

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-5382-74728

**INTERAKTÍVNA EDUKAČNÁ HRA „TAJOMSTVÁ
ELEKTRONIKY A FOTONIKY“
BAKALÁRSKA PRÁCA**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-5382-74728

INTERAKTÍVNA EDUKAČNÁ HRA „TAJOMSTVÁ ELEKTRONIKY A FOTONIKY“ BAKALÁRSKA PRÁCA

| | |
|--------------------------|-------------------------------------|
| Študijný program : | Aplikovaná informatika |
| Číslo študijného odboru: | 2511 |
| Názov študijného odboru: | 9.2.9 Aplikovaná informatika |
| Školiace pracovisko: | Ústav informatiky a matematiky |
| Vedúci záverečnej práce: | prof. Ing. Ľubica Stuchlíková, PhD. |



ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Andrej Dičér**
ID študenta: 74728
Študijný program: aplikovaná informatika
Študijný odbor: 9.2.9. aplikovaná informatika
Vedúca práce: prof. Ing. Ľubica Stuchlíková, PhD.
Miesto vypracovania: Ústav elektroniky a informatiky

Názov práce: **Interaktívna edukačná hra "Tajomstvá elektroniky a fotoniky"**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Hlavným zameraním práce je návrh a vývoj interaktívnej edukačnej hry "Tajomstvá elektroniky a fotoniky" ovládanej pomocou klávesnice vo vybranom programe. Hra bude vytvorená na báze objektovo orientovaného programovania. Vytvorená aplikácia bude voľne dostupná a voľne šíriteľná v rámci popularizácie vedy a techniky. Cieľovou skupinou sú žiaci základných a stredných škôl. Hra ich má pútavým spôsobom uviesť do problematiky objavov, vynálezov alebo zaujímavých technických riešení v oblasti elektroniky a fotoniky. Hra by mala obsahovať viacero rôznych úrovní s lineárne rastúcou obtiažnosťou. Jednotlivé úrovne budú vyžadovať nielen postreh, ale i prieskum situácie a otváranie ciest k poznaniu. Súčasťou hry budú jednoduché hlavolamy a zdroje nebezpečia.

Úlohy:

1. Oboznámte sa s princípmi tvorby interaktívnych hier vo vybranom programe na báze objektovo orientovaného programovania.
2. Navrhňte a vytvorte interaktívnu edukačnú hru "Tajomstvá elektroniky a fotoniky" zobrazujúcu vybrané problematiky z oblasti elektroniky a fotoniky s využitím objektovo orientovaného programovania.
3. Urobte optimalizáciu, otestujte jednotlivé úrovne a vypracujte používateľskú dokumentáciu.

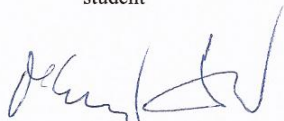
Zoznam odbornej literatúry:

1. BEALE, R. and SHARPLES, M. Design Guide for Developers of Educational Software. [Online] 2002.
[http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/0H420/EDUCdesignguide\[2002\].pdf](http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/0H420/EDUCdesignguide[2002].pdf)

Riešenie zadania práce od: 19. 09. 2016

Dátum odovzdania práce: 19. 05. 2017

Andrej Dičér
študent



prof. RNDr. Otokar Grošek, PhD.
vedúci pracoviska

SLOVENSKÁ TECHNICKÁ UNIVERZITA
V BRATISLAVE
Fakulta elektrotechniky a informatiky
Ústav informatiky a matematiky
Ilkovičova 3, 812 19 Bratislava



prof. Dr. Ing. Miloš Oravec
garant študijného programu

SÚHRN

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

| | |
|---------------------------------|---|
| Študijný program : | Aplikovaná informatika |
| Bakalárska práca: | Interaktívna edukačná hra „Tajomstvá elektroniky a fotoniky“ |
| Autor: | Andrej Dičér |
| Vedúci záverečnej práce: | prof. Ing. Ľubica Stuchlíková, PhD. |
| Miesto a rok predloženia práce: | Bratislava 2017 |

Predložená bakalárska práca sa zaoberá návrhom a vývojom interaktívnej edukačnej hry „Tajomstvá elektroniky a fotoniky“ v *HTML5* s využitím objektovo orientovaného programovania. Hra bola vytvorená vo forme prehliadačovej hry pre dvoch hráčov, žiakov základných alebo stredných škôl. Jedná sa o príbeh astronautov Fotóna a Elektróna na mesiaci s troma úrovňami a 9-timi minihrami. Príbeh je možné rozširovať na základe technickej dokumentácie uvedenej v práci, ktorá obsahuje podrobný popis naprogramovaných tried, ich vlastností a metód. Vytvorená aplikácia je voľne šíriteľná v rámci popularizácie vedy a techniky. Prvá funkčná verzia našej edukačnej hry je voľne dostupná na vzdelávacom portáli eLearn central.

Kľúčové slová: edukačná online hra, objektové programovanie, *HTML5*, multiplayer, javascript

ABSTRACT

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION
TECHNOLOGY

| | |
|-------------------------------|--|
| Study Programme: | Applied Informatics |
| Bachelor Thesis: | Interactive educational game „Secrets of electronics and photonics“ |
| Autor: | Andrej Dičér |
| Supervisor: | prof. Ing. Ľubica Stuchlíková, PhD. |
| Place and year of submission: | Bratislava 2017 |

The presented bachelor thesis deals with the design and development of an interactive educational game "Secrets of Electronics and Photonics" in *HTML5* using object-oriented programming. The game was created as a browser game for two players, students of elementary or secondary schools. The story is about two astronauts, Elektrón and Fotón, on the moon. The story is composed of three levels and nine minigames and can be expanded on the basis of the technical documentation given in the work, which contains a detailed description of the programmed classes, their properties and methods. The created application is freely distributable within the popularization of science and technology. The first functional version of our educational game is freely available on the eLearn Central Learning Portal.

Key words: educational online game, object-oriented programming, HTML5, multiplayer, javascript

Vyhlásenie autora

Podpísaný Andrej Dičér čestne vyhlasujem, že som Bakalársku prácu Interaktívna edukačná hra „Tajomstvá elektroniky a fotoniky“ vypracoval na základe poznatkov získaných počas štúdia a informácií z dostupnej literatúry uvedenej v práci.

Uvedenú prácu som vypracoval pod vedením prof. Ing. Ľubica Stuchlíková, PhD. .

V Bratislave dňa 17.05.2017

.....

podpis autora

Pod'akovanie

Rád by som sa poďakoval mojej vedúcej prof. Ing. Ľubici Stuchlíkovej, PhD. za výborné vedenie a spoluprácu pri riešení tejto bakalárskej práce.

Obsah

| | |
|---|----------|
| Úvod | 1 |
| 1 Analýza..... | 2 |
| 1.1 Úvod do problematiky | 2 |
| 1.2 Technológie..... | 2 |
| 1.2.1 Adobe Flash | 2 |
| 1.2.2 HTML5 | 3 |
| 1.2.3 CSS | 3 |
| 1.2.4 JavaScript..... | 4 |
| 1.2.5 Node.js | 4 |
| 1.2.6 Socket.io | 4 |
| 1.2.7 Použité technológie..... | 5 |
| 1.3 Objektovo orientované programovanie | 5 |
| 2 Opis riešenia | 7 |
| 2.1 Špecifikácia požiadaviek | 7 |
| 2.1.1 Cieľová skupina | 7 |
| 2.2 Návrh..... | 7 |
| 2.2.1 Návrh príbehu | 8 |
| 2.2.2 Návrh úrovní | 9 |
| 2.3 Implementácia | 10 |
| 2.3.1 Server | 11 |
| 2.3.2 Menu | 12 |
| 2.3.3 Hra | 13 |
| 2.3.4 Svet | 14 |
| 2.3.5 Hráč..... | 14 |
| 2.3.6 Úroveň | 16 |
| 2.3.7 Minihry | 17 |
| 2.3.8 Rozmiestnenie elementov | 23 |
| 2.3.9 Ovládanie | 24 |
| 2.3.10 Tvorba grafiky | 24 |

| | | |
|---|--|-----------|
| 2.3.11 | Diagram tried | 25 |
| 3 | Zhodnotenie | 26 |
| 3.1 | Testovanie | 26 |
| 4 | Technická dokumentácia..... | 27 |
| 4.1 | Nasadenie hry na server | 27 |
| 4.2 | Popis najdôležitejších tried hry | 27 |
| 4.2.1 | Trieda Game..... | 27 |
| 4.2.2 | Trieda Canvas | 29 |
| 4.2.3 | Trieda World..... | 30 |
| 4.2.4 | Trieda Sprite..... | 31 |
| 4.2.5 | Trieda Player..... | 32 |
| 4.2.6 | Trieda Teammate | 33 |
| 4.2.7 | Trieda GameObject..... | 34 |
| 4.2.8 | Trieda Stage | 35 |
| 4.2.9 | Trieda MiniGame..... | 36 |
| Záver | | 38 |
| Zoznam použitej literatúry | | 39 |
| Prílohy | | I |
| Príloha A: Štruktúra elektronického nosiča | | II |

Zoznam obrázkov a tabuliek

| | |
|---|----|
| Obrázok 1 - Úvodná stránka a menu | 11 |
| Obrázok 2 - Vytvorenie miestnosti..... | 13 |
| Obrázok 3 - Pripojenie sa do miestnosti | 13 |
| Obrázok 4 - Druhá úroveň | 16 |
| Obrázok 5 – Minihra LED pexeso | 18 |
| Obrázok 6 - Minihra s RGB LED | 19 |
| Obrázok 7 - Minihra zapojenie solárnych panelov..... | 20 |
| Obrázok 8 - Minihra zapojenie laserovej komunikácie | 20 |
| Obrázok 9 - Minihra triedenie laserov | 21 |
| Obrázok 10 - Minihra nabíjanie akumulátorov | 22 |
| Obrázok 11 - Informačný panel | 23 |
| Obrázok 12 - Sprite sheet postavy Elektrón..... | 24 |
| Obrázok 13 - Diagram tried na strane klienta | 25 |

Zoznam skratiek a značiek

WWW - World Wide Web

GIF - Graphics Interchange Format

EXE - executable

3D - trojrozmerný

HTML - HyperText Markup Language

API - Application Programming Interface

CSS - Cascading Style Sheets

ECMA - European Computer Manufacturers Association

OOP - objektovo orientované programovanie

HTTP - HyperText Transfer Protocol

PHP - Hypertext Preprocessor

RGB - Red-Green-Blue color format

LED - Light-Emitting Diode

URL - Uniform Resource Locator

ID - identifikačné číslo

FPS - snímky za sekundu

Úvod

Počet uchádzačov o stredné a vysoké školy s technickým zameraním každým rokom klesá, čo nie je v súlade so spoločenskými potrebami. Je nevyhnutné túto situáciu zmeniť. Chceme dosiahnuť, aby sa o vedu a techniku začali zaujímať už deti v rannom veku. V dnešnej technologickej dobe, kedy má každý doma smartfón, tablet alebo počítač sa deti rýchlo oboznamujú s týmito technológiami a sú často gramotnejší v tejto oblasti ako ich rodičia. Deti „surfujú“ na internete, pozerajú videá a hrajú hry. A preto, práve hranie hier je jedným z najlepších spôsobov ako zaujať a prilákať deti k vede a technike. Pomocou edukačných hier vieme pritiahnuť pozornosť mladých ľudí a uviesť ich do problémov danej oblasti a zároveň ich niečo nové naučiť. Hra je efektívnym nástrojom na získanie technických znalostí ale aj základov k spolupráci, čo je v praxi mimoriadne žiadané.

Cieľom tejto práce je návrh a vývoj voľne dostupnej a šíriteľnej interaktívnej edukačnej hry „Tajomstvá elektroniky a fotoniky“ pre popularizáciu vedy a techniky. Prvým krokom bude analýza problému. Musíme sa oboznámiť s rôznymi technikami vývoja hier a zvoliť si technológiu, ktorá nám bude vyhovovať. Vezmeme do úvahy hľadisko obsahu hry (čo budeme robiť), hľadisko dostupnosti (hra by mala byť ľahko dostupná pre každého kto si ju chce zahrať), ale aj hľadisko ďalšieho vývoja (zvoliť najnovšie technológie, ktoré sa ďalej rozširujú a vyvíjajú). Pri voľbe správnej stratégie zvažíme formu prehliadačovej hry pre dvoch hráčov, aby bola naša hra čo najzaujímavejšia, pritiahla čo najviac detí a naučila ich spolupracovať. Praktickým výstupom bakalárskej práce bude funkčná verzia hry.

1 Analýza

1.1 Úvod do problematiky

Pri edukačných hrách je veľmi dôležitým parametrom dostupnosť hry, aby si hru mohol zahrať každý. V dnešnej dobe majú už internet všetci. Preto naša edukačná hra bude online dostupná pre každého. Hráč sa pohodlne pripojí na webstránku a hrá, nemusí si hru sťahovať na disk do počítača. Hra musí byť zaujímavá a zábavná. Naša hra bude „multiplayer“, určená pre dvoch hráčov.

V hre budeme chcieť hráčovi priblížiť niektoré zaujímavé riešenia z oblasti elektroniky a fotoniky. Vysvetliť mu základné pojmy a ukázať, kde sa prakticky elektronika a fotonika využívajú.

Hra má byť naprogramovaná pomocou objektovo orientovaného programovania. Táto metóda programovania je na programovanie hier často používaná, pretože zlepšuje prehľadnosť kódu a znovupoužitie, čo nám pri veľkom projekte veľmi pomôže.

1.2 Technológie

1.2.1 Adobe Flash

Adobe Flash je program, ktorý poskytuje spoločnosť *Adobe*. Môžeme ním vytvárať vektorové a bitmapové animácie, obohatené o zvukové efekty a interaktivitu (návštevník môže zasahovať do priebehu animácie). Tieto animácie môžu byť použité ako súčasť WWW stránok, môžeme ich exportovať do formátu GIF, EXE alebo ako animáciu. (1).

Pre hranie hier vyvinutých pomocou *Flash* potrebujeme inštalovať plugin *FlashPlayer*.

Na programovanie vo *Flash* sa využíva *ActionScript*. Je to výkonný objektovo orientovaný programovací jazyk s online sieťovými nástrojmi a podporou 3D grafiky. *FlashPlayer* nie je podporovaný väčšinou mobilných a tabletových prehliadačov ale flash hry možno exportovať ako aplikácie pre iPhone, iPad, Android (2).

Výhodou *Flashu* je dátová nenáročnosť vďaka vektorovej grafike a možnosť jednoduchej tvorby rozsiahlych animácií a hier. Ďalšou výhodou je práca s hudbou a zvukom. Hráčom nie je k dispozícii zdrojový kód hry, preto ho nemôžu upravovať ani ukradnúť, čo je veľkou výhodou z hľadiska ochrany a bezpečnosti oproti *HTML5* (3).

V roku 2011 *Adobe* oznámilo, že končí s vývojom ich *Flash* prehrávača pre mobilnú platformu ako napríklad *Android* a namiesto toho svoje úsilie nasmeruje do vývoja aplikácií *Adobe AIR* pre Apple iOS AppStore. Zmena stratégie bola oznámená v článku na oficiálnom blogu spoločnosti, v ktorom Danny Winokur, viceprezident pre klientov *Flash* platformy povedal, že spoločnosť sa bude agresívnejšie zameriavať na vývoj *HTML5* platformy (4).

Adobe Flash je považovaný za nebezpečný a aplikácie v ňom vytvorené sú veľmi zraniteľné, preto prehliadač Google Chrome začal vo verzii 55 automaticky blokovať obsah prehrávaný cez *Adobe Flash* a prednosť dostal *HTML5* (5). Neskôr podstúpili tento krok aj ďalšie prehliadače. Posledným prehliadačom, ktorý začal blokovať *Adobe Flash* je Mozilla Firefox (6).

1.2.2 HTML5

HTML5 je najnovším štandardom značkovacieho jazyka používaného na popisovanie obsahu a vzhľadu webových stránok. Bol vyvinutý s cieľom vyriešiť problémy s kompatibilitou štandardu *HTML4*. Jeden z najväčších rozdielov medzi *HTML5* a predchádzajúcimi štandardmi je to, že staršie verzie vyžadujú plugíny a API. Preto sa webová stránka, ktorá bola vyvíjaná a testovaná v nejakom prehliadači nemusí rovnako zobrazovať v inom prehliadači. *HTML5* poskytuje jedno spoločné rozhranie, aby sa mohli prvky načítat ľahšie. Jedným z cieľov pri návrhu *HTML5* bola podpora pre multimédiá na mobilných zariadeniach. Preto boli zavedené nové prvky ako *video*, *audio*, *canvas* (7).

Prvý návrh *HTML5* bol zverejnený v januári 2008.

S nástupom poslednej verzie *HTML* sa používa označenie *HTML5* pre celý balíček, do ktorého okrem značkovacieho jazyka *HTML* na vytváranie štruktúry dokumentov, patria kaskádové štýly *CSS*, čo je samostatný jazyk na štylovanie dokumentov, tak *JavaScript*, ktorý je taktiež samostatným skriptovacím jazykom na strane užívateľa (8).

1.2.3 CSS

CSS je skratka pre kaskádové štýly. Kaskádové štýly sa používajú na štylovanie rozloženia a vzhľadu webových dokumentov. Môžu byť použité na definovanie farby písma, veľkosti tabuliek a mnoho ďalších aspektov webových dokumentov, ktoré boli predtým definované iba pomocou *HTML* (9).

1.2.4 JavaScript

JavaScript je samostatný interpretovaný programovací alebo skriptovací jazyk od *Netscape*. *JavaScript* beží na strane klienta. Jeho pôvodný názov bol *Mocha*. V rokoch 1996-1997 začali na *JavaScript*-e pracovať *ECMA* a bol vydaný oficiálny štandard *ECMA-262*. Najnovším štandardom je *ECMAScript6* ktorý vyšiel v roku 2015. Tento štandard okrem iného priniesol podporu pre OOP. *JavaScript* sa využíva pri programovaní hier v *HTML5* (10).

1.2.5 Node.js

Node.js je vývojová platforma vyvinutá na *Google V8 JavaScript* engine. Kým *JavaScript* väčšinou beží na strane klienta, *Node.js* je zamerané na vývoj aplikácií na strane servera. *Node.js* je určený na spustenie *JavaScript*-u na vyhradenom serveri HTTP a využíva jedno vlákno.

Tieto aplikácie sú riadené udalosťami (event-based) a bežia asynchrónne. Kód postavený na *Node.js* nepoužíva tradičný model prijímania, spracovávania, posielania, čakania, prijímania. Namiesto toho, *Node.js* spracováva prichádzajúce požiadavky v neustálom slede udalostí a posiela požiadavky menšieho rozsahu jednu po druhej, bez toho, aby čakal na reakciu. Jednou z hlavných výhod *Node.js*, podľa svojho tvorca Ryana Dahla, je to, že neblokuje vstupy / výstupy (I/O) (11).

1.2.6 Socket.io

Socket.io je vysokovýkonná a flexibilná na strane servera (server-side) a na strane klienta (client-side) knižnica pre *WebSocket*, ktorá zabezpečuje komunikáciu v reálnom čase (real-time) v prehliadači (12).

Socket.io je *JavaScript*-ová knižnica pre webové aplikácie bežiacie v reálnom čase. Umožňuje obojsmernú komunikáciu medzi webovým klientom a serverom. Má dve časti. Knižnicu na strane klienta, ktorá pracuje v prehliadači a knižnicu na strane servera pre *Node.js*. Obe knižnice majú skoro identické API. *Socket.io* tak ako *Node.js* je riadená udalosťami. *Socket.io* primárne používa *WebSocket* protokol s dopytovaním ako núdzovou možnosťou a zároveň poskytuje rovnaké rozhranie. Hoci môže byť použitý ako jednoduchá obálka pre *WebSocket*, ponúka mnoho ďalších funkcií, vrátane vysielania na

viaceré výstupy, ukladanie dát spojené s každým klientom a asynchrónne I / O. Ľahká inštalácia do *Node.js* (13).

1.2.7 Použité technológie

Pre vývoj sme sa rozhodli využiť *HTML5*. Je to momentálne najnovší a najperspektívnejší nástroj na vytváranie online hier. Na túto technológiu prechádza čoraz viac webových stránok, už na ňu prešli napríklad aj YouTube.com alebo Twitch.tv. *HTML5* nepotrebuje žiadne rozšírenia ako napr. *Flash*, ktorý potrebuje *FlashPlayer* a podporujú ho všetky z najpoužívanejších prehliadačov na trhu. Keďže je veľmi populárny, objavuje sa na internete aj veľa návodov a zdrojov, čo významne uľahčuje učenie. Keďže má byť naša hra naprogramovaná pomocou OOP, rozhodli sme sa použiť *ECMAScript6*. Tento najnovší štandard podporuje OOP. V našej hre potrebujeme spustiť *JavaScript* aj na strane servera. Na tento účel sme sa rozhodli využiť *Node.js*, ktorý je rýchly, výkonný a ľahko sa inštaluje na server. Keďže naša hra bude pre dvoch hráčov, musíme zabezpečiť komunikáciu medzi klientmi a serverom. To sme sa rozhodli uskutočniť pomocou *Socket.io*, čo je výkonná knižnica pre *WebSocket* a ľahko sa používa.

1.3 Objektovo orientované programovanie

Objektovo orientované programovanie (OOP) je programovací model vývoja softvéru využívajúci dátové štruktúry nazývané objekty a ich vzájomné interakcie. Objekty majú svoje vlastnosti a metódy, pomocou ktorých objekt vykonáva činnosti, na ktoré bol naprogramovaný (14). Základom je dátový typ trieda. Trieda je množina objektov s určitými vlastnosťami. Samotná trieda nedefinuje konkrétne objekty triedy, iba udáva aké vlastnosti bude mať každý objekt tejto triedy.

Tromi základnými piliermi OOP sú zapuzdrenie, dedičnosť a polymorfizmus.

Zapuzdrenie (encapsulation) znamená, že každý objekt danej triedy sa stará o svoje dáta a manipuluje s nimi iba on sám. Iné objekty s ním komunikujú pomocou metód.

Dedičnosť (inheritance) nám umožňuje odvodiť triedu od inej triedy. Odvodená trieda zdedí všetky vlastnosti a metódy pôvodnej triedy.

Polymorfizmus (polymorphism) nám zaručuje to, že ak v potomkovi definujeme metódu s rovnakým názvom a parametrami ako je definovaná v predkovi, túto metódu prekryjeme. Pri volaní sa potom vždy zavolá metóda od potomka (15).

Kód napísaný týmto spôsobom je prehľadnejší a znovu použiteľný.

Existuje veľa programovacích jazykov používajúcich princíp OOP, sú to napr. *C++*, *C Sharp*, *Java*, *PHP*. Najnovší štandard *ECMAScript6* taktiež podporuje OOP.

2 Opis riešenia

2.1 Špecifikácia požiadaviek

Naša hra má spĺňať tieto požiadavky:

- a) má byť naprogramovaná pomocou OOP
- b) bude voľne dostupná a voľne šíriteľná v rámci popularizácie vedy a techniky
- c) má pútavým spôsobom uviesť hráčov do problematiky zaujímavých technických riešení z oblasti elektroniky a fotoniky
- d) má obsahovať viacero úrovní s lineárne rastúcou obtiažnosťou
- e) súčasťou hry budú jednoduché hlavolamy a zdroje nebezpečia

2.1.1 Cieľová skupina

Našou cieľovou skupinou sú žiaci základných a stredných škôl.

Vo vyšších ročníkoch základnej školy sa žiaci začínajú rozhliadať po strednej škole. Títo žiaci často ešte nevedia presne čo by ďalej chceli študovať a robiť. Naša edukačná hra má vzbudiť záujem o elektroniku a fotoniku u týchto detí. Ukázať im, kde sa elektronika a fotonika v praxi využíva a ako to funguje. To by mohlo pomôcť nasmerovať deti na štúdium na technických školách.

2.2 Návrh

Pri návrhu riešenia sme sa zamerali na to, aby bola naša hra čo najzaujímavejšia, najzábavnejšia a vedela upútať pozornosť detí.

V našom návrhu hry „Tajmostvá elektroniky a fotoniky“ sme sa rozhodli dať prednosť hre pre dvoch hráčov (multiplayer) pred hrou pre jedného hráča (singleplayer). Hráč tak môže zdieľať zážitok z hry so svojim spoluhráčom. Vo väčšine hier, ktoré sú pre viac hráčov, stoja hráči proti sebe. V našej edukačnej hre sú hráči kolegami, spojencami na ceste za splnením svojho spoločného cieľa. Týmto chceme v deťoch podporiť spoluprácu a tímového ducha. To, že je hra pre dvoch hráčov prinesie aj istú dávku zodpovednosti a dôležitosti hráčovi, keďže bez neho nebude možné hru prejsť.

Chceme deti zaujať a udržať pri našej hre až do konca. Aby sa nám tento cieľ podarilo dosiahnuť, potrebujeme dať hráčom motiváciu hru dokončiť. To by mal zabezpečiť príbeh.

Všetky úlohy v našej hre priamo súvisia s príbehom. Tak má hráč pocit, že naozaj robí niečo zmysluplné. Odmenou za splnenie všetkých úloh je ukončenie príbehu so šťastným koncom.

Ďalším, veľmi dôležitým predpokladom úspešnej hry je dizajn a grafika. Prvé, čo hráč uvidí, keď spustí hru, je grafika. V tomto momente netuší o čom hra je, čo sa v nej odohráva avšak práve tento prvý dojem z hry môže rozhodnúť o tom, či hráč hru vypne alebo bude pokračovať.

Podľa špecifikácie požiadaviek má byť naša edukačná hra voľne dostupná a voľne šíriteľná. Preto sme sa rozhodli realizovať hru ako prehliadačovú aplikáciu. Vďaka jednoduchému prístupu k hre si bude môcť hru zahrať každý, kto má prístup na internet a internetový prehliadač. Ďalšou požiadavkou je to, že hra má byť naprogramovaná pomocou OOP. Po analýze dostupných technológií sme sa rozhodli využiť *HTML5*. Je to momentálne najperspektívnejšia technológia na vytváranie prehliadačových aplikácií a je podporovaná vo všetkých najrozšírenejších prehliadačoch. Súčasťou programovania v *HTML5* je aj *JavaScript (ECMAScript)*. Keďže *ECMAScript6* podporuje OOP je vhodným jazykom pre tvorbu našej hry. Pre programovanie hier v *HTML5* existuje množstvo enginov. My sme sa však rozhodli naprogramovať hru bez použitia enginu. Výhodou je to, že nás nebude nič obmedzovať, netreba sa učiť používať engine a máme lepší prehľad čo ako v kóde funguje.

2.2.1 Návrh príbehu

Ak chceme, aby bola hra zábavná potrebujeme dať hre zmysel. Hráča uvedieme do hry pomocou príbehu.

V našej hre budú hlavnými hrdinami postavy Elektrón a Fotón. Títo dvaja hrdinovia pristávajú na mesiaci a ich misiou je vyťažiť novoobjavený nerast, ktorý sa nachádza na mesiaci. Avšak na mesiac padne meteorit a poškodí ich pristávací modul. Na module je poškodený obvod so solárnymi panelmi, laserová komunikácia a dvere. Elektrón a Fotón musia využiť svoje schopnosti z elektroniky a fotoniky, aby na stanici pozbierali predmety potrebné na opravenie porúch na module a opravili ho. Taktiež musia splniť misiu a vyťažiť nerast, pre ktorý prišli. Keď opravia modul a splnia misiu odlietajú naspäť na Zem.

2.2.2 Návrh úrovní

Hra sa bude odohrávať na mesiaci a bude obsahovať 3 úrovne.

Hlavným cieľom prvej úrovne je potreba opraviť obvod so solárnymi panelmi, aby mal modul energiu. Preto budú musieť hráči pozbierať na stanici predmety, potrebné na to, aby poskladali nový obvod so solárnymi panelmi. K tomu, aby sa dostali ku všetkým predmetom potrebným na túto úlohu, budú musieť najprv otvoriť dvere na stanici vyriešením minihry „LED pexeso“. V tejto minihre musí každý z hráčov nájsť 4 páry. Pár v tejto hre predstavuje jednu kartičku so zasvietenou LED diódou a druhú kartičku s obrysom LED diódy, v ktorom sú informácie o danej dióde - materiál, z ktorého je dióda vyrobená a vlnová dĺžka svetla, ktoré dióda vyžaruje. Hráči môžu kliknúť na tlačidlo informácie „i“ a zobrazí sa im nápoveda k úlohe, kde si môžu prečítať zaujímavé informácie o materiáloch, vlnových dĺžkach, napätiach a farbách LED diód. Keď hráči úspešne nájdú všetky páry, otvoria sa im dvere do stanice. Na stanici pozbierajú predmety avšak jeden predmet sa nachádza za zamknutými dverami. Nad týmito dverami svieti RGB LED dióda červeným svetlom. Dvere sa dajú otvoriť iba ak svieti táto dióda zeleným svetlom. Aby mohli hráči vstúpiť do miestnosti musia v minihre zmeniť veľkosti odporov na jednotlivých diódach v RGB LED dióde tak, aby RGB LED dióda svietila ako zelená. Keď majú hráči zozbierané všetky potrebné predmety zo stanice, musia sa vrátiť naspäť k modulu a zapojiť obvod so solárnymi panelmi pomocou týchto predmetov. Keď úspešne zapoja tento obvod, prvá úroveň skončí a po prechodovej animácii začína druhá úroveň.

V druhej úrovni musia hráči opraviť laserovú komunikáciu. Preto sa potrebujú vrátiť na stanicu a zozbierať ďalšie predmety, potrebné na zostavenie obvodu s laserovou komunikáciou. Pri zbieraní predmetov opäť prichádzajú k zamknutým dverám, ktoré treba otvoriť. Ak chcú dvere otvoriť budú musieť prejsť ďalšou minihrou. V tejto minihre musia roztriediť pole vlastností laserov do košov. Do každého z košov musia vložiť správne využitie, výkon, vlnovú dĺžku a svetlo lasera. Keď vyplnia všetky koše otvoria sa dvere a môžu pokračovať. Ak sa chcú dostať ku kľúčovému predmetu – laseru na komunikáciu, musia prejsť cez laserové pole. Ak sa ich dotkne laserový lúč, poškodí sa im oblek a vráti ich pred laserové pole, kde si prezlečú poškodený oblek a môžu znovu pokračovať. Keď úspešne prejdú týmto laserovým poľom, dostanú sa k laseru, ktorý potrebujú. Po zozbieraní všetkých predmetov sa musia vrátiť k modulu. V module zapoja obvod

s laserovou komunikáciou. Týmto druhá úroveň končí a po prechodovej animácii začína tretia, posledná úroveň.

V tretej úrovni musia hráči opraviť dvere na module a splniť misiu, čiže vytážiť nerast. K tomu, aby vytážili nerast sa potrebujú presunúť vozidlom ku ťažiacemu stroju. Vozidlo má však vybité batérie, preto musia zobrať akumulátory z vozidla a odnieť ich na stanicu nabiť. Pri nabíjaní akumulátorov sa spustí minihra. Úlohou je vypočítať časy nabíjania akumulátorov pri daných kapacitách a nabíjacích prúdoch. Po úspešnom nabití akumulátorov sa hráči presunú k vozidlu, kde nabité akumulátory vložia do vozidla. Teraz sa môžu vozidlom presunúť ku ťažiacemu stroju. K tomu, aby vytážili nerast musia splniť úlohu. Touto úlohou je vybrať spomedzi viacerých aplikácií len tie, ktoré patria LED diódam. Po vytážení nerastu sa hráči vrátia naspäť na stanicu. Na stanici zozbierajú predmety, ktoré nájdu a presunú sa k modulu, aby naložili nerast a opravili dvere. V tejto poslednej minihre, musia hráči zapojiť odpor do obvodu s LED diódou, uzavrieť vodičom obvod s fotodiódou a spolu odstrániť bariéru, ktorá bráni svetlu, aby preniklo na fotodiódu. Po úspešnom opravení dverí a splnení misie, posádka odlieta naspäť na Zem.

2.3 Implementácia

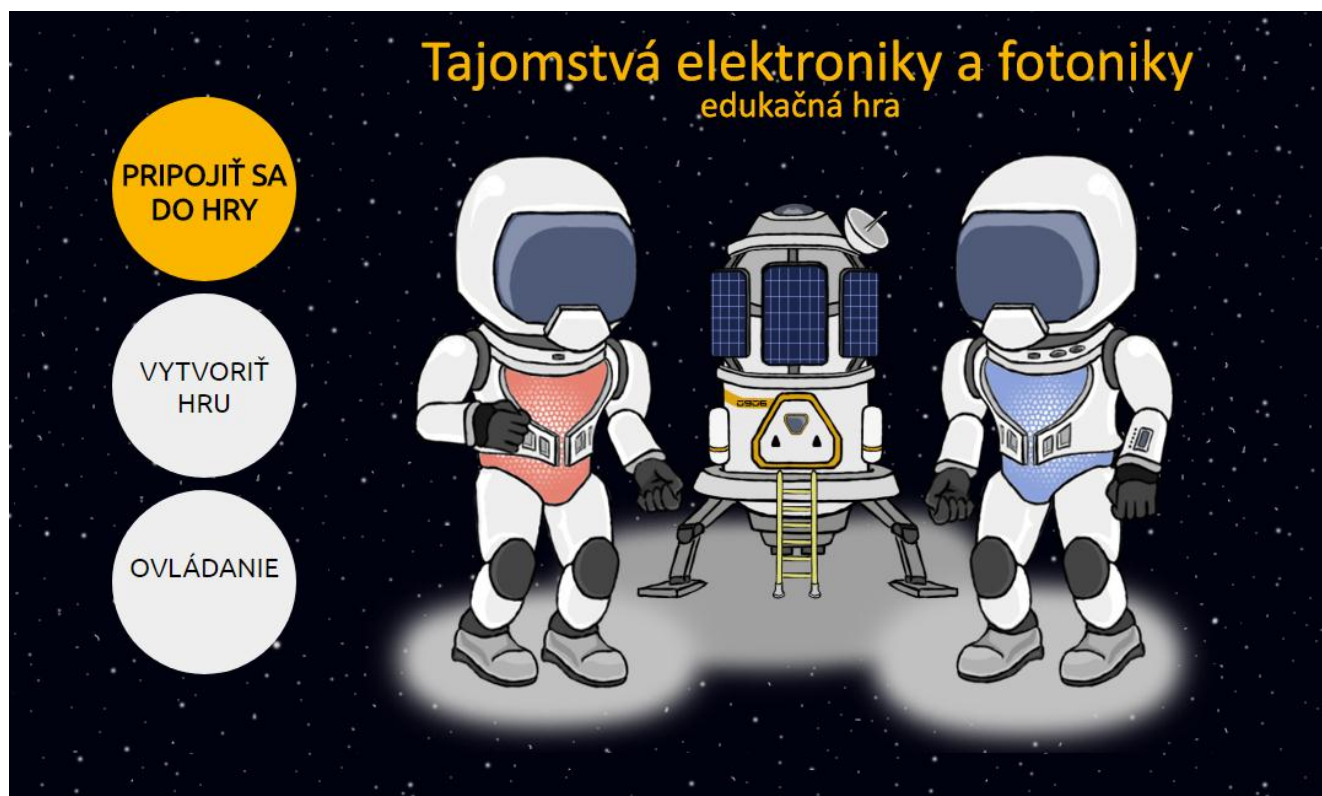
Keďže sme sa rozhodli pri tvorbe našej hry (*Obrázok 1*) nevyužiť žiadny engine, musíme si celú funkcionality a všetky triedy naprogramovať od základu.

Naša hra je pre viac hráčov, preto sme naprogramovali stranu klienta i stranu servera. Komunikácia medzi klientom a serverom je realizovaná pomocou *WebSocket*-ovej knižnice *socket.io*. Táto technológia je založená na posielaní správ medzi klientom a serverom.

Hráč sa po zadaní URL adresy dostáva na *HTML* stránku *index.html*. Na tejto stránke je vložený *HTML5* element *canvas* a sú tu importované všetky *JavaScript*-ové súbory. Formát všetkých elementov na tejto stránke je definovaný pomocou *CSS*.

Pri pripojení sa klienta na túto stránku sa v jednoduchom scripte vytvorí *socket* a objekt triedy *Game* a zavolá metódu *loadImages()*, ktorá načíta všetky obrázky, ktoré budú v hre vykresľované.

```
var socket = io();  
var game = new Game(socket);  
game.loadImages();
```



Obrázok 1 - Úvodná stránka a menu

2.3.1 Server

Na strane servera sú naprogramované dve triedy.

Prvou je trieda *Room*. Je logické, že trieda *Room* (miestnosť) musí byť naprogramovaná na strane servera. Objekty tejto triedy musia byť uchované aj keď klient ukončí hru a preto musia byť umiestnené na serveri. Táto trieda v sebe uchováva informácie ako je ID, názov miestnosti, heslo a pole hráčov, ktorí sú v nej pripojení. Toto pole je dvojrozmerné a nachádza sa v ňom *socket.id* hráča a typ postavy aký si zvolil. Trieda *Room* obsahuje aj metódu *isHere()*, ktorá zistí či sa v tejto miestnosti nachádza hráč so zadaným *socket.id*.

Druhou triedou je *Server*. Táto trieda obsahuje metódu *start()*. Po zavolaní tejto metódy je server spustený a počúva na danom porte. Ak príde nejaká správa od klienta server na ňu reaguje. Podľa toho aká správa príde, volá potrebné metódy. Napríklad ak príde správa „create room“, vytvorí novú miestnosť, čiže objekt triedy *Room* a pripojí do nej hráča.

```
// on CREATE ROOM
socket.on('create room', (data) => {
    var newRoom = new Room(data.id, data.name, data.password,
socket.id, data.charType);
    this.rooms.push(newRoom);
    socket.join(data.id);
});
```

2.3.2 Menu

Všetku funkcionálnosť spojenú s menu na úvodnej stránke má na starosti trieda *Menu*. V menu si hráč môže vybrať z možností „pripojiť sa do hry“, „vytvoriť hru“ a „ovládanie“.

Ak si hráč zvolí „vytvoriť hru“ (*Obrázok 2*) zavolá sa metóda *showForm()* a zobrazí sa okno s formulárom. Hráč vyplní pole „názov miestnosti“ a vyberie si postavu. Môže si vybrať medzi postavou Fotóna a Elektróna, ktorí sú hlavnými hrdinami našej hry. Ak hráč chce, aby jeho vytvorená miestnosť bola chránená heslom, vyplní aj pole „heslo“ avšak toto pole nie je povinné. Po kliknutí na tlačidlo „vytvoriť“ sa zavolá metóda *createRoom()*, ktorá odošle požiadavku pre vytvorenie miestnosti na server.

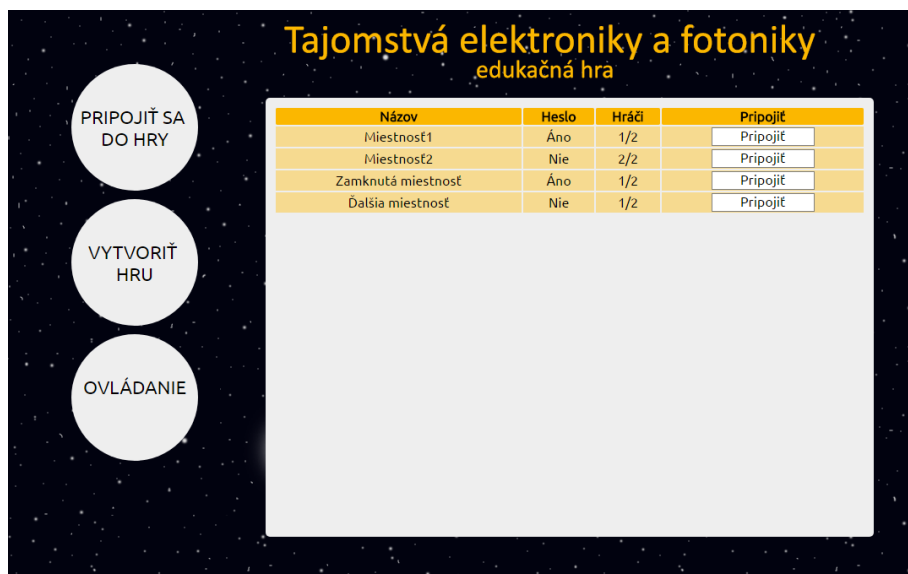
Ak si hráč v menu zvolí „pripojiť sa do hry“ (*Obrázok 3*) zavolá sa metóda *listRooms()*. Táto metóda odošle požiadavku na server, aby vrátil pole existujúcich miestností. Po tom ako server odpovie a vráti pole miestností, táto metóda vytvorí tabuľku s miestnosťami, ktorá sa zobrazí hráčovi. V tabuľke sa vytvorí ku každej miestnosti aj tlačidlo „pripojiť“, ktorým sa hráč môže pripojiť do danej miestnosti. Po kliknutí na toto tlačidlo sa zavolá metóda *joinRoom()*, ktorá odosiela na server požiadavku o pripojenie do danej miestnosti. Ak nie je miestnosť chránená heslom a nie je plná, pripojí hráča do miestnosti. Ak server odošle požiadavku na heslo zavolá sa metóda *createPasswordBox()*. Táto metóda vytvorí okno, do ktorého hráč zadá heslo. Pri odoslaní hesla je volaná metóda *sendPassword()*, ktorá odosiela heslo na server. Ak je zadané heslo správne pripojí hráča do hry. Ak je nesprávne zavolá sa metóda *show_message()* a vypíše hráčovi, že zadal nesprávne heslo alebo je miestnosť plná.

Zvolením možnosti „ovládanie“ v menu, sa volá funkcia *showControls()*, ktorá zobrazí obrazovku s prehľadom kláves, ktorými sa hra ovláda.

Trieda *Menu* obsahuje metódu *receiveMessages()*. Táto metóda slúži na prijímanie správ od servera a ďalej volá metódy, podľa toho akú správu server poslal.



Obrázok 2 - Vytvorenie miestnosti



Obrázok 3 - Pripojenie sa do miestnosti

2.3.3 Hra

Celá naša hra je jeden objekt triedy *Game*. Obsahuje všetky objekty, ktoré sa v hre nachádzajú. Táto trieda zabezpečuje plynutie hry. Nachádza sa v nej metóda *gameLoop()*, ktorá spúšťa herný cyklus. Tento cyklus zabezpečuje pohyb hráča, odosielanie pozície

hráča na server, aplikuje sa gravitácia, zisťujú sa aktuálne kolízie a prekresľuje plátno (*canvas*). Všetky tieto veci sa robia v každej iterácii cyklu.

Dôležitou metódou, ktorú obsahuje trieda *Game* je *receiveMessages()*. Táto metóda prijíma správy od servera a ďalej volá potrebné metódy. Kým sa nepripoja obaja hráči do hry, na plátne je vypísaný text „Čaká sa na pripojenie spoluhráča“. Túto scénu má na starosti metóda *waiting()*, ktorá vykresľuje tento text, pokým nepríde správa zo serveru že sa pripojil spoluhráč. Keď sú už obaja hráči pripojení, volá sa metóda *load()*. Pomocou tejto metódy sa vytvoria všetky potrebné objekty. Medzi nimi aj objekt hráča (trieda *Player*), spoluhráča (trieda *Teammate*) a svet (trieda *World*).

2.3.4 Svet

Trieda *World* predstavuje v našej hre celý svet, ktorý sa odohráva na mesiaci. Obsahuje v sebe referencie na objekty hráča (trieda *Player*), spoluhráča (trieda *Teammate*) fyziky (trieda *Physics*) a úrovne (trieda *Stage*). Táto trieda obsahuje metódy ako napr. *collisionsUpdate()*, ktorá volá metódu triedy *Physics* - *collisionDetection()*. Detekuje aktuálne kolízie hráča a spoluhráča a upravuje pole kolízií *collisions* v objekte hráča a spoluhráča. Taktiež zabezpečuje gravitáciu volaním metódy *gravityApply()* triedy *Physics*.

Ďalšou funkciou tejto triedy je kontrola kolízií s objektmi nachádzajúcimi sa v hre. Pomocou metódy *pickableColls_check()* zisťujeme, či hráč kolидуje s nejakým predmetom, ktorý sa dá zobrať do inventára. Ak zistí, že áno vloží do atribútu *canPickUp* triedy *Player* referenciu na tento objekt.

Metóda *interactions_check()* kontroluje, či obaja hráči (hráč aj spoluhráč) kolidujú s nejakým interaktívnym predmetom a či sa na tento predmet viaže nejaká minihra.

Ďalej trieda *World* obsahuje metódy *laserColls_check()* a *stationColls_check()*, ktoré slúžia na kontrolu kolízií so stanicou a lasermi.

Poslednou dôležitou metódou nachádzajúcou sa v triede *World* je *switchStage()*. Táto metóda slúži na prejsť na ďalšiu úroveň.

2.3.5 Hráč

Trieda *Player* predstavuje hráča. Ako sme už uviedli vyššie objekt hráča je uložený v triede *World*. Sú tu implementované metódy pre ovládanie hráča, samotný pohyb hráča

po svete, odosielanie informácií o hráčovi na server (napr. pozícia) a vykresľovanie hráča na plátno.

Ovládanie má na starosti metóda *control()*. Táto metóda vytvára *eventListener*-y na udalosti ako sú stlačenie klávesy (*keydown*), pustenie klávesy (*keyup*) a kliknutie myšou (*click*). Vo svete (mimo minihier) sa ovláda myšou iba odstraňovanie vecí z inventára hráča, preto ak *eventListener* zachytí udalosť kliknutia myšou, kontroluje, či bolo kliknuté na nejaký predmet v inventári. Ak áno, odstráni tento predmet z inventára hráča a vytvorí ho vo svete. Pri zachytení udalosti *keydown* *eventListener*-om sa kontroluje aká klávesa bola stlačená. Ak je stlačená niektorá zo šípok (vľavo, vpravo, hore) nastaví sa do poľa *moves* hodnota *true* pre daný smer pohybu. Pole *moves* obsahuje tri hodnoty typu *boolean*, ktoré reprezentujú smer pohybu (vľavo, vpravo, skok). Ak je stlačená klávesa *enter*, kontroluje sa, či je sú hráči pri nejakom objekte, na ktorý je naviazaná minihra. Ak áno, spúšťa sa minihra. Ak je stlačená klávesa *spacebar*, zistí sa, či hráč môže zobrať nejaký predmet do inventára. Poslednou kontrolovanou klávesou pri *keydown* udalosti je klávesa *escape*. Ak je stlačená práve táto klávesa, zobrazí sa dialógové okno, či chce hráč ukončiť hru.

Pohyb hráča je sprostredkovaný metódou *moving()*. Táto metóda kontroluje pole *moves* a ak je hodnota na danej pozícii zodpovedajúcej smeru pohybu *true*, spúšťa metódy *moveLeft()*, *moveRight()* a *jump()*. V týchto metódach sa už len kontrolujú kolízie, ak je možné posunúť sa v danom smere tak sa zmení pozícia hráča.

Ďalšou, veľmi dôležitou metódou triedy *Player*, je *sendPosition()*. Táto metóda zabezpečuje odosielanie informácií o hráčovi na server. Odosielajú sa informácie ako je pozícia hráča (súradnice x,y), rýchlosť a smer pohybu.

Metóda *render()* zabezpečuje vykresľovanie a animáciu postavy hráča. Animácia je tvorená pomocou *sprite sheet*. Pri tejto technike animovania, sa vykresľuje vždy iba časť obrázka, na ktorom je jedna snímka z animácie. Pomocou funkcie *canvasu* - *drawImage()*, vieme nastaviť pozíciu, odkiaľ sa má obrázok vykresľovať a aká veľká časť sa má vykresliť. Trieda *Player* obsahuje premennú *animCount*, je to číslo, ktoré zvyšujeme v každej iterácii cyklu a modulujeme počtom snímok na obrázku (v prípade ak je hodnota *animCount* väčšia ako počet snímok na obrázku, začíname opäť od začiatku). Túto hodnotu následne vynásobíme šírkou vykresľovanej časti obrázku a dostávame pozíciu

začiatku vykresľovania na osi x. To nám zabezpečuje posúvanie začiatkovej pozície vykresľovania doprava a vytvára animáciu.

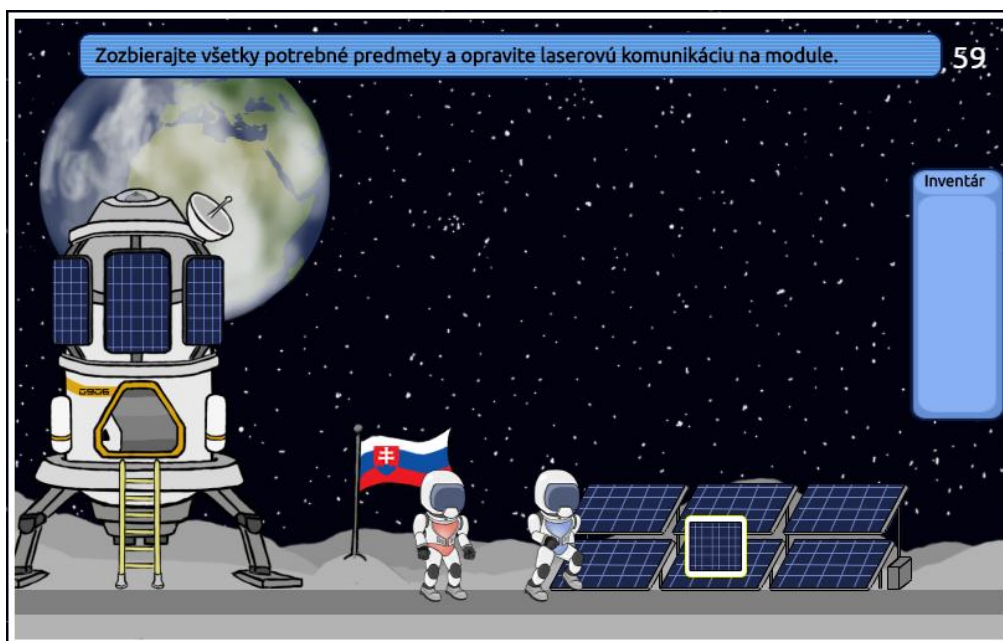
2.3.6 Úroveň

Úroveň reprezentuje trieda *Stage* (Obrázok 4). Objekt tejto triedy je uložený v triede *World*. Trieda *Stage* obsahuje pole *objects*. V tomto poli sú všetky objekty nachádzajúce sa v danej úrovni. Taktiež obsahuje pole *stageQuests*, v ktorom sú uložené všetky minihry tejto úrovne.

Metóda *load()* triedy *Stage*, zabezpečuje vytvorenie všetkých týchto objektov a minihier. Ako vstupný parameter vstupuje do tejto metódy číslo úrovne (1,2,3), ktorú chceme načítať. Podľa toho aké je toto číslo sa vytvoria objekty a minihry pre danú úroveň.

V tejto triede je implementovaná metóda *createGameObject()*, ktorá vytvorí nový objekt triedy *GameObject* do úrovne. Taktiež metóda *deleteObject()*, ktorá odstráni objekt z úrovne podľa názvu, ktorý je vstupným parametrom tejto metódy. Metóda *getObjectByName()* so vstupným parametrom *name* vráti objekt s daným názvom.

Metóda *render()* slúži na vykreslenie pozadia úrovne, ako aj všetkých objektov nachádzajúcich sa v poli *objects*.



Obrázok 4 - Druhá úroveň

2.3.7 Mini hry

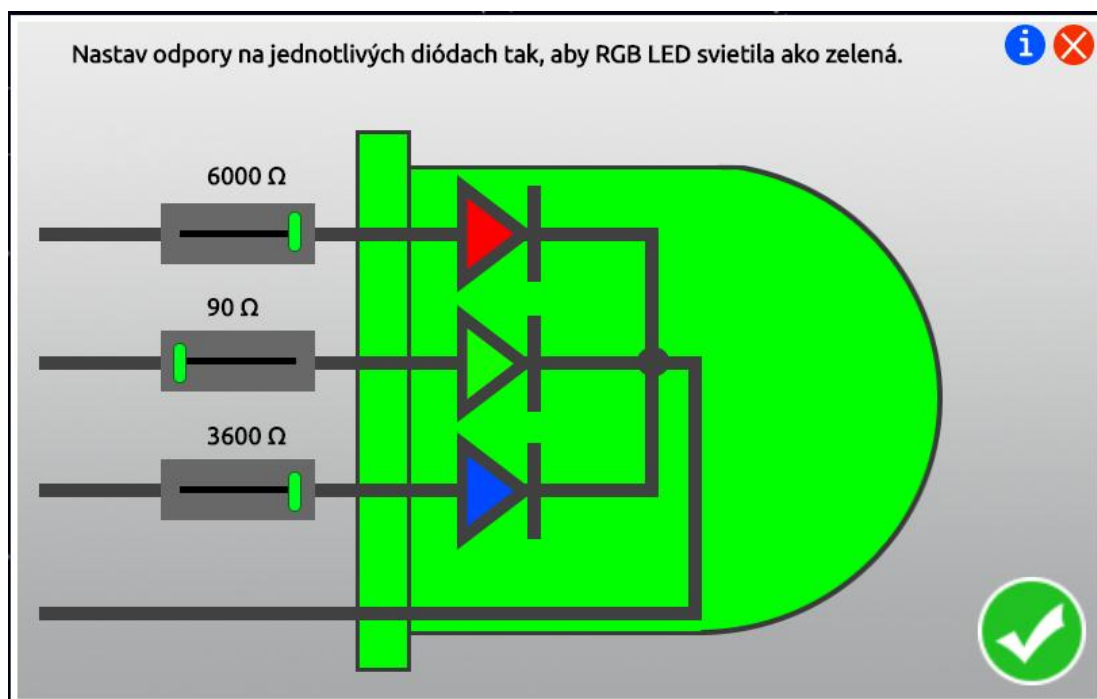
Všetky minihry v našej hre dedia triedu *MiniGame*. Táto trieda obsahuje základné, spoločné vlastnosti všetkých minihier. Všetky minihry obsahujú tlačidlo pre vypnutie minihry, tlačidlo pre potvrdenie riešenia, tlačidlo pre zapnutie nápovedy a samotnú nápovedu, čiže objekt triedy *InfoPage*. Ďalšími vlastnosťami, ktoré majú všetky minihry spoločné sú napr. názov minihry, názov objektu, na ktorý sa minihra viaže, referencia na objekt triedy *Canvas*, maximálny počet obrázkov za sekundu (fps), atď. V triede *MiniGame* je implementovaná metóda *go()*. Táto metóda slúži na spustenie minihry. Ukončí herný cyklus hry a spustí vlastnú metódu *gameLoop()*, ktorá rovnako ako v triede *Game* zabezpečuje plynutie hry - herný cyklus. V každej iterácii cyklu sa kontroluje, či sú splnené všetky úlohy v minihre a volá sa funkcia *render()*, ktorá prekresľuje plátno. Ak sú splnené všetky úlohy tlačidlo pre potvrdenie je aktívne. Ak je na toto tlačidlo kliknuté volá sa metóda *confirm()*. Táto metóda odosiela správu na server, že bola úloha splnená a spúšťa herný cyklus triedy *Game*.

Jednou z úloh v prvej úrovni je minihra LED Pexeso (*Obrázok 5*). Táto minihra je objekt triedy *Pexeso*, ktorá dedí triedu *MiniGame*. Pri vytvorení tohto objektu sa volajú metódy *createCards()* a *shuffle()*. Metóda *createCards()* slúži na vytvorenie objektov triedy *Card*, ktoré predstavujú karty pexesa. Metóda *shuffle()* má na starosti náhodné rozmiestnenie kariet po ploche. Ovládanie minihry je realizované pomocou metódy *control()*. Ak je nájdený pár volá sa metóda *pairFound()*. Táto metóda inkrementuje premennú, ktorá obsahuje počet nájdených párov a skontroluje, či sú nájdené všetky páry. Ak hráči nájdu všetky páry a kliknú na tlačidlo pre potvrdenie, zavolá sa metóda *win()*, ktorá ukončí herný cyklus minihry, spustí herný cyklus hry a otvorí dvere s ktorými je táto minihry previazaná.



Obrázok 5 – Minihra LED pexeso

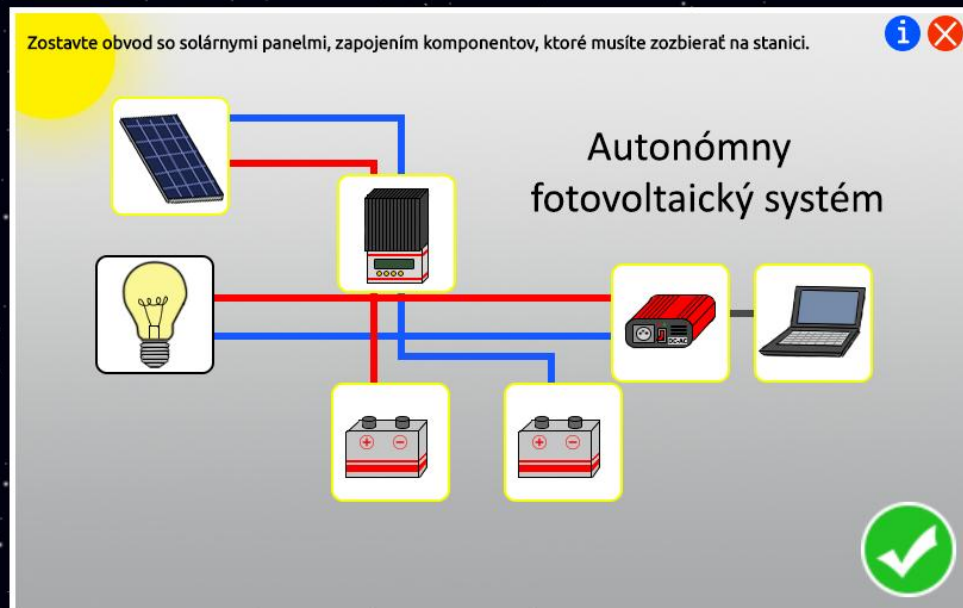
Ďalšou minihrou v prvej úrovni je úloha s RGB LED diódou (Obrázok 6). Táto úloha je objekt triedy *RgbLedQuest*, ktorá dedí triedu *MiniGame*. V tejto minihre je úlohou nastaviť odpory na jednotlivých diódach RGB LED diódy tak, aby svietila ako zelená. Ovládanie minihry má na starosti metóda *control()*. Farba RGB LED diódy sa mení v závislosti od nastavených hodnôt odporov. Keďže máme tri LED diódy a na každej z nich môžeme nastaviť tri rôzne hodnoty odporu, to znamená, že existuje 27 rôznych kombinácií nastavení odporov. Ak by sme chceli meniť pri každej kombinácii vykresľovaný obrázok, potrebovali by sme 27 rôznych obrázkov. Preto vykresľujeme diódu pomocou metód *canvas-u fillRect()*, *arc()* a *fill()*. Farbu výplne meníme podľa nastavených hodnôt odporov, s ktorými sa menia aj hodnoty farby RGB (red, green, blue). R podľa odporu nastaveného na červenej LED dióde, G podľa odporu nastaveného na zelenej LED dióde a B podľa odporu nastavenej na modrej LED dióde. Toto vykresľovanie RGB LED diódy sa realizuje v metóde *render()*, kde sa vykresľujú všetky obrázky tejto minihry.



Obrázok 6 - Minihra s RGB LED

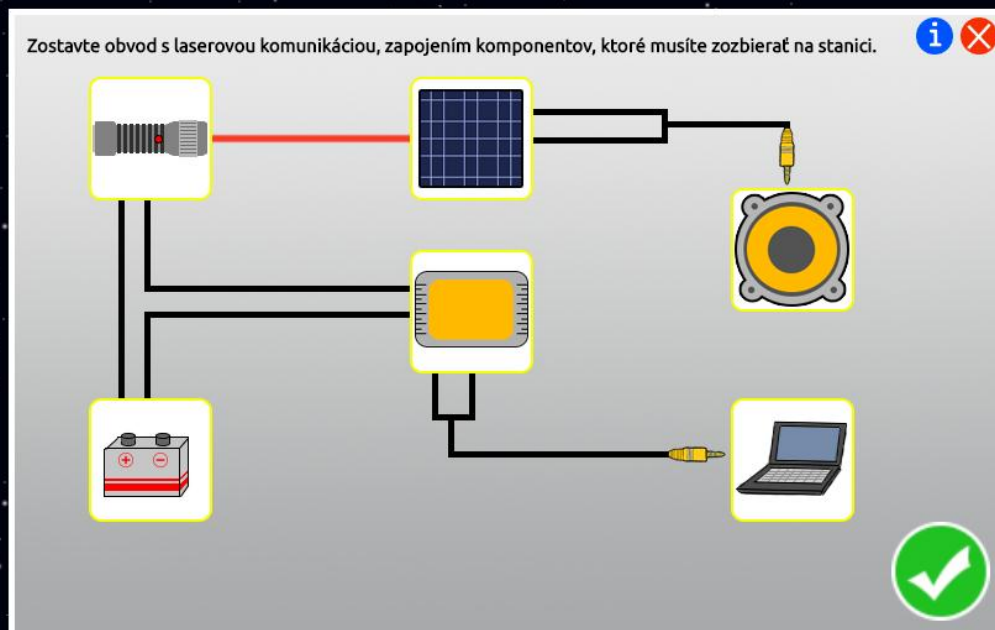
Poslednou minihrou v prvej úrovni je úloha, v ktorej majú hráči pomocou nazbieraných predmetov opraviť obvod so solárnymi panelmi (Obrázok 7). Táto minihra je objektom triedy *SolarPanelsRepairQuest*. V druhej úrovni sa stretneme s úlohou opraviť laserovú komunikáciu (Obrázok 8). Táto minihra je objekt triedy *LaserCommunicationRepairQuest*. Tieto dve minihry sú veľmi podobné, mení sa iba obvod a objekty, z ktorých ho skladáme. Preto je väčšina metód týchto minihier dedená z triedy *FixCircuitMiniGame*, ktorá ďalej dedí triedu *MiniGame*. Keď sa spustí minihra zavolá sa metóda *load()*, ktorá načíta predmety z inventára hráča do minihry. Potom hráči pomocou týchto predmetov skladajú obvod, či už so solárnymi panelmi alebo s laserovou komunikáciou. Minihra sa ovláda pomocou „Drag&Drop“. Ovládanie má na starosti metóda *control()*. Pri udalosti „mousedown“ sa kontroluje, či hráč drží myšku na nejakom predmete. Ak áno, uloží sa tento predmet do premennej *snappedObject*. Pri udalosti „mousemove“ sa mení pozícia chyteného predmetu v závislosti od polohy kurzoru myšky. Pri udalosti *mouseup* sa skontroluje, či bol uchytený nejaký predmet a ak bol, či bolo tlačidlo myšky pustené nad správnym miestom, kde predmet patrí. Ak týmto spôsobom zostavia hráči celý obvod a potvrdia tlačidlom pre potvrdenie, zavolá sa metóda *win()*. Tá ukončí herný cyklus minihry a spustí scénu príbehu (prechod medzi úrovňami).

Tajomstvá elektroniky a fotoniky edukačná hra



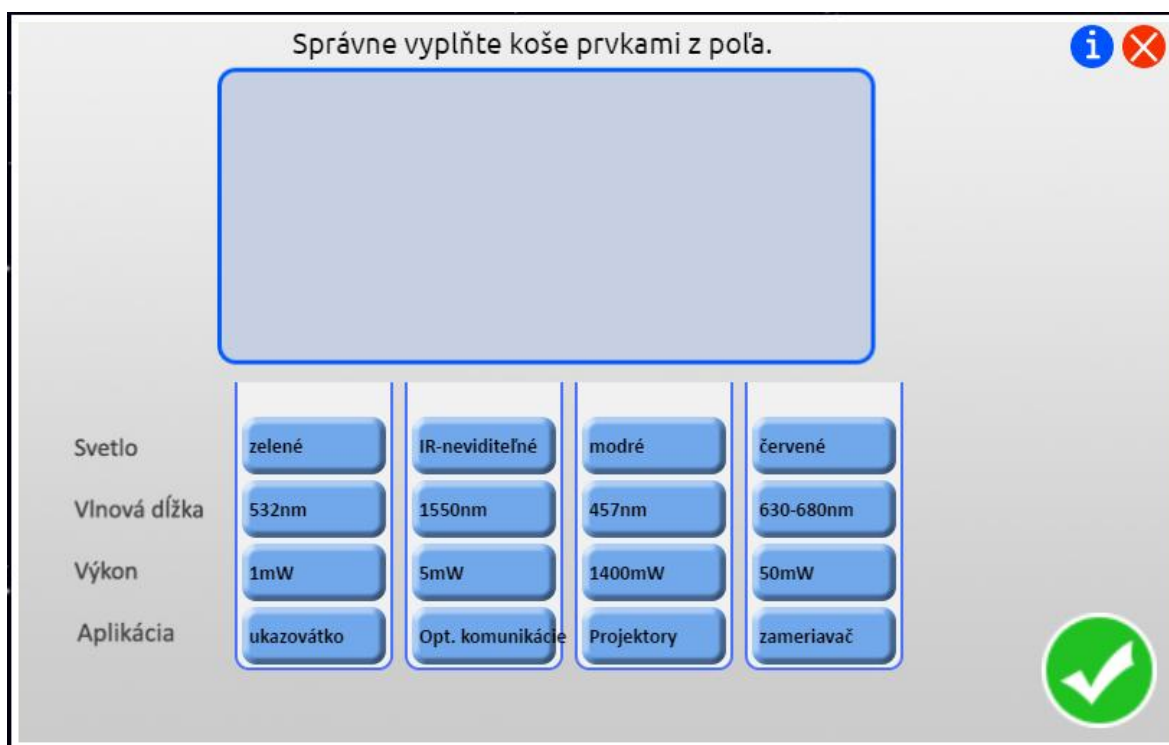
Obrázok 7 - Mini hra zapojenie solárnych panelov

Tajomstvá elektroniky a fotoniky edukačná hra



Obrázok 8 - Mini hra zapojenie laserovej komunikácie

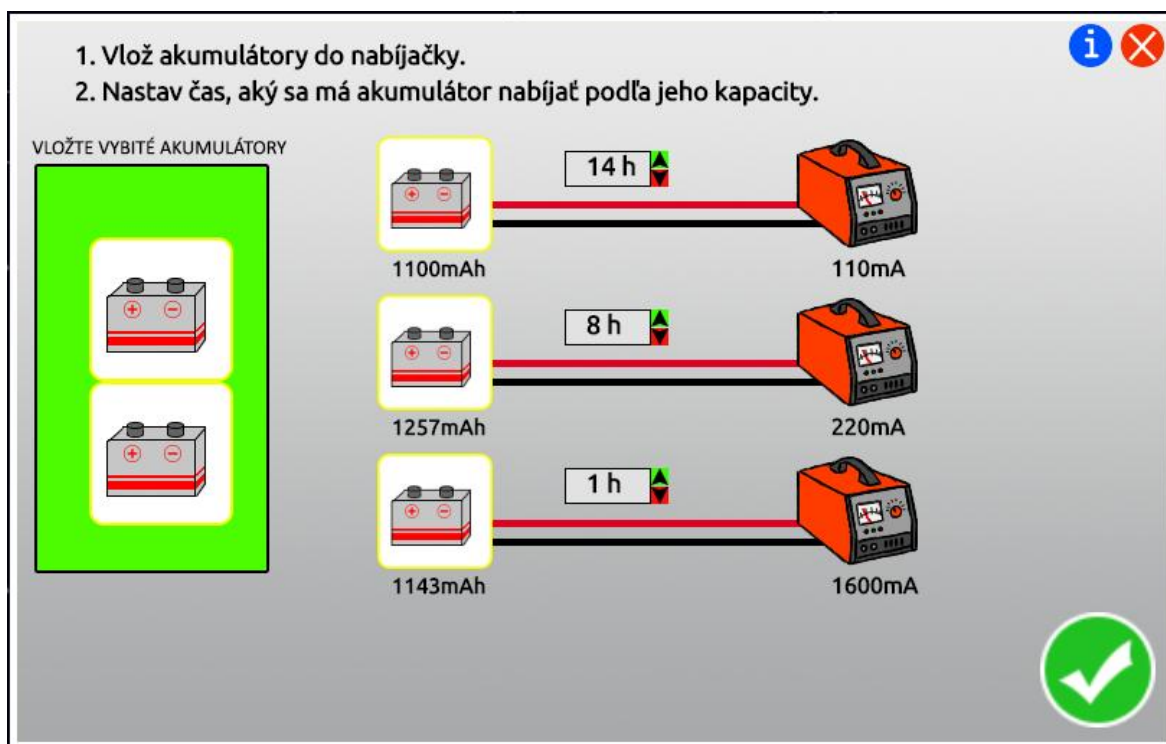
Ďalšou minihrou v druhej úrovni je úloha s triedením vlastností laserov do košov (Obrázok 9), podľa typu laseru. Táto minihra je objektom triedy *LaserTypesQuest*, ktorá dedí triedu *MiniGame*. Táto trieda obsahuje štyri koše, čo sú objekty triedy *Basket* a šesťnásť vlastností, ktoré sú objektami triedy *Item*. Pri vytvorení tejto minihry sa volajú metódy *createItems()* a *shuffle()*. Metóda *createItems()* sa postará o vytvorenie všetkých objektov vlastností a metóda *shuffle()* ich náhodne rozmiestni po ploche. Úlohou je, pomocou „Drag&Drop“ preniesť vlastnosť do správneho koša. Tieto vlastnosti však musia byť do košov vkladané v poradí aplikácia, výkon, vlnová dĺžka, svetlo. Ak je táto podmienka splnená zavolajú sa metódy *addItem()* a *nextItemTypeSet()* objektu triedy *Basket* do ktorého bola vlastnosť vložená. Metóda *addItem()* pridá objekt triedy *Item* do tohto koša a *nextItemTypeSet()* zmení typ vlastnosti, ktorý môže byť následne do koša vložený, napr. ak bola teraz vložená vlastnosť typu výkon, ďalší typ vlozenej vlastnosti musí byť vlnová dĺžka. Po roztriedení všetkých vlastností do správnych košov a potvrdení riešenia sa volá metóda *win()*.



Obrázok 9 - Minihra triedenie laserov

V tretej úrovni je potrebné dostať sa na vozidle k ťažiacemu stroju. Keďže sú akumulátory vo vozidle vybité je potrebné ich nabiť, pričom musia hráči prejsť minihrou

nabíjanie akumulátorov (Obrázok 10). Táto minihra je objekt triedy *AccumulatorChargeQuest*. Táto trieda obsahuje tri objekty triedy *Timer*. Na týchto časovačoch musia hráči nastaviť správny čas nabíjania akumulátora, podľa nabíjacieho prúdu nabíjačky a kapacity akumulátora. V tejto triede je implementovaná funkcia *calculate_time()*, ktorá vypočíta dobu nabíjania a tá sa potom porovnáva s tou, ktorú zadal hráč. Ak sú hodnoty na časovačoch nastavené správne, sú vložené do nabíjačky vybité akumulátory a hráči kliknú na tlačidlo potvrdiť akumulátory sa nabijú.



Obrázok 10 - Minihra nabíjanie akumulátorov

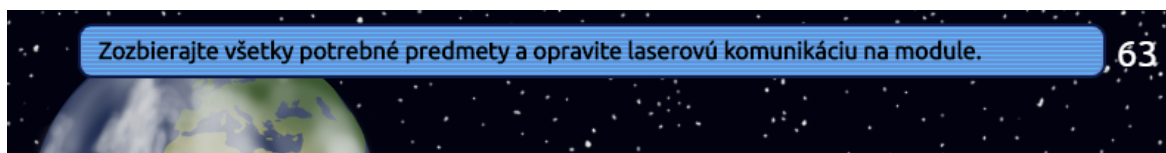
Minihra s vyberaním aplikácií LED diód je objekt triedy *ChooseCorrectQuest*. Ovládanie minihry zabezpečuje metóda *control()*. Minihra sa ovláda metódou „Drag&Drop“. Súčasťou tejto triedy je aj objekt triedy *Zone*. Tento objekt predstavuje oblasť, do ktorej hráči vkladajú aplikácie LED diód. Ak hráč vloží nejakú aplikáciu do tejto oblasti, zavolá sa metóda *isAllowed()* triedy *Zone*, ktorá skontroluje či je táto aplikácia správna. Ak vráti táto metóda *true* tak sa pomocou metódy *addObject()* vloží táto aplikácia do objektu triedy *Zone*. Trieda *Zone* obsahuje metódu *render()*, ktorá vykresľuje

aplikácie v nej vložené. Vykresľovanie celej minihry má na starosti metóda *render()*, ktorá je súčasťou triedy *ChooseCorrectQuest*.

Poslednou minihrou v našej hre je úloha, v ktorej je potrebné opraviť dvere na module. Táto minihra je objektom triedy *DoorRepairQuest*. V tejto minihre je cieľom opraviť obvod s LED diódou, obvod s fotodiódou a odtlačiť bariéru, ktorá sa nachádza medzi týmito dvomi obvodmi a zabraňuje svetlu z LED diódy preniknúť na fotodiódu. Opravenie obvodov je opäť realizované metódou „Drag&Drop“. Hráč musí presunúť predmet (vodič alebo odpor) na dané miesto v správnom obvode. Keď sú obvody opravené, hráči musia odtlačiť bariéru. Pri tomto úkone musia hráči spolupracovať. Keď spolu držia myšku na bariére, začne sa posúvať, až kým ju úplne neodsunú. Všetka táto funkcionality je implementovaná v metóde *control()*. Pri odtláčaní bariéry, keď hráč klikne na bariéru (objekt triedy *Barrier*) sa odosiela správa na server, ktorý ju odošle spoluhráčovi. Keď príde táto správa druhému hráčovi nastaví premennú *teammateRemoving* na *true*. Potom pri kliknutí hráča na bariéru ak je v tejto premennej uložená hodnota *true*, volá sa metóda *move()* objektu triedy *Barrier*. Tá zabezpečí posunutie bariéry. Po úspešnom odsunutí bariéry a stlačení tlačidla potvrdiť sa zavolá metóda *win()*, ktorá ukončí herný cyklus minihry a spustí poslednú scénu príbehu ako modul odlieta z mesiaca.

2.3.8 Rozmiestnenie elementov

V hornej časti obrazovky bude zobrazený informačný panel (*Obrázok 11*). Tento panel je objekt triedy *InfoBox* a slúži na zobrazovanie informácií hráčom, napr. aká je aktuálna úloha. V pravej časti obrazovky je vykreslený inventár hráča.



Obrázok 11 - Informačný panel

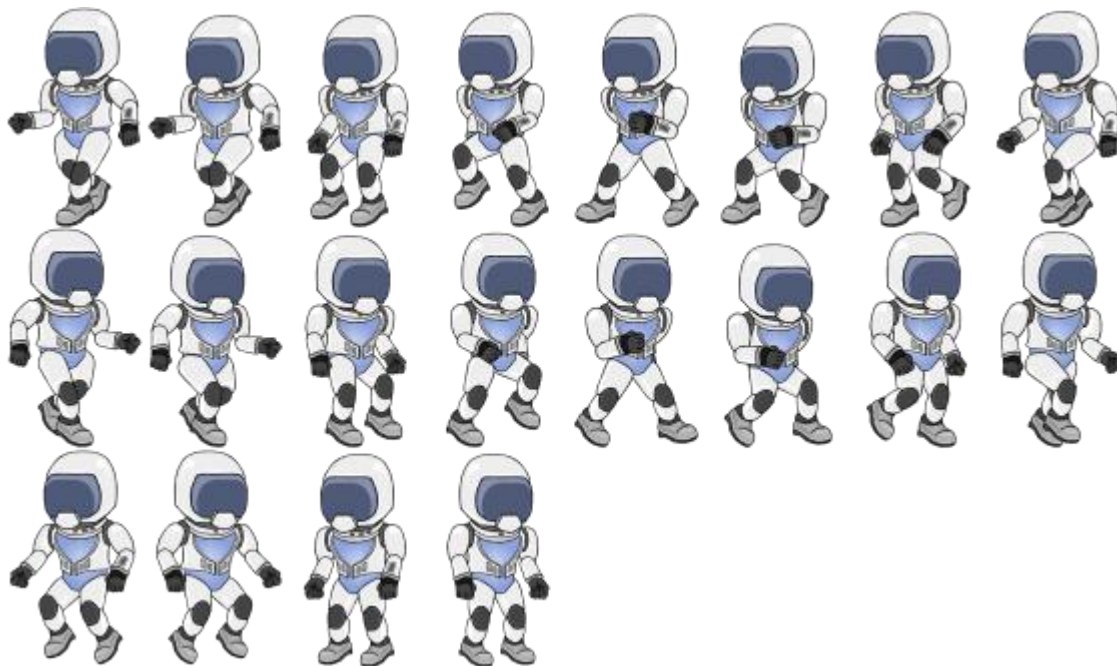
2.3.9 Ovládanie

Hráč sa môže pohybovať pomocou „šípok“ na klávesnici. Zobrať predmet môže pomocou klávesy *spacebar*. Vyhodenie predmetu z inventára je možné pomocou ľavého tlačidla na myške. Kliknutím na predmet v inventári (ktorý sa zobrazuje v pravej časti obrazovky) je predmet vyhodený z inventára. Ak sa obaja hráči nachádzajú pri predmete, na ktorý sa viaže minihra, tak pomocou klávesy *enter* môžu spustiť danú minihru. Minihry sa ovládajú pomocou myšky metódou „Drag&Drop“ alebo klikaním ľavým tlačidlom myšky. Hra sa dá ukončiť stlačením klávesy *escape*. Po jej stlačení sa zobrazí dialógové okno, či chce hráč hru ukončiť.

2.3.10 Tvorba grafiky

Všetky grafické podklady do našej edukačnej hry sme si vytvorili pomocou programu Gimp na základe vlastných návrhov. Gimp je považovaný za najlepší slobodný grafický nástroj na úpravu a tvorbu rastrovej grafiky.

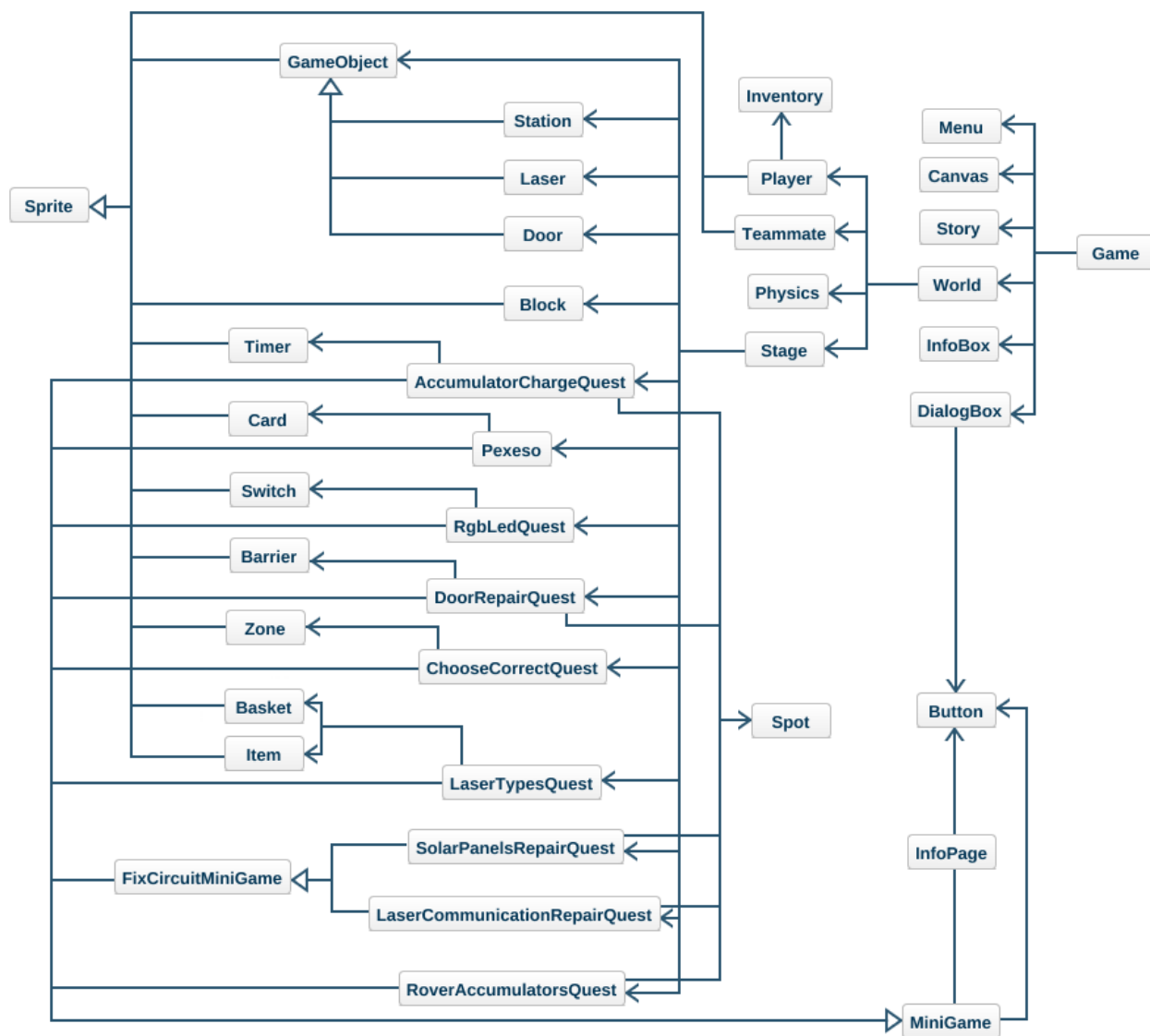
Animácie obrázkov sme robili pomocou techniky *sprite sheet* (Obrázok 12). V jednom obrázku sú všetky snímky animácie obrázku. V intervale meníme pozíciu odkiaľ obrázok vykresľujeme a to nám vytvára animáciu.



Obrázok 12 - Sprite sheet postavy Elektrón

2.3.11 Diagram tried

Jednotlivé triedy na strane klienta a ich vzájomné prepojenia môžeme zobraziť pomocou diagramu tried (Obrázok 13).



Obrázok 13 - Diagram tried na strane klienta

3 Zhodnotenie

Pri návrhu a vývoji našej edukačnej hry „Tajomstvá elektroniky a fotoniky“ sme sa stretli s otázkou bezpečnosti a ochrany zdrojového kódu voči manipulácii s ním. Keďže sme využili technológiu *HTML5*, všetky zdrojové kódy sú písané v jazyku *JavaScript*. Tieto zdrojové kódy môže hocikto vo svojom prehliadači prepísať alebo dopísať vlastný kód. Tak by si mohol tento hráč prepísať niektoré premenné ako napr. rýchlosť pohybu a ďalšie veci, ktoré by mali dopad na beh hry. Jedným spôsobom ako sťažiť takémuto hráčovi prepísanie kódu (napr. rýchlosti pohybu) je kontrola premenných na serveri. Aj táto kontrola sa však dá obísť. Ďalším spôsobom je zneprehľadniť kód pre klienta. To sa dá realizovať napr. takzvanou minifikáciou (minification), čo znamená odstránenie nepotrebných znakov bez zmenenia funkcionality. Zabrániť prepísaniu kódu sa však nedá, dá sa to len sťažiť, čo však nie je zámerom našej hry. Pokiaľ sa niekto inšpiruje našim kódom je to vítané v rámci popularizácie informatiky.

Ak by sme vyvíjali hru napr. v engine *Unity*, kód nie je tak ľahko zobraziteľný v prehliadači avšak na hranie takýchto hier je potrebný *Unity* plugin.

Keďže našou cieľovou skupinou sú žiaci základných a stredných škôl, nepredpokladáme, že by mali schopnosti na to, aby upravovali naše zdrojové kódy. A keďže je naša hra pre dvoch hráčov, ktorí spolu nesúperia ale spolupracujú, nemôžu týmto spôsobom poukázať hru iným hráčom, len sebe.

3.1 Testovanie

Naša edukačná hra bola testovaná v najnovších verziách prehliadačov Google Chrome, Mozilla Firefox a Opera. Na testovanie bol použitý notebook značky Lenovo, typ B590 s procesorom Intel Core i3-3110M 2,4GHz, grafickým procesorom nVidia GeForce GT 720M a 4GB RAM. Všetky testy prebehli bez problémov.

4 Technická dokumentácia

4.1 Nasadenie hry na server

Pre nasadenie hry na server je potreba inštalácie frameworku *node.js*, ktorý zabezpečuje spustenie *JavaScript*-u na strane servera. Ďalej je potrebné doinštalovať do *node.js* knižnicu *socket.io*, pomocou ktorej je realizovaná komunikácia medzi klientom a serverom. *Socket.io* sa dá nainštalovať cez konzolu, jednoduchým príkazom pomocou *npm*.

```
npm install socket.io --save
```

Npm je balíčkovací systém pre *node.js*, ktorý je súčasťou základnej inštalácie *node.js*. Potom sa už len konzolovým príkazom spustí script *server.js*, ktorý má byť spustený na strane servera.

```
node server.js
```

4.2 Popis najdôležitejších tried hry

4.2.1 Trieda Game

Celá naša hra je objekt triedy *Game*. Táto trieda v sebe uchováva celú funkcionálnosť hry. Môžeme sa z nej dostať ku všetkým objektom a ich metódam. Má 12 atribútov:

- 1) *socket* - zabezpečuje obojstrannú komunikáciu v reálnom čase
- 2) *menu* - referencia na objekt triedy *Menu* (dátový typ *Menu*)
- 3) *canvas* - referencia na objekt triedy *Canvas* (dátový typ *Canvas*)
- 4) *infobox* - referencia na objekt triedy *InfoBox* (dátový typ *InfoBox*)
- 5) *world* - referencia na objekt triedy *World* (dátový typ *World*)
- 6) *story* - referencia na objekt triedy *Story* (dátový typ *Story*)
- 7) *images* - obrázky namapované na názvy (dátový typ *2D array (string, Image)*)
- 8) *request* - *RequestAnimationFrame* číslo snímku, ktorým sa dá zrušiť animačný cyklus (dátový typ *DomHighResTimeStamp*)
- 9) *fps* - Počet snímok za sekundu (dátový typ *Integer*)
- 10) *lastCalledTime* - *Timestamp* posledného volania
- 11) *lastFrameTimeMs* - *Timestamp* poslednej snímky

12) *timestep* - interval vykresľovania snímok (datový typ *Double*)

Metódy triedy *Game*:

constructor(socket)

Do konštruktora triedy *Game* vstupuje jediný parameter a tým je *socket*. *Socket* je základným prvkom *socket.io*. Každý *socket* má svoje jedinečné ID a metódy pomocou ktorých komunikuje so serverom. *Socket*-y odosielajú a prijímajú správy.

V konštruktoze zadefinujeme a inicializujeme základné vlastnosti triedy *Game*.

loadImages()

Táto metóda vytvorí objekty typu *Image* pre všetky obrázky, ktoré sa v našej hre vyskytujú. Následne ich vkladá do dvojrozmerného poľa *images* a k nim priradí názov obrázka. Týmto spôsobom vlastne namapuje obrázok na názov, čiže *string*. Podľa tohto názvu neskôr vyberáme z poľa objekty typu *Image*.

load(roomid, charType)

Do metódy *load()* vstupujú ako parametre *roomid*, čo je ID miestnosti v ktorej sa nachádza hráč, a *charType*, ktorý nám určuje, aký typ postavy si hráč vybral (Elektrón alebo Fotón).

V tejto metóde definujeme vlastnosti tejto triedy. Vytvárame objekty tried *InfoBox*, *DialogBox*, *Player*, *Teammate* a *World*.

findImage(name)

Parametrom vstupujúcim do tejto metódy je *name*, čiže názov. Podľa tohto názvu metóda vyberie a vráti objekt typu *Image* z dvojrozmerného poľa *images*, ktorý je na tento názov namapovaný.

waiting()

Táto metóda je volaná keď sa čaká na pripojenie spoluhráča do hry. Vykresľuje obrázok, ktorý sa zobrazuje pri čakaní. Ak príde zo serveru správa „player join“ zavolá sa metóda *start()* objektu triedy *Story* a začne sa príbeh.

start()

Táto metóda spúšťa herný cyklus *gameLoop()*, metódu *receiveMessages()* a ovládanie hráča.

receiveMessages()

Služi na prijímanie správ od servera a volá ďalšie metódy, podľa typu správy ktorá prišla.

gameLoop()

Táto metóda spúšťa herný cyklus vytvorením *requestAnimationFrame* a v každej iterácii tohto herného cyklu volá metódy hráča *moving()* a *sendPosition()*. Taktiež volá metódy triedy *World* - *collisionsUpdate()*, *collisionsCheck()* a *gravity()*. Nakoniec volá metódu triedy *Canvas* - *redraw()*, ktorá prekreslí plátno.

end(reason)

Ukončí hru a vypíše dôvod na obrazovku. Dôvod (*reason*) je vstupným parametrom tejto metódy.

4.2.2 Trieda Canvas

Pomocou tejto triedy vykresľujeme našu hru na plátno (*HTML element canvas*).

Má 6 atribútov:

- 1) *element* - *HTML5 element canvas*, na ktorý vykresľujeme hru
(dátový typ *HTML DOM element*)
- 2) *ctx* - *Canvas context* (dátový typ *CanvasRenderingContext2D*)
- 3) *cW* - šírka plátna (dátový typ *Integer*)
- 4) *cH* - výška plátna (dátový typ *Integer*)
- 5) *startDrawX* - x pozícia začiatku vykresľovania objektov vo svete
(dátový typ *Integer*)
- 6) *startDrawY* - y pozícia začiatku vykresľovania objektov vo svete
(dátový typ *Integer*)

Metódy triedy *Canvas*:

constructor(canvas)

Vstupným parametrom konštruktora je *id* elementu *canvas*. V tejto metóde sa inicializujú atribúty tejto triedy. Atribúty *startDrawX* a *startDrawY* sa nastavujú na 0.

setStartDrawX(sdx)

Nastavuje atribút *startDrawX* na hodnotu, ktorá je vstupným parametrom tejto metódy.

redraw(game)

Služi na prekreslenie plátna. Najprv vyčistí plátno a potom volá metódy *render()* všetkých objektov, ktoré chceme vykresliť. Vstupným parametrom je referencia na objekt triedy *Game*.

4.2.3 Trieda World

Táto trieda predstavuje svet. Má 4 atribúty:

- 1) *player* - hráč (referencia na objekt triedy *Player*, dátový typ *Player*)
- 2) *teammate* - spoluhráč
(referencia na objekt triedy *Teammate*, dátový typ *Teammate*)
- 3) *physics* - fyzika (referencia na objekt triedy *Physics*, dátový typ *Physics*)
- 4) *stage* - aktuálna úroveň
(referencia na objekt triedy *Stage*, dátový typ *Stage*)

Metódy triedy *World*:

constructor(game, player, teammate)

Vstupnými parametrami do konštruktora je referencia na objekt triedy *Game*, referencia na objekt triedy *Player* a referencia na objekt triedy *Teammate*. Tieto referencie vloží do atribútov *player* a *teammate*. Táto metóda taktiež vytvára objekt triedy *Physics* a *Stage*.

switchStage(stageNum, game)

Táto metóda slúži na zmenenie úrovne podľa vstupného parametra *stageNum*. Vytvára nový objekt triedy *Stage* a volá jeho metódu *load()*. Tento novovytvorený objekt sa potom priradí do atribútu *stage*. Druhým vstupným parametrom je referencia na objekt triedy *Game*.

collisionsUpdate()

Volá metódu triedy *Physics* - *collisionDetection()*. Následne aktualizuje pole kolízií hráča a spoluhráča.

gravity()

Volá metódu *gravityApply()* triedy *Physics*.

collisionsCheck(game)

Volá metódy, ktoré kontrolujú kolízie. Vstupným parametrom je referencia na objekt triedy *Game*.

pickableColls_check()

Táto metóda kontroluje, či v poli kolízií hráča nie sú objekt, ktorý sa dá zobrať do inventára. Ak je, uloží do atribútu *canPickUp* triedy *Player* referenciu na tento objekt.

stationColls_check()

Kontroluje pole kolízií hráča a spoluhráča. Ak sa niektorý z nich nachádza na stanici, čiže je s ňou v kolízii, volá metódu *enter()* objektu triedy *Station*. Inak volá metódu *leave()*.

laserColls_check(game)

Kontroluje kolízie s lasermi. Ak je hráč v kolízii s laserom zmení jeho pozíciu. Vstupným parametrom je referencia na objekt triedy *Game*.

interactions_check()

Kontroluje kolízie s interaktívnymi objektmi v hre a ak obaja hráči kolidujú s nejakým predmetom a na tento predmet je naviazaná minihra uloží referenciu na objekt tejto minihry do atribútu *quest* triedy *Stage*.

4.2.4 Trieda Sprite

Trieda *Sprite* ma 14 atribútov:

- 1) *width* - šírka objektu (dátový typ *Integer*)
- 2) *height* - výška objektu (dátový typ *Integer*)
- 3) *x* - súradnica x (dátový typ *Integer*)
- 4) *y* - súradnica y (dátový typ *Integer*)
- 5) *speed* - rýchlosť objektu (dátový typ *Integer*)
- 6) *animCount* - číslo, ktoré sa inkrementuje a podľa neho sa vykresluje snímka v obrázku pri animácii (dátový typ *Integer*)
- 7) *looping* - uvádza, či sa objekt mení počas herného cyklu (1) alebo nie (0) (dátový typ *Integer*)
- 8) *solid* - uvádza, či je objekt pevný (1) alebo nie (0) (dátový typ *Integer*)
- 9) *interactive* - uvádza, či je objekt interaktívny (1) alebo nie (0) (dátový typ *Integer*)
- 10) *pickable* - uvádza, či sa dá objekt zobrať do inventára (1) alebo nie (0) (dátový typ *Integer*)
- 11) *visible* - uvádza, či je objekt viditeľný (1) alebo nie (0) (dátový typ *Integer*)
- 12) *img* - obrázok (dátový typ *Image*)
- 13) *imgW* - šírka obrázku (dátový typ *Integer*)
- 14) *imgH* - výška obrázku (dátový typ *Integer*)

Metódy triedy *Sprite*:

constructor(w,h,x,y,image,looping,solid,interact,pickable,visible)

Inicializuje atribúty podľa vstupných parametrov. Popis atribútov je uvedený vyššie.

setPosition(x,y)

Nastaví pozíciu objektu podľa vstupných parametrov *x* a *y*.

render(ctx)

Ak je objekt viditeľný (*visible* = 1), vykreslí ho na plátno. Vstupným parametrom je atribút *ctx* objektu triedy *Canvas*.

4.2.5 Trieda **Player**

Táto trieda predstavuje hráča a dedí triedu *Sprite*. Má 14 atribútov:

- 1) *socket* - zabezpečuje obojstrannú komunikáciu v reálnom čase
- 2) *type* - typ postavy „electron“ alebo „photon“ (dátový typ *String*)
- 3) *room* - ID miestnosti, v ktorej je hráč pripojený (dátový typ *String*)
- 4) *collisions* - pole objektov, s ktorými je hráč v kolízii
- 5) *inventory* - referencia na objekt triedy *Inventory* (dátový typ *Inventory*)
- 6) *moves* - pole pohybov, indexy: 0 - vľavo, 1 - vpravo, 2 - skok
(dátový typ *Array of Boolean*)
- 7) *direction* - smer pohybu (0 - vľavo, 1 - vpravo, 2 – skok) (dátový typ *Integer*)
- 8) *MAXspeed* - maximálna rýchlosť (dátový typ *Integer*)
- 9) *MAXjump* - maximálna výška doskoku (dátový typ *Integer*)
- 10) *jumpHeight* - výška doskoku (dátový typ *Integer*)
- 11) *jumpSpeed* - rýchlosť skoku (dátový typ *Integer*)
- 12) *jumping* - hráč práve skáče (dátový typ *Boolean*)
- 13) *controlBlock* - blokovanie ovládania (dátový typ *Boolean*)
- 14) *canDrive* - hráč môže riadiť vozidlo (dátový typ *Boolean*)
- 15) *canPickUp* - predmet, ktorý práve môže zobrať do inventára
(dátový typ *GameObject*)

Metódy triedy *Player*:

constructor(game,w,h,x,y,type,image,room)

Inicializuje atribúty podľa vstupných parametrov. Popis atribútov je uvedený vyššie.

control(game)

Vytvára *eventListener*-y a zabezpečuje ovládanie hráča. Vstupným parametrom je referencia na objekt triedy *Game*.

moving(game)

Kontroluje hodnoty v poli *moves* a podľa toho, ktoré pohyby sú nastavené na *true* volá metódy *moveLeft()*, *moveRight()* a *jump()*. Vstupným parametrom je referencia na objekt triedy *Game*.

moveLeft(game)

Kontroluje, či je možné, posunúť sa doľava. Ak áno, zmení pozíciu hráča tak, že odpočíta hodnotu atribútu *speed* od aktuálnej pozícií *x*. Vstupným parametrom je referencia na objekt triedy *Game*.

moveRight(game)

Kontroluje, či je možné, posunúť sa doprava. Ak áno, zmení pozíciu hráča tak, že pripočíta hodnotu atribútu *speed* k aktuálnej pozícií *x*. Vstupným parametrom je referencia na objekt triedy *Game*.

jump(world)

Kontroluje, či hráč môže vyskočiť. Ak áno, pripočíta hodnotu atribútu *jumpSpeed* k aktuálnej pozícií *y*. Vstupným parametrom je referencia na objekt triedy *World*.

sendPosition(game)

Odosieľa pozíciu a ďalšie informácie o hráčovi na server. Vstupným parametrom je referencia na objekt triedy *Game*.

render(canvas)

Vykresľuje a animuje postavu hráča. Vstupným parametrom je referencia na objekt triedy *Canvas*.

4.2.6 Trieda Teammate

Táto trieda predstavuje spoluhráča, dedí triedu *Sprite*. Má 4 atribúty:

- 1) *type* - typ postavy „electron“ alebo „photon“ (dátový typ *String*)
- 2) *direction* - smer pohybu (0 - vľavo, 1 - vpravo, 2 – skok) (dátový typ *Integer*)
- 3) *jumping* - spoluhráč práve skáče (dátový typ *Boolean*)

- 4) *collisions* - pole objektov, s ktorými je hráč v kolízii

Metódy triedy *Teammate*:

constructor(w,h,x,y,type,image)

Inicializuje atribúty podľa vstupných parametrov. Popis atribútov je uvedený vyššie.

updatePosition(teammate)

Nastavuje pozíciu spoluhráča a ďalšie atribúty, ktoré získava zo vstupného parametra *teammate*.

render(canvas)

Vykresľuje a animuje postavu spoluhráča. Vstupným parametrom je referencia na objekt triedy *Canvas*.

4.2.7 Trieda *GameObject*

Táto trieda dedí triedu *Sprite*. Väčšina objektov vo svete je objektom tejto triedy.

Má 3 atribúty dátového typu *String*:

- 1) *name* - jedinečný názov objektu
- 2) *type* - typ objektu
- 3) *description* - popis objektu

Metódy triedy *GameObject*:

constructor(name,type,desc,w,h,x,y,img,looping,solid,interactive,pickable,visible)

Inicializuje atribúty podľa vstupných parametrov. Popis atribútov je uvedený vyššie.

render(canvas)

Ak je objekt viditeľný, vykreslí ho na plátno. Vstupným parametrom je referencia na objekt triedy *Canvas*.

renderInventory(ctx)

Vykresľuje objekt v inventári (nepohybuje sa s pohybom po svete). Vstupným parametrom je atribút *ctx* objektu triedy *Canvas*.

4.2.8 Trieda Stage

Táto trieda predstavuje jednu úroveň hry. Obsahuje všetky objekty a minihry v danej úrovni. Má 7 atribútov:

- 1) *stageNum* - číslo úrovne (dátový typ *Integer*)
- 2) *objects* - pole objektov, ktoré sa nachádzajú v danej úrovni
(objekty tried *GameObject*, *Block*, *Station*, *Laser*, *Door*)
- 3) *bgImg* - obrázok pozadia úrovne (dátový typ *Image*)
- 4) *quest* - referencia na objekt minihry, ktorú je možné spustiť
- 5) *stageQuests* - pole minihier v úrovni
- 6) *questAvailable* - uvádza, či je možné spustiť nejakú minihru (dátový typ *Boolean*)
- 7) *mission* - popis misie (dátový typ *String*)

Metódy triedy *Stage*:

constructor(stageNum)

Inicializuje atribúty. Číslo úrovne nastaví podľa vstupného parametru *stageNum*.

load(game)

Podľa atribútu *stageNum*, teda čísla úrovne, vytvára všetky objekty a minihry v tejto úrovni. Objekty vkladá do poľa *objects* a minihry do poľa *stageQuests*. Vstupným parametrom je referencia na objekt triedy *Game*.

createGameObject(name,type,desc,w,h,x,y,img)

Vytvorí nový objekt triedy *GameObject* v tejto úrovni s vlastnosťami, ktoré sú vstupnými parametrami tejto metódy. (názov, typ postavy, popis, šírka, výška, súradnica x, súradnica y, obrázok)

deleteObject(DelObject)

Z poľa *objects*, odstráni objekt, ktorého referencia je vstupným parametrom tejto metódy.

getObjectByName(objName)

Vráti objekt z poľa *objects*, podľa názvu, ktorý je vstupným parametrom metódy.

render(canvas)

Táto metóda má na starosti vykreslenie všetkých objektov v danej úrovni. Vstupným parametrom je referencia na objekt triedy *Canvas*.

4.2.9 Trieda MiniGame

Túto triedu dedia všetky minihry v našej hre. Obsahuje spoločné vlastnosti a metódy minihier. Každá minihra má svoju metódu *control()*, ktorá slúži na ovládanie. Ďalej metódu *checkDone()*, ktorá slúži na kontrolu splnenia úloh. Taktiež metódu *load()*, pomocou ktorej sa načítavajú potrebné veci v minihre, metódu *win()*, ktorá sa volá pri dokončení minihry a *render()*, ktorá slúži na vykresľovanie minihry.

Trieda *MiniGame* má 14 atribútov:

- 1) *name* - názov minihry (dátový typ *String*)
- 2) *MGobject* - názov objektu, na ktorý je minihra viazaná (dátový typ *String*)
- 3) *canvas* - referencia na objekt triedy *Canvas* (dátový typ *Canvas*)
- 4) *request* - *RequestAnimationFrame* číslo snímku, ktorým sa dá zrušiť animačný cyklus (dátový typ *DomHighResTimeStamp*)
- 5) *controlBlock* - povolenie ovládania minihry (dátový typ *Boolean*)
- 6) *NumTasksForPlayer* - počet úloh pre hráča (dátový typ *Integer*)
- 7) *solvedTasks* - počet vyriešených úloh (dátový typ *Integer*)
- 8) *confirm_button* - tlačidlo pre potvrdenie riešenia (dátový typ *Button*)
- 9) *exit_button* - tlačidlo pre vypnutie minihry (dátový typ *Button*)
- 10) *info_button* - tlačidlo pre zapnutie nápoedy (dátový typ *Button*)
- 11) *infoPage* - nápoeda - referencia na objekt triedy *InfoPage* (dátový typ *InfoPage*)
- 12) *maxFPS* - maximálny počet snímok za sekundu (dátový typ *Integer*)
- 13) *interval* - interval snímok (dátový typ *Double*)
- 14) *lastFrameTime* - *Timestamp* poslednej snímky

Metódy triedy *MiniGame*:

constructor(name, MGobj, canvas, numoftasks, infoImgs)

Inicializácia atribútov podľa vstupných parametrov.

go(game)

Volá metódu *load()*. Zablokuje ovládanie hráča, odblokuje ovládanie minihry, zruší herný cyklus hry a zavolá metódu *gameLoop()*. Vstupným parametrom je referencia na objekt triedy *Game*.

gameLoop()

Zabezpečuje herný cyklus minihry. V každej iterácii cyklu volá metódu *checkDone()* a *render()*.

clickButton(game,click_x,click_y)

Táto metóda kontroluje, či bolo kliknuté na nejaké tlačidlo. Vstupnými parametrami sú súradnice kliknutia (*click_x*, *click_y*) a referencia na objekt triedy *Game*.

deleteObject(objName)

Odstráni objekt z minihry podľa názvu, ktorý je vstupným parametrom metódy.

confirm(game)

Táto metóda je volaná pri kliknutí na tlačidlo potvrdiť. Odosiela na server správu, že bola minihra dokončená a volá funkciu *win()* tejto minihry. Vstupným parametrom je referencia na objekt triedy *Game*.

preRender()

Slúži na vykreslenie obrázkov a tlačidiel, ktoré sú vo všetkých minihrách rovnaké. Túto metódu volajú triedy, ktoré dedia triedu *MiniGame*, vo svojej metóde *render()*.

Záver

Edukačné hry majú potenciál sa stať veľmi efektívnym nástrojom na zvýšenie záujmu detí o vybrané témy, preto je žiaduce ich použitie v rámci popularizácie vedy a techniky.

Cieľom tejto bakalárskej práce bolo vytvoriť edukačnú hru „Tajomstvá elektroniky a fotoniky“ na princípe objektovo orientovaného programovania. Podľa špecifikácie mala byť naša edukačná hra voľne dostupná a šíriteľná v rámci popularizácie vedy a techniky. Po analýze dostupných technológií sme sa rozhodli pre *HTML5*. Hru sme navrhli ako prehliadačovú hru pre dvoch hráčov. Našou cieľovou skupinou sú žiaci základných a stredných škôl, preto sme kládli dôraz na pútavý príbeh a grafiku. Implementovali sme spolu tridsaťdeväť tried na strane klienta a dve triedy na strane servera. Výsledkom tejto bakalárskej práce je funkčná edukačná hra. Hra obsahuje tri úrovne, v ktorých sa spolu nachádza deväť edukačných minihier. Tieto minihry sa venujú zaujímavým technickým riešeniam z oblasti elektroniky a fotoniky.

Boli splnené všetky zadefinované ciele. Prvá funkčná verzia našej edukačnej hry je voľne dostupná na vzdelávacom portáli eLearn central (URL: <http://uef.fei.stuba.sk/moodleopen/>).

Táto bakalárska práca bola vytvorená s podporou agentúry Kega Ministerstva školstva, vedy, výskumu a športu slovenskej republiky v rámci grantu 020STU-4/2015.

Zoznam použitej literatúry

1. Čo je flash? *di.ics.upjs.sk*. [Online] [Dátum: 12. máj 2017.]
https://di.ics.upjs.sk/informatika_na_zs_ss/studijny_material/grafika/flash/co_je_flash.htm.
2. How to Learn Flash and AS3 for Game Development. *gamedevelopment.tutsplus.com*. [Online] [Dátum: 12. máj 2017.] <https://gamedevelopment.tutsplus.com/articles/how-to-learn-flash-and-as3-for-game-development--gamedev-636>.
3. Flash a webová grafika. *tvorba-webu.cz*. [Online] [Dátum: 12. máj 2017.]
<https://www.tvorba-webu.cz/grafika/flash.php>.
4. Adobe končí s vývojom Flash technológie. *macweb.sk*. [Online] [Dátum: 12. máj 2017.]
<http://www.macweb.sk/adobe-konci-s-vyvojom-flash-technologie-pre-mobilne-zariadenia/>.
5. Prehliadač Google Chrome už natívne blokuje Flash. *zive.sk*. [Online] [Dátum: 12. máj 2017.] <http://www.zive.sk/clanok/121374/prehliadac-google-chrome-uz-nativne-blokuje-flash>.
6. Firefox is latest browser to kill off Adobe Flash support. *theinquirer.net*. [Online] [Zitat vom: 12. máj 2017.] <http://www.theinquirer.net/inquirer/news/2465543/mozilla-firefox-is-latest-browser-to-kill-off-adobe-flash-support>.
7. HTML5. *searchmicroservices.techtarget.com*. [Online] [Dátum: 12. máj 2017.]
<http://searchmicroservices.techtarget.com/definition/HTML5>.
8. HTML5: co přináší a proč se o něj zajímat. *root.cz*. [Online] [Dátum: 12. máj 2017.]
<https://www.root.cz/clanky/html5-co-prinasi-a-proc-se-o-nej-zajimat/>.
9. CSS Definition. *techterms.com*. [Online] [Dátum: 12. máj 2017.]
<https://techterms.com/definition/css>.
10. A Short History of JavaScript. *w3.org*. [Online] [Dátum: 12. máj 2017.]
https://www.w3.org/community/webed/wiki/A_Short_History_of_JavaScript.
11. Node.js. *whatis.techtarget.com*. [Online] [Dátum: 12. máj 2017.]
<http://whatis.techtarget.com/definition/Nodejs>.
12. Real Time Multiplayer in HTML5. [Online] [Dátum: 12. máj 2017.]
<http://buildnewgames.com/real-time-multiplayer/>.
13. Socket.io. *wikipedia.org*. [Online] [Dátum: 12. máj 2017.]
<https://en.wikipedia.org/wiki/Socket.IO>.

14. Objektové programovanie. *wikipedia.org*. [Online] [Dátum: 12. máj 2017.]

https://sk.wikipedia.org/wiki/Objektové_programovanie.

15. OOP – Zapúzdrenosť, dedičnosť, polymorfizmus. *zmuda.6f.sk*. [Online] [Dátum: 12. máj 2017.]

http://www.zmuda.6f.sk/ucText/java/OOP_2_ZapuzdrenostDedicnostPolymorfizmus.pdf.

Prílohy

| | |
|--|----|
| Príloha A: Štruktúra elektronického nosiča | II |
|--|----|

Príloha A: Štruktúra elektronického nosiča

Elektronický nosič obsahuje všetky zdrojové kódy a grafické podklady k interaktívnej edukačnej hre „Tajomstvá elektroniky a fotoniky“. Taktiež obsahuje užívateľskú príručku a dokument bakalárskej práce.

Štruktúra nosiča:

- bakalarska_praca.pdf
- hra
 - data
 - Img
 - js
 - minigames
 - *AccumulatorChargeQuest.js*
 - *DoorRepairQuest.js*
 - *ChooseCorrectQuest.js*
 - *LaserCommunicationRepairQuest.js*
 - *LaserTypesQuest.js*
 - *minigame.js*
 - *Pexeso.js*
 - *RgbLedQuest.js*
 - *RoverAccumulatorsQuest.js*
 - *SolarPanelsRepairQuest.js*
 - *entities.js*
 - *game.js*
 - *menu.js*
 - *physics.js*
 - node_modules
 - style
 - *style.css*
 - *favicon.ico*
 - *index.html*
 - *package.json*
 - *server.js*
- uzivatelska_prirucka.pdf

Zložka **img** obsahuje všetky grafické podklady k hre

Zložka **node_modules** - priečinky prostredia *Node.js*