná rozšiřitelnost. Vycházejíce z tohoto cíle a z těchto tří priorit si přesně určíme, jakou funkcionalitu musí prototyp poskytovat.

3.1 Editační rozhraní

Zásadní částí aplikace je grafické editační rozhraní, které umožní uživateli tvorbu. K reprezentaci díla využijeme orientovaný ohodnocený multigraf, jehož výhody byly popsány v předchozí kapitole a který umožní snazší tvorbu CBTA než obyčejné textové rozhraní. Nástroj tedy musí být schopný zobrazit tento graf a umožnit autorovi přidávání a odstraňování vrcholů a hran.

Tyto vrcholy a hrany ponesou textovou informaci. V případě hran jde o text volby, v případě vrcholů pro lepší orientaci v grafu umožníme zadání titulku a textu pasáže. Titulek pasáže a text volby bude možné editovat po dvojkliku na vrchol/hranu, text pasáže však může být poměrně dlouhý a takový postup by nemusel být vhodný. Proto musí být součástí editačního rozhraní panel zobrazující detail vrcholu/hrany, který se objeví po označení daného elementu a který o něm bude obsahovat informace. V případě vrcholů bude pak možné v jejich detailu editovat text dané pasáže.

Kromě těchto funkcí, jenž jsou pro tvorbu grafu nezbytné, stanovíme také několik požadavků, jejichž účelem bude tvorbu usnadnit. Zejména chceme, aby měl uživatel možnost zobrazení grafu si zorganizovat. Musíme mu proto umožnit pohybovat s vytvořenými vrcholy a hranami a měnit velikost vrcholů. Dále umožníme kopírování, vystřihování a vkládání vrcholů a hran, obnovení stavu před poslední úpravou a zpět po ní, přibližování a oddalování. Pro lepší preciznost a snadný pohyb grafem bude editační rozhraní doplněno o panel se zmenšeným náhledem grafu, který uživateli umožní rychlý pohyb tažením myši.

Při přidávání vrcholu bude mít uživatel na výběr, zda vrchol reprezentuje běžnou pasáž, prolog nebo epilog. Z hlediska textových dat se budou tyto tři typy vrcholů chovat stejně, budou však graficky odlišeny. Engine bude dále kontrolovat, zda nebyla porušena omezení spojená s prologem a epilogy. Tedy alespoň počty výchozích/vstupních hran.

3.2 Ukládání a publikování

Náš nástroj musí být dimenzován na tvorbu rozsáhlých děl IF. Není tedy možné očekávat, že bude autor schopný celý výtvor dokončit bez přestávek a naráz. Proto musí být možné nehotovou práci uložit a později opět načíst. Ukládání by mělo probíhat buďto do souboru v uživatelově zařízení, nebo jako rychlé uložení do paměti a načtení z paměti webového prohlížeče, bude-li mít výsledný nástroj podobu webové aplikace. Není nutné, aby byl ve fázi prototypu obsah souboru šifrován nebo jinak obfuskován.

Poté, co autor dokončí svou práci, bude mít možnost dílo publikovat. Výsledná CBTA by měla být vyexportována do takového formátu, aby byla náročnost jejího spuštění pro hráče minimální. Je tedy nutné, aby nástroj nabízel vlastní rozhraní pro hráče, které jim umožní výsledný příběh projít, nebo aby proběhl export do formátu, který umožňuje jeho hraní pomocí veřejně přístupných aplikací jako je například webový prohlížeč.

Nástroj by měl při exportu nabízet možnosti přizpůsobení grafického vzhledu díla. Tohoto přizpůsobení bude ve fázi prototypu dosaženo nabídkou alespoň tří předdefinovaných stylů, z nichž si bude autor moci vybrat.

3.3 Rozšíření

Poslední prioritou, kterou jsme se doposud nezabývali, je snadná rozšiřitelnost. Tomu, jak jí docílíme, se budeme věnovat v kapitole věnované návrhu architektury. Prozatím si však stanovíme, jakým způsobem dokážeme, že výsledný prototyp kritérium snadné rozšiřitelnosti opravdu splňuje. Tímto důkazem bude implementace vlastního rozšíření.

Několik možností, jak engine obohatit o novou funkcionalitu, si popíšeme v kapitole Budoucí rozšíření. V rámci prototypu si však postačíme s přidáním funkce kontrolních bodů – takzvaných checkpointů. Toto rozšíření umožní uložit hráčův postup v určitém bodě, tak aby se v průběhu hry mohl vrátit do takto uloženého stavu, například když se dostane do epilogu, se kterým není spokojený. Tvůrce bude schopný definovat, které pasáže jsou kontrolními body, tak že místo běžné pasáže přidá do grafu speciální typ vrcholu.

Kapitola 4
Analýza konkurence

Cílem této části je prozkoumat nabídku alternativních řešení pro tvorbu interaktivní fikce. U každého si rozebereme jeho nedostatky a jak se může naše řešení odlišit. Zároveň si zanalyzujeme silné stránky těchto konkurenčních řešení a co se od nich můžeme naučit.

Jak si ukážeme v této kapitole, aspirující autor IF má k dispozici pestrou nabídku nástrojů, ze které vybírat. Mnoho z nich jsou nezávislé projekty a jejich kvalita se různí. Chceme-li vyvíjet vlastní IF engine, měli bychom se v této nabídce orientovat a ujistit se, že neimplementujeme již existující řešení.

Konkurenční řešení si rozdělíme na tři skupiny, které poté zvlášť prozkoumáme. v první řadě se budeme zabývat nástroji, které naplňují stejný účel, jenž chceme naplnit my. Půjde tedy o enginy specializované na tvorbu CBTA her, ale také hypertextové fikce, která se od CBTA, co do implementace, odlišuje pouze v detailech.

Následně rozebereme nástroje, které mají podobný účel. Do této kategorie spadají enginy specializované na jiné žánry IF, ale například také nástroje, které sice samy o sobě neumožňují vytvořit IF hru, ale pomáhají organizaci interaktivního narativu v rámci složitějších her.

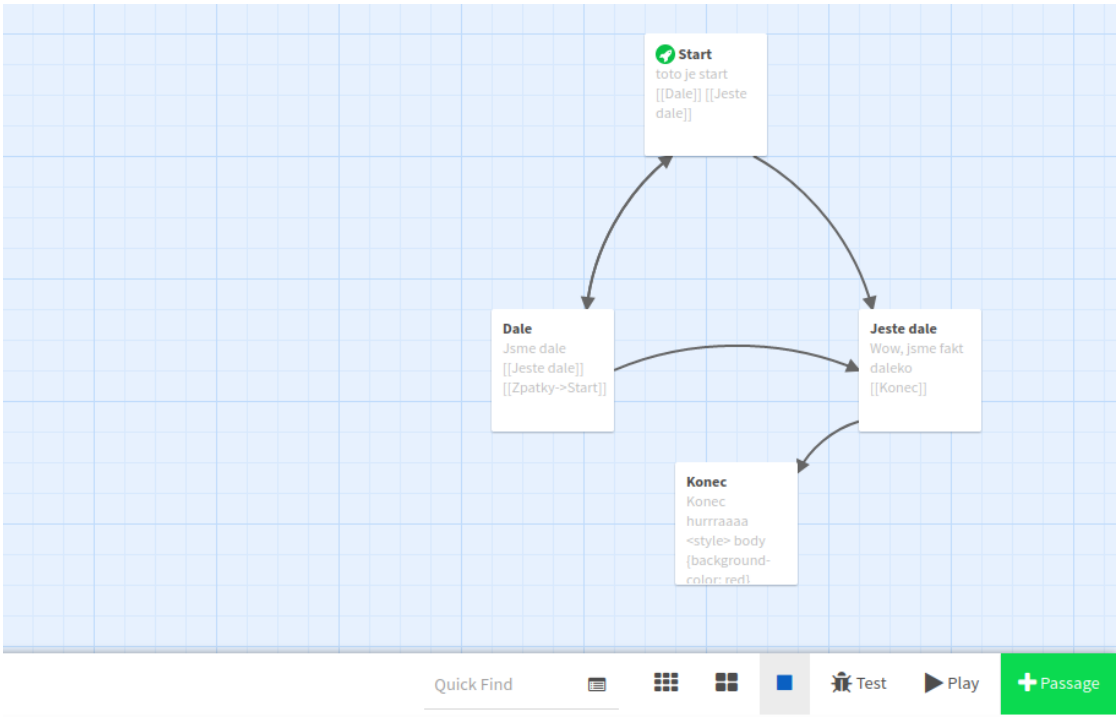
Poslední skupinou budou nástroje, jejichž účel je obecnější než tvorba IF, přesto ji však umožňují. Zde půjde především o všeobecné herní enginy nebo o programovací jazyky.

4.1 CBTA a HF nástroje

Ve 20. století byl trh digitální IF dominován parser-based tituly. 21. století však přineslo nový trend spojený s již popsaným rozmachem nezávislé tvorby. Mnohé nástroje podporující tuto tvorbu se totiž zaměřovaly právě na CBTA a HF. Jelikož předmětem této práce je návrh a implementace prototypu CBTA enginu, jsou pro nás tyto nástroje přímou konkurencí a měli bychom jim věnovat největší pozornost.

4.1.1 Twine

Z nástrojů tohoto typu je nesporně nejpopulárnější *Twine* [22], [23]. Vyvíjený od roku 2009 a podporovaný neziskovou organizací *IFTF* se stal *Twine* zásadním pilířem komunity tvůrců IF a dal vzniknout mnoha kreativním a dobře hodnoceným výtvorům. Zaměřuje se na tvorbu HF, využívá zobrazení projektu pomocí grafu a umožňuje publikování díla ve formátu HTML. Jak svým účelem, tak způsobem, kterým tento účel naplňuje, má tedy *Twine* nejblíže k nástroji, který bude produktem této práce. Tento překryv funkcionality a zároveň fakt, že jde o populární volbu, jsou důvody, proč se na analýzu tohoto enginu zaměříme nejpodrobněji.



Obrázek 4.1 Uživatelské rozhraní *Twinu*

Začít pracovat s *Twine* je velmi jednoduché. Je možné ho spustit na *Windows*, *macOS* i na *Linuxu*, zároveň však nabízí online verzi, nový uživatel tedy nástroj ani nemusí instalovat[1]. Samotná tvorba si udržuje tento snadný přístup. Autor přidává pasáže jedním kliknutím a dvojklikem na odpovídající vrchol v grafu se dostane do dialogu, kde je může upravovat a vytvářet z nich vycházející volby. Dílo je možné exportovat do HTML souboru a tento jednoduše spustit. Kromě toho umožňuje *Twine* spustit příběh z libovolného vrcholu v průběhu psaní, měnit startovní vrchol, sledovat statistiky jako počet slov a pasáží, zvýraznit pasáže, které obsahují hledaný text a nastavit, zda se u pasáží bude zobrazovat jen titulek, titulek s částí textu, nebo nic.

Pro expertnější uživatele jsou však možnosti *Twine* mnohem obsáhlejší. Pasáže totiž nemusí obsahovat pouze čistý text. Uživatel disponující znalostí jazyků HTML, CSS a JavaScript je schopen příběh obohatit o obrázky, zvuky, formátovat text nebo přidávat skripty, které se spustí se spuštěním příběhu nebo s otevřením pasáže. Příběh vytvořený v tomto nástroji je při exportu zkompilován a následně interpretován pomocí takzvaných Story Formats – run-time enginů, které se v JS formě spustí s otevřením výsledného HTML souboru. Story Format se stará o navigaci mezi pasážemi a může nabízet další funkce, jako je automatické přidání tlačítka zpět, nebo interpretace maker, které autor vloží do pasáží. *Twine* nabízí čtyři základní „Příběhové formáty“ – *SugarCube*, *Harlowe*, *Snowman* a *Chapbook*, a volba mezi nimi zásadně ovlivní, jaké funkce jsou uživateli přístupné. Kromě toho existuje mnoho formátů třetích stran poskytujících ještě větší variabilitu funkcí.

Twine je tedy snadno dostupný, snadno použitelný a jeho výstup je snadno publikovatelný. Navíc tato jednoduchost použití nebrání existenci velkého množství rozšiřujících funkcí, které pro uživatele s hlubší znalostí poskytují komunitně vyvíjené překladače. Jaký je tedy důvod, aby existoval další podobný nástroj?

[1] Část tohoto online rozhraní vidíme na obrázku 4.1.

V první řadě jde o UX. *Twine* sice využívá grafovou reprezentaci pro zobrazení struktury příběhu, uživatelova schopnost manipulace s tímto zobrazením je však omezena na přesouvání vrcholů a výběr ze čtyř přednastavených velikostí. Hrany není možné upravovat nijak a jejich přidávání probíhá textově, což je zbytečně náročné. *Twine* také nenabízí náhledový panel k orientaci a přibližování je podobně omezeno jako nastavování velikostí vrcholů, což způsobuje náročnější orientaci ve velkém rozvětveném grafu.

Další nedostatky nevychází z chybějící funkcionality, ale z uživatelsky nepříjemného návrhu. Jedná se zejména o nutnost otevírat dvojklikem dialog pasáže pokaždé, když chce uživatel upravit její text nebo i text jejího titulku a voleb. Zaměření na HF znamená, že vytváření voleb je složitější. Musí se zde vepsat do textu pomocí speciálních znaků. *Twine* po sepsání volby do textu pasáže automaticky vytvoří novou pasáž propojenou hranou, jejíž titulek je textem dané volby. Takové chování obvykle nebude to, co uživatel vyžaduje, neboť text volby většinou není dobrým shrnutím obsahu následující pasáže. V tomto případě musí uživatel v textu krom volby specifikovat název následující pasáže. Samotné nástrojové menu je dále nepřehledné, obsahuje spoustu volného místa, i když většina voleb je skryta pod nedostatečně označeným rolovacím tlačítkem.

Závažnějším problémem je však přístup k pokročilé funkcionalitě. Autor, který chce obohatit svůj příběh o tyto funkce, má dvě možnosti. Buďto stráví čas zkoumáním Story formátů a hledáním takového, který nabízí vhodná makra, nebo se naučí programovat v JS a kód si napíše sám. Samotné použití makra pak často může být náročné, obsahuje-li zvláštní parametry. Kdyby byl namísto toho uživateli nabídnut speciální typ vrcholu pro danou funkci, postup jeho práce by se změnil minimálně a byl by odstíněn od vnitřního fungování nástroje.

Možnost doplnit do hry vlastní CSS styl a vlastní skripty je sice vítaná, avšak v případě že by byl výstup aplikace reprezentován ve srozumitelné formě, nebyl by pro uživatele problém doplnit tyto soubory po exportu. Postup aplikovaný enginem *Twine*, tedy vložení všech dat do jednoho souboru a následně organizace jejich prezentace dodaným skriptem, způsobuje nečitelnost pro člověka, která vzhledem k open source povaze tohoto nástroje není nutně pozitivem. Naopak při neznalosti jazyka CSS není uživatel schopen upravit vzhled prezentace a je nucen využít výchozí styl – bílé písmo na černém podkladě.

Problémy, které jsme zde identifikovali, nelze opravit pouhým přidáním Story formátu. Z toho vyplývá, že má smysl vytvořit alternativní nástroj, který se bude více zaměřovat na příjemnost používání. Pokud chceme, aby se tento nástroj dostatečně odlišil od *Twine*, který mezi IF enginy dominuje, musí umožnit přístup k pokročilé funkcionalitě i uživatelům, kteří jsou technicky méně zdatní. Zároveň by neměl být podceněn vliv špatného UX – nového uživatele množství takových problémů snadno odradí a dlouhodobý uživatel je sice začně přehlížet, přesto mu však opakovaně znepříjemňují práci.

4.1.2 ChoiceScript

ChoiceScript [24] je skriptovací jazyk vyvinutý společností *Choice of Games* pro tvorbu CBTA her. Kromě definování pasáží a voleb umožňuje *ChoiceScript* vkládat obrázky nebo nastavovat, upravovat a vypisovat hodnoty proměnných. Sám o sobě nenabízí grafické rozhraní a je dimenzován spíše na lineárnější narativy. Příběh je zde rozdělen na scény, které hra postupně prochází, hráčovo rozhodnutí v jedné scéně tedy často nastaví proměnnou, která ovlivní jinou scénu, příběh však většinou má podobnou strukturu nehledě na vybrané volby. Toto je vidět i na výsledné hře, kde se hráč nepohybuje výběrem volby, ale stisknutím tlačítka Next, kterému mohlo předcházet zatržení jedné z nabízených voleb. Toto zaměření na nepříliš rozvětvený příběh dává smysl, jelikož pouhý skriptovací jazyk, jak jsme si již rozebrali na začátku kapitoly o teorii grafů, není k popisu velmi rozvětvených příběhů vhodný. V případě *ChoiceScriptu* je toto omezení o to zřetelnější, že jednotlivé úrovně voleb se oddělují odsazením. Uživatel je tímto motivován nezanořovat se příliš hluboko a raději více střídat scény.

Přes toto omezení má *ChoiceScript* výraznou výhodu a tou je publikace her v něm napsaných. *Choice of Games* totiž autorům nabízí pod svojí značkou distribuci jejich výtvorů na nejrůznější platformy, mezi jinými jsou to *iOS*, *Android*, *Kindle* a *Steam*. *Choice of Games* se svými autory úzce spolupracuje a nabízí nejrůznější smlouvy obsahující zálohy, honoráře a intelektuální vlastnictví.

Hlavní výhodou našeho enginu oproti *ChoiceScriptu* jsou usnadnění spojená s grafickým rozhraním, která poskytuje tvůrci. Dále ho můžeme předčit v přístupnosti - uživatel *ChoiceScriptu* musí mít nainstalované prostředí *NodeJS*, stáhnout a extrahovat kód z *GitHub* repozitáře, spustit si lokální server a najít ve staženém adresáři soubory se skriptem, které teprve může upravovat. V oblasti publikace výtvorů nemá v tuto chvíli cenu *ChoiceScriptu* konkurovat.

4.1.3 Ink a Inklewriter

Dalším velkým hráčem na poli IF je společnost *Inkle* [25] z Velké Británie. Toto herní studio, které stojí mimo jiné za oceňovanou archeologickou detektivkou *Heaven's Vault*, pracuje také na nástrojích pro psaní CBTA her.

Prvním z jejich produktů je *Ink* – skriptovací jazyk pro popis rozvětvených narativů a s ním spojený editor *Inky*. *Ink* nabízí podobnou funkcionalitu jako *ChoiceScript* – psaní pasáží a voleb, používání proměnných a podmínek, přidávání obrázků, stylování pomocí CSS a skriptování pomocí JS. Stejně jako *ChoiceScript* naráží *Ink* na limity spojené s použitím skriptovacího jazyka namísto grafického rozhraní a podobně jako *ChoiceScript* používá větší skupiny pasáží, zde nazývané „knots“, aby problémy s přehledností zmírnil. Narozdíl od *ChoiceScriptu* se méně zaměřuje na sekvenční příběh, který volby příliš neovliňují, ale klade větší důraz na větvení. *Inky* je možné stáhnout z *GitHub* repozitáře a po rozbalení spustit, nebo je dostupný jako plugin k *Unity* enginu. Příběh psaný v *Inku* je možné exportovat jako JSON soubor nebo do podoby webových stránek.

Dalším nástrojem od společnosti *Inkle* je *Inklewriter*, který stejně jako *Ink* slouží ke psaní interaktivního narativu, narozdíl od něj jde však spíše o jakýsi skicák, který by měl posloužit například autorům složitějších her, aby si jejich příběh rozvhrli. Uživatelské rozhraní *Inklewriteru* je pokročilejší než obyčejný textový editor – odděluje jednotlivé pasáže a přehledně ukazuje, které volby ještě nemají sepsanou následující pasáž. Neumožňuje však přehledně zobrazovat příběh rozvětveně, pouze jednu z jeho větví a volby z ní vycházející. *Inklewriter* je dostupný formou webové aplikace a umožňuje export příběhu do *Inku*.

Přestože nástroje od společnosti *Inkle* činí více kroků k přehlednosti než například *ChoiceScript*, způsob jejich používání se stále nevyrovná grafovému přístupu, který hodláme implementovat.

4.1.4 Další

Kromě těchto velkých jmen je svět IF plný menších nástrojů, které nabízí různé výhody a mají různá omezení. Jedním z nich je *Quest* [26]. Tento engine umožňuje vytvářet hry, jenž kombinují CBTA a PBTA. Jeho uživatel si definuje místnosti, objekty a postavy spolu s možnostmi, jak s nimi interagovat. Tyto možnosti poté hráč ve hře vidí a nemusí je hádat, jako je tomu v případě her používajících parser. Jde tedy o zajímavou alternativu, pokud se tvůrci zdají obyčejné CBTA hry moc jednoduché a PBTA moc netransparentní. Pro nováčka na poli IF je však *Quest* moc složitý.

Squiffy [27] se podobá *ChoiceScriptu* a *Inku*. Jde o skriptovací jazyk s HTML výstupem sloužící ke psaní CBTA. *Squiffy* je dostupné online a pro začínajícího autora je přístupnější než výše popsané alternativy, nenabízí však tolik možností a naráží na stejné problémy jako ostatní skriptovací jazyky.

Tabulka 4.1 Srovnání konkurenčních řešení

Funkce	*Vorpal*	*Twine*	*ChoiceScript*	*Ink*	*Inklewriter*
Grafové editační rozhraní	•	•			
Snadná publikace	•	•	•	•	
Podpora skriptování	•	•		•	
Snadno dostupné pokročilé funkce	•		•	•	
Uživatelská příjemnost	•		•		•
Dostupnost bez instalace	•	•			•
Podpora komunity		•	•		

Nástroje *Ramus* [28] a *Undum* [29] jsou HTML šablony s JS skripty, které si uživatel stáhne a vpisuje do nich příběh. Pro znalce HTML jde o rychlou alternativu ke složitějším nástrojům, postrádají však různá zjednodušení a rozšiřující funkce těchto enginů.

Zajímavým nástrojem je francouzský engine *Moiki* [30], který se v mnohém podobá *Twine*. *Moiki* vizualizuje příběh pomocí grafu, jehož zobrazení však uživatel nemůže nijak ovládat. Dále umožňuje spoustu funkcí jako jsou obrázky, zvuky nebo práce s proměnnými, narozdíl od *Twinu* jsou však tyto funkce zabudované do enginu a snadno přístupné z uživatelského rozhraní, není tedy nutné vyznat se v dalších technologiích jako je JS. Na druhou stranu se uživatel v sofistikovaném editačním rozhraní hůře vyzná a fakt, že *Moiki* používá graf pouze k zobrazení příběhu a ne k interakci s uživatelem, přispívá k náročnějšímu přidávání voleb a následujících pasáží. Dalším problémem tohoto enginu je neúplný překlad z Francouzštiny.

Podobný problém s překladem má italský *LibroGameCreator* [31]. Tento nástroj nenabízí velké množství funkcí nad rámec volení větví příběhu, na druhou stranu nabízí poměrně rozsáhlou funkcionalitu pro podporu této tvorby – například validaci příběhu nebo export do mnoha formátů, mimo jiné *ChoiceScriptu* a *Squiffy*. Schází mu však grafové zobrazení a dostatečná dokumentace a výukové materiály.

Posledním nástrojem, který si uvedeme je *GameBook Authoring Tool* [32] od nezávislého autora *Crumbly Head Games*. Tento engine nabízí jednoduché grafové zobrazení a poměrně příjemné uživatelské rozhraní. Schází mu však pokročilejší funkce a svobodnější práce s grafem. Dalším nedostatkem je, že plná verze tohoto programu je placená, což mu na trhu plném zdarma dostupných nástrojů s širší funkcionalitou nedává šanci.

V tabulce 4.1 vidíme přehledné srovnání výhod a nedostatků největších konkurenčních nástrojů[2]. Je z ní zřejmé, že *Twine* svojí kvalitou dominuje, vidíme však, kde ho můžeme předčit. *Ink* a *Inklewriter* sice samy o sobě splňují pouze pár kritérií, navzájem se ale dobře doplňují a zkombinujeme-li jejich skóre, vyrovnají se i enginu *Twine*. Největší slabou stránkou *Vorpalu* bude práce s komunitou, která nemá smysl, dokud nebude engine dostatečně kvalitní. V budoucnu jde však o důležitý aspekt, na který se zaměřit.

4.2 Ostatní IF nástroje

Ukázali jsme si tedy, že i mezi poměrně početnými nástroji, které nám přímo konkurují, se najde místo pro nový CBTA engine. Jak už ale víme, IF zahrnuje mnoho dalších herních žánrů a s těmito žánry se pojí mnohé další specializované enginy, které není radno přehlížet. Podíváme se tedy i na tyto nástroje.

[2] *Vorpal* zde, vzhledem k fázi svého vývoje, není hodnocen tak přísně jako ostatní nástroje. Chybí mu například jisté funkce, které jsou pro dosažení uživatelské příjemnosti potřeba, je však navržen od základu tak, aby bylo UX kvalitní, a doplnění těchto funkcí už je snadná záležitost, narozdíl od *Twine*, jehož UI by bylo třeba celé přestavět.

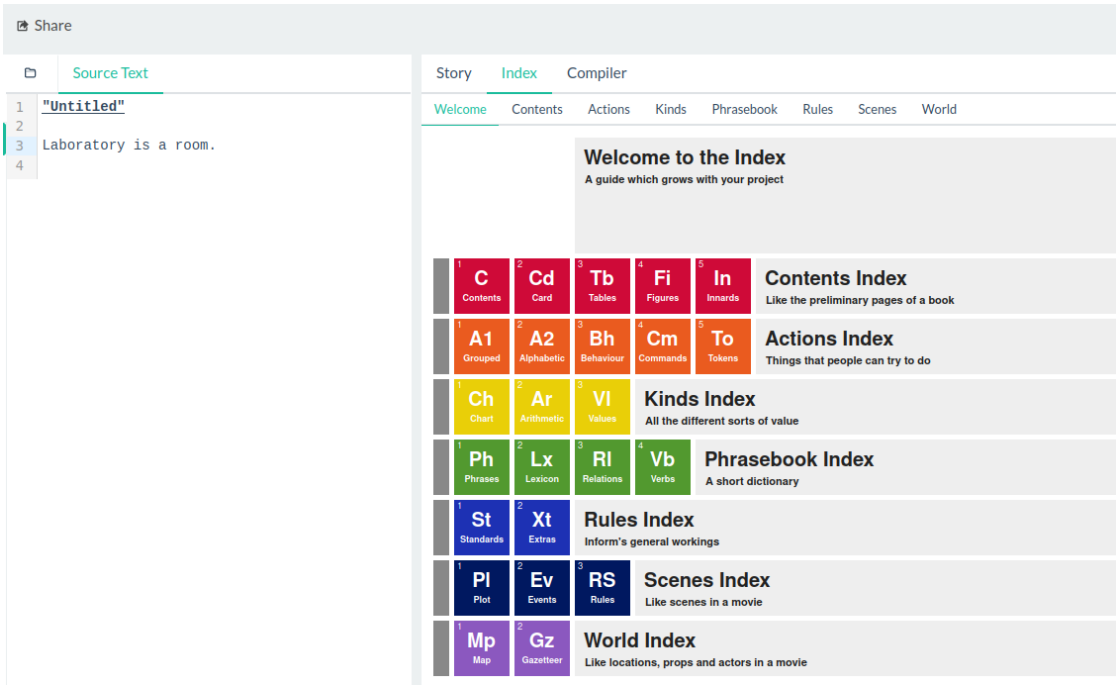
4.2.1 Inform a TADS

Přestože si CBTA a HF užívají v posledních letech roztoucího množství dedikovaných nástrojů, počátky digitální IF jsou spojeny s používáním parseru. Podíváme se proto na dva nejpopulárnější PBTA nástroje [33].

Prvním z nich je *Text Adventure Development System* [34] neboli *TADS*. Jde o programovací jazyk a sadu knihoven vydaný roku 1988 společností *High Energy Software*. Od té doby prošel *TADS* několika verzemi, z nichž poslední je v době psaní této práce *TADS 3* z roku 2006 silně inspirovaný jazyky Java a C++. Hry napsané v *TADS* se kompilují do formátu spustitelného ve virtuálním stroji.

Druhým systémem je *Inform* [35] vyvinutý roku 1993 matematikem, básníkem a informatikem Grahamem Nelsonem. Stejně jako *TADS* se *Inform* v průběhu let vyvíjel, až do poslední verze *Inform 7* vydané také v roce 2006, která je v podstatě novým jazykem. Oproti programátorskému *TADS* si *Inform* zakládá na přirozeném jazyce. Zaměřuje se tak spíše na spisovatelskou než na vývojářskou stranu média. S tímto zaměřením se také pojí jednodušší a uživatelsky příjemnější rozhraní vývojového prostředí pro práci s tímto jazykem, které se podobá rozhraní dříve popsaného enginu *Quest* a které můžeme vidět na obrázku 4.2. Hry psané v *Inform* se také spouští ve virtuálním stroji.

Oba tyto nástroje se těší dlouhé tradici a poměrně vysoké popularitě. Tvorba PBTA je však běžně náročnější než CBTA, na které se soustředí náš nástroj. Tato volba vychází ze zaměření na co největší jednoduchost pro uživatele, které nás ostatně přivedlo i k IF. Implementace parseru by mohla být jedním z budoucích rozšíření tohoto nástroje, v této chvíli se však není třeba obávat závažného konkurování mezi ním a PBTA enginy.



Obrázek 4.2 Uživatelské rozhraní online verze *Informu 7*

4.2.2 Ren'Py

Ren'Py [36] je engine umožňující tvorbu visual novel vytvořený v roce 2004 Tomem Rothamelem v jazyce Python. Podobně jako u *ChoiceScriptu* se části příběhu zobrazují sekvenčně, nerozhodne-li autor jinak. Jde tedy o nástroj, který nevěnuje zvýšenou pozornost větvení, což se odráží na nedostatku funkcí ulehčujících orientaci v rozvětveném příběhu. Mimo tento nedostatek je práce v *Ren'Py* jednoduchá a uživatelsky příjemná. Kromě větvení se tento nástroj stará i o zobrazování postav a lokací, hudbu a zvukové efekty a umožňuje používání proměnných s hodnotami pravda nebo nepravda. Pokročilejší uživatelé mají dále možnost vkládat kód napsaný v Pythonu, který se v dané části příběhu spustí. Projekty je možné exportovat do komprimovaných spustitelných souborů.

Konkurenční vztah našeho nástroje a tohoto enginu se podobá vztahu s PBTA enginy. V současné době se oblasti jejich působení liší, je však možné, že s budoucími rozšířeními dojde k jejich překryvu. V takovém případě je náš engine schopný předčit *Ren'Py* v jednoduchosti, přístupnosti na webu, orientaci na větvení a z dlouhodobého pohledu i šíří možností.

4.2.3 Další

Kromě *TADS* a *Informu* můžeme najít mnoho dalších PBTA enginů. *Adrift* [37] se podobně jako *Inform* zaměřuje na neprogramátory, jeho rozhraní je však složitější. *Adrift* používá graf, nezobrazuje ovšem s jeho pomocí rozvětvený příběh, nýbrž mapu světa.

Toto využití grafu v kontrastu se způsobem, jak jej používá například *Twine*, ukazuje na důležitý rozdíl mezi PBTA a CBTA hrami. Při tvorbě CBTA autor popisuje přímo příběh. Definuje vždy situaci, ve které se hráč nachází, a jaké jsou jeho možnosti. Oproti tomu autor PBTA popisuje spíše herní svět a interakce s objekty v něm. Obvykle tedy definuje popis místnosti, v níž se hráč nachází, předměty a postavy, které jsou v ní umístěny, a možnosti, jak s nimi interagovat.

PBTA enginy jsou postaveny na těchto definicích lokací, postav a objektů. Jednodušší přístup k tvorbě a zaměření na příběh, spíše než na objevování světa nebo inventářové hádanky, ke kterým PBTA vybízí, jsou tudíž aspekty, v nichž se náš engine bude od těchto nástrojů lišit, a nebudeme si je zde dále důkladně popisovat. Další příklady těchto nástrojů jsou *Dialog* [38] nebo *The Quill* [39].

Nakonec pouze zmíníme nástroje, které neslouží přímo ke tvorbě IF, ale jako podpora pro tvorbu nelineárních příběhů ve složitějších hrách. Příkladem mohou být *ChatMapper* [40] nebo komplexnější *Articy:draft* [41], který využívají i autoři hry *Disco Elysium*. Tyto nástroje jsou obvykle placené a neumožňují vytvořený příběh publikovat, pouze v některých případech vyzkoušet rozpracované dialogy. Jakožto komerční produkty dosahují většinou vyšší úrovně kvality než běžné IF enginy a nabízí mnoho dalších funkcí jako například různé pokročilé databáze postav, lokací a předmětů. Jejich účel se liší natolik, že nepředstavují vážnou konkurenci.

4.3 Univerzální nástroje

Poslední skupina nástrojů slouží obecnějšímu účelu, než je tvorba IF, z čehož vyplývají dvě skutečnosti. Zaprvé mají o mnoho početnější uživatelskou základnu. Je samozřejmé, že například Python ovládá více programátorů než *ChoiceScript*. Zadruhé to znamená, že neumožňují tvorbu IF her s takovou jednoduchostí jako specializované enginy, jelikož pro ni nejsou optimalizovány.

Tyto výhody a nevýhody jsou dány podstatou těchto nástrojů a udávají naši pozici z hlediska konkurování. Cílem IF enginu by nemělo být předčit univerzální nástroj v počtu uživatelů, ale předčit ho v počtu uživatelů, kteří ho používají právě pro tvorbu IF. Toho lze docílit pouze soustředěním na jednoduchost tvorby IF, jedinou přednost, které velké všeobecné enginy a programovací jazyky nemohou dosáhnout, a zároveň se snažit, aby se uživatel cítil co nejméně

omezován hranicemi tohoto žánru a neměl tak motivaci vyhledat složitější ale méně limitovaný nástroj.

4.3.1 Programovací jazyky

Doménově nespecifické programovací jazyky jsou v digitální sféře tím nejuniverzálnějším nástrojem. Programátor, který si tedy pro tvorbu interaktivního příběhu zvolí například jazyk Java, nenarazí narozdíl od mnohých specializovaných enginů na problém s omezeními tohoto nástroje. Na druhou stranu nemůže očekávat žádné zjednodušení a bude se muset spoléhat zejména na špatně čitelnou strukturu podmíněných příkazů.

Mnohé programovací jazyky nabízí knihovny, které tvorbu her zjednodušují, jako je například knihovna *Pygame* v jazyce Python. Tyto balíčky užitečných funkcí jsou mezistupněm mezi samotným programovacím jazykem a herním enginem a mají mnoho využití. Ovlivní ovšem pouze výsledný program, ne jeho tvorbu. Autor se tedy stále nevyhne například problémům spojeným s organizací rozvětveného příběhu.

Ač nejde přímo o programovací jazyky, HTML a CSS umožňují tvorbu IF skrze odkazy mezi stránkami. Díky jejich rozšířenosti jde o jednoduchý způsob, jak příběh reprezentovat. Pokud však jde o samotnou tvorbu příběhu, naráží tyto jazyky na podobné problémy jako jazyky programovací.

Další zajímavou možností je programovací jazyk Prolog, který využívá paradigmatu logického programování a funguje tak dosti podobně jako PBTA hry. Program v Prologu je sestaven z logických pravidel, na které se uživatel dotazuje. Je tak například možné vytvořit mapu jeskyně, po níž se může hráč pohybovat, s místnostmi, které obsahují předměty, s nimiž může interagovat.

4.3.2 Herní enginy

Všeobecné herní enginy jsou v podobné pozici jako programovací jazyky, byť ne do takového extrému. Jde o nástroje umožňující o mnoho více, než autor IF potřebuje, a tedy o mnoho komplikovanější než by pro něj bylo nutné. Na druhou stranu umožňují obvykle snadnou práci s GUI[3], hudbou a zvukovými efekty a podobně. Navíc si udržují univerzalitu programovacích jazyků a žánrově autora neomezují, zpravidla totiž některý z programovacích jazyků používají.

Mezi největší všeobecné herní enginy patří *Unity* [42], *Unreal* [43], *GameMaker* [44] nebo *Godot* [45]. Kromě nich existuje i mnoho specializovaných herních enginů, které nemají k IF daleko. Jde například o *RPG Maker* [46] pro roleplayingové hry, kde příběh hraje velkou roli, *Adventure Game Studio* [47] pro point-and-click adventury, které jsme si popsali v kapitole o interaktivní fikci, nebo *Bitsy* [48] - engine pro tvorbu drobných interaktivních zážitků, kde hráč interaguje s postavami a objekty.

Náročnost práce spojenou s velkým rozkročením těchto nástrojů jde do jisté míry obejít použitím různých pluginů věnovaných IF. Jde o nástroje, které si uživatel většinou snadno nainstaluje a které rozšíří funkci daného enginu. Hlavním problémem tohoto přístupu je fakt, že pro začátečníka je výběr a instalace enginu, pochopení jeho základních funkcí a následný výběr, instalace a pochopení pluginu stále příliš komplikovaný proces.

4.3.3 Prezentační editory

Ač nejde o jejich zamýšlenou funkci, nástroje pro tvorbu prezentací, jako je například *PowerPoint*, umožňují tvorbu interaktivního příběhu, pokud poskytují funkce nelineární prezentace. Je-li v tomto nástroji možné přidat do slidu tlačítko s odkazem na jiný slide, mohou být tyto odkazy použity k navigaci v rozvětveném narativu.

[3] Grafické uživatelské rozhraní

Pochopitelně pak většinou není možné očekávat velké rozšiřující funkce jako podpora proměnných. Na druhou stranu nabízí tyto nástroje lepší možnosti grafické úpravy včetně animovaných přechodů mezi pasážemi. Problémem může být také fakt, že prezentace se obvykle v daném nástroji vytváří i zobrazuje. Pokud by si tedy chtěl člověk hru vytvořenou v prezentačním editoru zahrát, musel by mít přístup k danému programu nebo alespoň k programu s ním kompatibilním.

Kapitola 5
Analýza technologií

Cílem této kapitoly je kriticky zhodnotit programovací jazyky, frameworky, knihovny, technická řešení a nástroje, které jsou pro naši implementaci dostupné. Na základě tohoto hodnocení následně provedeme výběr technologií, které využijeme, a tyto vybrané technologie důkladněji popíšeme.

Jak bylo popsáno v kapitole 3, našimi hlavními prioritami při vývoji enginu jsou snadná tvorba, snadná publikace a snadná rozšiřitelnost. Tato kritéria se odrazí na volbě technologií, které k implementaci použijeme, při této volbě však posuzujeme několik dalších záležitostí. Jde o dostupnost a cenu technologie, rychlost používání, omezení, která technologie přináší, kvalitu dokumentace a množství tutoriálů a v neposlední řadě osobní preferenci a znalost.

5.1 Platforma

Prvním bodem je určit, jakým způsobem bude uživatel k aplikaci přistupovat. Toto rozhodnutí bude mít velký dopad na snadnost používání a dostupnost pro uživatele, jedná se tedy o důležitou volbu. Nabízí se tři možnosti – desktopová aplikace, mobilní aplikace nebo webová aplikace.

První variantou je desktopová aplikace. Jde o program, který si uživatel nainstaluje na svůj počítač. Tento přístup má tu výhodu, že po instalaci nemusí být uživatel připojen k internetu, pokud chce aplikaci používat. Na druhou stranu od něj ze začátku vyžaduje jistou angažovanost, jelikož nemůže začít nástroj používat ihned, ale musí projít instalačním procesem. Dalším problémem je nutnost udržování aplikace na několika operačních systémech nebo výběru jednoho operačního systému na úkor ostatních.

Mobilní aplikace má oproti desktopové výhodu snazší instalace přes *Google Play* nebo *App Store*. Zároveň by pro uživatele byla přístupnější, vzhledem k tomu, že mobil může používat kdykoli. Problémy této varianty jsou nedostatečně velká obrazovka – engine bude vyžadovat velkou plochu pro zobrazení grafu, přetrvávající nutnost počítat s různými operačními systémy a také mentální nastavení uživatele – mobilní aplikace obvykle bývají jednodušší a často volnočasové, narozdíl od serióznějšího pracovnějšího prostředí osobního počítače.

Poslední možností je webová aplikace. Tato varianta nám umožní velmi rychlý přístup pro uživatele a možnost používání napříč operačními systémy. Pokud bychom se zároveň časem rozhodli rozšířit rozhraní aplikace, aby bylo použitelné i na menších obrazovkách, je možné se k ní dostat i pomocí mobilu. Hlavním nedostatkem webové aplikace je potřeba připojení k internetu. Vzhledem k tomu, jak je dnes internet rozšířený, je tato možnost nejlepší volbou a použijeme ji pro náš nástroj. V budoucnu však můžeme implementovat i desktopové verze, které umožní práci offline uživateli.

5.2 Programovací jazyk a framework

Předtím, než se rozhodneme pro programovací jazyk, musíme si položit otázku, zda-li chceme náš nástroj implementovat jako SPA – Single-page application, nebo MPA – Multiple-page application [49].

V případě MPA musí klient poslat žádost serveru při každé změně obsahu stránky. Server odpoví novou HTML stránkou, kterou poté klient zobrazí. Oproti tomu SPA zašle klientovi jednu vysoce interaktivní stránku, která mění svůj obsah pomocí skriptu bez komunikace se serverem, jenž už slouží pouze k zasílání dat z databáze, k níž aplikace přistupuje. SPA tedy nemusí při každé změně načítat celou novou stránku, ale pouze její části.

SPA jsou díky tomu, že nemusí čekat na server, rychlejší a responzivnější. Pro vývojáře přináší jasnou separaci dat od způsobu jejich zobrazení, snadnější debugging pomocí vývojářských nástrojů v prohlížeči a více přímý vývoj – je například možné zpracovat data zadaná uživatelem do formuláře přímo v daném komponentu uživatelského rozhraní. Dalším přínosem SPA přístupu, který je pro náš účel velmi relevantní, je fakt, že po původním načtení stránky není nutné, aby byl klient připojen k internetu, nepotřebuje-li od serveru další data. Tato skutečnost společně s rychlostí výsledné aplikace, snazším vývojem a osobní preferencí silně převažuje nad negativy SPA jako je pomalejší první načtení aplikace, nutnost povolit JS v prohlížeči nebo nepředvídatelné chování tlačítek pro navigaci v historii prohlížeče. Pro vývoj našeho nástroje tedy použijeme SPA přístup.

V případě našeho prototypu se budou veškerá data ukládat do paměti prohlížeče nebo na počítač uživatele, není tedy nutné, aby aplikace zahrnovala databázi a backend. Zbývá nám rozhodnout v jakém jazyku, případně v jakém frameworku, naprogramujeme klientskou část aplikace. Framework je kostrou webové aplikace a sadou nástrojů, které zjednodušují její vývoj. SPA Frameworky zajišťují například navigaci v aplikaci, propojení se serverem, automatizované testování nebo provázání dat mezi DOMem[1] a skriptem aplikace. Vzhledem k enormnímu ulehčení, které používání SPA frameworku nabízí oproti čistému JS, tuto technologii využijeme k implementaci našeho nástroje.

Třemi nejpopulárnějšími SPA frameworky jsou *React* [50], *Angular* [51] a *Vue.js* [52]. Byť je *React* označován jako framework, jedná se spíše o knihovnu, jelikož sice nabízí mnoho funkcí, ale zaměřuje se na vykreslování UI a nevynucuje určitou strukturu projektu. *Angular* je oproti tomu silně strukturovaný a to zejména do modulů, komponentů a servis. Komponenty jsou základním kamenem aplikace a skládají se z HTML šablony, TypeScript kódu, definice stylu a souboru s automatickými testy. *Vue* projekty jsou též organizovány do komponentů, které jsou však definovány v jednom souboru obsahujícím HTML, CSS i JS.

React i *Vue.js* nabízí větší rychlost vývoje, jelikož však náš nástroj navrhujeme s vyhlídkami na jeho budoucí rozšiřování, musíme počítat s tím, že může narůst v poměrně složitou aplikaci. Z tohoto důvodu bude výhodné zvolit si pro jeho implementaci *Angular*, jehož dobrá strukturovanost pomůže udržet kód srozumitelný. V tomto rozhodnutí zároveň hraje velkou roli osobní preference a zkušenost.

Angular pracuje s programovacím jazykem TypeScript, dále jen TS, který je nadmnožinou JS – tedy kód psaný v JS je platným kódem v TS. Tento jazyk, vyvíjený společností *Microsoft*, rozšiřuje možnosti JS zejména o statické typování, tedy možnost určit datový typ proměnné, parametru nebo návratové hodnoty a nechat kompilátor vynucovat jeho dodržování. TS je kompilován do JS, aby bylo možné jeho kód spouštět ve webovém prohlížeči.

5.3 Způsob publikace

Co se týče způsobu publikace, nabízí se dvě možnosti. Buď to bude možné hru z nástroje exportovat do spustitelného souboru nebo bude export probíhat do formátu, který spustíme pomocí

[1] Document Object Model – datová reprezentace webového dokumentu

jiného nástroje. Výhoda první možnosti je, že hráč nebude muset instalovat žádný software výjma hry samotné. Výhoda druhé možnosti tkví v jednodušším vytvoření souboru, který bude obsahovat hru. Pokud si navíc zvolíme takový nástroj, který je uživatelům snadno dostupný nebo lépe, který již je ve většině případů v jejich vlastnictví, ztratí výhoda exportu do spustitelného souboru význam.

Takový nástroj existuje, jde o webový prohlížeč. Tento software umožňuje zobrazování HTML stránek, mezi nimiž se uživatel může pohybovat prostřednictvím odkazů, jejichž vzhled může být upraven pomocí CSS a které mohou být doplněny JS kódem. Reprezentace hry pomocí HTML navíc umožňuje snadnou publikaci například na webových stránkách uživatele.

Pokud zvolíme publikaci hry pomocí HTML, musíme ještě ustanovit, jaká bude sestava exportovaných souborů. Nabízí se možnost vytvořit pro každou pasáž příběhu samostatný soubor HTML a pro každou volbu do něj přidat jeden odkaz na další soubor. Alternativní přístup by byl exportovat celou hru do jednoho souboru HTML a tento upravovat pomocí JS kódu. Hra by v tomto případě byla vlastně Single-page aplikací. Tento postup využívá například *Twine* se svými příběhovými formáty a jeho hlavní výhody jsou, že uživateli stíží zasahování do kódu a že je srozumitelnější pro uživatele, kteří se nevyznají ve fungování webových stránek. Pro naši aplikaci zvolíme první variantu a to ze dvou důvodů. Prvním je jednoduchost implementace takového exportu v porovnání s potřebou vytvořit vlastní skript pro management SPA. Druhým důvodem je tradice otevřenosti, která se pojí s papírovými gamebooky. Čtenář tohoto typu IF mohl dle svého uvážení listovat knihou, vracet se v případě nespokojenosti nebo prozkoumávat různé větve příběhu a užívat si tak plně interaktivity tohoto média. Tuto otevřenost zachováme a pasáže děl vytvořených v našem nástroji budou transparentně rozdělovány do vlastních souborů.

Využití webových stránek pro reprezentaci CBTA hry má ještě jednu výhodu. Možnost vkládat do textu pasáží JS kód pomocí <script> tagu. Uživatel tak získá nekonečné možnosti turingovsky kompletního programovacího jazyka, aniž bychom museli cokoli navíc implementovat. Jak je popsáno v kapitole zabývající se funkčními požadavky – náš nástroj bude nabízet rozšiřující funkcionalitu jednoduše pomocí speciálních pasáží a bez nutnosti programování, možnost skriptovat však zpřístupní tuto funkcionalitu expertním uživatelům již v ranných fázích vývoje a bez jakýcholi omezení.

5.4 Knihovna pro práci s grafy

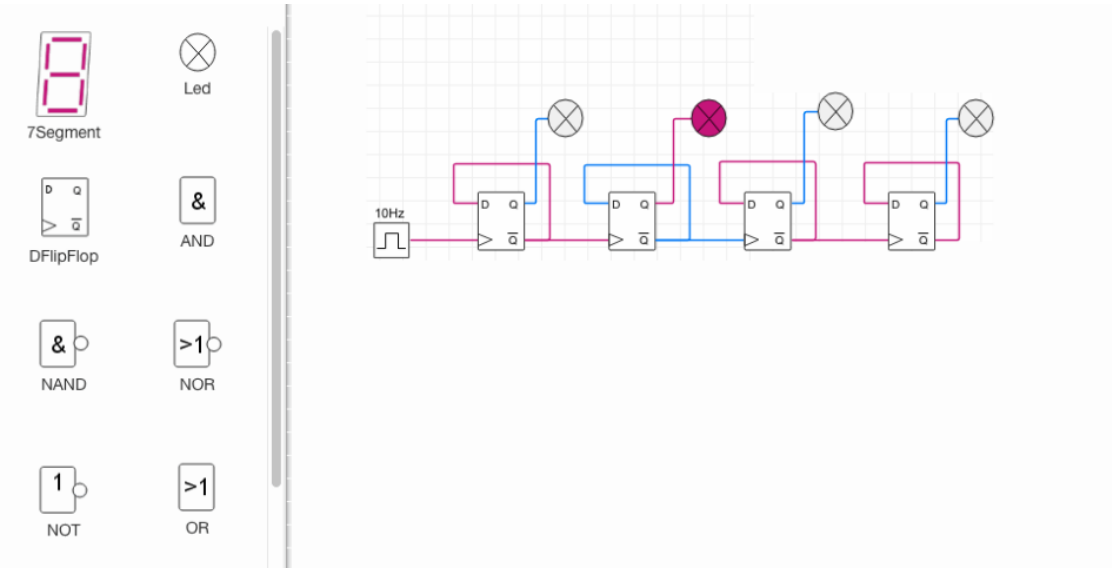
Důležitou součástí naší aplikace bude rozhraní pro tvorbu grafu. Implementace takovéto funkcionality není jednoduchou záležitostí, bude proto vhodnější, abychom využili již dostupnou knihovnu nebo komponentu.

Po takové knihovně budeme vyžadovat, aby graf mohl tvořit uživatel dynamicky, abychom mohli upravovat vzhled vrcholů a aby bylo možné propojovat jeho vrcholy a hrany s námi definovanými daty – zejména půjde o text pasáží a případná další data spojená se speciálními typy pasáží. Pokud se nám podaří najít více knihoven, které splňují tyto požadavky, vybereme jednu z nich podle toho, kolik funkcí nabízí navíc[2] a jak jednoduše se s nimi pracuje. Navíc budeme preferovat komponenty přímo vytvořené pro *Angular* před JS knihovnami.

První knihovnou, kterou jsme uvažovali byla *RxZu*, která nabízí poměrně jednoduchý, ale vizuálně velmi dobře zpracovaný systém pro dynamické tvoření grafů. Problém zde byl zejména naprosto nedostatečná dokumentace a nemožnost upravovat datovou strukturu vrcholů[3]. Dalšími negativy byl nedostatek rozšiřujících funkcí a systém přidávání hran pomocí portů.

Při našem hledání jsme se často setkávali s knihovnami určenými spíše pro datovou vizualizaci než pro interaktivní tvorbu grafu uživatelem. Jde například o knihovnu *Sigma.js*, která sice nabízí jistou interaktivitu, ne však na takové úrovni jakou bychom potřebovali[4]. Dalšími knihovnami

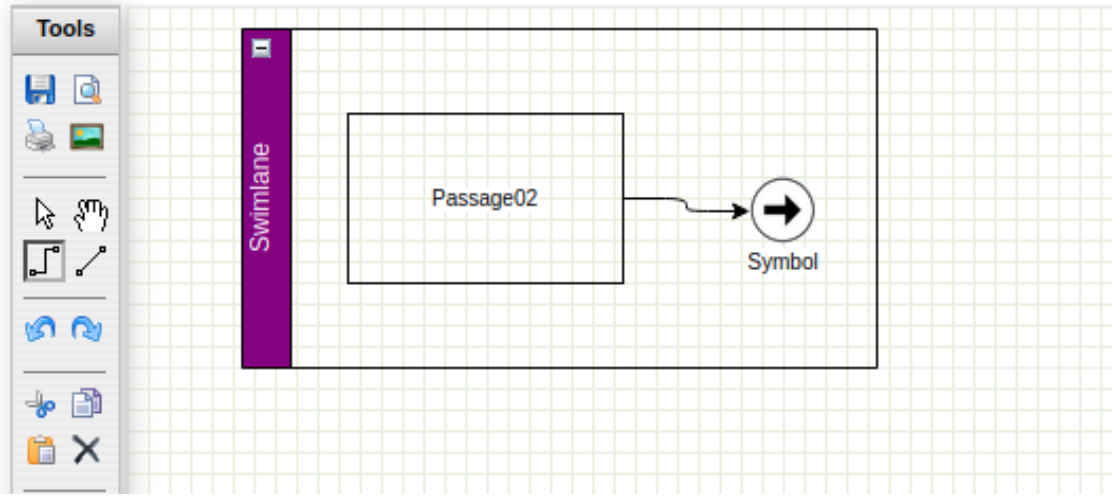
[2] př.: historie akcí, vodicí linky, přehledové zobrazení
[3] nebo alespoň domnělá nemožnost, na základě nedostatečné dokumentace
[4] Není zde například možné měnit titulek vrcholu po dvojkliku.



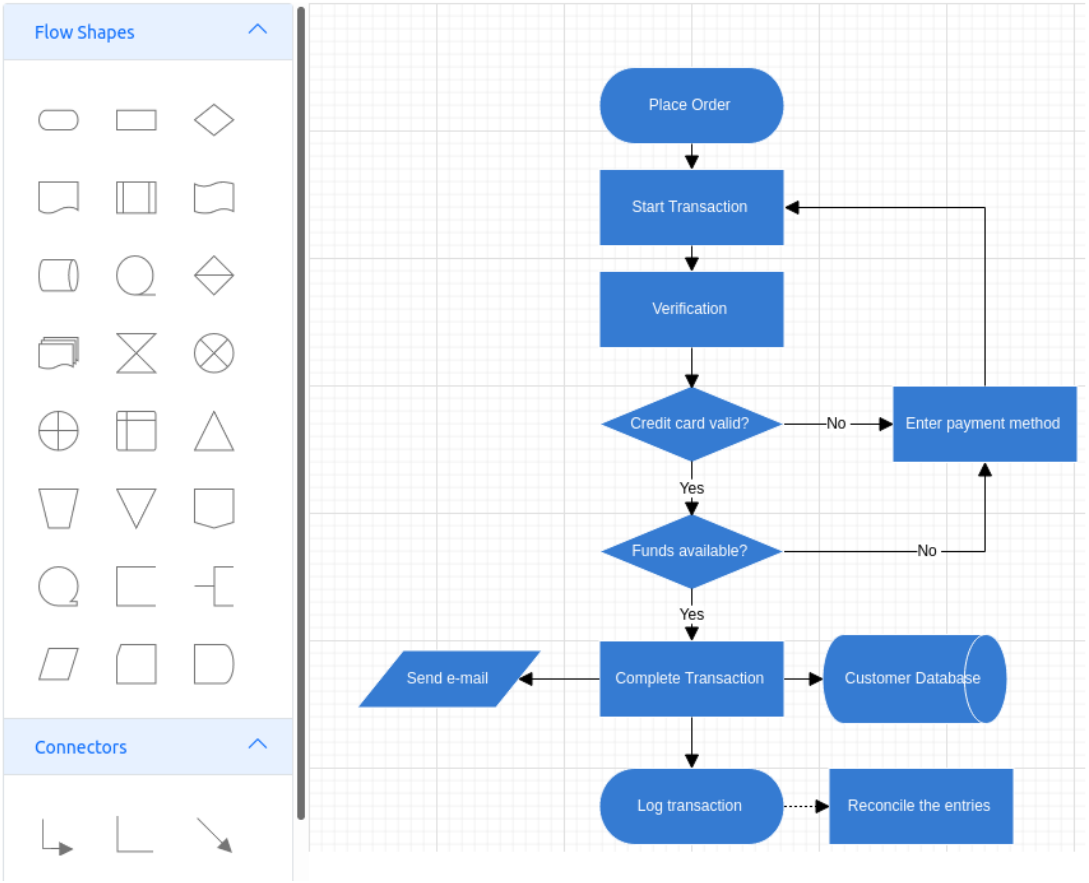
Obrázek 5.1 Demo projekt knihovny *Draw2D*

zaměřující se spíše na vizualizaci byly *Ngx-echarts* nebo *Cytoscape*. Z důvodu nízké interaktivity jsme též zavrhli knihovnu *Swimlane/ngx-graph*.

Do finálního výběru se dostaly knihovny *Draw2D* [53], *MxGraph* [54] a diagram z UI knihovny od společnosti *Syncfusion* [55]. Všechny tyto knihovny splňují naše základní podmínky, liší se však v našich širších kritériích. *Draw2D* a *MxGraph* jsou JS knihovny, *Syncfusion* však nabízí verzi daného komponentu přímo pro *Angular*. Dokumentace všech tří nejsou ideální, nejvíce je však tento nedostatek znát v případě *Draw2D* a poté v případě *MxGraphu*. Velkou výhodou knihovny *MxGraph* je osobní zkušenost. Pro *Vorpal* však využijeme knihovnu *Syncfusion*, jelikož krom zmíněných předností nabízí spoustu jednoduše implementovatelných rozšiřujících funkcí a její výchozí přístup k vrcholům a hranám se shoduje s našimi potřebami - v první řadě jde o prvky, které nesou textovou informaci.



Obrázek 5.2 Demo projekt knihovny *MxGraph*



Obrázek 5.3 Demo projekt knihovny *Syncfusion*

Příklady demo projektů knihoven Draw2D, MxGraph a Syncfusion vidíme na obrázcích 5.1, 5.2 a 5.3.

5.5 Podpůrné nástroje

Poslední technologie, které musíme zvolit, jsou podpůrné nástroje. Jde zejména o verzovací systém a o management úloh. Vzhledem k tomu, že je naše aplikace vyvíjena jednočlenným týmem, není v této chvíli nutné volit komunikační platformu.

Verzovací systém je software, který umožňuje ukládání kódu, spravování jeho verzí, jeho sdílení a spolupráce více uživatelů. Tato spolupráce může být řešena uzamčením souboru, dokud ho uživatel nepřestane upravovat – tato strategie se používá zejména pro projekty, kde není jednoduché spojovat změny, jako například grafická díla. Druhým způsobem organizace společné práce více přispěvatelů je vytvoření kopie projektu a následné sloučení těchto kopií. Tento přístup se obvykle využívá u kódu, protože je možné snadno sloučit změny, které najednou provedlo více uživatelů.

Velmi rozšířeným příkladem takového systému je *Git*, který umožňuje větvení verzí projektu a jejich slučování a který synchronizuje kód mezi lokálním repozitářem uživatele a vzdáleným repozitářem na serveru. *Git* používají například platformy *GitHub* a *GitLab*, které nabízí servery pro vzdálené repozitáře, grafické rozhraní pro práci s *Gitem* a různé další funkce spojené s managementem projektu.

GitHub se zaměřuje na rychlost nasazení projektu. Oproti tomu *GitLab* cílí spíše na spolehlivost. Pro náš projekt zvolíme *GitLab*, hlavním motivačním faktorem pro toto rozhodnutí je však osobní zkušenost.

Kromě verzovacího systému potřebujeme rozhodnout, jak budeme organizovat úlohy spojené s vývojem. Tyto úlohy potřebujeme vytvářet, rozdělovat do skupin, přidělovat řešitelům a sledovat jejich životní cyklus. Nástroje, které toto umožňují jsou například uživatelsky velmi příjemné ale poněkud jednoduché *Trello*, *Asana* nebo sofistikovaná *Jira*. Pro nás dostatečnou funkcionalitu však v tomto ohledu nabízí i *GitLab*, nemá tedy velký smysl vybírat další nástroj.

Kapitola 6

Návrh architektury

V této kapitole navrhneme z jakých součástí se bude naše aplikace skládat a jak budou tyto součásti interagovat. Hlavními kritérii hodnocení tohoto návrhu jsou snadná rozšiřitelnost a údržba kódu, kterých docílíme zejména aplikací objektově orientovaných návrhových vzorů a inteligentním rozdělením zodpovědnosti mezi jednotlivé třídy.

Jak již bylo řečeno, aplikace psaná v *Angularu* se skládá zejména z komponentů a servis. Komponenty jsou základní stavební bloky této aplikace, mají HTML šablonu a TS kód zajišťující interaktivitu a výpočty. Servisy jsou třídy, které komponentům poskytují nějakou přidanou funkci jako například komunikace se serverem. Komponent získá instanci servisy při svojí konstrukci pomocí konceptu zvaného dependency injection – tedy poskytnutí závislostí třetí stranou, aniž by je komponent musel konstruovat sám. Komponenty, servisy a další prvky aplikace jsou seskupovány do modulů.

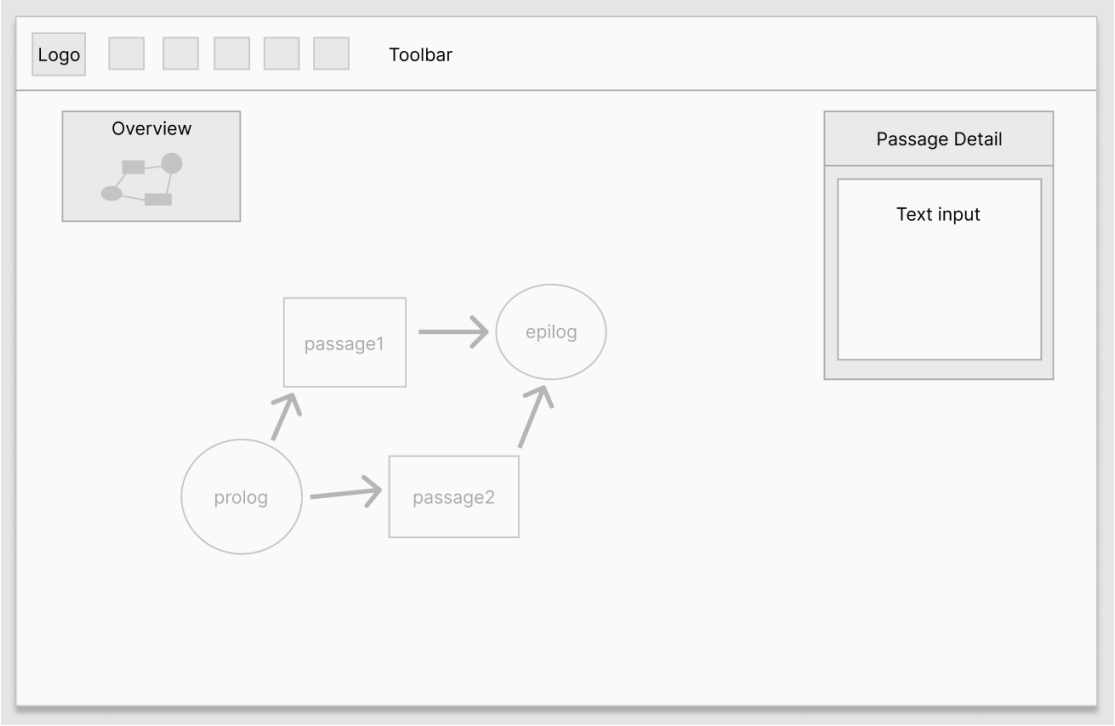
Jelikož je naše aplikace poměrně celistvá, bude nám stačit jeden všeobecný modul. Návrh architektury si rozdělíme na tři hlavní části. Zaprvé půjde o uživatelské rozhraní a s ním úzce spojenou komponentovou strukturu projektu. Zadruhé si popíšeme jakým způsobem budeme implementovat nástroje, které uživateli poslouží ke tvorbě příběhového grafu, ukládání projektu a podobně. Zatřetí se budeme zabývat exportem – popíšeme si, jaké třídy v něm budou figurovat a jak spolu budou interagovat v rámci procesu exportu.

Při návrhu každé z těchto tří oblastí budeme mít na paměti hlavní priority, které jsme si stanovili v kapitole 3. Z hlediska návrhu uživatelského rozhraní půjde zejména o snadnost tvorby v našem nástroji, jakmile se přesuneme k návrhu architektury nástrojů a exportu, největší důležitosti nabyde snadná rozšiřitelnost aplikace.

6.1 Uživatelské rozhraní

Základem uživatelského rozhraní bude plocha pro kreslení příběhového grafu. Tato plocha bude zabírat většinu obrazovky, abychom maximalizovali počet pasáží a voleb, které jsme uživateli schopni zobrazit najednou. Součástí kreslicího rozhraní budou dva panely popsané v kapitole 3 – přehledový panel a panel detailu. Nad pracovní plochou bude umístěno nástrojové menu, jehož součástí budou logo s názvem enginu a dále tlačítka jednotlivých nástrojů. Vizuální návrh naší aplikace – takzvaný *wireframe*, můžeme vidět na obrázku 6.1.

Všechny tyto prvky budou umístěny do komponentu *DiagramModeler*, jehož zodpovědností bude sdílení dat mezi nimi. Nástrojové menu potřebuje přístup k pracovní ploše, aby ji mohly jeho nástroje ovlivňovat nebo pracovat s daty o grafu. Přehledový panel tento přístup též vyžaduje,



Obrázek 6.1 Návrh rozložení uživatelského rozhraní

aby bylo možné zobrazovat přehled plochy a panel detailu musí znát aktuálně vybranou pasáž nebo volbu.

Nástroje v menu budou reprezentovány pomocí komponentu *MenuItem*, který obaluje příkazy a dodává jim grafické zastoupení v aplikaci. Příkazy vzniknou v komponentu nástrojového menu a definují chování nástrojů, jejich popisek a ikonku. Více podrobně se budeme architekturou systému nástrojů zabývat v následující sekci.

Výpis kódu 6.1 HTML kód DetailSwitcher komponentu

```
<app-basic-node-detail
    [element]="element"
    *ngIf="show[0]">
</app-basic-node-detail>

<app-basic-edge-detail
    [element]="element"
    *ngIf="show[1]">
</app-basic-edge-detail>
```

Posledním problémem týkajícím se návrhu uživatelského rozhraní, který musíme vyřešit, je fungování panelu detailu. Vzhled tohoto komponentu by se měl totiž lišit v závislosti na typu prvku, jehož detail zobrazuje. Pokud chceme zobrazit detail obyčejné volby, stačí nám jen její text. Oproti tomu, detail pasáže jejíž součástí je definice proměnných, bude muset obsahovat formulář s typem, názvem a iniciální hodnotou této proměnné. Abychom umožnili tuto nutnou heterogenitu panelu detailu, obalíme ho dalším komponentem, který nazveme *DetailSwitcher*. Jeho zodpovědností bude držet instance detailů všech dostupných typů a zobrazovat jen ten z nich, který odpovídá typu prvku, jenž mu zašle *DiagramModeler*. Při budoucím rozšiřování aplikace o nové typy prvků v grafu bude tedy nutné v *DetailSwitcheru* definovat vzhled nového

detailu nebo přiřadit novému prvku jeden z již definovaných detailů. V kódu 6.1 a 6.2 vidíme, jak tento komponent ovládá viditelnost jednotlivých detailů pomocí pole *show*, předává detailům data o elementu a rozhoduje o tom, který detail zobrazit v metodě *changeElement*.

Výpis kódu 6.2 TS kód DetailSwitcher komponentu

```
export class ElementDetailSwitcherComponent implements OnInit {

    public show: boolean[];
    public element: any;

    ngOnInit(): void {
        this.resetDetail();
    }

    public resetDetail () {
        this.show = [ false, false ];
    }

    public changeElement( element: any ) {
        this.element = element;
        this.resetDetail();
        if ( this.element.id.substring(0,4) === "node" ){
            switch ( this.element.addInfo.passageType ){
                case PassageType.Checkpoint_Save:
                case PassageType.Checkpoint_Load:
                case PassageType.Basic:
                case PassageType.Prolog:
                case PassageType.Epilogue:   this.show[0] = true; break;
            }
        }
        else if ( this.element.id.substring(0,9) === "connector" ) {
            this.show[1] = true;
        }
    }
}
```

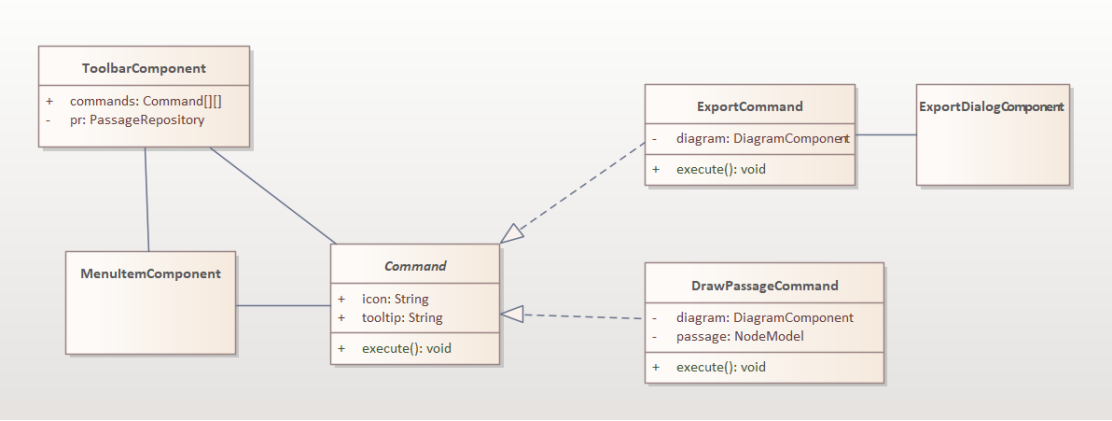
6.2 Nástroje

Dalším úkolem, který před námi stojí, je navrhnout, jak budeme přistupovat k nástrojům. Samozřejmě by bylo možné definovat všechny přímo v komponentu *Toolbar*, tento postup však vede na velkou nepřehlednou třídu *ToolbarComponent*, což koliduje s naším cílem, aby byl kód co nejpřehlednější a snadno se tedy udržoval a rozšiřoval. Abychom se vyvarovali vzniku této monolitní třídy, využijeme návrhový vzor command [56].

Command enkapsuluje nějaký příkaz jako samostatný objekt. Jeho použitím dosáhneme vyčlenění zodpovědnosti za spouštění příkazu na *MenuItemComponent*, který drží instanci daného commandu, a zodpovědnosti za provedení příkazu na daný command. Tento přístup nám též v budoucnosti umožní snadné logování a pohyb v historii úprav.

Na obrázku 6.2 vidíme, jak spolu interagují jednotlivé třídy zajišťující fungování nástrojů v naší aplikaci. *MenuItemy* jsou generovány *ToolbarComponentem* na základě pole *Commandů* vytvořeného při jeho iniciaci, jak vidíme v kódu 6.3[1]. Všechny *Commandy* dědí z rodičovské třídy

[1] *Commandy* v *ToolbarComponentu* ukládáme do dvojrozměrného pole, abychom je mohli vizuálně rozdělit do oddělených skupin.



Obrázek 6.2 Diagram tříd zobrazující systém nástrojů

Command, kterou můžeme vidět v ukázce kódu 6.4. Každý *Command* implementuje metodu *execute* a definuje ikonku a tooltip, kterými je v menu reprezentován.

Specifickým typem nástroje jsou nástroje pro přidávání pasáží. V jejich případě není nutné používat dědičnosti pro odlišení jednotlivých typů pasáží, postačí nám kompozice. Vytvoříme tedy jeden *DrawPassageCommand*, jenž si bude držet instanci pasáže, kterou umožňuje přidávat. Protože chceme udržet na minimu počet změn nutných pro implementaci nového typu pasáže, není vhodné, aby byl za udržování seznamu těchto typů zodpovědný *ToolbarComponent*. Přidáme proto třídu *PassageRepository*, jejímž účelem bude tato data uchovávat. *ToolbarComponent* tedy vždy jen projde dostupné pasáže v tomto repozitáři a pro každou vytvoří samostatný nástroj, který slouží k jejímu vykreslování. Detailněji si *PassageRepository* rozebereme v následující sekci.

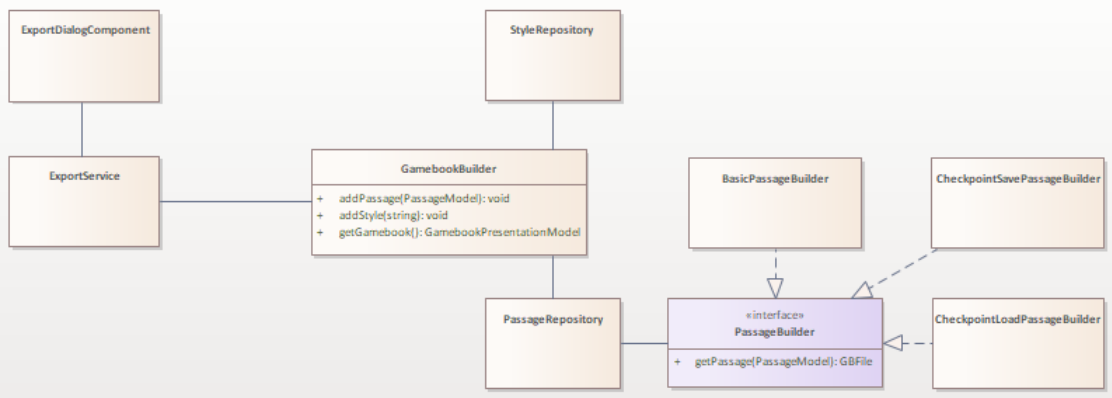
Výpis kódu 6.3 *MenuItem* komponenty generované v *ToolbarComponentu*

```
<section id="tools">
    <span class="tool-set" *ngFor="let set of commands">
        <app-menu-item *ngFor="let c of set" [command]="c">
        </app-menu-item>
    </span>
</section>
```

Výpis kódu 6.4 Rodičovská třída příkazů

```
export abstract class Command
{
    public icon: string;
    public tooltip: string;

    constructor ( icon: string, tooltip: string ) {
        this.icon = icon;
        this.tooltip = tooltip;
    }
    abstract execute (): void;
}
```



Obrázek 6.3 Diagram tříd zobrazující systém exportu

6.3 Export

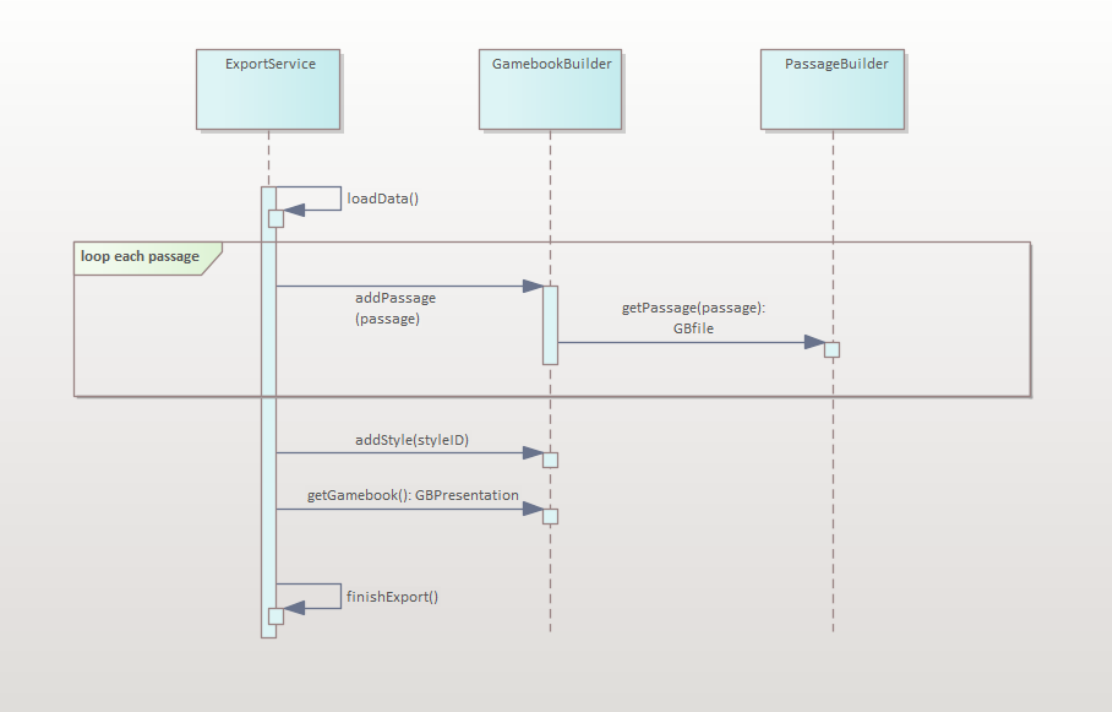
Poslední součástí našeho návrhu architektury je systém exportu. Jeho účelem je převod datové struktury příběhového grafu na hratelnou sadu souborů. Pro tento účel si nejprve definujeme dvě datové struktury. *GamebookModel* drží zjednodušenou reprezentaci grafu, kterou získáme na začátku procesu exportu z grafové komponenty převedením z její datové reprezentace. Jde o seznam pasáží, každá se svým id, titulkem, textem, typem a seznamem voleb, které z ní vychází a které mají id, text, výchozí a koncovou pasáž. Druhou datovou strukturou je *GamebookPresentationModel*, který obsahuje text a názvy generovaných souborů.

Na obrázcích 6.3 a 6.4 vidíme diagram tříd a sekvenční diagram, které popisují participující třídy a průběh procesu exportu. Export je zahájen z nástrojového menu spuštěním *ExportCommandu*, který zavolá *ExportService*. Tato třída se stará o celý průběh exportu. Nejprve vytvoří z diagramu jednodušší a lépe srozumitelný *GamebookModel*, následně postupně předává data z tohoto modelu další třídě – *GamebookBuilderu*, který má na starosti generování textu jednotlivých pasáží. Nakonec *ExportService* získá *GamebookPresentationModel* a vytvoří z něj příslušné soubory.

Builder [56] je návrhovým vzorem, který separuje vytvoření objektu od jeho reprezentace. Drží si postupně vytvářený objekt a pomocí metod volaných zvenku k němu přidává další části. V našem případě se jedná zejména o metodu *addPassage*. Abychom však umožnili snadné přidávání nových typů pasáží, delegujeme vytvoření pasáže na další třídu – *PassageBuilder*. Třídy tohoto typu budou implementovat rozhraní *IPassageBuilder*, které bude vyžadovat metodu *getPassage(data: PassageModel)*. Jak je ukázáno v kódu 6.5, builder pro běžnou pasáž postupně vygeneruje HTML text ve třech metodách – *setStart*, *setContent* a *setEnd*. *SetContent* dále rozčleníme na metody *setTitle*, *setText* a *setDecisions*. Takovéto rozvržení nám umožní dědit z této třídy a doplnit například mezi obsah a konec HTML souboru nějaký skript.

Aby měl *GamebookBuilder* jednotlivé *PassageBuildery* kde získat, použijeme výše zmíněnou třídu *PassageRepository*. Ta tedy bude pro každý typ pasáže, kromě informací o jejím vzhledu v grafu a v menu, držet také instanci odpovídajícího *PassageBuilderu* a bude tak mít zodpovědnost za všechna data, která se týkají nového typu pasáže.

Nakonec chceme přidat možnost, aby si uživatel vybral jeden ze tří předvolených stylů. K tomuto výběru poslouží nový komponent *ExportDialog*, který otevřeme po zavolání *ExportCommandu* a který předá *ExportService* ID vybraného stylu. *ExportService* následně zavolá na *GamebookBuilderu* metodu *addStyle(id: string)*, který si daný CSS soubor získá ze třídy *StyleRepository*, jenž má zodpovědnost za jejich udržování.



Obrázek 6.4 Sekvenční diagram zobrazující proces exportu

Výpis kódu 6.5 Zjednodušený kód třídy *BasicPassageBuilder*

```
export class BasicPassageBuilder implements IPassageBuilder {

  protected id: string;
  protected text: string;

  public getPassage(data: PassageModel): GBFile {
    this.setStart();
    this.setContent(data);
    this.setEnd();
    return {id: this.id, text: this.text};
  }

  protected setContent(data: PassageModel){
    this.setTitle(data);
    this.setText(data);
    this.setDecisions(data);
  }

  ...

}
```

Kapitola 7
Implementace a testování

Cílem následující části je samotná tvorba prototypu aplikace. Popíšeme si její proces a problémy, se kterými jsme se v jejím průběhu setkali. Následně tento prototyp otestujeme a to jak z technického hlediska – objevení a ostranění chyb, tak z hlediska UX. Hlavním bodem tohoto testování bude tvorba vlastní CBTA hry.

7.1 Průběh implementace

První fáze implementace spočívala v zapojení diagramové komponenty a otestování jejích možností. Šlo v podstatě o drobný prototyp modelovací funkcionality, kde jsme si vyzkoušeli, jak je možné dynamicky v aplikaci přidávat vrcholy a hrany do grafu, jaké nástroje tato komponenta nabízí a jak jsou strukturována data reprezentující graf. Po vytvoření tohoto rychlého POC[1] jsme si založili nový projekt a začali pracovat přímo na budoucí aplikaci.

Základem této aplikace bylo rozložení prvků UI, zejména diagramového modeleru. K němu jsme přidali dva panely – přehled a detail, nástrojové menu a připravili též dialog exportu.

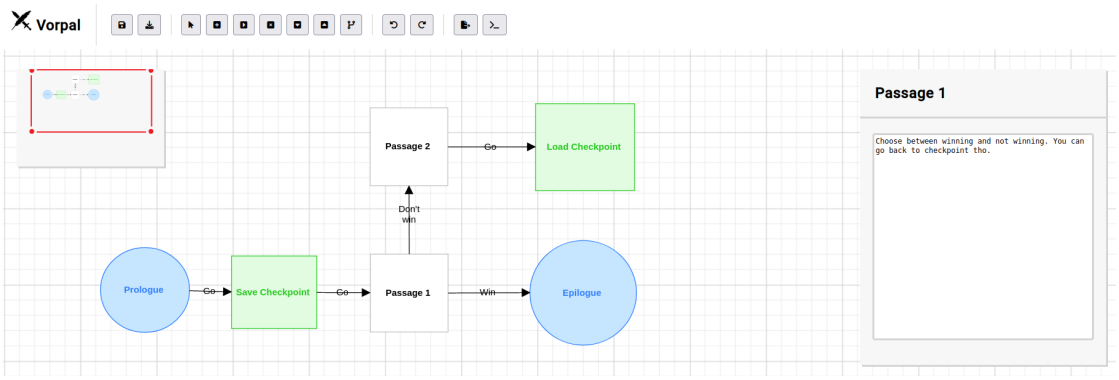
Pokračovali jsme implementací nástrojů, tak jak je popsáno v kapitole Návrh architektury – každý nástroj v menu zastupuje komponent *MenuItem*, který drží instanci třídy *Command*, respektive třídy z ní dědící. Tato třída poskytuje funkci, která se provede po stisknutí tlačítka *MenuItemComponentu*. Zároveň s vytvářením této struktury bylo nutné upravovat diagramový modeler. Implementované nástroje umožňují vytváření voleb a pasáží různých typů, pohyb v historii úprav diagramu a výpis dat diagramu do konzole.

Po zajištění této základní kreslicí funkcionality došlo na propojení pasáží a voleb v grafu s jejich detailem za použití *PassageSwitcheru* a přidání textu pasáže a možnosti ho upravovat. Dále jsme implementovali ukládání projektu do paměti prohlížeče a načítání z ní a přidali korespondující nástroje do nástrojového menu.

Následujícím úkolem bylo umožnit export projektu do hratelné reprezentace. Postupně jsme se tedy propracovali od *MenuItemu* v nástrojovém menu přes *ExportCommand*, *ExportService* a *GamebookBuilder* až po jednotlivé *PassageBuildery*, jak je popsáno v předchozí kapitole. Také jsme přidali repozitář pasáží a přesunuli sem definici pasáží, která se doposud nacházela v nástrojovém menu.

Posledním velkým úkolem bylo přidat pasáže pro definici kontrolních bodů. Díky tomu, že jsme v návrhu počítali s tímto rozšiřováním, šlo pouze o dodání dvou nových *PassageBuilderů* a drobnou úpravu v *DetailSwitcheru*. Vyzkoušeli jsme si tedy, že *Vorpal* je tímto způsobem velmi rychle a snadno rozšiřitelný. Nakonec už jsme jen upravovali vzhled aplikace, odstraňovali chyby a provedli refaktoring kódu. Finální vzhled *Vorpalu* můžeme vidět na obrázku 7.1

[1] Proof of concept – drobný experimentální projekt dokazující, že zvolené řešení problému bude fungovat



Obrázek 7.1 Uživatelské rozhraní *Vorpalu*

7.2 Řešené problémy

V rámci implementace jsme narazili na jeden významný problém – práci s lokálním úložištěm prohlížeče v lokálních HTML souborech. Lokální úložiště je funkce HTML *Web Storage API*, která umožňuje ukládat data v prohlížeči klienta. Tato data nejsou zasílána serveru, mají formu dvojic text:hodnota, jsou perzistentní napříč celou doménou a lze k nim snadno přistupovat z webových stránek pomocí JS.

Takto fungující úložiště je přesně to, co potřebujeme pro implementaci velkého množství rozšiřujících pasáží. Příkladem jsou pasáže pro práci s proměnnými, které musí mít kde ukládat hodnoty těchto proměnných, aby k nim mohla jiná pasáž přistupovat. Dalším příkladem je ukládání kontrolních bodů, které jsme implementovali v rámci této práce a kde se projevil podstatný nedostatek lokálního úložiště.

Tímto nedostatkem je fakt, že prohlížeč pracuje s HTML soubory umístěnými na počítači uživatele, jako by každý náležel pod vlastní doménu. To znamená, že perzistence lokálního úložiště mezi soubory neplatí a při lokálním hraní nebo testování není možné kontrolní body používat.

V rámci této práce jsme otestovali, že po nasazení hry na web vše funguje, jak má. Řešení tohoto problému tedy necháme až na případné budoucí rozšíření aplikace. Nabízí se několik možností, jak se k němu postavit. První z nich je vytvořit testovací rozhraní přímo v aplikaci – umožnit autorovi spustit příběh z libovolné pasáže přímo z modeleru a hrát hru, jako by došlo k jejímu exportu. Alternativně by bylo možné přidat možnost exportovat projekt do SPA, jak jsme si popsali v kapitole Analýza technologií.

Kromě této překážky jsme se při implementaci potýkali zejména s komponentou pro vykreslování a upravování grafu, jejíž dokumentace se ukázala v některých případech jako nedostatečná a její chování jsme museli v některých případech odvozovat experimentálně nebo z jejího kódu.

7.3 Tvorba testovací hry

Abychom co nejvíc do hloubky otestovali možnosti *Vorpalu*, bude testovací hra, kterou vytvoříme, beletrií. Kdybychom se rozhodli tvořit výukový materiál, například příběh popisující školní výlet plný nástrah, který by žáky poučil o bezpečnosti a první pomoci, nemusel by být výsledný produkt tak spletitý. Oproti tomu interaktivní povídka není vázána nutností hráče něco naučit, volby nejsou tak jasně rozděleny na správné a špatné a svým účelem také vybízí k rozsáhlejšímu a rozvětvenějšímu narativu, což je pro nás důležité, neboť jen tak otestujeme limity našeho enginu.

První věcí, které si se zahájením tvorby všímáme, jsou drobné nedostatky v kvalitě UX. Jde například o fakt, že kreslicí nástroje přepnou kurzor do módu přidávání pasáží nebo voleb. Uživatel však obvykle pracuje tak, že přidá jednu pasáž a následně přejde k upravování jejího textu, musí tedy po každém přidání přepnout z kreslicího nástroje na obvyklý kurzor. Další nedostatky jsou mírná nepřehlednost nástrojového panelu – stálo by za zvážení, zda doplnit ikonky nástrojů stálým textem, nemožnost přidávat volbám klouby a měnit tak jejich tvar, nemožnost pohybovat panelem s přehledem pracovní plochy nebo čas od času nastávající situace, kdy volba není na jedné nebo obou stranách připojena k pasáží, což není dostatečně signalizováno. Velmi by také pomohlo, kdyby uživatel viděl alespoň část textu pasáží přímo v grafu a nemusel tak vždy přepínat mezi pasážemi v panelu. Tato možnost by navíc umožnila rychlý přehled o tom, které pasáže jsou již dopsány a kam je ještě třeba text doplnit. Drobným problémem je též fakt, že detail pasáže nereaguje na její odstranění.

Jak postupně tvořený narativ rozšiřujeme, přicházíme na další problémy spojené s limity enginu. Jde zejména o nedostatečnou velikost pracovní plochy. Ta by ideálně měla umožňovat nekonečné scrollování do všech stran. Na další problém narážíme v rozložení detailu pasáže, když zadáme příliš dlouhý titulek a pole pro textový vstup se posune mimo panel. Když vyexportujeme rozpracovaný projekt, vidíme, že podobně se chová grafické rozložení vytvořené hry, je-li moc dlouhý titulek pasáže nebo text volby.

Dále otestujeme funkce ukládání a načítání. Možnost uložit projekt v paměti prohlížeče je vítaná při kontinuální práci, kterou si chceme v průběhu zálohovat. Pokud však uživatel tvorbu na čas přeruší, uložení v prohlížeči mu nedodá dostatečný pocit, že je jeho projekt v bezpečí. Ukládání do souboru na počítači uživatele je funkce, kterou bude třeba přidat v blízké budoucnosti, a která navíc ulehčí společnou práci na projektu pro více autorů. Problémová je též interakce mezi přehledem plochy a načítáním projektu.

Co se týče exportu, je třeba opravit zejména chybu, ke které dochází při pokusu o export projektu s nepojmenovanými volbami. Nedostatkem je též fakt, že engine exportuje velké množství souborů místo toho, aby vytvořil jeden archiv a umístil soubory do něj.

Potvrdili jsme si též potřebu budoucích rozšíření. V jedné části příběhu si hlavní postava prohledává kapsy. Dávalo by smysl, aby měla tuto možnost v jakékoli chvíli, to by však znamenalo, že autor musí přidat tuto volbu, do všech pasáží příběhu a také že musí nějak vyřešit, jak se z této větve vrátit zpět.

Poslední skutečností při tvorbě testovací hry, bylo skriptování. Vyzkoušeli jsme si, že je jednoduše možné přidat do běžné pasáže zvuk pomocí dvou řádků kódu a několik dalších interakcí jako je například výpis textu do konzole prohlížeče.

Některé z těchto problémů, jako je export voleb bez textu nebo špatně rozložený detail a exportované soubory, opravíme rovnou. Mnohé z nich jsou však nad rámec této práce a proto si je pouze poznamenáme jako vhodné kandidáty k budoucímu rozšiřování aplikace. Přestože jsme identifikovali poměrně velké množství potřebných úprav, práce ve *Vorpalu* byla poměrně jednoduchá a rychlá. Již ve fázi prototypu je znát, že námi zvolený přístup má co nabídnout a po provedení několika úprav bude tento nástroj možno srovnávat s jinými alternativami na trhu.

7.4 Nasazení a uživatelské testování

Po vlastním otestování přišel čas aplikaci i testovací gamebook zpřístupnit na webu a provést testování širším vzorkem uživatelů. K nasazení jsme použili platformu *Netlify* [57], která nabízí hostování stránek ze souboru nebo formou CI[2]. Mimo to poskytuje zdarma doménová jména, která budou pro naše účely postačující.

[2]Continuous Integration - metoda integrace repozitáře kódu a webové stránky umožňující téměř instantní nasazení změn.

Kromě *Vorpalu*[3] samotného, jsme zpřístupnili také testovací gamebook ve dvou různých stylech – jeden z nich jednoduchý s jednobarevným pozadím[4], druhý s obrázkem na pozadí[5], abychom mohli porovnat jejich kvality.

Testování enginu proběhlo podobně jako v případě tvorby testovací hry. Účastnily se ho postupně čtyři osoby s různou úrovní znalosti informatiky – inženýr s vystudovanou informatikou, který se tímto oborem živí, dva studenti, jeden z bakalářského, druhý z magisterského studia, a jeden student střední školy, bez obsáhlejších informatických znalostí. Cílem tohoto testování bylo zejména ověřit zjištění, ke kterým jsme došli při tvorbě testovací hry, a zjistit, jaké objevené problémy jsou jak závažné.

Před začátkem testování jsme provedli krátkou ukázku funkcionality *Vorpalu*, následně jsme testující vyzvali, aby vytvořili jednoduchý příběh o pár pasážích. V průběhu této tvorby se ukázalo, že problémová je zejména skutečnost, že volby nemusí být vždy propojeny s pasážemi. Testující dále potvrzovali, že nástrojový panel není tak přehledný, jak by mohl být, a ti více informaticky zdatní též zmiňovali nepříjemnou potřebu ručně měnit nástroje z kreslicího na obyčejný kurzor. Nejasnosti dále vzbudilo fungování pasáží pro kontrolní body, přesněji to, zda je možné použít pasáž pro načtení kontrolního bodu před uložením. Celkový pocit z práce ve *Vorpalu* byl však pozitivní a žádný z problémů, na něž jsme během něj narazili, by testující neodradil v používání nástroje.

Ve druhé fázi uživatelského testování jsme se zabývali námi vytvořeným gamebookem. Testujících bylo tentokrát osm, s různými úrovněmi znalosti informatiky a s různým vztahem ke tvůrčímu psaní. V první řadě jsme chtěli zjistit odpovědi na následující otázky:

- Je hra funkční[6]?
- Jsou oba zveřejněné styly dobře čitelné a esteticky přijatelné?
- Jakým způsobem hráč hru hraje? Zkoumá různé cesty, snaží se dostat do přijatelného konce, nebo přestane při dosažení prvního, byť negativního konce?
- Zdá se testujícím koncept interaktivní fikce zajímavý a měli by zájem o rozsáhlejší dílo?

Z hlediska funkčnosti nenarazili hráči na žádné nedostatky. Preference stylů se poměrně vyrovnaně rozdělily, žádný z testujících však neshledal ani jeden ze stylů špatně čitelným nebo výrazně hůře vypadajícím, styl s obrázkem na pozadí se však nezobrazoval správně na mobilním zařízení. Zajímavá byla odpověď na třetí otázku. Očekávali jsme, že většina uživatelů se dostane do pasáže s titulkem "Konec", která obsahuje více méně pozitivní konec příběhu, a hru ukončí. Pasáž jsme tímto způsobem nazvali schválně z důvodu, abychom sledovali reakci hráče. Více než polovina hráčů však i poté využila možnosti v příběhu se vrátit a dále ho prozkoumávat. Interaktivní fikce se testujícím vesměs jevila jako velmi zajímavý žánr a většina z nich by projevila zájem o hypotetické dílo knižního rozsahu a kvality. Jedním z podnětů, které zazněly, bylo, že pro takové dílo by byla více vhodná mobilní platforma, aby uživatel mohl četbu snadno přerušit a znovu se k ní vrátit.

[3] na doméně https://chipper-flan-0adc5b.netlify.app
[4] https://storied-torte-34f3c0.netlify.app
[5] https://lustrous-parfait-23262b.netlify.app
[6] Fungují správně odkazy mezi stránkami, grafické rozložení stránek, kontrolní body a zvukové efekty přidané pomocí JS?

Kapitola 8
Budoucí rozšíření

V této kapitole si rozebereme, kam by se mohl vývoj ubírat v budoucnosti. Jaká rozšíření by bylo vhodné doplnit a jak se přiblížit hernímu enginu, který jsme si popsali v úvodu.

Jak již bylo řečeno, nástroj vyvinutý v rámci této práce je pouze POC a má ještě daleko k tomu, aby mohl konkurovat enginům jako je *Twine*. Díky tomu, jak byl navržen, však umožňuje snadné rozšiřování, zejména co se týče přidávání nových nástrojů a nových typů pasáží.

Budoucí vývoj by se měl v první řadě zaměřit[1] na rozšíření základní funkcionality, jako je například vymáhání omezení různých typů pasáží. Až v momentě, kdy budou základní kreslicí možnosti dosahovat dostatečné kvality, by se měl vývoj přesunout k přidávání dalších typů pasáží, přidávání zjednodušující funkcionality a rozšiřování nastavení celého projektu. V této fázi by měl nástroj pomalu svou kvalitou předčit většinu konkurence – nabízet všechny možné funkce, které zůstávají ve sféře CBTA, snadno dostupné neexpertním uživatelům pouhým přidáváním a vyplňováním předdefinovaných pasáží. Zároveň by se měla navýšit jeho uživatelská příjemnost, ve které většina konkurenčních nástrojů pokulhává. Poslední fází by měla být platformizace nástroje a práce s komunitou, tak aby byly pokryty konkurenční výhody, které nabízí například používání *ChoiceScriptu*. Zároveň s touto platformizací by mohly být přidávána další rozšíření, která by umožnila vytvářet hry mimo žánr CBTA.

8.1 Základní úpravy

Tato rozšíření zahrnují zejména funkce spojené s tvorbou grafu, které většina nástrojů pracujících s grafy poskytuje. Jednou z nejdůležitějších funkcí z této skupiny je kontrola omezení pasáží a voleb. Aktuální prototyp kontroluje, že do prologu nesmí vstupovat žádné volby a z epilogu nesmí žádné vycházet, nesleduje ovšem už, zda nevložil uživatel do projektu více než jeden prolog, nebo zda volba spojuje dvě pasáže. Tyto kontroly by měly probíhat minimálně před exportem, nejlépe ovšem kontinuálně při každé změně.

Dále bude nutné přidat do nástrojového menu nejrůznější funkce, které jsou ve finálním stavu této práce dostupné pouze přes klávesové zkratky, jako je přiblížení a oddálení, kopírování, vystřihování a vkládání nebo označení všech pasáží a voleb. Bylo by též vhodné přidat na modelovací plochu vodicí linky pro zarovnání objektů vůči sobě a mít možnost upravit náhled na plochu tak, aby obsahoval všechny objekty.

Nakonec bude třeba změnit způsob, kterým se přidávají pasáže. Ve chvíli psaní této práce, kdy aplikace nabízí pět typů pasáží, je možné k nim přistupovat pomocí tlačítek na nástrojovém menu. S postupujícím vývojem však budou přidávány další a další typy pasáží, časem tedy nástrojové

[1] kromě úprav navržených a popsaných v průběhu testování

menu nebude dostačující. Tento problém je možné vyřešit sérií roletek[2]. Z uživatelského hlediska je však příjemnější implementace paletkového systému přístupného skrze panel na pracovní ploše. Tento panel by bylo možné přesouvat a měnit jeho velikost. Obsahoval by paletky pasáží – tedy hierarchicky uspořádané skupiny různých typů pasáží, jenž by se zobrazovali stejným způsobem jako na pracovní ploše a tažením by je bylo možné umístit do grafu. Uživatel by měl možnost vytvářet vlastní paletky často používaných typů pasáží a několik typů umístit i do nástrojového menu nebo jim definovat klávesové zkratky pro rychlý přístup.

8.2 Speciální pasáže

Hlavním nositelem rozšiřovacích funkcí by měly být pasáže různých typů. V kapitole věnované funkčním požadavkům jsme si popsali a později v kapitole 7.1 implementovali pasáže pro ukládání a načítání kontrolních bodů. Tyto pasáže jsou vzorem pro implementaci dalších rozšíření. Každý nový typ pasáže je nutné přidat do *PassageRepository* a ošetřit, jakým způsobem bude interagovat s panelem detailu, kde bude autor moci danou pasáž dále konfigurovat.

Speciální typy pasáží slouží zejména neexpertnímu uživateli. Jelikož je možné do textu běžné pasáže vkládat JS kód, je uživatel znalý skriptování schopen obstarat většinu těchto rozšiřujících funkcí pouze s jejím použitím. Motivací za jejich zavedením však je jednoduchost a přehlednost, kterou práce s nimi poskytuje.

Příkladem funkcionality, kterou mohou obstarat speciální pasáže, je práce s proměnnými. Bude-li autor moci definovat proměnné, provádět nad nimi operace, zobrazovat jejich obsah, povolovat a zakazovat na základě jejich obsahu volby a naplňovat je vstupem od hráče, jeho dílo nabude úplně nové úrovně interaktivity. Tímto způsobem bude možné například získat od hráče jméno postavy a používat ho v příběhu, sledovat hráčův inventář, nebo mu přidělit schopnosti, které může ve hře používat a postupně je vylepšovat[3].

Další speciální pasáže mohou rozšiřovat možnosti kontrolních bodů. Základní pasáže pro práci s kontrolními body uloží jednu pasáž nebo vytvoří volbu, která do ní hráče vrátí. Jiné typy pasáží by mohly uložit kontrolní bod bez mazání předchozího nebo nechat hráče rozhodnout, jestli chce pasáž uložit.

Prvkem, který vyžaduje o něco složitější implementaci, jsou globální pasáže. Tento typ pasáže by umožnil, aby hráč viděl dané volby ze kterékoli jiné pasáže v příběhu a mohl je kdykoli vybrat. V grafu by tedy do ní žádné volby nevedly, pouze by z ní vycházely a jiný typ pasáže by umožnil vrátit se v příběhu zpět na místo, odkud byla tato pasáž vybrána. Mělo by být též možné zpřístupnit globální pasáž na základě proměnné a ovládat tak, ze kterých pasáží je hráč schopný do ní vstoupit.

Speciální pasáže by též měli umožňovat snadnou práci s obrázky, videem a zvukem. Nakonec by mohly vznikat různé exotičtější druhy pasáží, s jejichž pomocí bude například možné zapojit do hry soubojový systém nebo různé minihry. Tyto neobvyklejší typy pasáží by již mohly být vyvíjeny komunitou a doinstalovávány do enginu podle potřeby.

8.3 Zjednodušující rozšíření

Tyto funkce nejsou stěžejní pro uživatelovu schopnost nástroj ovládat, v některých případech mu však ušetří dlouhou monotónní práci nebo ulehčí orientaci v grafu.

Jde například o kontrolu souvislosti, tedy o ujištění, že je možné dostat se z prologu do kterékoli jiné pasáže. Tuto kontrolu je možné poměrně snadno implementovat například pomocí algoritmu prohledávání do hloubky, neboli DFS. Problém nastane s použitím rozšíření jako jsou

[2] Př.: Hlavní tlačítko „Přidat pasáž“ se rozbalí a nabídne několik často používaných typů a dále další rozbalovací možnosti jako „Kontrolní body“ nebo „Proměnné“.

[3] V budoucnu pravděpodobně vyvstane též potřeba implementace speciálních typů voleb, například pro podmíněné volby. Pokud k tomu dojde, měla by jejich architektura fungovat analogicky k architektuře pasáží.

proměnné, kdy kontrolní algoritmus bude muset testovat všechny možné hodnoty proměnných, nebo držet vztahy proměnných vůči sobě a při testování podmínek kontrolovat jejich splnitelnost bez narušení těchto vztahů[4].

Vorpal by měl také nabízet textové vyhledávání napříč pasážemi nebo různé automatické úpravy zobrazení a uspořádání grafu, jako například seřazení pasáží podle vzdálenosti od prologu. Uživateli by též mohlo pomoci s orientací v grafu, kdyby mohl pasáže popisující stejnou část příběhu seskupovat do kapitol nebo podle vlastního klíče opatřovat barevnými štítky. Takto zorganizované pasáže by dále mohlo být možné skrývat nebo zobrazovat v závislosti na jejich kapitole nebo štítcích.

8.4 Projektové nastavení

Poslední skupinou rozšíření týkajících se přímo práce s nástrojem jsou globální nastavení projektu. V rozsahu této práce je jediné takové nastavení – výběr jednoho ze tří předvolených stylů, který se vyexportuje spolu s pasážemi. Uživatel by však měl být schopen upravovat mnoho dalších nastavení jako je například možnost vracet se v příběhu zpět.

Samotná práce se styly nabízí mnoho prostoru ke zlepšení. Předvolené styly by měly zůstat možností a jejich množství by se mělo navýšit, náročnějšímu uživateli by však mělo být poskytnuto jednoduché rozhraní pro nastavení nejrůznějších aspektů vzhledu výsledného díla, aniž by musel používat CSS.

8.5 Platformizace

Poslední skupina rozšíření se týká podpory publikace uživatelských výtvorů a komunitních interakcí. Jedná se zejména o online katalog výtvorů, který umožní jejich sdílení, prodej a spolupráci na nich. K tomuto účelu bude nutné přidat databázi a serverovou část, které zároveň umožní ukládání rozpracovaných projektů na server.

V takovém katalogu by mělo být možné psát recenze a hodnotit díla a zobrazovat jejich statistiky jako například kolik hráčů se dostalo do kterého konce. Dále by zde mohl fungovat systém ocenění neboli achievementů jako například na platformě *Steam*. Katalog by také mohl nabízet podporu pro různé soutěže, jako například vytvořit hru na určité téma.

Dalším artiklem, který by bylo možné v katalogu sdílet a prodávat mohou být rozšiřující pasáže. Stejně jako v případě vytvořených her by měli mít uživatelé možnost vzájemně si tato rozšíření komentovat a hodnotit a mohla by být též předmětem soutěží.

Existence katalogu otevírá cestu k monetizaci aplikace. Jelikož příjemnost používání je jedním z hlavních principů našeho nástroje, není monetizace pomocí zapojení reklamy vhodná. Pokud však budou uživatelé schopni prodeje svých děl, dává smysl, aby se z těchto zisků odečetla drobná provize na další vývoj enginu.

[4] Tento algoritmus by pracoval na podobném principu jako jazyk Prolog, který využívá unifikaci proměnných – jejich navázání na určitou hodnotu nebo jinou proměnnou, a backtracking - vrácení posledních kroků unifikace a vyzkoušení jiné možnosti.

Závěr

Cílem této práce bylo položit základní kámen enginu pro tvorbu interaktivní fikce. Tento prototyp byl formován na základě porozumění historii tohoto žánru a slabých a silných stránek konkurenčích řešení a poskytuje tak nový přístup k problémům, se kterými se potýká mnoho jiných nástrojů. *Vorpal* využívá teorie grafů a vhodně zvolených technologií, které mu umožňují naplnit základní pilíře, jenž jsme si při jeho návrhu stanovili – snadná tvorba, snadná publikace a snadná rozšiřitelnost.

Vorpal klade důraz na uživatelskou příjemnost a zaměřuje se na neexpertního uživatele. Systém speciálních pasáží umožňuje přístup k pokročilé funkcionalitě i autorům neznalým programování. Na druhé straně neomezuje programátory, kteří mají možnost přidávat libovolné funkce ve formě skriptů. Díla vytvořená v tomto enginu získá uživatel ve formě transparentní webové prezentace, kterou může dále upravovat a distribuovat dle svého uvážení. Architektura *Vorpalu* nechává díky využití vhodných návrhových vzorů otevřenou cestu rozšiřování tohoto prototypu a rozumné rozdělení zodpovědnosti mezi jednotlivé komponenty aplikace napomáhá snadnému porozumění kódu.

Identifikovali jsme potřebné úpravy a nastínili jsme cestu dál. Přestože možnosti *Vorpalu* jsou v této chvíli omezené, věříme, že jde o silný základ pro kvalitní nástroj, který pomůže vzbuzovat zájem o informatiku, herní design a tvůrčí psaní, podpořit tvorbu zajímavých výukových materiálů, revitalizovat žánr interaktivní fikce a nastavit vyšší nároky na kvalitu narativu v herním průmyslu.

Příloha A

Testovací gamebook

Odkazy na publikovanou hru byly zmíněny v kapitole 7.4, v této příloze si ovšem ukážeme a okomentujeme obrázky a výpisy z kódu, které popisují proces její tvorby a její fungování.

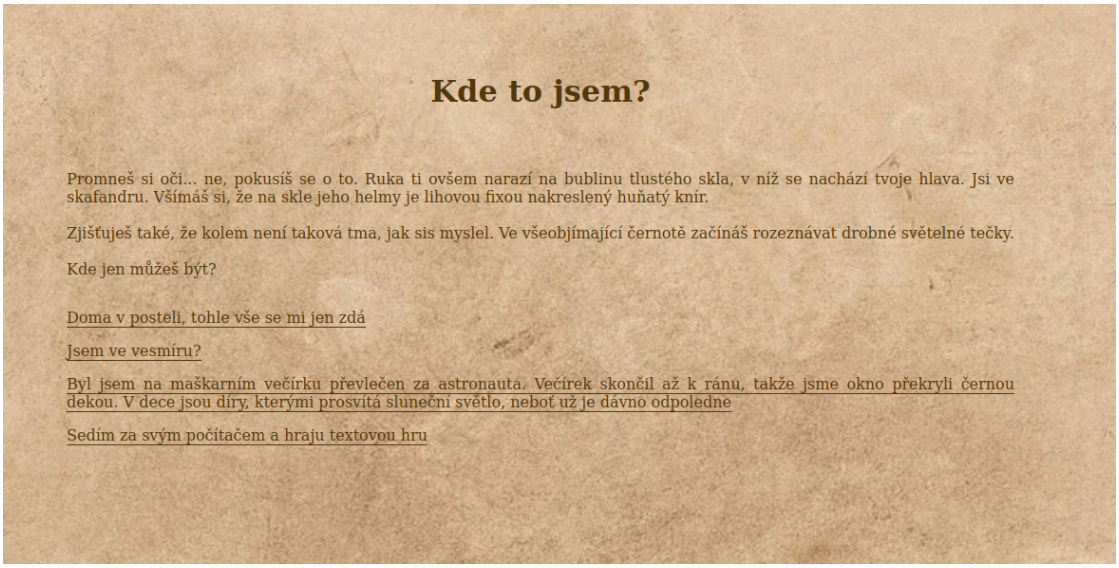
Obrázky A.1, A.2 a A.3 ukazují, jak vypadají tři grafická zpracování, mezi nimiž si může autor pracující ve *Vorpalu* vybírat při exportu svého díla. První z nich je defaultní volbou, jedná se o dobře přehledný styl, který je aplikovatelný na příběh jakéhokoli žánru. Výhoda tohoto stylu je zároveň to, že je možné ho používat i při hraní na mobilním zařízení. Druhý a třetí styl jsou laděné do fantasy a sci-fi žánrů. Pro publikaci testovacího gamebooku byly zvoleny první a třetí styl.

Na obrázku A.4 můžeme vidět soubory, z nichž se testovací gamebook skládá. Nachází se zde soubor *gbstyle.css* s definicí CSS stylu, soubor *index.html*, který označuje prolog a ostatní soubory pasáží, které jsou nazvány svým ID v rámci *Vorpal* projektu. Dále zde můžeme vidět dva *mp3* soubory, které jsme přidali manuálně, abychom je mohli použít při skriptování.

Obrázek A.5 ukazuje HTML kód vygenerovaný *CheckpointLoadPassageBuilderem*. Kromě hlavičky a textu pasáže zde vidíme dva skripty – jeden je součástí textu pasáže a napsali jsme ho manuálně při tvorbě gamebooku. druhý, který do pasáže přidává možnost restartu, byl vygenerován *CheckpointLoadPassageBuilderem*.

Nakonec se na obrázku A.6 můžeme podívat, jak vypadal graf znázorňující příběh testovacího gamebooku. Pořád jde o poměrně krátkou hru, už tak jsou ovšem vidět některé problémy editačního rozhraní *Vorpalu*. Jde zejména o nemožnost přidávat klouby do hran a pohybovat jejich popisky. Výrazně by také pomohlo, kdyby *Vorpal* zobrazoval popisky voleb pouze do určité délky.

Obrázek A.1 Ukázka prvního stylu



Obrázek A.2 Ukázka druhého stylu

Obrázek A.3 Ukázka třetího stylu

Obrázek A.4 Struktura exportovaného projektu

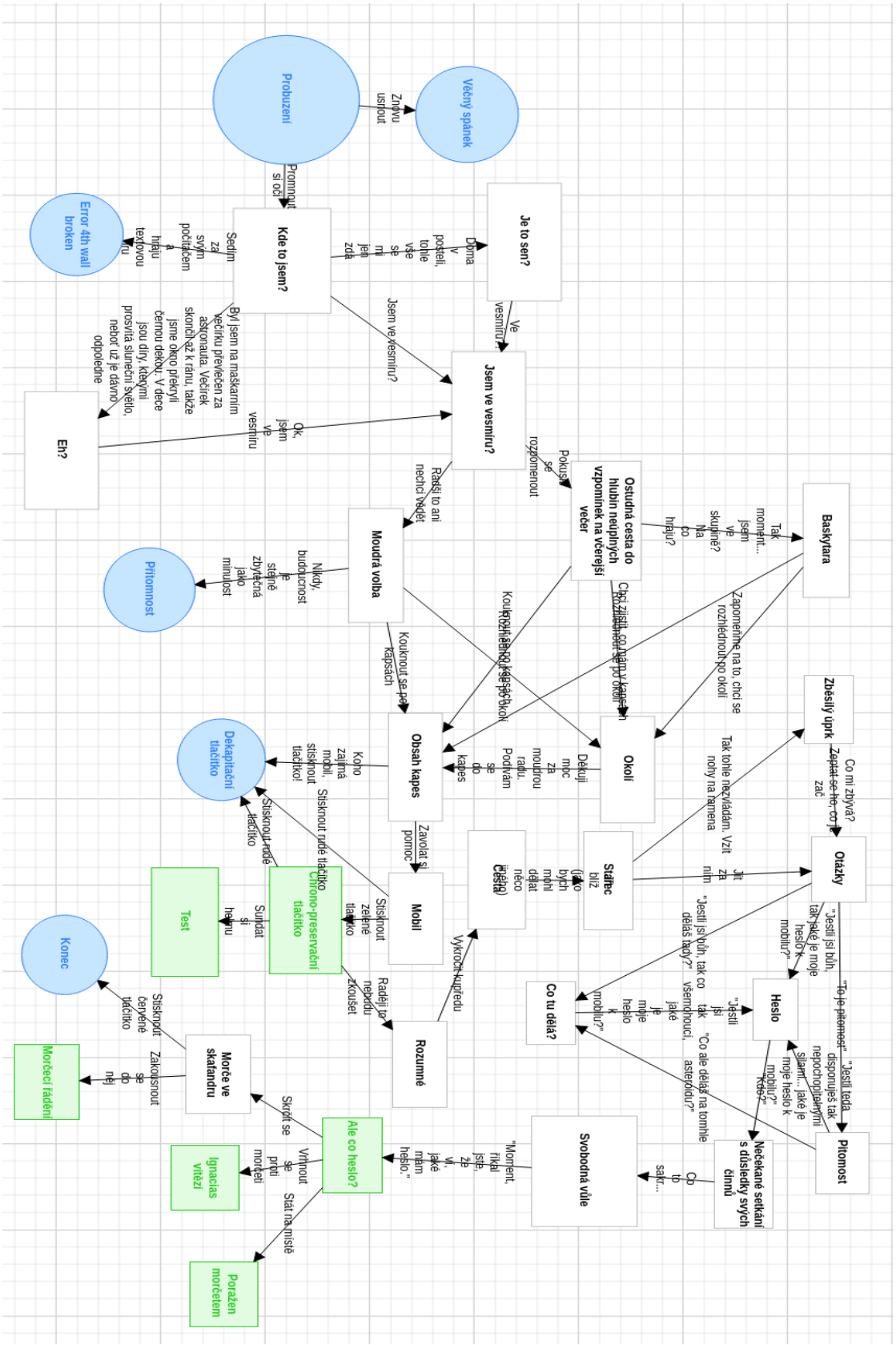
```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="gbstyle.css">
</head>
<body>
<h1>Ignacias vítězí</h1>
<p>
Podaří se ti chytit Ignacia do levé ruky.
Morče drápe a kouše, tak s ním zatřeseš a chystáš se ho zahodit,
když v tom se mu podaří stisknout červené tlačítko na tvojí rukavici.<br\><br\>

Dekapitační protokoly tvojí helmy jsou spuštěny.

<script>
var audio = new Audio('gore.mp3');
audio.play();
</script>
</p>
<script>
  let save = localStorage.getItem("checkpoint");

  let d = document.createElement('div');
  let a = document.createElement('a');
  a.appendChild(document.createTextNode("Load Checkpoint"));
  a.href = save + ".html";
  d.appendChild(a);
  document.body.appendChild(d);
</script>
</body>
</html>
```

Obrázek A.5 Ukázka kódu jedné pasáže gamebooku



Obrázek A.6 Zobrazení projektu ve *Vorpalu*

Bibliografie

1. DJUNDIK, Pavel. *Steamdb - Instant Search Indie* [online]. 2013 [cit. 2022-05-07]. Dostupné z: `https://steamdb.info/instantsearch/?refinementList%5C%5Btags%5C%5D%5C%5B0%5C%5D=Indie`.
2. CAROLL, Lewis. *Through the Looking-Glass, and What Alice Found There*. Macmillan, 1871. ISBN 0194230198.
3. TENNIEL, John. *The Jabberwock* [online]. 1871 [cit. 2022-05-07]. Dostupné z: `https://en.wikipedia.org/wiki/Jabberwocky#/media/File:Jabberwocky.jpg`.
4. WYSARDRY. *Categories of Interactive Fiction?* [Online]. 2020 [cit. 2022-05-07]. Dostupné z: `https://intfiction.org/t/categories-of-interactive-fiction`.
5. BERR. *Multi-User Dungeon* [online]. 2010 [cit. 2022-05-07]. Dostupné z: `https://tvtropes.org/pmwiki/pmwiki.php/Main/MultiUserDungeon`.
6. IMDB. *Disco Elysium Awards* [online]. 2019 [cit. 2022-05-08]. Dostupné z: `https://www.imdb.com/title/tt11177022/awards`.
7. PEDERCINI, Professor Paolo. *Branching narrative from Borges to Twine* [online]. 2015 [cit. 2022-05-07]. Dostupné z: `http://cmuems.com/2015b/branching-narrative-from-borges-to-twine`.
8. KATZ, Demian. *Series - Miscellaneous Works by the Oulipo* [online]. 2020 [cit. 2022-05-07]. Dostupné z: `https://gamebooks.org/Series/292`.
9. PLOWMAN, Martin. *Brief history of gamebooks* [online]. 2016 [cit. 2022-05-07]. Dostupné z: `http://gamesvsplay.com/a-brief-history-of-gamebooks`.
10. LEBLING, Dave. *Classic Game Postmortem: Infocom's Zork* [online]. 2015 [cit. 2022-05-07]. Dostupné z: `https://www.youtube.com/watch?v=FXdmo2j_CiQ`.
11. SCOTT, Jason. *Infocom: The Documentary* [online]. 2017 [cit. 2022-05-07]. Dostupné z: `https://www.youtube.com/watch?v=OXNLWy7rwH4`.
12. BUCK, David. *Adventures In Interactivity* [online]. 2019 [cit. 2022-05-07]. Dostupné z: `https://tedium.co/2019/06/27/adventure-games-interactive-fiction-history`.
13. ACKS, William; COSTREL, France; LACROIX, Sam. *High Score: Role Players* [online]. 2020 [cit. 2022-05-07]. Dostupné z: `https://www.netflix.com/cz-en/title/81019087`.
14. KERKEZ, Zoran. *Interaktivní fikce [online]*. 2016 [cit. 2022-05-07]. Dostupné také z: `https://theses.cz/id/0fpv40/`. Diplomová práce. Univerzita Tomáše Bati ve Zlíně, Fakulta multimediálních komunikacíZlín. SUPERVISOR: MgA. Radek Petříček, Ph.D.
15. MCINTOSH, Jason. *A brief history of text-based games and open source* [online]. 2018 [cit. 2022-05-07]. Dostupné z: `https://opensource.com/article/18/7/interactive-fiction-tools`.

16. TRHOŇ, Ondřej. *Občas stačí jen slova: jak textovky mění herní svět* [online]. 2018 [cit. 2022-05-07]. Dostupné z: https://art.ceskatelevize.cz/inside/obcas-staci-jen-slova-jak-textovky-meni-herni-svet-fWD86.
17. DILLE, Flint; PLATTEN, John Zuur. *The ultimate guide to video game writing and design.* Lone Eagle, 2008. ISBN 9781580650663.
18. KRAMARZEWSKI, Adam; NUCCI, Ennio De. *Practical Game Design.* Packt Publishing, 2018. ISBN 1787121798.
19. HANUŠ, Petr. *Teorie grafu* [online]. 2010 [cit. 2022-05-08]. Dostupné z: https://brkos.math.muni.cz/files/download/Teorie%5C%20graf%5C%C5%5C%AF.pdf.
20. KOVÁŘ, Petr. *Úvod do teorie grafu* [online]. 2021 [cit. 2022-05-08]. Dostupné z: https://homel.vsb.cz/~kov16/files/uvod_do_teorie_grafu.pdf.
21. KNOP, Dušan; MALÍK, Josef; SUCHÝ, Ondřej; TVRDÍK, Pavel; VALLA, Tomáš. *Algoritmy a grafy 1* [online]. 2020 [cit. 2022-05-08]. Dostupné z: https://courses.fit.cvut.cz/BI-AG1/lectures/index.html.
22. ROSSI, Giulia Carla. *Writing Tools for Interactive Fiction* [online]. 2020 [cit. 2022-05-08]. Dostupné z: https://blogs.bl.uk/english-and-drama/2020/04/writing-tools-for-interactive-fiction-.html.
23. KLIMAS, Chris. *Twine* [online]. 2009 [cit. 2022-05-08]. Dostupné z: https://twinery.org/.
24. GAMES, Choice of. *ChoiceScript* [online]. 2011 [cit. 2022-05-08]. Dostupné z: https://www.choiceofgames.com/make-your-own-games/choicescript-intro/.
25. STUDIOS, Inkle. *Inkle* [online]. 2011 [cit. 2022-05-08]. Dostupné z: https://www.inklestudios.com.
26. TEXTADVENTURES.CO.UK. *Quest* [online]. 2011 [cit. 2022-05-08]. Dostupné z: http://textadventures.co.uk/quest.
27. TEXTADVENTURES.CO.UK. *Squiffy* [online]. 2014 [cit. 2022-05-08]. Dostupné z: http://textadventures.co.uk/squiffy.
28. PLAY, No Time To. *Ramus* [online]. 2011 [cit. 2022-05-08]. Dostupné z: https://notimetoplay.org/engines/ramus/.
29. MILLINGTON, Ian. *Undum* [online]. 2018 [cit. 2022-05-08]. Dostupné z: https://idmillington.github.io/undum/.
30. MOIKISTORIES. *Moiki* [online]. 2021 [cit. 2022-05-08]. Dostupné z: https://moiki.fr/en.
31. POROPAT, Matteo. *Libro Game Creator* [online]. 2013 [cit. 2022-05-08]. Dostupné z: http://www.matteoporopat.com/librogame/.
32. GAMES, Crumbly Head. *Gamebook Authoring Tool* [online]. 2013 [cit. 2022-05-08]. Dostupné z: https://www.crumblyheadgames.co.uk/the-gamebook-authoring-tool/.
33. LEINONEN, Juhana. *IFDB statistics, part 2: Development systems* [online]. 2012 [cit. 2022-05-08]. Dostupné z: https://nitku.net/blog/2012/11/ifdb-statistics-part-2-development-systems.
34. SOFTWARE, High Energy. *TADS* [online]. 1988 [cit. 2022-05-08]. Dostupné z: https://www.tads.org/.
35. NELSON, Graham. *Inform* [online]. 1993 [cit. 2022-05-08]. Dostupné z: http://inform7.com/.
36. ROTHAMEL, Tom. *Ren'py* [online]. 2004 [cit. 2022-05-08]. Dostupné z: https://www.renpy.org/.

37. WILD, Campbell. *Adrift* [online]. 2011 [cit. 2022-05-08]. Dostupné z: https://www.adrift.co/.

38. AKESSON, Linus. *Dialog* [online]. 2018 [cit. 2022-05-08]. Dostupné z: https://www.linusakesson.net/dialog/.

39. ZEPHLABS. *The Quill* [online]. 2015 [cit. 2022-05-08]. Dostupné z: https://thequill.app/.

40. HYPERSPACE. *Chatmapper* [online]. 2010 [cit. 2022-05-08]. Dostupné z: https://www.chatmapper.com/.

41. SOFTWARE, Articy. *Articy:Draft* [online]. 2014 [cit. 2022-05-08]. Dostupné z: https://www.articy.com/en/.

42. TECHNOLOGIES, Unity. *Unity* [online]. 2005 [cit. 2022-05-08]. Dostupné z: https://unity.com/.

43. GAMES, Epic. *Unreal Engine* [online]. 1998 [cit. 2022-05-08]. Dostupné z: https://www.unrealengine.com/en-US.

44. OVERMARS, Mark. *Game Maker* [online]. 1999 [cit. 2022-05-08]. Dostupné z: https://gamemaker.io/en/gamemaker.

45. JUAN LINIETSKY, Ariel Manzur. *Godot* [online]. 2014 [cit. 2022-05-08]. Dostupné z: https://godotengine.org/.

46. ENTERBRAIN, KADOKAWA Future Publishing. *RPG Maker* [online]. 1992 [cit. 2022-05-08]. Dostupné z: https://www.rpgmakerweb.com/.

47. JONES, Chris. *Adventure Game Studio* [online]. 1997 [cit. 2022-05-08]. Dostupné z: https://www.adventuregamestudio.co.uk/.

48. DOUX, Adam Le. *Bitsy* [online]. 2001 [cit. 2022-05-08]. Dostupné z: https://bitsy.org/.

49. CHURYLOV, Maksym. *SPA vs MPA: The definitive guide for decision makers* [online]. 2018 [cit. 2022-05-08]. Dostupné z: https://www.mindk.com/blog/single-page-applications-the-definitive-guide.

50. WALKE, Jordan. *React* [online]. 2013 [cit. 2022-05-08]. Dostupné z: https://reactjs.org.

51. GOOGLE. *Angular* [online]. 2016 [cit. 2022-05-08]. Dostupné z: https://angular.io.

52. YOU, Evan. *Vue.js*. 2014. Dostupné také z: https://vuejs.org.

53. HERZ, Andreas. *Draw2D* [online]. 2018 [cit. 2022-05-09]. Dostupné z: https://github.com/freegroup/draw2d.

54. BENSON, David. *MxGraph* [online]. 2005 [cit. 2022-05-08]. Dostupné z: https://github.com/jgraph/mxgraph.

55. SYNCFUSION. *Syncfusion Angular Diagram* [online]. 2020 [cit. 2022-05-08]. Dostupné z: https://ej2.syncfusion.com/angular/documentation/diagram.

56. GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994. ISBN 0201633612.

57. BIILMANN, Mathias; BACH, Christian. *Netlify* [online]. 2014 [cit. 2022-05-08]. Dostupné z: https://www.netlify.com.

Obsah přiloženého média

- `readme.txt` — stručný popis obsahu média
- `Gamebook` — adresář s hratelným testovacím gamebookem
- `Text`
  - `Source` — adresář s LaTeX kódem
  - `thesis.pdf` — PDF bakalářské práce
- `Vorpal`
  - `Build` — sestavená aplikace připravená k nasazení na server
  - `Source` — zdrojový kód enginu *Vorpal*