Západočeská univerzita v Plzni
Fakulta aplikovaných věd Katedra informatiky
a výpočetní techniky

Diplomová práce

Návrh a implementace nového uživatelského rozhraní pro hru Space Traffic

Plzeň, 2016 Bc. Jan Kotalík

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne Bc. Jan Kotalík

Abstract

Design and implementation of new user interface for the game Space Traffic

This thesis deals with design and implementation of user interface for the game Space Traffic. The objective of this thesis is to create user interface of the game, which makes the game playable. This thesis also contains detailed analysis of user interface of the game in state before this thesis started and it's possible improvements. There are also described technologies considered to use. This thesis contains final choice between these technologies along with reasons why it was used. This thesis describes the way of implementation of designed user interface and the way of implementation of support tools, which were created for improving effectiveness of future implementation of Space Traffic user interface.

Abstrakt

Návrh a implementace nového uživatelského rozhraní pro hru Space Traffic

Tato práce se zabývá návrhem a implementací uživatelského rozhraní ke hře Space Traffic. Jejím cílem je vytvořit uživatelské prostředí hry, které by umožnilo ji hrát. Tato práce obsahuje podrobnou analýzu stavu uživatelského rozhraní hry před započetím této práce. Dále jsou zde popsány technologie, jejichž použití bylo zvažováno. Práce obsahuje rovněž konečnou volbu mezi těmito technologiemi spolu s důvody, proč byla některé dána přednost před jinou či z jakého důvodu byla použita. Je zde také popsán způsob implementace navrženého rozhraní a popis implementace podpůrných nástrojů, které byly vytvořeny za účelem zefektivnění budoucí implementace uživatelských rozhraní pro hru Space Traffic.

Obsah

Poděkování

Tímto bych chtěl poděkovat studentům spolupracujícím na projektu Space Traffic, zejména pak druhému správci projektu Bc. Veronice Švecové za nadstandardní úsilí dovést naši práci na projektu k úspěchu.

Dále děkuji členům Katedry informatiky a výpočetní techniky, především svému vedoucímu práce Ing. Petru Vaněčkovi, Ph.D. za to, že nás vzal pod svá křídla a doc. Ing. Přemyslu Bradovi, MSc. PhD. za pomoc se získáváním podpory pro projekt.

1 Úvod

Tato diplomová práce se zabývá problematikou tvorby moderního uživatelského rozhraní webových aplikací. Jejím účelem je provést implementaci grafického uživatelského rozhraní k webové hře Space Traffic, která je studentským projektem na Katedře informatiky a výpočetní techniky Západočeské univerzity v Plzni [1].

Bude zapotřebí seznámit se s technologickými možnostmi tvorby uživatelských rozhraní, zobrazitelného na moderních zařízeních a prohlížečích. Také bude potřeba analyzovat stav projektu a jeho uživatelského rozhraní před provedením této práce, zvážit i jeho technologické možnosti a vytvořit uživatelské rozhraní, které umožní hráči hry ovládat a zobrazit hotové prvky hry. Webová hra Space Traffic obsahuje mnoho vlastností, funkčních pouze na backendu a hráči by měl být umožněn přístup k těmto vlastnostem.

Rovněž je třeba brát ohled na celkovou ovladatelnost a intuitivnost hry. Hráč by měl být schopen obejít se bez návodů, uživatelských příruček či mimoherních pomůcek. Uživatelské rozhraní by mělo být srozumitelné běžnému uživateli, který je zvyklý pohybovat se v prostředí webu a má zájem o webové hry.

Součástí práce je i modernizace grafického zobrazení, zejména pak mapy hvězdného systému, kde zapotřebí věnovat pozornost celkovému dojmu ze hry, jelikož hra musí zaujmout na první pohled. Cílovou skupinou hráčů jsou pak žáci středních, případně základních škol, u kterých je první dojem velice důležitý.

V práci je potřeba počítat s celou řadou faktorů, jako je již hotová architektura a celková implementace projektu. Bude zapotřebí odladit nedostatky hotové implementace a připravit dodatečné nástroje, aby byla tato práce co nejvíce využitelná pro další vývojáře projektu Space Traffic.

2 Game design

Game design webové hry Space Traffic je již podrobně popsán v diplomové práci Martina Štěpánka [2]. Tato kapitola se však bude podrobněji zabývat herním designem z hlediska tvorby uživatelského rozhraní a uživatelské přístupnosti.

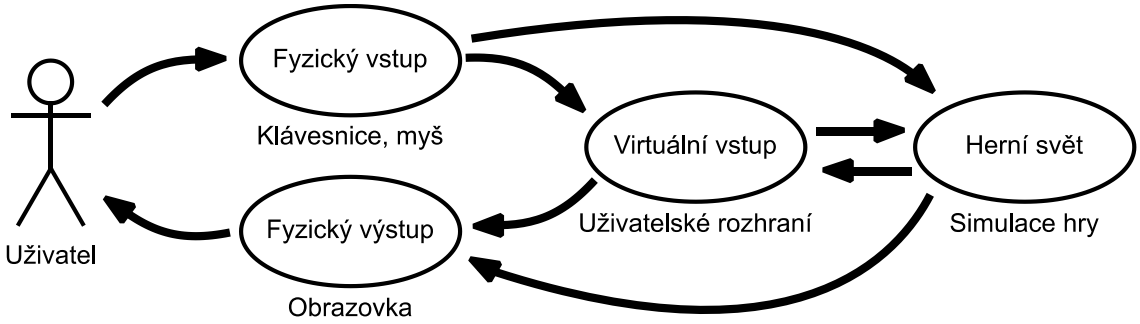
2.1 Herní rozhraní

Hráč musí nějakým způsobem provozovat interakci s herním světem. Obecně je prakticky lhostejné, zda jde o ovladač a televizi, klávesnici a monitor nebo o figurky a hrací desku. Kapitola 13 knihy *The Art of Game Design* se touto interakcí zabývá hlouběji [3].

2.1.1 Fyzický vstup a fyzický výstup

Hráč herní svět nějakým způsobem ovlivňuje svým vstupem, který lze nazvat pojmem fyzický vstup. Hra na jeho činnost pak reaguje fyzickým výstupem [3]. Mezi herní svět a fyzický vstup s výstupem je pak typicky přidán virtuální vstup, tedy uživatelské rozhraní, které je možné ovlivňovat pomocí fyzického vstupu.

Nyní se omezme pouze na fyzický vstup a výstup pro případ webové hry. Ve webové hře je fyzickým vstupem typicky klikání a pohyb myší. Fyzický výstup je pak vyobrazení herního světa na obrazovce v prohlížeči. Virtuálním vstupem je pak uživatelské rozhraní, tedy například odkazy a tlačítka, které slouží k ovládání herního světa. Tuto interakci znázorňuje obrázek 2.1.



Obrázek 2.1: Znázornění fyzického vstupu a výstupu

2.1.2 Interakční smyčka

Hráč by měl z chování vstupu být schopen určit pravidla, podle kterých se svět chová. Například pokud hráč klikne na podtržený text (odkaz), předpokládá, že se nějakým způsobem změní zobrazení. Stejně tak předpokládá, že při kliknutí na tlačítko, které obvykle spouští nějakou činnost (například odesílá formulář k nakoupení více surovin) se tato činnost provede. Důležité je, že pokud se akce z nějakého důvodu

odchýlí od obvyklého chování (například se akce neprovede, protože hráč nemá dostatek peněz), musí o tom být hráč informován okamžitě po pokusu provést akci.

Hráč pak na nově vzniklou situaci opět reaguje, čímž vzniká interakční smyčka, která se neustále opakuje. Hráčova rozhodnutí přitom silně ovlivňuje zobrazovaný výstup. Je tedy velmi důležité, kolik a jaké informace jsou hráči zobrazeny. Příliš málo informací může hráče zmást, pokud je však zobrazeno informací příliš mnoho, může hráč přehlédnout důležitější z nich. Je tedy zapotřebí zobrazovat především informace, které hráč vědět potřebuje (zamýšlená akce se nezdařila) a které vědět chce (podrobné zobrazení majetku ve hře).

2.1.3 Módy rozhraní

Změna obvyklých pravidel herního rozhraní (hráč například klikne na tlačítko smazat a tím pádem se při kliknutí na herní prvek místo zobrazení detailu prvek smaže) se nazývá mód rozhraní. Módy rozhraní jsou výborným způsobem, jak do hry přidat rozmanitost herního prostředí, avšak vzniká zde nebezpečí, že hráč bude zmaten, když si neuvědomí, že se mód rozhraní změnil [3]. Tomuto nebezpečí lze předcházet několika způsoby:

- používat co nejmenší množství módů
- vyhnout se překrývání módů
- co nejvíce změnit vzhled různých módů

Při přidávání každého módu bychom tedy měli zvážit, zda je skutečně nutné jej přidat. Není na škodu mít módů více, ale měla by existovat snaha co nejvíce omezit jejich počet. Herní rozhraní by zároveň nemělo být ve více módech najednou, jelikož pak vzniká u hráče pochybnost o tom, jak bude v takovém případě rozhraní a herní svět reagovat na jeho akce.

Módy by pak v každém případě měly vypadat co nejrůzněji, aby se předcházelo jejich záměně. Toho lze docílit několika způsoby:

- změnou velkého a viditelného prvku na obrazovce
- viditelná změna chování herního prvku
- velká změna zobrazovaných dat
- změna pohledu kamery

Obecně je vhodné použít viditelné změny na obrazovce. Tím se zajistí, že si hráč velmi pravděpodobně všimne, že se chování hry změnilo.

2.2 Zábavnost hry

Hra je zábavná, pokud hráče zaujme a upoutá natolik, aby ji byl ochoten opakovaně hrát. Obvyklým způsobem, jak zjistit, zda je hra zábavná, je dotazovat je jejích hráčů a betatesterů [4]. Kapitola 11 knihy *Game Design Workshop* [4] uvádí několik faktorů,

které jsou klíčové pro zábavnost her. Některé z nich je třeba zvážit i z hlediska grafiky, obsahu hry a uživatelského rozhraní.

2.2.1 Prozkoumávání světa hry

Prozkoumávání světa hry znamená, že hráče lze zaujmout zajímavým rozvržením světa či jeho grafickým zobrazením. Ve hře by mělo být možné zkoumat detaily a postupně objevovat nové možnosti, které nejsou na začátku hry dostupné. Z hlediska grafiky a uživatelského rozhraní tento bod vyžaduje, aby grafické zobrazení herního světa bylo dostatečně interaktivní a mělo originální vzhled.

2.2.2 Zajímavé možnosti hry

Ve hře by měl hráč dělat zajímavá rozhodnutí, která nemají jednoznačně pozitivní či negativní vliv na hráčův postup. Tato možnost je zde zmíněna, protože z hlediska ovládání hry je zapotřebí, aby hráč o této možnosti volby věděl. Herní možnosti by tedy v uživatelském rozhraní měly být srozumitelné a mělo by být jasné, že je potřeba se mezi nimi rozhodnout.

2.2.3 Rozvoj hráče

Rozvojem hráče a jeho postupem se zabývá především game design celé hry, ale z hlediska uživatelského rozhraní je nutné, aby byl hráč o tomto svém postupu neustále informován. Tímto způsobem by byl prakticky povzbuzován v tom, aby pokračoval v hraní. Měl by tedy například neustále vidět klíčové prvky hry, jakým je například množství financí a majetku u obchodních her. Zároveň je vhodné zdůraznit, zda je postup pro hráče pozitivní či negativní, aby měl možnost přemýšlet o svém postupu.

2.2.4 Seberealizace

Prvek seberealizace je popsán jako touha hráče vyjádřit se prostřednictvím herního světa. Kromě toho, že hráčova činnost herní svět přímo ovlivňuje, je tedy vhodné zajistit, aby měl možnost personalizace. Může jít například o změnu vzhledu prvků hry nebo vlastní pojmenování hráčova majetku ve hře. Takto si hráč ke své pozici ve hře vytváří osobní vztah a má pocit, že se podílí na celkovém vzhledu hry.

3 UX design

User experience design je proces zvyšující uživatelské uspokojení pomocí zlepšování použitelnosti, přístupnosti a potěšení poskytovaných interakcí mezi uživatelem a produktem [5]. Z tohoto procesu lze vyvodit zásady tvorby uživatelských rozhraní, jejichž dodržování povede k návrhu a následné implementaci uživatelského rozhraní s uživatelsky přívětivými vlastnostmi. V této kapitole budou stanoveny právě tyto zásady. Celá tato práce se zabývá uživatelským rozhraním webové aplikace, následující principy tedy předpokládají, že jde o zobrazení ve webovém prohlížeči.

3.1 Interaktivní objekty na stránce

Steve Krug definuje svůj „první zákon použitelnosti“ slovy „Nenuťte mě myslet!“ [6]. Znamená to, že uživatel by při prvním pohledu na stránku neměl být zmatený a měl by být schopný intuitivně najít to, co hledá. K tomu napomůže celková jednoduchost rozhraní a eliminace přílišného množství informací. Zároveň by mělo být jednoznačně zřejmé, která část stránky slouží například jako menu nebo vyhledávání.

3.1.1 Odkazy a tlačítka

Dále je také zmíněno, že je důležitý vzhled a konkrétní text popisku interaktivních prvků, například tlačítek. Nápisy na nich by měly být jednoznačné a zřejmé a jejich vzhled by měl sám o sobě implikovat, že jde o tlačítko, na které lze kliknout. Stejně tak by hypertextové odkazy měly mít v celé aplikaci konzistentní vzhled a být rozpoznatelné. Při návrhu těchto klíčových prvků je zapotřebí neustupovat grafickému designu a vzhledu, neboť výsledný efekt může uživatele frustrovat a vést k jeho odchodu ze stránky. V praxi však obvykle nebývá možné udělat vše zcela zřejmé a z této zásady je možné mírně ustupovat. Je ale třeba mít ji neustále na paměti při vytváření menu, tlačítek a hypertextových odkazů.

Prvky na stránce, na které lze kliknout, by také měly být snadno rozpoznatelné, a to na první pohled. Změna vzhledu kurzoru nebo vzhledu prvku až jako reakce na najetí myší není dostačující.

3.1.2 Navigace

Navigačních menu může být na stránce více, ale mělo by být zcela jasně a na první pohled rozpoznatelné, které menu se týká hlavního obsahu na stránce. Uživatel by také neměl mít možnost dostat se do situace, kdy nepozná, v jaké části stránky se právě nachází. Je zapotřebí si uvědomit, že při pohybu na webu uživatel nemá smysl pro:

- velikost – uživatel nemá tušení, kolik stránek může web obsahovat
- směr – uživatel zpravidla nevnímá svůj postup jako „vpřed“ nebo „zpět“

- umístění – bez jasného označení uživatel neví, v jaké části webu se nachází

Z těchto důvodů je potřeba uživateli v navigaci označovat, na které stránce se právě nachází. V případě vnořených umístění je pak vhodné zobrazovat tzv. „drobečkovou navigaci“ [7]. Je také možné uživateli pomoci zobrazováním názvu stránky v jejím nadpise.

3.1.3 Konvence umístění a chování objektů

Pro pozice a chování některých interaktivních objektů existují nepsané konvence, které vyplývají z typického vzhledu většiny webových stránek. Uživatel tak má přirozené tendence hledat prvky uživatelského rozhraní tam, kde se obvykle nachází. Jedná se například o:

- logo (nahoře uprostřed nebo vlevo, při kliknutí přesměruje na homepage)
- údaje o přihlášeném uživateli, tlačítko pro odhlášení (vpravo nahoře, při kliknutí zobrazí možnosti a nastavení přihlášeného uživatele)
- navigační menu (vlevo, nahoře, popřípadě submenu vpravo)
- drobečkové menu, nadpis stránky (nahoře)
- upozornění a zprávy aplikace (nahoře)
- hlavní obsah (uprostřed)
- copyright, informace o autorech (dole)

Tyto konvence je potřeba dodržet, pokud neexistuje podstatný důvod designovat web jinak.

3.1.4 Maximální počet kliknutí

Některé zdroje o designu webových aplikací zmiňují tzv. „Pravidlo tří kliknutí“ [8]. Jedná se o snahu dosáhnout toho, aby uživatel mohl jakékoli své akce dosáhnout pomocí maximálně tří kliknutí. Mnohé zdroje o moderním UX designu však toto pravidlo popírají a označují jej za mýtus [9] [6]. Zároveň tvrdí, že nijak nezáleží na počtu kliknutí, pokud jsou tato kliknutí provedena intuitivně a bez přemýšlení. Z toho vyplývá, že z hlediska moderního UX designu je lepší vyhnout se nepřehledným stránkám s velkým množstvím odkazů a raději nabídnout uživateli zřetelné a jasné možnosti, které se budou dále zanořovat.

3.2 Informace zobrazené uživateli

Účelem webové stránky je typicky zobrazit uživateli informace. Formát a povaha těchto informací jsou důležité pro to, aby je uživatel zaznamenal.

3.2.1 Texty na stránce

Uživatel obvykle nečte texty na webové stránce [6], pokud k tomu nemá důvod – například pokud je právě textový obsah to, co hledal. Místo čtení uživatel typicky

stránku „skenuje“, tedy ji rychle zhlédne a hledá slova a prvky, které ho případně zaujmou. Z toho vyplývá několik zásad při psaní textu a návrhu uživatelského rozhraní:

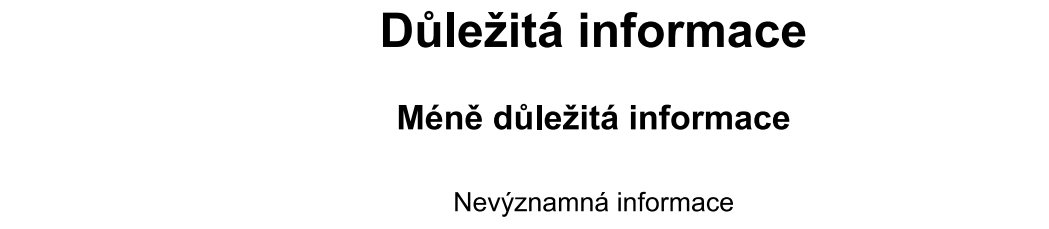
- vyhnout se psaní zbytečných slov
- nepsat na stránku „happy talk“
- nepsat na stránku podrobné instrukce

U složitých textů je zpravidla možné drasticky redukovat počet slov, aniž bychom ztratili jakoukoli informaci. Zároveň je nutné nepsat na stránku tzv. „happy talk“ [6], tedy odstavec vítající uživatele a vyjadřující naši radost nad tím, že navštívil naši stránku. Stejně tak je nevhodné psát na stránku jakékoli další texty s informacemi, které uživatele nezajímají. Rovněž je nevhodné a prakticky i zcela zbytečné psát na stránku podrobné instrukce, které se netýkají aktuální činnosti uživatele, protože je velmi pravděpodobně uživatel nebude číst.

3.2.2 Důležitost informací

Některé informace sdělované uživateli jsou důležitější než jiné a zobrazení a vzhled těchto informací by měl odpovídat jejich důležitosti. Podstatnější informace jsou tedy zobrazeny větším či tučnějším písmem, popřípadě se nacházejí na viditelnějším místě stránky nebo jsou zvýrazněny barevně. Účelem je dosáhnout toho, že si uživatel všimne důležitých informací, když na stránku pohlédne. U důležitějších informací je také pravděpodobnější, že uživatel hledá právě je. Tento koncept naznačuje obrázek 3.1.

Pro tento účel je vhodné stanovit si konvence, kterými se bude řídit celá aplikace. Tak je možné dosáhnout toho, že si uživatel zvykne na rozvržení stránky a tím snadno rozliší důležitost informací.

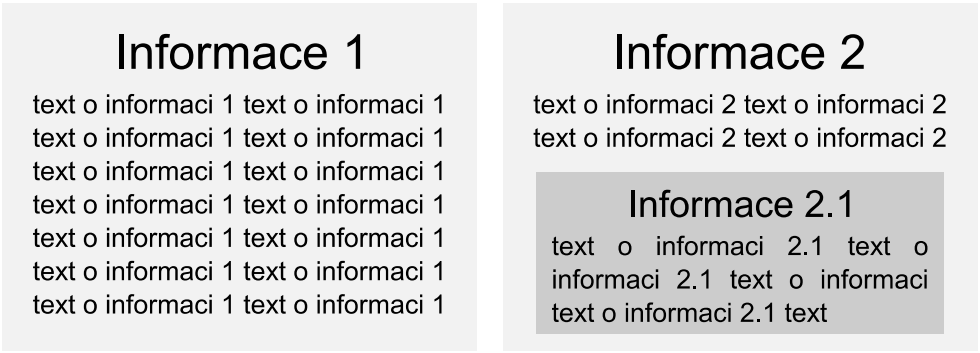


Obrázek 3.1: Zobrazení důležitosti informací

3.2.3 Propojení informací

Informace, které souvisejí s jinými, by měly být logicky propojeny i vizuálně [6]. Je například vhodné spojit informace jejich vložením pod společný nadpis, vizuálně je odlišit od ostatních informací a nebo je vložit do jasně definované oblasti na stránce.

Zároveň platí, že vnořené informace (tedy informace, které jsou podkategorií jiných informací) by měly být vnořené i vizuálně. Typicky se jedná o zobrazení podkategorií například v e-shopech. Příklad vizuálního rozdělení je na obrázku 3.2.



Obrázek 3.2: Příklad vizuálního rozdělení informací

3.3 Barvy

Barevné schéma na stránce může usnadnit uživateli pochopení významu jednotlivých ovládacích prvků a zobrazených informací. Například je obecně přijatá konvence, že při volbě uživatele mezi možnostmi „ano“ a „ne“ je tlačítko „ano“ zelené a tlačítko „ne“ červené. Lze si však zvolit libovolné barevné schéma a zvolit si konvenci, která vyhovuje designu webové stránky.

Volba barev je také důležitá z hlediska celkové přitažlivosti stránek. Pokud jsou navíc barvy originální, může si je uživatel spojit přímo se stránkou a vytváří se tak povědomí o značce.

3.3.1 Volba barev

V zásadě je pro design webu vhodné zvolit si omezené množství různých barev a během návrhu používat výhradně tyto barvy [10] s výjimkou bílé a černé jako kontrastních barev pro text. K volbě vhodného barevného rozlišení existuje několik postupů:

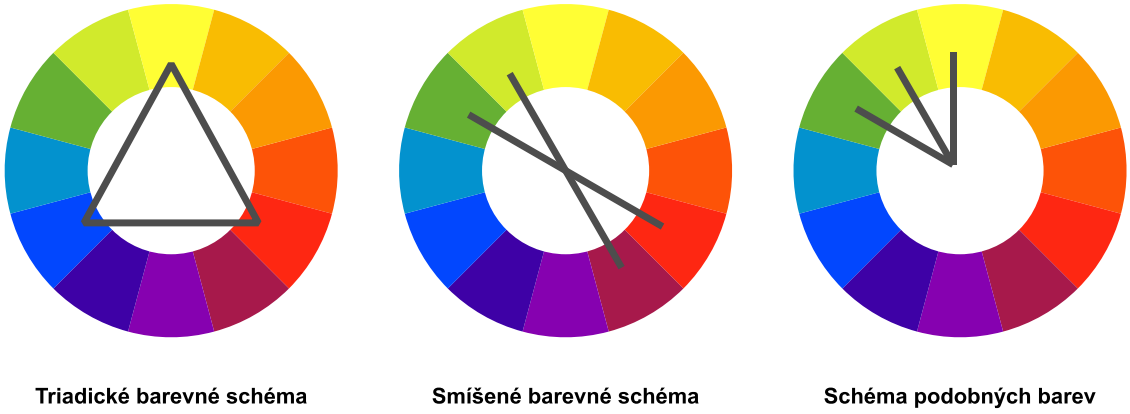
- triadické barevné schéma
- smíšené barevné schéma
- schéma podobných barev

Volby barev pomocí těchto schémat jsou znázorněny na obrázku 3.3.

Triadické barevné schéma volí tři barvy z barevného spektra, které jsou od sebe v barevném spektru stejně vzdálené a tvoří tudíž trojúhelník. Tyto tři barvy jsou tak zcela různé a lze je použít nezávisle na sobě.

Smíšené barevné schéma volí vždy po dvou barvách z opačné strany barevného spektra. Tímto způsobem lze získat optimální barvy a přitom mít při designu stránky dostatečnou svobodu volby.

Schéma podobných barev volí několik barev, které jsou blízko sebe na jedné části barevného spektra. Obvykle se tyto tři barvy liší především odstínem. Je však vhodné volit barvy také s různým kontrastem.



Obrázek 3.3: Volba barevného rozložení podle schémat

3.3.2 Barva textu

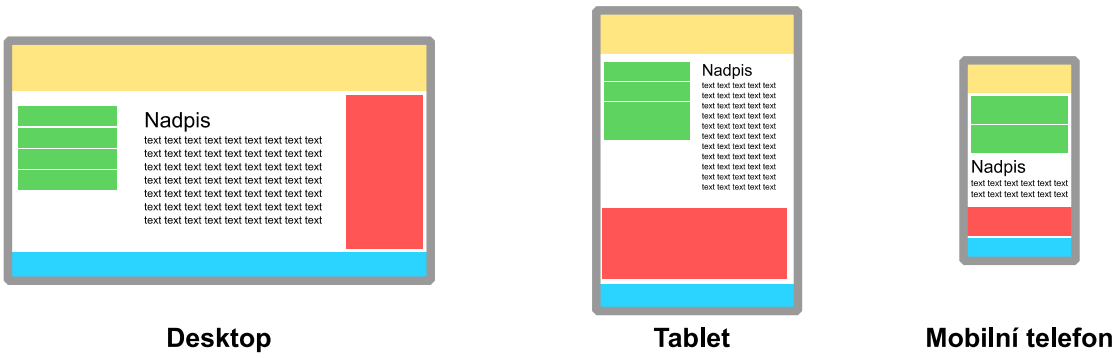
Text by měl být v každém případě co nejvíce kontrastní ke svému pozadí. To znamená, že jeho barvu je vhodné volit buď na opačné straně barevného spektra, či použít bílý text na tmavých pozadích a černý na světlých. Pokud pozadí pod textem nemá jednotnou barvu a mění se, je zapotřebí zajistit, aby zvolenou barvu textu bylo možné použít v každé části pozadí.

3.4 Responzivní design

Jedná se o schopnost webu přizpůsobovat svůj obsah aktuální velikosti zobrazení (velikosti okna prohlížeče, resp. displeje) [11]. Této vlastnosti lze dosáhnout pomocí technologie CSS. Tato technologie umožňuje použít různé postupy a techniky, kterými lze responzivní design vhodným způsobem vytvořit. Jedná se například o:

- grid view (rozdělení stránky na části)
- media queries (různá CSS pro různé velikosti zobrazení)

Existují také podpůrné CSS knihovny, usnadňující tvorbu responzivního designu, jde například o W3.CSS [12] nebo Bootstrap [13]. Příklad zobrazení webu s responzivním designem na různých zařízeních je na obrázku 3.4.



Obrázek 3.4: Příklad zobrazení responzivního webu

4 Usability testování

Testy použitelnosti, neboli usability testy slouží k pochopení, jakým způsobem uživatel stránku používá a jak se na ní chová. Jde o iterativní proces, ve kterém nacházíme problémy související s UX designem. Tato kapitola se zabývá problematikou toho, jak provádět usability testy webové aplikace s omezenými zdroji. Veškeré informace v této kapitole jsou čerpány z knihy *Dont make me think!* [6].

4.1 Jak testovat

Při usability testech je vybrán uživatel nebo více uživatelů, které pozorujeme při práci s testovanou stránkou. Snažíme se uskutečnit především dva druhy testování:

- „Get it“ testing
- Key task testing

„Get it“ testování spočívá v zjišťování, zda uživatel vůbec pochopil koncept a princip fungování stránky. Při tomto testování pozorujeme zejména jeho reakce na stránku a zda se na stránce orientuje. Nijak nezasahujeme do uživatelovy činnosti.

Key task testing je testování toho, zda je uživatel schopen splnit úkol, který mu zadáme. Konkrétní podobu úkolu je vhodné volit tak, aby samotného uživatele zajímal výsledek. Například úkol u e-shopu s potravinami „Najděte potravinu, kterou jste naposledy sháněli.“ je lepší než „Najděte bezlepkové těstoviny za méně než sto korun.“. Uživatel je pak více emočně vázán a tím pádem se více snaží úkol splnit.

4.2 S kým testovat

Testovat s jedním uživatelem je stále o 100% lepší než netestovat s žádným [6]. Uživatel, se kterým testujeme, nemusí být součástí cílové skupiny, na které se naše stránka zaměřuje a nemusí mít žádnou kvalifikaci. Také je lepší testovat projekt v raných fázích vývoje menším počtem uživatelů, než v pozdější fázi větším počtem, protože tak mohou výsledky projekt více pozitivně ovlivnit.

4.2.1 Počet testovacích uživatelů

Jednotlivé testy můžeme provádět s větším či menším počtem uživatelů. Pokud testujeme s větším počtem uživatelů (větším počtem se myslí zhruba 8 osob), je pravděpodobné, že objeví více závažných problémů než skupina s menším počtem uživatelů (zhruba se 3 osobami). Pokud však místo jednoho testu s větším počtem uživatelů provedeme dva testy s menším počtem uživatelů, odhalíme více problémů, neboť v mezičase budeme moci odstranit zásadní problémy objevené v prvním testu a ve druhém testu tak uživatelé budou moci objevit problémy, které nebyly odhalitelné v prvním testu. Z toho vyplývá, že s omezenými zdroji testovacích uživatelů je efektivnější uspořádat více testů s menším počtem uživatelů.

4.2.2 Komunikace s uživateli

Při usability testování budeme v roli pozorovatele v průběhu testu komunikovat s testovacími uživateli. Před započetím bychom uživateli měli zdůraznit několik důležitých faktů:

- snažíme se o vylepšení použitelnosti aplikace
- testujeme aplikaci, *ne* uživatele
- uživatel nemůže během testu udělat nic špatně
- uživatel se nemusí stydět upozorňovat na chyby aplikace
- během testu si pozorovatel bude dělat poznámky o *aplikaci*
- pro výsledky testu je lepší, když uživatel bude přemýšlet nahlas
- uživatel se může zeptat na cokoli, ale v zájmu testu možná pozorovatel neodpoví hned nebo zcela jednoznačně, aby zjistil, jak se uživatel vypořádá s problémem bez pomoci třetí osoby
- pořizujeme záznam (video, nahrávání hlasu, snímky obrazovky)

Účelem je navodit prostředí, ve kterém se uživatelovo chování co nejvíce podobá jeho chování při běžném používání aplikace. Uživatel by neměl mít pocit, že špatné plnění úkolu je jeho chyba a neměl by se bát upozornit na nedostatky testované aplikace. Pokud je uživatel z testu nervózní, je potřeba situaci odlehčit například nezávaznou osobnější konverzací.

4.3 Jak zhodnotit výsledky

Co nejdříve po skončení testu (nebo více testů po sobě) je vhodné sejít se s vývojáři aplikace a zhodnotit výsledky. Nejprve je třeba výsledné problémy protřídit a zhodnotit, které z nich je potřeba vyřešit. *Problémy k řešení jsou obvykle očividné pro každého, kdo sleduje průběh testu.* [6]. Dále se rozhoduje, jakým způsobem problém vyřešit. Usability testování je cyklický a iterativní proces, takže řešení nemusí být zcela dokonalé a lze jej vylepšovat v dalších iteracích.

4.3.1 Typické problémy

Uživatelé se neorientují v aplikaci. Nechápou, k čemu aplikace slouží a jak a proč se používá, případně si myslí, že účel aplikace chápou, ale pletou se. Takový problém lze řešit zviditelněním hlavních prvků aplikace.

Dalším typickým problémem je, že uživatelé hledali na stránce určitá slova, která nenalezli. Obvykle to znamená, že rozvržení a kategorizování použité na stránce se neshoduje s tím, které by použil samotný uživatel, nebo by uživatel použil jiné, podobné slovo.

Třetím typickým problémem je informační šum. Je způsoben tím, že na stránce je příliš informací a prvků, ve kterých se uživatel „ztrácí“ a často přehlédne to, co hledá.

Řešením je typicky buď snížit množství informací na stránce nebo více zviditelnit klíčové prvky a informace.

4.3.2 Třídění a řešení problémů

Při vyhodnocování, které problémy vyřešíme a které ne, se lze řídit několika zásadami. Můžeme například ignorovat problémy, ze kterých se uživatel téměř okamžitě a bez pomoci dostane. Typickým příkladem je hledání položky, která může být pod dvěma různými kategoriemi. V takovém případě část uživatelů položku najde ihned a jiná část uživatelů se splete. Uživatelé, kteří se spletou, se však obvykle vrátí o krok zpět a zvolí správnou kategorii.

Při řešení problémů bychom si měli dát pozor na impulzivní přidávání nových prvků a informací, například vysvětlení nebo instrukcí. Často však lepší řešení spočívá v odebrání jiného prvku, který odvádí od významu stávajícího řešení a tak způsobuje daný problém.

Pokud při testování uživatel zmíní, že by potřeboval novou funkcionalitu v systému, je třeba brát tuto informaci s nadhledem a nevěnovat se ihned implementaci. Uživatelé aplikace totiž v danou chvíli často mají dobrý a funkční způsob, jak dosáhnout stejného výsledku a ve skutečnosti by pravděpodobně nebyli ochotní přecházet na jiný, byť mírně jednodušší způsob.

Během testování je efektivní hledat vylepšení, která nejsou náročná na implementaci a přitom jsou pro koncového uživatele velmi hodnotná. Typicky jde o tlačítka a odkazy v rozhraní nebo řešení, která nebyla hledána během implementace a jsou očividná při prvním testování na skutečném uživateli.

Zároveň je také důležité při změnách aplikace myslet na dopady, které tato změna může mít. I malá změna se totiž může velice projevit a způsobit problémy jinde v aplikaci.

5 Projekt Space Traffic

Tato kapitola popisuje studentský projekt Space Traffic, jeho účel a stručnou historii jeho stavu. Jedná se o projekt webové hry, vyvíjené výhradně studenty Katedry informatiky a výpočetní techniky Fakulty aplikovaných věd Západočeské univerzity v Plzni.

5.1 Účel projektu

Projekt jako takový má několik stanovených cílů. Přestože se tato práce zabývá přímo analýzou a implementací hry, jsou zde zmíněny dva hlavní cíle pro hlubší pochopení významu projektu a hry Space Traffic.

5.1.1 Vzdělávání hráčů

Space Traffic je webová hra, ve které se hráči mají zabývat přepravou zboží za účelem zisku a růstu jejich vlastních herních zdrojů. V tomto se Space Traffic podobá běžným dostupným hrám. Hlavní částí, která by měla sloužit vzdělávání hráče, je však možnost neovládat obchodování ručně, ale vytvářet skripty a programy, podle kterých se lodě budou chovat. Cílovou skupinou jsou především žáci základních a studenti středních škol. Této cílové skupině by pak měla hra zábavnou formou představit základní možnosti programování. Gamedesign hry je navržen tak, aby bylo možné hru hrát i bez nutnosti používat skriptování lodí, avšak psaní skriptů je pro hráče jedinou možností, jak dále rozvíjet svůj herní úspěch [2].

5.1.2 Vzdělávání vývojářů

Druhým, neméně významným a často opomíjeným cílem je vzdělávání studentů pracujících na projektu. Space Traffic získává své lidské zdroje výhradně prostřednictvím zadávání alternativních semestrálních prací pro předměty na Katedře informatiky a výpočetní techniky. Pod záštitou projektu Space Traffic se vývojáři učí pracovat v týmu i v případě, že daný předmět týmovou práci přímo nepodporuje. Pro týmové práce pak projekt nabízí stabilní a členy katedry ovlivnitelný koncept, kterým lze vzdělávat studenty pracující na projektu. Jelikož je Space Traffic webová technologie a zabývá se herním prostředím, je zajímavý pro studenty se zájmem o tato odvětví informačních technologií.

Projekt má i svou znalostní bázi a ukazuje vývojářům, jak je zachovávání znalostí nezávisle na lidech nezbytné pro hladký chod jakéhokoli projektu. Space Traffic je kvůli svému konceptu, kdy se jednotliví vývojáři rychle střídají a zpravidla se navzájem neznají, mimořádně citlivý na kvalitu a rozsah dokumentace a znalostní báze.

Součástí projektu je i virtuální server, jehož hardware je spravován katedrou nezávisle na studentech pracujících na projektu, ale jeho software, poskytované služby, správa a údržba jsou převážně pod kontrolou správců projektu. Výhodou toho je, že si

student může na výukovém prostředí zkusit práci se skutečným serverem, aniž by pracoval s cennými daty či riskoval způsobení významnějších škod způsobených případnou nefunkčností některé se služeb serveru.

Největší výukový přínos má pak projekt pro jeho správce. V akademických letech 2014/2015 a 2015/2016 byl autor této práce jedním ze dvou správců projektu spolu s Bc. Veronikou Švecovou. Za tuto dobu získal velké množství zkušeností, zejména pak v oblastech praktického projektového řízení, vývoje architektonicky složitých aplikací, vedení týmu vývojářů, správy softwarových zdrojů projektu a dlouhodobého udržování znalostní báze. Možnost svěřit studentovi projekt, nad nímž má prakticky plnou kontrolu a také plnou zodpovědnost, avšak stále se pohybuje ve školním prostředí, činí ze Space Trafficu zajímavou vzdělávací příležitost pro studenty, kteří svému profesnímu a případně akademickému rozvoji chtějí věnovat více času.

5.2 Vývoj projektu

Implementace hry Space Traffic vzniká výhradně jako výstup semestrálních, bakalářských a diplomových prací studentů. Z tohoto důvodu obsahuje hra a její uživatelské rozhraní větší množství ucelených součástí. Protože však není možné zadat v rámci semestrální práce drobnější úpravy stávající implementace, vzniká zde technický dluh. Příkladem může být chybějící podpora pro provádění akcí herního světa, získávání id přihlášeného uživatele nebo nefunkční prototypy v uživatelském rozhraní. Část těchto nedostatků byla vyřešena autorem této práce v rámci předmětu KIV/NET v letním semestru roku 2015. Aby byla však hra použitelná pro uživatele, je potřeba tyto nedostatky odstranit úplně. Účelem této diplomové práce tak není pouze vytvořit uživatelské rozhraní, ale i vyplnit hluchá místa v implementaci a převést celou hru do hratelného stavu.

5.3 Verze projektu

V letech 2012/2013 a 2013/2014 došlo k nárůstu technického dluhu do té míry, že byla další implementace velice náročná. Navíc výsledná verze, která se nachází na katedrálním repozitáři pod verzovacím systémem SVN obsahovala velké množství chyb, nekvalitního kódu a chybějících funkcionalit. Navíc zcela chyběla dokumentace těchto funkcionalit na wiki projektu.

Po podrobné analýze kódu, konzultacích s garanty projektu a bývalými vývojáři projektu bylo rozhodnuto o vrácení se k verzi z konce roku 2011/2012 a jejím převedení na repozitář GitHub [14]. Pro ten účel byly podniknuty kroky, aby bylo možné z hlediska autorských práv převést celý projekt pod licenci Apache License 2.0. Do této verze pak byly převáděny funkcionality z původní verze na SVN, které byly analýzou kódu vyhodnoceny jako použitelné. Jednalo se například o herní achievementy, převedené autorem této práce v rámci předmětu KIV/NET. Nicméně se převod

některých částí (funkcionalita Budovy a pozemky) nezdařil z důvodu personálních problémů.

6 Stávající implementace UI

Stávající implementací projektu se myslí verze popsaná v odstavci 5.3 z konce akademického roku 2014/2015. Tato verze obsahovala základní uživatelské rozhraní, které z větší části sloužilo převážně jako placeholder. Původní správce projektu Petr Vogl stanovil požadavek na vylepšení a zkompletování tohoto uživatelského rozhraní, což vedlo ke vzniku zadání této diplomové práce. Tato kapitola analyzuje verzi uživatelského rozhraní z konce roku 2014/2015.

6.1 Architektura

Hra Space Traffic se skládá z několika podprojektů. Jedná se o:

- Core (entity, výpočetní postupy)
- GameServer (herní simulace, poskytování služeb a přístup k databázi)
- GameUI (MVC framework, zobrazování webového UI)
- GameServerScripts (skripty pro game server, spíše pro budoucí použití)
- StarSystemEditor (editor hvězdných systémů nezávislý na hlavní aplikaci)

Všechny tyto podprojekty využívají jazyk C# a framework .NET 4.0. Interakce s uživatelem probíhá výhradně v projektu GameUI, který je postaven na MVC frameworku ASP.NET 3. GameUI se ale musí dotazovat GameServeru na stav hry a ovlivňovat herní svět pomocí akcí. GameUI také nemůže přímo přistupovat k databázi, ale používá komunikaci s GameServerem.

6.2 Použité technologie

Tato část se zabývá použitými technologiemi na serverové (podprojekt GameServer) a klientské (podprojekt GameUI) části hry Space Traffic. U každé je potřeba posoudit její možnosti při dalším vývoji. Také je na místě zvážit jejich aktualizace, neboť verze projektu je několik let stará.

6.2.1 GameServer

Z hlediska uživatelského rozhraní zajišťuje GameServer především práci s daty a reakce herního světa na příkazy uživatele, jako je například posílání lodí z planety na planetu. K tomu využívá několika technologií:

- Entity Framework
- Microsoft SQL Server 2008
- Microsoft Windows Communication Foundation (WCF)
- NLog

GameServer je dle architektury Martina Štepánka [2] jediná součást hry, která má přístup k databázi. K tomu využívá **Entity Framework** (verze 4.3.1), což je otevřená knihovna využívající ORM pro přístup k databázi. Ve Space Trafficu se využívá přístup Code First. Tato verze byla shledána jako dostačující pro účely modernizovaného uživatelského rozhraní.

Microsoft SQL Server 2008 je v projektu použit jako databázové prostředí, které v dostatečné míře zajišťuje potřeby práce s daty v projektu.

Microsoft Windows Communication Foundation (WCF) je technologie, sloužící pro komunikaci mezi projekty GameUI a GameServer. Funguje jako servisně orientovaná architektura (SOA). Tyto dva projekty tak mohou být spuštěny nezávisle na sobě a mohou být spuštěny i na různém hardware, pokud k sobě mají přístup po síti. Tato architektura se však vyznačuje složitější implementací. Pro účely komunikace uživatelského rozhraní s herní simulací plně dostačuje.

Pro logování a záznamy je použita knihovna pro prostředí .NET **NLog.** Tato knihovna vytváří záznamy o činnosti herního prostředí. Její činnost je pro účely projektu dostatečná.

6.2.2 GameUI

Herní uživatelské rozhraní je realizováno pomocí MVC frameworku ASP.NET [15]. Pro práci s uživatelským rozhraním pak používá několik dalších technologií:

- HTML5
- jQuery
- AJAX
- SVG
- CSS3
- LESS, dotless

HTML5 [16] je značkovací jazyk sloužící pro definici zobrazení webové stránky. V současnosti je již zavedeným standardem podporovaným všemi moderními prohlížeči jako jsou Google Chrome nebo Mozilla Firefox. Použití této technologie je nezávislé na předchozí instalaci, avšak je vhodné prohlížeč informovat o jeho použití prostřednictvím deklarace DOCTYPE. Ve hře Space Traffic byl DOCTYPE pro HTML5 již deklarován a aplikace tak používá poslední stabilní verzi.

jQuery [17] je knihovna nad jazykem JavaScript, který je interpretován všemi moderními prohlížeči. Zjednodušuje práci s DOM elementy, tedy s objekty definované pomocí HTML, oproti použití samotného JavaScriptu. Také funguje jako další úroveň abstrakce nad dalšími vlastnostmi jazyka JavaScript. K tomu poskytuje některé knihovní funkce, které nejsou součástí JavaScriptu a přímo tím podporuje vývoj na klientské části aplikace. V projektu Space Traffic je v současnosti použitá verze 1.6.4, která je poměrně zastaralá (je již dostupná stabilní verze jQuery 2.2)

a nepodporuje mnoho užitečných knihovních funkcí obsažených v novějších verzích, což může mít velký vliv na vývoj a rychlost uživatelského rozhraní. Při případné aktualizaci je však třeba brát ohled na stávající implementaci, neboť ve verzi jQuery 1.9 bylo mnoho knihovních funkcí dostupných ve starších verzích změněno nebo vyřazeno [18].

AJAX (Asynchronous JavaScript and XML) [19] je obecná technika, která umožňuje komunikaci klientské části aplikace v prohlížeči se serverovou částí (v tomto případě je serverovou částí myšlený ASP framework z podprojektu GameUI). Jeho podpora je součástí knihovny jQuery. Pomocí AJAXu lze vytvářet HTTP požadavky v již načtené stránce asynchronně a na pozadí tak, že nijak neovlivní stávající zobrazení stránky (například nevyžaduje její obnovení). Přijatou odpověď pak lze vnitřně zpracovat JavaScriptem a knihovnou jQuery a případně zobrazit uživateli nové informace.

Jako formát dat lze použít například JSON nebo lze pomocí požadavků získávat přímo HTML kód, který je dle potřeby umístěn na stránku. Aktualizace této technologie přímo závisí na použité verzi jQuery.

SVG (Scalable Vector Graphics) [20] je technologie využívající značkovací jazyk XML k definici vektorové 2D grafiky, která je poté vykreslována prohlížečem. V projektu Space Traffic je tato technologie využita především k vykreslování map hvězdných systémů. Podpora SVG je v dnešních moderních prohlížečích již obsažena, avšak v případě některých složitějších prvků SVG se podpora jednotlivých prohlížečů různí [21]. SVG jako obecně podporovaný vektorový formát nabízí oproti rastrovým formátům několik výhod [22]:

- škálovatelnost (možnost změny velikosti zobrazení bez ztráty kvality)
- podpora animací
- možnost vytvářet jej a měnit programovým kódem (např. pomocí JavaScriptu)
- zdrojový kód je čitelný i pro vývojáře
- přenositelnost (podporují jej moderní webové prohlížeče)

CSS3 (Cascading Style Sheets) [23] slouží pro definování vzhledu HTML elementů. Je tak možné oddělit definici rozvržení obsahu a definice jeho vzhledu. CSS popisuje například velikost HTML elementu, barvu jeho textu, jeho umístění apod. Poslední verze CSS3 poskytuje navíc další možnosti [24]:

- selektory (více možností, jak vybrat element k uplatnění CSS pravidla)
- textové efekty
- 2D/3D transformace
- animace
- zaoblení rohů
- barevné gradienty

- použití obrázku při definici okraje
- media queries (definice vzhledu za různých podmínek – například různá zobrazení při různých velikostech displeje, slouží pro tvorbu responzivního designu)

CSS3 je podporováno moderními prohlížeči a proto je možné jeho možnosti využít při práci na novém uživatelském rozhraní.

LESS [25] je dynamický jazyk pro tvorbu stylesheetů. Jde o rozšíření CSS, které umožňuje použití dynamických prvků, například:

- proměnné
- funkce
- mixiny (možnost „oddědit“ od jiné CSS třídy)
- import jiných less (css) souborů
- vnořená pravidla (zjednodušující zápis CSS selektorů)
- výpočty (například výpočet šířky elementu jako dvojnásobku jeho výšky)
- knihovní funkce (například pro zesvětlení barvy)

LESS je prakticky jen způsob zápisu CSS, protože se jeho zápis vždy před interpretací prohlížečem převede na CSS. To lze provádět buď ještě na serveru či až přímo v prohlížeči. V případě projektu Space Traffic je použita knihovna dotless, která převod provádí na serverové části (GameUI).

6.3 Současný stav

Tento odstavec obsahuje kritické zhodnocení jednotlivých částí současného uživatelského rozhraní ve hře. Celkový vzhled této verze můžeme vidět na obrázku v příloze B.

6.3.1 Mapa hvězdného systému

Uprostřed okna prohlížeče je vyobrazena mapa hvězdného systému, se kterým může hráč pracovat. Podle původního návrhu se má jednat o zobrazení v radarovém stylu. Jde však o klíčový prvek hry, který navíc ovlivňuje hráčův první dojem. Z hlediska dnešních technologií a hráčských standardů je celkový grafický vzhled hvězdy, planet, jejich oběžných drah, popisků a červích děr nedostačující.

Pro vykreslení hvězdného systému je použita technologie SVG, jejíž potenciál je však značně nevyužitý a již v době první implementace se počítalo s případným rozšířením. Mapou lze kliknutím a tažením pohybovat do stran. Planety se pak pohybují podle implementovaného fyzikálního modelu po svých drahách a popisky je následují. Z hlediska implementace jsou planety, hvězdy a červí díry přehledně popsány pomocí objektů vytvořených v JavaScriptu a vykreslovány pomocí dalších JS objektů, které k nim přísluší.

K pohybům hvězdným systémem by tak bylo žádoucí přidat možnost přiblížení a oddálení. K tomu by se dalo využít kolečko myši či tlačítka v uživatelském rozhraní. Hvězdná mapa by tak mohla být větší a podrobnější, neboť by si hráč mohl jednotlivé detaily přiblížit. Aby se s mapou pohodlně pracovalo, měla by se přibližovat vždy k aktuální pozici kurzoru, pokud je přibližována pomocí kolečka myši.

Grafické zobrazení hvězdy, planet a červích děr by mohlo využít potenciál SVG či CSS3 animací, čímž by se staly graficky atraktivnější pro hráče. Bylo by například možné využít transformací objektů, které by byly zobrazovány na pozadí planety jako SVG pattern.

Oběžné dráhy planet jsou pak na mapě příliš výrazné a působí spíše jako rušivý prvek. Jejich zobrazení by mělo být spíše takové, aby byly na hranici viditelnosti a případně by se mohly zvýrazňovat při přejetí kurzorem. Celkově jsou také planety příliš blízko sebe a jejich oběžné dráhy se protínají, což působí velmi nepřehledným dojmem. Planety a jejich dráhy by se proto měly oddálit. Díky možnosti přiblížit a oddálit hvězdný systém pak nebude vznikat problém toho, že je hráč všechny neuvidí.

Popisky planet a červích děr jsou graficky poměrně nezajímavé a přitom by jejich změna nevyžadovala větší množství nového kódu. Bylo by také možné u nich zobrazovat informační prvky či tlačítka. Gamedesign Space Trafficu například počítá s tím, že obchodníci a základny budou jen na některých planetách. Tyto planety by pak měly být nějakým způsobem označeny a prostor jejich popisku je pro to ideální.

6.3.2 Pozadí hry

Na pozadí se v současné verzi opakuje rastrový obrázek s hvězdnou oblohou. Pro účely hry je dostačující. Pokud ale budeme rozšiřovat možnosti hvězdné mapy o přibližování a celkově zvýšíme interaktivnost mapy, jak je popsáno v odstavci 6.3.1, bude toto pozadí působit zbytečně staticky. Pozadí by mohlo dynamicky reagovat na interakci uživatele s mapou, čímž by prohloubilo jeho hráčský zážitek.

Pozadí by se mohlo skládat z více vrstev hvězd, které by pak reagovaly různě na hráčovy akce. Tato technika se nazývá Paralax scrolling [26]. Při přiblížení by se částečně mělo přibližovat i hvězdné pozadí, avšak v menší míře než samotná mapa. „Vzdálenější“ vrstvy hvězdného pozadí by pak měly reagovat v menší míře než „bližší“ vrstvy. Stejný princip by pak platil i při pohybu mapou do stran. Tímto postupem by pak vznikl efekt, který by na hráče působil dojmem, že je mapa plastická.

6.3.3 Prvky uživatelského rozhraní

Hvězdná mapa je obklopena jednotlivými prvky uživatelského rozhraní, jako je hlavní menu vlevo, kontextové menu vpravo a informační lišta nahoře. Graficky tyto prvky působí spíše zastarale a je jich na obrazovce příliš mnoho najednou. Jejich rozvržení také způsobuje, že mapa je „zavřená“ mezi nimi, hráč je jimi rozptylován a zhoršují tak jeho interakci s mapou. Pro řešení tohoto jevu se nabízí skrýt tyto prvky

a zobrazovat je pouze v případě, kdy s nimi hráč chce pracovat. Hráč by také měl mít možnost je zavřít, když se bude chtít vrátit k mapě.

První dojem je pro celkový hráčský zážitek velice důležitý, takže by prvky uživatelského rozhraní měly být ve výchozím nastavení skryté, aby se dal prostor graficky zajímavější hvězdné mapě. Zobrazeny by pak měly být jen ty ovládací prvky, které bude hráč potřebovat pro započetí hry.

Při potřebě zobrazit větší množství informací či možností se pak otevírá hlavní okno, které se nachází uprostřed. Je to dobrý způsob, jak odvést pozornost hráče od mapy a nechat ho věnovat se jiným věcem. Hlavní okno by mělo zůstat zachováno s možností ho skrýt. Zobrazovat přes celou obrazovku by se však mělo jen v případech, kdy přímo nesouvisí s děním na hvězdné mapě nebo když je obsah příliš velký a nevejde se jinam.

6.3.4 Chybějící rozhraní

V projektu je mnoho funkcionalit, které uživatelské rozhraní úplně postrádají a není z různých důvodů řešeno. Je mezi nimi i několik klíčových funkcí hry. Jedná se o tyto funkcionality:

- plánování cest lodí
- opravování lodí
- tankování paliva do lodí
- nákupy a prodeje na planetě

Do databáze je také zapsáno velké množství dat, které se ale ve hře nikde nezobrazuje, i když jsou tyto informace pro hráče zásadní a prakticky nepostradatelné. Jde v první řadě o:

- informace o zboží na planetách
- informace o lodích na planetách
- informace o zboží na lodích
- detaily stavu jednotlivých lodí
- zobrazení aktuálního počtu kreditů a lodí hráče (v současné verzi jde pouze o placeholdery)

Dále chybí i grafické rozhraní k prvkům, bez kterých se uživatel sice obejde, ale usnadňovaly by mu interakci se hrou nebo přispívaly k celkovému dojmu hry. Těchto prvků je obrovské množství, lze však zmínit několik nejdůležitějších:

- seznam základen
- seznam lodí
- popisy planet a soustav

Většina odkazů v menu pak odkazuje na nefunkční prototypy. Funkčnost některých z nich však není pro hru v této fázi vývoje důležitá a proto je možné je z rozhraní úplně vypustit. Jde například o:

- seznam událostí
- zprávy mezi uživateli
- žebříčky a skóre
- uživatelský manuál

Některé další prvky jsou implementovány v rámci jiných prací na projektu. Jde jmenovitě o uživatelský profil, tlačítko pro odhlášení a všechen obsah, který se zobrazuje nepřihlášenému uživateli. Ty není zapotřebí v této práci rozebírat a autor této práce na jejich vývoj dohlíží z pozice správce projektu.

6.3.5 Grafika herních objektů

Hra aktuálně používá nejednotný grafický styl, který působí nekonzistentně a rušivě. Tento jev můžeme pozorovat například při porovnávání obrázků lodí při jejich nákupu, vyobrazení planet a prvků v menu. Grafický styl všech tří se silně odlišuje. Při návrhu rozhraní by bylo záhodno navrhnout jednotný grafický styl pro všechny prvky ve hře a pokud možno i pro uživatelské rozhraní, aby bylo zobrazení konzistentní.

7 Návrh nového UI

Na základě analýzy stávajícího stavu z kapitoly 6 a nastíněných hrubých řešení uvedených problémů můžeme navrhnout nové uživatelské rozhraní. Toto rozhraní by mělo být pro uživatele přívětivější a atraktivnější. Tato kapitola popisuje návrh nového rozhraní včetně předpokládaného chování aplikace.

7.1 Požadavky

Nejprve je třeba zvážit reálné technické požadavky projektu a aplikace. Vzhledem k podmínkám, způsobu vývoje a univerzitnímu prostředí se požadavky mírně odchylují od běžných požadavků na projekty podobného typu.

7.1.1 Podpora prohlížečů

Cílovou skupinou hráčů jsou především žáci základních a studenti středních škol. Také se u těchto hráčů předpokládá zájem o hry a informační technologie. Projekt se také objevuje při prezentaci Fakulty aplikovaných věd, například na Dni otevřených dveří. Předpokládá se tedy, že tato skupina hráčů bude používat vyspělý moderní prohlížeč v jedné z posledních verzí. Podle konvencí projektu jsou tedy plně podporovanými prohlížeči:

- Google Chrome
- Mozilla Firefox

Na těchto dvou prohlížečích musí vždy fungovat každá implementovaná vlastnost hry a uživatelského rozhraní. Jelikož Google Chrome používá vykreslovací jádro WebKit, použité i u jiných prohlížečů (například u prohlížeče Opera), je prakticky počet podporovaných prohlížečů vyšší.

Počítá se i s prohlížeči, na kterých nemusí být každá vlastnost hry zobrazená stejně jako na plně podporovaných prohlížečích, ale mělo by být možné z nich hru nějakým způsobem ovládat. Žádný z ovládacích prvků by pak neměl být zcela nedostupný. Do této skupiny prohlížečů patří například:

- Internet Explorer
- Opera
- Safari
- prohlížeče na mobilních zařízeních

Hra by v uvedených prohlížečích a zařízeních měla být ovladatelná, avšak nemusí být uživatelsky příjemná a atraktivní.

7.1.2 Podpora mobilních zařízení

Povaha hry Space Traffic od začátku projektu počítá s hraním výhradně na osobních počítačích. Předpokládá se použití většího displeje a s vyšším hardwarovým výkonem, pokud by to bylo zapotřebí. Samotný gamedesign hry příliš nepřeje kompaktnímu zobrazení na menších displejích a předchozí implementace vyžaduje relativně vysoký výkon na straně prohlížeče. Proto bylo již v začátcích této práce po poradě se správci projektu a vedoucím práce rozhodnuto, že podpora na mobilních zařízeních není vyžadována.

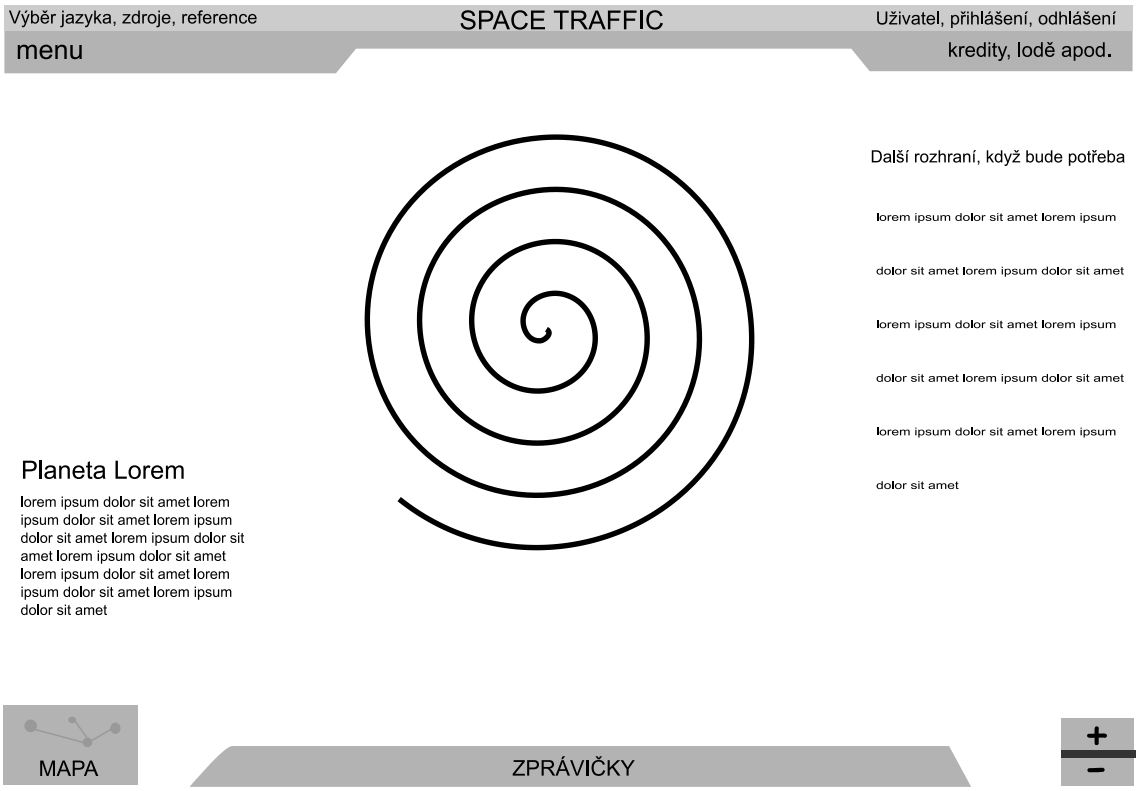
Mobilní prohlížeče a zařízení však do jisté míry umí pracovat s webovými stránkami určenými pro osobní počítače a proto je možné implementovat alespoň základní podporu, která by hráči umožňovala hru ovládat i z mobilních zařízení (například z tabletu či mobilního telefonu), byť za cenu omezení uživatelské přívětivosti. Při návrhu uživatelského rozhraní by se tedy mělo počítat s mobilními zařízeními, avšak plná podpora není vyžadována a měly by se realizovat jen části s malými požadavky na zdroje a velkým dopadem na použitelnost hry.

Součástí této práce je i návrh responzivního designu, ale z výše uvedených důvodů je vyžadována plná podpora jen pro zobrazení velikosti 1024 pixelů na šířku a 768 pixelů na výšku.

7.2 Návrh rozvržení UI

Nové uživatelské rozhraní bylo na začátku práce hrubě navrženo jako variace původního rozhraní s ohledem na zásady z kapitoly 3 a kritické hodnocení z kapitoly 6. Tento hrubý návrh vidíme na obrázku 7.1. Jedná se o návrh rozvržení prvků uživatelského rozhraní, nikoli o grafický návrh skutečného vzhledu hry. Zvýrazněny jsou části, které by měly být interaktivní a poskytovat uživateli možnost hru ovládat.

Návrh měl sloužit pro předběžné stanovení vzhledu hry, které by bylo možné dále rozvíjet a měnit. Rozvržení se během práce ukázalo jako vhodné a změnilo se jen nepatrně. Prakticky se změnila pouze spodní část, kterou nakonec vyplnily pouze informační zprávy. Tlačítko pro hvězdnou mapu z rozhraní zmizelo a přesunulo se do hlavního menu. Tlačítka pro přibližování a oddalování mapy v návrhu zůstaly, ale zobrazují se pouze na mobilních zařízeních, kde není možné k přibližování používat kolečko myši.



Obrázek 7.1: Hrubý návrh rozložení prvků uživatelského rozhraní

7.2.1 Hlavní menu

Hlavní menu se nachází v levé části. Menu je implicitně skryté a pro jeho otevření je zapotřebí na něj nejprve kliknout, protože poté zbytečně zabírá místo na obrazovce. Mělo by však zůstat otevřené či zavřené i při obnovení stránky. V hrubém návrhu se počítalo s tím, že menu samotné bude obsahovat položky, které uživatel nevyužívá příliš často a že se důležitější tlačítka a odkazy budou nacházet ve stejné části obrazovky vedle tlačítka pro otvírání menu. Během realizace však bylo zjištěno, že menu obsahuje méně položek, než bylo očekáváno. Z tohoto důvodu zůstala levá část realizována tak, jak bylo původně navrženo.

7.2.2 Osobní rozhraní

Jedná se o úzký pruh v horní části obrazovky, sloužící pro zobrazování informací o přihlášeném uživateli a jeho osobní nastavení. Také se zde nacházejí prvky a odkazy, které nesouvisejí přímo s hrou. Oproti původnímu vzhledu jsou tyto prvky uživatelského rozhraní odděleny od grafických prvků samotné hry, aby bylo jasné, že jimi nelze herní simulaci nijak ovlivnit. Také je toto umístění obvyklé u ostatních webů a tuto nepsanou konvenci je vhodné dodržet.

V levé části se nacházejí tlačítka pro volbu jazyka. Hra však v současném stavu nepodporuje lokalizaci a tato práce nezahrnuje její implementaci, takže budou tlačítka prozatím nahrazena informací o tom, že jde o českou verzi hry. Tato část by také měla obsahovat odkazy na použité zdroje, typicky licencované obrázky nebo použité

knihovny. Stejně tak by zde měla být zmínka o autorech a informace o autorských právech. Ve spodní části by tyto informace zabíraly neúměrné množství místa a proto bylo rozhodnuto, že se zobrazí až po kliknutí na odkaz, zobrazený právě v levém horním rohu.

Pravá horní část pak obsahuje informace o přihlášeném uživateli. Podle nepsaných konvencí by se vpravo mělo nacházet tlačítko pro odhlášení a případně informace o přihlášeném uživateli a odkaz na nastavení jeho uživatelského profilu. Na začátku akademického roku 2015/2016 obsahovala hra pouze prototypový přihlašovací formulář a neimplementované funkce pro registraci, odhlášení, obnovu hesla apod. Proto bylo rozhodnuto vytvořit zadání implementace práce s uživatelem. Funkcionalita a obsah této části osobního rozhraní je tím pádem v plné režii vývojáře (nebo skupiny vývojářů), kteří budou toto zadání plnit.

7.2.3 Logo

Původní stav popsaný v kapitole 6 obsahuje logo s nápisem „Space Traffic“, umístěné v levém horním rohu. Toto logo je však pro návrh nového rozhraní příliš velké. Nový návrh tak obsahuje pouze nápis „Space Traffic“, a to v řádku. Nachází se uprostřed v horní části obrazovky.

Navržené logo by mělo fungovat jako odkaz na domovskou stránku, tedy sloužit hlavně pro obnovování stránky. Absence obrázkového loga by ale mohla způsobit, že uživatelé nebudou nápis vnímat jako logo a tudíž jim nedojde, že na něj lze kliknout. Výhodou ale je úspora místa oproti původnímu rozhraní, která se jeví jako důležitější pro celkový vzhled aplikace.

7.2.4 Informační panel

Informačním panelem se myslí oblast vpravo pod přihlašováním. Měla by obsahovat informace, které jsou pro hráče klíčové a měl by je na obrazovce vidět v každém okamžiku. V současné implementaci jde o zobrazení počtu kreditů a lodí. Tyto informace by měly být viditelné a tudíž budou graficky zvýrazněny jednoznačnými ikonkami.

Počítá se s tím, že prvky zde budou především informačního rázu. Při kliknutí ale mohou zobrazovat dodatečné informace, například u počtu lodí zobrazit seznam konkrétních lodí. U takových ikonek ale musí být graficky zdůrazněno, že je možné na ně kliknout, například podtržením textu.

7.2.5 Flash zprávy

Jedná se o informační zprávy, které se zobrazují bezprostředně jako reakce aplikace na nějakou hráčovu akci, například oznamují úspěšný nákup lodi nebo mohou hráče informovat, že na zamýšlený let nemá dostatek paliva. Tyto zprávy by se měly nacházet přímo pod logem v prostoru mezi hlavním menu a informačním panelem.

Zpráva se podle současné implementace může zobrazit pouze jedna při každém požadavku, pokud budou však použity ajaxové požadavky, může se zpráv zobrazovat i více. Z toho důvodu by měla zobrazená zpráva po čase zmizet a je třeba počítat s tím, že se jich zobrazí více než jedna a budou se řadit pod sebe.

7.2.6 Mapa hvězdného systému

Uprostřed obrazovky se dle návrhu má nacházet mapa hvězdného systému, kde se hráč právě nachází. Umístění a způsob fungování této mapy je převzatý z původního uživatelského rozhraní. Vylepšení se bude týkat především vzhledu a souvisejících funkcionalit:

- přibližování mapy
- paralaxové pozadí
- vzhled planet a dalších prvků
- animace planet a dalších prvků
- zobrazení informací o planetě přímo v mapě

Umístění mapy mezi prvky uživatelského rozhraní se nijak nezmění.

7.2.7 Informace o objektech mapy

Planety a hvězdné systémy obsahují ve svých definicích popis, který se však nikde nezobrazuje. Tento popis je v hrubém návrhu zobrazen vlevo dole. Měl by být umístěný tak, aby se nad něj vešlo zobrazené hlavní menu. Má jít o prostý text, sloužící jen jako kulisa herního světa a nemá obsahovat žádné speciální grafické či ovládací prvky. Pozadí pod tímto textem je zcela či částečně průhledné (záleží na tom, zda bude pozadí pod ním narušovat jeho čitelnost).

Text se bude zobrazovat pouze po kliknutí na objekt mapy, který obsahuje popis. Jde tedy o jednotlivé planety v hvězdných systémech. Popis samotného hvězdného systému se pak zobrazí při kliknutí na hvězdu.

7.2.8 Kontextové okno

V pravé části obrazovky vidíme kontextové okno. Jeho velikost na šířku by měla být stejná jako je šířka informačního panelu. Zde by se měly zobrazovat informace a uživatelské rozhraní nezbytné pro ovládání hry. Výhodou tohoto okna je, že je možné vidět a ovládat mapu i když je otevřené. Přesto by mělo být možné ho zavřít. Klíčové ovládání hry se podle jejího gamedesignu zpravidla týká právě mapy nebo je alespoň vhodné ji při ovládání jejích prvků vidět, takže toto okno bude obsahovat například:

- detaily planety
    - cena paliva a oprav na planetě
    - dostupné zboží na planetě
    - hráčovy lodě na planetě
- detaily lodě

- náklad na lodi
- možnost natankovat nebo opravit loď

Tato část může obsahovat samozřejmě i informace, které svou velikostí a obsahem nevyžadují otevření většího středového okna (viz níže).

7.2.9 Středové okno

V návrhu není vykresleno středové okno, jelikož by překrývalo mapu. Toto středové okno by mělo zobrazovat podstatné a příliš velké informace. Otevírat by se mělo buď po kliknutí na položku menu, nebo při specifických případech kdy to vyžaduje povaha zobrazovaného obsahu. Toto okno by mělo být možné zavřít a mělo by svou velikost přizpůsobovat tomu, zda je otevřené menu nebo kontextové okno.

7.2.10 Informační lišta

Ve hře se počítá s událostmi a modelováním herní ekonomiky, což jsou informace, které hráče mohou a nemusejí zajímat. Měl by jim však být neustále vystaven, aby si mohl všimnout nějaké pro něj podstatné informace, například o tom, že planeta s kterou běžně obchoduje zdražila všechno zboží.

Pro tyto účely slouží informační lišta, která je schopná dynamicky zobrazovat důležité textové informace. Návrh počítá s tím, že se zobrazené zprávy budou cyklicky střídat, případně bude možné je zastavit nebo kliknutím zobrazit všechny.

7.3 Grafický návrh rozhraní

Grafická podoba nového návrhu byla řešena pomocí dostupných technologií během vývoje. Jednalo se zejména o technologie CSS a SVG. Bylo to z tohoto důvodu, aby se maximálně využil potenciál těchto technologií a rozhraní bylo dostatečně dynamické a rozšiřitelné pro další vývoj. Z těchto důvodů nebyl předem vytvořený grafický návrh, avšak bylo stanoveno několik technických zásad, kterým se vývoj uživatelského rozhraní měl řídit. Díky tomu nebylo nutné zdržovat se implementací grafických prvků z návrhu, které by v kódu neměly přímou podporu a naopak více využít grafické vlastnosti, které tyto technologie nabízí.

7.3.1 Barvy

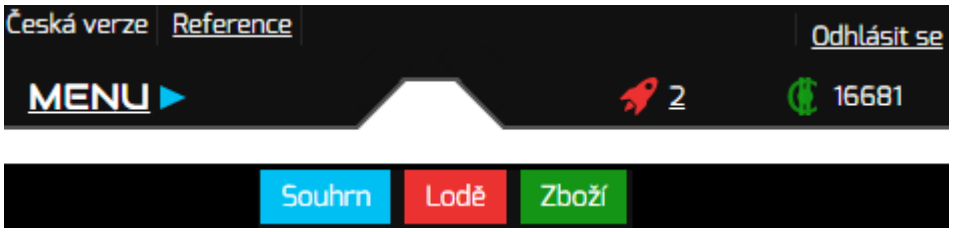
V odstavci 3.3.1 jsou uvedeny způsoby, jakými lze zvolit barvy zobrazované na stránce a celém navrhovaném webu. Při původním hrubém návrhu se uvažovaly barvy v odstínech modré s použitím schématu podobných barev. Tento motiv se však opakuje v mnoha hrách s vesmírnou tématikou a proto od něj z hlediska originality bylo upuštěno.

Při návrhu prototypu bylo rozhodnuto, že pro herní rozhraní bude použito schéma podobných barev a to v odstínech tmavě šedé až černé. Ve hře je zobrazený vesmír, který je černý a zabírá většinu obrazovky, s čímž bylo nutné počítat. Použitím téměř

černých oken s šedými okraji bylo dosaženo toho, že se kontrast na obrazovce příliš nemění a hra je tudíž na pohled příjemná. Navíc je v těchto oknech dobře čitelný bílý text.

Pro interaktivní prvky bylo vytvořeno triadické barevné schéma, které je s většinovou tmavou barvou kontrastní a tudíž výrazné. Jedná se o odstín červené, zelené a modré. Modrá barva musela být mírně zesvětlena kvůli kontrastu s černou, takže zvolené schéma není přesně triadické.

Výhodou tohoto rozložení je, že výrazné barvy použité pro uživatelské rozhraní jsou dobře viditelné. Těmto barvám také můžeme přiřadit význam, se kterým si je uživatel spojí – například zelenou můžeme používat vždy na ikony a prvky související s počtem kreditů, červenou na prvky související s loděmi a modrou používat jako neutrální. Další výhodou takto zvolených barev je jejich snadná rozlišitelnost a proto se hodí pro skupiny tlačítek. Příklad použití takto zvolených barev ukazuje obrázek 7.2.



Obrázek 7.2: Příklady použití zvolených barev pro rozhraní a tlačítka

7.3.2 Písma

Návrh obsahuje použití dvou speciálních písem. Jedno písmo bude použito pro běžný text a druhé pro nadpisy a případně popisky planet. Písmo pro webovou aplikaci musí být buď dostatečně rozšířené, aby jej obsahovaly běžně používané operační systémy, nebo lze použít sady webových písem třetích stran, které jsou dostupné z webu. V tomto případě jsou použita písma z Google Fonts [27], konkrétně následující rodiny písem:

- Exo (pro běžný text)
- Audiowide (pro nadpisy)
- Helvetica, Arial, sans-serif (jako alternativy)

Alternativní písma jsou zálohou pro případ, že by se písma z Google Fonts z nějakého důvodu nemohla zobrazit. Při výběru písem z Google Fonts pak byly brány v potaz následující požadavky podle priorit:

1. kompletní podpora českých znaků
2. rychlost načítání písem
3. vizuální přínos hře
4. podobnost s alternativními písmy

Tato písma by měla být obsažena v celé hře.

7.3.3 Grafický styl

Hra by měla dle analýzy z kapitoly 6 obsahovat jednotný styl pro grafické prvky. Jelikož je na vykreslování mapy používáno SVG, je vhodné použít tuto technologii i na běžné prvky jako jsou ikony nebo obrázky lodí. Tím se zajistí, že tyto prvky bude možné přenést na mapu a naopak, což je výhodou pro další vývoj. Zároveň je možné takto vytvořené ikony a obrázky zvětšovat a zmenšovat, případně transformovat pomocí CSS dle aktuální potřeby.

Použití této technologie částečně definuje grafický styl pro zmíněné prvky. Nelze použít rastrové obrázky (například fotografie nebo složité kresby) a prvky budou vypadat spíše schematicky.

Pro definici nových grafických prvků si stanovíme několik zásad pro kreslení a generování nových prvků. Dodržování těchto zásad zajistí jednotný grafický styl. Jde o následující zásady:

- vytvářet jednoduché SVG objekty
- použít malé množství objektů (řádově jednotky)
- používat jen několik barev (maximálně 5 různých barev)
- nevykreslovat okraje objektů

Výsledkem jsou obrázky snadno použitelné pro další vývoj, které lze navíc upravovat pouze pomocí CSS. Příklady grafických prvků vytvořených tímto postupem můžeme vidět na obrázku 7.3.

Obrázek 7.3: Příklady grafických objektů ve hře Space Traffic

7.4 Single-page aplikace

Původní návrh fungoval tak, že odkazy v menu a zavírání hlavního okna přesměrovávalo stránku. Technologie AJAX byla sice použita, avšak pouze pro účely načítání zdrojů hvězdné mapy a obnovování achievementů ve hře. To vedlo k několika vlastnostem, které byly pro uživatele nepříjemné:

- „problikávání“ stránky způsobené dobou mezi odesláním požadavku a zpracováním odpovědi
- zpomalení způsobené potřebou neustále znovu načítat mapu hvězdného systému a další prvky na obrazovce
- zavření oken otevřených uživatelem

Z těchto důvodů bylo nové rozhraní navrženo jako single-page aplikace [28], realizovaná pomocí technologie AJAX. Stránka by se neměla obnovovat celá. Na základě akcí uživatele by se měly vykonávat jednotlivé požadavky, které následně provedou hráčskou akci nebo vykreslí požadované okno s informacemi.

7.4.1 Velikost okna

Celá aplikace by se měla vejít do okna prohlížeče tak, aby nebylo zapotřebí posouvat zobrazení okna a prohlížeč nezobrazoval posuvníky. Kromě navržení grafický prvků tak, aby se vešly na jedinou obrazovku je zapotřebí přizpůsobit i velikost zobrazovaného obsahu. Dle tohoto návrhu lze počítat s výškou obrazovky 768 pixelů. Jakýkoli větší obsah musí být možné zobrazit. Jednotlivá okna na obrazovce by pro případ takového obsahu měla zobrazovat vertikální posuvníky od určité velikosti obsahu.

7.4.2 Tlačítko pro obnovení

Použití technologie AJAX a neobnovování stránky způsobí, že některá data budou ztrácet na aktuálnosti (například stav a pozice konkrétní vesmírné lodi). To lze řešit pravidelným obnovováním oken, což ale často vytěžuje serverovou část. Prodleva mezi obnovením by navíc nesměla být příliš velká a tak by mnoho požadavků bylo zcela zbytečných.

Z výše uvedených důvodů je navrženo přidat k vybraným datům tlačítko, které způsobí obnovení okna. Toto tlačítko by mělo být reprezentováno symbolem zatočené šipky a při obnovení by mělo provést viditelnou změnu, aby uživatel poznal, že obnovení proběhlo i když se zobrazení dat nezměnilo.

7.5 Rozhraní plánovače

V rámci této práce je potřeba v zájmu hratelnosti hry doimplementovat uživatelské rozhraní k plánovači cest lodí. Tato část obsahuje návrh této funkcionality.

7.5.1 Požadavky na plánování

Plánovač v aktuální verzi dokáže vytvořit souvislou cestu mezi planetami a lze naplánovat i nákup a prodej zboží. U této funkcionality se počítá s tím, že plánovač bude využíván skriptovacím jazykem pro ovládání lodí. Podle gamedesignu by mělo být možné cestu plánovat i pomocí uživatelského rozhraní, ale možnosti takového plánování by měly být omezené. Důvodem je dát hráči důvod, aby postupem času začal své lodě skriptovat a tak plnil jeden z účelů projektu – seznamoval se zábavnou formou s programováním. Plánování pomocí UI bude tedy mít následující omezení:

- najednou bude možné plánovat jen cestu z jedné planety na druhou
  - loď ale bude moci letět na planetu v jiném hvězdném systému
- nebude možné plánovat žádnou akci, například:

- nákup zboží
- prodej zboží
- tankování lodi
- opravování lodi

Takto vytvořené rozhraní bude navíc výrazně jednodušší a srozumitelnější pro hráče i pro vývojáře, zabývajícího se jeho implementací.

7.5.2 Chování UI při plánování

V detailu každé lodě bude tlačítko, kterým půjde naplánovat cesta. Při kliknutí na něj se zobrazí hlavní okno, kde budou tyto prvky:

- jméno planety, ze které loď startuje
- jméno zvolené planety, kam má loď letět
- jména případných červích děr, přes které loď poletí
- tlačítko pro spuštění plánu (start lodi)
- tlačítko pro resetování plánu (například „plánovat znova“)

Informace v hlavním okně se budou aktualizovat s tím, jak bude hráč volit planety či červí díry, kam má loď letět. Kliknutím na tlačítko pro reset se hráč vrátí do výchozího stavu, kdy začal plánovat. Kliknutím na tlačítko pro start se zavře hlavní okno a loď se vydá na svou cestu.

7.5.3 Plánovací mód

Hráč by měl pochopit, že se očekává kliknutí na planetu a uvědomit si, že kliknutí na planetu má při plánování jiný efekt. Pro to bude použit postup zmíněný v odstavci 2.1.3 o herních módech. Volba planety pro plánovač je typickým příkladem herního módu a proto je dle nastudovaných principů potřeba zdůraznit hráči, že se jeho herní relace nachází v tomto módu.

Podle analýzy je vhodným způsobem zdůraznění změna velkého prvku na obrazovce. Tím je v tomto případě zobrazení hlavního okna, které obsahuje výše zmíněné prvky. Zároveň půjde o další zmíněný způsob zviditelnění, a sice nezvyklou změnu dat. Půjde totiž o otevřené velké okno, avšak s menší velikostí, než je obvyklé. Prvky by také měly být zarovnány na střed, aby se tato nezvyklost ještě více zvýraznila.

Dalším způsobem je viditelná změna chování herního prvku. Jelikož jde ale o mód výběru planety, muselo by se měnit chování planety. Musela by se například začít otáčet jiným směrem, což by bylo pro herní simulaci matoucí. Je ale možné změnit místo chování jejich vzhled, konkrétně pak jejich popisek, který je na běžných okolností mimo mód výběru vždy viditelný. U planet, na které je možné letět, se změní barva textu, případně okraje nebo barva výplně ikon. Planetám, na které není možné letět (nemají základnu), by pak měl jejich popisek úplně zmizet. Červí díry by v tomto módu

měly mít změněný popisek stejně jako planety, na které je možné letět, avšak změněná barva by měla být jiná než u planet.

Posledním způsobem zviditelnění uvedeným v teoretické části je změna pohledu kamery. Grafika hry Space Traffic je výhradně dvojrozměrná, takže pohled nelze změnit. Z hlediska hratelnosti je ale potřeba, aby se zobrazovaný hvězdný systém změnil na ten, kde se nachází planeta, ze které loď startuje. To lze považovat za podstatnou změnu, podobnou změně pohledu kamery.

8 Implementace podpůrných nástrojů

Pro usnadnění dalšího vývoje bylo zapotřebí implementovat nástroje usnadňující práci s uživatelským rozhraním. Tato kapitola se zabývá návrhem a implementací zmíněných podpůrných nástrojů.

8.1 Ajaxová komunikace

V kapitole zabývající se návrhem je zmíněno, že by se uživatelské rozhraní mělo více orientovat na obsluhu uživatelských požadavků prostřednictvím technologie AJAX. Pro komunikaci samotnou lze použít připojenou knihovnu jQuery. Pro další vývoj je však potřeba řešit několik technických problémů:

- načítání obsahu pomocí AJAXu
- odesílání odkazů pomocí AJAXu
- odesílání formulářů a jejich dat pomocí AJAXu
- aktualizace stávajících informací zobrazených na obrazovce

Všechny tyto problémy lze řešit zvlášť pro každou vlastnost systému, čímž by ale vznikal nepřehledný kód. Pro tento účel vznikla v rámci této práce jQuery knihovna `Ajax.js`, která usnadňuje řešení zmíněných problémů. Zároveň také zapouzdřuje komunikaci pomocí AJAXu, takže v budoucnu bude jednodušší tento způsob komunikace dále rozšiřovat, například z hlediska bezpečnosti.

8.1.1 Odesílání odkazů

V one-page aplikaci často potřebujeme odeslat odkaz tak, aby při kliknutí na něj prohlížeč nepřesměroval stránku. Zároveň je žádoucí, aby odkaz samotný (atribut *href*) byl generovaný serverem a zamezilo se tak případným chybám. Právě proto je tato část implementována tak, aby stačilo html odkazu vygenerovanému serverem přiřadit třídu (v atributu *class*) ajax, čímž by se odkaz stal ajaxovým. Příklad takového odkazu můžeme vidět na obrázku 8.1.

```
<a href="url/volana/ajaxem" class="ajax" >Klikni na mě!</a>
```

Obrázek 8.1: Příklad použití ajaxového odkazu

Implementace takového odkazu je provedena tak, že za pomoci knihovny jQuery jsou vybrány všechny odkazy se třídou *ajax* na stránce a těm je poté přiřazen tzv. click event handler. Ten je definován funkcí, která se spustí při kliknutí a je jí předán objekt s informacemi o kliknutí. Nad tímto objektem je pak pomocí jQuery zavolána funkce *preventDefault()*, která zajistí, že prohlížeč při kliknutí nepřesměruje stránku. Ve funkci handleru je také možné volat objekt *this*. Pomocí něj můžeme přistoupit k HTML elementu odkazu a zjistit, jakou adresu obsahuje jeho atribut *href*. Právě takto je tato adresa ve funkci zjištěna a zavolána z prohlížeče pomocí jQuery funkce *ajax()*.

Funkci *ajax()* je předáno nastavení, obsahující v parametru adresu url, která se volá a také parametry *success* a *error*. Jedná se o callback funkce, které jQuery zavolá v případě úspěšné, respektive neúspěšné odpovědi serveru na takto vyslaný požadavek.

Callback *error* je využit na to, aby případná nastalá chyba (například když odpověď na požadavek obsahuje kód http 500) byla vypsána do konzole prohlížeče. V dřívější verzi této práce byla chyba vypisována do vyskakovacího okna, avšak působila až příliš rušivě.

Callback *success v* případě ajaxového odesílání odkazů není nutný, ale je použit pro implementaci dalších vlastností knihovny `Ajax.js` (viz níže).

8.1.2 Načítání obsahu

Aby bylo načítání nového obsahu ze serveru co nejjednodušší, byla pouze rozšířena součást na odesílání odkazů. K odkazu stačí přidat atribut *data-related-element*, jehož nastavení zajistí, že ajaxová komponenta vezme odpověď se serveru (ASP frameworku) a vloží ji do elementu s daným id. Příklad takto nastaveného odkazu vidíme na obrázku 8.2.

```
    <a href="url/volana/ajaxem" data-related-element-id="mainPanel"
class="ajax" >Klikni na mě!</a>
```

Obrázek 8.2: Příklad použití ajaxového odkazu načítajícího obsah

Takto načtený obsah se na straně serveru generuje typicky jako `PartialView`, pomocí kterého lze použít šablonovací systém ASP frameworku a vygenerovat pouze HTML kód, který má být zobrazen v daném okně.

Na vygenerované a vložené HTML je pak pomocí callbacku *success* znovu aplikováno bindování click handleru pro případ, že by obsahovalo další odkazy, které se mají zpracovávat ajaxově. Při načtení obsahu do elementu je také jeho rodičovskému elementu přidána třída *open*, pomocí které lze zajistit otevření dotyčného okna při načtení obsahu. Nad elementem, do kterého byl načten obsah je také zavolána událost *load*, která může být odchycena jinde v kódu. Takto lze případně doimplementovat další reakci javascriptu na načtení obsahu.

8.1.3 Formuláře

Pro odesílání formulářů pomocí AJAXu byl zvolen podobný způsob jako pro odesílání odkazů, ale konkrétní implementace se liší. Formuláři lze nastavit ajaxové odesílání tak, že se mu přidá rodičovský element se třídou *ajax* (viz obrázek 8.3). Alternativně stačí nastavit třídu *ajax* samotnému elementu `form`. Tento dvojí způsob existuje kvůli možnosti vygenerovat formulář pomocí ASP.

```
<div class="ajax">
    <form>
            <!-- prvky formuláře-->
    </form>
</div>
<form class="ajax">
    <!-- prvky formuláře-->
</form>
```

Obrázek 8.3: Příklady použití ajaxových formulářů

Implementace funguje podobně jako u odesílání odkazů. Na událost odeslání označeného formuláře se pověsí handler, který zavolá *preventDefault()*, čímž zastaví odesílání. Následně formulář předá externí knihovně `jQuery Form Plugin` [29]. Ta se postará o odeslání pomocí AJAXu. Formulář se na serverové části pak chová stejně, jako kdyby byl odeslán běžným způsobem.

8.1.4 Informace ze serveru – technologie

Poslední požadovanou funkcí komponenty je aktualizace informací ze serveru. Příkladem je například množství kreditů hráče. Z různých důvodů se toto množství může změnit v databázi. Je třeba zajistit, aby se během řádově několika sekund tuto informaci o změně dověděl i hráč. Bylo zváženo použití několika různých technologií, které by umožnily řešení tohoto problému:

- Traditional polling
- Long polling
- WebSocket Protocol

Traditional polling [30] je z těchto tří technologií nejméně náročný na pochopení dalšími vývojáři a implementaci. Jedná se o způsob, kdy jsou na server v pravidelném intervalu (typicky několika sekund) vysílány požadavky zjišťující stav daných dat (například právě množství hráčových kreditů). Klient si pak tato data pravidelně obnovuje.

Výhodou tohoto způsobu je jednoduchost. Nevýhodou je ale zbytečné vytěžování serveru, když se klient doptává na data, která má v tu chvíli aktuální a získaná informace pro něj není nijak přínosná. Server navíc obvykle musí kvůli získání informace využívat databázi, která je také takto vytěžována.

Při tomto způsobu je tedy nutné počítat s tím, že požadavky bude posílat více uživatelů zároveň a mohlo by dojít k zahlcení. Kvůli tomu je zapotřebí při implementaci zvažovat:

- počet takto získávaných informací
- délka časového intervalu mezi požadavky
- počet uživatelů (spuštěných prohlížečů)

Long polling [31] je variací traditional pollingu. Rozdíl spočívá v tom, že při pravidelném vysílání požadavků si server požadavek „podrží“ po určitou dobu. Odpověď na požadavek pak posílá buď po vypršení timeoutu, nebo když nastane změna nad aktualizovanými daty (například tedy až ve chvíli, kdy se změní počet kreditů). Klient pak čeká na odpověď déle a mezitím nevysílá další požadavky.

Výhodou oproti traditional pollingu je menší množství požadavků zpracovávaných serverem. Cenou za to je složitější implementace a obsluha.

Web Socket Protocol [32] je webová technologie, pomocí které klient komunikuje obousměrným komunikačním kanálem, podobně jako například u protokolu TCP/IP. Klient tak může velmi rychle dostat informaci o změnách na serveru. Je ale zapotřebí navázání komunikace.

Tento způsob je z hlediska vytížení serveru lepší než traditional polling a long polling. Pro vývojáře na projektu je ale náročnější se s ním naučit pracovat. Nezanedbatelnou výhodou je možnost posílat na klientskou část události až ve chvíli kdy nastanou, což je vzhledem k architektuře hry Space Traffic a její podpory událostí velmi žádoucí.

Tento protokol je pro prostředí .NET podporován od verze IIS 8 pro Windows Server 2012 [33].

8.1.5 Informace ze serveru – implementace

Při volbě technologie byl nejprve vyřazen Web Socket Protocol. Virtuální server projektu Space Traffic totiž používá operační systém Windows Server 2008 R2 a IIS7. Tento protokol tedy není stávající verzí serveru podporován. Aktualizace serveru a opětovná instalace všech jeho služeb by byla pro možnosti této práce časově příliš náročná a možnost použití Web Socket Protocolu by tuto časovou ztrátu kvalitativně nevynahradila.

Ze zbylých možností byl nakonec po poradě s vedoucím práce vybrán traditional polling. U informací vyžadujících obnovení v projektu Space Traffic se totiž předpokládá, že se budou měnit relativně často a potenciál long pollingu by zůstával spíše nevyužitý. Zkušenosti s projektem a jeho minulostí také ukazovaly, že při výběru technologie je lepší volit jednodušší možnost, aby s ní mohli snáze pracovat budoucí vývojáři.

Do implementované knihovny `Ajax.js` byla přidána podpora pro pravidelné požadavky na sever. Aby bylo její použití pro vývojáře co nejjednodušší, stačí zavolat jedinou funkci, které se předá jméno obslužného objektu (viz níže), data, počet sekund, po kterých se požadavky mají opakovat a callback funkci, které jsou předána vrácená data. Pokud není zadán počet sekund pro opakování, požadavek se odešle pouze jednou. Příklad zavolání funkce registrující požadavek u komponenty vidíme na obrázku 8.4.

```
$(document).ready(function () {
            ajax.send({
                    requestId: 'CreditsAmount',
                    relatedObject: 'CreditsAmount',
                    data: {},
                    repeatEvery: 5,/* každých 5 sekund */
                    callback: function (credits) {
                            $('.creditsAmount'). text(credits);
                    }
            });
});
```

Obrázek 8.4: Příklad registrace pravidelného požadavku na počet kreditů hráče

Jednotlivé žádosti, které vycházejí na stejný čas, se navíc seskupují do jediného požadavku, čímž jsou částečně kompenzovány nevýhody použitého traditional pollingu. Pro tento účel si instance knihovny ukládá jednotlivé registrované požadavky do kolekcí. Kolekce existují dvě – jedna pro pravidelné požadavky, kde požadavky zůstávají po celou dobu běhu, druhá pro jednorázové požadavky. Jednorázové požadavky jsou z kolekce vyjmuty v okamžiku zavolání jejich callbacku.

Spuštěná instance knihovny obsahuje vlastní hodiny, které kontrolují kolekci pravidelných požadavků a v případě, že je potřeba nějaké požadavky poslat, zavolá příslušnou funkci. Ta všechny aktuální požadavky sloučí do definované struktury a odešle je na server.

Při zpracování odpovědi opět rozloží data odpovědi. Poté jsou zavolány příslušné callbacky registrovaných pravidelných požadavků. V případě chyby (způsobené například špatným použitím knihovny) vypíše knihovna smysluplnou chybovou zprávu do konzole. V případě výjimky na serveru vypisuje tuto výjimku rovněž do konzole, jelikož předchozí vypisování do vyskakovacího okénka pomocí javascriptové funkce *alert()* se neosvědčilo (blokovalo prohlížeč při ukončení činnosti serveru).

Ke knihovně přísluší i controller na straně serveru nazvaný `AjaxController`. Tento controller umožňuje oddělit obsluhu požadavků od zbylých controllerů a vývojář tak nemusí řešit zadávání adres URL. Při příchozím požadavku „rozebere“ přijatou strukturu. V datech je vždy zadáno jméno obslužného objektu. Tento objekt se musí nacházet v namespace *SpaceTraffic.GameUi.Controllers.AjaxHandlers* a implementovat rozhraní `IAjaxHandleable`. Implementováním jeho metody handleRequest() lze zpracovat přijatá data a odeslat zpět libovolný objekt zpracovaných dat. Controller vytvoří objekt s příslušným názvem zadaným na klientovi a u všech požadavků vyřídí jejich obsluhu. Vývojář tak musí na serveru definovat jen jediný objekt a na klientovi zavolat jen jednu funkci. Podrobně je použití této knihovny popsáno v dokumentaci na wiki projektu Space Traffic [34].

8.2 Viewport manager

Obrazovka obsahuje menu, hlavní okno a kontextové okno a každý z těchto prvků může být viditelný nebo skrytý (neboli otevřený či zavřený). V projektu se již nacházel

javascriptový objekt, který se staral o velikost zobrazení těchto oken. Jeho název je `ViewportManager.js`. Vzhledem ke změně prvků v návrhu a možnosti jejich viditelnosti bylo samozřejmě nutné tento objekt aktualizovat. Původní implementace neobsahovala:

- podporu pro ajax
- podporu pro zavírání oken
- se skrytým menu

Původní podpora pro dvě okna, která mohou být skryta, byla rozšířena pro podporu tří oken. Zavírání oken bylo původně řešeno odkazem a přesměrováním stránky, což je pro stávající návrh nedostatečné.

Třída obsahovala metodu *doLayout()*, která se starala o překreslení oken. Vypočítávala velikost hlavního okna v závislosti na otevření zbývajících prvků po stranách. Tento výpočet byl upraven a rozšířen pro stávající vzhled aplikace. Také se začalo počítat s tím, že menu může být zavřené. Tyto změny prakticky znamenaly přepsání této metody.

Způsob zavírání oken byl kompletně předělán, vzhledem k ajaxovému použití byly tlačítka s křížky přidány do každého okna ještě před obsah, který se měnil a načítal ze serveru. Do ViewportManageru byly pak přidány metody *closeMainPanel()* a *closeContextPanel()*. Také do jeho metody *init()* přibyly obsluhy kliknutí na tato zavírací tlačítka. Tyto obsluhy u příslušného okna upravují jeho HTML atribut *class* a samotné zobrazení či skrytí okna je ošetřeno pomocí CSS. Po těchto úpravách bylo v dalším kódu možné volat překreslování velikostí oken a jejich zavírání. Změnou třídy `ViewportManager.js` a použitím jejích nových vlastností bylo dosaženo možnosti dynamických změn velikostí oken.

8.3 Flash zprávy

Podpora pro flash zprávy (zprávy oznamující uživateli například informaci o tom, že jeho akce byla úspěšná) byla sice v uživatelském rozhraní implementována, ale počítalo se s obnovením stránky. Příslušná zpráva se vždy uložila do cookies, odkud byla při přesměrování vyňata a zobrazena na obrazovce. Taková zpráva pak zmizela opět při dalším přesměrování. V ajaxové aplikaci není možné čekat na přesměrování, tudíž tato funkcionalita musela být změněna. Také bylo zapotřebí zamezit hromadění zpráv na obrazovce.

Existující skript tak byl rozšířen o funkci *renderFlashMessage()*, která prováděla zobrazení zprávy. Také bylo v již použité komponentě třetí strany nastaveno, aby zpráva postupně zmizela během 8 sekund. Tato funkce pak byla zavolána na různých místech aplikace, aby se zajistil správný výpis zpráv.

8.4 Aktualizace jQuery

Knihovna jQuery je pro správný chod stávající i dalších verzí projektu nezbytná. Podle analýzy této práce je třeba stávající verzi aktualizovat. I při vývoji samotném byly překážkou chybějící funkce novějších verzí, například metoda *on()* umožňující bindovat obslužné handlery pro libovolné události. Problémem je přechod přes verzi 1.9, kdy z jQuery ubylo mnoho zásadních funkcionalit, které však mohly být ve starší verzi hry používány.

Při pokusu aktualizovat na verzi 2.2 výpis z konzole prohlížeče skutečně oznamoval chybějící funkce nad různými objekty. Byl proveden pokus tuto funkcionalitu lokalizovat v kódu a nahradit ji alternativní verzí, avšak vzhledem k chybějící dokumentaci této funkcionality to nebylo možné.

Pro účely migrace ale existuje plugin *jquery-migrate* [35], který doplňuje podporu pro staré a vyřazené funkce. Aplikace tohoto pluginu vyřešila problém aktualizace a nebyl zaznamenán žádný problém s výkonem. Aktualizace na verzi jQuery 2.2 tedy proběhla úspěšně a pro další vývoj je možné využívat novější funkce této knihovny.

9 Implementace grafického vzhledu mapy

Nový návrh uživatelského rozhraní předkládá nutnost zcela změnit grafickou podobu a chování mapy hvězdného systému, na které jsou zobrazeny planety a červí díry obíhající hvězdu. Tato kapitola se zabývá implementací změn navržených v odstavci 7.2.6.

9.1 Přibližování a oddalování mapy

Formát SVG a jeho vlastnosti jsou ideální pro přidání možnosti přibližování. Zvětšením kresba neztratí na kvalitě. Měnit přiblížení by se také mělo v závislosti na pozici kurzoru, aby bylo dosaženo podobného chování jako například u map. Jedná se o relativně jednoduchou úlohu na transformaci souřadnic, avšak bylo zapotřebí přizpůsobit ji aktuální implementaci, která umožňuje pohyb mapou do stran. Byla hledána existující jednoduchá řešení, avšak při jejich použití se obvykle ukázalo, že jsou pro uživatelsky přívětivé chování hry nedostačující.

9.1.1 Mousewheel plugin

Prvním problémem bylo odchytit pomocí javascriptu událost točení kolečkem myši. Stávající verze jQuery neobsahovala podporu pro tuto událost. Proto byl použit plugin *jquery-mousewheel* [36], který zajišťuje odchytávání kolečka myši na běžných prohlížečích.

Kromě odchycení eventu také funkci handleru předává parametr *delta*, označující, jak moc bylo s kolečkem otočeno. Také je možné podle něj zjistit, zda jím bylo otočeno dopředu nebo dozadu.

9.1.2 Transformace mapy

Mapou jde v původní verzi hýbat do stran. Tuto funkcionalitu zajišťuje třída `SvgViewportManager`. Nejprve si vždy vypočítá souřadnice mapy z událostí pohybu myší a ty pak použije pro transformaci mapy pomocí SVG atributu *transform*. Konkrétně použije jeho část *translate(x, y)*, která nastavuje pozici v 2D souřadnicích.

Tato translace byla následně upravena, aby reflektovala nový parametr třídy *zoom*. Obě souřadnice byly pro translaci vyděleny velikostí *zoom*, aby se mapa přibližovala podle svého středu a „neujížděla“ do strany. Dále je k atributu *transform* přidána součást *scale()*, která definuje škálování (zvětšení nebo zmenšení) kresby. Její velikost je určena aktuálním přiblížením.

Přiblížení samotné se pak ovlivňuje točením kolečkem a velikostí parametru *delta*. Také je definována konstanta *MOUSEWHEEL_ZOOM_SPEED*, která určuje míru přiblížení. Při otočení kolečkem vpřed (nahoru) se aktuální hodnota přiblížení touto konstantou násobí, při otočení vzad (dolů) se pak stejnou konstantou dělí.

$$zoom_{in} = zoom_{current} * MOUSEWHEEL\_ZOOM\_SPEED$$

$$zoom_{out} = \frac{zoom_{current}}{MOUSEWHEEL\_ZOOM\_SPEED}$$

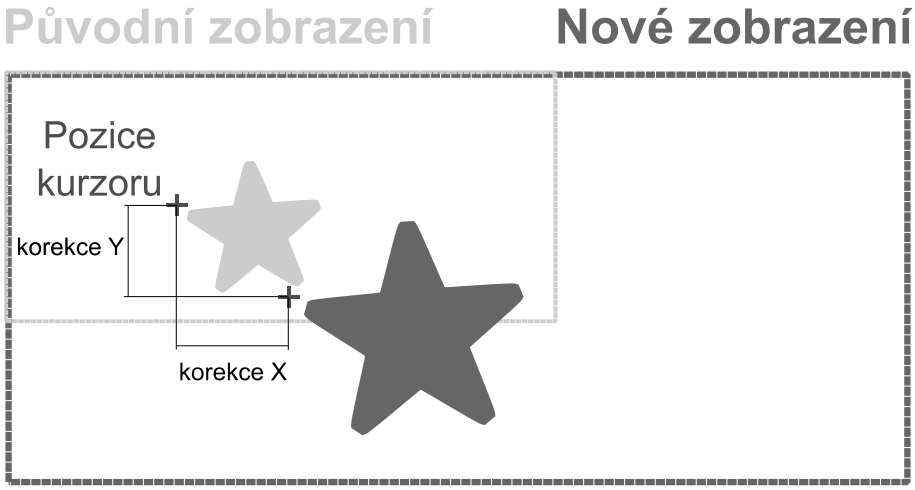
kde $zoom_{current}$ je aktuální hodnota přiblížení (přičemž hodnota 1 je stav bez úpravy přiblížení). $zoom_{in}$ a $zoom_{out}$ jsou nové hodnoty *zoom* po aplikaci přiblížení nebo oddálení. Takto je dosaženo přibližování, kdy se při každém pohybu kolečkem mapa zvětší o relativně stejnou velikost.

9.1.3 Přiblížení na pozici kurzoru

Mapa se nyní přibližuje vzhledem ke svému středu, tedy na souřadnice [0, 0]. Pro lepší hráčský zážitek by se ale měla přibližovat k aktuální pozici kurzoru. To vyžaduje úpravu pozice mapy v závislosti na pozici kurzoru a míře přiblížení. Prakticky jde o to nastavit souřadnice tak, aby na pozici kurzoru byla odpovídající souřadnice, na kterou už ale bylo aplikováno zvětšení. Tento problém je naznačený na obrázku 9.1. Řešení navíc muselo být kompatibilní se stávající implementací pohybování mapou. Použití samotné transformace a transformačních matic tudíž vedlo k nepřesnostem při používání rozhraní. K implementaci byl tak použit následující vztah, určující korekci souřadnice X, respektive Y.

$$korekce = X - (X * (1 + zoom_{změna}))$$

Kde X je původní hodnota souřadnice X nebo Y a $zoom_{změna}$ je absolutní hodnota rozdílu mezi původním přiblížením a novým přiblížením. Vypočítaná korekce pak byla přičtena k původní souřadnici v případě přiblížení a odečtena v případě oddálení. Předpokládáme, že souřadnice jsou vztaženy přímo k SVG kresbě a tudíž nezávislé na aktuálním zvětšení. V reálné implementaci bylo nutné tyto souřadnice dopočítat.



Obrázek 9.1: Problém korekce pozice kresby po přiblížení

Tato korekce byla aplikována na souřadnice X i Y, a to na každou zvlášť. Díky tomu je mapa při každém přiblížení či oddálení „srovnána“ tak, aby souřadnice,

na kterou ukazoval kurzor, se nacházela pod kurzorem i po změně přiblížení. Zároveň tento způsob neovlivňuje implementaci posouvání mapy po souřadnicích. Implementace vyžadovala velké množství času pro odladění této funkcionality.

9.2 Vykreslování objektů mapy

Vzhled jednoduchých objektů byl řešen pouze pomocí CSS. Šlo například o dráhy planet a červích děr. Složitější objekty ale vyžadovaly generování jiného SVG kódu. Stávající implementace generovala typicky pouze kruhy či jednoduché schematické obrázky. Bylo zapotřebí zasáhnou i do definic SVG a definic stylů stránky kvůli animacím. Javascriptové objekty, starající se o vykreslování prvků mapy byly rozšířeny tak, aby mohly tyto definice ovlivňovat a generovat do nich kód. Pro jednotlivé prvky mapy pak byl zvolen různý způsob, jak vylepšit jejich vzhled.

9.2.1 Datový model objektů

Informace o planetách v jednotlivých hvězdných systémech jsou uloženy v souboru XML, kde jsou definovány statické parametry planet. Pomocí již existujícího kódu jsou pak předávány jednotlivým modelům, například modelu `Planet.js`. Během této práce bylo přidáno několik dalších potřebných parametrů. Jednalo se například o dodatečné informace o vzhledu planet, jejich herní popis a informaci o tom, zda je na planetě základna. Muselo být také upraveno příslušné schéma, používané pro validaci XML při startu herní simulace.

Tato data pak byla používána k vykreslování a chování jednotlivých prvků mapy. Například při kliknutí na planetu bylo možné zobrazit popis planety v levé dolní části obrazovky.

9.2.2 Popisky objektů

Stávající implementace obsahovala kód, který generoval vzhled popisků planet a červích děr. Jde o třídu `SvgNameplate.js`. Pro změnu vzhledu stačilo změnit generovaný SVG kód.

Generované popisky ale musely počítat s velikostí textu, které se měla přizpůsobovat ji jejich velikost. Bylo nutné vyřešit, jak zjistit šířku psaného textu, která může být různá. Toho bylo dosaženo tak, že text byl nejprve vygenerován do stránky, kde byl skrytý pomocí CSS. Z vygenerovaného elementu byla zjištěna šířka, která byla posléze použita pro generování popisku. Dalším problémem byla aplikace Google Fontu, který měl rozdílnou velikost než běžné písmo a nebylo možné takový text „změřit“, neboť se obvykle načetl se zpožděním. Tento problém byl vyřešen nalezením podobně širokého standardního písma, které bylo použito pro skrytý text. Důsledkem toho byla potřebná šířka určena jen přibližně, ale s dostatečnou přesností.

U každého popisku, který náleží planetě, by také měla být značka, zda se jedná o planetu se základnou (tedy planetu, kde je možné nakupovat a prodávat). Podle

modelu `Planet.js` se vykreslovací objekt `SvgNameplate.js` nyní rozhoduje, zda a jak toto rozhraní vykreslit.

9.2.3 Červí díry

Červí díry byly předělány tak, že se generuje několik spirálovitých křivek. Vzhled těchto křivek je definován pomocí CSS. Pomocí CSS je také definována animace křivek, které se otáčejí podle společného středu. Rychlost tohoto otáčení je pro křivky různá, což vytváří spolu s jejich průsvitností zajímavý vířivý efekt. Je také aplikováno mírné rozmazání pomocí SVG.

Chování červích děr při události *hover* (najetí kurzorem) muselo být upraveno, jelikož v původní verzi byl jejich SVG objekt „přenesen na popředí“, čímž se ale resetovala animace. Současné červí díry tedy nemění svou pozici ve vrstvách mapy, ale pouze změní barvu.

9.2.4 Hvězdy

Hvězdy jsou v XML definovány pomocí tří barev. Hlavní z nich tvoří pozadí hvězdy. Dále je pro hvězdu vytvořena SVG definice, která obsahuje vzorek (SVG element `<pattern>`) použitý pro její pozadí. Pomocí tohoto vzorku je možné aplikovat na výplň hvězdy animace, které zůstanou oříznuté. Vzorek je vlastně čtverec, který je v pozadí opakován. Hvězda může mít tím pádem libovolnou velikost a počet objektů ve vzorku nemusí být příliš vysoký.

Vlastní animace hvězdy je implementována tak, že ve vzorku je definováno několik kruhů a elips různé velikosti a vzhledu. Tyto kruhy a elipsy jsou pak pomocí animace postupně zvětšovány a zprůhledňovány. Jejich barvy pochází z definic daného slunce (jsou aplikovány dvě, jedna u většího množství objektů). Animace kruhů a elips mají také různou dobu trvání a jsou různě zpožděné. Celý vzorek je pak natočený. Výsledný efekt pak připomíná náhodné erupce.

Dále jsou na hvězdy aplikovány CSS vlastnosti, které upravují jejich vzhled, například svit hvězdy je vytvořen pomocí tlustého průsvitného okraje.

9.2.5 Planety

V případě planet se rozlišuje, zda se jedná o planetu pevnou nebo plynnou. Podobně jako u hvězdy jsou pro každou planetu v XML definovány tři barvy. Jedna opět tvoří pozadí. Na animaci planety je rovněž použita definice se vzorkem jako v případě hvězdy, avšak v tomto případě je velikost vzorku větší než samotná planeta. Jeho výška je stejná, ale šířka je dvojnásobná. Takto je možné simulovat pomocí 2D grafiky otáčení planety, protože objekty ve vzorku, které se nacházejí mimo výřez kruhu, jsou „na opačné straně planety“.

Ve vzorku se nacházejí objekty, v případě pevných planet kruhy, v případě plynných elipsy. Ty jsou podobně jako v případě hvězdy animovány, ale nemění se

jejich velikost ani průhlednost, ale pozice. Všechny objekty se pohybují zprava doleva. V případě pevných planet se pohybují všechny objekty stejnou rychlostí, čímž se vytváří dojem, jako kdyby se planeta otáčela. V případě plynných mají objekty v animaci rychlost mírně různou, takže vzniká dojem, že planeta během otáčení víří.

U planety je také možné definovat v XML kromě jejích barev i rychlost otáčení, kterou je ovlivněna rychlost animace objektů ve vzorku. Objekty animace také mají různou velikost a pozici. Kromě toho také planetě lze přidat prstenec, čímž se vygenerují příslušné křivky. Prstenec je složený ze dvou částí kvůli překrytí planety.

9.2.6 Různorodost hvězd a planet

V XML lze definovat nejen barvy, ale částečně i vzhled animace – zejména velikost a pozice pohybujících se objektů. Objekty vykreslující planety využívají náhodný generátor, s nímž zanáší do parametrů animace určitou míru náhody. Planety mají díky tomu různé animace a vypadají, že je jejich povrch rozdílný. Hvězdy pak „bublají“ viditelně jiným způsobem.

Jednotlivé planety i hvězdy by ale měly být pokaždé stejné. Na generátor byl tedy aplikován seed, který zajistí, že generátor vygeneruje pro každou planetu stejné hodnoty. Tento seed lze definovat právě v XML u každé planety. V případě javascriptové funkce *Math.random()* ale seed použít nejde, protože tato funkce používá pro generování náhody mimo jiné i systémový čas. Z toho důvodu byla připojena knihovna třetí strany [37], poskytující funkci *Math.seedrandom()*, která tuto vlastnost vypne a přidá možnost zadat seed. Kvalita náhodné posloupnosti je snížena, ale pro generování grafického vzhledu to není prioritní. Vlastnost seedování je navíc po použití vypnuta.

9.2.7 Paralaxové pozadí

Původní implementace používala rastrový obrázek, který se opakoval po obou osách na celé stránce. Nová implementace generuje SVG vzorek, který se rovněž opakuje po celé stránce. Za pomoci náhodného generátoru se seedem (viz výše) je do vzorku vygenerován určitý počet (řádově desítky) hvězd v podobě SVG objektů. Takto jsou vygenerovány tři vrstvy s různou velikostí hvězd, které se překrývají a reprezentují „bližší“, „vzdálenější“ a „nejvzdálenější“ hvězdy.

Dvě vrstvy „bližších“ hvězd pak reagují na uživatele stejně jako hvězdná mapa, jen v menší míře. Obě vrstvy se přesouvají a přibližují, ale jejich konstanty jsou menší než u mapy (a konstanty „vzdálenější“ vrstvy jsou nižší než u „bližší“ vrstvy). Vzniká tak paralax efekt popsaný v odstavci 6.3.2.

Pro implementaci této funkcionality byla upravena třída `SvgViewportManager` podobně jako v případě přibližování a oddalování. Pro každou vrstvu a mapu se transformace počítají zvlášť.

9.3 Podpora v prohlížečích

Výsledná implementace mapy byla podrobena zkoumání na různých prohlížečích, zejména těch, jejichž plnou podporu vyžadují konvence projektu Space Traffic. Následně byla podle výsledků poupravena implementace.

9.3.1 Dotyková zařízení

Na dotykových zařízeních nebylo možné hýbat s mapou, protože stará implementace nepodporovala reakci na dotyk. Také nebylo možné mapu přibližovat, protože na dotykových zařízeních pochopitelně není kolečko na myši.

Pro tento účel byla hledána knihovna, která by podporu zajistila. Tu se ale najít nepodařilo. Byl tedy vytvořen skript `TouchDeviceSupport.js`, který reaguje na dotykové události na mapě a pomocí nich umožňuje hýbat mapou na dotykových zařízeních. To se děje pouze, je-li detekováno dotykové zařízení. Skript také v případě použití dotykového zařízení přidává na obrazovku pod mapu dvě tlačítka, umožňující přibližování a oddalování mapy.

9.3.2 Animace

Při ladění animací na prohlížečích Google Chrome a Mozilla Firefox byla zjištěna nepříjemná nekompatibilita. SVG animace byly nejprve animovány výhradně pomocí CSS. Ukázalo se ale, že Firefox nepodporuje css animace, kde se mění pozice či jiné atributy objektů, což je pro animace mapy nepostradatelná vlastnost. Animace tedy byly přepsány, aby používaly starší SVG technologii SMIL. Takto implementované animace fungovaly na obou prohlížečích, avšak Google Chrome upozorňoval, že SMIL animace jsou již zastaralé a v jeho další verzi nebudou podporovány.

Tento problém byl vyřešen použitím obou technologií. Skripty generující animace nyní detekují, na jakém typu prohlížeče jsou spouštěny a podle toho vyberou technologii, která je použita pro animace.

9.3.3 Výkon

Klientská část aplikace vyžaduje pro hladký běh poměrně velký výkon. Ukázalo se, že většinu tohoto výkonu vyžaduje předchozí implementace, pravděpodobně animace pohybu a dynamické výpočty rychlostí planet obíhajících hvězdu, což je bohužel nezdokumentovaná implementace, kterou je velmi obtížné a nežádoucí měnit. Animace také využívají nezanedbatelné množství výkonu, ale ukázalo se, že prohlížeče na mobilních zařízeních je samy vypínají. Při vývoji byl kladen důraz na omezení počtu vykreslovaných objektů a složitosti animací. Navíc jsou animace mapy vytvořené výhradně v SVG a CSS a nevyužívají žádné skripty. Tím pádem jejich vykreslení implementuje přímo jádro prohlížeče, díky čemuž je jejich vykreslování rychlejší než při použití skriptů. Stávající klientskou implementaci tak není možné více urychlovat bez ztráty důležitých vlastností.

10 Implementace uživatelského rozhraní

Podle návrhu uvedeného v kapitole 7 a analýzy v kapitole 6 je na projektu zapotřebí změnit stávající uživatelské rozhraní a přidat ho k funkcionalitám, kterým uživatelské rozhraní chybí. Tato kapitola se zabývá implementací těchto změn a nových rozhraní.

10.1 Celkový vzhled

Vzhled aplikace byl měněn pomocí technologií CSS3 a LESS. Kaskádové styly existující aplikace byly upraveny a zpřehledněny tak, aby bylo možné je dále upravovat. K definicím vzhledu byla použita taková pravidla, aby je podporoval co nejvyšší počet prohlížečů, zejména pak ty, které jsou stanoveny v konvencích projektu.

Kvůli rozsahu není možné uveden konkrétní postup, jakým bylo rozhraní vytvořeno. Rozhraní je celé vytvořené pomocí HTML, CSS a SVG, vyjma některých ikon a obrázků pro achievementy. Snahou bylo co nejvíce zapojit vektorovou grafiku. Výsledný celkový vzhled lze najít v příloze C.

10.1.1 Barvy

Použití barev bylo sjednoceno a všechny jsou uvedeny v LESS proměnných v souboru `colors.less`. Barevné rozložení v celé aplikaci tak lze snadno změnit v jediném souboru. Byly použity tři výrazné barvy (červená, zelená, světle modrá) pro tlačítka a uživatelské prvky a odstíny šedé až černé pro uživatelské rozhraní.

10.1.2 Písma

Podle návrhu byla vybrána dvě písma z databáze Google Fonts. Ta byla přilinkována přímo do layoutu hry. Pro CSS pravidla těchto písem (zejména *font-family*) byl definován soubor `fonts.less`. Tak lze opět jako v případě barev změnit typ písma globálně v celém projektu.

10.1.3 Ikony

Pro ikony v projektu, zejména pro značky představující kredity a lodě, byla vytvořena css podpora v souboru `icons.less`. Díky tomu lze ikonu do rozhraní přidat pouze za použití CSS třídy. Přidáním dalších css tříd je navíc možné ovlivňovat její velikost. Příklad je vyobrazen na obrázku.

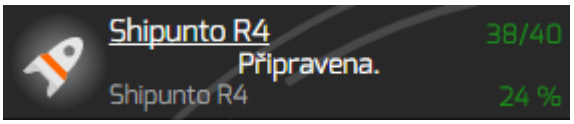
Použité obrázky ikon jsou vektorové, takže je možné jejich velikost libovolně měnit podle potřeby.

10.1.4 HTML části

Pro často opakované části rozhraní, jako jsou například informace o lodi nebo o zboží byly vytvořeny části HTML v projektu pojmenované jako *cards*. Jedná se o samostatné HTML šablony s již hotovými CSS a navrženým zobrazením. Jejich výhodou je, že je lze použít v kterékoli jiné šabloně. Chceme-li například vykreslit základní detail lodi, stačí do naší šablony přidat kód:

```
    @Html.Partial("ShipCards/_ShipCardBasic", new ViewDataDictionary {
{ "ship", ship }, { "index", i } })
```

kde pouze správně nastavíme relativní cestu k souboru karty a předáme potřebné proměnné. Vykreslí se detail lodi včetně vzhledu, jako je na obrázku 10.1. Těchto karet je vytvořených více pro větší či menší množství zobrazovaných informací. Kromě lodí existují také pro zobrazování zboží.



Obrázek 10.1: Příklad vykreslení „karty“ lodi

10.2 Hlavní prvky stránky

Při implementaci navržených prvků na stránce bylo využíváno podpůrné implementace, která je popsána v kapitole 8. Rozložení prvků bylo definováno v CSS. Obrazovka je nyní rozdělena na tři části. Poměr těchto stran lze změnit v několika LESS proměnných. Rozhraní je responzivní a reaguje na změnu velikosti okna. V rámci této práce byl pečlivě implementován i detailní vzhled pomocí CSS, vzhledem k rozsahu a srozumitelnosti kódu však nebude tato implementace popsána detailněji.

10.2.1 Hlavní menu

Z hlavního menu byly odstraněny odkazy na neimplementované funkcionality, čímž se menu podstatně zmenšilo. Bylo také skryto za tlačítko, ovládané pomocí javascriptu. Ten při kliknutí mění CSS třídy a styly příslušného okna a menu se tak zobrazuje či skrývá. Informace o otevření či zavření menu se také ukládá do cookie prohlížeče pomocí již připojené knihovny. Tato informace se čte při každém obnovení okna, takže menu zůstává ve stavu, který si uživatel zvolil.

Stávající prvky pak byly převedeny na ajaxovou verzi aplikace za použití podpůrných nástrojů. Pro ten účel bylo nutné upravit existující controllery, především pak rodičovský `TabsControllerBase.cs`, který umožňuje mít v jednom controlleru více vnořených záložek. Konkrétní chování těchto záložek bylo poupraveno na straně prohlížeče pomocí jQuery. Díky tomu se existující controllery dají použít stejně jako v předchozí verzi.

10.2.2 Informační panel

V informačním panelu byla využita možnost vytvoření opakovaného dotazu na server (viz kapitola 8), čímž se pravidelně aktualizují informace o počtu lodí hráče a počtu kreditů. Na klientské části byly k tomu účelu vytvořeny skripty `Credits.js` a `ShipsInterfaces.js`. Na serverové části potom objekty `CreditsAmount.cs` a `ShipsInfo.cs`. Pro vyšší přehlednost pak skript načítající počet kreditů zjišťuje, zda jde o zvýšení či snížení jejich počtu a podle toho je na okamžik obarví buď červeně nebo zeleně. Důvodem je, že u většího čísla hráč nemusí být schopen na první pohled rozlišit, zda jde o přírůstek nebo úbytek.

U výše zmíněných údajů jsou pak využité podpůrné CSS pro ikony. Pomocí nich je vedle každé informace příslušná ikona. V případě lodí je pak možné na údaj o jejich počtu kliknout. Tato možnost je zdůrazněna podtržením textu. Při kliknutí se do kontextového okna načte seznam lodí, kde je použita HTML část popsaná v odstavci 10.1.4.

10.2.3 Informační lišta

Spodní lišta obsahující informace je prozatím naplněna placeholdery. Pro její implementaci byl použit existující plugin třetí strany – marquee jQuery plugin [38]. Pluginů pro tuto funkcionalitu existuje více a byly vyzkoušeny i jiné možnosti. Tento plugin však umožňuje dynamické vkládání obsahu za běhu, které je pro přidávání dalších zpráv nezbytné. Pomocí CSS pak bylo docíleno správného chování tohoto pluginu. Vytvoření konkrétních zpráv informujících o dění ve hře nebylo součástí této práce, nicméně je takto zobrazování zpráv technicky umožněno.

10.3 Obsah a ovládání hry

Tato část se zabývá implementací zobrazení obsahu, ovládání a dalších možností hráče. Zde uvedená implementace byla vytvořena zcela nově, jelikož původní implementace tyto funkcionality vůbec neobsahovala.

10.3.1 Chování oken

Implementace základního chování oken je již popsána v kapitole 8, pokud ale bylo v rámci aplikace vyžadováno specifičtější chování (například vlastnost, že se okno se zbožím po nákupu zavře), bylo implementováno pomocí javascriptu a jQuery. Typicky se tento javascript přidával do stránky spolu s HTML, které se načítalo pomocí AJAXu. Tyto skripty často simulovaly chování uživatele, například kliknutí na tlačítko pro zavření okna za určitých podmínek.

V rámci jednotlivých oken se také řešila možnost obnovení okna. Tlačítka pro obnovení jsou vlastně obyčejné ajaxové odkazy, které načítají stejný obsah, jaký je již na obrazovce. Aby bylo poznat, že se obnovení podařilo, na tlačítko je aplikována CSS

animace rotace. Jelikož se při úspěšném obnovení obsahu znovu načte i samotné tlačítko, prohlížeč tuto animaci zopakuje při každém načtení.

10.3.2 Problém s referencemi

Aplikace používá Entity Framework pro načítání prvků z databáze. Ten je schopen načítat k entitám z databáze i další entity přes relace. Povaha některých objektů je ale taková, že zde vzniká kruhová závislost. Samotný Entity Framework dokáže tuto kruhovou závislost ošetřit, problém ale vzniká při pokusu taková data serializovat a poslat přes síť ASP frameworku. V takovém případě komunikace selže bez smysluplného chybového hlášení. Při použití debuggeru Visual Studia také nebylo rozpoznatelné, co za selháním stojí. Zmíněná chyba byla odhalena až pozorováním a zkoumáním rozdílů při jednotlivých případech. Při vývoji způsobil tento problém ztrátu velkého množství času spotřebovaného při pokusech chybu odstranit.

Řešením je odstranit z entit reference, které způsobují kruhovou závislost. Toto odstranění nemá vliv na předanou informaci, protože se odstraňují až reference na entity, které již známe. Podrobnější informace o tomto problému byly autorem této práce připsány na wiki projektu [39].

10.3.3 Základny

Seznam základen lze zobrazit kliknutím na položku v menu. Otevře se hlavní okno, kde jsou vyobrazeny všechny planety, které mají základnu. Pomocí služeb poskytovaných projektem `GameServer` se z databáze načtou všechny základny a zobrazí se v seznamu. U každé základny je vektorová ikona, která je stejná jako označení základny v mapě hvězdného systému. Hráč tak lépe pochopí význam ikony.

Při kliknutí na některou ze základen je zobrazen detail základny (viz níže). Do objektů zajišťujících zobrazení mapy byla přidána funkce, která umožňuje změnit aktuálně zobrazený hvězdný systém. Poté bylo možné přepnout hráče na hvězdný systém, kde se základna nachází.

10.3.4 Lodě

Data o lodích jsou získávána obdobně jako u seznamu základen. Každá loď má navíc svůj obrázek. Tento obrázek je ve formátu SVG. Je načítán nikoli jako obrázek, ale pomocí javascriptu vygenerován přímo do stránky. Výhodou tohoto postupu je, že na obrázek je možné aplikovat CSS styly a měnit například barvy lodě.

Seznam lodí je vykreslený pomocí HTML částí. Zobrazeny jsou důležité informace jako je aktuální činnost lodi, její jméno, typ, množství paliva a míra poškození lodi. Informace o aktuálním stavu lodi byla přidána do modelu na serveru a je aktualizována v databázi při událostech, které mění stav lodi. Při kliknutí na loď se v kontextovém okně otevře její detail a uživatelské rozhraní.

Seznam lodí je možné vygenerovat do hlavního okna (což se stane při kliknutí do menu) nebo do kontextového okna. Zobrazení seznamu lodí v kontextovém okně vidíme na obrázku 10.2.

Obrázek 10.2: Seznam vlastněných lodí v kontextovém okně

10.3.5 Detail základny

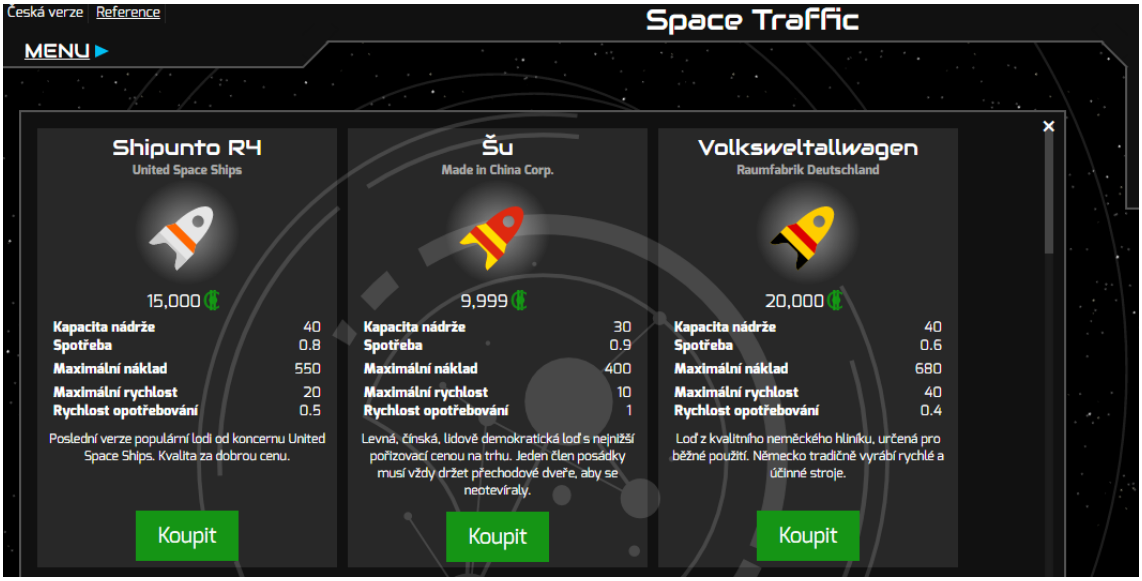
Detail základny lze spustit buď kliknutím na planetu se základnou, nebo na základnu v seznamu základen. Detail se pak načte pomocí AJAXu do kontextového okna. ASP získá z databáze data o základně a zpracuje je. K tlačítkům je pak přiřazen javascript, který zaznamenává kliknutí na ně do proměnné. Tato proměnná se čte při kliknutí na jinou základnu a podle ní se otevře příslušná záložka, aby uživatel při zkoumání například zboží na planetách nemusel neustále při přepínání planet volit záložku „Zboží“.

Detail základny obsahuje tři možnosti – souhrn, lodě a základny. V souhrnu je zobrazena cena oprav a paliva. Tyto údaje jsou načteny z databáze se samotným obchodníkem (každá planeta má v současnosti maximálně jednoho obchodníka, takže se obchodník načítá podle jména planety). Souhrn je také určen k případnému přidávání dalších informací o základně na planetě. Položka zboží zobrazuje dostupné zboží na planetě. To se načte spolu s obchodníkem pomocí Entity frameworku a zobrazí pomocí HTML „karty“. Poslední položkou ve vnořeném menu jsou lodě. Ty zobrazují všechny hráčovy lodě na planetě obdobným způsobem jako u zboží. Také se zde nachází tlačítko pro koupi lodi na této planetě.

10.3.6 Nákup lodi

Nákupy lodí byly realizovány v rámci jiné práce, ale uživatelské rozhraní této vlastnosti nezapadalo do aktuálního návrhu. Bylo tedy v rámci této práce předěláno. Také byl změněn způsob, jakým se vykresluje obrázek lodi. Nyní se loď vykresluje tak, že se do stránky načte SVG obrázek pomocí jQuery. Ten je pak možné ovlivnit

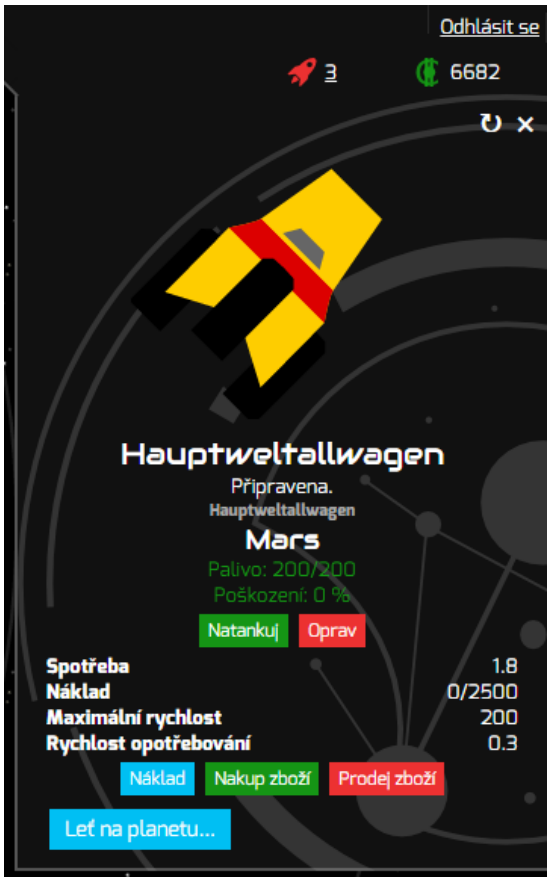
přidanou CSS třídou. Do XML definujícího lodě tak byl přidán parametr třídy. Nově se také při koupi lodě ukládá tato třída a jméno obrázku do databáze. Tak je možné vykreslit loď opakovaně (což je využito v HTML částech pro vykreslení lodi) a měnit její vzhled pomocí CSS. Tento způsob také do budoucna umožňuje rozšíření, kdy si hráč může měnit vzhled své lodi. Rozhraní pro nákup lodí a použití stejných zdrojových kreseb pro vykreslení různých lodí vidíme na obrázku 10.3.



Obrázek 10.3: Rozhraní pro nákup lodí

10.3.7 Detail lodi

Okno s detailem lodi a jejími možnostmi je pro hraní hry velmi důležité, takže bylo vhodné ho vzhledově odlišit od ostatních. Proto je zde obrázek lodi větší a informace o lodi se zarovnávají na střed. Zobrazení detailu lodi ukazuje obrázek 10.4. Obrázek lodi je vektorový, což umožnilo zvětšení nad obvyklou velikost. Tlačítka pro ovládání lodi (tankování, opravování, náklad, nákup a prodej zboží a tlačítko pro plánování trasy) nejsou kvůli přehlednosti seřazena v menu jako v případě základen, ale nacházejí se v místě informací, které přímo ovlivňují (například tlačítko pro tankování je poblíž informace o stavu paliva). Všechna tato tlačítka využívají podporu implementované knihovny pro AJAX.

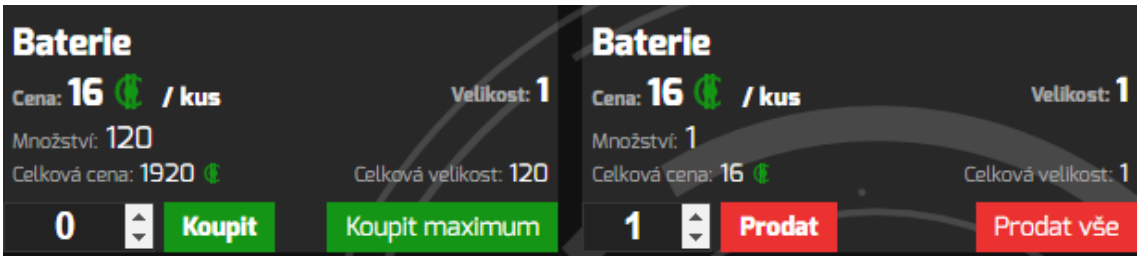


Obrázek 10.4: Detail lodi v kontextovém okně

Při načítání obsahu bylo také řešeno ošetření přístupu k lodi, tedy kontrola, zda hráči loď patří, zda není právě zaneprázdněna nějakou činností a podobně. Pro tento účel vzniklo v modelu několik parametrů, například parametru *IsAvailable*, který signalizoval, jestli je loď přístupná nějaké činnosti. Kontroly jsou v implementaci často vícekrát (na straně herní simulace a herního rozhraní) z toho důvodu, že simulace a webové rozhraní nejsou synchronizované. Tím se myslí, že když webové rozhraní vyvolá nějakou akci, na serveru k ní dojde se zpožděním, mnohdy i několika sekund.

10.3.8 Práce s nákladem

Při zobrazování stavu nákladu bylo hojně využito HTML součásti pro zboží. Bylo ale zapotřebí měnit data, která jsou zobrazována, například při zobrazení ceny. Ta je totiž různá pro nákup, prodej a již zakoupené zboží (obsahuje informaci, kolik kreditů stál jeden kus zboží). Také je zde využito formulářů s implementovanou podporou AJAXu. Pomocí jQuery a výpočtů v šabloně je pak při nákupu a prodeji zajištěna možnost nastavení maxima a minima, barevná zobrazení důležitých hodnot (například je zdůrazněno, když zboží došlo) a možnost koupit či prodat maximální počet zboží pomocí jednoho tlačítka. Tlačítka při nákupu a prodeji jsou barevně odlišena, aby se snížilo riziko, že si hráč splete nákup s prodejem. Ukázka tohoto odlišení je na obrázku 10.5.



Obrázek 10.5: Srovnání „karet“ zboží pro nákup a prodej

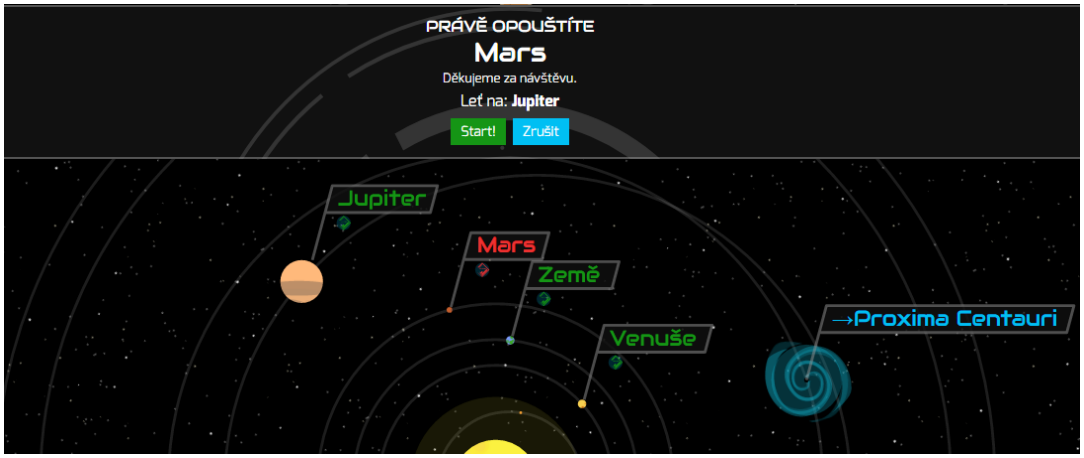
Nákup, prodej a seznam nákladu neměly uživatelské rozhraní, takže bylo v rámci této práce doimplementováno. Obsluha formulářů pro nákup a prodej byla přidána do příslušného controlleru. V rámci toho bylo nutné refaktorovat a pozměnit akce realizující nákup, prodej, nakládání a vykládání zboží. Vnitřní implementace totiž počítá s více možnostmi zadávání vstupu (například z automatického skriptu).

10.3.9 Tankování a opravy

Tato funkcionalita podobně jako nákupy a prodeje vyžadovala implementaci a ladění vnitřní logiky. Implementace formulářů pro tankování a opravy je obdobná jako u práce s nákladem. V herní simulaci navíc tankování i opravy trvají určitou dobu. Příslušné akce herní simulace tak byly upraveny, aby se zobrazoval správný stav lodi.

10.3.10 Plánovač

Implementace uživatelského rozhraní plánovače cest lodí zcela chyběla. Ajaxové tlačítko „Leť na planetu…“ spouští hlavní okno, obsahující skript pro plánování. Ten přepne mapu do jiného módu tak, že jí přiřadí CSS třídu. Tak je změněn vzhled mapy podle návrhu pomocí CSS pravidel. Pro tento účel bylo zapotřebí do mapy předat přes její modely více dat k rozlišení jednotlivých prvků mapy. Rozhraní plánovače a mapu v módu plánování vidíme na obrázku 10.6.



Obrázek 10.6: Plánovací mód

Skript pro plánování také pomocí jQuery obsluhuje události kliknutí na planetu a podle nich aktualizuje svá data, tedy planetu, na kterou se má letět a červí díry,

kterými se má cestou proletět. Tato data jsou při zadání zobrazena v okně. Také je možné data zadaná data zrušit. Při plánování trasy je mapa přepnuta na hvězdný systém, kde se loď nachází, aby nedocházelo ke špatnému zadávání.

U jednotlivých akcí herní simulace, které ovlivňovaly let lodi, byl aktualizován stav lodi, aby zprávy o jejím stavu byly stále aktuální. Plánovač na serverové části také neobsahoval podporu pro předání výsledku akce uživateli. Přelety lodí se totiž dějí v herní simulaci nezávisle na webovém rozhraní. Pokud ale loď například nemá dostatek paliva (a tudíž ani neodstartuje), hráč se to nijak nedověděl. Z tohoto důvodu byla doimplementována kontrola stavu lodi ještě před zadáním akce, což vyžadovalo analýzu a drobný refaktoring existujícího kódu. U zjišťování délky cesty tak musela být cesta vypočítána znovu ještě před skutečným startem lodi.

11 Doporučení pro další vývoj

Během implementace byly autorem této práce získány cenné znalosti stavu hry a jejího uživatelského rozhraní. Předpokládá se další vývoj aplikace a rozvoj možností hry, včetně jejího uživatelského rozhraní. Z poznatků této práce vyplynula doporučení a možnosti, kterými je vhodné se zabývat při dalším vývoji uživatelského rozhraní. Tato kapitola je popisuje.

11.1 Jméno lodi

Každá loď obsahuje v databázi sloupec pro jméno lodi. V současnosti jsou data v tomto sloupci shodná s typem lodi, který je uvedený v jiném sloupci. Předpokládá se totiž možnost změny jména lodi. Uživatel by měl mít možnost jméno lodi někde zadat, například při nákupu nebo v detailu lodi.

11.2 Změna vzhledu lodi

Podobně jako u jména jsou v kódu připraveny dobré možnosti pro umožnění hráči vybrat si rozložení barev své lodi. K tomu lze použít sloupce z databáze, které obsahují vektorový obrázek a CSS třídu lodi. Volitelně by také mohla být přidána možnost změnit celkové vzezření lodi, například pomocí zviditelnění předdefinovaných objektů v obrázku lodi (obrázek může obsahovat objekty, které jsou průhledné a mohou se zviditelňovat až s uživatelským nastavením). Obrázek musí za každou cenu zůstat vektorový. Při úpravě SVG kresby lodi třeba dbát na to, aby nebyl poškozený zdrojový soubor – některé objekty mají například nastavenou třídu. Zdrojové obrázky byly kresleny za pomoci vektorového editoru Inkscape.

11.3 Zprávy v informační liště

Informační lišta ve spodní části obrazovky prozatím obsahuje náhodné zprávy, které nemají s děním ve hře nic společného. Komponenta, která je zobrazuje, umožňuje dynamické přidávání a odebírání zpráv. Je tedy vhodné implementovat generování a zobrazování zpráv, které mají pro hráče význam. Volitelně je také možné doimplementovat možnost kliknout na informační lištu a zobrazit všechny zprávy.

11.4 Zobrazení doby trvání akce

V současnosti stav lodi ukazuje, že loď například letí vesmírem nebo tankuje palivo. Neexistuje však žádná možnost, jak zjistit, kdy budou tyto akce dokončeny. Tuto možnost lze však doimplementovat, informace o času dokončení může být například uložena v databázi a následně čtena webovým rozhraním. Uživateli by se pak zobrazoval například časovač, odpočítávající čas do dokončení akce.

11.5 Vylepšené obnovování stavu

V současné implementaci (tj. po dokončení této práce) je pro aktualizaci nutné, aby uživatel znovu načetl příslušnou součást, například pomocí obnovovacího tlačítka. Stálo by za zvážení, zda tento způsob nelze nějak efektivně vylepšit. Při analýze je však třeba brát velký ohled na vytížení serveru (například počtem požadavků) a složitost implementace. Současný systém je udělaný tak, aby přidávání nového UI bylo co nejjednodušší, protože to bude dělat každý vývojář a tak by to také mělo zůstat.

V rámci implementace této funkcionality je také vhodné zkoumat, která okna se hráči otevírají a zavírají během hry a případně provést změny, vedoucí k lepšímu hráčskému zážitku. Například lze řešit, zda se při plánování trasy má při kliknutí na planetu zobrazit kontextové menu základny, nebo zda má zůstat zobrazený detail lodi.

11.6 Výkonová náročnost klientské části

V současnosti je prohlížeč vytížen zejména vykreslováním mapy. K tomu dochází kvůli výpočtu drah planet, jejich animacím a obnovováním celého vektorového obrázku. Měla by se provést analýza, která část zabírá vyšší výkon a provést optimalizaci úzkých míst v kódu. Veškeré úpravy ale nesmí mít vliv na aktuální zobrazení. Je bezpodmínečně nutné počítat s tím, že výpočet drah je stejný jako na serverové části a vzdálenosti mezi planetami jsou pro hráče velmi důležité. I animace jsou pečlivě vyvážené a při jejich vývoji bylo dbáno na výkon, takže případné změny jsou vhodné až po hlubší analýze a konzultaci.

Jako možná řešení se jeví snížení FPS, které by současné vykeslování mapy mohlo umožňovat nebo možnost uživatelského nastavení, kde by si uživatel mohl například vypnout animace planet či paralaxové hvězdné pozadí.

11.7 Protihráči

Současná implementace se soustředí pouze na hráče, který právě hraje. Nemá žádnou možnost, jak vidět činnost ostatních uživatelů, například nemůže zjistit, kolik lodí jiných hráčů je na určité planetě. Původní implementace také obsahovala nefunkční žebříček hráčů, který by bylo možné zprovoznit (jeho položka v menu je pouze zakomentována a aplikace obsahuje příslušné controllery a šablony).

Při nákupu a prodeji také současná implementace nepočítá například s rychlým nakupováním a prodejem při více hráčích, jelikož nebylo možné tento stav vyzkoušet. Tento problém je zapotřebí analyzovat nejen z hlediska nákupů a prodejů.

11.8 Zobrazení lodi na mapě

Při létání lodí je striktně vypočítávána vzdálenost, kterou loď musí uletět. Lodě také oblétávají hvězdy. Herní simulace by měla mít informaci o tom, kde se právě loď

nachází. Tuto informaci by se nějakým způsobem měl dovědět uživatel. Zobrazená trasa lodi nemusí být úplně přesná, ale měla by trvat stejnou dobu. Obrázky lodí jsou ve formátu SVG a lze je tedy využít pro zobrazení na mapě. SVG technologie také podporuje animaci, kdy se daný objekt pohybuje po křivce. Při kliknutí na loď by se měl zobrazit její detail. Implementace této funkcionality je poměrně složitá, ale výsledek by měl být pro uživatele velmi zajímavý. Je na zvážení z hlediska výkonu, zda uživateli zobrazovat pouze jeho lodě, nebo i lodě ostatních hráčů.

11.9 Usability testy

V rámci této práce bylo provedeno několik usability testů. Jedná se však o iterativní proces, který by se měl čas od času zopakovat. Tato práce proces nastiňuje, ale byla omezena možnostmi starého uživatelského rozhraní. Do budoucna by bylo praktické nastavit stálý proces usability testování a pečlivě ho popsat na wiki projektu. Doporučuji dělat například tři testování v krátkém čase každý semestr, ideálně na začátku, protože na konci bývá obvykle málo času a vůle na opravy. Toto testování by bylo možné využít při konzultacích a při seznamování se s projektem. Typickým výstupem je totiž zpravidla relativně jednoduchá činnost, na které si správce projektu může ověřit schopnosti vývojáře.

11.10 Mobilní verze

Současná implementace nepodporuje displeje s menším rozlišením než 1024x768. Vzhledem ke zobrazeným prvků to není možné ani vhodné. Případná verze pro mobil by měla vykreslovat celou stránku zcela jiným a specifickým způsobem a mobilní verze by měla mít své vlastní šablony. Vytvoření mobilní verze je časově náročné a nepříliš důležité, ale tato implementace by se měla provést v době, kdy bude hra nasazena na produkci a bude ji hrát větší množství hráčů.

11.11 Zobrazení vývoje herní ekonomiky

Simulace herní ekonomiky počítá s tím, že se budou měnit ceny jednotlivých druhů zboží. Z této simulace by bylo vhodné získávat potřebná data a vytvořit z nich grafy nebo tabulky, které by byly užitečné pro hráče i vývojáře. Hráči by se tato data měla nějakým způsobem zobrazovat. Vývojář navíc bude mít možnost lépe nastavit parametry herní ekonomiky.

11.12 Další možnosti zlepšení hry

Tato část textu se zabývá doporučeními, které se netýkají přímo uživatelského rozhraní, ale je možné je uskutečnit. Jejich provedení nemá z hlediska funkčnosti hry větší význam a slouží jen pro představu, co by bylo možné doplnit.

11.12.1 Pronajímání skladů

Na serveru existuje akce pro nakládání, která se nyní volá hned po nákupu. Pro hráče by mohlo být zajímavou možností koupit si zboží na planetě, i když na ní právě nemá žádnou loď. Měl by se mu účtovat skladovací poplatek a náklad by mohl na loď naložit později. Implementace této funkcionality je náročnější, než se na první pohled zdá a vyžaduje relativně velké množství implementace. Měla by se implementovat až když si to vyžádá gamedesign.

11.12.2 Hráčská úroveň

V současnosti je ve hře úroveň hráče, která se postupně zvyšuje. Nemá však na hráče žádný vliv. Na základě gamedesignu by se s ní měly postupně odemykat další možnosti, například by úrovní mohla být podmíněna koupě zboží nebo lodí. To je ale zapotřebí doimplementovat a také přidat definice herního světa, aby hráč měl stále dostatečný výběr i s tímto omezením.

11.12.3 Rozšíření herního světa

Ve hře lze v souborech XML definovat hvězdné systémy s jejich planetami, dostupné lodě a dostupné zboží. V rámci této práce bylo několik definic přidáno, ale vždy je možné přidat další. Jde však čistě o tvůrčí činnost, která by měla být dělána buď dobrovolně nebo jako drobné doplnění zadání.

11.12.4 Vylepšování lodi

Parametry každé lodi jsou uloženy v databázi. Principiálně je tedy možné je měnit u jednotlivých lodí. K lodím by tedy mohla přibýt možnost vylepšovat motor, skladovací prostor apod. Jak je možné lodi vylepšit by mělo být definováno v jejich XML definicích.

Parametr výkon lodi není pravděpodobně v aplikaci k ničemu použitý a měla by se provést krátká analýza, zda tomu tak skutečně je a jak s tím naložit.

11.12.5 Uživatelské zprávy

Původní implementace naznačovala možnost odesílání zpráv mezi uživateli. V menu byla položka, která je v současné době zakomentována. Hra obsahuje i příslušné controllery a šablony, které ale nejsou v provozu. U této funkcionality je ale třeba se zamyslet nad tím, zda tento způsob posílání zpráv není zastaralý a nebylo by vhodnější implementovat ho jiným způsobem. Jde také o poměrně náročnou činnost a přednost by měly mít funkcionality zásadnější pro chod hry.

12 Podpůrné činnosti

Nepsaným cílem této práce bylo upravit existující implementaci tak, aby hru bylo možné alespoň na základní úrovni hrát. To vyžadovalo i hlubší zásahy do implementace, bez kterých by tento cíl nebylo možné naplnit, přestože se netýkaly přímo zadání této práce. Tato kapitola popisuje ty nejpodstatnější z nich. Zároveň je zde i obsažena činnost prováděná kvůli konvencím či cílům celého projektu.

12.1 Jednoduchá ekonomika hry

Aby bylo hru možné smysluplně hrát, bylo zapotřebí zajistit určité chování obchodníků, jako je například rozdílnost cen zboží na různých planetách. Implementace této ekonomiky je velice primitivní a počítá se s tím, že bude pouze dočasná a do hry bude doimplementována složitější simulace ekonomiky.

12.1.1 Chyby v nákupech a prodejích

Nákupy a prodeje je složitá funkcionalita, která ale z historických a personálních důvodů neobsahovala grafické rozhraní. Z toho důvodu pravděpodobně její tvůrci neměli možnost ji detailně odladit. Zároveň také neměli k dispozici pohled konečného uživatele, který odhalil několik chyb v celkovém designu. V rámci této práce bylo tedy odladěno několik drobných chyb. Typicky šlo o záměnu ID nebo nevhodný design z hlediska hratelnosti hry.

12.1.2 Prodejní logika

Původní implementace nákupů a prodejů počítala s tím, že každý obchodník bude prodávat i nakupovat všechny druhy zboží. Když ale bylo prodáno všechno zboží daného druhu, z databáze se tato obchodníkova položka vymazala. Tím se ale ztrácela informace o ceně daného zboží u toho či onoho obchodníka, což je z hlediska gamedesignu neudržitelné. Zároveň nebylo jasné, za jakou cenu by měl obchodník vykupovat zboží, které nenabízí. Kód byl tudíž v rámci této práce změněn tak, aby obchodník mohl mít nulové množství určitého produktu a řádek se z databáze nikdy nemazal. Obchodník tak nyní vykupuje jen zboží, které sám nabízí. Tato změna ale vyžadovala podrobné ladění této funkcionality, změnu některých dotazů na databázi a drobné změny jejich testů.

12.2 Herní obsah

Obsah hry je definován v souborech XML. Jejich editace není technicky náročná a jde čistě o tvůrčí činnost, avšak vytváření obsahu je časově náročné. Aby ale bylo možné plně demonstrovat funkčnost hry, byly v rámci této práce vytvořeny definice dvou druhů obsahu.

Pro demonstraci možností planet byly po rozšíření funkcionality editován soubory definující hvězdné systémy a oběžné dráhy planet. Soustavy byly zvětšeny, planetám hlavní soustavy byly přidány popisky a barvy.

Pro demonstraci možností obrázků lodí byla definována celá škála lodí. Také byly vytvořeny SVG kresby těchto lodí, aby bylo možné ukázat na příkladu. Kreseb bylo nakresleno celkem 5, každá má pak reprezentovat jednu cenovou kategorii lodí. Také byly vytvořeny 4 grafické styly (barevná rozložení) těchto lodí – standardní, „čínský“, „německý“, „americký“. Takto vzniklo 20 různých zobrazení lodí, z nichž ke každé byly definovány parametry.

12.3 Nasazení na produkční server

Pro účely prezentací a usability testů byla aplikace nasazena na produkční server. Kromě samotného nasazení bylo zapotřebí vyřešit problémy v aplikaci, které bránily tomu, aby na produkčním prostředí hra fungovala stejně jako na lokálním.

12.3.1 Base url path

V aplikaci se na různých místech používají relativní URL adresy. U mnohých ale chybělo přidání adresy kořenu, protože na lokálních strojích je to pouze znak „/“ a proto adresy fungují i bez něj. Z toho důvodu však na produkci vznikly chybné odkazy na obrázky, některé zdrojové soubory a URL v aplikaci. Všechny tyto odkazy byly v rámci nasazení opraveny a pro klientskou část byla přidána potřebná proměnná.

12.3.2 Přístup k souborům

Produkční prostředí také zamítalo přístup k některým zdrojům, především SVG kresbám a některým XML souborům, které se načítaly z prohlížeče. Absence těchto zdrojů samozřejmě způsobovala v aplikaci chyby. Do konfiguračního souboru aplikace tudíž byly přidány definice, které umožňovaly tyto a pro všechny případy i některé další zdroje načítat.

12.4 Dokumentace na wiki

Konvence projektu Space Traffic vyžadují dokumentaci kódu na vývojářské wiki [40]. Z toho důvodu je tato činnost obsažena i v zadání této práce. Koncepce této wiki vyžaduje spíše názornou a méně popisnou dokumentaci. Z tohoto důvodu byla v rámci této práce vytvořena dokumentace popisující především použití vytvořených podpůrných funkcionalit. Samotný způsob fungování byl pak popsán především u složitějších součástí. Předpokládala se znalost projektu a základních webových technologií. Zdokumentovány byly všechny podstatné provedené změny:

- ajaxová podpůrná knihovna [34]
- použitelné HTML a CSS prvky [41]
- mapa hvězdného systému [42]

- SVG kresby lodí [43]

Zbytek funkcionality byly zpravidla jednoduché skripty a CSS styly, které jsou samovysvětlující a nevyžadují konkrétnější dokumentaci. Zároveň byly na wiki popsány některé chyby [39] [44], se kterými se autor práce setkal.

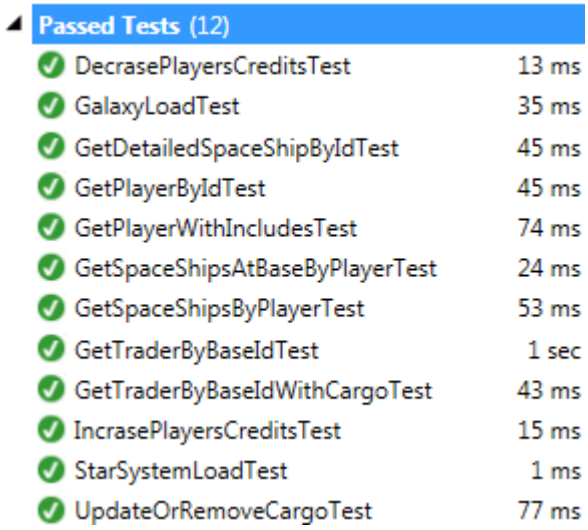
13 Testování

Aplikace Space Traffic aktuálně poskytuje dva způsoby unit testování. Přidání další technologie pro testování je u takto velkého projektu časově a analyticky náročná činnost, kterou není možné provést v rámci této práce. Pro účely automatického testování je tedy nutné vystačit si pouze s dostupnými technologiemi.

13.1 Visual C# unit testy

Projekt obsahuje několik testovacích podprojektů, které umožňují psát jednotkové testy a spouštět je přímo v prostředí Visual Studia. Testují se v nich zejména DAO služby, tedy služby přistupující k databázi. V rámci této práce bylo zapotřebí přidat několik metod do těchto služeb. Ke všem těmto metodám byly dopsány jednotkové testy. Několik metod bylo také změněno a vlivem změn v aplikaci několik testů přestalo úspěšně procházet. Tyto testy byly pozměněny, protože se změnilo i očekávané chování dané funkcionality (například šlo o změny prodejní logiky, popsané v odstavci 12.1.2).

Testy lze spustit pomocí Visual Studia tlačítkem z menu Test → Run → All tests. Projekt obsahuje i testy dřívější implementace. Příklad výsledků dopsaných a pozměněných testů v rámci této práce můžeme vidět na obrázku 13.1.



Obrázek 13.1: Výpis výsledků unit testů

13.2 Qunit testy

Jedná se o testovací prostředí na klientské části aplikace. Jsou závislé na knihovně jQuery a spouští se přímo v prohlížeči. Jsou tedy vhodné pro testování uživatelského rozhraní a přidružených jQuery komponent.

13.2.1 Úprava aplikace technologie

V poslední verzi byly tyto testy aplikovány mimo celou aplikaci, kde se zavolala příslušná URL adresa mimo aplikaci, která testy spouštěla. Také již existovalo několik testovacích případů, z nichž ale zhruba polovina neprocházela, pravděpodobně kvůli změnám v aplikaci. Ukázalo se, že tyto testy jsou závislé na přihlášení uživatele, protože vykonávaly i ajaxové požadavky.

Použití Qunit bylo tudíž rozděleno na dvě části – jedna byla aplikována v původním umístění a slouží pro testy, které nevyžadují, aby byl uživatel přihlášený. Jsou tedy vhodné spíše pro testování uživatelského rozhraní mimo hru samotnou.

Druhá část Qunit testů byla připojena přímo do herního prostředí. Spouští se opět pomocí specifické URL adresy, kterou je nutné zadat přímo do adresního řádku v prohlížeči. Mimo tuto URL se Qunit testy vůbec nepřipojují, aby nezpomalovaly běžný chod aplikace. Protože se jedná o one-page aplikaci a testy jsou připojeny do layoutu spolu se všemi běžně použitými závislostmi (pod spuštěnými testy vlastně běží samotná hra), bylo možné těmito jednotkovými testy kromě funkcí a komponent testovat také například chování či přítomnost konkrétních prvků na obrazovce.

Způsob testování nebyl zdokumentován, takže byl na wiki doplněn stručný návod pro jejich použití [45].

13.2.2 Qunit testování

V rámci této práce byly opraveny starší testy z minulých let, protože byly zastaralé. Bylo také připsáno několik desítek nových testovacích případů. Důraz byl kladen především na testování funkčnosti podpůrných součástí (například knihoven pro komunikaci) a kontrolu nezbytných prvků na obrazovce. Na obrázku 13.2 vidíme část testovacího výpisu pro přihlášeného uživatele. Všech 60 testů z výpisu jsou testy implementované či upravené v rámci této práce.

Obrázek 13.2: Výpis výsledků Qunit testů

13.3 Ruční testování

Během vývoje byla aplikace opakovaně testována ručně a výstup kontrolován vizuálně. Objevilo se několik důležitých scénářů, které odhalily několik chyb. Jednalo se zejména o chování nákupů a prodejů a příslušné zobrazení. Následuje příklad prováděného scénáře.

1. Zobrazení detailu konkrétní lodi
2. Kliknutí na tlačítko Nakup zboží
3. Kontrola toho, že nabízené zboží je včetně cen stejné jako zboží dostupné na dané základně, kde se loď nachází.
4. Zvolení množství nákupu a odeslání formuláře pro nákup
5. Kontrola dat v databázi, zejména koupené množství
6. Kontrola změny množství kreditů hráče
7. Kliknutí na tlačítko Náklad v detailu lodi a vizuální kontrola zboží, které bylo právě zakoupeno
8. Přelet na jinou planetu pomocí plánovače
9. Kliknutí na tlačítko Náklad a kontrola cen nákladu. Cena by měla být stále stejná (jde o údaj, za kolik kreditů bylo konkrétní zboží koupeno)
10. Kliknutí na tlačítko Prodej zboží a následná kontrola ceny. Měla by být stejná, jako je cena zboží u obchodníka na této planetě
11. Prodání zboží pomocí formuláře
12. Kontrola dat v databázi – množství a cena zboží u obchodníka, množství a cena zboží obsaženého v lodi

U dalších scénářů se jednalo o variace s větším množstvím lodí, s nákupy stejných druhů zboží za jinou cenu apod. Odhalené chyby byly opraveny.

13.4 Usability testy

Usability testování je spíše proces zlepšování uživatelského rozhraní, protože se ale jedná o způsob zajištění kvality aplikace, je jeho konkrétní použití popsáno v rámci této kapitoly. Byly provedeny celkem tři testy s určitým časovým odstupem a v různých fázích vývoje.

13.4.1 První test – Den otevřených dveří

Aplikace v raném stádiu vývoje uživatelského rozhraní byla prezentována na Dni otevřených dveří Fakulty aplikovaných věd dne 27. ledna 2016. Během toho mělo větší množství různých uživatelů možnost si hru vyzkoušet. Zvláštností bylo, že uživatelé nevěděli, že jsou testováni.

V době konání však aplikace měla hotovou prakticky jen mapu hvězdného systému. Přesto bylo pozorováním chování uživatelů, kteří prakticky byli cílovou skupinou pro hru, zjištěno několik užitečných faktů:

- uživatelé intuitivně pohybují mapou i bez informací o tom, jak se s mapou zachází
- uživatelé intuitivně používají kolečko myši k přibližování a oddalování mapy
- uživatelé klikají na planety a popisky planet vnímají jako objekty, na které je možné kliknout (v této verzi aplikace se však zobrazil pouze popis planety)
- uživatelé obvykle neobjevili možnost kliknout na hvězdu a zobrazit popis hvězdného systému
- uživatelé při kliknutí na červí díru pochopili, že se přenesli na jiný hvězdný systém a nebyli touto změnou zaskočeni

Jediným problémem zjištěným při tomto testu tak bylo to, že se hvězda části uživatelům nejeví jako objekt, na který je možné kliknout. Kliknutí na hvězdu však pouze zobrazuje popisek a jeho nezobrazení nijak uživatele neomezí při hraní hry. Ani v dalších verzích aplikace se nepočítalo s tím, že by kliknutí na hvězdu mělo zobrazovat důležité prvky ovládání. Řešení by navíc bylo relativně složité a vyžadovalo by další usability test. Z těchto důvodů byl problém při třídění vyškrtnut.

13.4.2 Druhý test – prezentace vedoucímu

Vedoucí této práce byl o průběhu práce pravidelně informován. Při předvedení aplikace, kde byly již implementováno uživatelské rozhraní potřebné ke hře, byl proveden usability test. Vedoucí představoval uživatele a zkušeného hráče s konkrétní představou, o čem hra je. Protože se jedná o hru, nebyly zadávány žádné konkrétní úkoly a uživatel byl pouze pobídnut, aby si zahrál.

Nalezené problémy byly tříděny již v průběhu testu a k dalšímu řešení byly vybrány jen problémy zásadního významu nebo změny s dobrým poměrem mezi náročností implementace a přínosem pro uživatele. Šlo o následující problémy:

- u plánovače by se tlačítka START a ZRUŠIT měla zobrazovat až když je nějaká cesta naplánována
- kontextové menu u základny by si mělo uchovat informaci o tom, která jeho část byla právě otevřena a ta by se měla zobrazit při otevření jiné základny
- při prodeji nastává chyba způsobující zdvojování zboží stejného druhu u obchodníka na planetě
- nákupy a prodeje by se měly zobrazovat v hlavním okně kvůli lepší přehlednosti (původně se zobrazovaly v kontextovém menu pod detailem lodi, podobně jako například formulář pro opravy lodi)
- pokud nějaké zboží obchodník pouze vykupuje (vlastní ho nulové množství), mělo by to být graficky zdůrazněno

- číslo celkového počtu kreditů by mělo být zformátováno pro větší přehlednost s mezerami
- v detailu lodi by se měla zviditelnit informace o tom, na které planetě se loď nachází
- v plánovacím módu mapy by měla být graficky odlišena planeta, ze které loď startuje
- u prodeje zboží se zobrazuje cena, za kterou bylo nakoupeno místo ceny, za které je možné ho prodat

Všechny tyto problémy byly následně odstraněny.

13.4.3 Třetí test – vývojáři

Testovací uživatele při třetím testu tvořili členové týmu vytvořeného v rámci předmětu KIV/ZSWI, kteří se podíleli na vývoji projektu. Jejich úkolem bylo vytvořit implementaci nesouvisející přímo s herním prostředím, přesto však samotnou hru již znali. Většina problémů zjištěných při druhém testu byla při třetím testu již odstraněna.

Testování probíhalo opět formou pozorování těchto uživatelů při hraní hry. Jediným zjištěným problémem bylo zobrazování prázdného hlavního okna za určitých okolností. Tato chyba byla následně opravena. Jiné problémy nebyly nalezeny.

13.4.4 Závěry

K usability testování nebyl v rámci této práce příliš velký prostor, především z časových a personálních důvodů. Testů by mohlo být více a měly by být prováděny s uživateli, kteří hru pokud možno neznají. Teoretická část ale tvrdí, že je vždy lepší provést testy s horší kvalitou než žádné testy. To se ostatně potvrdilo i v těchto testech, kde byly odhaleny chyby v aplikaci a řešení zjištěných problémů vedlo k lepší použitelnosti uživatelského rozhraní hry.

14 Závěr

Nepsaným cílem této práce bylo vytvořit hratelnou verzi hry Space Traffic. Vzhledem k rozsahu, složitosti a chybějícím detailům v implementaci muselo být pro splnění tohoto cíle vynaloženo nadstandardní množství času. Tento cíl byl přesto splněn a hru nyní poprvé v historii projektu může hrát i běžný uživatel. Grafické rozhraní přinejmenším splňuje dnešní nároky na vizuální vzhled webové hry a navíc je technicky zajímavé.

Nový a do detailu odladěný vzhled také vyžadoval velké množství implementace, kterou nebylo možné kvůli rozsahu popsat v této práci. Jednalo se o technicky nezajímavou implementaci, kde autor pouze využil své předchozí zkušenosti s vytvářením webových rozhraní. Nicméně byl tento kód pro celkový vzhled a chování aplikace zcela zásadní a bylo mu věnováno větší množství času a úsilí.

Při návrhu a implementaci bylo velmi dbáno na znovupoužitelnost kódu a jeho dokumentaci. Také byla doplněna dokumentace některých součástí hry, které nebyly předmětem této práce, ale bylo zapotřebí jejich použití. V rámci této práce nebyla zanedbána žádná část podpůrného kódu a nikde se nepočítalo s doplněním implementace někým jiným v budoucnu, byť přímo nesouvisela se zadáním práce. Obecně autorovi této práce záleželo nejen na uskutečnění diplomové práce, ale i na projektu Space Traffic samotném.

Autor této práce zastával na projektu v akademických letech 2014/2015 a 2015/2016 roli jednoho ze studentských správců. Konkrétně fungoval jako hlavní programátor a analytik, významně se podílel se na stanovení zadání pro další vývojáře a zodpovídal za repozitář pro verzovací nástroje. Také poskytoval technickou podporu na schůzkách s vývojáři a podílel se na správě virtuálního serveru z hlediska poskytovaných služeb.

Výše zmíněná činnost, realizace této práce a další zkušenosti s projektem Space Traffic měly pro autora velký vzdělávací přínos i přes svou časovou a technickou náročnost.

Reference

1. **FAV, studenti KIV.** Space Traffic. [Online] http://spacetraffic.kiv.zcu.cz/.

2. **Štepánek, Bc. Martin.** *Architektura a implementace webové hry pro více hráčů.* 2012.

3. **Schell, Jesse.** *The Art of Game Design.* 2008. ISBN: 978-0-12-369496-6.

4. **Fullerton, Tracy.** *Game design workshop.*

5. **Kujala, Sari, a další.** *UX Curve: A method for evaluating long-term user experience.* 20.4.2014.

6. **Krug, Steve.** *Don't make me think! A Common Sense Approach to Web Usability, Second Edition.* 2006. ISBN 0-321-34475-8.

7. Breadcrumb (navigation). *Wikipedia, The Free Encyklopedia.* [Online] [Citace: 9. duben 2016.] https://en.wikipedia.org/wiki/Breadcrumb_(navigation).

8. Three-click rule. *Wikipedia, The Free Encyklopedia.* [Online] [Citace: 8. duben 2016.] https://en.wikipedia.org/wiki/Three-click_rule.

9. **Gócza, Zoltán.** UXMyths. [Online] [Citace: 9. duben 2016.] http://uxmyths.com/post/654026581/myth-all-pages-should-be-accessible-in-3-clicks.

10. **Cannon, Thomas.** An Introduction to Color Theory for Web Designers. *envatotuts+*. [Online] 12. Září 2012. [Citace: 9. duben 2016.] http://webdesign.tutsplus.com/articles/an-introduction-to-color-theory-for-web-designers--webdesign-1437.

11. Responsive Web Design - Introduction. *W3Schools.* [Online] [Citace: 10. leden 2016.] http://www.w3schools.com/css/css_rwd_intro.asp.

12. W3.CSS. *W3Schools.* [Online] http://www.w3schools.com/w3css/.

13. *Bootstrap.* [Online] http://getbootstrap.com/.

14. SpaceTraffic. *GitHub.* [Online] https://github.com/SpaceTrafficDevelopers/SpaceTraffic.

15. ASP.NET. [Online] http://www.asp.net/.

16. HTML5. *W3C.* [Online] https://www.w3.org/TR/html5/.

17. *jQuery.* [Online] https://jquery.com/.

18. jQuery Core 1.9 Upgrade Guide. *jQuery.* [Online] https://jquery.com/upgrade-guide/1.9/.

19. AJAX. *Wikipedia, The Free Encyklopedia.* [Online] https://en.wikipedia.org/wiki/Ajax_(programming).

20. About SVG. *W3C.* [Online] https://www.w3.org/Graphics/SVG/About.html.

21. Can I use SVG? *Can I use...* [Online] http://caniuse.com/#search=SVG.

22. **Eisenberg, J. David Sebastopol.** *SVG Essentials.* místo neznámé : O'Reilly Media, Inc., 2002.

23. CSS Introduction. *W3C.* [Online] http://www.w3schools.com/css/css_intro.asp.

24. CSS3 Introduction. *W3C.* [Online] http://www.w3schools.com/css/css3_intro.asp.

25. Dynamický jazyk pro tvorbu stylesheetů. *less.* [Online] http://www.lesscss.cz/.

26. **Bone, Sonny.** Parallax Scrolling: A Simple, Effective Way to Add Depth to a 2D Game. *envatotuts+*. [Online] [Citace: 15. duben 2016.] http://gamedevelopment.tutsplus.com/tutorials/parallax-scrolling-a-simple-effective-way-to-add-depth-to-a-2d-game--cms-21510.

27. *Google Fonts.* [Online] https://www.google.com/fonts.

28. Single-page application. [Online] https://en.wikipedia.org/wiki/Single-page_application.

29. *jQuery Form Plugin.* [Online] http://jquery.malsup.com/form/.

30. Polling. *Wikipedia, The Free Encyklopedia.* [Online] https://en.wikipedia.org/wiki/Polling_(computer_science).

31. **Hanson, Joe.** What is HTTP Long Polling? *PubNub.* [Online] [Citace: 24. listopad 2015.] https://www.pubnub.com/blog/2014-12-01-http-long-polling/.

32. About HTML5 WebSocket. *websocket.org.* [Online] [Citace: 24. listopad 2015.] http://www.websocket.org/aboutwebsocket.html.

33. IIS 8.0 WebSocket Protocol Support. *IIS.* [Online] Microsoft. [Citace: 24. listopad 2015.] http://www.iis.net/learn/get-started/whats-new-in-iis-8/iis-80-websocket-protocol-support.

34. **Kotalík, Jan.** AJAXová komunikace. *Space Traffic development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=AJAX.

35. jquery/jquery-migrate. *GitHub.* [Online] https://github.com/jquery/jquery-migrate/tree/1.x-stable#readme.

36. jquery/jquery-mousewheel. *GitHub.* [Online] https://github.com/jquery/jquery-mousewheel.

37. **Bau, David.** Random Seeds, Coded Hints, and Quintillions. *davidbau.com.* [Online] [Citace: 20. leden 2016.] http://davidbau.com/archives/2010/01/30/random_seeds_coded_hints_and_quintillions.html.

38. Marquee jQuery Plug-in v1.0.01. *Giva labs.* [Online] http://www.givainc.com/labs/marquee_jquery_plugin.cfm.

39. **Kotalík, Jan.** Pád při načítání z ASP pomocí Include. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=P%C3%A1d%20p%C5%99i%20na%C4%8D%C3%ADt%C3%A1n%C3%AD%20z%20ASP%20pomoc%C3%AD%20Include.

40. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code.

41. **Kotalík, Jan.** Uživatelské rozhraní. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=U%C5%BEivatelsk%C3%A9+rozhran%C3%AD.

42. —. Mapa hvězdného systému. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Mapa+hv%C4%9Bzdn%C3%A9ho+syst%C3%A9mu.

43. —. Lodě. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Lod%C4%9B.

44. —. Objekt komunikace, System.ServiceModel.Channels.ServiceChannel, nelze použít pro komunikaci, protože je ve stavu chybný. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Objekt%20komunikace%20nelze%20pou%C5%BE%C3%ADt%20pro%20komunikaci%2C%20proto%C5%BEe%20je%20ve%20stavu%20chybn%C3%BD..

45. —. Testování. *Space Traffic Development.* [Online] http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Testov%C3%A1n%C3%AD.

Přílohy

Příloha A – uživatelská příručka

Tento dokument slouží jako uživatelská příručka k hraní hry Space Traffic.

Spuštění

Ke spuštění projektu v lokálním prostředí je zapotřebí mít nainstalované následující nástroje:

- Visual Studio 2013
- MS SQL Server 2012
- ASP .NET MVC 3
- NLog

Podrobnosti včetně odkazů, podrobného nastavení a postupu instalace těchto nástrojů najdete na vývojářské wiki projektu (*http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Instalace+nástrojů*).

Poté můžete projekt spustit:

1. Otevřete si projekt (SpaceTraffic.sln**) ve** Visual Studiu
2. V Solution Exploreru klikněte pravým tlačítkem na solution a zvolte nastavení
3. Nastavte Startup project tak, aby se spouštěly projekty GameServer a Game UI
4. Spusťte projekt pomocí Visual Studia tlačítkem Start

Podrobnější návod s obrázky najdete na vývojářské wiki (*http://spacetraffic.kiv.zcu.cz/Code/tiki-index.php?page=Spuštění+projektu*).

Demo hry by také mělo běžet na produkčním prostředí, které ale může obsahovat zastaralou verzi (*http://spacetraffic.kiv.zcu.cz/Demo/*).

Hraní hry

Pro hraní hry se musíte nejprve zaregistrovat a přihlásit pomocí příslušných odkazů. Po přihlášení uvidíte hvězdnou mapu a uživatelské rozhraní hry. Uživatelské rozhraní při nepřihlášeném uživateli není součástí této práce.

Mapa

Mapu hvězdného systému lze přibližovat a oddalovat pomocí kolečka myši. Kliknutím a tažením lze pak mapou pohybovat. Kliknutím na jednotlivé planety zobrazíte jejich popis a pokud jde o planetu se základnou, zobrazí se uživatelské rozhraní s informacemi pro tuto planetu. Kliknutím na hvězdu se zobrazí informace

o aktuálním hvězdném systému. Kliknutím na červí díru se lze přepínat mezi hvězdnými systémy hry.

Menu

Kliknutím na tlačítko MENU v levém horním rohu otevřete menu. Jeho položky vám umožňují zobrazit seznam všech vašich lodí (položka LODĚ), seznam planet se základnou (položka ZÁKLADNY), mapu hvězdných systémů a jejich propojení (položka MAPA) a svůj uživatelský profil včetně nastavení (položka PROFIL). Nad tlačítkem MENU také můžete pomocí tlačítka Reference zobrazit autory podílející se na projektu, zdroje obrázků a použité komponenty třetích stran.

Detail základny

Kliknutím na planetu se základnou nebo na základnu ze seznamu základen se zobrazí detail základny. Zde se nachází tři tlačítka, která po kliknutí zobrazují různé informace.

Souhrn ukazuje, jak je na planetě drahé palivo a opravy.

Lodě ukáže seznam lodí nacházejících se na této planetě. Také zobrazí tlačítko **Koupit loď**, které otevírá okno pro nákup lodě. V tomto okně se nachází seznam všech dostupných lodí a tlačítkem Koupit lze tuto loď na planetě zakoupit. Po zakoupení lodi klikněte na tlačítko Lodě, loď by se měla zobrazit v seznamu.

Zboží zobrazí seznam veškerého zboží, které tato planeta nabízí a vykupuje. U zboží se zobrazuje cena za kus, počet kusů, celková cena a další informace.

Detail lodi

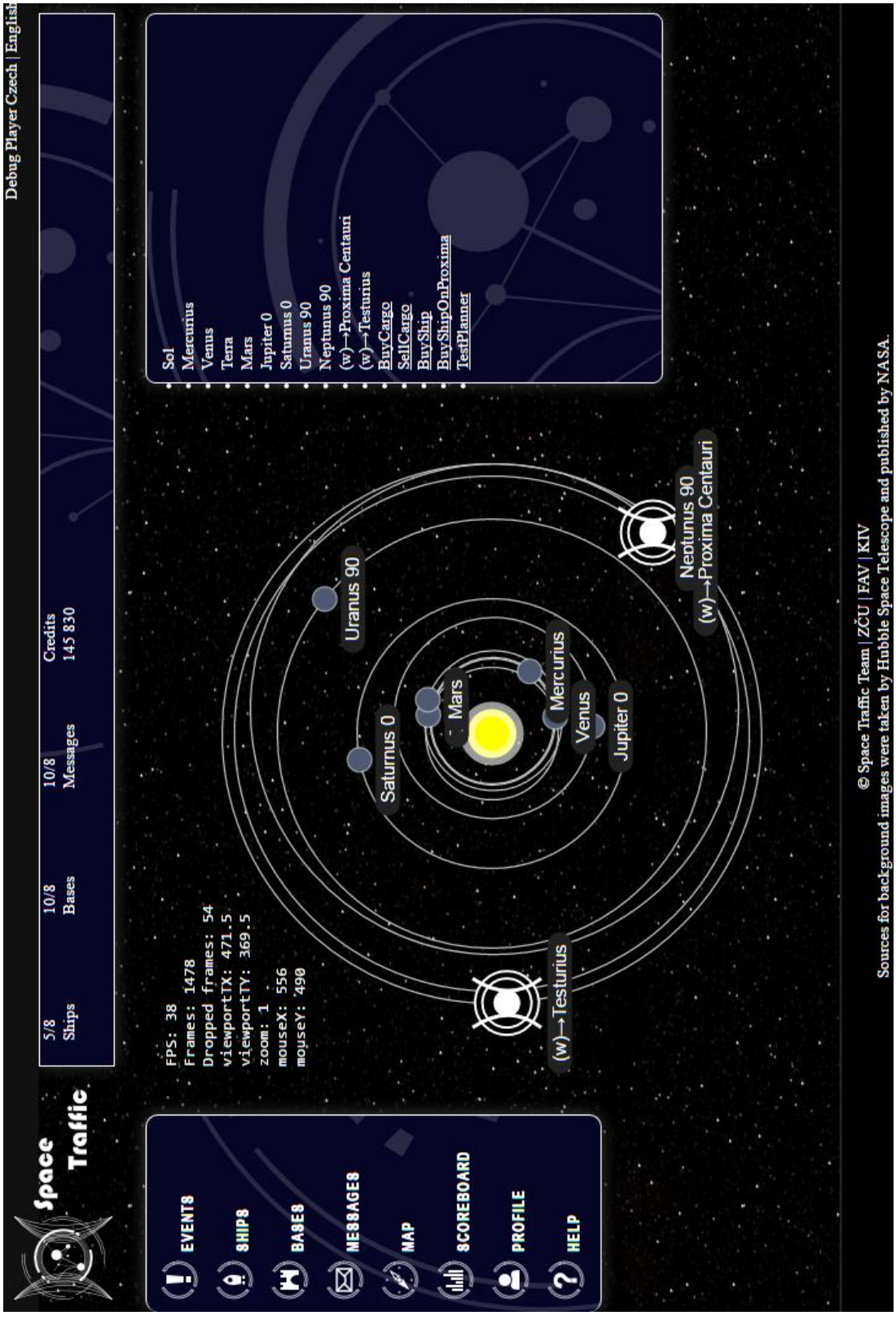
Kliknutím na loď v jednom ze seznamů lodí zobrazí její detail. Je zde zobrazena loď a všechny informace o ní. Pokud se loď nachází na základně, jsou zde tlačítka pro tankování a opravu lodě, pomocí kterých lze doplnit palivo nebo opravit loď, aby byla schopna letu. Také jsou zde tlačítka **Nakup zboží** a **Prodej zboží**. Pomocí nich lze nakoupit zboží a naložit ho na tuto loď. Stejně tak jde zboží z lodi prodat prodejci za cenu na aktuální planetě. Zboží, které loď převáží, lze zobrazit tlačítkem **Náklad**.

V detailu lodi se také nachází tlačítko **Leť na planetu…**, kterým lze lodi přikázat přelet mezi planetami. Kliknutím na něj se zobrazí plánovací okno a mapa se přepne do plánovacího módu. Kliknutím do mapy zvolte planetu, na kterou chcete letět. V případě, že se nachází v jiném systému, přepněte se do něj přes červí díry, které má loď použít k přeletu. Po volbě planety se zobrazí tlačítka **Start a Zrušit**. Pomocí tlačítka Zrušit můžete začít plánovat znovu. Tlačítkem Start zapříčiníte start lodi. Loď letí na planetu určitou dobu. Její stav můžete sledovat v detailu lodi. Je ale nutné tento stav ručně obnovovat.

Příklad postupu ve hře

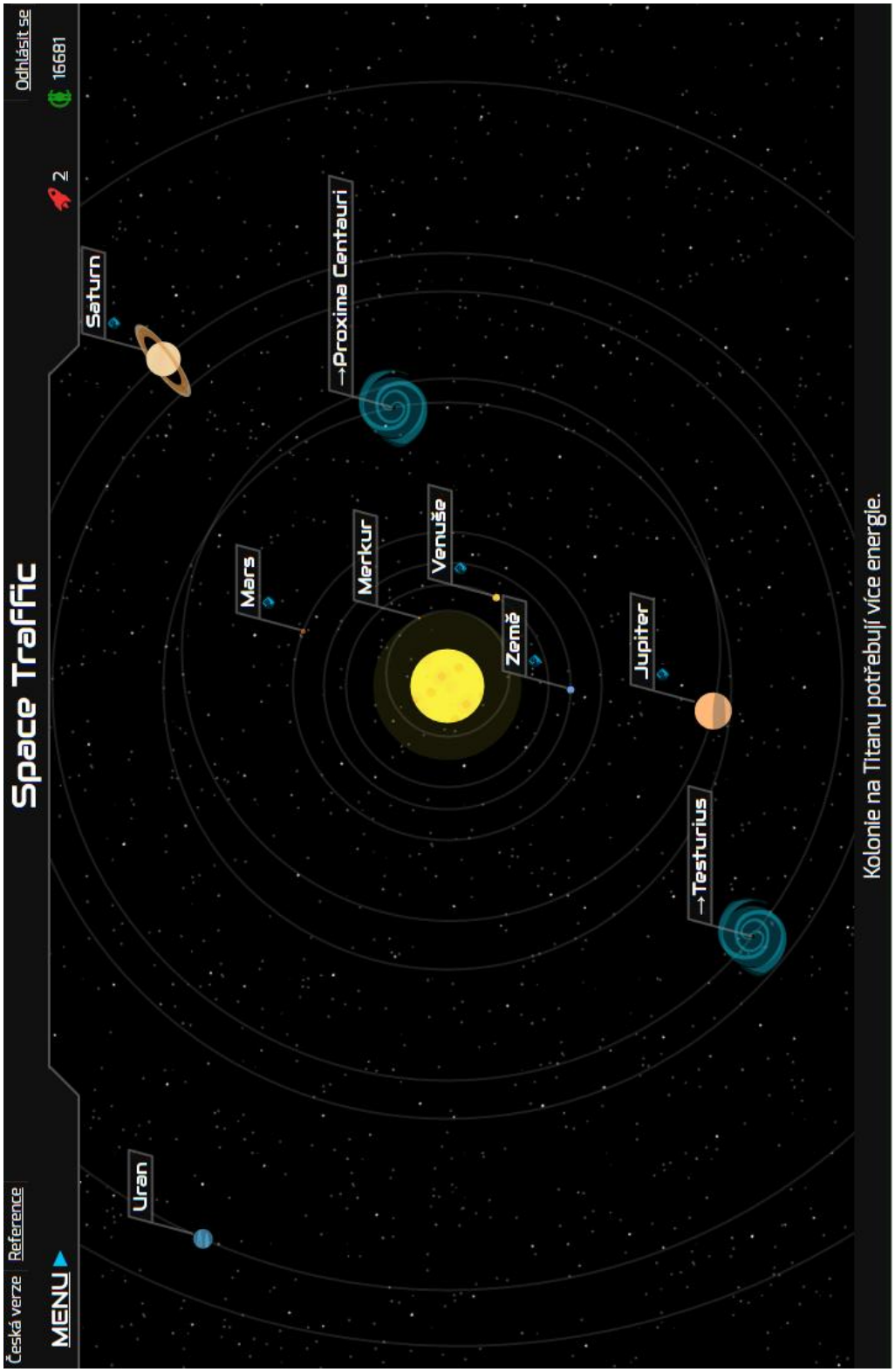
1. Registrujeme se, přihlásíme se
2. Klikneme na planetu se základnou
3. V detailu základny klikneme na tlačítko Zboží
4. Proklikáním ostatních planet najdeme zboží, které chceme převážet (tj. takové, které je na dvou různých planetách a má na jedné z nich vyšší cenu)
5. Klikneme na planetu, kde je zboží levnější
6. V detailu základny klikneme na tlačítko Lodě
7. Klikneme na tlačítko Koupit loď
8. Vybereme například první loď ze seznamu a klikneme na tlačítko Koupit
9. Klikneme na tlačítko LODĚ v menu, nebo na ikonu lodí v pravém horním rohu
10. Vybereme naši loď ze seznamu
11. Klikneme na tlačítko Nakup zboží
12. U zboží, které jsme si vybrali dříve, klikneme na tlačítko koupit maximum
13. V detailu lodi klikneme na tlačítko Leť na planetu…
14. Klikneme na planetu, kde je námi vybrané zboží dražší
15. Pokud se planety na mapě přibližují, můžeme počkat, až budou blíže, jejich vzdálenost ve hře totiž hraje roli
16. Klikneme na tlačítko Start
17. Najdeme loď v seznamu, zobrazíme její detail a jednou za čas klikneme na tlačítko se zatočenou šipkou pro obnovení
18. Až loď dorazí na zvolenou planetu, klikneme na Prodej zboží
19. U námi vybraného zboží klikneme na Prodat vše
20. Nyní jsme vydělali kredity! Pokračujeme ve hře

Příloha B – původní vzhled hry



Zobrazení hry Space Traffic v prohlížeči k 23.5.2015

Příloha C – nový vzhled hry



Zobrazení hry Space Traffic v prohlížeči k 18.4.2016