**Katedra počítačov a informatiky**

**Fakulta elektrotechniky a informatiky**

**Technická univerzita v Košiciach**

**Futbalový brankár**

**Počítačová grafika**

**Systémová príručka**

|  |  |
| --- | --- |
| **Šk. rok: 2021/2022**  **Meno učiteľa: Ing. Miriama Mattová** | **Autor: Samuel Kucko** |

Obsah

[Zoznam obrázkov 2](#_Toc88561435)

[1. Funkcia programu 3](#_Toc88561436)

[2. Popis programu 3](#_Toc88561437)

[2.1. Popis riešenia 3](#_Toc88561438)

[2.2. Popis algoritmov a údajových štruktúr, globálnych premenných 4](#_Toc88561439)

[2.2.1. Algoritmus na generovanie herného poľa 4](#_Toc88561440)

[2.2.2. Globálne premenné modulu labyrint.js 5](#_Toc88561441)

[2.2.3. Globálne premenné modulu stopwatch.js 6](#_Toc88561442)

[2.3. Popis modulov, tried a podprogramov 6](#_Toc88561443)

[2.3.1. Labyrint.generate.js 6](#_Toc88561444)

[2.3.2. Labyrint.js 6](#_Toc88561445)

[2.3.3. Stopwatch.js 7](#_Toc88561446)

[2.3.4. Ťažiskové funkcie implementované v menu podstránkach 8](#_Toc88561447)

[3. Preklad programu 9](#_Toc88561448)

[3.1. Zoznam zdrojových textov 9](#_Toc88561449)

[3.2. Požiadavky na technické prostriedky pri preklade 9](#_Toc88561450)

[3.3. Požiadavky na programové prostriedky 9](#_Toc88561451)

[3.4. Vlastný preklad 9](#_Toc88561452)

[4. Nadväznosť na iné programové produkty 10](#_Toc88561453)

[5. Zhodnotenie riešenia 11](#_Toc88561454)

# Zoznam obrázkov

Obr. 1 Navrhnutý Use Case Diagram (Diagram prípadov použitia).................................................... 5

# Funkcia programu

Témou môjho semestrálneho projektu bola jednoduchá športová 3D hra - brankár. Dôležitou súčasťou riešenia bolo zamerať sa na použitie zákonov fyziky v hre a správne animovanie pohybov brankára.

Webová aplikácia jednoduchej športovej 3D hry – brankára pozostáva z dvoch základných súčastí a k nim prislúchajúcich funkcionalít. Prvou z nich je menu, ktoré sa v rámci štruktúry projektu člení na úvodnú alebo štartovaciu stránku, podstránku kde hrá prebieha (Play), informačnú podstránku (How to play). Druhá súčasť projektu pozostáva zo samotnej grafickej scény, ktorá vykresľuje aktuálny hru v 3D priestore.

Na podstránke „How to play“ sa používateľ dozvie všeobecné informácie o hre, a aj ako funguje a ako ju ovládať. V hre sa počíta počet gólov, koľko po sebe bez chytenia dokáže používateľ streliť. V hre sú 2 ukazovatele počtu gólov, jeden je aktuálny počet gólov za sebou, druhý je najlepší dosiahnutý. Po reštartovaní stránky sa však najlepší dosiahnutý resetuje, pretože hra nemá prepojenie s databázou.

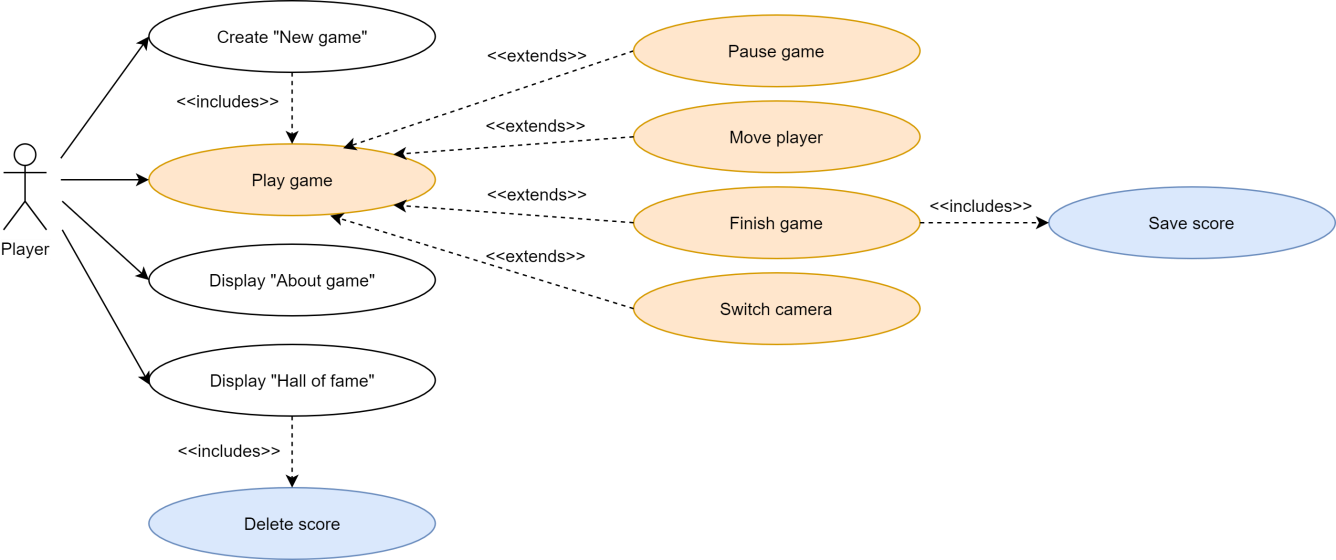
# Popis programu

Webová aplikácia hry 3D labyrint je tvorená komplexnou funkcionalitou vybraných webových technológií, ktoré sú detailnejšie charakterizované v kap. 5. Nasledujúce časti systémovej príručky sú zamerané na opis ťažiskových algoritmov, modulov a funkcií, ktoré súvisia s generovaním herného poľa, hlavným modulom, ktorý zabezpečuje inicializáciu a vykresľovanie 3D scény, ukladaním získaných dát o aktuálne prebiehajúcom levely a skóre hráča do lokálneho úložiska a implementáciou časomiery.

## Popis riešenia

Nižšie uvedený obrázok zobrazuje navrhnutý diagram prípadov použitia z pohľadu výstupu v podobe webovej aplikácie, ale aj z pohľadu možností hráča v rámci herného cyklu. Funkčné prvky 3D hernej scény sú znázornené oranžovou farbou a reprezentujú možnosti a ovládanie hry.

Modrá farba znázorňuje prvky, ktoré zabezpečujú prácu s dátami v lokálnom úložisku.



Obr. 1 Navrhnutý Use Case Diagram (Diagram prípadov použitia)

## Popis algoritmov a údajových štruktúr, globálnych premenných

Nasledujúce podkapitoly rozoberajú opis ťažiskového algoritmu na generovanie herného poľa (*labyrint.generate.js*) a zoznam globálnych premenných v module *labyrint.js a stopwatch.js*.

### Algoritmus na generovanie herného poľa

V rámci implementácie bolo zvolené ako vhodnejšie riešenie dynamické generovanie herných polí (levelov) na rozdiel od statického vytvorenia pevného počtu rovnakých herných polí. Na tento účel bol vybraný algoritmus implementovaný v jazyku javascript, ktorý je voľne dostupný na nasledujúcej webovej adrese: [http://www.roguebasin.com/index.php?title=Simple\_maze#Maze\_Generator\_in\_Javascript.](http://www.roguebasin.com/index.php?title=Simple_maze#Maze_Generator_in_Javascript)

Algoritmus zabezpečuje vygenerovanie náhodného labyrintu pri zadaní ľubovoľnej šírky a výšky poľa. Výška a šírka poľa je limitovaná zadaním nepárnych hodnôt, ktoré tento algoritmus nepodporuje.

#### Globálne premenné algoriritmu na generovanie herného poľa (labyrint.generate.js)

* Map[] – vygenerované herné pole (labyrint)
* WIDTH – šírka herného poľa
* HEIGHT – výška herného poľa
* DIRECTIONS – pole smerov a ich vektorov (*'N', 'S', 'E', 'W'*)

#### Zoznam funkcií algoriritmu na generovanie herného poľa (labyrint.generate.js)

* prefill: function()– funkcia zabezpečuje predvyplnenie herného poľa *map[]* prázdnymi objektami (nevyhnutné pred generovaním).
* shuffle: function(o)– funkcia zabezpečuje náhodné premiešanie smerov

*directions* z poľa *[ 'N', 'S', 'E', 'W' ]*, ktoré je parametrom funkcie a vracia toto pole náhodne premiešané.

* carve: function(x0, y0, direction)– najdôležitejšia funkcia algoritmu.

Parametrami je polovica šírky herného poľa, polovica výšky herného poľa a inicializačný smer. Funkcia využíva rekurzívny backtracking. Napĺňa postupne každý prvok poľa *map[]* informáciami o tom, v ktorých smeroch nie je daný prvok ohraničený „stenou“ teda je priechodný a taktiež uchováva informáciu o tom či už bol prvok navšívený alebo nie. Pre každý smer z vygenerovaného poľa, ktoré vracia funkcia *shuffle(o)* je rekurzívne opäť volaná funkcia *carve(x0, y0, direction)*. Takto je postupne naplnený každý prvok poľa *map[]* informáciami o jeho orientácií v štyroch smeroch.

* output: function()– funkcia, ktorá zabezpečuje vizualizáciu vygenerovaného herného poľa (labyrintu). Oproti jednoduchej implementácií z uvedenej url adresy, je funkcia doplnená o implementáciu na úrovni Three.js. Funkcia prechádza poľom *map[]* a zisťuje v cykle pre každý prvok poľa, či v smere *'S' a 'E'* je prvok nepriechodný. Ak je prvok v tomto smere nepriechodný, tak funkcia na úrovni Three.js vloží na dané miesto v 3D scéne horizontálny, resp. vertikálny objekt steny (v závislosti od smeru *'S' alebo 'E'*).
* generate: function()– jednoduchá funkcia, ktorá volá postupne vyššie opísané funkcie (*prefill(), carve(x0, y0, direction) a output()*), čím zabezpečí vygenerovanie herného poľa.
* setMapSize: function(height, width)– jednoduchá funkcia, ktorá nastaví na základe parametrov veľkosť poľa *map[]*.

### Globálne premenné modulu labyrint.js

camera – premenná, ktorá drží objekt kamery Three.js. scene – premenná, ktorá drží objekt scény Three.js, resp. Physi.js. renderer – premenná držiaca objekt rendereru Three.js.

plane – premenná, ktorá drží objekt roviny. sphere – premenná držiaca objekt gule, konkrétne skyboxu.

finishBox – premenná držiaca objekt kocky, ktorý slúži na ukončovanie levelov. player – premenná, ktorá drží objekt hráča (avatara). initGameData – objekt, ktorý drží inicializačné dáta hry. cameraCoordinates – pole objektov súradníc polohy kamery. cameraIndex – index súradníc aktuálnej kamery. cameraPosition – objekt súradníc aktuálnej kamery.

animationFrame – premenná držiaca aktuálny snímok animácie. isPaused – premenná držiacia informáciu o tom, či je hra pozastavená.

### Globálne premenné modulu stopwatch.js

startTime – premenná držiacia počiatočný čas. updatedTime – premenná držiacia aktualizovaný čas. difference – premenná držiacia rozdiel medzi aktualizovaným a počiatočným časom. tInterval – premenná držiacia hodnotu výstupu funkcie *setInterval()*.

savedTime – premenná držiacia uložený (pozastavený) čas.

paused – premenná držiacia informáciu o tom, či je časomiera pozastavená.

running – premenná držiacia informáciu o tom, či časomiera beží.

## Popis modulov, tried a podprogramov

Podkapitoly tejto časti sa venujú opisu ťažiskových programov a funkcií implementovaných v moduloch *labyrint.generate.js*, *labyrint.js*, *stopwatch.js* a funkcií implementovaných v menu podstránkach aplikácie.

### Labyrint.generate.js

Jednotlivé funkcie implementované v rámci tohto modulu sú detailne opísané v kap. 3.2.1.

### Labyrint.js

Je základy modul 3D hernej scény, ktorý zabezpečuje inicializáciu a vykresľovanie Three.js scény. Medzi ťažiskové funkcie modulu patria *init(), render(), update(), initLevel() a nextLevel()*.

Uvedené funkcie sú dôležité z pohľadu vykresľovania 3D objektov a celkovej hernej logiky.

#### Zoznam ťažiskových funkcií (labyrint.js)

* init: function()– funkcia inicializuje elementárne objekty Three.js a Physi.js a postupne volá funkcie pre možnosť zmeny kamery, pozastavenia hry, incializáciu levelu a kontrolu, či level nie je dokončený.
* render: function(o)– funkcia *render()* sa vykonáva v slučke. Volá funkciu

*simulate()* na objekte scény, pre simuláciu fyzikálnych vlastností (Physi.js). Zabezpečuje vykresľovanie scény v reálnom čase, priradenie pozície kamery na pozíciu avatara a volanie funkcie *update()*.

* update: function()– funkcia zabezpečuje pohybové schopnosti hráča (player) v 3D

scéne. Reaguje na stlačenie tlačidiel „W“, „A“, „S“, „D“.

* initLevel: function()– funkcia získa inicializačné dáta hry z lokálneho úložiska.

Následne na objekte *labyrintGenerator* volá funkciu modulu *labyrint.generate.js setMapSize(height, width)* a *generate()*. V poslednej časti volá trojicu funkcií pre pridanie 3D objektov, avatara a osvetlenia do scény.

* nextLevel: function()– funkcia v úvode zavolá funkciu modulu *stopwatch.js,* ktorá pozastaví časomieru. Následne dôjde k aktualizácií dát v lokálnom úložisku (zmena aktuálneho levelu na ďalší). Na konci funkcie sa vyhodnotí podmienka, ktorá zistí či ďalší level existuje. Pokiaľ áno, načíta súbor *index.html,* pokiaľ nie, uloží skóre hráča do lokálneho úložiska a načíta súbor *finishGame.html*.

### Stopwatch.js

Tento modul rieši chod, pozastavenie a reštart časomiery. Dáta získane z tohto modulu sú ukladané do lokálneho úložiska ako výsledný čas (skóre) hráča.

#### Zoznam ťažiskových funkcií (stopwatch.js)

* startTimer: function()– zabezpečuje začiatok počítania časomiery.
* pauseTimer: function()– zabezpečuje pozatavenie časomiery.
* getShowTime: function()– zabezpečuje vykresľovanie časomiery v tvare

„HH:MM:SS.sss“.

* getSavedTime: function()– funkcia vracia uložený čas alebo hodnotu 0

v prípade, že nie je uložený žiadny čas.

### Ťažiskové funkcie implementované v menu podstránkach

Ďalšie dôležité funkcie sú implementované priamo v skriptoch niektorých html podstránok. Najdôležitejšie funkcie obsahujú súbory *newGame.html* a *score.html*.

#### Zoznam ťažiskových funkcií (newGame.html)

* saveInitData: function(event)– funkcia získa inicializačné dáta z podstránky a zavolá funkciu *validateGameData(gameData)*. Pokiaľ validácia prebehne v poriadku dáta sú uložené do lokálneho úložiska.
* validateGameData: function(gameData)– funkcia kontroluje, či je vytvorený

aspoň jeden level a či šírka a výška každého levelu obsahuje iba párne hodnoty. V prípade, že niektorá z uvedených podmienok nie je splnená, funkcia vracia hodnotu *false* a zobrazí chybové hlásenie.

* addNewLevel: function()– funkcia pridá do *html* kódu kartu pre nastavenie

nového levelu.

* deleteLevel: function(level)– funkcia vymaže z *html* kódu kartu pre

nastavenie levelu, ktorý je parametrom funkcie.

* getLevelsDimensions: function()– funkcia vracia pole objektov, kde každý

objekt predstavuje šírku a výšku jedného herného poľa.

#### Zoznam ťažiskových funkcií (score.html)

* fillScoreTable: function()– funkcia zabezpečuje naplnenie elementu tabuľky

dátami z lokálneho úložiska.

* sortScoreData: function(scoreData)– funkcia zotriedi skóre (výsledný čas)

hráčov od najlepšieho po najhoršie.

* scoreConverter: function(milliseconds)– funkcia prekonvertuje čas

v milisekundách na minúty a sekundy a vracia čas zapísaný v danom formáte.

* deleteScoreData: function()– funkcia vymaže dáta z lokálneho úložiska.

# Preklad programu

## Zoznam zdrojových textov

Všetky opísané algoritmy, moduly a funkcie sú k dispozícií v projekte „pg-peterrosa“ na platforme Glitch, konkrétne na URL adrese [https://glitch.com/edit/#!/pg-peterrosa.](https://glitch.com/edit/#!/pg-peterrosa)

## Požiadavky na technické prostriedky pri preklade

Nižšie uvedené požiadavky na programové prostriedky a aplikácia opísaných webových technológií popísaná v kap. 5, bola overená na nasledujúcom hardvérovom technickom vybavení, ktoré poskytovalo stredný až výborný výkon. Uvedené hardvérové vybavenie je preto možné považovať za dostačujúce pre nasledujúci vývoj aplikácie.

* **Operačný systém** – Windows 10 Home
* **Procesor** – Intel Core i5-3210M 2.50 GHz
* **Grafická karta** – AMD Radeon HD 7670M/2GB
* **Operačná pamäť** – 8.00 GB

## Požiadavky na programové prostriedky

Webová aplikácia beží priamo vo webovom prehliadači na platforme Glitch. Programové prostriedky pre správny chod aplikácie preto spočívajú vo výbere vhodného webového prehliadača, ktorý má plnú podporu pre vykresľovanie 3D grafiky pomocou knižnice WebGL, resp. jej nadstavby Three.js.

Aplikácia bola otestovaná na týchto webových prehliadačoch - Google Chrome 9+, Firefox 4+, Opera 15+, Safari 5.1+ a Microsoft Edge v desktopovej verzii. Odporúča sa využívanie prehliadača Google Chrome 9+ alebo Mozilla Firefox 4+. Najlepší výkon v rámci 3D scény sa podarilo dosiahnuť použitím prehliadača Google Chrome 9+.

## Vlastný preklad

Webová aplikácia beží priamo vo webovom prehliadači na platforme Glitch. Z tohto dôvodu nie je potrebný pre nasledujúci vývoj aplikácie vlastný preklad či inštalácia doplnkových knižníc alebo modulov.

Pre spustenie projektu na lokálnom zariadení je potrebné úspešne stiahnuť projekt z platformy Glitch. Po úspešnom stiahnutí projektu už stačí iba otvoriť projekt v ľubovoľnom IDE podporujúcom písanie zdrojového kódu v jazyku JavaSript. Projekt bol vyvíjaný použitím IDE JetBrains WebStorm vo verzii 2019.1.4. Pre budúci vývoj ale použitie tohto vývojového prostredia nie je podmienkou.

# Nadväznosť na iné programové produkty

Na vývoj webovej aplikácie boli využité nasledujúce webové technológie. Pred vykonávaním dodatočných úprav, či pokračovaní v ďalšom vývoji sa preto odporúča ich stredná až pokročilá znalosť.

* **Three.js** – Javascript knižnica používaná na tvorbu a zobrazovanie 3D počítačovej grafiky vo webovom prehliadači.
* **Physi.js** – Plugin pre Three.js, používaný na jednoduchú implementáciu a simulovanie fyzikálnych vlastností 3D objektov.

Okrem uvedených doplnkových programových prostriedkov sa odporúča minimálne základná až pokročilá znalosť elementárnych webových technológií ako HTML, CSS a JavaScript.

# Zhodnotenie riešenia

Implementované riešenie webovej aplikácie hry 3D labyrint splnilo podľa môjho názoru dostatočnú časť z vopred stanovených požiadaviek. Na druhej strane, kvôli určitým obmedzeniam využitých technológií a obmedzeniam časových kapacít boli niektoré návrhové postupy vynechané a predstavujú možnosti pre budúce doplnenie funkcionality. Nedopracované časti systému úzko súvisia s jeho niektorými aktuálnymi obmedzeniami, ktoré sú evidované v nasledujúcom zozname:

* ***Obmedzenie v súvislosti s neexistenciou herného módu pre dvoch hráčov.***
* ***Obmedzenie v súvislosti s ukladaním dát iba do lokálneho úložiska.***
* ***Neumožnenie prihlasovania a registrácie používateľa.***
* ***Neumožnenie načítania posledného stavu hry.***
* ***Neumožnenie zadávania nepárnych hodnôt pre veľkosť herného poľa.***
* ***Obmedzenia súvisiace s nedostatočným výkonom na starších webových prehliadačoch.***

Tento zoznam poskytuje budúcim vývojárom prehľad o nedopracovaných častiach aplikácie a základnú osnovu pre ďalší vývoj. Pri nasledujúcom vývoji je možné zamerať sa na uvedené obmedzenia, ako aj na skvalitnenie herného zážitku. Vizuálnu stránku hry je možné rozšíriť o výber grafického módu, rozšírenejší výber avatarov (prípadne umožnenie načítania vlastného avatara zo súboru) a pod. Logika hry môže byť dopracovaná o ďalšie možnosti ako napríklad zbieranie bonusových predmetov, zbieranie zrýchľujúcich predmetov, doplnenie implementácie príšer, ktoré naháňajú hráča a sťažujú mu tak prechod labyrintom atď.