UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Tomáš Zachoval

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Tvorba virtuálního řídícího panelu

Tomáš Zachoval

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Tomáš Zachoval**
Osobní číslo:	**I14204**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Tvorba virtuálního řídícího panelu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem bakalářské práce je vytvořit integrované vstupně výstupní zařízení, na kterém bude možné prezentovat další směřování vývoje centrálních řídících panelů.

Vlastní případová studie vytvoří nadstavbu pro počítačovou hru: simulaci hvězdné válečné lodi. Na hlavním panelu, který bude odpovídat vzhledu panelů ze sci-fi filmů bude možné nastavovat základní charakteristiky lodi: hospodařit s energií, řídit systém letu, štítů, střelby, senzorů atp.

V rámci teoretické práce se student zaměří na vyhledání obdobných řešení a na tvorbu game conceptu a story boardu pro nejméně dva scénáře.

Rešerše a vlastní jednoduchý demonstrátor by měly sloužit k prozkoumání celého odvětví a umožnit budoucí komerční rozvoj řešení. Student se v rešerši zaměří především na technologie, které hardwarově umožňují spojit vstupní i výstupní zařízení do jednoho celku a to při odpovídajícím vypovídací hodnotě.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

DILLE, Flint. The ultimate guide to video game writing and design. New York: Watson-Guptill Publications, 2007. ISBN 158065066X.
PECINOVSKÝ, Rudolf. OOP: Naučte se myslet a programovat objektově. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2126-9.
KNUTH, D. E.: Umění programování - Základní algoritmy, Brno, Computer Press 2008, ISBN: 978-80-251-2025-5.
WRÓBLEWSKI, Piotr. Algoritmy: datové struktury a programovací techniky. Vyd. 1. Překlad Marek Michalek, Bogdan Kiszka. Brno: Computer Press, 2004, 351 s. ISBN 80-251-0343-9.
KEOGH, Jim; DAVIDSON, Ken. Datové struktury bez předchozích znalostí : průvodce pro samouky. Vyd 1. Brno : Computer Press, 2006. 223 s. ISBN 80-251-0689-6.

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**

Termín odevzdání bakalářské práce: **12. května 2017**

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Zdeněk Šilar, Ph.D.
pověřený vedením katedry

V Pardubicích dne 31. března 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 10. 5. 2017 Tomáš Zachoval

PODĚKOVÁNÍ

Na tomto místě bych chtěl poděkovat všem, kteří mě po celou domu mého studia podporovali a pomáhali mi. Zejména hlavně své rodině a přátelům. Dále bych chtěl poděkovat panu Ing. Brožkovi za jeho vedení při tvorbě této práce. Speciální poděkování patří kolegovi Maděrovi za jeho ochotu a pomoc v průběhu celého studia. Děkuji moc všem.

ANOTACE

Bakalářská práce se zabývá technologickou syntézou, potřebnou pro výrobu řídícího panelu hvězdné válečné lodi. Využívá běžně dostupné technologie, kterými jsou dotykové panely, projekční zařízení, generické GUI. Cílem je předat čtenáři základní informace o těchto zařízeních. Dále je věnována pozornost práci s JavaFX a nástrojům, které byly použity pro vývoj grafické interface aplikace.

KLÍČOVÁ SLOVA

Dotykové panely, projekční zařízení, JavaFX, GUI, NetBeans, CSS, scénář

TITLE

Creating a virtual controll panel

ANNOTATION

The bachelor thesis deals with the technological synthesis required for the production of the control panel of the space battleship. It uses commonly available technologies such as touch panels, projection devices, generic guis. The goal is to provide readers with basic information about these devices. Furthermore, attention is paid to the work with javafx and the tools that were used to develop the graphical interface of the application.

KEYWORDS

touch foil, projection equipment, JavaFX, GUI, NetBeans, CSS, screenpaly

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

SEZNAM PŘÍLOH

SEZNAM ZKRATEK A ZNAČEK

API	Application programming interface
CD	Compact disk
CSS	Cascading Style Sheets
GUI	Graphic user interface (Grafický uživatelské rozhraní)
ICT	Information and Communication Technologies
IR	Infra red
LCD	Liquid crystal display
LED	Ligth emitting diode
MSAS	Multinational Space Agenci Security
OSS	Open-source (otevřená licence)
UI	User interface (uživatelské rozhraní)
USB	Universal serial bus
UML	Unified Modeling Language
UV	Ultraviolet

Typografické konvence

Typografické konvence jsou nastaveny takto: části textu v kurzívách značí odborné názvy tříd, metod, atributů či nástrojů. Tabulky a obrázky jsou číslovány dle jejich výskytu. Odkazy na použitou literaturu jsou číslovány dle jejich prvního využití v textu.

1 Úvod

Tato práce je založena na technologické syntéze hardware dotykových zařízení a aplikovaného softwarového vývoje. Cílem je vytvoření virtuálního ovládací panel vesmírné válečné lodi. Tento panel je v první řadě určen milovníkům science fiction, kteří si na vlastní kůži chtějí vyzkoušet pilotování a obsluhu vlastního vesmírného plavidla. Pro zvýšení efektu a prožitku z vesmírné simulace, je využita v dnešní době ne příliš známá, ale dobře dostupná technologie dotykových a projekčních zařízení. Tato technologie slouží jako vstupně výstupní zařízení, pro ovládání softwaru herního simulátoru, který je součástí práce. V rámci práce je pojem hra a simulace volně zaměnitelný.

Teoretická část se věnuje softwarovým technologiím, které byly použity pro vývoj simulace. V první řadě se věnuje programovacímu jazyku Java a jejímu grafickému rozhraní JavaFX, ve kterém je aplikace tvořena. Dále je zde věnována část dílčím technologiím, které byly vyžity. Za zmínku stojí grafické editory, které byly použity pro tvorbu a úpravu textur. Další část pojednává o dotykových technologiích, které by bylo možné využít pro interaktivní ovládání. Zde jsou části popisující projekční a dotykové fólie a infra rámy. Poslední část je věnována hře – simulační aplikaci, jejímu ovládání a způsobu jakým funguje.

Úkolem praktické části bakalářské práce je vytvořit funkční vesmírný simulátor, který obsahuje minimálně dva scénáře. Uživatel musí hospodařit s energií, kterou má za úkol rovnoměrně rozkládat do různých systémů lodi podle toho, zda se aktuálně ocitá po palbou nepřátelských lodí nebo se pouze přesouvá po herní ploše. V simulaci se nacházejí tři druhy štítů a více druhů zbraní. Doba jejich nabíjení a účinnost je přímo úměrná množství energie, které obsahují jednotlivých zbraňové systémy. Pro oživení a zpestření simulace, si uživatel může spustit jednotlivé scénáře, kde se nachází více druhů cizích lodí. Některé jsou přátelské, jiné mohou na hráče útočit. Úkolem uživatele je pak vyřadit nepřátelské lodi z provozu, aniž by byl zničen on sám.

2 Použité technologie a nástroje

Tato kapitola je věnována technologiím a softwaru potřebnému pro sestavení simulátoru. V první podkapitole jsou čtenáři popsány základní informace o programovacím jazyce Java. Další je věnována její knihovně Java FX, na které je postaveno GUI aplikace. Následuje část o vývojovém prostředí, jež bylo použito. Poslední část podává informace o softwarech, ve kterých byla vyvíjena či upravována grafická část simulátoru.

2.1 Java

„*Společnost Sun definuje programovací jazyk Java jako jednoduchý, objektově orientovaný, distribuovaný, robustní, bezpečný, nezávislý na architektuře, portabilní, interpretovaný, vysoce výkonný a vícevláknový.“* [1] Pro tvorbu aplikace by bylo možní použít i jiné jazyky například C++ nebo C#. Jazyk Java je však pro svoji jednoduchost a přenositelnost mnohem vhodnější než C++, naopak C# je jazyk mnohem obsáhlejší a byl by v tomto případě dle mého názoru i vhodnější. Ale pro sestrojení simulátoru jsem zvolil právě jazyk Java ve verzi Java 8, protože jsem s ním pracoval ve škole a jsem s ním dobře seznámen.

2.2 Java FX

JavaFX je open source softwarová struktura pro vývoj aplikací. Je považována za nástupce Swing pro vývoj grafického uživatelského prostředí na platformě Java. Knihovna JavaFX je veřejně přístupná v Java API. JavaFX obsahuje velké množství funkcí, což z ní dělá preferovanou volbu při vývoji klientsky náročných aplikací.

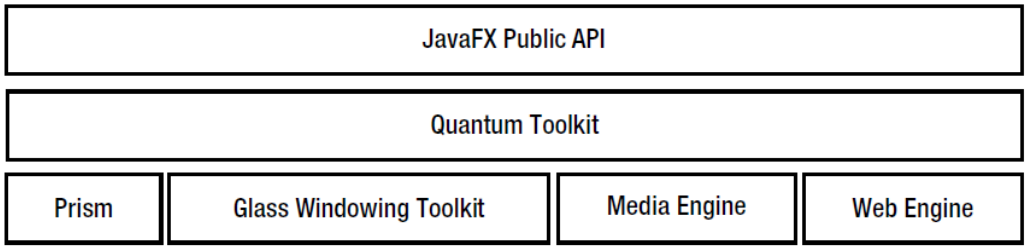
JavaFX je napsána v Javě, což jí umožňuje využívat všechny její výhody a funkce jako je například genericita, lambda výrazy, vícevláknové operace a další. Je možné využívat jakýkoliv editor či vývojové prostředí jako je kupříkladu NetBeans ke kompilaci debuggingem, a tvoření balíčků ve vaší aplikaci. Podporuje datovou vazbu napříč jejími knihovnami.

JavaFX nabízí dva způsoby, jak vytvořit uživatelské rozhraní. Prvním je pomocí kódu použitím FXML, což je značkovací jazyk založený na XML, definováním uživatelského rozhraní deklarativně. Druhou možností je použít nástroj zvaný *Scene Builder*, který je visuální editor pro práci s FXML, jenž je poskytován firmou Oracle.

Podporuje velké množství multimédií pro přehrávání zvuku či videa. Výhodou je navíc využívání většiny dostupných kodeků na této platformě. Další výhodu je, že JavaFX umožňuje vložit webový obsah do aplikace. V neposlední řadě je nutné zmínit, že podporuje vkládání

efektů a animací, jež jsou zejména velice důležité pro vývoj herních aplikací. Napsáním několika řádů kódu lze tak dosáhnou poměrně sofistikované animace. [2]

V pozadí JavaFX API se nachází celá řada komponent využívajících nativních knihoven a dostupného hardwaru a softwaru. Komponenty JavaFX jsou uveden na obrázku 1.



Obrázek 1: Komponenty JavaFX[1]

GUI v JavaFX je konstruováno jako scene graf. Je to kolekce vizuálních hierarchicky uspořádaných prvků zvaných uzle. Scene graf je postaven s použitím veřejného JavaFX API. Jeho uzly zpracovávají uživatelské vstupy a gesta. Mohou mít různé efekty, transformace a stavy. Dále obsahují jednoduché uživatelské ovládací prvky, jako jsou tlačítka, textová pole, dvoudimenzionální (2D) a vícedimenzionální (3D) tvary, obrázky, media (audio a video), webový obsah a grafy.

Prism je hardwarově akcelerované grafické potrubí používané pro vykreslování scene grafu. Pokud není k dispozici Java 2D slouží jako nouzový mechanismus pro vykreslování.

Glass Window Toolkit poskytuje grafické služby a služby pro vytváření oken, časovače a podobné. Používá k tomu nativní operační systém. Toolkit je také zodpovědný za řízení fronty událostí v JavaFX. Ty jsou řízeny jedním operujícím vláknem na systémové úrovni zvaném JavaFX Application Thread. Všechny vstupní uživatelské události jdou právě do tohoto vlákna.

Media Engine je odpovědný za poskytování podpory médií v JavaFX, například je to přehrávání audia nebo videa. Využívá kodeky dostupné na platformě. Používá samostatné vlákno pro zpracování těchto medií a využívá JavaFX Application Thread pro synchronizaci se scene grafem. Media Engine je založen na GSreamer, což je open source multimediální framework.

[1] Zdroj [2]

Web Engine zodpovědný za vkládání webového obsahu do scene grafu. J založen na WebKit, což je open source webový prohlížeč. Využívá HTML5, kaskádové styly, Java script a Document Object Model.

Quantum toolkit je abstrakce pro komponenty. Usnadňuje tak koordinaci se všemi výše zmíněnými složkami na nižší úrovni. [2]

2.3 Kaskádové styly

„*CSS vzniklo někdy kolem roku 1997. Je to kolekce metod pro grafickou úpravu webových stránek. Ta zkratka znamená Cascading Style Sheets, česky "kaskádové styly". Kaskádové, protože se na sebe mohou vrstvit definice stylu, ale platí jenom ta poslední.“* [3]

Kaskádové styly neboli CSS slouží pro nastavení stylů zobrazení elementů napsaných v jazycích HTML, XHML či XML a FXML. V tomto případě byl využit pro stylování tlačítek, slideru a progres baru aplikace. Příklad kódu pro nastavení stylu tlačítka je uveden v příloze E.

2.4 NetBeans

Prostředí, které bylo zvoleno pro vývoj aplikace, je NetBeans IDE ve verzi 8.1. Primárně je určené pro vývoj aplikací v jazyce Java, nicméně umožňuje i programování v dalších jazycích, například PHP, C, C++, Javascript. Prostředí je vytvořeno v jazyce Java, což ho činní multiplatformním a lze ho tedy spustit na většině operačních systémech jako je Windows, Linux, Mac OS, a další. Prostředí je částečně vyvíjené komunitou a částečně firmou Oracle, díky čemuž jsou neustále vytvářeny nové zásuvné moduly s rozšiřujícími funkcemi a uživatelům je poskytnuta vysoká podpora. [3]

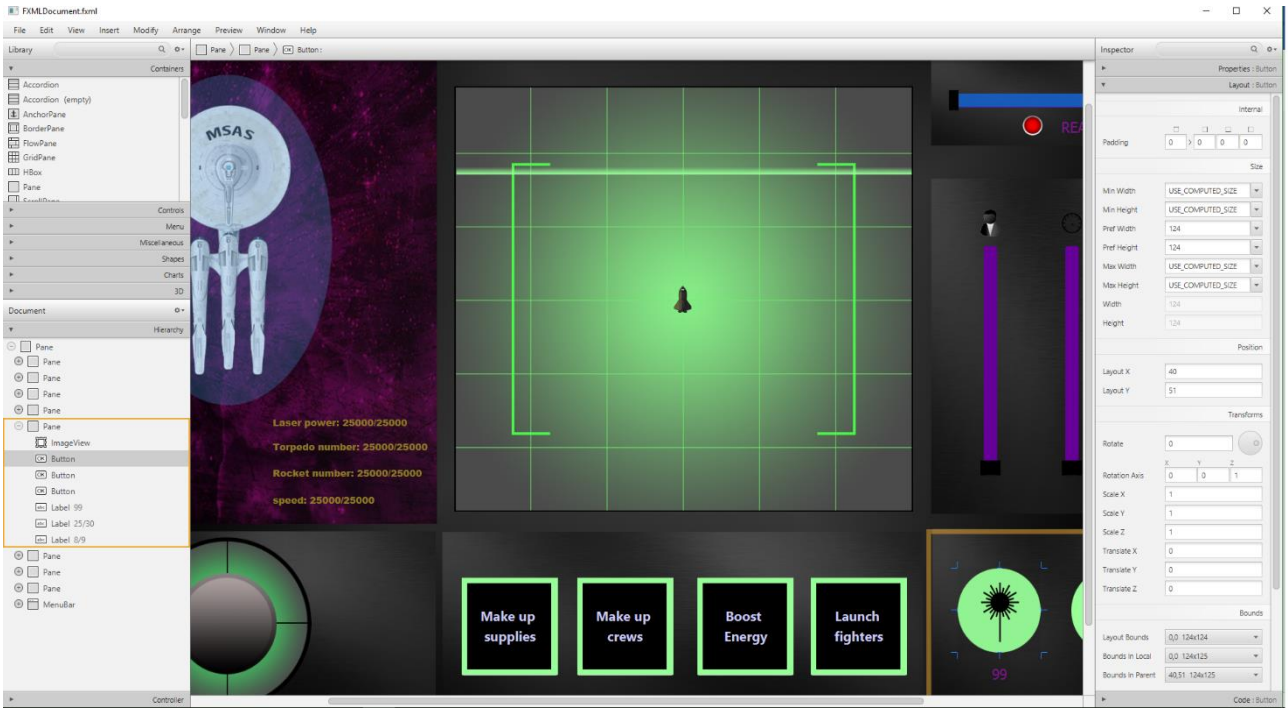
Pro srovnání s konkurencí je vhodné zmínit Eclipse. Toto vývojové prostředí je podobně jako NetBeans v open source licenci. Základním rozdílem je že Eclipse má v základní verzi pouze nejnutnější prvky. Další nástroje, například pro tvorbu GUI jsou ve formě pluginu. To může být pro některé uživatele výhodnější. Pro tuto práci nebyl zvolen právě kvůli již zmíněným pluginům. Jejich instalace a dohledávání je určeno uživatelům, kteří se v nich vyznají a mají s tímto prostředím více zkušeností. Dalším konkurenčním vývojovým prostředím je například IntelliJ IDEA od firmy Jetbrains. To je oproti předchozím komerční, což byl hlavní důvod, proč nebylo použito. [4, 5]

2.5 JavaFX Scean Builder

„JavaFX je vizuální nástroj podporující technologii drag&drop pro vytváření grafického uživatelského rozhraní v JavaFX aplikacích bez potřeby psaní kódu. Uživatelé pouze „přetahují“ komponenty na pracovní plochu, upravují jejich vlastnosti a aplikují různé styly, zatímco se v pozadí generuje výsledný FXML kód, který lze samozřejmě upravovat. Manuální změny jsou ihned viditelné v návrhovém zobrazení. Scene Builder je napsána jako JavaFX aplikace a ukazuje tak sílu a eleganci právě této nové platformy, která je od vydání verze Java 8 součástí standardní Java knihovny. Nástroj je určen pouze pro vytváření aplikací postavených na JavaFX platformě, není tedy možné vyvíjet grafické rozhraní pro jiné jazyky.“ [6]

Jeho výhody jsou i v podpoře kaskádových stylů pro úpravu komponent, ty byly v simulátoru použity například při tvorbě stylu tlačítek.

Pokud by tento nástroj nebyl využit, bylo by nutné veškeré kódy psát ručně. Někomu tento způsob vytváření GUI JavaFX aplikace vyhovuje. Ale mnohem snazší a pro uživatele přívětivější, je využití právě tohoto nástroje. Použit byl ve verzi 2.0 a práci z grafickými komponenty bezpochybně urychlil. Na obrázku 2 je screen zachycující práci s tímto nástrojem. [7]



Obrázek 2: Rozhraní nástroje Scene Builder

2.6 Adobe Photoshop

Pro tvorbu a návrhu textur simulátoru byl využit grafický editor od firmy Adobe v bezplatné zkušební verzi Photoshop CC 2015. Používán byl zejména proto, že práce v něm je vyučována ve škole. Využita byla zkušební verze, která byla pro tuto práci zcela dostatečná. Druhým program, který byl využit, byl Adobe Ilustrátor CC 2015. Tento editor se specializuje zejména na úpravu a vytváření vektorové grafiky. Použit byl opět ve zkušební verzi.

Ze známé konkurence by bylo možné vybrat například Balíček grafických editorů Corel od firmy Corel Corporation. Nabízí opět více editorů. Pro úpravu rastrových obrázků je to Corel Photo Paint, a pro vektorovou grafiku je to Corel Draw. Editory od firmy Corel nabízí podobné funkce jako editory od firmy Adobe. Pokud by byl potřeba grafický editor na delší než vymezenou zkušení dobu, tak pro studenty je tento balíček vhodnější. Jejich hlavní výhodou je nižší cenová relace, za kterou jsou poskytovány.

2.7 Zdroje ikon

Ve hře byly využity volně licencované ikony z internetu. Obrázky, u kterých autor vyžaduje uvedení zdroje jsou následující:

Ikony cizích lodí: Icon made by Freepik from www.flaticon.com

Obrázek prasklého skla: http://gallery.yopriceville.com/Free-Clipart-Pictures/Decorative-Elements-PNG/Broken_Glass_Effect_Transparent_PNG_Clip_Art_Image#.WQSFl4jyiUl

3 Dotyková a projekční zařízení

Tato kapitola je věnována dotykovým zařízením a projekčním foliím. Zabývá historickým vývojem dotykových zařízení. Dále se zaměřuje na popis typů, parametrů a způsobu využití projekčních a dotykových zařízení.

3.1 Vznik a vývoj dotykových zařízení

O počátku vývoji dotykových zařízení je možné najít mnoho článků na internetu. Je to například článek ze serveru *zive.cz Dotyková technologie: začalo to před 50 lety*. [8] Nebo článek ze serveru automatizace.hw.cz zařízení *Současný stav vývoje rezistivních dotykových ploch / displejů*. [9] Celkovou historii však nejlépe shrnuje následující článek ze serveru Mobilnet.cz [10]:

„*Za skutečnými kořeny dotykových displejů však musíme ještě mnohem hlouběji do minulosti. Za praotce dotykových displejů je některými považován E. A. Johnson, který v krátkém článku popsal principy fungování kapacitních dotykových displejů již v roce 1965. V roce 1967 na to navázal delším článkem doplněným o diagramy a fotografiemi a v roce 1968 popsal použitelnost dotykové technologie při řízení letového provozu.*

Na počátku 70. let inženýr CERNu (Evropské organizace pro jaderný výzkum) Ben Stumpe za pomocí svého kolegy Franka Becka vyvinul transparentní dotykovou obrazovku, která je založena na Stumpově práci v televizním průmyslu na počátku 60. let. Jeho transparentní dotyková obrazovka vyrobená CERNem byla zařazena do užívání v roce 1973.

Na dotykové technologii se ve stejné době usilovně pracuje i na druhé straně Atlantiku. V roce 1971 přichází s prvním dotykovým senzorem doktor Sam Hurst, instruktor Univerzity v Kentucky. Senzor, který Hurst nazval Elograph, ještě nebyl transparentní jako moderní dotykové obrazovky, ale přesto to byl významný milník.

Hurst si na svém objevu staví své další podnikání a zakládá společnost Elographics. V roce 1974 přichází s první opravdovou dotykovou obrazovkou s transparentním povrchem a v roce 1977 si patentuje pětidrátovou rezistivní technologii, která se používá dodnes. První verze jeho rezistivních displejů se začínají vyrábět v roce 1982.“ [10]

Z článku vyplývá, že dotykové technologie nejsou technologií až tak novou. Ale až v dnešní době se ale staly cenově dostupnějšími, a to především díky novým způsobům výroby, což má za následek jejich rozšíření v komerční oblasti.

3.2 Technologie dotykových zařízení

Nejpoužívanějšími a veřejností neznámějšími dotykovými zařízeními jsou kapacitní a rezistivní displeje v mobilních telefonech. O fungování těchto displejů opět velmi dobře pojednává níže uvedený článek Pavla Škopka z mobilnet.cz.

„Rezistivní (odporové) displeje jsou vývojově starší technologií používanou v dotykových mobilních telefonech. V současnosti existují dva typy rezistivních displejů – čtyřvodičový a pětivodičový. Oba typy přitom fungují na stejném principu, ale liší se v odolnosti.

Panel rezistivního displeje se skládá z několika vrstev, které ohraničují skleněný panel a pružnou membránu na povrchu displeje. Skleněný panel i dotyková membrána jsou pokryty vodivými vrstvami – spodní elektrovodivou a horní odporovou, které od sebe odděluje pro oko neviditelná síť podpěr, mezi kterými je tenká vzduchová vrstva.

Obě vodivé vrstvy jsou připojeny k řídicímu a vyhodnocovacímu modulu. V okamžiku, kdy displej zapneme, začne spodní elektrovodivou vrstvou procházet elektrický proud. Jakmile se dotkneme displeje, membrána se prohne a horní odporová vrstva se spojí se spodní elektrovodivou, čímž mezi nimi začne procházet elektrický proud. Řídící a vyhodnocovací modul následně vyhodnotí polohu a velikost bodu dotyku.

Díky tomu, že rezistivní displeje mají pružnou svrchní vrstvu, je možné je ovládat téměř čímkoli a to je proti kapacitním displejům jejich hlavní výhoda. Vůbec nezáleží, jestli se displeje dotýkáte holým prstem, rukavicí, stylusem nebo jakýmkoli klacíkem. Další výhodou rezistivních displejů je pak nízká cena, nízká spotřeba energie, vysoká reakční rychlost a větší přesnost.Jejich hlavní nevýhodou je, že jejich průzračnost je pouze 80 procent. To, co nevadí u monochromatických dotykových terminálů, je u mobilních telefonů, kde je kladen na maximální věrnost barev, problém.

Kromě toho však jsou rezistivní displeje náchylné i na mechanické poškození svrchní pružné vrstvy. V případě čtyřvodičové odporové technologie, která je nejvíce rozšířená, se uvádí životnost displeje na pět milionů dotyků v jednom místě.“ [10]

Dalším typem jsou kapacitní displeje. Tento druh displeje popisuje Pavla Škopka z mobilnet.cz takto. „*Kapacitní dotykový displej pracuje s přirozenou vodivostí lidského těla. Skleněný panel, který je jinak izolant, je v tomto případě potažen tenkou transparentní vodivou vrstvou například indium tin oxidu (ITO).*

Jakmile se dotknete displeje, naruší se elektrostatické pole displeje a mezi jeho povrchem a špičkou prstu vznikne kapacita, která uzavře elektrický obvod. Tyto změny jsou měřitelné jako změny kapacitního odporu. Místo dotyku je určeno použitím různých technologií a lokace tohoto místa je odeslána řadiči, který jej dále zpracuje.

Právě nutnost použití prstu nebo jiného elektricky vodivého předmětu je hlavní nevýhodou kapacitních displejů. Stačí si vzít rukavice a už vám displej nebude fungovat. Dá se to řešit pomocí speciálních kapacitních stylusů nebo speciálně upravených rukavic, které mají ve špičkách prstů všitou plošku z vodivých vláken, která přenese proud mezi displejem a prstem.

Naopak hlavními výhodami kapacitních displejů je vysoké rozlišení, vysoká světelná propustnost, která dosahuje 93 procent +/- 2 procenta, nezávislost funkčnosti na prachu a mastnosti a obecně vysoká odolnost. Jestliže rezistentní displej ve čtyřvodičové variantě měl životnost pět milionů dotyků, tak v případě kapacitních displejů je to více než 300 milionů dotyků v jednom místě. Podtypem kapacitních displejů jsou pak takzvané projekční kapacitní displeje, které vyzařují elektrické pole do prostoru. Můžete je například překrýt nevodivým odolným sklem a ještě více tak zvýšit odolnost displeje nebo dokonce ovládat telefon, aniž byste se displeje přímo dotkli." [10]

Další technologie používající infračervené záření za využití infračervených diod, se používá například u infra rámů. Tato technologii se věnuje samostatná podkapitola. Poslední technologií, kterou je vhodné zmínit je SAW. Využívá ultrazvukové vlny přecházející přes displej. Po dotyku je tato vrstva narušena a řadič z informací o přerušení vypočítává místo dotyku.

3.3 Dotykové fólie

Dotykové fólie jsou tvořeny z tenkých lepících akrylových listů. Tato technologie se používá pro přeměnu obyčejných skleněných ploch na dotyková zařízení. Vzhledem k tomu, že nejsou příliš citlivá na světlo, mohou být instalována i na místech, kde je umělé osvětlení nebo velmi silné sluneční světlo. Interaktivní fólie mohou být použity na jakémkoliv nekovovém povrchu, protože fungují pouze s nevodivými materiály. Jsou flexibilní, čili je možné je aplikovat i na zakřivené povrchy.

Technologie interaktivní dotykové fólie je určena pro práci přes skleněnou vrstvu, a to až do výšky 24 mm. Fólie je aplikována na vnitřní straně skleněné plochy. Následuje ji fólie projekční, na kterou je pomocí systému zadní projekce promítán obraz projektorem. Obraz a zařízení jsou tedy umístěny na druhé straně skla než uživatel. To znamená, že senzory citlivé

na dotek jsou plně chráněny a uživatel nepřichází do kontaktu s elektronickým zařízením, což dělá zařízení plně bezpečné.

Odezva fólie se pohybuje kolem 18–50 ms (pro porovnání odezva dnešních moderních monitorů dosahuje asi 5 ms), její přesnost je 1–2 mm u menších velikostí a 8-10 mm u velkých velikostí. Jsou k dispozici ve verzi 2, 10, 20 nebo 100 souběžnými dotykovými body. Mohou být také opatřeny ochranou proti UV záření pro prodloužení dlouhodobé výkonosti ve venkovních instalacích. Folie pro snímání dotyku využívají technologii kapacitního dotykového panelu. Senzor měří změny v kapacitním odporu, zasílá je řadiči, který tyto informace zpracovává a určuje polohu. Interaktivní fólie jsou standardně k dispozici ve velikostech úhlopříčky od 30 palců do 100 placů. Společnosti zabývající se výrobou a prodejem těchto zařízení nabízejí také možnost vytvoření vlastní velikosti fólie. Příklad dotykové fólie je zobrazen na obrázku 3. [11]

Obrázek 3: Dotyková fólie UGO! Touch[2]

3.4 IR rámy

Infrarámy (nebo zkráceně IR rámy) jsou zařízení, které se připevní na LED nebo LCD obrazovku a promění ji v plně funkční dotykové zařízení. Díky technologii Plug & Play je připojení pro uživatele velmi jednoduché. Jsou k dispozici v mnoha různých provedeních a velikostech v úhlopříčkách od 32 palců do 1. Na trhu je možné se setkat s 2, 6, 12 a 32 souběžnými dotykovými body.

Rámy jsou opatřeny IR LED buňkami, které jsou schopné rozpoznat velikosti tvarů a objektů a tím jsou schopny jim přiřadit různé funkce. Užitečné je to zejména pro to, aby se zabránilo

[2] Zdroj [12]

interferenci s dlaní ruky, a tím nechtěnému dotyku. Napájení a komunikaci zajišťuje USB sběrnice, čímž je také zajištěna jednoduchá instalace. Využívána je technologie infračerveného záření. Spočívá v rozmístění diod a detektorů po okraji rámu, místo dotyku je vypočítáno z narušených struktur paprsků.

Využití IR rámu tedy spočívá především v jednoduché a nepříliš finančně náročné modifikaci z obyčejné obrazovky na dotykovou. [13]

3.5 Projekční folie a jejich typy

Projekční fólie slouží jako prostor, kam je obraz promítán. Dále mají za úkol zlepšit viditelnost obrazu pomocí reflexní vrstvy. Na trhu se můžeme setkat se dvěma typy. Projekční fólie se zadní projekcí a projekční fólie s přední projekcí.

Prvním typem je přední projekční fólie. Zde je projektor klasicky umístěn před folií. Tato fólie poskytuje vysoký kontrast a svítivost, proto jsou vhodné pro používání v osvětleném prostředí. Jsou totiž vyrobeny z více vrstev reflexního materiálu, a tedy mnohem lépe odráží světlo vyzařované projektorem. Jsou jakousi alternativou pro projekční plátna a nátěry. Jejich výhoda spočívá především ve snadné aplikaci a možnosti kreativního využití, kdy je možné je upravit do různých tvarů. Využití tohoto typu je především v situacích, kdy uživatel nezasahuje přímo do promítaného obrazu. Například v přednáškových síních, na veletrzích a v interiéru budov. Hlavní výhodou přední projekce je její mobilita a nenáročnost na přípravu.

Druhým typem je fólie se zadní projekcí. Projektor je umístěn za projekční fólií, která je umístěna na rubové straně nosné konstrukce (typově skla). Tyto fólie se vyrábí z kontrastního materiálu, čímž zajišťují i viditelnost při velkém světle. Fólie jsou ve většině případů vybaveny samolepící vrstvou. Díky tomu je možné je snadno nalepit na sklo či plexisklo. Hlavní výhoda zadní projekce spočívá v tom, že uživatel se může přiblížit k projekčnímu zařízení, aniž by narušil obraz. Ve spojení s dotykovou fólií tak může tvořit velmi uživatelsky přívětivé dotykové zařízení s mnohostranným využitím. V dnešní době je možné se s nimi setkat poměrně často jako s informačními tabulemi v obchodních centrech, interaktivními informačními centry ve městech. Jsou použitelné v barech a restauracích, protože fólie jsou chráněny sklem, a tudíž jsou chráněny proti nechtěnému přístupu kapalin. Dále je možné tyto fólie využít jako různé reklamní poutače reagující na dotek. V kombinaci s rozšířenou realitou je možné využití jako zkušební kabinky v obchodech s oblečením bez nutnosti si oděv opravdu oblékat. Proto je také tato kombinace nejlepším řešením pro interaktivní ovladač simulátoru této práce.

4 Vlastní hra

Kapitola Vlastní hra se zabývá popisem hry, všemi herními atributy a komponenty. V první podkapitole je uveden kontext. Následuje popis komponent. Čtenář se v této části dozví veškeré principy o fungování vlastní hry. Další podkapitola se věnuje scénářům a v poslední je popsán způsob, jakým je hra tvořena z programového hlediska.

4.1 Popis a kontext

Scénář hry se nachází ve 200 let vzdálené budoucnosti, kdy lidstvo vynalezlo dostatečně silné energetické generátory a motory pro cestování ve vesmíru.

Díky této převratné novince je možné těžit suroviny z různých vesmírných těles. Tímto se začala psát nová epocha lidstva. Pomocí nově nabytých surovin nastala další technologická revoluce a lidstvo začalo expandovat právě do vesmíru. Byly vystavěny orbitální stanice u těžebních center napříč celou naší Galaxií. Vznikaly nové korporace, firmy i jednotlivci, jež se zabývali těžbou a dopravou surovin nebo transportem lidí samotných. Naneštěstí se však objevila i různá společenství, jež se chtěla na těchto poctivě vydělávajících osobách snadno obohatit. Čímž vznikl fenomén vesmírného pirátství.

Z těchto důvodů musela být založena nová nadnárodní bezpečnostní agentura pro ochranu vesmírného prostoru MSAS (Multinational Space Agenci Security), která má za úkol pátrat ve vesmíru po pirátských lodích, zajistit ochranu poctivých lidí, hájit bezpečnost a pořádek v naší Galaxii.

V této hře se hráč chopí kapitánského řídícího panelu jedné z lodí MSAS. Vydává se do vesmíru pátrat po nepřátelských plavidlech a trestat je dle zákona. K tomu využívá početný arzenál lodi a zkušenou posádku, u které musí dbát o její bezpečnost.

4.2 Základní komponenty

V této podkapitole jsou popsány všechny základní objekty, se kterými je možné se ve hře setkat. Komponenty jsou popsány nejdříve v kontextu hry a následně z pohledu programové analýzy.

4.2.1 Vlastní loď

Vlastní loď má více funkcí a atributů například velikosti posádky, integrita trupu, maximální integrita štítů a podobné. Proto je jí podrobněji věnována celá podkapitola 4.6 Vlastní loď.

4.2.2 Cizí lodě

Cizí lodě představují základní objekty, se kterými je možné se v simulátoru setkat. Plní zde především roli soupeře. Ovládány mohou být umělou inteligencí nebo prvky náhody.

V kontextu hry jsou cizími loděmi všechny lodě, na které je možné v simulátoru vesmíru narazit. V základu jsou děleny na přátelské a nepřátelské. Dále jsou děleny dle typu. Nejsilnější bojové lodě vybavené všemi typy zbraní jsou *BattleShip*. Menší raketové lodě vybavené naváděnými rakety jsou typu *RocketShip*. Lodě vybavené torpédy a výkonnými lasery se nazývají *AtackShip* neboli útočná loď a posledním speciálním typem lodě je *CargoShip.* Tato loď není vybavena zbraněmi. Jedná se pouze o nákladní loď, která prolétá vesmírem a na loď ovládanou uživatelem neútočí.

V samostatném programu jsou cizí lodi tvořené vlastními třídami *BattleShip, AtackShip, RocketShip* a *CargoShip.* Jsou umístěny v balíčku *ship* a zděděny ze třídy *ForeginShip,* která je potomkem abstraktní třídy *SpaceShip.* Z této třídy je dále zděděna třída pro *MyShip,* která vytváří instanci vlastní lodě. Abstraktní třída *SpaceShip,* tedy obsahuje všechny společné atributy těchto tříd. V konstruktoru této třídy jsou nastaveny pouze hodnoty pro souřadnice tedy atribut *CoordX* a *CoordY,* dále pak hodnota *true* u atributu live což značí nezničenost lodi. Ve třídě *ForeginShip* jsou přidány další společné atributy pro jednotlivé lodi. Ty jsou odlišně nastaveny v konstruktoru jednotlivých tříd lodí. Nastavení a popis atributů všech typů lodí je možné vidět v následující tabulce 1.

Tabulka 1: Nastavení atributů cizích lodí

Název atributu	Typ atributu	Popis atributu	Nastavení atributu dle třídy lodě			
			BattleShip	*RocketShip*	*AtackShip*	*CargoShip*
shipHull	integer	Maximální integrita trupu lodi.	3500	1500	1500	1000
Shield	integer	Maximální kapacita štítů.	1500	600	2000	500
Crue	integer	Maximální množství posádky.	1000	200	200	100
capacity Rocket	integer	Maximální kapacita raket.	100	200	0	0
capacity Torpedo	integer	Maximální kapacita torpéd.	200	0	300	0
powerLaser	integer	Síla laseru.	90	60	70	0

Friendly	boolean	Hodnota určuje, zda je loď přátelská.	false	False	false	true
TypeOfShip	enum	Určuje typ lodě.	BattleShip	RocketShip	AtackShip	CargoShip

Atributy z tabulky 1 jsou základními charakteristikami cizích lodí. Dále se zde nacházejí další atributy. Atribut *picture* pro vykreslení ikony lodě, *typeWeap* uchovávající hodnotu aktuálně používané zbraně, *timeOfReloading* drží hodnotu nabytí zbraně a atribut *reloaded*, který určuje, zdali je zbraň nabitá.

4.2.3 Zbraně

Zbraně se v simulátoru nachází z důvodu interakce s okolním prostředím a především s loděmi cizími. Bez nich by nebylo možné plnit hlavní scénáře simulátoru.

V kontextu hry se je možné se setkat se třemi typy zbraní. Prvním je Laser. Cizí lodi mají jeho sílu staticky nastavenou, u lodi vlastní je proměnlivá. Určuje jej množství přidělené energie. Jeho dostřel je nastaven konstantě. Tato zbraň je přizpůsobena boji na krátkou vzdálenost, její dosah je tedy pouze tři sta jednotek. Druhým typem zbraně je torpédo. Je to velice účinná útočná zbraň, která se nabíjí pomaleji než laser. U vlastní lodě je pro použití potřeba středně velké množství energie pro zahájení palby. Nevýhodou torpéda je schopnost pouze přímého letu, jeho dosah je tisíc jednotek. Poslední zbraní, kterou je možné ve hře použít, je raketa. Jedná se o nejsilnější, ale také o energeticky nejnáročnější typ zbraně, její nabíjení trvá nejdéle. Po vystřelení je automaticky navedena na označený cíl. Dosah této zbraně je u lodi vlastní staticky nastaven na tisíc pět set jednotek.

Z pohledu programové analýzy jsou zbraně tvořeny atributem lodi. Jejich síla a dostřel je konstantně nastaven. U vlastní lodi jsou tvořeny konstantně ve třídě *GameLogick.*

4.2.4 Stíhače

Stíhače jsou zde podobně jako zbraně pro interakci a oživení herního prostoru. Z obecného pohledu jsou brány spíše jako zbraň.

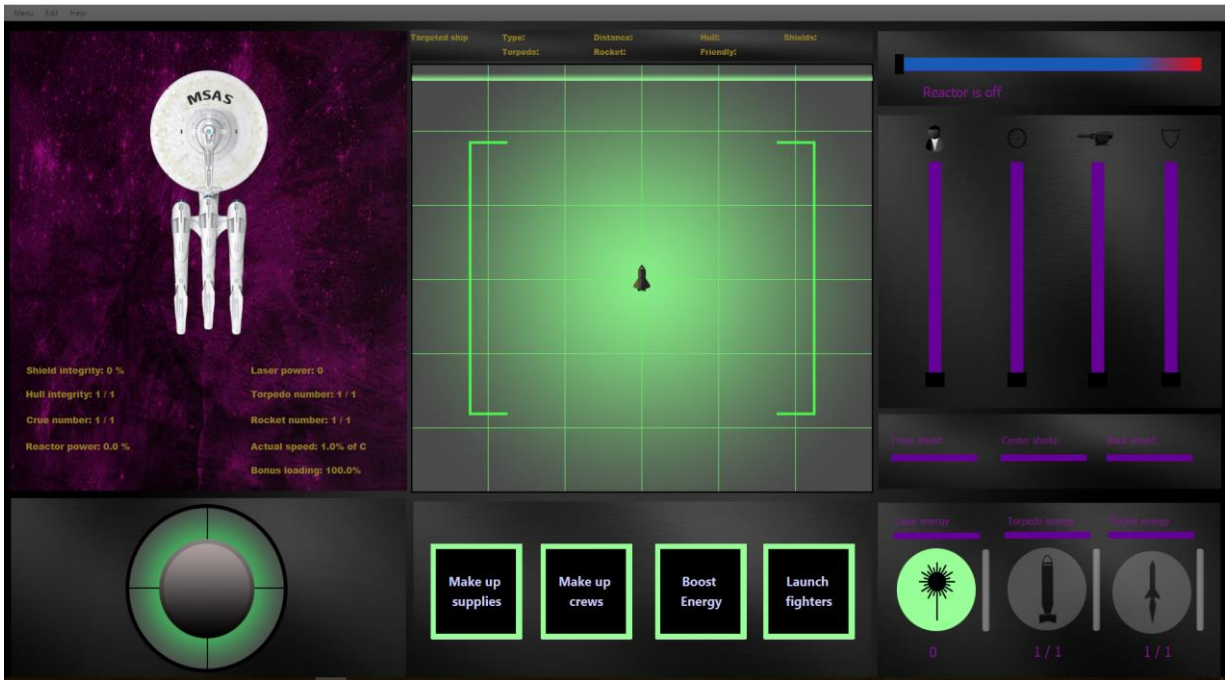
V kontextu hry je stíhač speciálním typem lodě, který je vázán na svou mateřskou loď. Je vybaven lasery a může nést malé množství raket. Jeho doba působení je omezena kapacitou paliva. Při jeho nedostatku se vrací na svou mateřskou loď. Vyvolávají se pomocí bonusů a jejich útok je veden na všechny nepřátelské lodi v dosahu.

Z programového hlediska jsou tvořeny pouze metodou pro zásah všech lodí v určitém dosahu.

4.3 Grafické uživatelské rozhraní

V této podkapitole je popsána grafická interface hry. Postupně jsou zde popsány všechny její části. Po spuštění se uživateli zobrazí celý panel lodi. V tomto módu je plně interaktivní, pouze je vypnut radar a ve vesmíru se nenachází žádné lodi. Základní ovládání se provádí pomocí menu baru, který je popsán níže.

Na obrázku č. 4 je vidět kompletní snímek obrazovky panelu. V horní části je se nachází menu bar, v něm se na první pozici nachází položka menu. Po kliknutí na tuto položku se zobrazí další dvě položky. První je *simulation,* pomocí které je možné zapnout odlišné scénáře hry a tím aktivovat hru samotou. Druhá položka *close* simulaci ukončí. Další položkou je možné přepínat simulaci do plného rozlišení, kdy není zobrazeno okno a zpět. Po kliknutí na poslední položku se zobrazí informace o simulaci a autorovi.

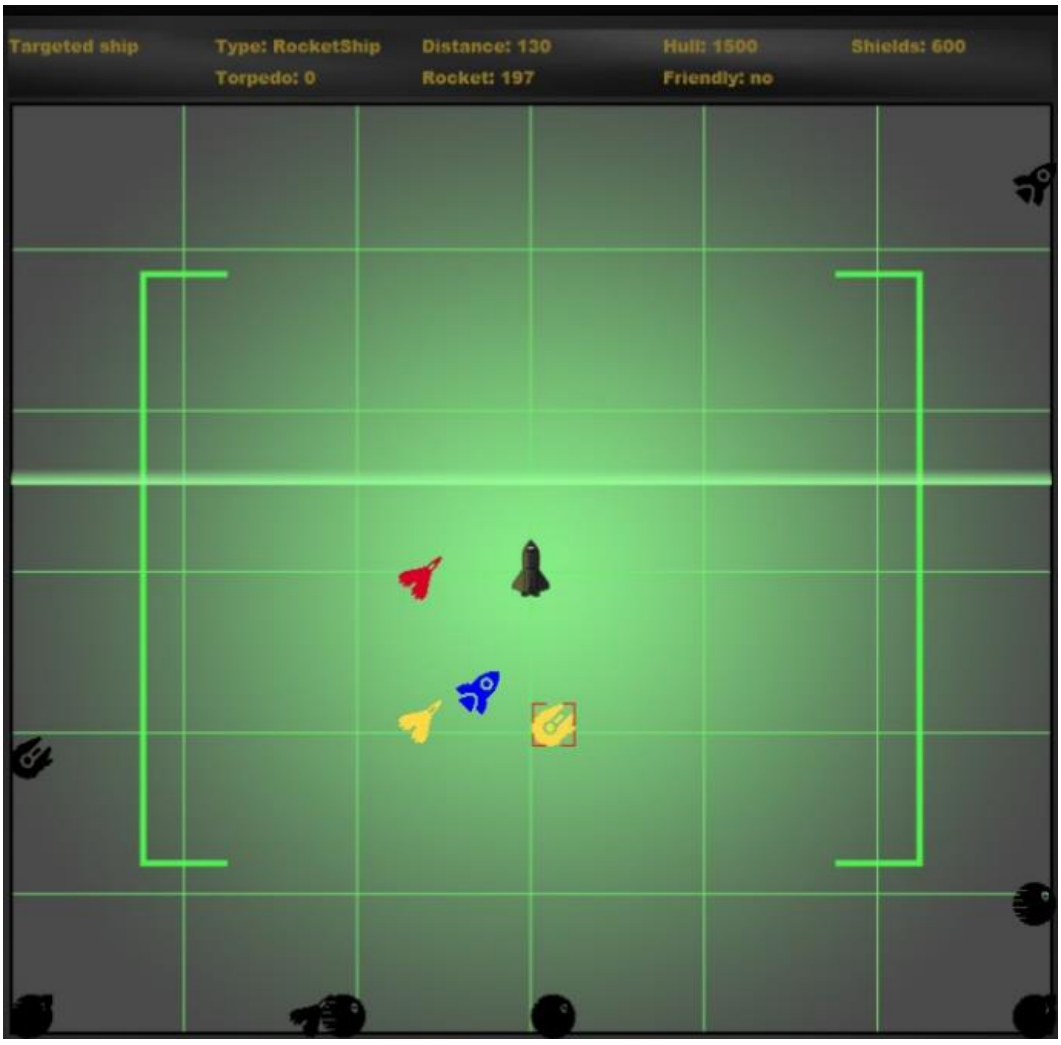


Obrázek 4: Snímek celého panelu

Na obrázku č. 5 je informační část panelu. Zde jsou zobrazeny všechny stavy lodě. Na levé straně nahoře se nachází informace o síle štítů, které jsou také vizuálně zobrazeny kolem lodi. Čím méně průhledný štít je, tím více energie se v daném štítu nachází. Pod štíty je vypsána integrita trupu, následuje výpis počtu posádky a aktuálního výkon reaktoru. Na pravé straně jsou zobrazeny stavy zbraní, kdy první hodnota ukazuje aktuální stav a druhá hodnota maximální stav. Na druhém řádku je tedy vidět stav torpéd a na třetím je zobrazen stav raket. Dále je v procentech vypsána rychlost lodi. Stoprocentní rychlost značí rychlost světla. Poslední informace ukazuje aktuální stav nabytí bonusů, které je možné ve hře využívat.

Obrázek 5: Stav lodi

Obrázek č. 6 je radar. Úkolem této komponenty je v první řadě zobrazovat směr, kterým se loď ve vesmíru plaví. Druhým úkolem radaru je zobrazovat okolní lodě. Pokud jsou lodě mimo viditelné pole, jsou zobrazeny u kraje radaru. Pokud je ikona lodě černá jedná se o loď, která na uživatelskou loď nereaguje. Žlutá ikona lodi značí, že daná loď nabíjí své zbraně. Červenou barvou je následně zobrazen výstřel lodi. V případě že ikona zmodrá jedná se o loď přátelskou, která na hráče neútočí, ale pouze proplouvá kolem. Po kliknutí na nepřátelskou loď je možné ji zaměřit. Pokud je tak učiněno, zobrazí se kolem lodi zaměřovací rámeček. Na horním panelu se následně vypíší informace o zaměřené lodi. V druhé textové položce je vypsán typ lodi, třetí položka zobrazuje aktuální vzdálenost od vlastní lodi. Ve čtvrté položce je zobrazena integrita trupu a v poslední integrita štítů cizí lodi. Na dalším řádku jsou zobrazeny informace o počtu torpéd, raket a informace o tom zdali je loď přátelská.

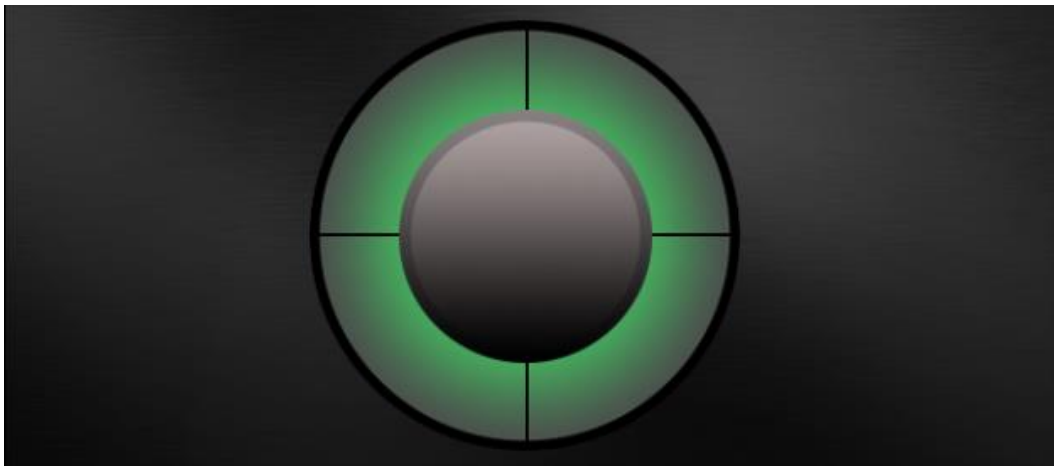


Obrázek 6: Radar

Na obrázku č. 7 je panel pro ovládání hlavních sytému lodi. Pomocí posuvníků se nastavuje výkon jednotlivých systémů. Výkon se následně projevuje i na ukazateli reaktoru, jenž je vidět v horní čísti obrázku. První posuvník určuje systémy pro posádku, druhý rychlost lodi, třetí posuvník přesměrovává energii do zbraňových systémů. Posledním posuvníkem se ovládá integrita štítu. Při přetížení reaktoru se spustí červená kontrolka a informační text. Ve spodní části se nastavuje energie pro jednotlivé části štítu. První posuvník určuje, kolik energie pro štíty putuje do předního štítu, druhý množství energie ve středovém štítu. A poslední v zadním štítu.

Obrázek 7: Ovládací systémy

Obrázek č. 8 znázorňuje joystick pro ovládání směru vesmírného plavidla. Kliknutím a přetáhnutím vnitřního kolečka k okraji vnějšího kolečka nasměrujeme loď do požadovaného směru. Vnitřní kolečko je následně přichyceno k jednomu ze čtyř směrů dle toho, ke kterému je nejblíže. Směr je následně zobrazen na radaru.



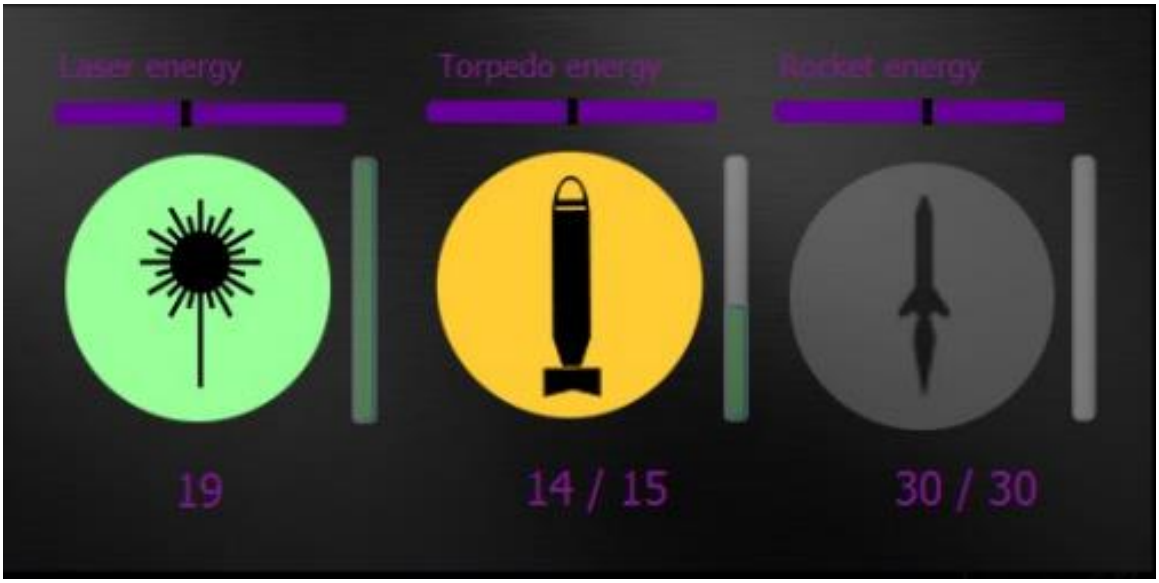
Obrázek 8: Joystick

Tlačítka pro ovládání bonusů lodi je možné vidět na obrázku č. 9. První tlačítko doplní zásoby zbraní vždy o tři rakety a deset torpéd. Druhé doplní posádku o jednu polovinu původního stavu posádky. Třetí tlačítko restartuje dobu, po kterou je generátor přetížen. Poslední tlačítko aktivuje stíhače, které udělí poškození okolním lodím ve vzdálenosti do pětset jednotek o tři tisíce jednotek. Vždy lze použít pouze jeden bonus za určitý čas. Po použití se tlačítka stanou neaktivními. Více informací o bonusech se nachází v samostatné podkapitole vlastní loď.

Obrázek 9: Bonusy

Poslední částí je panel pro ovládání zbraní. Zobrazen je na obrázku č. 10. V horní části jsou zobrazeny posuvníky, které přivádějí energii vymezenou pro zbraně do jednotlivých zbraňových systémů. První tlačítko je určeno pro laser. Síla laseru se nabíjí postupně na maximální hodnotu. Aktuální hodnota nabití je vidět na komponentě progres bar na levé straně od tlačítka. Hodnota je také textově vypsána pod tlačítkem. Hodnota určuje jeho aktuální sílu. Druhé tlačítko slouží k vypuštění torpéd a třetí k vypuštění raket. Pokud jsou tlačítka šedivá, nebylo přesměrováno dostatečné množství energie do systémů dané zbraně. Žluté tlačítko značí nabíjení zbraně. Aktuální hodnota nabytí je pak znázorněna v progres baru na levé straně od

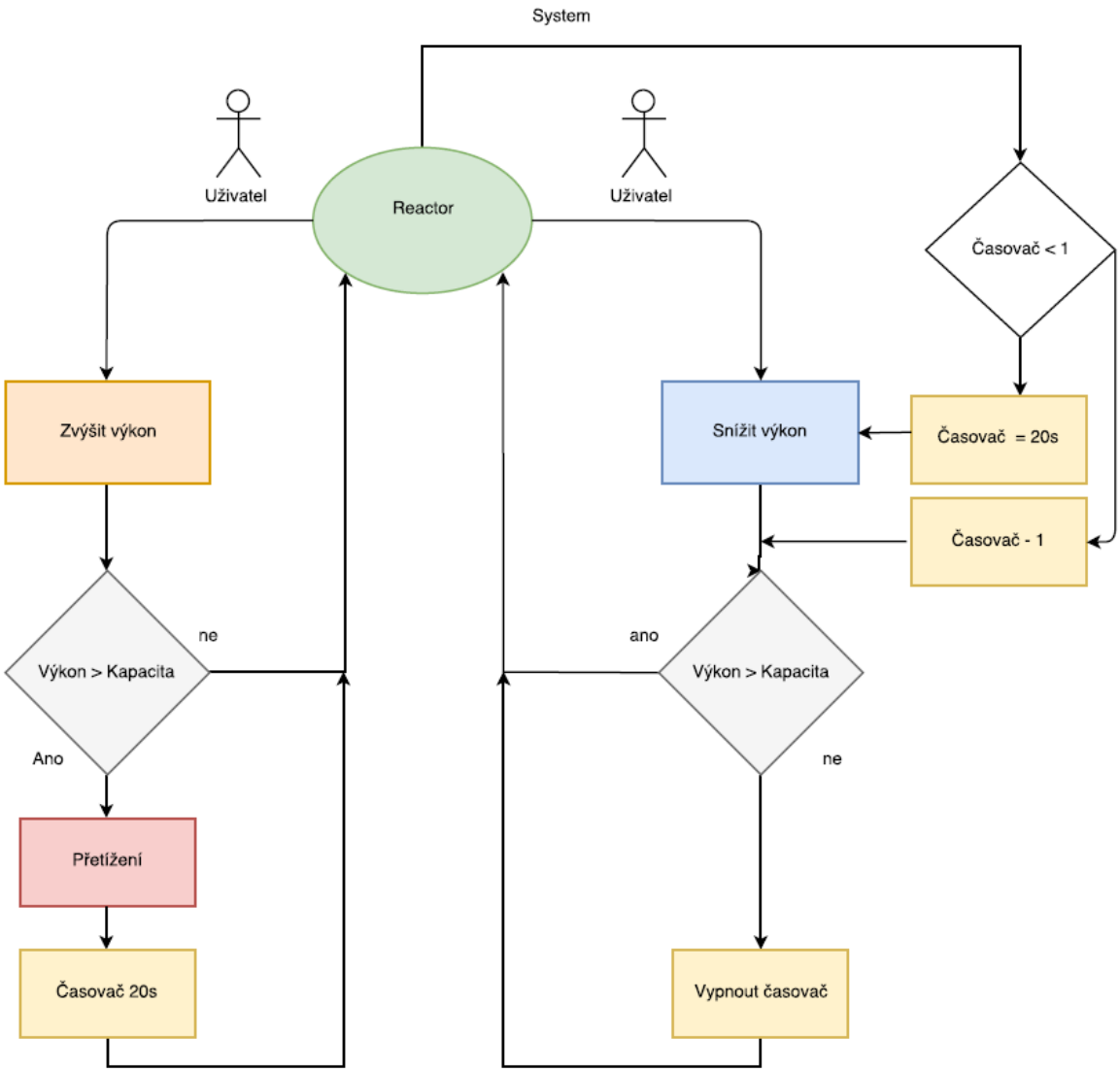
tlačítka. V případě že zbraň nabitá, tak tlačítko zezelená a je připraveno k palbě. Pro výstřel z rakety je nutné mít označený cíl. Hodnoty pod tlačítky určují aktuální stav munice.



Obrázek 10: Zbraně

4.4 Flow diagram energie

Na obrázku č. 11 je zobrazen flow diagram, v něm je graficky znázorněn princip chování reaktoru. Při uživatelském zvýšení výkonu reaktoru dojede k jeho přičtení. Pokud výkon přesahuje nad jeho kapacitu, dojde k spuštění odpočtu časovače. Po uplynutí časového intervalu nastává selhání reaktoru a následuje restart všech systémů lodi. Výkon je tedy snížen automaticky systémem. V případě že uživatel sám sníží výkon pod kapacitu rektoru, je odpočet zastaven. Čas odpočtu však není resetován, protože již došlo k částečnému poškození reaktoru. Odpočet se tedy restartuje pouze po selhání reaktoru nebo pomocí bonusů.



Obrázek 11: Flow diagram generátoru

4.5 Scénáře

Hra obsahuje tři scénáře. V kontextovém menu si je uživatel volí dle vlastního uvážení. Scénáře na sobě nejsou nijak závislé a vždy po jejich spuštění je hra restartována.

4.5.1 Scénář 1

Prvním typem je *tutorial* neboli trénink. Zde je možné s vyzkoušet ovládání lodi, pohyb po herním prostoru, zaměření a střelbu na cíl. Na herním prostoru se nachází jen jedna nepřátelská loď, která se nepohybuje. Scénář 1 je proto primárně určen pro začínající hráče, aby se seznámili s technikou ovládání simulátoru.

4.5.2 Scénář 2

Druhým scénářem je *catch*. Do herního plánu se zde náhodně vygeneruje dvacet pět lodí. Hráčovým úkolem je najít všechna nepřátelská plavidla a následně je zničit. Všechny lodi se vzdalují daleko do vesmíru. Nachází se zde pět bitevních lodí, sedm raketových lodí, osm útočných a pět nákladních lodí. Úkolem hráče je tyto lodi dostihnout a zneškodnit. V tomto scénáři je možné si plně vyzkoušet pohyb lodi po herní ploše a pronásledování nepřátelských plavidel. Pro splnění úkolu je zapotřebí zvolit správné rozložení energie pro pohyb a obranu. Tento typ hry je doporučen hráčům, kteří se již seznámili s herními technikami.

4.5.3 Scénář 3

Třetím scénářem je *defend*. V tomto případě se do hry podobně jako v předchozím scénáři vygenerují plavidla. V herním plánu se nachází deset bitevních lodí, osm raketových, sedm útočných a pět nákladních. Celkem je tedy v tomto scénáři třicet lodí. Všechny nepřátelské lodi se pohybují směrem k hráči, který má za úkol se útočícím lodím ubránit. Tento scénář je nejobtížnější. Hráč musí zvolit vhodný poměr rozmístění energie a sledovat úhel, pod kterým se přibližující se nepřátelské lodě střílí a podle toho správně volit rozložení energie ve štítech. Tento scénář je doporučený pouze zkušeným hráčům, kteří hledají výzvy.

4.5.4 Konec hry

Ve hře mohou nastat dva konce. V prvním případě hráč porazí všechny nepřátelské lodě. Po zničení poslední nepřátelské lodě se hráči objeví zpráva o výhře se statistikami. V druhém případě simulace končí, pokud dojde k zničení lodi uživatele, nebo k úmrtní všech členů posádky. V tomto případě dojde k zobrazení praskliny na radaru a jeho zastavení. Cizí lodě již nejsou dále vykreslovány.

4.6 Vlastní loď

Vlastní loď je vesmírné plavidlo ovládané hráčem. Jedná se o nejdůležitější objekt simulátoru.

Všechny její prvky jsou ovládány pomocí samostatných panelů, které byly popsány v přechozí podkapitole.

Z kontextu programové analýzy, je vlastní loď tvořena třídou *MySpaceShip,* která je potomkem třídy *SpaceShip*. Atributy a nastavení pro všechny tři scénáře jsou zobrazeny v následující tabulce.

Tabulka 2: Nastavení atributů vlastní lodi

Název atributu	Typ atributu	Popis atributu	Nastavení atributů dle scénáře		
			tutorial	*catch*	*defend*
shipHull	integer	Integrita trupu	6000	9000	15000
crue	integer	Počet posádky	5000	5000	6000
capacityRocket	integer	Kapacita raket	9	15	17
capacityTorpdo	integer	Kapacita torpéd	15	30	40
powerLaser	integer	Základní síla laseru	0	0	0

4.6.1 Rychlost lodi

Parametr rychlost lodi určuje, jak rychle se loď pohybuje po herním plánu. Nastavení probíhá na prvním posuvníku, jenž je zobrazen na obrázku č. 4. Rychlost je zobrazena na informačním panelu v procentech.

4.6.2 Systémy posádky

Systém se stará o komfort a bezpečí posádky. Při útoku na loď snižuje šanci na zranění jejich členů. Počet zraněných členů posádky se vypočítává náhodně v rozmezí nula až dvacet, pokud je loď zasažena.

4.6.3 Síla štítů

Síla štítů určuje množství energie, které je přiděleno obraně lodi. Toto množství energie je dále hráčem přerozdělováno mezi tři druhy štítů. Přední štít chrání před střelami přicházejícími z úhlu nula až šedesát stupňů. Druhý centrální štít, který chrání před útoky z pod úhlem větším než šedesát a menším než sto dvacet stupňů. Poslední zadní štít, chrání před střelami přicházejícími z úhlu většího než sto dvacet stupňů.

4.6.4 Zbraně

Vlastní loď má tři typy zbraní. Pro jejich použití je nutné převést do zbraňových systémů určité množství energie. To se provede navýšením hodnoty v třetím posuvníku. Pro aktivaci torpéd je zapotřebí minimálně dvacet pět jednotek energie, pro aktivaci raket padesát jednotek. Pokud

jsou zbraně aktivní, jejich tlačítka změní barvu z šedých na žlutou. Pro zahájení nabíjení zbraní je zapotřebí převést energii určenou pro zbraně do jednotlivcích zbraňových systémů. Tato energie se nastavuje pomocí posuvníků nad tlačítkem. Nabíjení zbraně je vizuálně zobrazeno na komponentě progres bar, nacházející se vlevo od jednotlivých tlačítek. Pokud je zbraň nabitá, tlačítko změní bravu na zelenou.

První typem zbraně je laser. Dostřel této zbraně je tři sta jednotek. A jeho síla je určená dle aktuálního stavu nabití. Druhým typem zbraně je torpédo. Vlastní loď disponuje větším množstvím těchto zbraní. Jejich nevýhodou je schopnost pouze přímého letu od čela lodě. Dostřel torpéda je tisíc jednotek a loď, kterou zasáhne, uštědří poškození tisíc jednotek. Posledním typem je raketa. Pro vystřelení z této zbraně je nutné mít cíl zaměřený. Dolet rakety je až tisíc pět set jednotek a poškození, které způsobí je dva tisíce jednotek.

4.6.5 Bonusy

Bonusy jsou speciální tlačítka, která nám vyvolají jeden ze čtyř bonusů. Bonus lze vyvolat jen jednou za určitý časový úsek. Tato doba je nastavena na čtyřicet sekund. První tlačítko doplňuje munici ve zbraních třemi raketami a deseti torpédy. Druhé doplňuje posádku o polovinu původního množství. Třetí tlačítko resetuje odpočet, po kterém dojde k přetížení reaktoru. Poslední vypustí stíhače, které poškodí všechny nepřátelské lodě v okolí o tři tisíce jednotek ve vzdálenosti do pětiset jednotek.

4.6.6 Generátor

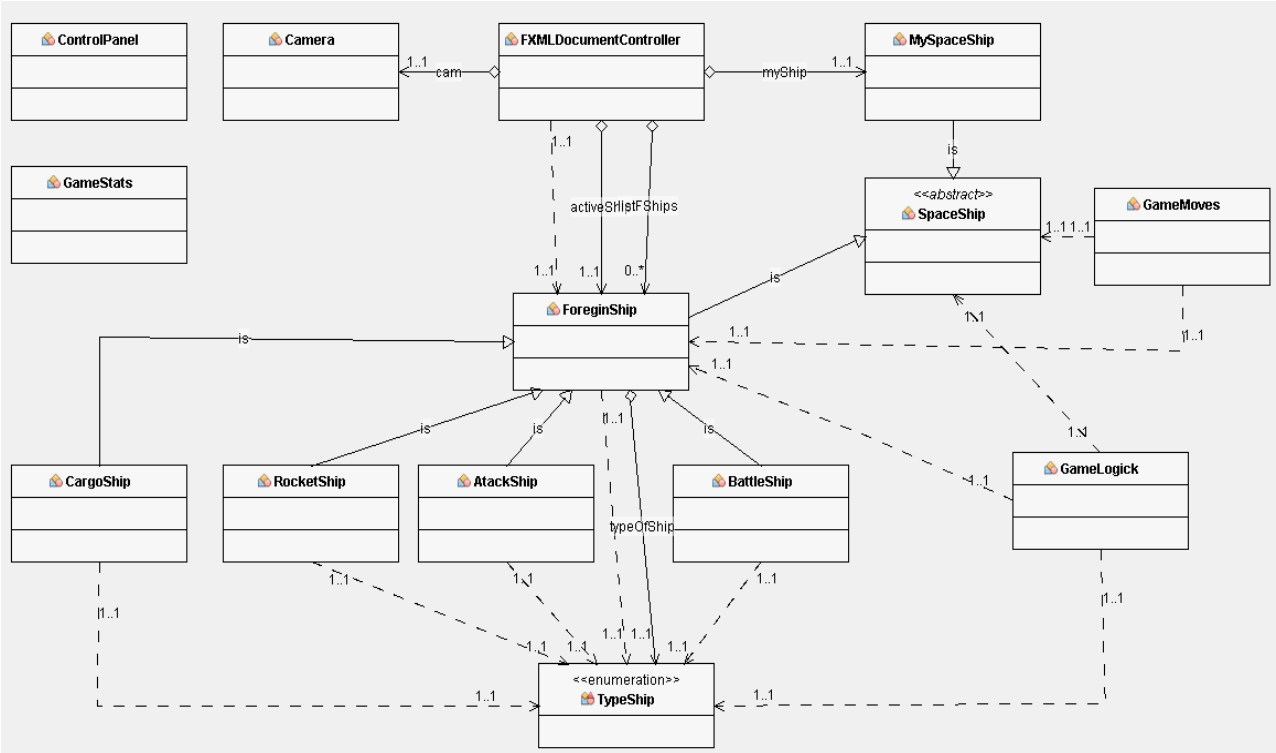
Generátor je jedna z nejdůležitějších část lodi. Jeho vytíženost je zobrazena na vodorovném posuvníku v panelu ovládacích systémů lodi. Pokud požadovaný výkon převýší jeho kapacitu, dojde k přetěžování a spustí se časový interval, který je stanoven na dvacet sekund. Po uplynutí tohoto intervalu je reaktor přetížen a jeho výkon automaticky snížen. Dochází tak k vypnutí všech systémů lodi. Pokud uživatel sníží energii a odpočet byl již spuštěn, odpočet se zastaví na jeho aktuální hodnotě. Tato hodnota se opět navýší až po restartování reaktoru. Lze ji také navýšit ručně, pomocí bonusu *boost energy*.

5 Programování

V této části je vyobrazen UML diagram a další podkapitoly jsou věnovány informacím o jednotlivých třídách hry – simulátoru.

5.1 Uml digram

Základní UML diagram znázorňuje všech čtrnáct tříd programu, jejich asociace, vytváření instancí a dědičnost. Jednotlivé třídy jsou podrobně popsány v následujících podkapitolách. Základní UML model je možné vidět na obrázku č.12. V modelu jsou skryty jednotlivé metody a atributy tříd, pro jeho přehlednost.



Obrázek 12: Uml diagram

5.2 Třída ControlPanel

Třída *ControlPanel* je základní třídou obsahující instanci *Scene,* která překládá FXML kód popisující, jak vlastní GUI vypadá. Ukázka FXML kódu je zobrazeno v příloze A. Dále je zde inicializována hlavní *stage,* která má v parametru instanci *scene* a vytváří hlavní okno programu. Třída *ControlPanel* dále obsahuje metodu *main,* která aplikaci spouští a metodu getScene díky, které je možné předat instanci *Scene* kontroleru a tím přepínat celý program do pleného rozlišení.

5.3 Třída FXMLDocumentController

Třída *FXMLDocumentController* je hlavní třída celé grafické části programu. Inicializují se zde všechny komponenty. Mezi ně patří například tlačítka, *labely* s texty, *progress bary*, panely a *imageView*, což jsou komponenty pro zobrazení obrázku. Obrázky jsou následně vykreslovány na jednotlivé panely. V neposlední řadě zde také probíhá inicializace obsluhy všech komponent jako je například odposlechu slideru. Příklad takovéto inicializace je uveden v příloze B. Třída *FXMLDocumentController* dále vytváří instance třídy *MyShip* a instanci datové struktury *arrayList*, ve které uchovává objekty jednotlivých cizích lodí.

5.4 Abstraktní třída SpaceShip

Abstraktní třída *SpaceShip* obsahuje společné atributy a metody cizích lodí a vlastní lodě. Je typu *abstract*, proto obsahuje pouze abstraktní metody a její instance nejsou vytvářeny. Jsou vytvářeny pouze instance tříd, které z této třídy dědí.

5.5 Třída MySpaceShip

Třída *MySpaceShip* je třída, která vytváří instanci vlastní lodě. Tato třída dědí z abstraktní třídy *SpaceShip* a využívá všechny její atributy a metody které jsou zde implementovány. Instance je vytvářena ve třídě *FXMLDocumentController*.

5.6 Třída ForeginShip

Třída *ForeginShip*, je společná třída pro všechny cizí lodě. Dědí z abstraktní třídy *SpaceShip* a implementuje všechny její metody. Tato třída navíc obsahuje další atributy, které v lodi vlastní nebyly potřeba. Třída obsahuje metodu *action*, která se stará o nabíjení zbraní. V konstruktoru jsou nastavovány společné atributy všech cizích lodí.

5.7 Třída BattleShip

Třída *BattleShip* je potomkem třídy *ForeginShip* a využívá výčtového datového typu *BattleShip*. Třída slouží k vytváření instancí lodí typu BattleShip. Všechny specifické parametry, jakou je například síla trupu nebo štítů, jsou nastavovaný v konstruktoru. Instance jsou vytvářeny v metodě *generationShips*, která se nachází ve třídě *GameLogic*.

5.8 Třída AtackShip

Třída *AtackShip* je zděděná z třída *ForeginShip* a využívá výčtového datového typu *AtackShip*. Parametry specifické pro tuto třídu jsou také nastavovány v konstruktoru této třídy.

5.9 Třída RocketShip

Třída *RocketShip* je opět jako předchozí dvě třídy cizích lodí potomkem třídy *ForeginShip. Od o*bou tříd se liší pouze parametry a ikonami, které jsou nastavovány v konstruktoru nebo metodách třídy. Instance jsou také vytvářeny v m metodě *generationShips,* která se nachází ve třídě *GameLogic.*

5.10 Třída CargoShip

Třída *CargoShip* je poslední třídou cizích lodí. Specifická je především natavením atributu *friendly,* který je standardně nastaven na *true*. Tento typ lodě, je tedy v jejím výchozím nastavení přátelský. Pro snadnější rozeznání využívá vlastního výčtového datového typu *CargoShip. Z*ákladní parametry jsou nastavovány v konstruktoru a instance je vytvářena, podobně jako u předchozích tříd dědících ze třídy *ForeginShip,* ve třídě *GameLogic*.

5.11 Enum TypeShip

Enum *TypeShip* jsou výčtové typy pro jednodušší rozpoznání lodí typy které se zde nachází jsou: *BattleShip, RocketShip, AtackShip, CargoShip.*

5.12 Třída GameLogick

Třída *GameLogic* obsahuje statické metody herní logiky. Jsou to metody: *generationShips* pro generování lodí na začátku hry, *weaponFire* pro udělení poškození cizí lodi, *torpedoFire* pro výstřel z torpéda. Dále pak metody *luanchFighters,* která vyvolá poškození u cizích lodí v okolí, *angleOfFire* metoda vracející úhel, pod kterým nepřátelská loď pálí a *destroyShip, což je* metoda starající se odstranění objektu zničené lodě. Poslední je metoda *demageMyShip,* která obsluhuje poškození vlastní lodě. Kód této metody je uveden v příloze C.

5.13 Třída GameMoves

Třída *GameMoves* podobně jako třída *GameLogic* obsahuje statické metody. Tyto metody mají za úkol především zajisti pohyby všech lodí. Třída obsahuje následující metody: První metoda je *moveWithShip.* Sloužící pro pohyb vlastí lodě. Dle pohybu vlastní lodě jsou následně spuštěny metody pro posun nepřátelských lodí na radaru. Jsou to metody *shipUp, shipDown, shipLeft* a *shipRiht.* Dále třída *GameMoves* obsahuje metody pro pohyb cizích lodí dle scénáře. Je to metoda *moveWithForeginShipRun*, kterou spouští druhý scénář hry. Pozice cizích lodí jsou zde přičítány, což znamená že se lodě neustále vzdalují. Dále metoda *moveWithForeginShip,* kterou spouští třetí scénář hry. Tato metoda nejdříve vypočítává pozici

cizí lodě vůči lodi vlastní, a dle toho se loď přiblíží k lodi hráče. Kód metody *moveWithForeginShip* je pro ukázku vyobrazen v příloze D.

5.14 Třída GameStats

Třída *GamesStats* obsahuje statické metody a atributy držící a následně upravující informace o statistikách. Atribut *destriedEnemyShips* drží informace o počtu zničených nepřátelských lodí, *destroyedFriendlyShips* drží informace o počtu zničených přátelských lodí a *enemyShipsInGame* drží celkový počet nepřátelských lodí ve hře.

5.15 Třída Camera

Poslední třídou, která se v programu nachází je *Camera*. Slouží k přepočtu souřadnic, na kterých se vykreslují cizí lodě. Obsahuje pouze atributy se souřadnicemi a metody pro jejich obsluhu.

6 ZÁVĚR

Prvním úkolem bakalářské práce bylo zvolit vhodnou technologii pro vývoj softwaru vlastní hry. Zvolena byla platforma JavaFX a vývojové prostředí NetBeans IDE. Dalším úkolem bylo vybrat nejvhodnější vstupně výstupní zařízení, které by bylo možné použít pro ovládání. Z různých druhů zařízení, které jsou v práci popsány, byla nejvhodnější kombinace dotykové fólie a projekční fólie se zadní projekcí. Tím je úkol splněn. Posledním úkolem bylo vytvořit panel, který bude odpovídat panelům ze sci-fi filmů a bude tak nadstavbou pro počítačovou hru, která je součástí práce – i tento podproblém byl zpracován a popsán v rámci této práce.

V současné době jsou splněny všechny úkoly, které byly v zadání práce. Uživatel může nastavovat základní charakteristiky lodě. Mezi tyto charakteristiky patří: hospodaření s energií, řízení systému letu, nastavování více druhů štítů, nabíjení zbraní a řízení střelby. Simulace obsahuje tři scénáře, ve kterých je uživatelova loď v interakci s cizími loděmi.

Pro splnění této práce jsem využíval znalostí, kterých jsem nabyl ve škole. Jsou to především znalosti z objektově orientovaného programování, počítačové grafiky, lineární algebry a algoritmizace.

Přestože bylo zadání splněno, má práce další potenciál. Do budoucna by bylo možné zlepšit především grafiku hry. Rád bych například vylepšil střelbu lodí, kdy by se každá vystřelená střela vykreslovala podobně jako lodě. Radar by bylo možné přenést na celou novou obrazovku a aktuální radar by sloužil pouze pro zobrazení nejbližších objektů. Dále bych rád přidal nové typy lodí, a také přátelské lodě, které by ve hře kooperovali s hráčem. V úvahu přichází i možnost do hry přidat hráče, který by ovládal další loď. To by ze aktuální simulační aplikace vytvořilo plnohodnotnou hru pro více hráčů, což by ji dělalo ještě zábavnější.

7 Použitá literatura

[1] NOVOTNÝ Luděk. 2003. Historie a vývoj jazyka Java. [online] Dostupné z: http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xnovotn8.htm. [Accessed 25 March 2017].

[2] SHARAN, Kishori. *Learn JavaFX 8: building user experience and interfaces with Java 8*. Expert's voice in Java. ISBN 978-1-4842-1142-7.

[3] JANOVSKÝ, Dušan. CSS styly – úvod. *Jak psát web* [online]. 2015? [cit. 2017-04-30]. ISSN 1801-0458. Dostupné z: https://www.jakpsatweb.cz/css/css-uvod.html

[4] NetBeans IDE – The Smarter and Faster Way to Code. *NetBeans IDE* [online]. Oracle America, ©2017 [cit. 2017-04-29]. Dostupné z: https://netbeans.org/features/index.html

[5] *IntelliJ IDEA* [online]. Praha 4: JetBrains Software Product, ©2017 [cit. 2017-04-30]. Dostupné z: https://www.jetbrains.com/idea/

[6] JavaFX Scene Builder. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2015 [cit. 2017-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/JavaFX_Scene_Builder

[7] JavaFX Scene Builder A Visual Layout Tool for JavaFX Applications [online]. Oracle, ©1995-2017 [cit. 2017-03-26]. Dostupné z: http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html

[8] JAVŮREK, Karel. Dotyková technologie: začalo to před 50 lety. *Zive.cz* [online]. 2013-09-03 [cit. 2017-05-03]. ISSN 1213-8991. Dostupné z: http://www.zive.cz/clanky/dotykova-technologie-zacalo-to-pred-50-lety/sc-3-a-170384

[9] VOJÁČEK, Antonín. Současný stav vývoje rezistivních dotykových ploch / displejů. *Automatizace.hw.cz* [online]. 2010-10-24 [cit. 2017-05-03]. Dostupné z: http://automatizace.hw.cz/soucasny-stav-vyvoje-rezistivnich-dotykovych-ploch-displeju

[10] ŠKOPEK, Pavel. Techbox: dotykové displeje – čím se liší rezistivní od kapacitního? Mobilnet.cz [online]. 24net, 2013-03-15 [cit. 2017-05-02]. Dostupné z: https://mobilenet.cz/

[11] TOUCHWINDOW: Touch Foil [online]. Itálie: TOUCHWINDOW, ©2017 [cit. 2017-03-28]. Dostupné z: http://www.touchwindow.it/en/touch-foil.html

[12] UGO! MEDIA, S. R. O. *Dotyková fólie UGO! Touch* [online]. ©2016 [cit. 29.4.2017]. Dostupný na http://www.ugo-media.eu/materialy/dotykova-folie/dotykova-folie-ugo-touch

[13] TOUCHWINDOW: OVERLAY MULTI-TOUCH [online]. Itálie: TOUCHWINDOW, ©2017 [cit. 2017-03-28]. Dostupné z: http://www.touchwindow.it/en/overlay-multi-touch.html

8 Přílohy

Příloha A - Ukázka FXML kódu

V této ukázka FXML kódu je vidět rozložení layoutů panelu radaru. A obrázků, které se na něm nachází.

```
<Pane fx:id="PaneRadar" layoutX="645.0" layoutY="82.0" prefHeight="640.0"
prefWidth="710.0">
      <children>
            <ImageView fitHeight="649.0" fitWidth="727.0" layoutX="1.0"
      layoutY="7.0" pickOnBounds="true">
                  <image>
                        <Image url="@../pictures/Radar3.png" />
                  </image>
            </ImageView>
            <ImageView fx:id="imageWievMyShipRadar" fitHeight="40.0"
      fitWidth="30.0" layoutX="351.0" layoutY="311.0" pickOnBounds="true"
      preserveRatio="true">
                  <image>
                        <Image url="@../pictures/MojeLod.png" />
                  </image>
            </ImageView>
            <ImageView fx:id="ImageViewRadarLine" fitHeight="11.0"
      fitWidth="725.0" layoutX="2.0" layoutY="22.0" pickOnBounds="true">
                  <image>
                        <Image url="@../pictures/RadarVyhledavani.png" />
                  </image>
            </ImageView>
      </children>
</Pane>
```

Příloha B - Ukázka kódu inicializace listeneru slideru předního štítu

V příloze B se nachází ukázka kódu pro inicializaci slideru předního štítu. Jsou zde upravovány hodnoty slideru centrálního a zadního štítu vzhledem k použité energii předního štítu.

```
 sliderFrontSh.valueProperty().addListener((ObservableValue<? extends
Number> observable, Number oldValue, Number newValue) -> {
            int tem = (int) sliderFrontSh.getValue();
            if (tem > energyFrontShield && (tem + energyCenterShield +
energyBackShield) >= (int) energyShield) {
                int over = (int) ((tem + energyCenterShield +
energyBackShield) - energyShield);
                if (over == 1) {
                    if (sliderBackSh.getValue() > 0) {
                        sliderBackSh.setValue(sliderBackSh.getValue() - 1);
                    } else if (sliderCenterSh.getValue() > 0) {
                    sliderCenterSh.setValue(sliderCenterSh.getValue() - 1);
                    }
                }
                if (over > 1) {
                    for (int i = 0; i < Math.rint(over / 2); i++) {
                        if (sliderBackSh.getValue() > 0) {
                        sliderBackSh.setValue(sliderBackSh.getValue() - 1);
                        } else if (sliderCenterSh.getValue() > 0) {
sliderCenterSh.setValue(sliderCenterSh.getValue() - 1);
                        }
                    }
                    for (int i = 0; i < Math.rint(over / 2); i++) {
                        if (sliderCenterSh.getValue() > 0) {
sliderCenterSh.setValue(sliderCenterSh.getValue() - 1);
                        } else if (sliderBackSh.getValue() > 0) {
                        sliderBackSh.setValue(sliderBackSh.getValue() - 1);
                        }
                    }
                }
            }
            energyFrontShield = tem;
            imageFrontSh.setOpacity(energyFrontShield / 100);
       labelFrontSh.setText("Front shield: " + energyFrontShield + "%");});
```

Příloha C - Ukázka kódu poškození vlastní lodě

V příloze C je ukázka kódu pro poškození vlastní lodě vzhledem k úhlu, pod kterým na tuto loď dopadají střeli. Pokud je loď zničena metoda vrací hodnotu true.

```
public static boolean demageMyShip(MySpaceShip myShip, double demage, int
angele, double shield1, double shield2, double shield3,int crue){
        double finalDemage;
        if(angele>0&&angele<60){
            finalDemage = (demage/100)*shield1;
        }else if(angele>=60&&angele<120){
            finalDemage = (demage/100)*shield2;
        }else{
            finalDemage = (demage/100)*shield3;
        }

        myShip.setShipsHull((int) (myShip.getShipsHull()-(demage-
finalDemage)));
        if(demage-finalDemage>0){
           int temp = 100- crue;
           Random rn = new Random();
           int random = rn.nextInt(100+1);
           if(random>temp){
           }else{
               int crueDest = rn.nextInt(20+1);
               myShip.setCrue(myShip.getCrue()-crueDest);
               if(myShip.getCrue()<0){
                   return true;
               }
           }
         }
        return myShip.getShipsHull()<0;
       }
}
```

Příloha D - Ukázka kódu pohybu cizí lodě

V příloze D se nachází metoda, která posune cizí lodí. Nejprve je vypočítáno, kde se loď nachází vůči lodi vlastní a následně dojde k úpravě souřadnic tak, aby se k uživatelovi lodi přiblížila.

```
public static void moveVithForegineShip(ForeginShip ship, MySpaceShip
myShip, Camera cm) {


        if (ship.getTypeOfShip() != TypeShip.CargoShip) {
            if (ship.getCoordY() < myShip.getCoordY()) {
                ship.setCoordY(ship.getCoordY() + 1);
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() - 1);
                }
                                    } else {
                ship.setCoordY(ship.getCoordY() - 1);
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() + 1);
                }
            }
            if (ship.getCoordX() < myShip.getCoordX()) {
                ship.setCoordX(ship.getCoordX() + 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() + 1);
                }
            } else {
                ship.setCoordX(ship.getCoordX() - 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() - 1);
```

```
                }
            }
        } else if (ship.getCicle() == false) {
            if (ship.getCoordX() > 10000) {
                ship.setCicle(true);
            } else {
                ship.setCoordY(ship.getCoordY() + 1);
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() - 1);
                }
                ship.setCoordX(ship.getCoordX() + 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() + 1);
                }
            }
       }else{
          if (ship.getCoordX() < 50) {
                ship.setCicle(true);
            } else {
                ship.setCoordY(ship.getCoordY() - 1);
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() + 1);
                }
                ship.setCoordX(ship.getCoordX() - 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() - 1);
                }
            }
```

```
        }
    }
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
            maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() - 1);
                }
                ship.setCoordX(ship.getCoordX() + 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() + 1);
                }
            }
        }else{
          if (ship.getCoordX() < 50) {
                ship.setCicle(true);
            } else {
                ship.setCoordY(ship.getCoordY() - 1);
                if (ship.getCoordY() > cm.y && ship.getCoordY() < cm.y +
maxY) {

ship.getPicture().setLayoutY(ship.getPicture().getLayoutY() + 1);
                }
                ship.setCoordX(ship.getCoordX() - 1);
                if (ship.getCoordX() > cm.x && ship.getCoordX() < cm.x +
maxX) {

ship.getPicture().setLayoutX(ship.getPicture().getLayoutX() - 1);
                }
            }
        }
    }
```

Příloha E - Příklad CSS stylu nastavení zobrazení tlačítka rakety

Příloha F je ukázkou kódu k nastavení vzhledu tlačítek pomocí CSS. V první části je standartní vzhled tlačítka. V další části je nastavena barvu při stisknutí tlačítka.

```
.button{
    -fx-background-radius: 9em;
    -fx-background-color: #99ff99;
    -fx-background-image: url("../pictures/Raketa.png");
     }

.button:pressed{
    -fx-background-color:  blueviolet;
}
.button:disabled{
    -fx-background-color: #999999;
}
```

Příloha F - Celkový UML diagram všech tříd volně vložený

Součástí práce je volně vložený UML diagram vytištěný na papíru formátu A3.

Příloha G - CD s aplikací volně vložené

Volně vložené CD obsahuje všechny kódy programu a celý projekt z NetBeans. V projektu se nachází spustitelný soubor s koncovkou .jar, spustitelný soubor s koncovkou .exe. Dále CD ROM obsahuje celou práci ve formátu PDF.