STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST
Obor č. 18: Informatika

Na křižovatky s rozumem

Jakub Merta, Filip Kocháň
Zlínský kraj

Otrokovice 2021

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST
Obor č. 18: Informatika

Na křižovatky s rozumem

Know Your Crossroads

Autoři: Jakub Merta a Filip Kocháň
Škola: Gymnázium a Jazyková škola s právem státní jazykové zkoušky Zlín, nám. T. G Masaryka 2734, 760 01 Zlín
Kraj: Zlínský kraj
Konzultant: Mgr. Michal Mikláš

Otrokovice 2021

Prohlášení

Prohlašujeme, že jsme svou práci SOČ vypracovali samostatně a použili jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašujeme, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemáme závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Otrokovicích dne 30. 3. 2021 Filip Kocháň, Jakub Merta

Poděkování

Na úvod bychom chtěli poděkovat našemu spolužákovi Kryštofu Bednaříkovi, který výrazně pomohl s vizuální stránkou našeho projektu tím, že vytvořil textury aut, značek, cest a semaforů, a to jako vektorovou grafiku v programu Adobe Illustrator v rámci své závěrečné maturitní práce z předmětu Grafika, design a multimédia.

Nemůžeme také nezmínit našeho učitele informatiky Michala Mikláše, který nám vnuknul myšlenku zúčastnit se Středoškolské odborné činnosti a na kterého jsme se mohli obracet s jakýmikoli dotazy.

Anotace

Tato práce se zabývá webovou aplikací, v níž si může uživatel testovat a trénovat řešení různých dopravních situací na křižovatkách, a to vytvářením vlastních dopravních situací, které se následně přehrají v animačním okně, nebo odpovídáním na předem definované testové otázky.

Klíčová slova

autoškola; křižovatky; JavaScript; webová aplikace

Annotation

The aim of this document is to describe our web application where users can improve their ability of solving different traffic situations either by creating their own situation, which will then play in the animation window, or by answering predefined exam questions.

Keywords

driving school; crossroads; JavaScript; web application

Obsah

1 Úvod

Tento dokument popisuje projekt nazvaný *Na křižovatky s rozumem*, který naleznete na adrese `www.nakrizovatkysrozumem.cz`. Naším cílem bylo vytvořit takovou aplikaci, kde by si (nejen) uživatelé připravující se na testy potřebné pro získání řidičského průkazu mohli procvičit dopravní situace – to, s čím se setkají jak v testových otázkách, tak – a to hlavně – v reálném provozu.

Před uživatele jsou formou testových otázek, které má následně zodpovědět, postaveny jednotlivé dopravní situace. Po ukončení testu se následně může ke každé situaci vrátit a přehrát si ji.

Kromě pouhého sledování předdefinovaných dopravních situací a odpovídání na otázky s nimi spojenými si však může uživatel nasimulovat situaci vlastní a následně sledovat, jak by se v provozu odehrála – a to považujeme za hlavní výhodu naší aplikace oproti ostatním webovým stránkám, které vás připravují na testy.

Pokud jde o otázky typu *Kdo může rozhodnout o odstranění vozidla, které neoprávněně stojí na parkovišti?*, většinou se prostě naučíte danou poučku nebo definici nazpaměť. U dopravních situací však nemusí pouhá poučka nebo obrázek stačit – je potřeba si ji dobře představit.

Inspirací k vytvoření tohoto projektu nám bylo to, že jsme sami v nedávné době absolvovali autoškolu a aplikace, jako je ta naše, by nám přišla vhod.

2 Použité technologie

2.1 HTML, CSS

Nikoho asi nepřekvapí, že při tvorbě webové aplikace použijeme HTML a CSS. Hlavní část naší aplikace se však odehrává spíše v JavaScriptu, a proto se HTML a CSS nebudeme nijak zvlášť zabývat.

2.2 JavaScript

JavaScript (`https://www.javascript.com/`) můžeme dnes nalézt na mnoha webových stránkách a náš projekt není výjimkou.

JavaScript je objektově orientovaný skriptovací jazyk, nejčastěji ho lze najít na webových stránkách, jeho interpretaci provádí webový prohlížeč uživatele. Jinými slovy JavaScript běží na straně klienta, ne serveru. [1]

My jsme použili JavaScript zejména pro animaci jednotlivých dopravních situací a fungování/logiku automobilů na křižovatkách.

2.2.1 knihovna p5.js

JavaScriptových grafických knihoven je spousta, jmenovat můžeme například Two.js (`https://two.js.org/`), anime.js (`https://animejs.com/`) nebo Raphaël (`https://dmitrybaranovskiy.github.io/raphael/`).[2]

My jsme se rozhodli použít knihovnu p5.js (`https://p5js.org/`), a to hlavně z důvodu jednoduchosti, přehledné dokumentace na oficiálním webu a také nespočtu ukázek možného použití této knihovny na YouTube kanále *The Coding Train* (`https://www.youtube.com/user/shiffman`).

V ukázce kódu **1** můžeme vidět použití dvou základních funkcí z knihovny `p5.js` – `setup()` a `draw()`.

```
let shade = 0;

function setup() {
  createCanvas(800, 600);
}

function draw() {
  background(shade);
  shade++;
  shade %= 256;
}
```

ukázka kódu 1: Použití `setup()` a `draw()`

Na začátku programu založíme proměnnou `shade`, která bude určovat barvu pozadí. Poté se zavolá funkce `setup()`. Ta se používá pro definování vlastností prostředí, jako je například velikost obrazovky.

Zavoláním metody `createCanvas(800, 600)` se vytvoří HTML element `<canvas>` s rozměry 800 px × 600 px. Element `<canvas>` se používá pro kreslení grafiky na webové stránce. [3]

Metoda `draw()` se zavolá jedenkrát pro každý snímek animace. Metoda `background(shade)` nastaví barvu pozadí pro `canvas` na hodnotu proměnné `shade`, která v našem případě určuje stupně šedi. Pro `shade = 0` je pozadí černé, pro `shade = 255` je pozadí bílé.

Každý snímek animace se hodnota proměnné `shade` zvětší o 1. Matematická operace modulo (značí se operátorem `%`) zajistí, že pokud se `shade` dostane na hodnotu 256, jeho hodnota se vrátí zpátky na 0. Výsledkem tohoto programu by tedy bylo animační okno, jehož pozadí by se plynule měnilo z černého na bílé pořád dokola.

2.3 PHP

PHP (`https://www.php.net/`) je skriptovací jazyk používaný ke tvorbě dynamických internetových strá-nek a webových aplikací.

Rozdíl oproti JavaScriptu je třeba v tom, že PHP skripty jsou prováděny na straně serveru a k uživateli je přenášen až výsledný produkt, kterým je samotný HTML kód.[4]

My jsme PHP využili například pro propojení s databází a kontrolu správně zadaného hesla (viz ukázka **2**).

```php
$sql = "SELECT `password` FROM `uzivatele`
  WHERE `username` = '" . $jmeno . "'";
$result = mysqli_query($conn, $sql);
$row = mysqli_fetch_assoc($result);
$heslodb = $row['password'];
$heslo = $_POST['heslo'];

if(password_verify($heslo, $heslodb)) {
  // uspesne prihlaseni...
} else {
  // prihlasovaci udaje zadany chybne...
}
```

ukázka kódu 2: Ověření hesla v PHP

Proměnná `$sql` obsahuje dotaz na databázi. Chceme vybrat heslo (`` `password` ``) z tabulky `` `uzivatele` ``, kde jméno uživatele odpovídá uživateli, který se právě pokouší přihlásit.

Metoda `mysqli_query($conn,$sql)` provede daný dotaz `$sql` na databázi a výsledek se uloží do proměnné `$result`.[5]

Metoda `mysqli_fetch_assoc($result)` vrátí asociativní pole odpovídající načtenému řádku z databáze. Pokud již nejsou další řádky k dispozici, vrátí metoda `null`. Vrácená hodnota se uloží do proměnné `$row`.[6]

Do proměnné `$heslodb` se uloží hash[1] vytvořený metodou `password_hash()`, který je uložen u daného účtu v databázi, do proměnné `$heslo` se uloží heslo, které zadal uživatel při pokusu o přihlášení ke svému účtu.

[1] Hash je digitální otisk textu, který je výsledkem hashovací funkce. Má podobu jedinečného shluku čísel a písmen. Používá se tam, kde nechceme, aby třetí strana odhalila námi napsanou zprávu.[7]

Metoda `password_verify($heslo, $heslodb)` nakonec ověří, zdali hash `$heslodb` odpovídá heslu `$heslo`, které zadal uživatel. Pokud ano, přihlášení proběhlo úspěšně.[8]

2.4 MySQL a SQL

V naší aplikaci používáme databázi, ve které uchováváme tato data:

- přihlašovací údaje uživatelů,
- dokončené testy,
- odpovědi vybrané v testu uživatelem,
- otázky k testovým situacím,
- pozice jednotlivých aut pro testové situace,
- možné odpovědi na testové otázky,
- nahlášené chyby.

Pro databázi jsme použili systém MySQL (`https://www.mysql.com`), který je spolu s jazykem PHP a webovým serverem Apache (`https://httpd.apache.org/`) součástí WampServeru (`https://www.wampserver.com/en/`).

Pro ukládání dat do databáze, manipulaci s daty a pro jejich následné získávání z databáze používáme dotazovací jazyk SQL. Příkaz pro vybrání pěti náhodných dopravních situací do testu z databáze v jazyce SQL může vypadat takto (ukázka **3**):

```
1 SELECT `idsituace` FROM `situace` ORDER BY RAND() LIMIT 5
```

ukázka kódu 3: Výběr pěti náhodných situací v SQL

Pro identifikaci dané situace stačí její `id`, proto budeme příkazem `SELECT` z tabulky `` `situace` `` vybírat pouze `` `idsituace` ``. Požadovaný počet výsledků, to je v našem případě 5, zařídí příkaz `LIMIT`. Jednotlivé situace však potřebujeme vybrat náhodně, toho docílíme použitím `ORDER BY RAND()`.

2.5 JSON

JSON (JavaScript Object Notation) je formát pro výměnu dat, který je jednoduše zapisovatelný a čitelný člověkem. Tento formát umí pojmout pole, objekty, čísla, řetězce a speciální hodnoty jako `true`, `false` nebo `null`.[9][10]

My jsme pomocí JSONu zapisovali pruhy – části, ze kterých pak může být složena křižovatka jako celek (více v kalitole 4).

2.6 Git a GitHub

Protože jsme na projektu pracovali dva, systém správy verzí – Git, a místo, kde můžeme společný kód sdílet – GitHub (`https://github.com/`), pro nás byly naprostou nezbytností.

Oceňujeme například možnost porovnávání jednotlivých úprav, hledání rozdílů mezi nimi, a v neposlední řadě větvení a slučování (merge), které nám práci hodně zefektivnilo.

2.7 LaTeX

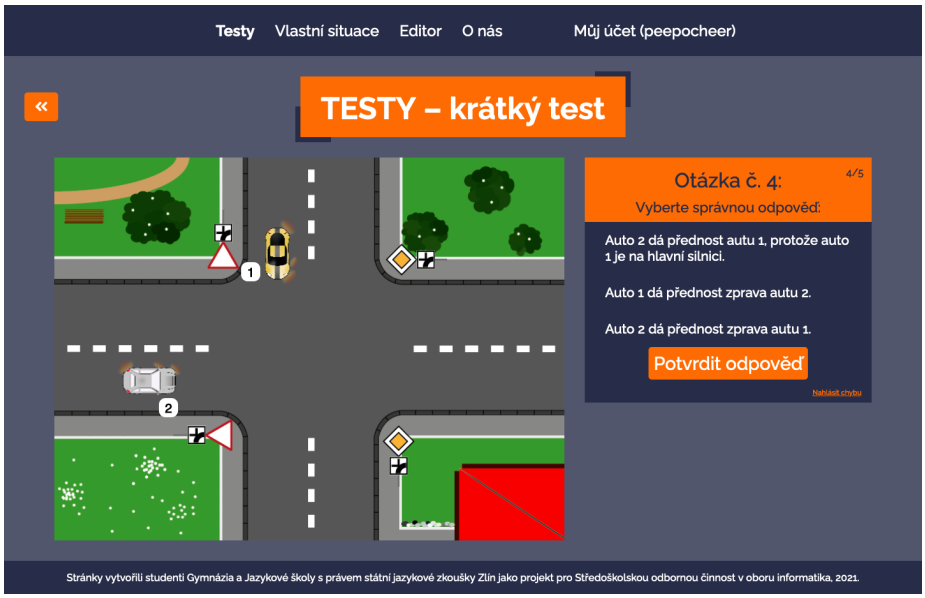
Pro zdokumentování naší závěrečné práce jsme využili sázecího systému LaTeX. Samotné psaní práce potom proběhlo v aplikacích:

- Overleaf (`www.overleaf.com/`),
- Visual Studio Code (`https://code.visualstudio.com/`) s rozšířením Latex Workshop (`https://marketplace.visualstudio.com/items?itemName=James-Yu.latex-workshop`).

3 Hlavní části aplikace

Naše stránka má dvě hlavní sekce – testovací sekci a sekci pro tvorbu vlastních dopravních situací.

3.1 Testy



obrázek 1: Sekce *Testy* – test

Zde si uživatel nejprve vybere typ testu, a to buď podle počtu otázek, nebo podle obtížnosti. Na výběr je několik typů:

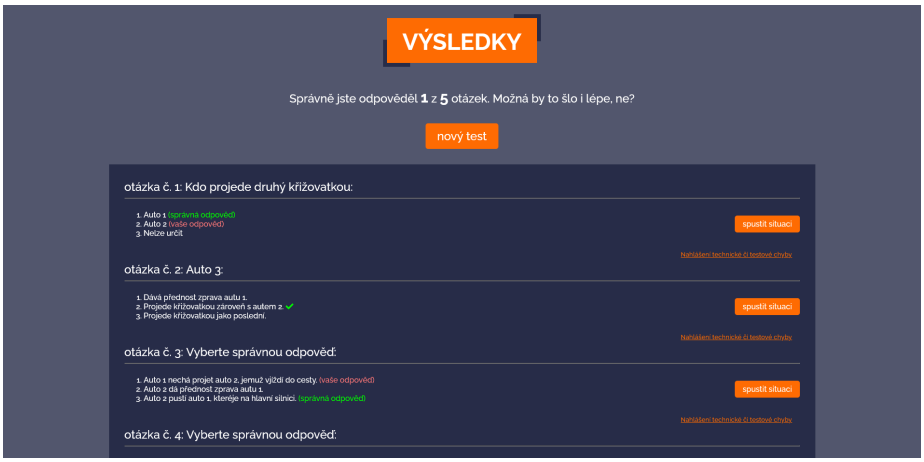
- krátký test – 5 náhodných otázek,
- střední test – 10 náhodných otázek,
- dlouhý test – 20 náhodných otázek,
- lehký test – 10 otázek, ve kterých ostatní uživatelé nejméně chybují,
- těžký test – 10 otázek, ve kterých ostatní uživatelé chybují nejvíce.

Poté přijde na samotný test. Uživateli se začne přehrávat určitá dopravní situace. Předtím, než auta projedou křižovatkou, se ukázka zastaví a uživatel vybere k dané situaci jednu ze tří odpovědí. Po potvrzení odpovědi se načte další situace.[2] Testovací prostředí lze vidět na obrázku **1**. Po dokončení testu se

[2]Uvažovali jsme o dokončení animace hned po potvrzení odpovědi, ale nakonec jsme se rozhodli uživateli dát možnost vrátit se ke všem situacím naráz až po dokončení testu. Nepřišlo nám ideální průběžně sdělovat uživateli výsledky testu, jednak by to mohlo ovlivnit jeho další odpovědi a jednak rozhodně není povzbuzující hned od začátku vědět, že máte nízkou úspěšnost.

uživateli zobrazí výsledková tabulka s odpověďmi a výslednou úspěšností, viz obr. **2**.

Pokud uživatel v nějaké situaci chyboval, nebo se k ní prostě chce jen vrátit, může si ji ve výsledkové tabulce zpětně přehrát (a tentokrát už celou naráz, bez zastavení), to můžeme vidět na obrázku **3**.



obrázek 2: Sekce *Testy* – výsledková tabulka

obrázek 3: Sekce *Testy* – zpětné přehrání situace

3.2 Vlastní situace

Snažili jsme se najít takové webové stránky, které se zabývají zkušebními testy z autoškoly. Našli jsme jich hned několik, níže vypsané jsou ty, na které podle nás uživatel, který si chce zkusit cvičné testy, s největší pravděpodobností narazí.

- `https://etesty2.mdcr.cz/Home/Tests/ro`
- `www.bezpecnecesty.cz/`
- `www.autoskola-testy.cz/`
- `www.autoweb.cz/testy-autoskol/`
- `www.vsechny-autoskoly.cz/testy_online/`

I když jsme hledali sebevíce, na žádném z těchto webů jsme nenašli místo, kde by si uživatel vytvořil vlastní dopravní situaci a sledoval, jak se odehraje.

A přesně to jsme se v našem projektu pokusili napravit. Právě sekcí *Vlastní situace* se chceme odlišit od webových stránek vypsaných výše a umožnit tak uživatelům naší webové stránky jedinečnou zkušenost.

3.2.1 Tvorba vlastní situace

Jak si takovou vlastní situaci vytvořit?

Jako první si uživatel vybere křižovatku. Na výběr jsou křižovatky typu X a T s různými pozicemi hlavních a vedlejších cest. Po vybrání křižovatky následuje přidání auta do situace, to lze provést následovně:

1. Uživatel najede myší na místo, na které chce auto přidat. Auto lze vždy přidat pouze na místo, kde začíná cesta. Po najetí na dané místo se zobrazí silueta auta (viz obr. **4**).

2. Po kliknutí se zvýrazní možné trasy, kterými se auto může z dané pozice vydat. To můžeme vidět na obrázku **5**. Uživatel přiřadí jednu z tras danému autu tak, že na ni klikne. Tím se auto přidá do dopravní situace.

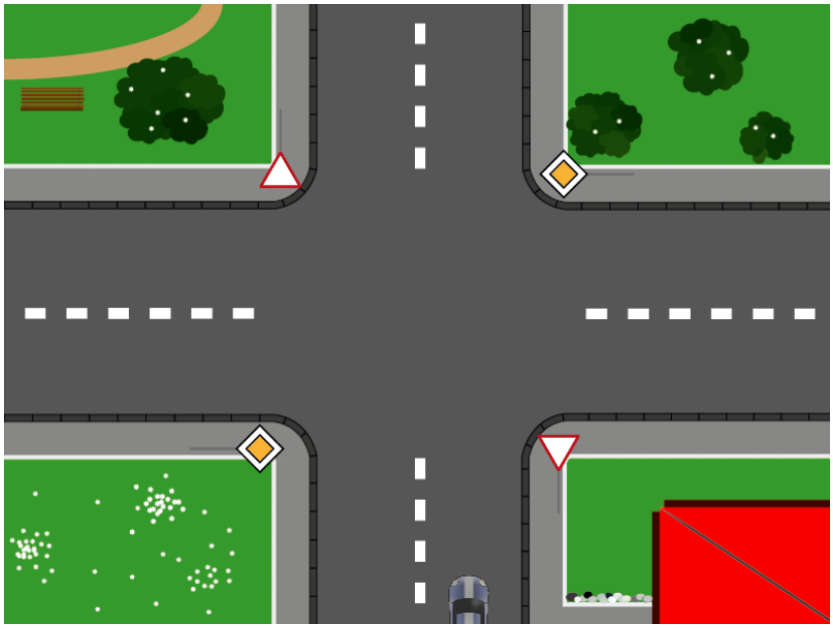
Pokud chce uživatel změnit trasu auta, stačí na auto kliknout a trasu mu přiřadit znova.

Pro úplné smazání auta z dopravní situace je potřeba nejprve dané auto vybrat kliknutím na něj, poté lze auto odstranit kliknutím na ikonu koše, která se zobrazí vedle něj.

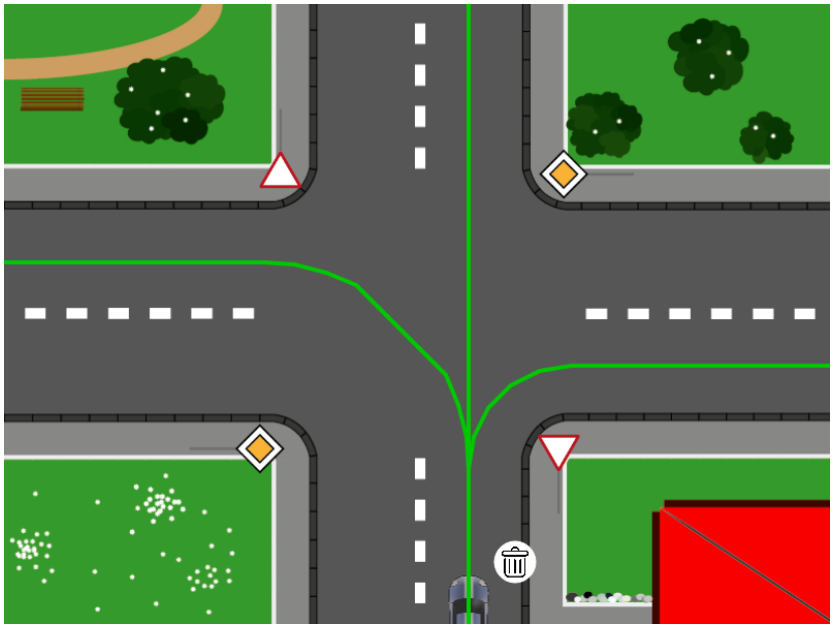
Po dokončení konfigurace může uživatel animaci spustit. V průběhu animace ji lze kdykoli pozastavit a poté zase spustit. Auta lze také přidávat během probíhající animace.[3]

Prostředí vlastních situací můžeme vidět na obrázku **6**. Jeho součástí jsou kromě animačního okna tlačítka, kterými lze animaci spustit či pozastavit, tlačítkem „Vymazat vše“ lze odstranit všechna auta z křižovatky.

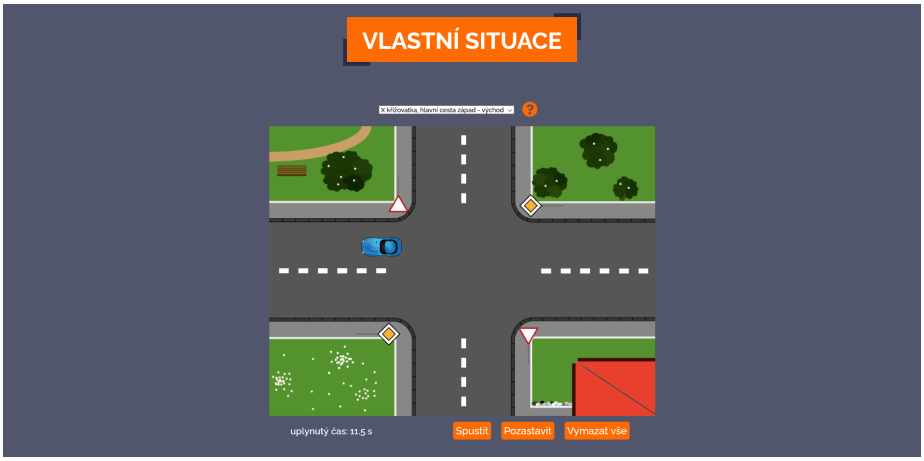
[3] Aby se zabránilo nevyžádanému zastavování aut uprostřed křižovatky, nové auto lze přidat pouze pokud křižovatkou neprojíždí jiné auto.



obrázek 4: Sekce *Vlastní situace* – silueta auta



obrázek 5: Sekce *Vlastní situace* – výběr trasy



obrázek 6: Sekce *Vlastní situace*

3.3 Můj účet

V naší aplikaci má uživatel možnost založit si účet. Tím získá přístup k testům, které v minulosti dokončil. Jednotlivé testy si může zpětně projít, může se podívat, jak na jednotlivé otázky odpovídal, a také si může situaci ke každé otázce přehrát.

Prostředí sekce *Můj účet* lze vidět na obrázku **7**.

obrázek 7: Sekce *Můj účet* – seznam testů

4 Třída Cesta

Každý typ křižovatky je definovaný jako nová instance třídy `Cesta`, která má několik vlastností:

- `sitPruhu` – Pole, jsou zde uložena místa, po kterých se auto může pohybovat. Více v kapitole 4.3.
- `znacky` – Pole, jsou zde uloženy jednotlivé dopravní značky, které se na cestě vyskytují.
- `spawny` – Pole, obsahuje místa, na která lze přidávat nová auta. Tato místa jsou podrobněji popsány v kapitole 4.4.

4.1 Zápis JSON

Informace o každé cestě jsme ukládali ve formátu JSON.

Na ukázce **4** je JSON reprezentující člověka, který má definováno jméno, věk, seznam oblíbených automobilů a informaci o tom, jestli je zaměstnaný.

```
{
  "jmeno": "Petr Buk",
  "vek": 51,
  "auta": [
    "Felicia",
    "Velorex",
    "Fiat Multipla"
  ],
  "zamestnany": true
}
```

ukázka kódu 4: Zápis JSON

Nyní si pojďme tento zápis podrobněji probrat. JSON je založen na dvou strukturách:

- **Kolekce párů „název-hodnota“**, to je v různých programovacích jazycích realizováno jako objekt nebo asociativní pole.
- **Seřazený seznam hodnot**, který je ve většině jazyků je realizován jako pole nebo seznam.

Pro zápis JSON také platí několik pravidel:

- Objekt, neboli soubor párů název-hodnota, začíná levou složenou závorkou a končí pravou složenou závorkou.
- Každý název je následnován dvojtečkou, poté následuje hodnota, jednotlivé páry jsou mezi sebou odděleny čárkou.
- Pole, neboli soubor hodnot, začíná levou hranatou závorkou a končí pravou hranatou závorkou.

- Hodnoty pole jsou odděleny čárkami.
- Hodnotou může být řetězec ohraničený uvozovkami, číslo, `true`, `false`, `null`, objekt nebo pole, jednotlivé hodnoty mohou být do sebe vnořovány.

Daný člověk z ukázky **4** je definován jako objekt, proto jsou všechny jeho vlastnosti obaleny ve složených závorkách. Jednotlivé vlastnosti daného člověka jsou odděleny čárkami. Jelikož je `Petr Buk` řetězec, je uzavřený do dvojitých uvozovek.

Vlastnosti `vek` ani `zamestnany` nejsou řetězce, proto do uvozovek uzavřeny nejsou.

Jednotlivá auta tvoří pole, a jsou tedy zapsána do hranatých závorek.[9]

4.2 Zápis cesty v JSON

Nyní se můžeme zabývat samotným zápisem dané cesty v JSON.

```
{
  "nazev": "T krizovatka",
  "textura": 1,
  "sitPruhu": [
    {
      "startx": 450,
      "starty": 450,
      "sizex": 0,
      "sizey": -300,
      "hlavni": 0
    },
    {
      "startx": 450,
      "starty": 150,
      "sizex": 0,
      "sizey": -150,
      "hlavni": 0
    },
    {
      "startx": 550,
      "starty": 250,
      "sizex": -300,
      "sizey": 0,
      "hlavni": 1
    }
  ],
  "znacky": [
    {
      "typ": 9,
      "posx": 515,
      "posy": 420,
      "angle": 0
    },
```

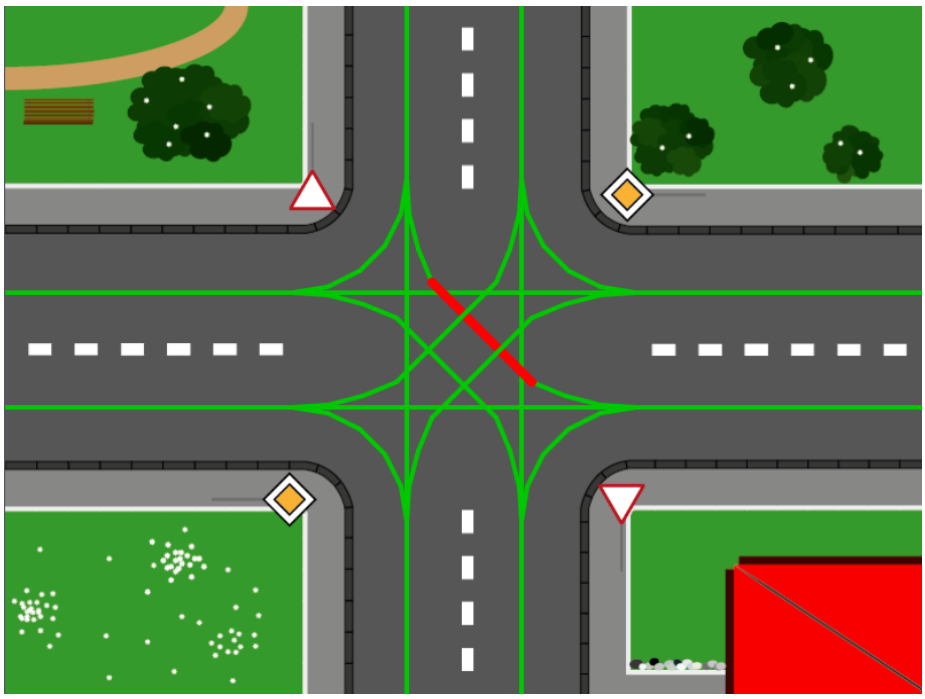
```
    {
      "typ": 17,
      "posx": 280,
      "posy": 410,
      "angle": 90
    },
    {
      "typ": 11,
      "posx": 520,
      "posy": 185,
      "angle": 270
    }
  ]
}
```

ukázka kódu 5: Zápis cesty v JSON

Na ukázce **5** vidíme, že daná cesta má název, tím je řetězec, a texturu (obrázek, zobrazuje se jako pozadí v animačním okně), která je reprezentována číslem.

Dále má cesta vlastnost `sitPruhu` – to je pole obsahující objekty třídy `Pruh`, kterou se podrobněji zabýváme v kapitole 4.3. Celá křižovatka se skládá zhruba z 50 takovýchto pruhů, pro přehlednost jsou vypsány pouze 3.

Na obrázku **8** jsou zvýrazněny všechny pruhy z pole `sitPruhu` pro danou křižovatku.



obrázek 8: Jednotlivé pruhy v cestě

Následuje pole `znacky`, kde jsou uloženy typy a pozice jednotlivých značek, které se v křižovatce vyskytují.

4.3 Třída Pruh

Jednotlivé části cesty nazýváme pruhy, jsou to instance třídy `Pruh` s těmito vlastnostmi:

- `start` – Vektor (objekt třídy `p5.Vector`, který ukládá komponenty `x` a `y`), souřadnice začátku pruhu.
- `destination` – Vektor, souřadnice místa, kde pruh končí.
- `nextPruhy` – Pole obsahující pruhy, které začínají v místě, kde daný pruh končí.
- `previousPruhy` – Pole obsahující pruhy, které končí v místě, kde daný pruh začíná.
- `hlavni` – Hodnota 1, pokud je pruh součástí hlavní cesty, hodnota 0, pokud je pruh součástí vedlejší cesty.

Nový pruh definujeme z informací, které jsou uvedeny v JSON souboru dané cesty (ukázka **5**) jako `new Pruh(startx, starty, sizex, sizey, hlavni)`.

Každý pruh je úsečkou, která je dána začátečním bodem `[startx; starty]`, velikostí `(sizex; sizey)` a informací, jestli je pruh součástí hlavní, nebo vedlejší cesty. Koncový bod daného pruhu (uložený v proměnné `destination`) má potom souřadnice `[startx + xsize; starty + ysize]`. Tyto pruhy jsou uloženy v poli `sitPruhu` a společně tvoří místa, po kterých se mohou jednotlivá auta pohybovat.

4.3.1 Konkrétní pruh

Na ukázce **6** je zápis jednoho konkrétního pruhu z pole `sitPruhu` v JSON. Tento pruh je zvýrazněn červeně v obrázku **8**.

```
{
  "startx": 372,
  "starty": 241,
  "sizex": 87,
  "sizey": 87,
  "hlavni": 0
}
```

ukázka kódu 6: Konkrétní pruh v JSON

4.3.2 Metoda getNextPruh()

Součástí třídy `Pruh` je také několik metod, např. na ukázce **7** je ta, která každému pruhu najde pruhy, které z něj vycházejí.

```
getNextPruh(sitPruhu) {
  for (let i = 0; i < sitPruhu.length; i++) {
    if (this.destination.x == sitPruhu[i].start.x
        && this.destination.y == sitPruhu[i].start.y) {

          this.nextPruhy = [...this.nextPruhy, sitPruhu[i]];
          sitPruhu[i].previousPruhy =
            [...sitPruhu[i].previousPruhy, this];

    }
  }
}
```

ukázka kódu 7: Metoda `getNextPruh()`

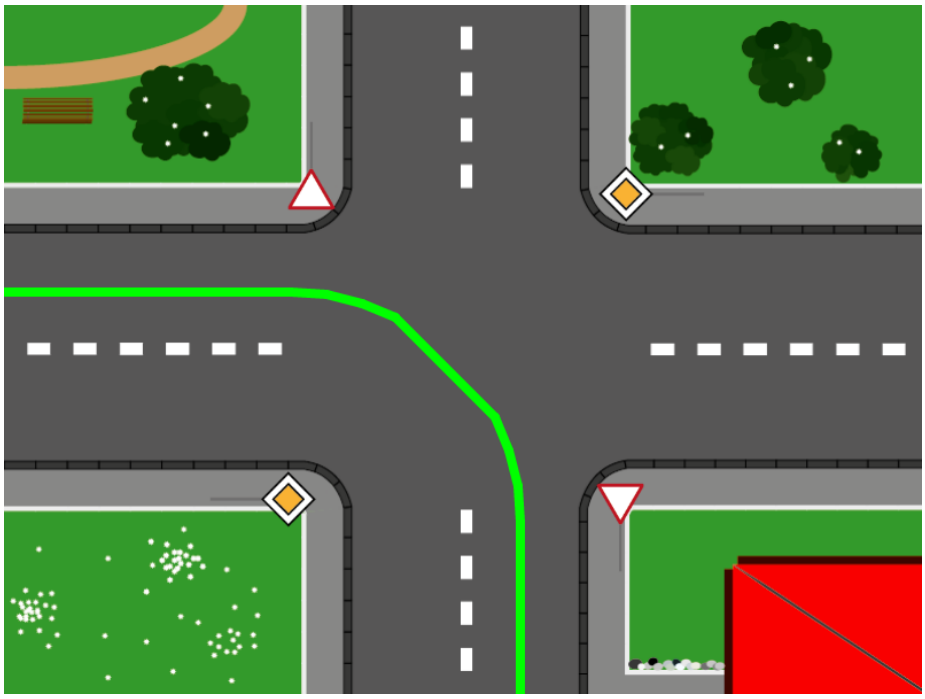
Metoda projde všechny pruhy v poli `sitPruhu`. Pokud se najde takový pruh `p2`, že jeho začáteční souřadnice odpovídají koncovým souřadnicím daného pruhu `p1`, přidá pruh `p2` do pole `p1.nextPruhy` a naopak pruh `p1` přidá do pole `p2.previousPruhy`.

Takzvaný *spread operátor*, „tři tečky“, je způsob, jak v JavaScriptu přidat prvek do pole. Tedy zápis `arr = [...arr, item]` přidá prvek `item` do pole `arr`.[11]

4.4 Třída Spawn

Termínem `spawn` v našem textu rozumíme místo, kde alespoň jeden pruh začíná a žádný pruh nekončí. Je to instance třídy `Spawn`.

Taková místa jsou na křižovatce 3 nebo 4, každé auto poté začíná na jedné z těchto pozic. V sekci „Vlastní situace“ je možné přidávat nová auta také pouze na tyto pozice.



obrázek 9: Trasa v křižovatce

4.5 Trasa

Auto může z jedné začáteční pozice dojet na různá místa. Jako trasu označujeme pole obsahující několik na sebe navazujících pruhů, které vedou vždy z jednoho spawnu do jednoho konečného místa.

Z každého spawnu potom vede tolik tras, do kolika konečných míst se z něj lze dostat. Na obrázku **9** jsou pruhy jedné takové trasy zvýrazněny.

4.5.1 Metoda najitCile()

Metoda, která najde jednotlivá místa, na která je možné dojet z jedné začáteční pozice (respektive ze začátečního pruhu), je na ukázce **8**. Pokud daný pruh nemá žádné pruhy, které by z něj pokračovaly, přidá se do pole `arrayCilu` jako jeden z cílových pruhů. Jinak se metoda `najitCile()` zavolá pro všechny pruhy, které z něj pokračují.

Hodnota `visited` je zde proto, aby se na případném kruhovém objezdu bralo v úvahu pouze jedno projetí tímto kruhovým objezdem a zamezilo se tak vyvolávání metody `najitCile()` „donekonečna“.

```
najitCile(arrayCilu) {
  this.visited = 1;
  if (this.nextPruhy.length == 0) {
      arrayCilu.push(this);
  } else {
    this.nextPruhy.forEach(pruh => {
      if (!pruh.visited) {
          pruh.najitCile(arrayCilu);
      } else {
          this.visited = 0;
      }
    })
  }
}
```

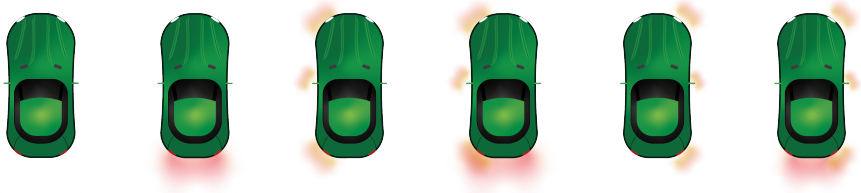
ukázka kódu 8: Metoda `najitCile()`

5 Třída Auto

Jako další je zde třída `Auto`. V této třídě je popsána logika chování aut spolu s pravidly provozu. Každé auto (autem v našem textu rozumíme instanci třídy `Auto`) má potom několik vlastností:

- `trasa` – Pole po sobě jdoucích pruhů, po kterých se bude auto pohybovat.
- `pruh` – Pruh (instance třídy `Pruh`), na kterém se auto aktuálně nachází.
- `pos` – Pozice auta (objekt třídy `p5.Vector`).
- `autaKterymDavaPrednost` – Pole obsahující auta, kterým dané auto při průjezdu křižovatkou musí dát přednost.
- `autaKIgnorovani` – Pole obsahující auta, na která dané auto při průjezdu křižovatkou nemusí brát zřetel. (Jejich trasy se nijak nekříží.)
- `textura` – Obrazová (rastrová) textura daného auta. Textura auta je pole obsahující 6 obrázků ve formátu PNG, jednotlivé obrázky reprezentují:
  - auto jedoucí rovně,
  - brzdící auto,
  - auto s levým blinkrem,
  - auto s levým blinkrem a brzdovým světlem,
  - auto s pravým blinkrem,
  - auto s pravým blinkrem a brzdovým světlem.

  Na obrázku **10** můžeme vidět, jak taková textura vypadá.
- `aktualniTextura` – Jeden z šesti výše uvedených obrázků, který se v daný snímek animace ukazuje.



obrázek 10: Textura auta

5.1 Pohyb aut

Nově vytvořené auto se nachází na prvním pruhu své trasy, neboli na pruhu `trasa[indexTrasy]`, kde `indexTrasy = 0`.

Hodnota `progress` říká, jakou část z aktuálního pruhu dané auto už urazilo. Pokud platí, že `progress >= 1`, znamená to, že auto dorazilo na konec pruhu a aktuálním pruhem se stane pruh následující.

Pokud chceme, aby se auto pohybovalo, stačí zvětšovat `progress` o `aktualniRychlost`. Pokud by `progress` překročilo 1, zvětšíme `indexTrasy` o 1 a `progress` vynulujeme.

Zrychlování, případně zpomalování, zajistíme zvětšováním či zmenšováním hodnoty `aktualniRychlost`, kterou přičítáme k `progress`. To je vidět v metodě `go()` na ukázce **9**.

Pokud může dané auto jet dopředu (metoda `muzeJet()` vrátí `true`), hodnota `aktualniRychlost` se může zvětšovat, dokud nedosáhne hodnoty `maxRychlost`. To je maximální rychlost, jakou se může auto pohybovat. V opačném případě se `aktualniRychlost` zmenšuje, dokud nedojde k nule.

```
go() {
  let increment = this.maxRychlost / 20;
  if (this.muzeJet()) {
    this.brzdi = 0;
    if (this.aktualniRychlost + increment < this.maxRychlost) {
      this.aktualniRychlost += increment;
    } else {
      this.aktualniRychlost = this.maxRychlost;
    }
  } else {
    this.brzdi = 3;
    if (this.aktualniRychlost - increment > 0) {
      this.aktualniRychlost -= increment * 1.5;
    } else {
      this.aktualniRychlost = 0;
    }
  }
  this.progress += this.aktualnirychlost;
}
```

ukázka kódu 9: Metoda `go()`

Pokud auto dojede na konec posledního pruhu, odstraní se z vozovky.

5.2 Pravidla provozu

Všechna auta mají určeno několik pravidel, podle kterých řídí své chování.

Každé auto má své pole `autaKterymDavaPrednost`, které obsahuje auta, kterým dané auto při průjezdu křižovatkou musí dát přednost. Dané auto může projet křižovatkou, je-li jeho pole `autaKterymDavaPrednost` prázdé a nestojí-li mu jiné auto v cestě.

Klíčové je tedy zjistit, kdy danému autu přidat jiné auto do pole `autaKterymDavaPrednost`.

Pravidla jsme se snažili formulovat tak, aby co nejvíce odpovídala realitě. Mějme `auto1` a `auto2`. `Auto1` do svého pole `autaKterymDavaPrednost` přidá `auto2`, pokud:

- je `auto2` na hlavní silnici, zatímco `auto1` na hlavní silnici není, a trasy těchto aut se kříží.

- jsou obě auta buďto na hlavní, nebo na vedlejší silnici, a `auto2` přijíždí vůči `autu1` zprava nebo jede proti `autu1` a `auto1` odbočuje vlevo.

Pokud se trasy těchto aut nijak nekříží, `auto2` se může přidat do pole `auto1.autaKIgnorovani` (a obráceně).

Algoritmus, který tato pravidla realizuje, je na ukázce **10**.

Auto `auto1` prochází postupně všechna ostatní auta, které jsou na vozovce. Pokud auto `auto1` ještě danému autu `auto2` nedává přednost, a ještě se neukázalo, že trasy těchto aut se nekříží, nebo že auta jedou do společného místa, přejde se na rozhodnutí, zda dá `auto1` přednost `autu2`, či nikoli.

Pokud je `auto1` na vedlejší silnici, zatímco `auto2` je na hlavní, pak dává `auto1` automaticky přednost `autu2`.

Pokud jsou obě auta na silnici stejné úrovně, tak aby `auto1` dalo přednost `autu2`, musí platit, že `auto1` jede vůči `autu2` zleva.

Situace, kde `auto1` dává `autu2` přednost zprava, nebo kde `auto1` odbočuje vlevo a proti němu jede `auto2`, není potřeba rozlišovat. V obou případech totiž `auto1` vjíždí zleva do cesty `autu2`.

```
this.ostatniAuta.forEach(auto => {
  if (!this.autaKterymDavaPrednost.includes(auto) &&
      !this.autaKIgnorovani.includes(auto)) {
        let spolecnaCast = this.jedouNaStejneMisto(auto)
                           || this.kriziSeTrasy(auto);
        if (spolecnaCast) {
          if (this.pruh.hlavni < auto.pruh.hlavni) {
            this.autaKterymDavaPrednost =
              [...this.autaKterymDavaPrednost, auto];
          }
          else if ((this.pruh.hlavni == auto.pruh.hlavni) &&
            this.jedeZleva(spolecnaCast, auto)) {
              this.autaKterymDavaPrednost =
                [...this.autaKterymDavaPrednost, auto];
          }
        }
        else {
          this.autaKIgnorovani = [...this.autaKIgnorovani, auto];
        }
  }
})
```

ukázka kódu 10: Přidání aut do `autaKterymDavaPrednost`

5.3 Další metody třídy Auto

Součástí třídy `Auto` je také několik metod.

5.3.1 Metoda jeZacyklene()

Na ukázce **11** je metoda `jeZacyklene()`, která vrátí hodnotu `true`, pokud se dané auto nachází v jakési nekonečné smyčce, kde jedno auto dává přednost

druhému autu, to dává přednost třetímu a třetí auto dává přednost autu prvnímu. (Více v kapitole 5.4.)

Výchozí hodnota `visited` pro všechna auta je `false`. Pokud algoritmus dojde k autu, které už bylo navštíveno, vrátí hodnotu `true`. Pokud dojde k ještě nenavštívenému autu, nastaví tato metoda `visited` na `true` a následně vyvolá metodu `jeZacyklene()` na všechna auta, kterým dává dané auto přednost.

Toto se opakuje, dokud metoda nenarazí na auto, které bylo už navštíveno, nebo dokud se u všech aut, kterým dané auto dává přednost, zjistí, že nejsou ve smyčce. V tomto případě vrací `false`.

```
jeZacyklene() {
  if (this.visited) {
    return true;
  } else {
    this.visited = true;
    for (let i = 0;
      i < this.autaKterymDavaPrednost.length; i++) {
      let a = this.autaKterymDavaPrednost[i];
      if (a.jeZacyklene())
        return true;
      this.autaKterymDavaPrednost.forEach(auto => {
        auto.visited = false;
      })
    }
    return false;
  }
}
```

ukázka kódu 11: Metoda `jeZacyklene()`

5.3.2 Metoda kriziSeZleva()

Další metodou je metoda `kriziSeZleva()`, která vrátí hodnotu `true`, pokud se trasa jednoho auta kříží s trasou druhého auta tak, že první auto jede vůči druhému zleva (a musí mu tedy dát přednost). Metoda je na ukázce **12**.

Metoda má jako argument 2 pruhy. Z pruhu `p2` přijíždí `auto1`, z pruhu `p1` pak `auto2`. pokud je z-souřadnice vektorového součinu těchto pruhů menší než 0, pak `auto1` přijíždí vůči `autu2` zleva.

```
kriziSeZleva(pruhy) {
  const p2 = pruhy[0];
  const p1 = pruhy[1];

  const temp = p2.start.copy().sub(p1.start);
  const cross = p1.size.cross(temp).z;

  return cross < 0;
}
```

ukázka kódu 12: Metoda `kriziSeZleva()`

5.3.3 Metoda blikatSBlinkrem()

Metoda `blikatSBlinkrem()` (na ukázce **13**) každé půl sekundy střídavě vypíná a zapíná blinkr auta, které právě zatáčí.

Klíčové slovo `async` značí metodu, která neblokuje další zpracování hlavního toku programu. [12]

U daného auta se zobrazuje blinkr, je-li hodnota `blinkrON true`. Metoda `blikatSBlinkrem()` počká 1 sekundu a poté zamění vypnutý blinkr za zapnutý, a obráceně. Metoda `blikatSBlinkrem()` se zavolá znova, pokud dané auto pořád zatáčí.

```
async blikatSBlinkrem() {
  await this.timeout(500);
  this.blinkrON = !this.blinkrON;
  if (this.zataceni)
    this.blikatSBlinkrem();
}
```

ukázka kódu 13: Metoda `blikatSBlinkrem()`

5.3.4 Metoda show()

Pro zobrazení daného auta na `canvas` je použita metoda `show()`. Můžete ji vidět na ukázce **14**.

Pokud blinkr auta zrovna svítí, vybere se textura auta se zapnutým blinkrem. Pokud ne, vybere se textura bez blinkru. Vlastnost `brzdi` určuje, zda se bude zobrazovat brzdové světlo. Vybraná textura se potom zobrazí na pozici `pos` daného auta a otočí se o úhel `dir.heading()`.

```
show() {
  if (this.blinkrON) {
    this.aktualniTextura =
      this.textura.dostat(this.brzdi + this.zataceni);
  } else {
    this.aktualniTextura = this.textura.dostat(this.brzdi);
  }
  push();
  imageMode(CENTER);
  translate(this.pos.x, this.pos.y);
  rotate(HALF_PI + this.dir.heading());
  image(this.aktualniTextura, 0, 0);
  pop();
}
```

ukázka kódu 14: Metoda `show()`

5.4 Problémová situace

Ačkoliv ve většině situací tato pravidla fungují velice dobře, může se stát, že někdy se s těmito pravidly moc daleko nedostanete. Příkladem je situace na obrázku **11**.

obrázek 11: Ukázka problémové situace

Modré auto přijíždějící zleva dává přednost zprava zelenému autu. Zelené auto také dává přednost zprava, a to bílému autu. Bílé auto však odbočuje vlevo, a musí tedy dát přednost v jízdě protijedoucímu modrému autu.

Dostali jsme se tedy do situace, kde si všechna auta navzájem dávají přednost a ve výsledku nikdo křižovatkou neprojede. Jak se tedy v dané situaci zachovat?

Učebnicové řešení je následující: Bílé auto si najede do křižovatky tak, aby za ním mohlo odbočit zelené auto. Poté projede modré auto a nakonec může bílé auto dokončit odbočování.[13]

Ne vždy však auto jedoucí zprava může najet do křižovatky tak, aby ještě nechalo dostatek místa k projetí auta jedoucího zespodu. To se může velice jednoduše stát třeba s ohledem na velikost nákladního automobilu.

V reálném provozu se poté taková situace dá řešit tak, že některý z řidičů se prostě vzdá své přednosti v jízdě. Toto řešení jsme se také rozhodli použít v naší aplikaci. Jako testová otázka se zde tato situace zatím ale neobjeví.

6 Databáze

Naše aplikace využívá služby databáze, kterou poskytuje webhosting. Databáze slouží k zaznamenávání jednotlivých uživatelů a jejich dokončených testů a k ukládání dopravních situací, které se v testech vyskytují.

6.1 Tabulky dopravních situací

Databáze funguje jako úložiště dat pro dopravní situace, které se vyskytují v testech. Používáme systém tří tabulek, které jsou propojeny unikátním identifikačním číslem (`idsituace`), které má každá dopravní situace. Pomocí tohoto čísla systém pozná, ke které dopravní situaci patří dané auto či odpověď.

V hlavní tabulce jsou data, které se skládají z čísla mapy, která se využívá, textu otázky, a čísla správné odpovědi. Dále se také u každé situace zaznamenává, kolikrát u ní absolventi testu chybovali a každá má své identifikační číslo. Druhá tabulka pak obsahuje data pro vozidla, které se u otázky, zobrazí. Jedná se o jejich rychlost, začáteční pozici a trasu. Třetí tabulku tvoří možné odpovědi, které má uživatel na výběr.

6.1.1 Hlavní tabulka situací – situace

Name	Type	Collation	Null	Default	Extra
idsituace	int		No	None	Auto Increment, Primary Key
mapa	int		No	None	
spravnaodpoved	int		No	None	
otazka	varchar(255)	utf8_czech_ci	Yes	NULL	
kolikratspatne-zodpovezeno	int		No	0	

tabulka 1: Struktura atributů hlavní tabulky `situace`

`Name` označuje název atributu. `Type` označuje datový typ atributu, zpravidla se jedná o `int` či `varchar`. `Collation` určuje, jakou sadu znaků atribut používá, když datový typ atributu je např. `varchar`. `Null` určuje, jestli atribut může nabývat prázdné hodnoty. `Default` určuje základní hodnotu, ta se přednastaví atributu při zápisu do tabulky. `Extra` obsahuje informace, zda atribut má vlastnost `Auto increment` či `Primary key`.

Tato tabulka se skládá z 5 atributů. První atribut `idsituace` obsahuje unikátní identifikační číslo dané dopravní situace. Dále je zde přidána vlastnost `Auto Increment`, která automaticky vygeneruje unikátní číslo při novém zápisu do databáze, aby nenastala situace, kde budou mít 2 situace stejné identifikační číslo. Další vlastnost tohoto atributu je `Primary Key`, podle které se identifikuje každý záznam v tabulce. `Primary Key` musí být unikátní a nesmí být prázdný, a proto jsme vybrali tento atribut.

Všechny atributy jsou datového typu `int`[4] s výjimkou na text otázky (atribut `otazka`), který je typu `varchar`[5], protože se nejedná o číslo. Tyto dva datové typy využíváme i v nadále zmíněných tabulkách. Tento atribut také narozdíl od ostatních jako jediný může nabývat prázdné hodnoty. Tuto možnost má nastavenou pro případy, kdy není v testu třeba otázky, neboť uživatel musí vybrat z odpovědí, které popisují celkovou dopravní situaci, a není třeba nějak specifikovat, na co se ho aplikace ptá. Na místě otázky se v testu zobrazí pouze text „Vyberte správnou odpověď:“. Tato prázdná hodnota je v základu nastavena každé situaci, dokud se nepřepíše jinou otázkou.

Dále pak atribut `mapa` označuje číslo textury použité během situace, atribut `spravnaodpoved` je číslo odpovědi, která je správná. Atribut `kolikratspatnezodpovezeno` zaznamenává, kolikrát uživatelé chybovali v dané situaci. Funguje tak, že se atributu vždy na konci každého testu zvýší hodnota o 1, jestliže v ní uživatel udělal chybu. Při tvorbě nové situace v databázi se tomuto atributu přiřadí hodnota 0, dokud někdo nedokončí test, ve kterém by v této dopravní situaci chyboval.

idsituace	mapa	spravnaodpoved	otazka	kolikratspatne-zodpovezeno
1	2	1	NULL	0
2	2	1	Jako první projede křižovatkou:	4
3	2	2	NULL	3
4	2	2	NULL	4

tabulka 2: První 4 řádky tabulky `situace`

obrázek 12: Ukázka situace bez zadaného textu v otázce

[4]Datový typ `int` značí celé číslo.
[5]`Varchar` je datový typ, který se používá v MySQL k zápisu řetězců.

6.1.2 Tabulka aut

Tato tabulka obsahuje data ohledně automobilů v každé dopravní situaci. Má 6 atributů, kde žádný nesmí zůstat prázdný při zápisu do tabulky.

- `idsituace` – Identifikační číslo situace, do které auto patří.
- `spawn` – Určuje, z jaké cesty přijíždí auto do křižovatky.
- `trasa` – Číslo trasy, kterou se má vydat auto v křižovatce.
- `rychlost` – Určuje, jakou rychlostí se auto pohybuje.
- `prednostnijizda` – Určuje, o jakou skupinu vozidel se jedná.[6]
- `idauta` – Identifikační číslo daného auta, toto identifikační číslo má vlastnosti `Auto Increment` a `Primary Key`.

Name	Type	Collation	Null	Default	Extra
idsituace	int		No	None	
spawn	int		No	None	
trasa	int		No	None	
rychlost	int		No	None	
prednostnijizda	int		No	0	
idauta	int		No	None	Auto Increment, Primary Key

tabulka 3: Struktura atributů tabulky `autasituace`

6.1.3 Tabulka odpovědí

Tato tabulka obsahuje data odpovědí, které má uživatel při testu na výběr.

- `idsituace` – Identifikační číslo situace, ke které odpověď patří.
- `odpoved` – Obsahuje samotný text dané odpovědí k dopravní situaci.
- `idodpovedi` – Identifikační číslo odpovědi, má vlastnosti `Auto Increment` a `Primary Key`.

Name	Type	Collation	Null	Default	Extra
idsituace	int		No	None	
odpoved	varchar(255)	utf8_czech_ci	No	None	
idodpovedi	int		No	None	Auto increment, Primary Key

tabulka 4: Struktura atributů tabulky `odpovedisituace`

[6] Tento atribut má přednastavenou hodnotu 0, neboť většina vozidel jsou osobní vozidla. Například sanitní vůz má číslo 1.

6.2 Záznamy o uživatelích

Vzhledem k tomu, že naše stránka umožňuje založit si účet, tak musí také zahrnovat záznamy o uživatelských jménech a heslech. Při založení účtu ukládáme pouze zadané uživatelské jméno a heslo. Dále je pak každému uživateli přiděleno unikátní identifikační číslo, které byste našli v tabulce pod názvem `iduzivatele`.

Hesla zadána uživatelem jsou při zápisu do tabulky hashována pomocí PHP funkce `password_hash()`.

```
1   echo password_hash("33trpasliku", PASSWORD_DEFAULT);
2  /*
3  Output:
4  $2y$10$pnFpj0Bgi8zeedcoBqqq.OnSSs8xfBBAILMDSlnLSQ5LW8vgrnUHi
5  */
```

ukázka kódu 15: PHP metoda `password_hash()` a její výstup

Tabulka se skládá ze 3 atributů.

- `iduzivatel` – Identifikační číslo uživatele, toto identifikační číslo má vlastnosti `Auto Increment` a `Primary Key`.
- `username` – Text, ze kterého se skládá uživatelské jméno.
- `password` – Hashovaný text, ze kterého se skládá uživatelské heslo.

Name	Type	Collation	Null	Default	Extra
iduzivatel	int		No	None	Auto Increment, Primary Key
username	varchar(255)	utf8_czech_ci	No	None	
password	varchar(255)	utf8_czech_ci	No	None	

tabulka 5: Struktura atributů tabulky `uzivatele`

6.3 Záznamy o testech

Při dokončení testu jsou jeho výsledky zapsány do databáze. Používáme k tomu 2 tabulky, které jsou propojené přes unikátní číslo přidělené každému vykonanému testu (`idtestu`).

6.3.1 Tabulka testů

V této tabulce jsou uložené informace o dokončeném testu. Tyto informace se skládají z úspěšnosti absolventa, data, kdy byl test vykonán, typu testu a identifikátoru uživatele, který test vykonal. V případě, že během testu nebyl nikdo přihlášený, má atribut obsahující číslo uživatele hodnotu 0.

- `idtestu` – Identifikační číslo testu, které má vlastnosti `Auto Increment` a `Primary Key`.
- `typtestu` – Číslo, které určuje o typ testu.
- `iduzivatele` – Identifikační číslo uživatele, který test vykonal.
- `pocetspravnychodpovedi` – Číslo, které určuje počet otázek odpovězených správně.
- `datum` – Datum vykonání testu, které využívá datový typ `date` a může nabývat prázdné hodnoty.[7]

Name	Type	Collation	Null	Default	Extra
idtestu	int		No	None	Auto Increment, Primary Key
typtestu	int		No	None	
iduzivatele	int		No	None	
pocetspravnychodpovedi	int		No	None	
datum	date		Yes	NULL	

tabulka 6: Struktura atributů tabulky `zaznamytestu`

6.3.2 Tabulka situací v testu

V druhé tabulce jsou pak samotné dopravní situace, z kterých se test skládal. U nich je také zapsáno, kterou odpověď uživatel zvolil. Dále je také nutné zaznamenat, jaké pořadí měla daná dopravní situace v testu, aby bylo možné při zpětném zobrazení onoho testu, vypsat situace v takovém pořadí, v jakém se vytvořily během testu.

- `idtestu` – Identifikační číslo testu, ve kterém se situace vyskytla.
- `idsituace` – Identifikační číslo situace, jaká se vyskytla v testu.
- `idodpovedi` – Identifikační číslo odpovědi, kterou uživatel vybral.

[7] Možnost nabývání prázdné hodnoty jsme zde dali, protože v době, kdy jsme přidali atribut datum, databáze už obsahovala nějaké množství testů, u kterých nebylo zaznamenáno datum vykonání testu. A protože jsme nechtěli o záznamy těchto testů přijít, nechali jsme atributu `datum` možnost nabývat prázdné hodnoty.

- `poradivtestu` – Číslo, které určuje v jakém pořadí se tato otázka zobrazila uživateli v testu.[8]

Atribut `idsituace` označuje číslo situace z tabulky **1**: `situace`. Oba atributy `idtestu` a `idsituace` mají vlastnost `Primary Key`. Dohromady zaručují, že se bude jednat o unikátní zápis v tabulce, a to z důvodu, že každá situace se může v každém testu vyskytnout pouze jednou, takže nikdy nenastane situace, kdy by jeden test obsahoval dvakrát stejnou situaci.

Name	Type	Collation	Null	Default	Extra
idtestu	int		No	None	Primary Key
idsituace	int		No	None	Primary Key
idodpovedi	int		No	None	
poradivtestu	int		No	None	

tabulka 7: Struktura atributů tabulky `zaznamyodpovedi`

6.4 Záznamy o chybách

Uživatelé naší aplikace mají možnost během testu nahlásit chybu, na kterou narazili. V naší poslední tabulce se nachází tyto nahlášení.

- `idchyby` – Identifikační číslo chyby, které má vlastnosti `Auto Increment` i `Primary Key`.
- `typ` – Číslo, které určuje, zda se jedná o chybu v otázce, či jiného formátu.
- `popis` – Popis chyby, co se uživateli zdá chybné.
- `idsituace` – V případě, že se jedná o chybu v otázce, zapíše se i `idsituace`, u které se chyba nachází.

Name	Type	Collation	Null	Default	Extra
idchyby	int		No	None	Auto Increment, Primary Key
typ	int		No	None	
popis	varchar(1000)	utf8_czech_ci	No	None	
idsituace	int		Yes	NULL	

tabulka 8: Struktura atributů tabulky `reporttable`

[8] Tento atribut je zde proto, aby se při zpětném zobrazení ukázaly otázky v takovém pořadí, v jakém je měl uživatel v původním testu.

7 Zpětná vazba

7.1 Testování uživateli

Naši aplikaci jsme dali k vyzkoušení několika uživatelům[9] a poté jsme provedli úpravy na základě jejich zpětné vazby.

Uživatelé nám vytýkali především rozložení prvků na stránce; na malou obrazovku se všechny ovládací prvky nevešly, na velkém monitoru byly jednotlivé části stránky až moc daleko od sebe. Aplikaci jsme se proto snažili optimalizovat pro obrazovku běžného notebooku či počítače.[10]

Dále jsme přidali vysvětlení, proč by si měl uživatel zakládat účet, jelikož to v předchozí verzi aplikace nebylo dostatečně jasně řečeno.

Také se objevovaly připomínky k ovládání v sekci vlastních situací, které zde bylo pro některé dost matoucí a nejasné. Proto jsme celý systém přidávání aut do dopravních situací přepracovali, aby byl pro uživatele intuitivnější.

Zpětná vazba však neobsahovala jen kritiku – uživatelé oceňovali především grafiku, plynulé animace dopravních situací a možnost tvorby vlastních situací.

7.2 Komunikace s autoškolami

Také jsme oslovili zhruba 20 autoškol, které jsme požádali o zpětnou vazbu na náš projekt. Některé z nich se nám poté ozvaly. Naši aplikaci hodnotili vesměs kladně, objevovaly se zde však některé připomínky, ty bychom mohli shrnout následovně:

- výraznější směrovky,
- vyšší frekvence blikání směrovek,
- absence dopravních situací s tramvajemi,
- absence dopravních situací s vozidly s právem přednostní jízdy,
- absence kategorie složitých dopravních situací (případně s více možnými odpověďmi),
- málo realistický vzhled animací.

Podle těchto připomínek jsme zvýraznili směrovky a zvětšili frekvenci jejich blikání tak, aby uživatel jednoduše poznal, zda auto právě zatáčí.

Také jsme do našeho systému přidali situace, ve kterých se objevují vozidla s právem přednostní jízdy, které zde předtím chyběly.

Také se nám povedlo spojit s panem Davidem Chmelou z Autoškoly Chmela Otrokovice, který nám sdělil své postřehy.

Například navrhl zaměřit naši práci pro děti v mateřských školách, které by mohla zaujmout interaktivita naší aplikace, a to zejména možnost vytvářet

[9] Těmito uživateli byli hlavně především spolužáci a lidé z našeho okolí.

[10] Vzhledem k povaze naší aplikace, kde je potřeba, aby bylo dostatečně zřetelné jak to, co se odehrává v animačním okně, tak i její ovládání, jsme brali v potaz obrazovky s rozlišením 1280 × 720 a výše, uživatelé stolních počítačů či notebooků s nižším rozlišením obrazovky mají navíc malé zastoupení (kolem 5 %).[14][15][16][17]

vlastní dopravní situace. Tento nápad se nám velice líbil, proto jsme s naším projektem oslovili mateřské školy, ty se nám však z důvodu jejich uzavření kvůli epidemické situaci zatím nestihly ozvat.

U jedné z autoškol se objevila výtka, že naše aplikace nemá dostatečně „realistický“ vzhled. Kvůli důvodu možného zaměření naší aplikace na mladší uživatele jsme se však rozhodli ponechat vizuální stránku naší aplikace spíše v „kresleném“ stylu. Myslíme si totiž, že tento grafický styl by mohl více zaujmout právě žáky mateřských škol a zároveň pro ostatní uživatele je tento styl dostatečně názorný.

Také nám doporučil přidat teoretické otázky s odpověďmi typu ano/ne, začátečnickou verzi testu s nejjednoduššími otázkami[11] nebo sekci „Výuka“, kde by byla základní pravidla provozu sepsána v odrážkách. Tyto změny jsme však bohužel zatím nestihli uskutečnit.

Upozornil nás na to, že v samotném testu se jednotlivá auta rozlišují barvami, my je v naší aplikaci označujeme čísly. Chtěli jsme však vyjít vstříc lidem s poruchou barevného vidění. I když by se někomu mohlo zdát, že je to zbytečné, protože takový člověk stejně řidičský průkaz nezíská, chtěli jsme, aby i tito lidé naši aplikaci mohli využívat – a když už ne jako řidiči, tak jako cyklisté nebo spolujezdci.

[11] To jsou otázky typu: *Které auto se nachází na hlavní silnici?*

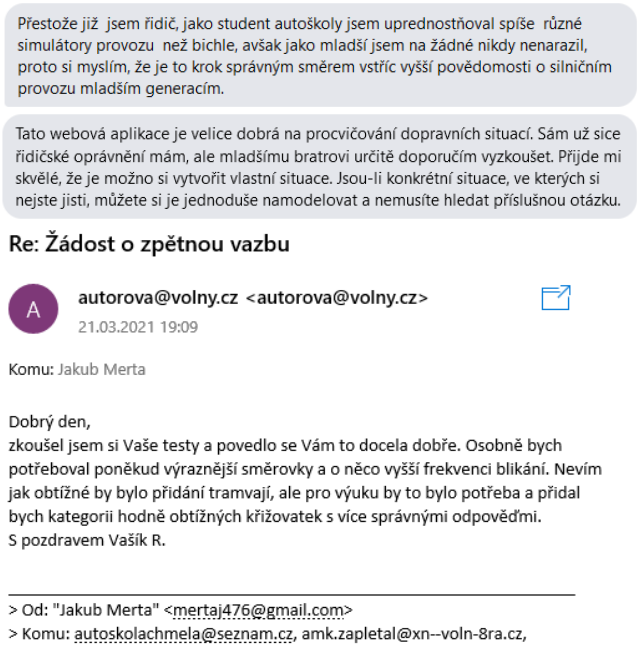
8 Závěr

Cílem naší práce bylo vytvořit pro uživatele atraktivní aplikaci zabývající se dopravními situacemi na křižovatkách, a věříme, že se nám to povedlo.

Uplatnění této aplikace vidíme například:

- pro uživatele připravující se na získání řidičského průkazu,
- pro uživatele s poruchou barevného vidění,[12]
- jako způsob, jak zábavnou vizuální formou seznámit žáky mateřských škol s dopravními pravidly,[13]
- jako pomůcku do hodin dopravní výchovy na základní škole.

To, že naše aplikace má reálné uplatnění a přínos, dokazuje jak kladná odezva dotazovaných autoškol, tak i zpětná vazba od ostatních uživatelů, která nám dělá radost (obrázek **13**).



obrázek 13: Zpětná vazba od uživatelů

8.1 Zkušenosti

Účastí na SOČ jsme se také něco naučili. Zkusili jsme si, jaké je to pracovat v týmu (byť jen dvoučlenném), a naučili jsme se používat službu GitHub.

[12] Naše aplikace potskytuje výhodu v označování jednotlivých aut v testu čísly, oproti běžnému způsobu označení aut barvami. A tím uživatele s poruchou barevného vidění nijak neznevýhodňuje.

[13] Grafické prostředí naší aplikace může mlaší uživatele zaujmout.

Mohli jsme zde zúročit vědomosti nabyté v hodinách informatiky – HTML, CSS a PHP. Zjistili jsme, jak důležité je umět si sám dohledávat informace, také jsme napsali delší práci v systému LaTeX, a to všechno v jednom projektu.

Jsme rádi, že jsme se do SOČ zapojili a že nám už nikdo nemůže tuto zkušenost vzít.

8.2 Budoucí plány

V naší aplikaci vidíme potenciál pro zlepšení její kvality a přidání nového obsahu. Na zlepšování kvality již pracujeme – co se týče připomínek uživatelů, přidali jsme do naší dopravní situace s vozidly s právem přednostní jízdy, upravili jsme rozhraní naší aplikace tak, aby bylo pro uživatele příjemnější a přehlednější a v neposlední řadě jsme zvýraznili směrovky.

Jsou zde však další možnosti, kterými můžeme úroveň naší aplikace zlepšit.

8.2.1 Mobilní verze

Prozatím jsme nepovažovali za důležité vytvořit mobilní verzi naší aplikace.

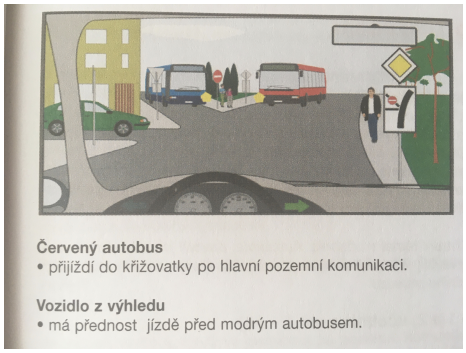
Vzhledem k tomu, že to nejdůležitější se odehrává právě v animačním okně, kde musí být všechny prvky (auta, blinkry, dopravní značky) dostatečně zřetelné, nepovažujeme potencionální prohlížení naší stránky na mobilním telefonu za ideální způsob. Navíc, samotná interakce s animačním oknem v sekci „Vlastní situace“ by byla pro uživatele bez myši velmi náročná.

I přes tyto nevýhody mobilní verze však uvažujeme pro vytvoření takové verze naší aplikace, kdy bude do jisté míry použitelná i na mobilním telefonu či tabletu.

8.2.2 Přidání více otázek a situací

Naše aplikace obsahuje dopravní situace s křižovatkami typu X a T. Po prohledání testových otázek v kapitole „Dopravní situace a křižovatky“ v učebnici autoškoly[13] jsme z celkového počtu 40 otázek zabývajících se předností na křižovatkách našli pouze jednu otázku s typem křižovatky, který se v našem systému neobjevuje, viz obrázek **14**.

Naše aplikace je však vytvořená obecně (na tvaru křižovatky nezáleží), a tak přidání nových typů křižovatky do systému není problém.



obrázek 14: Chybějící křižovatka

8.2.3 Tramvaje

Jsme si vědomi toho, že v naší aplikaci chybí dopravní situace s tramvajemi, které se v testu nejspíše objeví. Některé autoškoly nepovažovaly za nijak důležité přidání dopravních situací s tramvajemi. My bychom přesto tramvaje do naší aplikace rádi doplnili, jelikož se s nimi uživatel ve skutečném testu nejspíše setká.

9 Bibliografie

[1] *Wikipedie: Otevřená encyklopedie: JavaScript* [online]. ©2021 [citováno 28. 2. 2021]. Dostupné z: `https://cs.wikipedia.org/w/index.php?title=JavaScript&oldid=19439460`

[2] JavaScripting.com – The Database of JavaScript Libraries. *JavaScripting.com – The Database of JavaScript Libraries* [online]. ©2014 Salsita Software. All rights reserved. [cit. 29. 3. 2021]. Dostupné z: `https://www.javascripting.com/`

[3] HTML Canvas. *W3Schools Online Web Tutorials* [online]. [cit. 17. 3. 2021]. Dostupné z: `https://www.w3schools.com/html/html5_canvas.asp`

[4] *Wikipedie: Otevřená encyklopedie: PHP* [online]. ©2021 [citováno 1. 3. 2021]. Dostupné z: `https://cs.wikipedia.org/w/index.php?title=PHP&oldid=19316920`

[5] PHP: mysqli::query – Manual. *PHP: Hypertext Preprocessor* [online]. Copyright © 2001 [cit. 17. 3. 2021]. Dostupné z: `https://www.php.net/manual/en/mysqli.query.php`

[6] PHP: mysqli_result::fetch_assoc – Manual. *PHP: Hypertext Preprocessor* [online]. Copyright © 2001 [cit. 17. 3.2021]. Dostupné z: `https://www.php.net/manual/en/mysqli-result.fetch-assoc.php`

[7] Hash – digitální otisk, který záškodníci nepřečtou — Digitální pevnost. *Bojujeme za bezpečnější on-line svět — Digitální pevnost* [online]. [cit. 1. 3. 2021]. Dostupné z: `https://www.digitalnipevnost.cz/viki/hash`

[8] PHP: password_verify – Manual. *PHP: Hypertext Preprocessor* [online]. Copyright © 2001 [cit. 17. 3. 2021]. Dostupné z: `https://www.php.net/manual/en/function.password-verify.php`

[9] JSON. *JSON* [online]. [cit. 1. 3. 2021]. Dostupné z: `https://www.json.org/json-cz.html`

[10] *Wikipedie: Otevřená encyklopedie: JavaScript Object Notation* [online]. ©2020 [cit. 2. 3. 2021]. Dostupné z: `https://cs.wikipedia.org/w/index.php?title=JavaScript_Object_Notation&oldid=19182922`

[11] Kopírování objektů v JS. *Je čas.cz – moderní tvorba webových stránek* [online]. [cit. 20. 3. 2021]. Dostupné z: `https://jecas.cz/js-klonovani-objektu`

[12] JavaScript – asynchronní zpracování. *Webdesign, webové aplikace, grafika* [online]. [cit. 17. 3. 2021]. Dostupné z: `http://visionplus.cz/tutorialy/async-javascript.php`

[13] Autoškola. Praha: Business Media CZ, [2016]. *Vogel (Business Media CZ).* ISBN 978-80-87388-31-0.

[14] Laptop PC Display Specs (Size, Resolution...) Explained (2019) – Laptoping. *Laptoping – Laptop PCs Made Simple* [online]. Copyright ©2021 [cit. 10. 3. 2021]. Dostupné z: https://laptoping.com/laptop-display-specs-explained.html

[15] Desktop Screen Resolution Stats Worldwide — StatCounter Global Stats. *StatCounter Global Stats - Browser, OS, Search Engine including Mobile Usage Share* [online]. Copyright © StatCounter 1999 [cit. 10. 3. 2021]. Dostupné z: https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide

[16] Website Dimensions: 15 Screen Resolutions to Design For. *Top Agencies of 2021 by Category, Price, Location and More — DesignRush* [online]. Copyright ©[cit. 10. 3. 2021]. Dostupné z: https://www.designrush.com/trends/website-dimensions

[17] What Are The Best Screen Sizes For Responsive Web Design?. *Hobo Web: SEO Consultants (Scotland, Greenock)* [online]. Copyright ©2006 [cit. 10. 3. 2021]. Dostupné z: https://www.hobo-web.co.uk/best-screen-size/

[18] How to use recaptcha v2 on localhost? - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, and Build Careers [online].* [cit. 1. 3. 2021]. Dostupné z: https://stackoverflow.com/questions/46421887/how-to-use-recaptcha-v2-on-localhost

[19] Web Design in Milwaukee — *Making Spider Sense* [online]. [cit. 1. 3. 2021]. Dostupné z: https://www.makingspidersense.com/tutorials/tut-recaptcha.txt

[20] python - How can I check if two segments intersect? - Stack Overflow. *Stack Overflow - Where Developers Learn, Share, and Build Careers* [online]. [cit. 27. 2. 2021]. Dostupné z: https://stackoverflow.com/questions/3838329/how-can-i-check-if-two-segments-intersect

[21] analytic geometry - How to check if a point is inside a rectangle? - Mathematics Stack Exchange. *Mathematics Stack Exchange* [online]. [cit. 27. 2. 2021]. Dostupné z: https://math.stackexchange.com/questions/190111/how-to-check-if-a-point-is-inside-a-rectangle

[22] Hepunx.rl.ac.uk *WWW Server at RAL (28-Oct-1999)* [online]. [cit. 1. 3. 2021]. Dostupné z: https://hepunx.rl.ac.uk/~adye/jsspec11/titlepg2.htm

[23] LaTeX_Listings_JavaScript_ES6/js_listing.tex at master · ghammock/LaTeX_Listings_JavaScript_ES6 · GitHub. *GitHub: Where the world builds software · GitHub* [online]. Copyright © 2021 GitHub, Inc. [cit. 1. 3. 2021]. Dostupné z: https://github.com/ghammock/LaTeX_Listings_JavaScript_ES6/blob/master/js_listing.tex

Seznam obrázků

Seznam ukázek kódu

Seznam tabulek

Přílohy

1. Zdrojový kód aplikace – soubor zdrojovykod.zip.