

# *Slovenská technická univerzita v Bratislave*

*Fakulta informatiky a informačných technológií*

Ilkovičova 2

842 16 Bratislava 4

---

Tímový projekt 2016/2017 - dokumentácia k inžinierskemu dielu

## **Tím 13: EduSim**

ŠTUDENTI:

ADAM BLAŠKO

MARTIN CVIČELA

IVAN GULIS

TOMÁŠ LIŠČÁK

BRANISLAV MAKAN

MAREK MATULA

VEDÚCI TÍMOVÉHO PROJEKTU:

ING. MAREK LÓDERER

KONTAKTNÝ EMAIL: [mambit@googlegroups.com](mailto:mambit@googlegroups.com)

## Obsah

1	Úvod k inžinierskemu dielu.....	4
2	Globálne ciele.....	5
2.1	Zimný semester .....	5
2.2	Letný semester.....	5
3	Celkový pohľad na systém .....	6
3.1	Dátový model - vrstvomá štruktúra .....	6
3.2	Model architektúry - MVC.....	7
3.3	Diagram modulov systému.....	8
3.4	Diagram prípadov použitia .....	8
3.5	Diagramy tried.....	9
3.6	Zoznam priložených elektronických dokumentov .....	10
4	Opis modulov systému .....	11
4.1	Build a inštalácia .....	11
4.1.1	Build .....	11
4.1.2	Inštalácia - Windows .....	14
4.2	Moduly používateľského rozhrania.....	14
4.2.1	Používateľské rozhranie [Audi, Java].....	14
4.2.2	Toolbox a Debug Windows [Bentley].....	18
4.2.3	Popupy a tlačidlá [Eagle, Honda, Java].....	20
4.2.4	Kontextové menu [Ferrari] .....	23
4.2.5	Lokalizácia - resource files [Bentley].....	25
4.3	Moduly používateľskej interakcie .....	25
4.3.1	File browsing [Honda].....	26
4.3.2	Create, Save, Load [Eagle, Ferrari, GAZ, Honda] .....	27
4.3.3	Pohyb [Audi, Bentley, GAZ].....	30
4.3.4	Kolízie [Bentley, GAZ] .....	32
4.3.5	Vytvorenie inštancií komponentu [Audi] .....	32
4.3.6	Select [Audi, GAZ] .....	33
4.3.7	Deselect [Bentley, GAZ] .....	34
4.3.8	Multiselect [Ferrari, GAZ] .....	34
4.3.9	Mazanie inštancií komponentu [Bentley].....	35
4.3.10	Rotácia [Audi, GAZ] .....	36
4.3.11	Vykresľovanie čiar [Bentley, Cadillac] .....	36
4.3.12	Object Duplication [Ferrari].....	40
4.3.13	Undo Redo [GAZ, Honda, Infiniti] .....	41

4.3.14	CameraZoom [Audi] .....	43
4.3.15	Pohyb kamery [Bentley].....	43
4.3.16	Background [Bentley] .....	44
4.3.17	Grid [Audi, Bentley].....	44
4.4	Moduly simulačnej logike .....	45
4.4.1	Súčiastky [Audi].....	45
4.4.2	Back-End simulácie elektrického obvodu [Bentley] .....	46
4.4.3	Logika súčiastok [Bentley, Cadillac].....	47
4.4.4	Simulácia el. obvodu v grafickom prostredí [Cadillac][Dodge] .....	50
4.4.5	Meracie zariadenia [Dodge] .....	51
4.4.6	Zachytávanie zlého zapojenia obvodu [Honda] .....	51
4.4.7	Upozornenia na chybné zapojenia obvodu [Honda].....	51
4.4.8	Export do HTML.....	52
5	Inštalčná príručka a manuál .....	54
5.1	Inštalácia.....	54
5.2	Grafické rozhranie .....	54
5.3	Hlavné menu.....	54
5.3.1	Súbor .....	55
5.3.2	Upraviť .....	55
5.3.3	Zobrazenie .....	56
5.3.4	Pomoc .....	56
5.4	Kontextové menu .....	56
5.5	Používateľská príručka (manuál).....	57
5.5.1	Panel s hlavnými nástrojmi .....	57
5.5.2	Ovládacie prvky pracovnej plochy.....	57
5.5.3	Simulácia elektrických obvodov .....	61
5.5.4	Vytvorenie, načítanie a uloženie projektu .....	63

# 1 Úvod k inžinierskemu dielu

V súčasnosti väčšina prednášok je vo forme prezentácií. Typicky, študenti sedia v aule a počúvajú prednášajúceho. Takáto forma prednášok zvykne byť nezaujímavá a vyčerpávajúca, keďže študent pri tom nič nerobí. Prezentácie teda nie sú najlepšou formou na udržanie pozornosti študenta. Prednášky sa stávajú trápnosťou a študenti sa im začnú vyhýbať. Je známe, že najlepší spôsob učenia, je skúšať si veci.

Keďže sa hry celkovo stali súčasťou moderného života mnohých ľudí, v našom projekte sa snažíme spojiť tie dve činnosti dokopy. Vytváraný nástroj by mal mať formu hry a minimálne interaktívnym spôsobom obohatiť proces výuky pre študentov vysokých škôl, ale aj mladších žiakov. Tento nástroj bude zároveň slúžiť aj ako pomôcka na vytváranie učebných materiálov pre pedagógov.

Tento projekt vznikol v spolupráci s firmou Atos. Hlavná idea je vytvoriť nástroj, pomocou ktorého sa budú vytvárať interaktívne simulácie, pomocou ktorých sa budú môcť testovať aj vedomosti žiakov a študentov. Vytvorené simulácie sa budú exportovať do HTML5 webových stránok, čo umožní jednoduchý prístup každému, keďže stačí mať prehliadač s pripojením na internet. Testovacie moduly budú prepojené s existujúcimi testovacími systémami.

Softvér je implementovaný v Unity game engine. Bude podporovať viaceré edukačné moduly, ktoré sa budú ľahko vymieňať. V prvej iterácii, teda v rámci predmetu tímový projekt, sa vytvorí jadro nástroja a modul pre elektrické obvody.

## **2 Globálne ciele**

Cieľom nášho projektu je vytvorenie autoringového nástroja, ktorý bude slúžiť ako nástroj na zlepšenie výučby rôznych predmetov. Ako taký musí byť nástroj všeobecný a schopný simulovať rôzne domény akými sú napríklad elektrotechnika, fyzika, chémia a ďalšie. Z toho vyplýva, že nástroj musí byť modulárny a rozširiteľný. Na základe preferencií zákazníka - product ownera sa ako prvá doména vyvinie elektrotechnika a ďalšie domény sa podľa preferencií vyvinú dodatočne. Tieto autoringové simulácie musí byť možné z nástroja vyexportovať vo forme HTML stránky. Samotný nástroj bude postavený na platforme Unity.

### **2.1 Zimný semester**

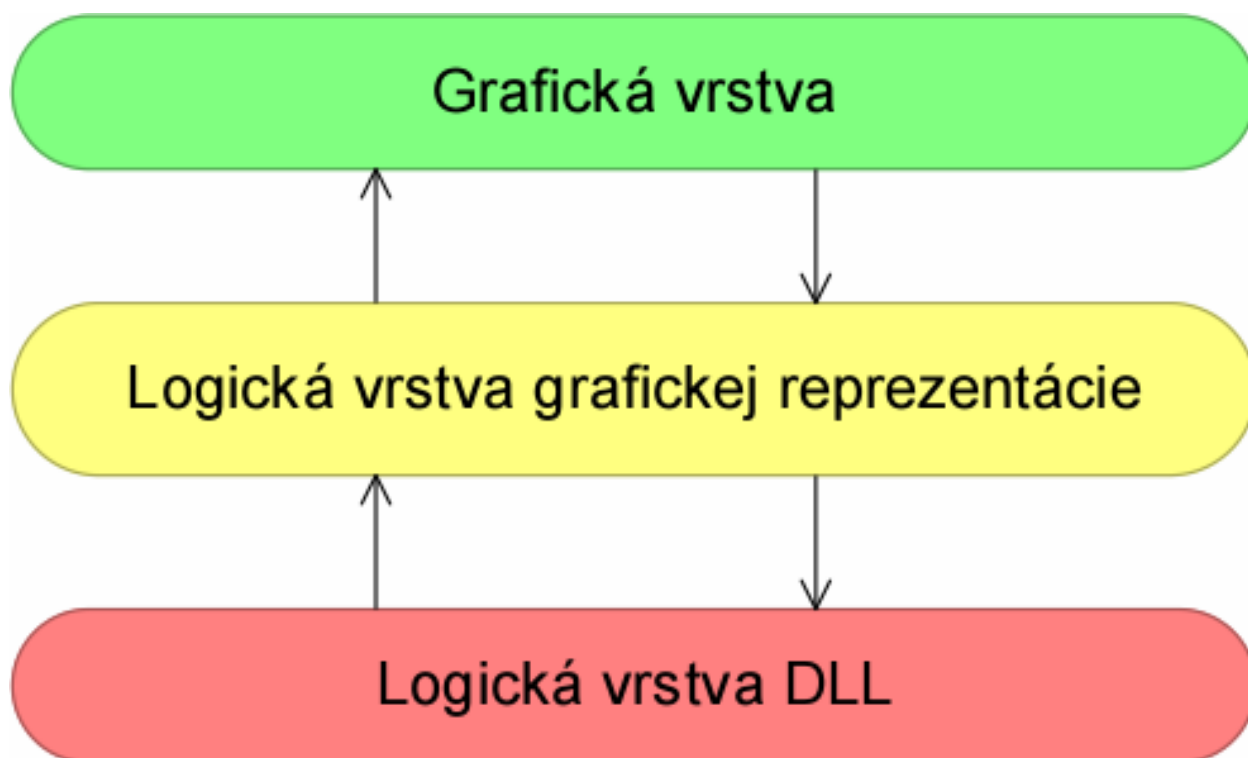
Základným cieľom nášho projektu pre zimný semester je vytvorenie funkčného prototypu výsledného produktu - MVP. Tento funkčný prototyp musí byť použiteľný produkt, ktorého využitie v praxi má zmysel aspoň do istej miery. Z toho vyplýva, že náš funkčný prototyp musí byť schopný modelovať a simulovať doménu elektrotechniky. Okrem funkčnej simulácie musí produkt samozrejme poskytovať aj podpornú funkcionálnu, bez ktorej by simulácia sama o sebe nemala žiaden význam. Takouto podpornou funkcionálnou sa myslí používateľské rozhranie, funkčná pracovná plocha na modelovanie elektrotechnických obvodov, na ktorých sa simulácia bude vykonávať, grafická reprezentácia elektrotechnických súčiastok a možnosť nastavovania parametrov týchto súčiastok.

### **2.2 Letný semester**

Ciele pre letný semester budú priebežne formulované product ownerom. Medzi všeobecné ciele však patrí integrácia so vzdelávacími systémami (napr. moodle), možnosť testovania študentov v nástroji, prípadne pridanie ďalších domén pre simulácie - napr. fyzika, dynamika, chémia. V rámci letného semestra sa taktiež zúčastníme súťaže TP cup, tým pádom ďalším cieľom je príprava na túto súťaž.

### 3 Celkový pohľad na systém

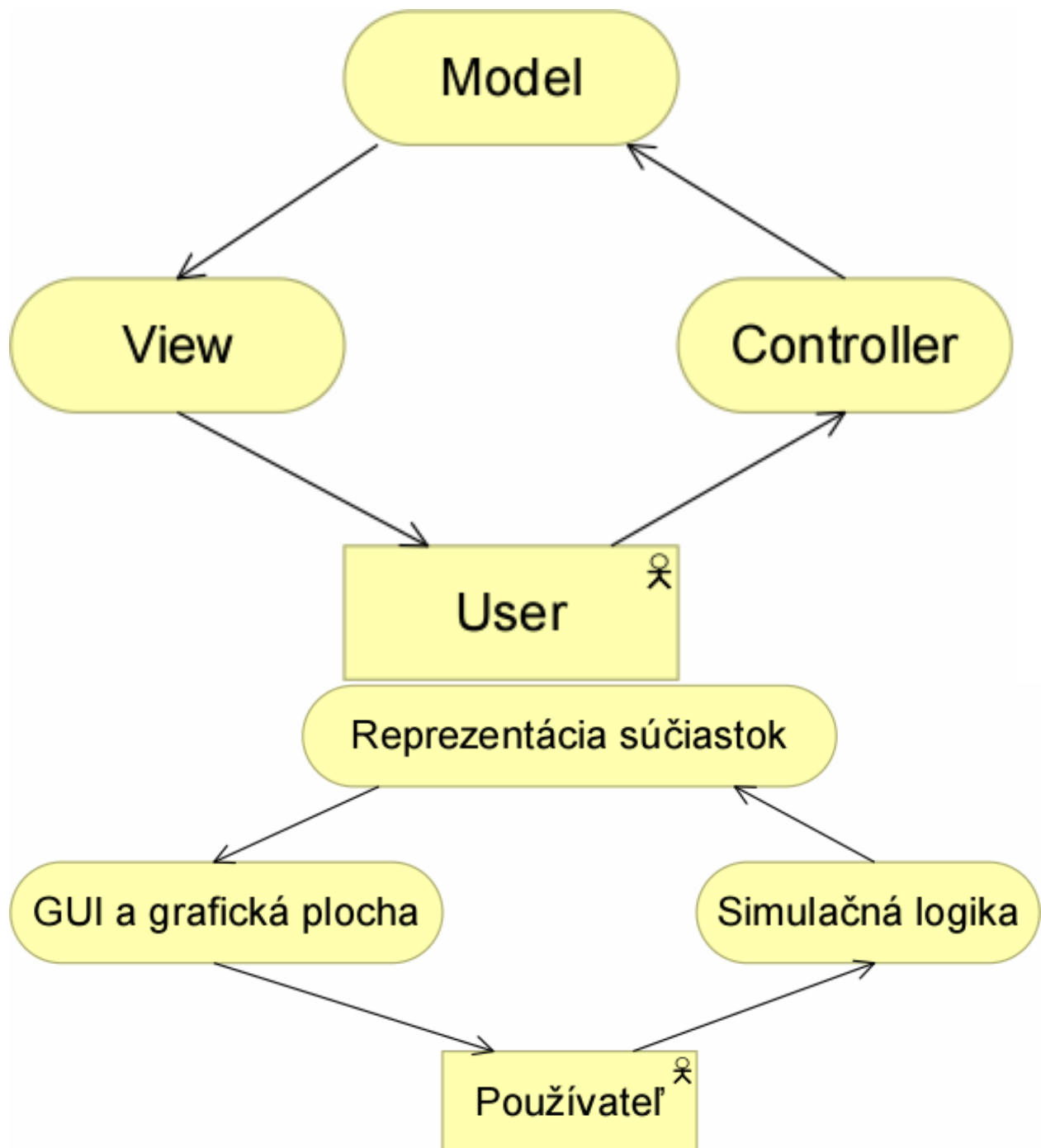
#### 3.1 Dátový model - vrstvová štruktúra



V našom systéme sme identifikovali vrstvovú architektúru.

1. **Grafická vrstva** - Komunikuje s Logickou vrstvou grafickej reprezentácie, ktorej poskytuje súbor vykreslených súčiastok na pracovnej ploche, ako aj informácie o ich umiestnení, logickom prepojení a používateľom nastavené atribúty.
2. **Logická vrstva grafickej reprezentácie** - Komunikuje s grafickou vrstvou a vytvára programovú reprezentáciu objektov pracovnej plochy a simulácie ako celku. Spúšťa samotnú simuláciu pomocou výpočtových funkcií Logickej vrstvy DLL. Pre použitie týchto funkcií tiež mapuje všetky získané grafické súčiastky na ich objektové reprezentácie v DLL.
3. **Logická vrstva DLL** - Poskytuje algoritmy na výpočty požadovaných hodnôt. Informácie posíla do Logickej vrstvy grafickej reprezentácie, ktorá aktualizuje programovú reprezentáciu a následne všetko zobrazí v Grafickej vrstve.

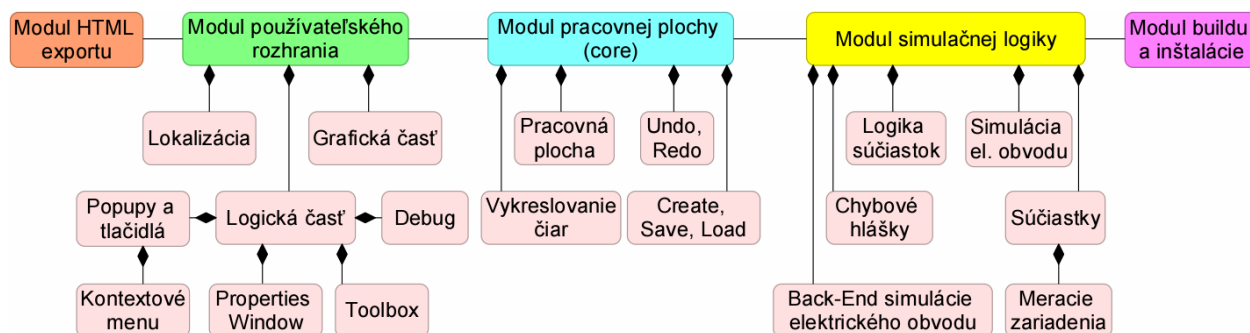
### 3.2 Model architektúry - MVC



Ako architektonický model používame architektúru MVC (Model-view-controller).

1. **Reprezentácia súčiastok** - Komponent Model, ktorý uchováva všetky informácie o súčiastkach a objektoch, s ktorými simulácia pracuje. Poskytuje pre View informácie, ktoré je potrebné zobrazit'.
2. **GUI a grafické plocha** - Komponent View, ktorý používateľovi zobrazuje informácie.
3. **Simulačná logika** - Komponent Controller, ktorý riadi celú simuláciu a manipuláciu so súčiastkami. Poskytuje používateľovi funkcie nad súčiastkami.

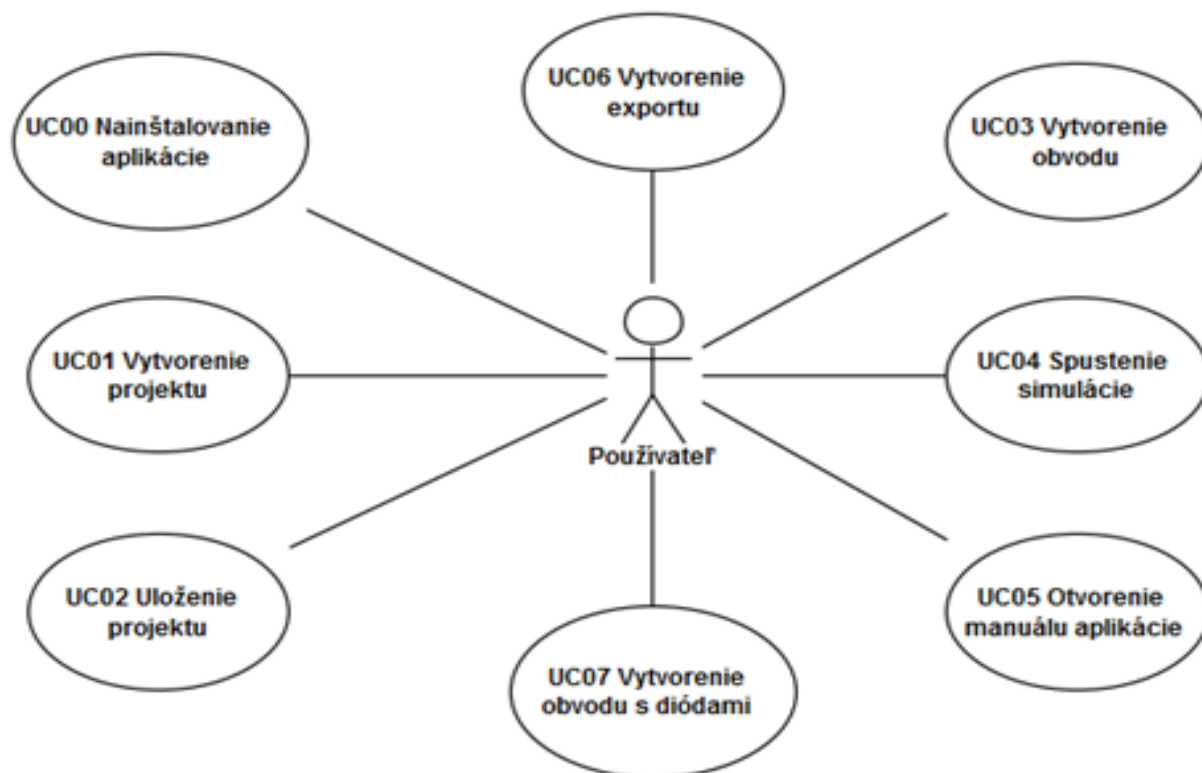
### 3.3 Diagram modulov systému



Náš vytváraný systém v súčasnosti obsahuje 3 väčšie moduly, ktoré sa delia na menšie časti:

1. **Modul používateľského rozhrania** - V tomto module sa vyvíja všetko ohľadom GUI. Od tvorby hlavného rozhrania, cez lokalizáciu (slovenčina-angličtina), po implementáciu jednotlivých komponentov (Properties, Toolbox, Debug...).
2. **Modul pracovnej plochy (core)** - V tomto module sa vyvíja všetko ohľadom pracovnej plochy. To zahŕňa funkcie na prácu s objektami a funkcie na vykresľovanie a lámanie čiar na spájanie objektov.
3. **Modul simulačnej logiky** - V tomto module sa vyvíja všetko ohľadom simulačnej logiky. Obsahuje grafový algoritmus na odstraňovanie uzlov, knižnicu na výpočtové algoritmy, mapovanie a programovú reprezentáciu súčiastok elektrického obvodu.

### 3.4 Diagram prípadov použitia





### 3.5 Diagramy tried

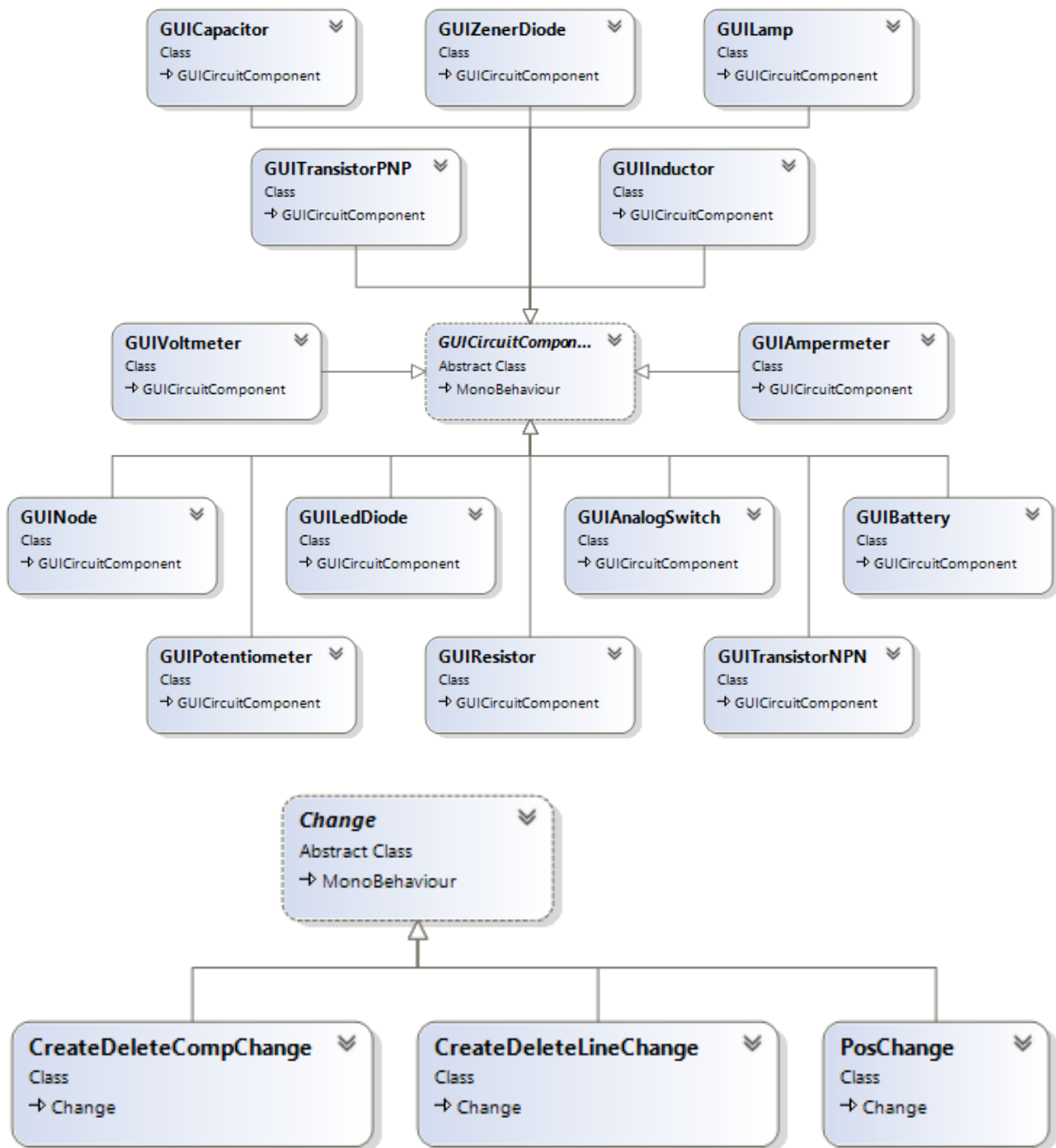


Diagram tried sa skladá z dvoch častí:

1. Hierarchia **Komponentov** - obsahuje programovú reprezentáciu grafických súčiastok elektrického obvodu.
2. Hierarchia **Zmien** - obsahuje reprezentáciu zmien komponentov pre undo/redo. Komponenty môžu zmeniť hodnoty atribútov, byť vytvorené alebo zmazané, posunuté alebo môžu byť spojené či odpojené vodičmi.

### **3.6 Zoznam priložených elektronických dokumentov**

1. Motivačný dokument
2. Metodiky
3. Manuál
4. Scenáre prípadov použitia

## 4 Opis modulov systému

### 4.1 Build a inštalácia

#### 4.1.1 Build

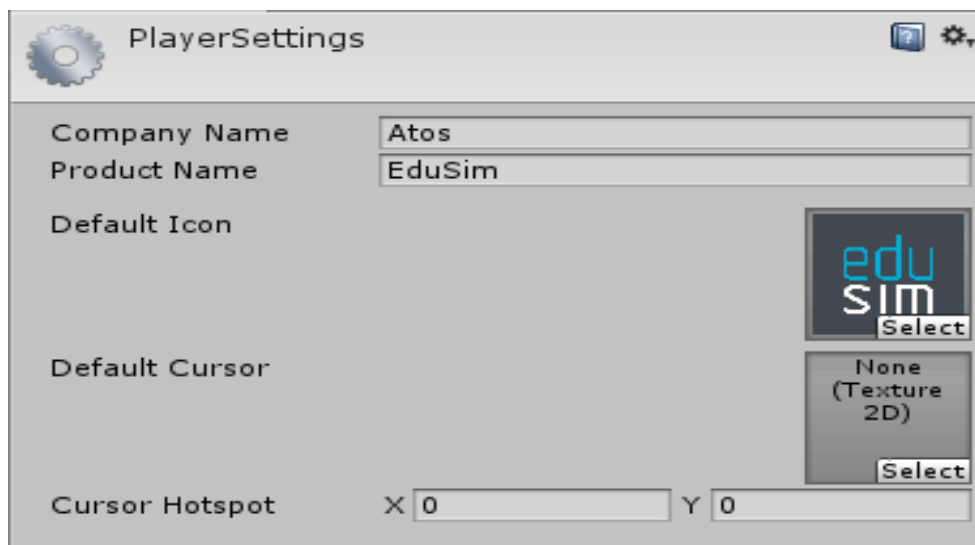
Buildovanie projektu znamená jeho zabalenie do spustiteľného súboru spolu so všetkými potrebnými knižnicami a pomocnými súbormi.

Buildovanie sa koná priamo v hernom engine Unity 3D. K nastaveniam buildu sa dá dostať nasledovne:

- otvoriť EduSim unity projekt,
- File > Build Settings > Player Settings.

V projekte sú nastavené nasledovné buildovacie nastavenia pre standalone spustiteľný EduSim program (nastavenia, ktoré tu nie sú uvedené, sme nemenili z ich prednastavených hodnôt):

- Základné nastavenia - meno, publisher, ikonky a kurzory myšky:



- Nastavenia pre PC, Mac a Linux - rozlíšenie:

Settings for PC, Mac & Linux Standalone

### Resolution and Presentation

#### Resolution

Default Is Full Screen\* ☐

Default Screen Width

Default Screen Height

Run In Background\* ☐

#### Standalone Player Options

Capture Single Screen ☐

Display Resolution Dialog

Use Player Log ☐

Resizable Window ☒

Mac App Store Validation ☐

Mac Fullscreen Mode

D3D9 Fullscreen Mode

D3D11 Fullscreen Mode

Visible In Background ☐

Allow Fullscreen Switch ☒

Force Single Instance ☒

▼ Supported Aspect Ratios

4:3 ☐

5:4 ☐

16:10 ☐

16:9 ☒

Others ☐

\* Shared setting between multiple platforms.

- Ostatné nastavenia - tuná je dôležité nastaviť **API Compatibility level** v časti *Optimization* na **.NET 2.0** (v opačnom prípade nefunguje EduSimFileExplorer - dovoľí otvoriť okno iba raz):

Other Settings

Rendering

Color Space\*
Gamma
Auto Graphics API for Win
Auto Graphics API for Mac
Static Batching
Dynamic Batching
GPU Skinning\*
Graphics Jobs (Experimental)
Virtual Reality Supported

Configuration

Scripting Backend
Mono2x
Disable HW Statistics\*
Scripting Define Symbols\*

Optimization

API Compatibility level\*
.NET 2.0
Prebake Collision Meshes\*
Preload Shaders
Preloaded Assets
Vertex Compression\*
Mixed ...
Optimize Mesh Data\*

Logging\*

Log Type	None	ScriptOnly	Full
Error	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Assert	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Warning	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Log	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Exception	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

\* Shared setting between multiple platforms.

Všetky pomocné súbory, ktoré nástroj potrebuje musia byť umiestnené na jednom z miest, ktoré sa s buildom exportujú. Takýto je priečinok *Assets/StreamingAssets*. Tento priečinok sa používa v hrách na rôzne videá cutsčénok. Aktuálne sa tu nachádzajú:

- Prvky HTML Exportu: HTML, CSS, Javascript súbory ako aj fonty a obrázky použité v exporte;
- Manuál: HTML, CSS, fonty a obrázky;
- Ukážky elektrických obvodov;
- Program EduSimFileExplorer.

### 4.1.2 Inštalácia - Windows

Tvorba inštalačného súboru je v podstate zabalenie buildu a pomocných súborov do spustiteľného súboru, ktorý pomocou natívneho Windowsáckeho inštalačného programu, nainštaluje nástroj EduSim na počítač používateľa.

Inštalačný súbor je generovaný pomocou nástroja [Inno Setup](#). Nástroj umožňuje pridať spustiteľný súbor a pomocné súbory v ľubovoľnej hierarchickej štruktúry do cieľového priečinka inštalácie. Výsledkom je spustiteľný súbor, ktorý nainštaluje program použitím inštalačného systému *Windows* - klasické inštalačné okno s voľbou cieľového inštalačného priečinku a vytvorenia ikony na desktope na žiadosť.

*Inno Setup* nastavenia sú uložené v skripte *edusim-software\Installer\EduSim\_inno\_setup.iss*. Po spustení skriptu sa v priečinku *edusim-software\Installer\Output* vytvorí spustiteľný súbor na inštaláciu nástroja.

## 4.2 Moduly používateľského rozhrania

### 4.2.1 Používateľské rozhranie [Audi, Java]

#### *GUI 1.0 [Audi]*

##### ANALÝZA

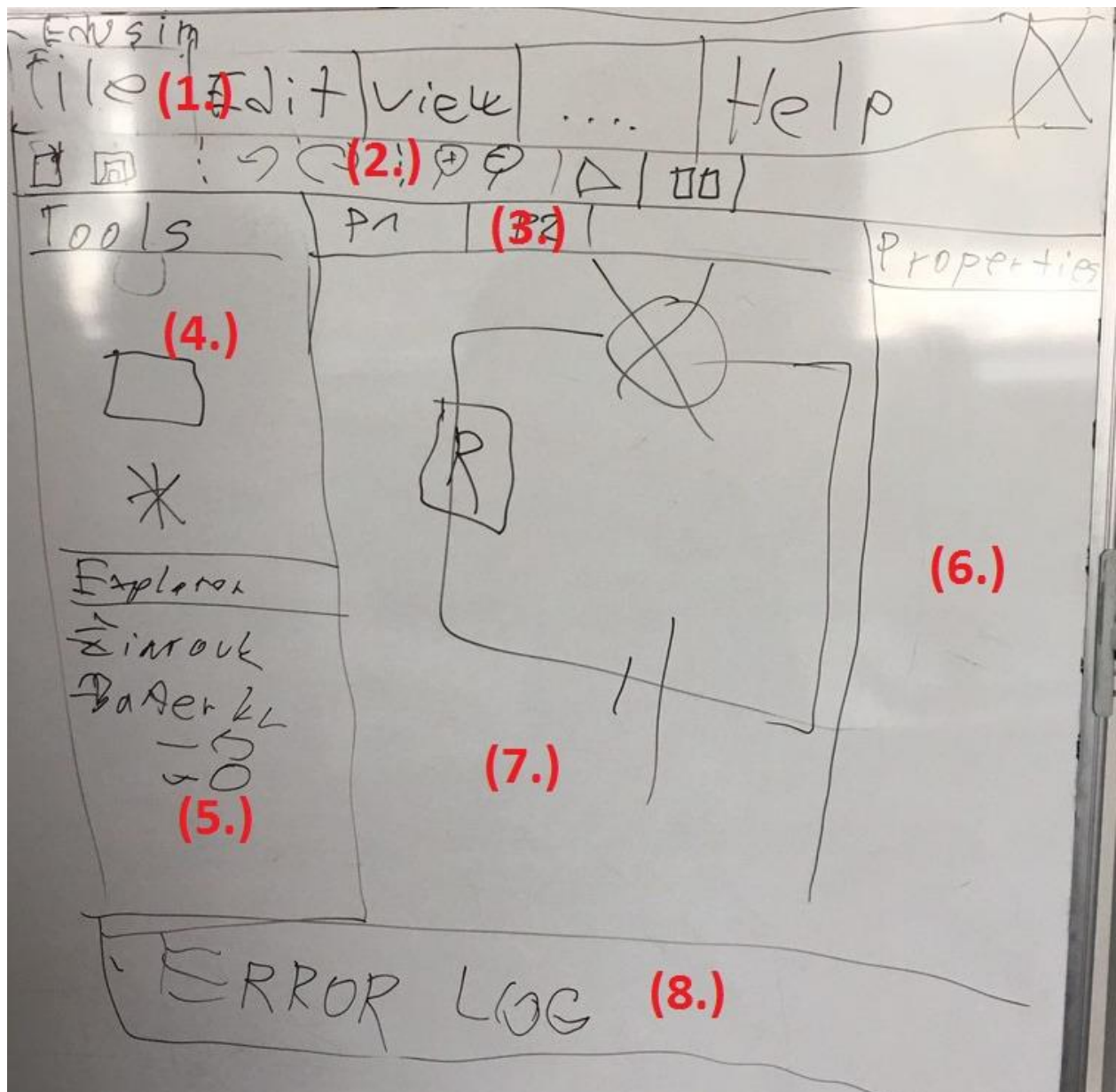
GUI predstavuje rozhranie, pomocou ktorého používateľ komunikuje so systémom.

Malo by spĺňať určité kritériá:

1. **intuitívnosť** - nápisy musia byť jednoznačné
2. **výstižnosť** - ikony tlačidiel musia vystihovať ich funkcionality
3. **familiárnosť** - kompozícia a vzhľad elementov by mal byť čo najviac podobný iným používaným aplikáciám

Používateľské rozhranie samotného implementačného prostredia Unity je možné do určitej miery zobrať ako vzor pre naše GUI.

Každá funkcia systému by mala mať aj svoj GUI komponent.

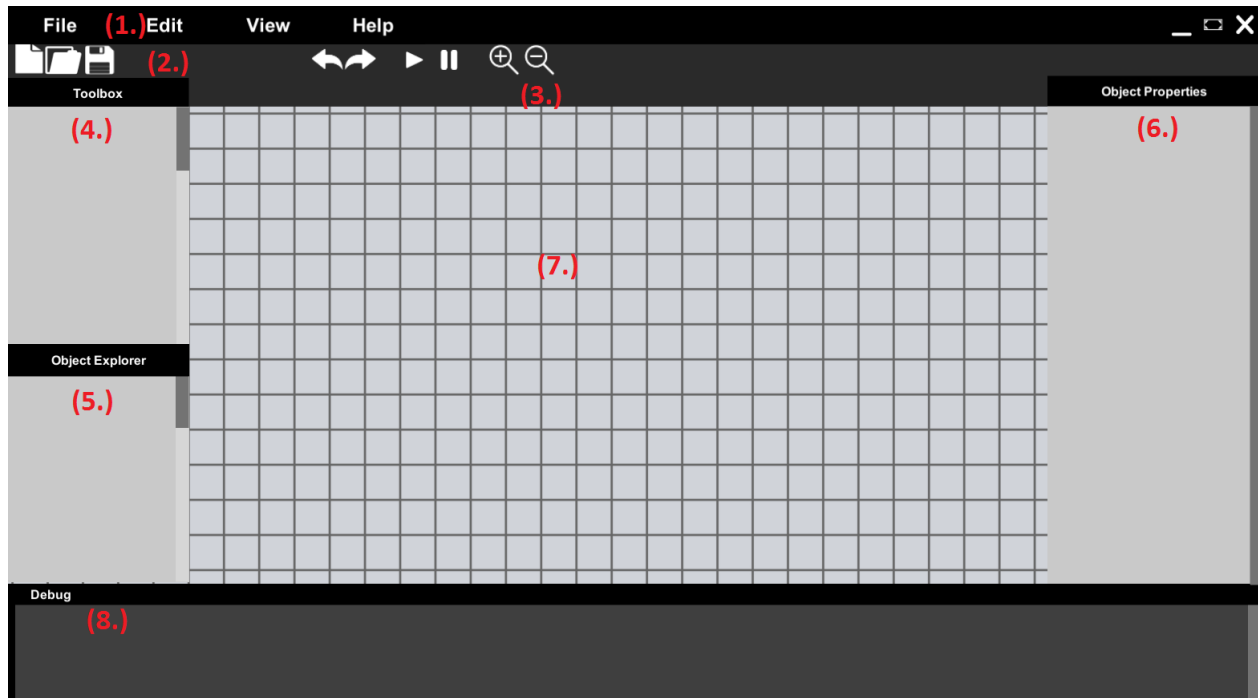


Jednoduchý návrh, ktorý sme načrtli na tabuľu, sa skladá z 8 väčších komponentov, ktoré sa ďalej členia na menšie objekty.

1. **Main menu** - panel by mal obsahovať základné textové menu s výberom možností po rozkliknutí
2. **Toolbar** - panel by mal obsahovať základné tlačidlá pre riadenie simulácie, grafickej plochy a projektu ako takého
3. **Tabs** - panel by mal obsahovať taby v prípade, že bude v jednom projekte existovať viac pohľadov
4. **Toolbox** - panel by mal obsahovať hlavičku s názvom a priestor pre všetky objekty, ktoré je možné pridávať do projektu na pracovnú plochu
5. **Object Explorer** - panel by mal obsahovať hlavičku s názvom a priestor pre objekty vytvoreného projektu
6. **Object Properties** - panel by mal obsahovať hlavičku s názvom a priestor pre atribúty označeného objektu z pracovnej plochy

7. **Pracovná plocha** - plocha s aktívnymi objektami projektu, s ktorými je možné manipulovať a simulovať ich správanie
8. **Error Log** - panel s priestorom na výpisy

## IMPLEMENTÁCIA



Každý komponent z návrhu má svoju grafickú implementáciu.

1. **Main menu** - obsahuje 7 tlačidiel:
  1. 4 tlačidlá na ľavej strane, ktoré po rozkliknutí otvoria svoje menšie menu textových tlačidiel
  2. 3 tlačidlá na pravej strane, ktoré reprezentujú bežné "Minimize", "Maximize" a "Close" funkcie
2. **Toolbar** - obsahuje 9 tlačidiel s ikonkami:
  1. prvé 3 reprezentujú funkcie pre ovládanie projektu ako celku, "New", "Open" a "Save"
  2. druhé 2 reprezentujú funkcie "Undo" a "Redo"
  3. tretie 3 reprezentujú funkcie "Play" a "Pause" pre samotnú simuláciu
  4. posledné 2 reprezentujú funkcie pre priblíženie a oddialenie pracovnej plochy
3. **Tabs** - neobsahuje žiadne prvky, taby budú generované skriptom
4. **Toolbox** - obsahuje hlavičku ako Text Field a skrolovaciu lištu na posúvanie obsahu
5. **Object Explorer** - obsahuje hlavičku ako Text Field a skrolovaciu lištu na posúvanie obsahu
6. **Object Properties** - obsahuje hlavičku ako Text Field a skrolovaciu lištu na posúvanie obsahu
7. **Pracovná plocha** - tento komponent bol implementovaný zvlášť [Pracovná plocha \[Audi, Bentley\]](#)
8. **Debug** - premenovaný z "Error Log", obsahuje panel s hlavičkou ako Text Field, skrolovacia lišta na posúvanie výpisov a Text Field na výpisy



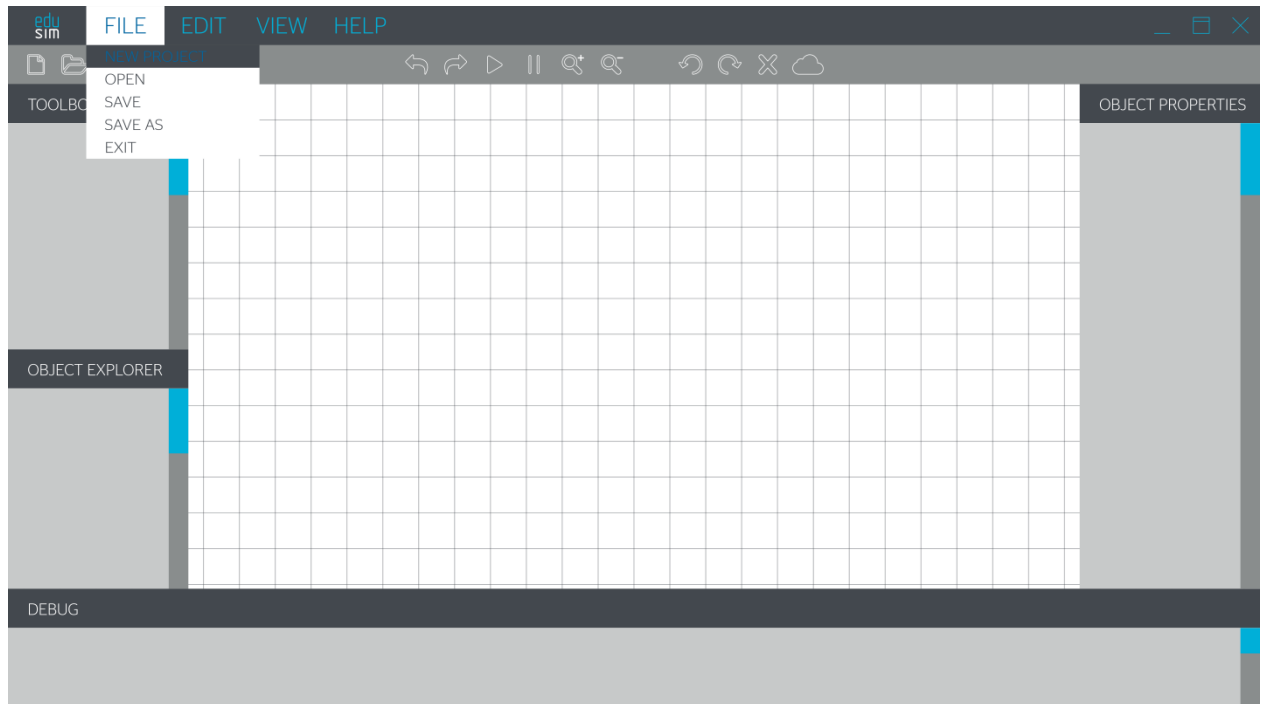
## GUI 2.0 [Java]

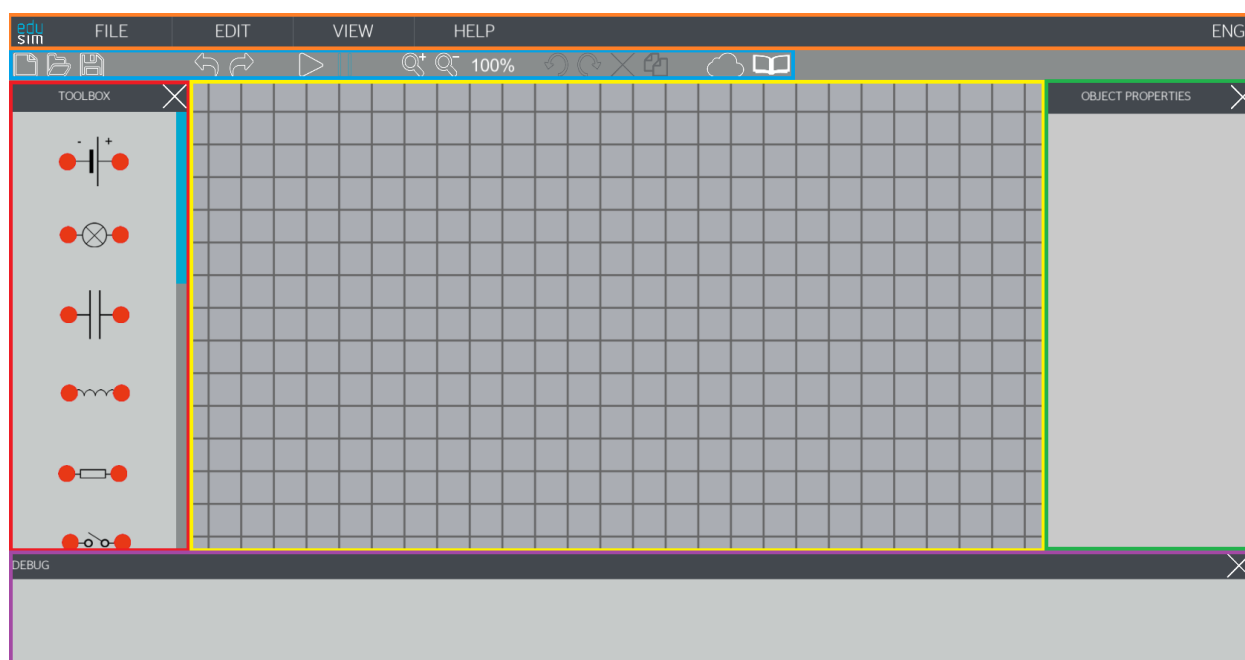
### ANALÝZA

Gui bolo potrebné redizajnovat' a odstrániť či pridať nepotrebné komponenty.

### NÁVRH

Návrh predstavoval súbor formátu .psd, ktorý obsahoval grafiku a farby pre všetky komponenty.





Farbami sú ohraničené rôzne časti simulátora. Jednotlivé časti sú nasledovné:

- Oranžová: Hlavné menu
- Modrá: Panel s hlavnými nástrojmi
- Červená: Panel so súčiastkami
- Žltá: Pracovná plocha
- Zelená: Vlastnosti súčiastky
- Ružová: Výpisy

Oproti pôvodnému GUI bol Object Explorer odstránený. Boli pridané ďalšie tlačidlá, ktorých funkcionality je popísaná v príslušných kapitolách.

## 4.2.2 Toolbox a Debug Windows [Bentley]

### *Analýza*

GUI - logická časť 1 sa skladá z implementácie dvoch hlavných komponentov - Toolboxu a Debugu.

**Toolbox** predstavuje komponent, v ktorom sú zhromaždené všetky použiteľné komponenty domény projektu. Tieto objekty používateľ presúva do pracovnej plochy, kde s nimi ďalej pracuje a tvorí podklad pre simuláciu.

Komponenty v Toolboxe nie je možné spájať, slúžia iba ako ikony. Tieto ikony potom generujú inštancie objektu, ktorý predstavujú.

Unity nepodporuje ako GUI elementy samotné objekty scény, takže Toolbox musí byť naplnený objektami s obrázkom ako atribútom.

Pohybovanie a klonovanie objektov už bolo implementované v [Pracovná plocha \[Audi, Bentley\]](#).

Hlavným problémom bude schovávanie objektov mimo Toolbox panelu, interagovať bude možné len s objektami v rámci Toolbox časti.

**Debug** panel predstavuje komponent, ktorý slúži na výpisy o udalostiach v simulácii.

Je nutné implementovať tzv. listener, ktorý sa bude starať o pridávanie správ do Text Fieldu a scrollovanie na aktuálnu správu.

### *Návrh*

Oba panely budú používať svoju skrolovaciu lištu, ktorá bude pripevnená k väčšiemu panelu na pozadí. Tento panel bude viditeľný len v rámci nehybnej vyrezanej časti, a bude sa posúvať vertikálne.

Do **Toolbox** panelu budú umiestnené jednotlivé objekty domény, ktoré budú mať nastavený tvoj tag označujúci predmet Toolboxu.

Každý objekt Toolboxu bude mať pridaný Image zhodný s textúrou, aby bol používateľom viditeľný. Pri vytiahnutí objektu z panelu sa mu zmení tag, takže sa bude správať ako objekt simulácie.

Do **Debug** panelu bude umiestnený jeden Text Field, kam sa budú skriptom posielať výpisy. Pre simuláciu výpisov dostane každý objekt domény skript na posielanie správ.

Funkcie Debugu však budú môcť používať aj priamo súčiastky, bez nutnosti vlastnenia skriptu na posielanie správ.

### *Implementácia*

**Toolbox** je implementovaný pomocou komponentu Scroll Rect, ktorý obsahuje Viewport, Masku a skrolovaciu lištu. Marka zakrýva všetko mimo Viewportu.

Viewport obsahuje panel ako kontajner, v ktorom sú umiestnené jednotlivé objekty domény. Pri pohybovaní skrolovacej lište sa tento panel pohybuje.

Všetky objekty v kontajneri majú tag ToolboxItemActive, čo umožňuje zakázať pohyb, a umožňuje vytvárať jeho klony (inštancie). Zároveň majú tieto objekty vypnutý Sprite Renderer, a zapnutý Image, takže sú vidieť.

Pri vytiahnutí objektu z Toolboxu sa skriptom aktivuje Sprite Renderer a zmení sa tag na ActiveItem, takže už nie je možné objekt ďalej klonovať, a je možné s ním interagovať funkciami pracovnej plochy.

Zabránenie interakcie s objektami mimo Viewportu je zabezpečené pomocou komponentu na kamere - 2D Raycaster. Tento komponent povoľuje interakciu len s objektami viditeľnými kamerou.

### 4.2.3 Popupy a tlačidlá [Eagle, Honda, Java]

#### *Analýza*

Kompletizácia grafiky 1. časť sa skladá z 3 pod-častí:

1. Pop-up okná pre tlačidlá z hlavného menu
2. Interaktivita tlačidiel hlavného menu
3. Interaktivita tlačidiel toolbaru

Okrem tlačidiel na tvorbu pop-upov musia byť interaktívne aj tlačidlá pre zobrazovanie/skrývanie bočných panelov Toolbox, Debug, Properties. Tlačidlá Play a Pause v menu Edit musia kopírovať funkcionality tlačidiel Play a Pause z toolbaru - viditeľne označené, kedy sú stlačené. Tlačidlá Otvoriť, Uložiť a Uložiť ako musia interagovať rovnako ako tlačidlá z toolbaru - otvárať windows explorer.

Tlačidlá na otáčanie a mazanie súčiastok by sa mali zobrazovať len ak je nejaká súčiastka označená.

#### *Návrh*

Zoznam pop-up panelov, budú otvárané z hlavného menu tlačidlami rovnakých názvov:

- Nastavenia - nastavenia programu
- O programe - panel s opisom programu
- Ukážky zapojení - panel s pred-pripravenými obvodmi

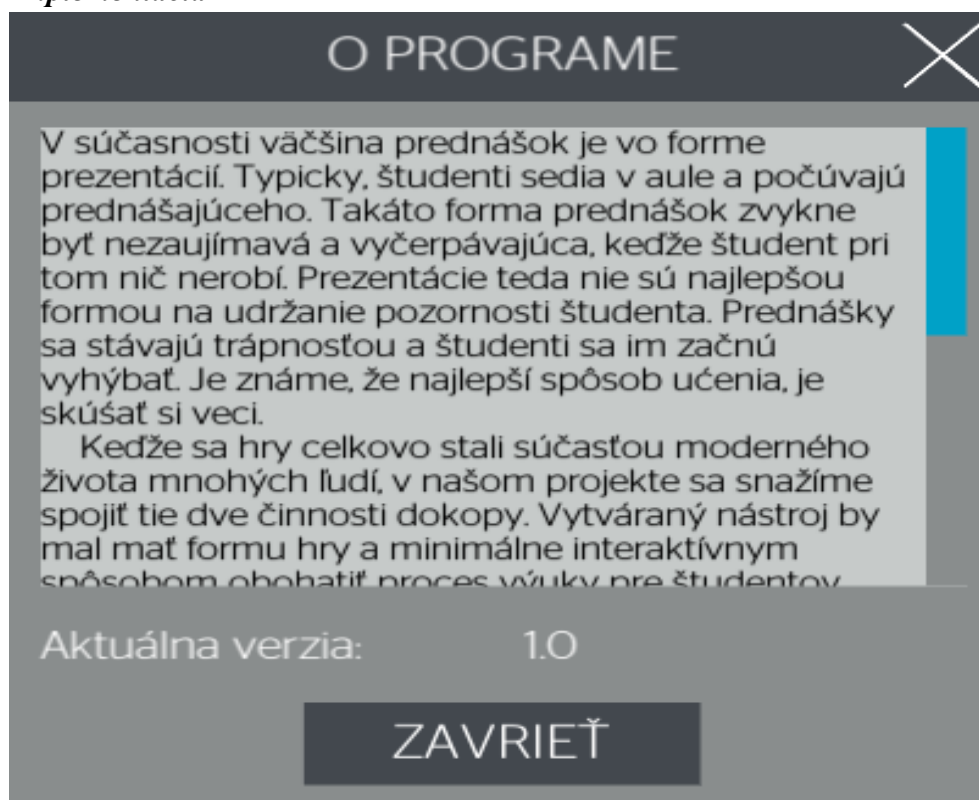
Zoznam tlačidiel hlavného menu, ktoré budú interaktívne:

- Nový projekt - otvorí nový projekt
- Otvoriť, Uložiť, Uložiť ako - otvoria windows explorer
- Spustiť, Pozastaviť - spustia alebo pozastavia simuláciu
- Tlačidlá View menu - budú ukazovať a skrývať bočné panely

Zoznam tlačidiel toolbaru, ktoré budú interaktívne:

- Nový projekt, Otvoriť, Uložiť, Uložiť ako, Undo, Redo, Play, Pause, Priblíženie, Oddialenie, Otočenie doľava, Otočenie doprava, Zmazanie, Duplikovanie, Export, Manuál

## Implementácia



Implementácia pop-up panelov, každý panel obsahuje 1-2 tlačidlá - na zrušenie a niektoré aj na pokračovanie:

- Nastavenia - 1 dropdown na nastavenie slovenského a anglického jazyka a informatívny štítok
- O programe - 1 textové pole s krátkym textom a informačné štítky s verziou programu
- Ukážky zapojení - panel tlačidiel na načítanie ukážok



Implementácia tlačidiel toolbaru, ktoré sú interaktívne:

- Nový projekt - otvorí okno Nový projekt
- Otvoriť, Uložiť - otvoria windows explorer
- Undo - krok späť
- Redo - krok vpred
- Spustiť - spustí simuláciu a zafarbí sa (odfarbí Pause button)
- Pozastaviť - zastaví simuláciu a zafarbí sa (odfarbí Play button)
- Priblíženie - priblíži grafickú plochu a pripočíta percentá do textového poľa
- Oddialenie - oddiali grafickú plochu a odráta percentá z textového poľa
- Otočenie doľava - označenú súčiastku otočí o 90 stupňov v protismere hodín
- Otočenie doprava - zmaže označenú súčiastku o 90 stupňov v smere hodín
- Zmazanie - zmaže označené súčiastky
- Duplikovanie - zduplikuje označené súčiastky
- Export - exportne obvod do HTML podoby
- Manuál - otvorí HTML manuál

Funkcionalita bola doimplementovaná do existujúcich skriptov z grafickej plochy a kamery, a namapovaná na tlačidlá.

Tlačidlá Otočenie doľava, Otočenie doprava, Duplikovanie a Zmazanie sa aktivujú pri označení súčiastky, a deaktivujú pri jej odznačení.



Implementácia tlačidiel hlavného menu, ktoré budú interaktívne:

- Tlačidlá v menu Zobrazenie skrývajú alebo ukazujú bočné panely. Skladajú sa z textu a checkboxu, ktorý je zaškrtnutý a tlačidlo je stlačené, ak je daný panel zobrazený.
- Tlačidlá Spustiť a Pozastaviť v menu Upraviť kopírujú funkcionality tlačidiel v toolbare - značenie je však rovnaké ako v menu Zobrazenie.

**File:** *Assets/Scripts/Menu buttons/MainMenuButtons.cs*

Predpoklady:

- Skript je v `_MainMenuManager` ako komponent.
- Objekt `_MainMenuManager` je priradený do `OnClick()` v tlačidlách, ktoré používajú metódy skriptu.

Funkcie:

- *PlayPauseButton(string action)* - riadi a označuje tlačidlá Play/Pause
- *Show"X"(GameObject guiComponent)* kde  $X = \{\text{Toolbox, Properties, ObjectExplorer, Debug}\}$  - zobrazia alebo skryjú bočný panel
- *OpenProject()* - otvorí windows explorer
- *SaveProject()* - otvorí windows explorer
- *SaveAsProject()* - otvorí windows explorer
- *Show"X"PanelMenu(GameObject guiComponent)* kde  $X = \{\text{File, Edit, View, Help}\}$  - zobrazia alebo skryjú dropdowny hlavného menu
- *Show"X"Canvas(GameObject guiComponent)* kde  $X = \{\text{NewProject, Settings, AboutProject, ReleaseNotes, ContactInfo, ReportBug}\}$  - zobrazia alebo skryjú pop-upy tlačidiel hlavného menu

### ***Testovanie***

Na overenie funkcionality bolo použité jednoduché testovanie, pri ktorom boli preklikané všetky tlačidlá hlavného menu a tak overená ich funkčnosť, vrátane tlačidiel Spustiť a Pozastaviť.

Pre overenie tlačidiel toolbaru bol vytvorený jednoduchý dvojsúčiastkový obvod, ktorom bola otestovaná funkcionality tlačidiel Otočenie doľava, Otočenie doprava, Zmazanie.

Funkcionality tlačidiel Priblíženie a Oddialenie bola rovnako otestovaná pri predošlom teste.

## **4.2.4 Kontextové menu [Ferrari]**

### ***Analýza***

Kontextové menu predstavuje panel, ktorý sa zobrazí po kliknutí pravým tlačidlom myši.

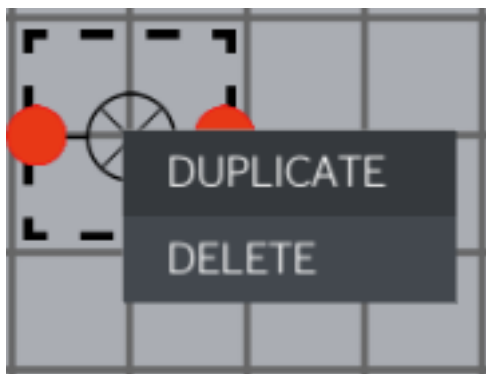
Pozostáva z tlačidiel pre manipuláciu so súčiastkami.

### ***Návrh***

Menu bude pozostávať z panelu, ktorý sa bude skrývať a zobrazovať na rovnakom princípe ako select/deselect. Pri kliknutí pravým tlačidlom na súčiastku sa aj sama selektne.

Funkcie kontextového menu:

- delete
- duplicate



## ***Implementácia***

Implementácia sa skladá z jedného skriptu, ktorý je pridaný všetkým súčiastkam a pozadiu. Jediná metóda `OnPointerClick()` zachytáva klikanie.

Na začiatku sa menu schová, a zablokuje sa všetky tlačidlá. Potom sa zisťuje, či bolo stlačené pravé tlačidlo a ak áno, kontextové menu sa zobrazí a presunie tesne pod kurzor myši (súradnice sa prepočítavajú funkciou `Camera.main.ScreenToWorldPoint(eventData.position)`). Nakoniec sa v sérii podmienok hľadajú skripty umiestnené na súčiastkach. Každý skript odblokuje svoje príslušné tlačidlo.

**File:** *Assets/GenericScripts/GenerateMenu.cs*

Predpoklady:

- skript je v prefaboch súčiastok a pozadiu ako komponent.

Funkcie:

- *OnPointerClick(PointerEventData arg0)* - deteguje klikanie myškou a pri kliknutí pravým tlačidlom upravuje kontextové menu

## ***Testovanie***

Funkcionalita bola otestovaná jednoduchým scenárom.

Testovací scenár:

1. Výber súčiastok z toolboxu
2. Kliknutie na súčiastku pravým tlačidlom
3. Kliknutie na tlačidlo Delete z kontextového menu



## 4.2.5 Lokalizácia - resource files [Bentley]

### *Analýza*

Unity engine ako framework natívne nepodporuje žiadnu formu lokalizácie textov, preto je potrebné vytvoriť si vlastný spôsob lokalizácie textov v aplikácií.

### *Návrh*

Základom lokalizácie je súbor alebo množina súborov, ktoré sú nositeľmi lokalizovaných textov. Pre účely nášho projektu sme zvolili XML súbor, ktorého elementami sú konkrétne podporované jazyky. Tento XML súbor sa používa ako zoznam kľúčov a hodnôt pre potrebné elementy používateľského rozhrania.

### *Implementácia*

Lokalizácia je implementovaná pomocou skriptu *Localization.cs* a triedy *ResourceReader.cs*. Lokalizované texty sa nachádzajú v súbore *Resources.xml*.

### *Localization*

Pre správnu funkcionálnosť lokalizácie je potrebné priradiť skript *Localization* spoločnému rodičovskému elementu všetkých textov, ktoré je potrebné lokalizovať. Pre zmenu používaného jazyka je potrebné volať metódu *ChangeLanguage* s argumentom jazyka, ktorý sa má použiť. Na definovanie jazykov sa používa *SystemLanguage Enumeration*, ktorý je definovaný v Unity. Pri prvom spustení sa nastavuje jazyk na jazyk systému.

### *ResourceReader*

Slúži ako pomocná trieda pre parsovanie XML súboru, ktorý obsahuje lokalizovaný text. V prípade, že sa požaduje čítanie jazyka, ktorý nie je obsiahnutý v XML súbore, použije sa default jazyk - anglický.

### *Resources*

Koreňovým elementom XML súboru je *Languages*, ktorý obsahuje zoznam podporovaných jazykov prostredníctvom svojich detí. Elementy konkrétnych jazykov (napr. *Slovak*) obsahujú zoznam lokalizovaných textových reťazcov, kde kľúčom je atribút *name* a obsah elementu je nositeľom lokalizovaného textu. Ako kľúč pre mapovanie textov na Unity elementy sa používa názov unity elementov.

## 4.3 Moduly používateľskej interakcie

Dôležitou súčasťou každého softvérového nástroja je pracovná plocha. Je to grafická časť nástroja, kde sa predpripravené komponenty medzi sebou spájajú do logického celku, nastavujú sa ich parametre a koná sa logika daného nástroja. Dôležité sú preto aj ovládacie prvky, ktoré sa v danom nástroji budú používať. Keďže ide o editovací nástroj, kde predpripravené komponenty spájame a nastavujeme, rozhodli sme sa, že jadro ovládacieho systému bude princíp drag and drop (ťahaj a uvoľni). Okrem toho, dôležité je vybrať si

konkrétny aktívny komponent, s ktorým chceme v danom okamihu niečo urobiť. Tomuto hovoríme selektovanie (označenie). Opakom je deselektovanie, kde selektovaný komponent uvoľníme. Ďalšou dôležitou funkciou je tvorba nových a mazanie existujúcich inštancií - create a delete. Potom sú tu funkcie na pohyb a rotáciu komponentov, ako aj rôzne ovládacie prvky pre kameru (zoom, pohyb). Na úzadie pracovnej plochy sme sa rozhodli dať mriežku (grid).

Všetky tieto komponenty sú bližšie opísané v nasledujúcich podčastiach.

### 4.3.1 File browsing [Honda]

Pre ukladanie a načítavanie projektov je potrebné implementovať file browser, ktorý preskúmava disk a umožňuje zvoliť miesto na disku, odkiaľ resp. kam sa projekt uloží.

#### *Analýza*

Pre kontinuitu používateľského zážitku a tiež na základe požiadaviek zákazníka je potrebné implementovať natívny File browser, na ktorý je používateľ zvyknutý. Podpora na túto implementáciu na strane unity neexistuje, čo podčiarkuje aj fakt, že sa predáva Asset presne s touto funkcionalitou za \$40 <https://www.assetstore.unity3d.com/en/#!/content/68064>.

#### *Návrh*

Našťastie spomínaný asset odokrýva dostatok svojej implementácie ako svojou dokumentáciou, tak aj štruktúrou súborov, ktoré obsahuje. Tento asset rieši problém tým, že má samostatný exe súbor, ktorý v jeho mene spúšťa natívny windows file browser cez .NET knižnice ([https://msdn.microsoft.com/en-us/library/system.windows.forms.openfiledialog\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.windows.forms.openfiledialog(v=vs.110).aspx)). Tento prístup napodobníme aj my v našej aplikácii a vytvoríme samostatný spustiteľný súbor, ktorý bude iba otvárať spomínaný natívny Windows file browser a vybratú cestu/súbor bude vypisovať na štandardný výstup, ktorý presmerujeme do našej aplikácie a budeme ho čítať. Ďalší problém vzniká so spúšťaním kódu na hlavnom vlákne, nakoľko Unity vo vedľajších vláknach neumožňuje volať funkcionalitu skriptov. Na tento účel použijeme voľne dostupnú Unity utilitu Dispatcher (<https://github.com/nickgravelyn/UnityToolbag/tree/master/Dispatcher>)

#### *Implementácia*

##### **Projekt:** EdusimFileBrowser

Na implementáciu bol vytvorený nový projekt, ktorý zastrešuje funkcionalitu natívneho file browseru, pričom bola čo najväčšia snaha o zachovanie znovupoužiteľnosti .exe súboru, ktorý je výsledkom kompilácie tohoto projektu. Táto znovupoužiteľnosť bola dosiahnutá parametrizáciou prostredníctvom konzolových prepínačov (options). Spustiteľný súbor z tohoto projektu bol zahrnutý do nášho Unity projektu ako StreamingAsset, takže sa načítava počas behu programu vtedy, keď je potrebný.

**File:** *FileBrowserHandler.cs*

Zastrešuje logiku spúšťania externých procesov pomocou štandardných Unity prostriedkov, pričom tiež spracováva výstup externého procesu a púšťa zodpovedajúci kód.

### 4.3.2 Create, Save, Load [Eagle, Ferrari, GAZ, Honda]

Modul zabezpečuje logiku vytvárania, ukladania a načítavania simulačných projektov. Projekt je zhluk súčiastok danej domény, ich vlastností a prepojení. Počas šprintu *Eagle* prebehla analýza celého modulu a boli porovnávané rôzne prístupy k perzistovaniu dát v grafickom engine Unity. Počas šprintu *Ferrari* bol vytvorený návrh, ktorého súčasťou bol prototyp ukladania dát. Počas šprintu *GAZ* bol vykonaný potrebný refaktoring simulačných objektov a bola implementovaná samotná logika vytvárania, ukladania a načítavania simulačných projektov.

#### *Create [GAZ]*

##### ANALÝZA

Vytvorenie novej simulácie je momentálne triviálna záležitosť. Je iba potrebné vymazať všetky simulačné elementy z aktuálnej simulácie. Okrem toho vyčistí premennú v ktorej je uložená aktuálna cesta k projektu (používaná pri save)

##### IMPLEMENTÁCIA

**File:** *Persistance.cs*

```
PUBLIC VOID CLEARSCENE()
```

Preiteruje všetky aktívne komponenty v scéne a zmaže ich. To isté spraví s čiarami - grafickými reprezentáciami logických spojení v simulácií.

#### *Save [GAZ]*

Implementácia dovoľuje ukladať jeden súbor na cestu, ktorú si používateľ zvolí.

##### ANALÝZA

Pre ukladanie je potrebné zozbierať informácie o všetkých objektoch, ktoré sa v aktívnej simulácii nachádzajú aj o ich vzájomných prepojeniach. Tieto informácie treba následne serializovať a uložiť na disk.

##### NÁVRH

Na serializáciu objektov a prácu s file systémom je potrebné použiť zodpovedajúce .NET knižnice, ktoré potrebnú funkcionálnu už poskytujú. Tieto knižnice sú v Unity jednoducho použiteľné a ich platformovo nezávislé fungovanie zabezpečuje framework Mono. Súbor, ktorý vznikne takouto serializáciou je binárny a tým pádom nie je jednoducho čitateľný a upravovateľný.

Iterovanie cez všetky aktívne simulačné elementy je jednoduché. Väčší problém však nastáva v tom, ako necyklicky zachytiť a zistiť ich prepojenie. Použitý prístup využíva čiary, čiže

grafické reprezentácie logických spojení v simulácií. Tie obsahujú začiatkový aj koncový bod daného spojenia (necyklické referencie, ktoré potrebujeme)

## IMPLEMENTÁCIA

**File:** *Persistence.cs*

Taktiež obsahuje serializovateľnú internú triedu *SerializationPackage*, ktorá reprezentuje finálny objekt, ktorý sa serializuje a zapisuje na disk (a následne aj načítava). Tento objekt je v podstate iba data objekt, ktorý obsahuje zoznam komponentov a ich prepojení.

Samotná serializácia je vykonaná triedou *BinaryFormatter*, ktorá je súčasťou .NET frameworku.

*PUBLIC VOID SAVE(STRING FILENAME)*

Parameter *fileName* je používateľom zvolená cesta na uloženie súboru. Ak príde argument prázdny, tak sa aplikácia pokúsi uložiť súbor na posledné miesto (save), ak príde neprázdny argument, tak aplikácia uloží súbor tak a miesto si zapamätá (Save as).

Metóda preiteruje všetky simulačné elementy a vypýta si ich entity. Tieto si uloží do listu. V ďalšom kroku to iste spraví s čiarami. Tieto dva listy uloží to *SerializationPackage*, ten serializuje a zapíše do súboru na disk.

## **Load [GAZ]**

### ANALÝZA

Načítanie je inverzná metóda k ukladaniu. Je potrebné zo súboru prečítať informácie o uloženom stave pracovnej plochy a tento stav opäť navodiť.

### NÁVRH

Na deserializáciu objektov a prácu s file systémom je potrebné použiť zodpovedajúce .NET knižnice, ktoré potrebnú funkcionálnu už poskytujú. Tieto knižnice sú v Unity jednoducho použiteľné a ich platformovo nezávislé fungovanie zabezpečuje framework Mono. Je potrebné zabezpečiť, aby bol načítavaný súbor validný.

Zo serializovaného zoznamu súčiastok a prepojení je potrebné vytvoriť reálne simulačné elementy a prepojenia.

## IMPLEMENTÁCIA

**File:** *Persistence.cs*

Obsahuje premenné, ktoré sú pomocou unity frameworku naplnené prefab objektami, ktoré reprezentujú jednotlivé komponenty.

## PUBLIC VOID LOAD(STRING FILENAME)

Parameter fileName hovorí o ceste k súboru, ktorý chceme načítať. Tento súbor načítame a pomocou triedy *BinaryFormatter* deserializujeme do triedy *SerializationPackage*. Táto trieda obsahuje listy entít simulačných elementov a čiar. Najskôr sa iteruje cez list entít a podľa typu entity sa inštancuje zodpovedajúci Unity simulačný element. Následne sa simulačnému elementu nastaví načítaná entita, čím sa simulačný element dostane do stavu, aký mal simulačný element, ktorý je uložený (pozícia, rotácia, vlastnosti...). Následne sa iteruje zoznamom serializovaných čiar a na základe toho sa prepájajú jednotlivé konektory simulačných elementov.

Tu prichádza na scénu koncept dočasných ID. S Elementami sa totiž serializujú aj Unity ID ich konektorov, a s čiarami zase Unity ID konektorov, ktoré prepájajú. Pri inštancovaní simulačných elementov sa konektorom nastaví toto ID do property *TemporaryId*. Na základe tejto property sa potom hľadajú konkrétne konektory, ktoré majú byť spolu prepojené čiarami a spájajú sa.

## Refactoring [GAZ]

Refaktoring bolo nutné vykonať hlavne z dôvodu serializácie objektov v jazyku C#.

## ANALÝZA

Serializovať sa dajú výhradne objekty, ktoré zoskupujú niekoľko primitívnych dátových typov, nie žiadne komplexné triedy obsahujúce ďalšie komplexné triedy. Hlavný dôvod, prečo je refaktoring potrebný, je ten, že triedy dediace od *MonoBehavior* sa serializovať nedajú, pričom všetky naše triedy, ktoré reprezentujú simulačné elementy (presne tie, ktoré potrebujeme serializovať) dedia priamo od tejto nadtriedy. Ďalším dôvodom pre nutnosť refaktoringu je zviazanosť našich Unity simulačných objektov so simulačnými objektami použitými v knižnici, ktorú používame na logickú simuláciu elektronických obvodov. Tieto objekty logickej simulácie taktiež nie je možné serializovať.

## NÁVRH

Pre správnu funkcionálnosť celého modulu je potrebné v prvom rade rozviazať Unity simulačné objekty s logickými simulačnými objektami, a následne zaviesť Entity triedy, ktoré budú predstavovať nosiče všetkých primitívnych dát, ktoré potrebujeme na serializáciu a následnú rekonštrukciu grafickej simulácie. Tieto objekty, inštancie Entity triedy musia byť serializovateľné. Pre každý typ simulačného elementu (napr. batéria, rezistor...) musí existovať práva jedna entity trieda, ktorá bude nosiť špecifické informácie pre daný typ elementu. Rovnako musí existovať práve jedna trieda, ktorá reprezentuje generický simulačný prvok a je nosičom informácií ako pozícia prvku a rotácia. Od tejto triedy budú dediť všetky konkrétne reprezentácie simulačných prvkov. Poslednou informáciou potrebnou na perzistovanie simulácie sú spojenia v simulácii.

## IMPLEMENTÁCIA

**File:** *GUICircuitComponent.cs*

Vytvorené metódy na vyplňanie Entity objektov pozíciou a rotáciou, vytvorené metódy na nastavovanie atribútov grafických simulačných objektov z entity objektu. Vylepšené dedenie metód od tejto triedy (abstract vs. virtual).

**Package:** *Entities*

Pridané entity triedy pre každý typ simulačného elementu aj s базovou triedou, od ktorej dedia ďalšie triedy.

**File:** *GUIBattery.cs, GUIResistor.cs ..*

Odstránené závislosti na logických simulačných objektoch, odstránenie redundantných závislostí na logické simulačné konektory, vylepšené inšancovanie reprezentovaného logického simulačného komponentu pri vytváraní simulácie, getter a setter pre Entity objekty, správna inicializácia grafickej reprezentácie konektorov.

### 4.3.3 Pohyb [Audi, Bentley, GAZ]

#### *Analýza*

Pohyb elementov sa v príbuzných softvérových nástrojoch stotožňuje s princípom drag and drop. Na tieto účely Unity ponúka vhodné funkcie [OnMouseDown](#) a trio funkcií [OnBeginDrag](#), [OnDrag](#) a [OnEndDrag](#). Alternatívne sa pohyb elementov vykonáva vo vopred definovaných veľkostiach posunu po plochy. Unity natívne podporuje klávesové vstupy.

#### *Návrh*

Pohyb bude implementovaný v dvoch tvaroch. Pohyb pomocou myšky sa uskutoční princípom drag and drop. Implementovaný bude funkciami [OnBeginDrag](#), [OnDrag](#) a [OnEndDrag](#), ktoré nám umožňujú veľmi dobre rozdeliť logiku pohybu do troch častí: začiatok, priebeh a koniec. Druhý tvar posunu bude vo vopred definovaných veľkostiach, ktoré budú presne také ako je veľkosť mriežky. Aktívny element sa bude zdanlivo pohybovať po mriežky.

#### *Implementácia*

**File:** *Assets/Scripts/GenericScripts/Draggable.cs, Assets/Scripts/GenericScripts/DragWithKeys.cs*

Predpoklady:

- aktívny komponent má pridaný skript *Dragable.cs* ako komponent,
- Container súčiastok má pridané oba skripty ako komponenty,
- aktívny komponent má 2D Box Collider,
- kamera má Physics 2D Raycaster,
- v scénke sa nachádza EventSystem.

## DRAG AND DROP

Jadro logiky tvoria tri natívne Unity funkcie, ktoré odchyťávajú eventy:

- `OnBeginDrag`
- `OnDrag`
- `OnEndDrag`

*PUBLIC VOID ONBEGINDRAG(POINTEREVENTDATA EVENTDATA)*

Funkcia zachytáva event začiatku ťahania myškou. V tejto funkcii sa koná inicializácia pohybu GameObjectu:

- deselektuje sa predtým selektovaná súčiastka a selektuje sa súčiastka, ktorú premiestňujeme (len ak premiestňujeme len jednu súčiastku)
- začiatková pozícia myšky pri stlačení na GameObject sa uloží do premennej `_mousePos`
- začiatková poloha GameObjectu sa uloží do premennej `_itemPos`
- ak je súčiastka vyberaná z toolboxu, vytvorí sa nová inštancia súčiastky a nastaví sa jej základné parametre
- sprístupnia sa GUI tlačidlá pre rotáciu a zmazanie súčiastky

*PUBLIC VOID ONDRAG(POINTEREVENTDATA EVENTDATA)*

Táto funkcia vykonáva samotné ťahanie predmetu po pracovnej plochy:

- počíta sa `mouseDiff` - rozdiel medzi začiatkovou pozíciou myšky a aktuálnou;
- na základe `mouseDiff` sa vypočíta aktuálna pozícia GameObjectu.
- ak sa presúva viac súčiastok naraz, pozície všetkých súčiastok sa prepočítavajú

*PUBLIC VOID ONENDDRAG(POINTEREVENTDATA EVENTDATA)*

Funkcia ukončuje pohyb:

- posledná pozícia GameObjectu z funkcie `OnDrag` sa zaokrúhľuje na najbližšiu 0.5 hodnotu:
  - prenásobí sa dvojkou;
  - zaokrúhli sa;
  - vydolí sa dvojkou;
- pozícia sa uloží ako finálna pre aktuálny pohyb GameObjectu
- zistia sa kolízie všetkých súčiastok volaním funkcie `Collision()` a v prípade vzniku kolízie sa vypočíta nové umiestnenie súčiastky

## POHYB POMOCOU KLÁVESNICI

Predpoklady:

- selektovaný aktívny komponent/komponenty.

Logika pohybu klávesnicou je implementovaná vo funkcii `Update()` v skripte `DragWithKeys.cs`, kde sa pozoruje stlačenie klávesov `W`, `A`, `S`, `D`. Veľkosť posunu je predurčená na 0.5f s oneskorením 0.25 sekúnd (definované v atribúte `_delay`). V každom frame sa volá funkcia `_decreaseDelay`, ktorá zníži tento delay za delta čas - čas potrebný na ukončenie posledného framu.

Ak je označených viac súčiastok naraz, funkcia prechádza celým Listom selektovaných súčiastok a pre každú vykoná ten istý pohyb. Na predídenie kolízií sa využívajú premenné uchovávané v skripte *ColisionUtils.cs*. Tento skript obsahuje statické premenné *Move* a *LastPressed*, ktoré zabezpečujú, že ak sa jedna súčiastka dotýka inej, pohyb žiadnej zo selektovaných súčiastok sa nevykoná, pretože by nastala kolízia.

#### 4.3.4 Kolízie [Bentley, GAZ]

##### *Analýza*

Pri umiestňovaní elementov na pracovnú plochu chceme zabrániť umiestneniu jedného aktívneho komponentu na druhý. Unity podporuje rôzne Box Collideri, ktoré riešia kolízie, ale potrebujú k tomu simulácie gravitačného poľa. V našej aplikácii sa takéto simulácie nedejú a drag and drop fyzika v Unity chápe ako teleportovanie, čo narušuje jej simulácie.

##### *Návrh*

Aby sme sa vyhli simulovaniu fyzických javov, ktoré sa bežne používajú v Unity hrách, rozhodli sme sa jednoducho prepočítať koncové koordináty aktívneho elementu pri konci jeho pohybu. Ak nastane kolízia dvoch aktívnych elementov, ten ktorý sa posledný pohyboval bude posunutý.

##### *Implementácia*

**File:** *Assets/Scripts/GenericScripts/Draggable.cs*

Predpoklady:

- aktívny komponent má pridaný skript ako komponent,
- aktívny komponent má 2D Box Collider,
- kamera má Physics 2D Raycaster,
- v scénke sa nachádza EventSystem,
- aktívne komponenty sú otagované ako *ActiveItem*.

Po skončení pohybu sa spúšťa funkcia *Colision()*. V nej sa prepočítajú koordináty a veľkosti všetkých aktívnych elementov. Ak nastane kolízia, posledný posúvajúci predmet bude posunutý napravo dole od kolíznej súčiastky na najbližšie voľné miesto. Kolízia sa overuje inkrementálne, čo znamená, že sa aj po posunutí znovu overí či neprišlo ku kolízii s ďalším aktívnym elementom.

Kolízia sa zisťuje pri každej zmene pozície súčiastky (duplicate, rotation, drag).

#### 4.3.5 Vytvorenie inštancií komponentu [Audi]

##### *Analýza*

Toolbox zvyčajne obsahuje preddefinované objekty, ktoré sa budú používať. Pomocou drag and drop princípu sa tieto objekty presunú na plochu čím sa vytvoria nové inštančia týchto objektov. Unity podporuje funkciu [Instantiate](#), ktorá vytvorí dvojicu pôvodného objektu.



## *Návrh*

Preddefinované objekty budú uložené v toolboxe. Pomocou drag and drop funkcie ich budeme prenášať na plochu, čím vytvoríme nové inštancie týchto objektov.

## *Implementácia*

**File:** *Assets/Scripts/GenericScripts/Draggable.cs*

Predpoklady:

- komponent má pridaný skript ako komponent,
- komponent má 2D Box Collider,
- kamera má Physics 2D Raycaster,
- v scénke sa nachádza EventSystem,
- komponent je otagovaný ako *ToolboxItemActive*.

V kroku začiatku drag and drop, vo funkcii *OnBeginDrag()* sa vytvorí nová inštancia objektu, ktorá sa ďalej ťahá na plochu.

### **4.3.6 Select [Audi, GAZ]**

## *Analýza*

Select slúži na vyznačenie komponentu, nad ktorým sa vykonajú nejaké funkcie. V príbuzných softvérových nástrojoch sa takáto funkcia rieši stlačením ľavým tlačidlom myšky. Unity natívne podporuje niekoľko rôznych eventov, ktorými sa odchyť kliknutie určitým tlačidlom myšky. Dva najpoužívanejšie riešenia sú funkcie [OnMouseDown](#) a [OnPointerClick](#).

## *Návrh*

Selektovanie sa bude konať stlačením ľavým tlačidlom myšky na aktívny komponent pracovnej plochy. Na tento účel sa použije funkcia [OnPointerClick](#). V nej vieme určiť, ktoré tlačidlo myšky pozorujeme a je odolnejšia na chyby (napr. pri [OnMouseDown](#) stlačením na UI sa event spustí aj na aktívny komponent v úzadí). Na zdôraznenie selekcie vykreslíme k selektovanému aktívnemu komponentu štvorec z trhaných čiar.

## *Implementácia*

**File:** *Assets/Scripts/GenericScripts/SelectObject.cs*

Predpoklady:

- aktívny komponent má pripojený skript ako Unity komponent,
- aktívny komponent má 2D Box Collider komponent,
- aktívny komponent má nastavený odkaz na SelectionBox prefab v skripte,
- kamera má Physics 2D Raycaster,
- v scénke sa nachádza EventSystem.

Funkcia `OnPointerClick()` poslúcha na stlačenie myškou. Po stlačení myškou sa v prípade potreby najprv vykoná deselekcia a až potom selekcia. V jednom okamihu môže byť selektovaný iba jeden aktívny komponent. Po selektovaní sa aktívnemu komponentu zapne *SpriteRenderer* komponent, ktorý obsahuje obrázok selection boxu. Taktiež sa sprístupnia grafické tlačidlá v GUI pre vymazanie a rotáciu súčiastky.

Selektovaný komponent je prístupný volaním *SelectObject.SelectedObjects[0]* atribútu, ktorý je public a static, z hociktorého iného skriptu.

### 4.3.7 Deselect [Bentley, GAZ]

#### *Analýza*

Deselektovanie sa v príbuzných softvérových nástrojoch koná stlačením myškou mimo aktívnych komponentov na pracovnej ploche.

#### *Návrh*

V úzadí pracovnej plochy bude background (jednofarebný obrázok). Keď na neho stlačíme myškou, znamená to, že sa pred ním nič nenachádza a môžeme deselektovať selektovaný obrázok.

#### *Implementácia*

**File:** *Assets/Scripts/GenericScripts/MultiSelect.cs*

Predpoklady:

- v úzadí existuje background - obrázok,
- background má 2D Box Collider komponent,
- skript je pripojený k backgroundu ako komponent.

V skripte *MultiSelect.cs* sa nachádza public funkcia *DoDeselect()*, ktorá zavolá deselect funkcie pre súčiastky aj pre čiary. Tie sú implementované v skriptoch *Line.cs* a *SelectObject.cs*. Pri deselecte sa List selektovaných objektov *SelectObject.SelectedObjects* premaže. Taktiež sa vypnú tlačidlá v GUI pre prácu so súčiastkou.

### 4.3.8 Multiselect [Ferrari, GAZ]

#### *Analýza*

V bežných interaktívnych prostrediach sa koná multiselect pomocou vykresľovania obdĺžnika ponad súčiastky, ktoré chceme selektovať. Taktiež sa používa aj alternatíva so stlačením klávesy ctrl a vyklikaním súčiastok pomocou myši.

#### *Návrh*

Pri stlačení myši sa zavolá event, ktorý spustí funkciu [OnPointerDown](#). Táto funkcia sa vykoná pri stlačení a držaní ľavého tlačidla myši. Pri pohybe myši sa bude z daného miesta

vykresľovať obdĺžnik. Ak tlačidlo myši pustíme, zavolá sa event [OnPointerUp](#), ktorý vykresľovanie štvorca zruší a súčiastky vo štvorci selektuje a pridá do Listu.

### ***Implementácia***

**File:** *Assets/Scripts/GenericScripts/MultiSelect.cs*

Predpoklady:

- na backgrounde je namapovaný skript *MultiSelect.cs* ako komponent.
- implementovaný pomocný skript *Assets/Scripts/Utils/MultiSelectUtils.cs*

Funkcia [OnGUI](#) sleduje eventy na GUI prostredia. Po kliknutí myšou funkcia *OnPointerDown()* spraví deselect všetkých selektovaných súčiastok. Následne sa nastaví premenná *\_isSelecting* na hodnotu *true* a uloží sa aktuálna pozícia myšky. Začne sa vykresľovanie obdĺžnika pomocou volania funkcií v skripte *MultiSelectUtils.cs*. Po pustení tlačidla myšky sa vykresľovanie skončí a pomocou funkcie *IsWithinSelectionBounds* sa pre každá aktívny element zistí, či sa nachádza vo vykreslenom obdĺžniku. Ak áno, objekt sa pridá do listu selektovaných súčiastok, zobrazí sa okolo neho selektor a sprístupnia sa GUI tlačidlá pre manipuláciu so súčiastkami.

## **4.3.9 Mazanie inštancii komponentu [Bentley]**

### ***Analýza***

Na mazanie elementov sa zvykne používať tlačidlo delete alebo pop-up okno v ktorom sa zvolí mazanie. V Unity existuje funkcia [Destroy](#), ktorá zmaže danú inštanciu GameObjectu.

### ***Návrh***

Pomocou klávesu Del sa zmaže selektovaný aktívny komponent. Ak je komponent pripojený čiarou na iný komponent, musia sa vymazať aj tieto čiary a aktualizovať zoznamy pripojených konektorov v súčiastkach, ktoré boli s vymazávanou súčiastkou spojené.

### ***Implementácia***

**File:** *Assets/Scripts/GenericScripts/Destroy.cs*

Predpoklady:

- selektovaný aktívny komponent.

Po stlačení klávesu *Del* sa najskôr overí, či selektovaný komponent nie je čiarou pripojený k iným komponentom. Funkcia získa listy čiarou pripojených konektorov k danej súčiastke. Pre každý nájdený konektor sa následne vymaže zo zoznamu pripojených konektorov konektor zmažavanej súčiastky. Ďalej sa nájdu všetky čiary, ktoré vychádzali z konektorov vymazavanej súčiastky a zmažú sa. Nakoniec sa vymaže súčiastka.

### 4.3.10 Rotácia [Audi, GAZ]

#### *Analýza*

Rotácia sa väčšinou koná pomocou myšky stlačením na roh elementu, ktorý chceme otočiť. Alternatívne riešenie je využitie klávesnice. Rotovať môžeme zvyčajne iba jeden element a musí byť predtým selektovaný. Rotácia viac elementov naraz rotuje všetky elementy okolo stredového bodu medzi rotovanými súčiastkami.

#### *Návrh*

Rotácia sa bude klávesnicovými skratkami *Q* (doľava) a *E* (doprava) nad vopred selektovaným aktívnym elementom. Ak bude selektovaných viac elementov naraz, rotácia sa vykoná podľa bodu stredu všetkých selektovaných súčiastok.

#### *Implementácia*

**File:** *Assets/Scripts/GenericScripts/Rotate.cs*, *Assets/Scripts/GenericScripts/RotateMultiObject.cs*

Predpoklady:

- selektovaný aktívny komponent.
- aktívny komponent má pridaný skript *Rotate.cs* ako komponent.
- Container súčiastok má pridaný skript *RotateMultiObject.cs* ako komponent.
- na GUI tlačidlách rotácie s namapované funkcie skriptov.

Vo funkcii Update sa pozoruje stlačenie klávesov *Q* a *E* na klávesnici. *Q* rotuje selektovaný komponent doľava. *E* rotuje selektovaný komponent doprava. Rotovať je možné aj GUI tlačidlami pre rotáciu.

V prípade rotácie viacerých objektov okolo stredového bodu sa najskôr vypočíta bod rotácie a následne sa každá súčiastka transformuje na inú rotovanú pozíciu.

### 4.3.11 Vykresľovanie čiar [Bentley, Cadillac]

#### *Analýza*

Elektrotechnické súčiastky sa v reálnom svete spájajú káblami. V EduSim simulácií tieto káble budú reprezentované vykresľovanými čiarami medzi konektormi súčiastok. Unity API poskytuje komponenty, ktoré dokážu medzi dvoma bodmi vykresliť čiaru rôznej farby či hrúbky. Jedným z takýchto komponentov je [LineRenderer](#). Podľa unity dokumentácie potrebujeme pridať prázdny *GameObject* a v ňom zahrnúť *LineRenderer* komponent. Čiara by sa mala začať vykresľovať po kliknutí na niektorý z konektorov a pri držaní stlačeného tlačidla na myši by koncový bod čiary mal mať súradnice aktuálnej pozície myši. Na detekciu kliknutia preto potrebujeme pridať konektorom [Collider](#) komponent. Čiara by sa mala vykresliť len v prípade, že sa úspešne spoja dva konektory. Taktiež po spojení dvoch súčiastok sa musí aktualizovať zoznam prepojení, aby sa dala správne implementovať logika elektrického obvodu.

## Návrh

Po pridaní Colliderov ku konektorom a vytvorení objektu zahrňujúceho komponent LineRenderer implementujeme dva skripty. Jeden skript s názvom *Line.cs* bude pridaný ku komponentu s LineRenderom a druhý skript s názvom *Connectable.cs* ku každému konektoru súčiastky. Connectable skript bude obsahovať public atribút, ktorý bude obsahovať list čiarou pripojených konektorov ku konkrétnemu konektoru. Collider konektoru deteguje kliknutie myši na daný konektor. Následne sa vytvorí klon objektu s LineRenderom (každá čiara budem mať vlastný LineRenderer). Connectable skript zaznamená pozíciu konektora a myši a tieto pozície pošle skriptu Line, ktorý medzi nimi vykreslí čiaru. Po skončení dragovania myšou sa overí, či sa myš nachádza na pozícii nejakého z konektorov. Je potrebné zabrániť spojeniu konektorov tej istej súčiastky alebo konektora samého zo sebou. Vykreslenej čiare je potrebné pridať dynamický Collider, ktorý mení veľkosť a pozíciu pri posúvaní pripojených súčiastok. Tento Collider bude slúžiť na select čiary a následnú možnosť vymazania čiary po stlačení klávesy delete. Implementovaná bude tiež možnosť zmeniť typ selektovanej čiary stlačením klávesy space. Čiara sa bude zalamovať dvomi spôsobmi - pravé zalomenie a ľavé zalomenie.

## Implementácia

### ŤAHANIE ČIAR

Connectable skript obsahuje funkcie:

- Start(),
- AddConnected (GameObject connected)

a funkcie rozhraní:

- IBeginDragHandler - OnBeginDrag (PointerEventData eventData) ,
- IDragHandler - OnDrag (PointerEventData eventData),
- IEndDragHandler - OnEndDrag (PointerEventData eventData).

### START()

Funkcia *Start()* inicializuje list objektov, do ktorého sa pridávajú konektory po spojení čiarou a volá sa vždy len pri spustení programu.

### ADDCONNECTED (GAMEOBJECT CONNECTED)

Funkcia *AddConnected (GameObject connected)* slúži na pridanie druhého konektora do listu (nie toho, ktorého skript sa vykonáva). Využíva unity metódu *SendMessage (message, object)*, ktorá zavolá v *objecte* metódu s názvom *message*.

### ONBEGINDRAG (POINTEREVENTDATA EVENTDATA)

Funkcia zachytáva event začiatku ťahania myšou - stlačenie ľavého klávesu na GameObject, ku ktorému je skript priradený. V tejto funkcii sa vytvorí inštancia *Line* objektu, ktorému sa nastaví pozícia konektora ako začiatkový a zároveň aj konečný bod pre vykresľovanie čiary.

*ONDRAG (POINTEREVENTDATA EVENTDATA)*

Táto funkcia je volaná od prvého stlačenia ľavého tlačidla myši až po pustenie tlačidla. V tejto funkcii sa aktualizuje konečná pozícia čiary podľa pozície myši.

*ONENDDRAG (POINTEREVENTDATA EVENTDATA).*

Táto funkcia sa volá v momente pustenia ľavého tlačidla myši. Funkcia najskôr porovná aktuálnu pozíciu myši so všetkými konektormi v scéne. Ak sa pozícia myši s niektorým konektorom zhoduje a sú splnené podmienky, že začiatkový aj konečný bod sa nachádzajú v scéne, spojené konektory nepatria jednej súčiastke a konektor nie je spájaný sám so sebou, aktualizujú sa zoznamy pripojených konektorov v konektoroch a vytvorí sa konečná inštancia čiary medzi konektormi. Ostatné dočasné inštancie (tie, ktoré nemajú koncový bod) sú zmazané.

Line skript obsahuje tieto funkcie:

- Update(),
- CheckSelect(),

a funkciu rozhrania:

- IPointerClickHandler - OnPointerClick (PointerEventData eventData),

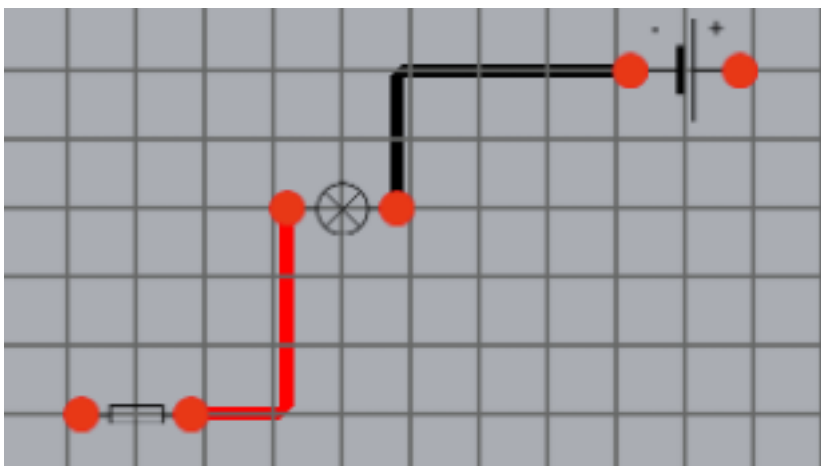
*UPDATE()*

Táto funkcia sa volá každý frame. Connectable skript jej posiela začiatkový a koncový bod - object, medzi ktorými má byť vykresľovaná čiara. Vo funkcii sa z týchto objektov získajú súradnice. Funkcia zavolá komponent *LineRenderer*, do ktorého pridá dané súradnice.

Funkcia overuje, či boli spojené dva konektory. Obsahuje privátnu premennú *TypeOfLine*, ktorá nadobúda hodnoty:

- *RightBreak*, čiže pravé zalomenie.
- *LeftBreak*, čiže ľavé zalomenie

Pre lepšie predstavenie, čím sa myslí pravé a ľavé zalomenie - ukážka na obrázku nižšie.



V obrázku na ľavej strane je ľavé zalomenie a vpravo je zas pravé zalomenie. Tieto režimy sa prepínajú stláčaním kláves *space*. Obrázok tiež znázorňuje ako vyzerá selektnutá čiara - červená.

Ďalej sa vo funkcii zavolá funkcia `AddCollidersToLine()`, ktorá čiare pridá komponent *BoxCollider2D*. Funkcia tiež overuje, či sa so súčiastkou nehýbalo alebo či sa nezmenil štýl zalomenia. V tom prípade treba Collider aktualizovať, pretože čiara mení svoju polohu na ploche. Aktualizácia Collidera prebieha tak, že starý Collider sa odstráni a vytvorí nový.

Na konci sa ešte zavolá funkcia `CheckSelect()`, ktorá zisťuje, či bolo na čiaru kliknuté myšou a podľa potreby ju selectne alebo deselectne.

*ONPOINTERCLICK (POINTEREVENTDATA EVENTDATA)*

Táto funkcia deteguje kliknutie na čiaru pomocou eventu *eventData*. Ak sa v momente kliknutia pozícia myši nachádza na *BoxCollider2D*, ktorý patrí danej čiare, čiaru zafarbí na červeno a do globálnej premennej *SelectedLine* uloží objekt selectnutej čiary. Funkcia tiež overuje, či predtým nebola selectnutá iná čiara. Ak áno, predtým selectnutú čiaru najprv vyfarbí naspäť na čierne. Selektnutú čiaru je možné klávesom *delete* vymazať, alebo stláčaním klávesy *space* meniť zalomenie čiary.

*ADDCOLLIDERTOLINE()*

Táto funkcia vytvára *BoxCollider2D* pre každú čiaru. Collider je pridaný ako *child* do konkrétnej čiary (do objektu *Line*). Konkrétne sa vytvárajú dva Collider. Využíva sa pozícia bodu zalomenia. Zistia sa dĺžky čiary od oboch konektorov k bodu zalomenia a funkcia nastaví šírku a dĺžku Collidera podľa toho, či je čiara zvislá alebo vodorovná. Funkcia X-ovú dĺžku collidera nastaví na celú dĺžku čiary a y-ovú dĺžku nastaví na veľkosť 0.3, pretože čiara je hrubá len 0.1 a na tak úzku čiaru by sa ťažko klikalo. Nakoniec sa matematicky zistí uhol rotácie collidera, podľa začiatočného a konečného bodu čiary.

*SwitchTypeLine.cs* skript slúži na spomínané identifikovanie typu zalomenia čiary. Skript v *Update()* funkcii deteguje každých 25 stotín sekundy stlačenie klávesy *space*. Po stlačení klávesy nastavuje premennú *TypeOfLine* v skripte *Line.cs*.

#### SELECT A DESELECT ČIAR

Select čiar je vykonávaný vďaka *BoxCollideru*, ktorý je dynamicky pridaný každej čiare. Select je implementovaný vo funkcii *OnPointerClick (PointerEventData eventData)* v skripte *Line.cs*, ktorá je popísaná vyššie v dokumentácii. Ak sa myšou klikne na plochu mimo, čiara sa deselectuje - vyfarbí sa na čierne a globálna premenná *SelectedLine* sa nastaví na *null*. Deselect je implementovaný v skripte *Deselect.cs*. Tento skript je pridaný každému objektu *Line*.

Deselect čiar sa vykonáva v skripte *Deselects.cs*. Tento skript je pridaný každému objektu *Line*. Skript obsahuje už spomínanú funkciu *OnPointerClick*, ktorá sleduje event kliknutia myšou na scénu. Po kliknutí na scénu mimo collidera čiary sa najprv overí, či je nejaká z čiar selectnutá, čiže globálna premenná *SelectedLine* nie je *null*. Ak premenná obsahuje objekt čiary, danej čiare nastaví farbu na čiernu a premennú *SelectedLine* nastaví na hodnotu *null*.

#### MAZANIE ČIAR

Mazanie čiar je implementované v skripte *Destroy.cs*. Tento skript je pridaný každému objektu *Line*. Ak je nejaká čiara selectnutá (bolo na ňu kliknuté myšou a má červenú farbu), v globálnej premennej *SelectedLine* je uložený objekt tejto čiary. Destroy skript overuje, či premenná *SelectedLine* obsahuje nejakú čiaru. Ak áno, zisťuje, či bola stlačená klávesa *delete*. Po stlačení tejto klávesy sa zo zoznamov pripojených konektorov dané konektory vymažú a nakoniec sa vymaže aj konkrétny objekt čiary.

### 4.3.12 Object Duplication [Ferrarri]

#### *Analýza*

V editoroch sa bežne používa funkcionálna kopírovanie objektov a ich multiplikácia pomocou nalepenia, známe ako Copy/Paste. Bežné skratky na toto bývajú Ctrl + C a Ctrl + V. V EduSim editore treba dávať pozor na umiestnenie elementov, aby neprišlo k ich prekryvaniu. Táto skutočnosť je už ohendlovaná pri zmene pozícií komponentov, no treba mať na mysli a kopírovanie viacerých objektov.

#### *Návrh*

Kopírovanie sa bude konať tak, že nové objekty sa objavia čím bližšie k originálu. Používateľ teda nebude mať kontrolu nad ich umiestnením hneď pri ich kopírovaní. Kopírovaný objekt sa umiestni čo najbližšie originálu smerom vpravo a nadol (v smere pravouhlého trojuholníka) s odsadením jedného štvorčeka.

#### *Implementácia*

**File:** *Assets/Scripts/Duplicate.cs*

Predpoklady:



- selektovaný aktívny komponent

Po stlačení tlačidla na kopírovanie sa selektované aktívne komponenty skopírujú na najbližšiu voľnú pozíciu v smere vpravo a nadol s odsadením jedného štvorčka.

### 4.3.13 Undo Redo [GAZ, Honda, Infiniti]

Funcionalita Undo/Redo spočíva v navrátení zmien a navrátení navrátenia zmien pre akúkoľvek akciu so súčiastkami. Tieto akcie sú:

- Presúvanie súčiastky, alebo skupiny označených súčiastok
- Zmena hodnôt atribútov súčiastky
- Rotácia súčiastky, alebo skupiny označených súčiastok
- Vytvorenie súčiastky
- Vymazanie súčiastky

#### *Analýza*

Funcionalita Undo/Redo je pre Unity implementovaná ako pomocný objekt knižnice Unity Editor, avšak táto nejde využiť vo vybudovanom riešení. Je to preto lebo bola mienená iba ako pomocný prostriedok pri vytváraní aplikácie a nie ako seriózna funkcionalita použiteľná v aplikácií. Z daného dôvodu sa navrhol celý systém Undo/Redo pre našu aplikáciu.

#### *Návrh*

Princíp navrátenia akcie spočíva v uložení zoznamu akcií, ktoré sa vykonali v podobe zmien súčiastok a v uložení zoznamu reverzných akcií pre navrátenia. Jedna akcia však sa môže týkať viac súčiastok pri viacnásobnom označení súčiastok a tým pádom môže obsahovať jedna akcia zoznam zmien, z ktorých každá zmena predstavuje jednu zmenu jednej súčiastky.

#### *Implementácia*

Celá implementácia základnej logiky každej Undo/Redo akcie je implementovaná v triede *UndoList*, v ktorej sa nachádzajú undo a redo listy akcií. Trieda *DoUndo* je namapovaná na tlačidlá v scénke, ktoré volajú *PerformUndo* a *PerformRedo* funkcie, ktoré volajú *Undo* a *Redo* funkcie triedy *Undolist*. *Undo* funkcia triedy *Undolist* vykoná reverznú akciu pre najnovšie pridanú akciu do undo listu, vymaže ju z listu a pridá ju do redo listu, ako akciu, ktorá bola navrátená a môže byť znovu navrátená do stavu nenavrátenej. V triede *Undolist* sa takisto nachádza funkcia *addUndo*, ktorá pridá undo akciu do undolistu a pokiaľ tento dosiahol svoj maximálny limit, tak vymaže najstaršie pridanú akciu z listu. Táto funkcia takisto vyčistí redo list z dôvodu toho, že ak sa pridá nová akcia a pritom je v redo liste niekoľko akcií, tak po stlačení redo by sa tieto akcie už nemali vykonať. *Undo* akcia obsahuje zoznam zmien, ktoré dedia od hlavného nadtypu zmien, triedy *Change*. Každá undo akcia obsahuje funkcie *UndoChanges* a *RedoChanges*, ktoré vykonajú všetky zmeny v liste zmien a to v prípade undo reverzné formy zmien a v prípade redo znovuaplikovanie zmien. Každá zmena má spoločné chovanie jej vykonania pri redo a jej reverznej akcie pri undo volané funkciami *UndoChange* a *RedoChange*.

## PRESÚVANIE A ROTÁCIA SÚČIASTKY, ALEBO SKUPINY OZNAČENÝCH SÚČIASTOK

Na zaznamenávanie presúvania, alebo rotácie súčiastok slúži typ zmeny na toto vytvorenie, zvaný PosChange, Trieda si ukladá informácie o zmene pozície, alebo rotácie a takisto id objektu, ktorého sa zmeny týkajú a aplikuje ich na tento objekt v prípade zavolania undoChange. V prípade zavolania redoChange sa všetky zmeny pozície, alebo rotácie invertujú a znova sa zavola undo funkcia undoChange, ale tentoraz nad týmito invertovanými zmenami.

## ZMENA HODNÔT ATRIBÚTOV SÚČIASTKY

Princíp triedy AttChange, ktorá je zodpovedná za uchovanie zmeny atribútov je rovnaký ako pri zmene pozície a teda takisto sa pri volaní undoChange aplikujú reverzné zmeny na atribúty objektu s daným id a pri volaní redoChange aplikujú invertované zmeny.

## VYTVORENIE A VYMAZANIE SÚČIASTKY

Trieda CreateDeleteCompChange obsahuje dva módy, v jednom je undo akcia vytvorenie súčiastky a redo akcia vymazanie súčiastky a v druhom naopak. Prvý sa používa pri zaznamenanie undo akcie pre vymazanie súčiastky a druhý pre vytvorenie. Samotné vytvorenie súčiastky pozostáva z dvoch sekvenčných krokov. Prvým je vytvorenie všetkých súčiastok, ktorých sa undo akcia vymazania týka a druhým je poprepájanie týchto novovytvorených súčiastok čiarami vrátane korektného priradenia konektorov konektorom súčiastok. Tie isté kroky sa tiež robia pri vytvorení súčiastok pre redo akciu vytvorenia.

## VYTVORENIE A VYMAZANIE ČIARY

Trieda CreateDeleteLineChange je založená na úplne rovnakom princípe, ako trieda CreateDeleteCompChange, pričom má tiež dva módy. Rozdiel je hlavne v tom, že v tejto triede sa nevytvárajú a nemažú súčiastky, ich konektory a čiary medzi nimi, ale len samotné čiary medzi dvoma súčiastkami. Na túto akciu je potrebné vedieť len id konektorov, medzi ktorými sa čiara nachádza.

## *Testovanie*

Tieto zmeny boli vytvárané a pridávané postupne na viacerých miestach v kóde:

- Zmena pozície súčiastky v Draggable triede
- Zmena rotácie v Rotate a RotateMultiObject triedach
- Vytvorenie súčiastky v Draggable triede
- Vymazanie súčiastky v Destroy triede
- Vymazanie čiary v Connectable triede
- Zmeny atribútov v triedach každej súčiastky

Zavedenie kódu do všetkých tried bolo otestované pri vývoji a takisto v rámci úlohy testovania na poslednom šprinte, pričom pripomienky a chyby boli opravené.

### 4.3.14 CameraZoom [Audi]

#### *Analýza*

Približovanie a vzdďalovanie kamery sa v príbuzných softvérových nástrojoch implementuje kolieskom myšky. Niekedy je potrebné stlačiť kláves CTRL (hlavne, keď nástroj podporuje skrolovanie myškou). Unity podporuje zmenu pozície kamery ako hociktorého iného GameObjectu. Alternatívne, podporuje zmenu ortografickej veľkosti, ktorou sa ovplyvní zoom.

#### *Návrh*

Na zoomovanie budeme používať koliesko myšky. Skrolovaním nahor sa priblíži obsah pracovnej plochy a skrolovaním nadol sa vzdiali. Napevno nastavím maximálne a minimálne zoomovanie. Je to preto, lebo nepodporujeme zmenu veľkosti komponentov.

#### *Implementácia*

**File:** *Assets/Scripts/GenericScripts/CamZoom.cs*

Predpoklady:

- kamera je ortografická,
- na kameru sa pridá skript ako komponent.

Vo funkcii *Start()* sa inicializujú:

- rýchlosť približovania/vzdďalovania,
- maximálne priblíženie,
- maximálne vzdialenie.

Približovanie sa koná točením kolieska myšky nahor. Vzdďalovanie sa koná točením kolieska myšky nadol. Zoomovanie sa koná nastavením atribútu *orthographicSize* v kamere.

### 4.3.15 Pohyb kamery [Bentley]

#### *Analýza*

Pohyb po pracovnej ploche sa zvyčajne rieši slajdermi (scroll bars). Občas sa použije aj prístup, kde sa pohyb koná stlačením stredného tlačidla myšky a potom sa ťahá do želaného smeru pohybu. Unity natívne podporuje input z myšky a povoľuje nastavenie pozície kamery, čo nám dovoľuje nehýbať všetkým na ploche, ale iba kamerou.

#### *Návrh*

Pohyb bude implementovaný pomocou myšky. Pri stlačení stredného tlačidla a následného ťahania myškou na strany sa vykoná pohyb kamery, čím sa vytvorí ilúzia posunu celej plochy.

## ***Implementácia***

**File:** *Assets/Scripts/GenericScripts/CameraMovement.cs*

Predpoklady:

- existuje background v úzadí - obrázok,
- k backgroundu je pripojený skript,
- k skriptu je nastavená referencia na kameru,
- background obsahuje 2D Box Collider,
- kamera obsahuje Physics 2D Raycaster.

Pohyb kamery sa koná stlačením stredného tlačidla myšky - kolieska, v dvoch fázach:

- pri stlačení sa uchovávajú začiatkové pozície myšky a kamery;
- pri pohybe myškou sa tieto pozície menia.

### **4.3.16 Background [Bentley]**

#### ***Analýza***

Defaultne, Unity úzadie je modré. V Unity sa úzadie zvykne nastavovať ako obrázok. Alternatívou je prefarbiť úzadie v kamere (z modrého na niečo iné). Background zobrazený ako obrázok v úzadí nám však umožňuje zachytávať na ňom eventy.

#### ***Návrh***

Umiestnenie jednofarebného obrázka do úzadia s box colliderom. Toto sa využije na zachytávanie eventov, napr. pre Deselect.

## ***Implementácia***

**File:** *Assets/Prefabs/Background.cs*

Predpoklady:

- prefab je umiestnený v scénke.

### **4.3.17 Grid [Audi, Bentley]**

#### ***Analýza***

Na úzadí pracovných plôch príbuzných softvérových nástrojov zvykne byť mriežka. Táto napomáha používateľovi pri umiestnení elementov. Unity umožňuje viacero prístupov na implementovanie mriežky, ani jeden však nie natívne. Pri vyhľadávaní sme sa stretli s viacerými riešeniami a viaceré vyskúšali.

## *Návrh*

Mriežka je umiestnená ako textúra na štvorci. Táto textúra sa dynamicky vykreslí na základe parametrov.

## *Implementácia*

**File:** *Assets/Scripts/GenericScripts/Grid.cs*

Predpoklady:

- prefab grid je v scénke,
- v prefabe je skript.

V skripte je možné nastaviť veľkosť mriežky, počet riadkov a stĺpcov. Mriežka je umiestnená v pozadí, aby neprekryvala žiadne komponenty na pracovnej plochy. Odporúčané je použiť dvojnásobok počtu riadkov a stĺpcov než je veľkosť mriežky - vtedy každý štvorček bude veľkosti  $0.5f \times 0.5f$ .

## **4.4 Moduly simulačnej logike**

### **4.4.1 Súčiastky [Audi]**

#### *Analýza*

Na základe internetového prieskumu boli zistené základné informácie o fyzikálnych veličinách v obvode a súčiastkach v ňom.

#### *Návrh*

Na základe prieskumu webu bolo identifikovaných niekoľko elektrických súčiastok a ku každej bolo identifikovaných niekoľko základných vlastností:

- **Rezistor** – odpor, tolerancia odporu(odchýlka k odporu), zaťaženie odporu(dovolený výkon premeny v teplo), dovoľené napätie(najväčšie dovoľené napätie medzi vývodmi súčiastky), teplotný súčiniteľ(zmena odporu pri zmene teploty o 1 stupeň)
- **Kondenzátor** – kapacita(veľkosť elektrického náboja pri jednotkovom elektrickom napätí), maximálne povolené napätie
- **Cievka** – indukčnosť(množstvo magnetického toku vyvolaného elektrickým tokom), práca(vykonaná cievkou po odpojení prúdu kým dosiahne nulovú hodnotu)
- **Akumulátor(sekundárny článok)**
- **Batéria(primárny článok)**
- **Spínač**
- **Žiarovka**

Takisto boli na základe používateľských prípadov použitia identifikované niektoré generické komponenty, ktoré sa budú nachádzať v každom vzdelávacom module.

- **Stlačiteľný bod**
- **Vyskakovacie okno**

- Merač
- Bod merania
- Textové pole

### ***Implementácia***

Súčiastky boli vytvorené ako prefabs s tým, že im boli zo začiatku priradené obrázky z internetu a neskôr boli tieto vymenené za obrázky poslané ATOS-om. Každému prefab súčiastky bol priradený skript jej triedy, ktorá bola vytvorená s identifikovanými atribútmi z návrhu. Bola vymyslená schéma pripájania súčiastok a to použitím jednotnej triedy *Connector*, ktorá reprezentuje konektor súčiastky, alebo uzol, ktorý sa pripája k ďalším súčiastkam, alebo uzlom. Každý prefab súčiastky má takisto dva konektory, ktoré majú priradený skript triedy *Connector*. Prepojenie dvoch konektorov je reprezentácia drôtu a je uskutočnené pridaním jedného konektoru do zoznamu pripojených konektorov druhého konektoru, pričom zoznam konektorov je ako atribút v každej triede *Connector*, a takisto pridaním druhého konektoru do zoznamu pripojených konektorov prvého konektoru. Plusy tohto zapojenia sú absencia drôtov a možnosť využitia polymorfizmu nakoľko má všetko, čo je zapojené, nadtriedu *Connector*.

## **4.4.2 Back-End simulácie elektrického obvodu [Bentley]**

### ***Analýza***

Pre potreby výpočtov vlastností elektrického obvodu ktorý si používateľ vytvorí, je potrebné mať k dispozícii nástroj ktorý dokáže simulovať najrôznejšie zapojenia elektrického obvodu.

Nakoľko vytváranie takéhoto simulačného nástroja vyžaduje výbornú znalosť elektrotechniky a bolo by časovo náročné sme sa rozhodli hľadať riešenia ktoré by bolo možné integrovať do nášho projektu.

Integrácia akejkoľvek knižnice alebo programu do nášho projektu EduSim kladie nároky na vzájomnú komunikáciu aplikácií. Navyše Edusim projekt je postavený na Unity 5.4, ktoré zo sebou nesie vlastnú implementáciu C# framework, ktorý navyše implementuje len verziu .net frameworku 3.5.

Jediným riešením napísaným v C# ktoré sa nám podarilo nájsť bol projekt SharpCircuit prístupný na githube - <https://github.com/Mervill/SharpCircuit>. Licencia tohoto riešenia je MIT/Boost C++.

Pre integráciu v projekte EduSim, je potrebné poskytnúť použiteľnú knižnicu alebo akékoľvek API ktoré by bolo možné integrovať do projektu pre využitie v simulácii.

### ***Návrh***

Nakoľko v samotnom repozitári projektu je ukážka použitia projektu na simulácii s dvomi rezistormi a baterkou, je možné odčítať približné použitie projektu SharpCircuit.

## ***Implementácia***

Z projektu SharpCircuit sme vytvorili C# class-library – DLL knižnicu, ktorú sme preložili pre Unity 3.5 framework.

## ***Testovanie***

Testovanie sme robili v dvoch fázach

Testovanie funkčnosti SharpCircuitProjektu

Pri tomto testovaní sme zisťovali či výpočty ktoré vykonáva projekt, sú zhodné so skutočnými fyzikálnymi vlastnosťami elektrického obvodu.

Testovanie integrácie do Unity projektu

Týmto testovaním sme objavili prvé problémy ktoré nastali ak DLL bola vytvorená .net frameworkom 4.5. Zistili sme že takto preloženú DLL nie je možné použiť pre Unity projekt.

### **4.4.3 Logika súčiastok [Bentley, Cadillac]**

#### ***Analýza***

Projekt EduSim je vytváraný v prostredí Unity. Knižnica ClassLibrarySharpCircuit ponúka objekty ktoré dokážu vytvoriť simuláciu, avšak nie pre grafické prostredie unity, ale pre konzolový výpis. Nakoľko používateľ si potrebuje simuláciu vytvoriť v grafickom prostredí programu EduSim je potrebné nemapovať objekty z DLL knižnice na grafické objekty simulácie.

V zásade sú potrebné dva typy mapovania

- Mapovanie jednotlivých objektov a ich prepojení na reprezentáciu v simulácii z DLL
- Mapovanie spúšťania simulácie z EduSim programu na spúšťanie simulácie v DLL knižnici ktorá vypočíta simuláciu

Nakoľko sa nám nepodarila nájsť podpora elektrotechnických uzlov v DLL knižnici, nie je možné dávať do simulácie pre DLL uzly, ale je nutné uzly odstrániť a nahradiť ich spojeniami s elektrotechnickými súčiastkami.

Pri mapovaní súčiastok sme identifikovali potrebu zapájať súčiastky s rôznym počtom konektorov (napríklad elektrotechnický uzol má jeden konektor, rezistor má dva konektory, tranzistor má tri konektory).

Jednotlivé elektronické súčiastky môžu mať rôzne vlastnosti (akumulátor napätie, rezistor odpor atď.). Tieto atribúty súčiastok je potrebné mať možnosť pred spustením simulácie meniť.

Nakoľko dll knižnica obvodu neobsahuje triedu uzla a umožňuje len priame prepojenie dll-konektorov súčiastok, je potrebné vytvorené uzly v scénke prehľadávať a zistiť tak pre všetky dll-konektory, ku akým ostatným dll-konektorom sú pripojené. Toto bude realizovať algoritmus, ktorého výstup by mal byť tým pádom zoznam dvojíc, z ktorých prvý element bude dll-konektor a druhý element bude list pripojených dll-konektorov ku konektoru v prvom elemente. Na realizáciu programu, ktorý spracuje stav scény aj s uzlami a vráti požadovaný výstup identifikovaný vyššie, je preto potrebné vytvoriť algoritmus využívajúci grafové prehľadávanie súčiastok a uzlov v scénke za účelom nájdenia prepojení jednotlivých konektorov súčiastok aj za podmienky, že je v scénke v danom prepojení medzi súčiastkami v ceste viacero uzlov.

### Návrh

Jednotlivé objekty elektrotechnických súčiastok sú agregované v skript súčiastok zo simulácie. Každá súčiastka bude obsahovať pole konektorov ktoré budú obsahovať odkazy na objekty konektorov príslušnej elektronickej DLL súčiastky. Každá elektronickej súčiastka bude dediť od triedy Component, pričom trieda Component bude obsahovať konektory zo scény. Konektory zo scény budú pri inicializácii objektu mapované na konektory z DLL knižnice. Takto sa zabezpečí mapovanie DLL konektorov na konektory reprezentované v scéne.

Atribúty súčiastok je potrebné meniť pomocou get/ set metód.

Navrhovaný grafový algoritmus sa podobá deep-first-search. Tento algoritmus je použitý na nájdenie všetkých konektorov súčiastok, ktoré sú priamo a nepriamo(pomocou uzlov) pripojené ku danému konektoru označenej súčiastky. Konektor reprezentuje uzol, alebo konektor akejkoľvek súčiastky a označená súčiastka reprezentuje súčiastku, pre ktorej konektory sa aktuálne zisťujú priamo a nepriamo pripojené konektory. Pred spustením algoritmu musí byť zachovaných niekoľko inicializačných pravidiel:

1. Uchovávanie zoznamu nasledovníkov(ďalších konektorov) u každého konektora(súčiastkového, ale aj uzla)
2. Uchovávanie si zoznamu navštívených a teda už uzavretých konektorov
3. Uchovávanie si zoznamu práve prehľadávaných konektorov
4. Uchovávanie si zoznamu konektorov patriacich súčiastkam
5. Uchovanie si označenia súčiastky

Jednotlivé kroky algoritmu grafového prehľadávania sú nasledovné:

1. Pri navštívení konektora sa tento zaradí do zoznamu prehľadávaných konektorov a zistí sa, či patrí súčiastke. Ak patrí súčiastke a táto súčiastka nie je označená súčiastka, tak sa tento konektor pridá do zoznamu konektorov patriacich súčiastkam v prípade, že sa tam už nenachádza. Po jeho pridaní sa odstráni zo zoznamu prehľadávaných konektorov a uloží sa do zoznamu navštívených konektorov. Ak ale nepatrí súčiastke, znamená to, že je uzol a prejde sa na krok 2.
2. Navštívia sa všetky konektory daného konektora po jednom za podmienky, že sa nenachádza v zozname navštívených konektorov a ani v zozname práve prehľadávaných konektorov.



Algoritmus na zistenie všetkých pripojených konektorov pre všetky konektory všetkých súčiastok prejde všetky súčiastky a po jednej ich označí za aktuálne prechádzanú. Ďalej pre každú takúto súčiastku prejde všetky jej konektory a na každý zavolá grafový algoritmus.

## **Implementácia**

### PRIDÁVANIE SÚČIASTOK DO ELEKTRICKÉHO SIMULAČNÉHO OBVODU

V projekte je trieda *GUICircuit* ktorá agreguje obvod elektronickej simulácie z DLL knižnice. Obvod je reprezentovaný statickou premennou *sim*. Do tejto simulácie je možné pridávať elektronické objekty kedykoľvek zavolaním *GUICircuit.sim.Create<TypeOfElectronicComponent>()*. Z pravidla sa to deje pri pridávaní elektronickej súčiastky do scény kedy sa aj mení tag súčiastky z *ToolboxItemActive* na tag *ActiveItem*. Tag *ToolboxItemActive* predstavuje elektronické súčiastky ktoré nie sú v obvode uvažované, ale slúžia len na výber z toolboxu.

### GRAFOVÝ ALGORITMUS NA PRECHÁDZANIE UZLOV

Z pohľadu inicializačných pravidiel je zoznam nasledovníkov prítomný v triede *Connector* a je to jej atribút *connectedConnectors[]*. Trieda *Connector* obsahuje takisto dll-konektor súčiastky, ktorej patrí, ak patrí nejakej súčiastke. Zoznamy navštívených, prehľadávaných a konektorov patriacich súčiastkam sú implementované pomocou knižnice *System.Collections* a konkrétne jej kolekcií *Stack*.

Celý algoritmus je spustený zavolaním funkcie *untangle*, ktorá berie ako argument zoznam súčiastok scény a vracia list štruktúr *ConnectionsOfComponent*. Táto štruktúra obsahuje dll-konektor a zoznam dll-konektorov, ktoré grafový algoritmus nájde ako k nemu priamo, alebo nepriamo pripojené. Funkcia *untangle* prejde všetky súčiastky a postupne každú najskôr označí a na každý jej konektor zavolá funkciu *explore* grafového algoritmu. Táto funkcia naplní zoznam konektorov patriacich súčiastkam, ktoré rekurzívne nájde, pričom tieto konektory sú dll-konektory a sú to práve konektory priamo a nepriamo pripojené k danému prechádzanému konektoru označenej súčiastky. Po tejto funkcii sa celý zásobník vyprázdni do štruktúry *ConnectionsOfComponent* a konkrétne do jej druhého atribútu *connectedDllConnectors*. Následne sa všetky tri zásobníky vyprázdnia, aby bol možný aj ďalší beh funkcie *explore*.

Funkcia *explore* je rekurzívna, pričom začína testom konektora s null hodnotou a pokračuje vložením konektora do zoznamu otvorených konektorov. Ďalej testuje, či je to konektor súčiastky a ak hej a ešte nie je v zozname patriacich nejakej súčiastke, tak ho tam pridá. V inom prípade je tento konektor uzol a zavolá sa rekurzívne znova na všetky jeho konektory, pokiaľ nie sú v zozname prehľadávaných, alebo navštívených konektorov. Po týchto krokoch konektor vyhodí zo zoznamu prehľadávaných a vloží ho do zoznamu navštívených.

### ELEKTRONICKÉ SÚČIASTKY

Vždy ako sa pridá nová elektronická súčiastka do scény v ktorej sa vytvára elektrický obvod sa volá funkcia *start* danej triedy. Nakoľko objektu sa zmení tag z *ToolboxItemActive* na tag *ActiveItem*. Takto vieme v metóde *Start()* identifikovať že pridaná súčiastka bude súčiastka v elektrickom obvode a preto ju pridáme do elektrického obvodu.

Vlastnosti súčiasok je možné meniť pomocou get/ set atribútov tried súčiasok.

## ***Testovanie***

### GRAFOVÝ ALGORITMUS NA PRECHÁDZANIE UZLOV

Ako testovanie bolo vytvorených pár umelých scénok, súčiasok a ich prepojené konektory spolu s uzlami, ktoré boli predané tomuto algoritmu. Vykonávanie algoritmu bolo sledované pomocou *Debug.Log()* výstupov.

## **4.4.4 Simulácia el. obvodu v grafickom prostredí [Cadillac][Dodge]**

### ***Implementácia***

#### SPUSTENIE SIMULÁCIE

Každý objekt v scénke je gameobject a k nemu priradené komponenty. Jeden z komponentov je aj jeho skript na ktorého inštanciu sa dá objektu scény dopytovať funkciou `GetComponent<>()`. Práve táto funkcia slúži potom na vytiahnutie inštancií komponentov gameobjektov zo scény, ktoré sa dostanú pomocou funkcie `UnityEngine.Object.FindObjectsOfType(typeof(GameObject))`, ktorá vráti pole všetkých objektov v scénke. Tieto objekty je však treba ešte odfiltrovať len na súčiasky a preto sa ešte testuje, či majú priradený tag `ActiveItem`. Všetky takto získané súčiasky sa vložia do zásobníka `sceneItems`.

Po získaní súčiasok zo scény sa celý zásobník skopíruje do listu súčiasok, ktorý sa predá ako argument funkcii `untangle` grafového algoritmu, ktorý vráti zoznam štruktúr o veľkosti počtu súčiasok. Tieto štruktúry sa nazývajú spojenia pre komponent a obsahujú pre každú súčiasku zoznam spojení o veľkosti počtu konektorov tejto súčiasky. Zoznam spojení je takisto štruktúra obsahujúca jeden dll-konektor tohto konektora a zoznam všetkých dll-konektorov k nemu pripojených identifikovaný práve grafovým algoritmom.

Takýto zoznam štruktúr spojení pre komponent sa prechádza vo slučke, pričom sa vo vnorenej slučke prechádza zoznam spojení pre každý konektor tohto komponentu. Nakoniec sa v najvnorenejšej slučke prechádza zoznam dll-konektorov pripojených k danému konektoru a pomocou funkcie `sim.Connect` sa po jednej položke spojí celý tento zoznam s daným dll-konektorom tohto konektora.

Takto poprepávané súčiasky môžu byť teraz podrobené simulácii pomocou funkcie `sim.doTick`. Táto funkcia je opätovane spúšťaná vo funkcii `update`, ktorá sa volá každou snímkou vykresľovania obrazovky.

Simulácia je spustená po stlačení tlačidla "štart", kedy sa vytvorí nový objekt simulácie dll knižnice, načítajú sa všetky objekty zo scény a tento objekt sa každému z nich predá na inicializáciu ich dll častí. Nasledovne sa odstránia uzly a príde sa na priame prepojenia súčiasok grafovým algoritmom. Tieto prepojenia sa pospájajú pre daný objekt simulácie. Po

stlačení "stop" tlačidla simulácia prestane tick-ať, pričom pre jej spustenie je znova potrebné stlačiť "štart", kedy sa znova vykonajú hore uvedené kroky. Užívateľ teda môže meniť štruktúru obvodu hocikedy, ale len po stlačení "štart" tlačidla sa mu tieto zmeny preukážu aj v simulácii.

#### 4.4.5 Meracie zariadenia [Dodge]

Nakoľko priamo simulačná knižnica nemá podporu žiadnych meracích zariadení, bolo potrebné zariadenia vytvoriť vlastné.

##### *Voltmeter*

Zariadenie voltmeter je určené na meranie napätia na súčiastkach obvodu. Mernou jednotkou sú volty [V].

Na implementáciu zariadenia merajúceho napätie sa použil odpor.

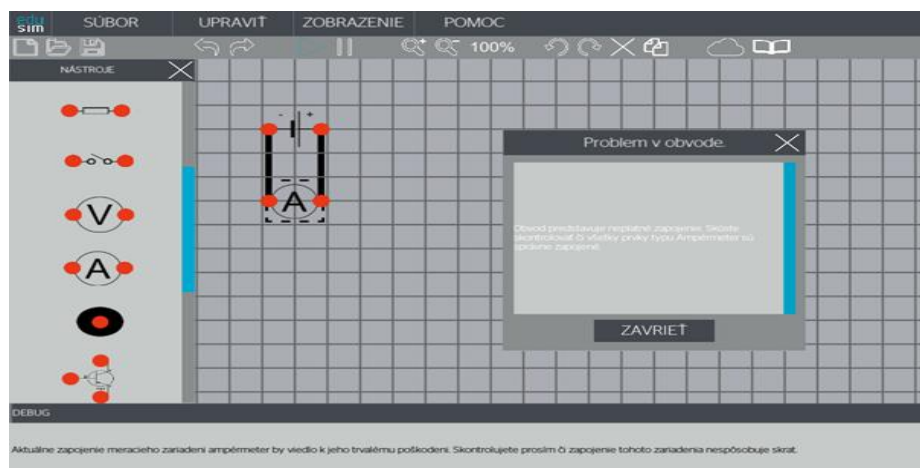
##### *Ampérmeter*

Zariadenie ampérmeter je určené na meranie prúdu medzi súčiastkami obvodu. Mernou jednotkou sú ampéry [A].

Na implementáciu zariadenia merajúceho prúd sa použil odpor.

#### 4.4.6 Zachytávanie zlého zapojenia obvodu [Honda]

Ak používateľ vytvorí nesprávne zapojenie obvodu, zobrazí sa príslušná chybová hláška, ktorá sa snaží podľa možnosti čo najbližšie analyzovať simuláciu a poskytnúť používateľovi pohľad na miesto spôsobujúce problém.



#### 4.4.7 Upozornenia na chybné zapojenia obvodu [Honda]

Cieľom tejto časti programu je poskytnutie informácie používateľovi o tom, že zapojenie nie je platné. Zároveň sa aj pokúšame priblížiť čo najpresnejšie čo je dôvodom neplatnosti zapojenia. V niektorých

prípadoch sa snažíme opraviť chyby v zapojení, pokiaľ vieme že zapojenie je správne ale simulačný framework nezohľadňuje všetky vlastností daného obvodu. Príkladom takéhoto zapojenia je sériové zapojenie dvoch a viacerých kondenzátorov, ktoré nepredstavujú neplatné zapojenie ale bez odporu medzi nimi simulačný engine padne.

#### 4.4.8 Export do HTML

##### *Analýza*

Úlohou exportovacieho modulu je vytvoriť reprezentáciu schémy vytvorenej v nástroji EduSim do prostredia webu a tento export zabaliť do ZIP archívu. HTML5 podporuje element *canvas*, do ktorého sa dajú vykresľovať prvky WebGL vrátane obrázkov a jednoduchých geometrických tvarov.

Tvorba ZIP archívov je podporovaná v .NET 4.5+. Keďže Unity používa .NET 0.2 treba nájsť alternatívnu cestu.

##### *Návrh*

Pri exporte sa vytvára export pre všetky prvky ktoré majú tag *ActiveItem* a *LineItem* ako aj určité špecifiká (napr. šípka pri potenciometri).

Používajú pozície prvkov ako sú zobrazené v Unity scénke. Takto máme zabezpečené že komponenty a ich konektory zachovajú svoje rozmiestnenie aj v exporte. Pozície čiar a konektorov sú vždy umiestnené s offsetom polovičného rádiusu - polomer kružnice okolo exportovaného komponentu.

Súradnice v Unity sú v inom kvadrante ako v HTML5 canvas komponente. Preto sa kamera vždy rotuje počas exportovania dva krát o 180°. Pričom po prvej rotácii sa vytvorí export a druhou rotáciu sa docieľi východzí stav.

Javascript kód vykresľuje všetky exportované komponenty do HTML5 elementu *canvas*. *Tento element nie je podporovaný v starších verziách HTML.*

Na kompresiu celého exportu do archívu ZIP použijeme externú knižnicu.

##### *Implementácia*

**File:** *Assets/Scripts/Simulation/ExportHTML.cs*

Export začína volaním funkcie *MakeHtmlExport*. Táto vyhladá všetky *GameObjecty* a exportuje ich parametre (koordináty a obrázky). Funkciou *\_getRelativePosition* sa počítajú pozície pre konektory a čiary.

HTML, CSS A FONTY

HTML dokument je statický. Nalinkovaný je na JavaScript, CSS a fonty. Jeho jadrom je element *<canvas>*, kde sa vykresľuje exportovaná schéma a časť s výpisom informácií. Umiestnený je v koreni priečinku s menom *export.html*.

CSS dokument je umiestnený v osobitnom priečinku *css/edusim.css*. Obsahuje všetky definície použitých štýlov. HTML neobsahuje žiadne inline CSS.

Export využíva osobitné fonty od Googlu - Open Sans. Tieto sú umiestnené v osobitnom priečinku *fonts*.

## JAVASCRIPT

Template (vzor) na exportovanú formu JavaScriptu je umiestnený v *js/edusim-pattern.js*. Tento súbor sa načíta v EduSime a do premennej *hotspots* sa pridajú koordináty komponentov. Výsledný Javascript sa premenuje na *js/edusim.js*, ktorý už aj *export.html* rozpozná.

Veľkosť canvasu sa vždy nastaví na veľkosť exportovanej schémy. Toto sa udeje hneď pri spustení JavaScriptu (teda hneď pri načítaní stránky). Je dôležité rozlíšiť nasledovné tri veľkosti:

- Veľkosť exportovanej schémy - rozdiel medzi koordinátami komponentov umiestnených najviac doľava a doprava.
- Veľkosť elementu canvas - rozlíšenie canvasu, resp. koľko bodov sa na canvase môže vykresliť. Nastavuje sa priamo v atribútoch canvas elementu alebo JavaScriptom.
- Zobrazovacia veľkosť canvasu - nastaviteľné pomocou CSS. Predstavuje v akej veľkosti sa canvas zobrazí na obrazovke. Neovplyvňuje rozlíšenie canvasu.

Interakcia myšky a komponentov je implementovaná vo funkcii *handleMouseMove*.

## FILE EXPLORER

Explorácia súborového systému je odvodená z implementácie pre Save a Load pričom je upravená o prácu so ZIP súbormi.

*Pre viacej informácii si pozrite časť o Save a Load.*

## ZIP ARCHÍV

Pri exportovaní sa obsah súboru EduSimExport archivuje do formátu ZIP. Na tento účel je použitá osobitná knižnica DotNetZip, keďže v čase vývoja Unity používal .Net 2.0, v ktorej nie je trieda ZipFile (.NET 4.5+).

Práca s knižnicou je jednoduchá. Vytvorí sa objekt archívu a pridávajú sa do neho súbory do zvolených priečinkov a podpriečinkov.

## 5 Inštalačná príručka a manuál

Vítajte na stránkach manuálu nástroja EduSim! V zozname **vľavo** si môžete zvoliť oblasť, ktorá Vás zaujíma.

Manuál je sprevádzaný obrázkami ukážok použitia nástroja.

### 5.1 Inštalácia

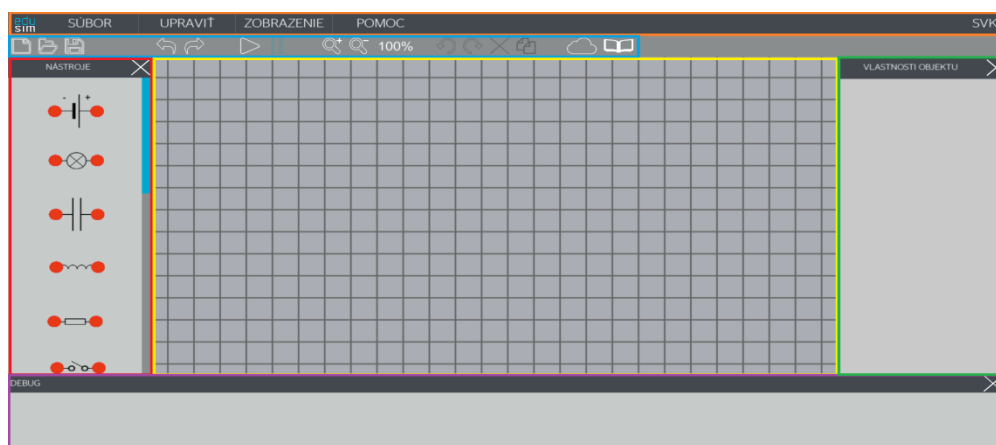
EduSim nainštalujete spustiteľným súborom *EduSim\_setup.exe*, ktorý nainštaluje program použitím inštalačného systému Windows - klasické inštalačné okno s voľbou cieľového inštalačného priečinku a vytvorenia ikony na desktope na žiadosť.

Nainštalovaný softvér obsahuje nasledovné položky:

- EduSim\_Data - dáta, ktoré sú potrebné pre správnu funkcionálnosť softvéru
- unins000.exe - odinštalovanie
- EduSim.exe - spúšťač

### 5.2 Grafické rozhranie

Grafické rozhranie simulátora vyzerá nasledovne:



Farbami sú ohraničené rôzne časti simulátora. Jednotlivé časti sú nasledovné:

- Oranžová: Hlavné menu
- Modrá: Panel s hlavnými nástrojmi
- Červená: Panel so súčiastkami
- Žltá: Pracovná plocha
- Zelená: Vlastnosti súčiastky
- Ružová: Výpis chýb a hlások

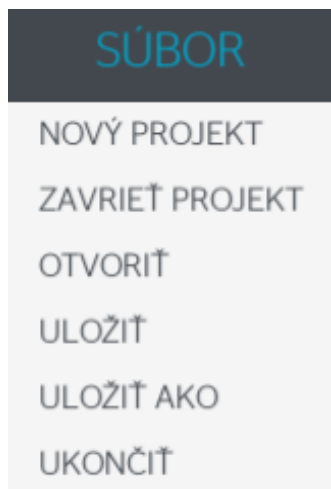
### 5.3 Hlavné menu

Hlavné menu obsahuje položky:

1. Súbor
2. Upraviť

3. Zobrazenie
4. Pomoc

### 5.3.1 Súbor



Položka súbor v hlavnom menu obsahuje nasledovné položky:

- Nový projekt - na vytvorenie nového projektu
- Zavrieť projekt - na vymazanie pracovnej plochy
- Otvoriť - na načítanie projektu
- Uložiť - na uloženie projektu
- Uložiť ako - na uloženie projektu s iným menom a/alebo koncovkou
- Ukončiť - na uzavretie simulátora

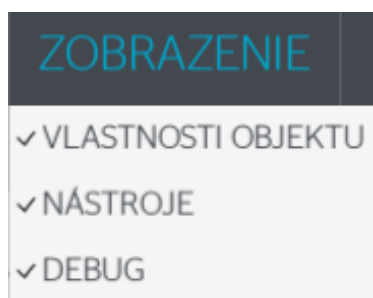
### 5.3.2 Upraviť



Položka upraviť v hlavnom menu obsahuje nasledovné položky:

- Duplikovať - na duplikovanie komponentu pracovnej plochy
- Zmazať - na mazanie komponentu pracovnej plochy
- Spustiť - na spustenie simulácie
- Pozastaviť - na pozastavenie simulácie
- Nastavenia - na nastavenie simulátora

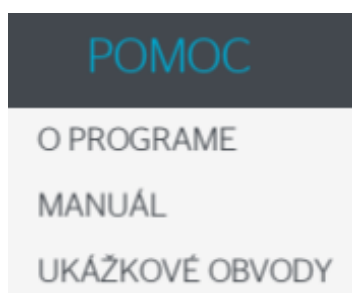
### 5.3.3 Zobrazenie



Položka zobrazenie v hlavnom menu obsahuje nasledovné položky na zobrazenie/schovanie okien nástroja:

- Vlastnosti objektu - okno na nastavenie vlastností komponentov na pracovnej ploche, napr. vlastnosti súčiastok elektrického obvodu
- Nástroje - okno s komponentami, napr. súčiastky elektrického obvodu
- Debug - okno s kontrolnými a chybovými hláškami simulátora

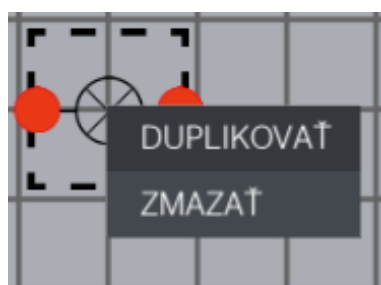
### 5.3.4 Pomoc



Položka pomoc v hlavnom menu obsahuje nasledovné položky:

- O programe - výpis informácií o nástroji a zobrazí aktuálnu verziu
- Manuál - zobrazí manuál na použitie nástroja. Manuál je možné otvoriť aj príslušným tlačidlom na toolbare.
- Ukážkové obvody - zobrazí panel s tlačidlami pre načítanie pred-pripravených obvodov

## 5.4 Kontextové menu



Kontextové menu sa zobrazí po kliknutí pravým tlačidlom myšky na súčiastku alebo pracovnú plochu. Obsahuje nasledovné položky:

- Duplikovať - na duplikovanie komponentu pracovnej plochy
- Zmazať - na mazanie komponentu pracovnej plochy



## 5.5 Používateľská príručka (manuál)

### 5.5.1 Panel s hlavnými nástrojmi

V tomto paneli je niekoľko ikoniek majúcich rozličné funkcionality:

- Nový projekt - Vytvoriť novú schému
- Otvoriť - Otvoriť existujúcu schému
- Uložiť - Uložiť aktuálnu schému
- Undo - Vykonalie akcie navrátenia
- Redo - Vykonalie akcie navrátenia
- Spustiť - Spustenie simulácie
- Pozastaviť - Pozastavenie simulácie
- Priblížiť - Priblíženie plochy (zoom in)
- Oddialiť - Oddialenie plochy (zoom out)
- Otočiť doprava - Otočenie súčiastky, alebo skupiny súčiastok v smere hodinových ručičiek
- Otočiť doľava - Otočenie súčiastky, alebo skupiny súčiastok proti smeru hodinových ručičiek
- Zmazať - Zmazanie súčiastky, alebo skupiny súčiastok
- Duplikovať - Duplikovanie súčiastky, alebo skupiny súčiastok
- Export - Vygenerovanie html exportu schémy
- Manuál - Prechod na stránku manuálu vysvetľujúcu funkcionality programu

### 5.5.2 Ovládacie prvky pracovnej plochy

#### *Umiestnenie súčiastok elektrického obvodu na pracovnú plochu*

Súčiastku elektrického obvodu umiestnite na pracovnú plochu nasledovným spôsobom:

1. V paneli nástrojov stlačte ľavým tlačidlom myšky na súčiastku, ktorú chcete umiestniť na pracovnú plochu.
2. So stlačeným ľavým tlačidlom myšky posuňte kurzor myšky v smere pracovnej plochy. Zvolená súčiastka bude sledovať váš kurzor.
3. Keď budete spokojný s pozíciou vašej súčiastky, pustite ľavé tlačidlo myšky. Súčiastka sa umiestni so zarovnaním podľa najbližších štvorčekov. Týmto sa vytvorí nová súčiastka, ktorá bude simulovať samostatnú súčiastku elektrického obvodu zvoleného typu.

#### *Kolízia súčiastok*

Keď sa pokúsite umiestniť súčiastku tak, že sa dve súčiastky budú navzájom prekrývať, súčiastka, ktorú sa práve pokúšate umiestniť na kolíziu pozíciu, sa posunie vedľa súčiastky, ktorá tam už je. Posun záleží od toho, ktoré časti súčiastok sa prekrývajú.

#### *Selekcia komponentov*

Komponent na pracovnej plochy môžete selektovať, čím sa vám sprístupnia ovládacie prvky pre daný komponent. Ovládacie prvky v hlavnom paneli nástrojov sa sprístupnia tak, že sa zmení ich farba.

Selektovať komponent môžete nasledovne:

1. Stlačte ľavým tlačidlom myšky na komponent, ktorý chcete selektnúť. Selektnutý komponent bude vyznačený v závislosti od komponentu:
  - Súčiastky budú mať štvorec vykreslený trhanou čiarou okolo nich.
  - Čiary zmenia farbu.
2. Stlačte a potiahnite ľavé tlačidlo myši cez komponent, ktorý chcete označiť.

### ***Selekcia viacerých komponentov***

Označiť viacero komponentov naraz je možné kliknutím ľavého tlačidla myšky a potiahnutie vytvoreného štvorca nad označované komponenty.

Súčiastky je následne možné ovládať ako jeden komponent, vrátane funkcie mazania, otáčania a posúvania po pracovnej ploche.

Súčiastky odznačíte kliknutím na pracovnú plochu.

### ***Mazanie súčiastok***

Keď chcete zmazať súčiastku z pracovnej plochy, postupujte nasledovne:

1. Stlačte ľavým tlačidlom myšky na súčiastku, ktorú chcete vymazať z pracovnej plochy.
2. Vykonať jednu zo 4 možností:
  1. Stlačte klávesu Del na klávesnici
  2. Stlačte pravým tlačidlom myšky na súčiastku a vyberte možnosť “Zmazať”
  3. V hlavnom menu Upraviť vyberte možnosť “Zmazať”
  4. Stlačte ľavým tlačidlom myšky na tlačidlo na mazanie súčiastok v hlavnom paneli nástrojov.

Vymazať je možné aj viac súčiastok naraz pomocou označenia viacerých komponentov pomocou potiahnutia ľavého tlačidla myši ponad komponenty.

Pri vymazaní súčiastky sa vymažú aj všetky čiary, ktoré boli napojené na súčiastku.

### ***Prepojenie súčiastok elektrického obvodu káblami***

V simulačnom nástroji sú káble predstavené čiarami. Prepojiť môžete iba súčiastky, ktoré sú umiestnené na pracovnej ploche. Na prepojenie dvoch súčiastok (A a B) postupujte nasledovne:

1. Stlačte ľavým tlačidlom myšky na jeden z konektorov súčiastky A.
2. So stlačeným ľavým tlačidlom myšky posuňte kurzor myšky ku konektoru súčiastky B, ku ktorému chcete súčiastku A pripojiť. Týmto súčiastky A a B prepojíte a medzi nimi bude môcť tiecť simulovaný elektrický prúd.

### ***Zalomenie čiar***

Vykreslenú čiaru môžete zalomiť v tvare písmenka L.

Na zalomenie čiary, postupujte nasledovne:

1. Stlačte ľavým tlačidlom myšky na čiaru, ktorú chcete zalomiť.
2. Stlačte tlačidlo Space na klávesnici.

### ***Presun súčiastok na pracovnej ploche***

Po uložení súčiastky na pracovnú plochu, môžete zmeniť jej pozíciu.

Na zmenu pozície súčiastky na pracovnej ploche, postupujte nasledovne:

1. Stlačte ľavým tlačidlom myšky na súčiastku, ktorú chcete posunúť na pracovnej ploche.
2. So stlačeným ľavým tlačidlom myšky posuňte kurzor myšky na pozíciu kam chcete súčiastku umiestniť. Zvolená súčiastka bude sledovať váš kurzor.
3. Keď budete spokojný s pozíciou vašej súčiastky, pustíte ľavé tlačidlo myšky. Súčiastka sa umiestni so zarovnaním podľa najbližších štvorčekov.

### ***Pohyb viacerých komponentov naraz***

Ak chcete pohnúť s viacerými komponentmi naraz, označte ich potiahnutím ľavého tlačidla myši ponad komponenty a využite jednu z nasledujúcich možností:

1. Stlačte ľavé tlačidlo myši ponad hociktorým z označených komponentov a potiahnite na miesto, kam chcete. Všetky označené súčiastky sa budú pohybovať rovnakým smerom.
2. Použite klávesové skratky W+S+A+D na pohyb všetkých označených súčiastok.

### ***Rotácia komponentov***

Komponenty na pracovnej ploche môžete rotovať dvomi spôsobmi:

1. Pomocou klávesov na klávesnici
2. Pomocou tlačidiel v hlavnom paneli nástrojov

Rotovať je možné aj viac komponentov naraz podľa stredového bodu. Aplikácia sama rozozná, či treba rotovať len súčiastku alebo viac súčiastok.

### ***Rotácia pomocou klávesových skratiek***



Rotovať komponent pomocou klávesových skratiek môžete nasledovne:

1. Označte komponent, ktorý chcete rotovať
2. Stlačte kláves na klávesnici:
  1. Q pre rotovanie doľava
  2. E pre rotovanie doprava

### ***Rotácia pomocou tlačidiel hlavného panelu nástrojov***

Rotovať komponent pomocou tlačidiel hlavného panelu nástrojov môžete nasledovne:

1. Označte komponent alebo viac komponentov, ktoré chcete rotovať
2. Stlačte tlačidlo v hlavnom paneli nástrojov:

1.  pre rotovanie doľava
2.  pre rotovanie doprava

### ***Duplikovanie súčiastok***

Keď chcete duplikovať súčiastku na pracovnej ploche, postupujte nasledovne:



1. Stlačte ľavým tlačidlom myšky na súčiastku, ktorú chcete duplikovať na pracovnej ploche.
2. Vykonajte jednu z 2 možností:
  1. Stlačte pravým tlačidlom myšky na súčiastku a vyberte možnosť “Duplikovať”
  2. V hlavnom menu Upraviť vyberte možnosť “Duplikovať”

### ***Undo/redo zmeny vykonanej na ploche***

Typy zmien, ktoré môžu byť navrátené sú:

1. Vytvorenie súčiastky, alebo skupiny súčiastok
2. Vymazanie súčiastky, alebo skupiny súčiastok
3. Presunutie súčiastky, alebo skupiny súčiastok
4. Rotácia súčiastky, alebo skupiny súčiastok
5. Vymazanie čiary

Pre vykonanie undo/redo akcií stlačte nasledovné tlačidlá v hlavnom paneli:

1. Stlačte na  pre redo akciu
2. Stlačte na  pre undo akciu

### ***Posun pracovnej plochy***



Na posunutie pracovnej plochy, postupujte nasledovne:

1. Stlačte stredným tlačidlom myšky na ľubovoľné miesto na pracovnej ploche.
2. So stlačeným stredným tlačidlom myšky posuňte kurzor v smere kam sa chcete posunúť. Posun funguje akoby ste hýbali kamerou, ktorú máte nad pracovnou plochou.
3. Keď budete spokojný s posunom pracovnej plochy, pustite stredný kláves myšky.

### ***Približovanie a vzdďalovanie pracovnej plochy***

Priblížiť a vzdialiť komponenty pracovnej plochy môžete dvomi spôsobmi:

1. Pomocou myšky - otočte stredné tlačidlo (krúžok) myšky:
  1. v smere nahor - priblíženie
  2. v smere nadol - vzdialenie
2. Pomocou tlačidiel v hlavnom paneli nástrojov:

1. Stlačte na  pre priblíženie
2. Stlačte na  pre vzdialenie.

## ***Používateľské skratky***

Existujúce klávesové skratky:

- Zmazanie: delete
- Otočenie doľava: Q
- Otočenie doprava: E
- Posunúť doľava: A
- Posunúť doprava: D
- Posunúť dole: S
- Posunúť hore: W
- Zalomenie čiary: space
- Manuál: F1

## **5.5.3 Simulácia elektrických obvodov**

### ***Tvorba simulácií elektrického obvodu***

Na vytvorenie simulácie elektrického obvodu budete používať nástroje a pracovnú plochu. Simuláciu vytvoríte v troch krokoch:

1. Umiestnite súčiastky elektrického obvodu na pracovnú plochu
2. Nastavíte vlastnosti umiestnených súčiastok
3. Prepojte súčiastky elektrického obvodu na pracovnej ploche čiarami, ktoré simulujú káble

### ***Nastavenie vlastností súčiastok***

Súčiastky pri ich umiestnení na pracovnú plochu nadobudnú prednastavené hodnoty ich vlastností. Na zmenu týchto vlastností postupujte nasledovne:

1. Stlačte na súčiastku ľavým tlačidlom myšky.
2. V okne vlastností upravte žiadanú hodnotu.

Vlastnosti súčiastok môžete meniť aj za behu simulácie. Simulácia sa upraví podľa novonastavených hodnôt vlastností súčiastok.

### ***Typy hodnôt a ich nastavenie***

Je viacero typov hodnôt a v závislosti od typu sa mení aj nastavenie. Pri vnášaní hodnôt netreba potvrdiť vnesenú hodnotu. Hodnota sa uloží samá, keď stlačíte mimo políčka alebo stlačením klávesa enter.

Na nastavenie jednotlivých typov hodnôt postupujte nasledovne:

1. Celočíselná hodnota bez obmedzenia - vpište celočíselnú hodnotu do políčka.
2. Celočíselná hodnota s obmedzením - použite posuvník na nastavenie hodnoty alebo vpište hodnotu do políčka.
3. Desatinné číslo bez obmedzenia - vpište celočíselnú hodnotu do políčka. Desatinnú časť oddel'te bodkou (".").
4. Desatinné číslo s obmedzením - použite posuvník na nastavenie hodnoty alebo vpište hodnotu do políčka. Desatinnú časť oddel'te bodkou (".").

5. Pravdivostná hodnota (áno/nie) - stlačte na štvorček na zmenu hodnoty.
  1. Keď je štvorček označený, hodnota je “áno”.
  2. Keď štvorček nie je označený, hodnota je “nie”.

### ***Odpojenie dvoch súčiastok na pracovnej ploche***

Aby ste odpojili súčiastku, musíme vymazať čiaru, ktorou je súčiastka napojená na inú súčiastku.

Na vymazanie čiary, postupujte nasledovne:

1. Stlačte ľavým tlačidlom myšky na čiaru, ktorú chcete vymazať medzi dvomi súčiastkami.
2. Stlačte tlačidlo Del na klávesnici.


### ***Spustenie simulácie***

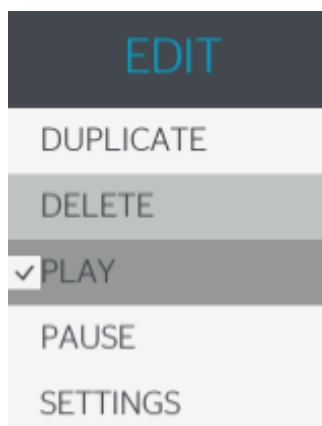
Každá simulácia v nástroji má dva stavy:

1. Spustená
2. Pozastavená

### ***Spustenie simulácie***


Na spustenie simulácie, urobte jeden z nasledovných krokov:

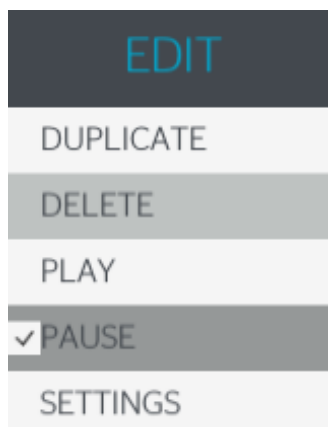
- Stlačte tlačidlo šípky (spustiť)  v hlavnom paneli nástrojov.
- Stlačte na položku spustiť v položky Upraviť v hlavnom menu.



### ***Pozastavenie simulácie***



Na pozastavenie simulácie, urobte jeden z nasledovných krokov:

- Stlačte tlačidlo dvoch vertikálnych čiar (pozastaviť)  v hlavnom paneli nástrojov.
- Stlačte na položku pozastaviť v položky Upraviť v hlavnom menu.



### ***Zobrazenie aktívneho stavu simulácie***

Aktívny stav je zobrazený červenou farbou v hlavnom paneli nástrojov a označeným políčkou v položke Upraviť v hlavnom menu:

-  simulácia je spustená.
-  simulácia je pozastavená.

### ***Vygenerovanie HTML exportu***

Pre vygenerovanie exportu kliknite ľavým tlačidlom myšky na ikonku obláčika v hlavnom paneli nástrojov, ktorý otvorí bežný Windows Explorer.

HTML export bude uložený do vami vybraného priečinka.

## **5.5.4 Vytvorenie, načítanie a uloženie projektu**

### ***Vytvorenie***

Tlačidlo pre vytvorenie nového projektu v momentálnom nastavení spraví iba to, že vymaže všetky objekty z pracovnej plochy.

### ***Uloženie***

Tlačidlo pre uloženie projektu v momentálnom nastavení otvorí bežný Windows Explorer, ktorý perzistuje všetky objekty z pracovnej scény do jedného súboru na disku vo formáte .es.

### ***Načítanie***

Tlačidlo pre načítanie projektu v prvom kroku vymaže všetky objekty z pracovnej plochy. V druhom kroku otvorí Windows Explorer a prečíta súbor, ktorý mu vyberiete. Z tohoto súboru vytvorí objekty na pracovnej ploche nanovo.

## *Nastavenia*



V hlavnom menu Upraviť → Nastavenia sa nachádzajú nastavenia aplikácie.

Kliknutím na tlačidlo s aktuálnym jazykom je možné zobrazit' a vybrať dostupné jazyky (slovenčina a angličtina).

Pre opustenie nastavenia kliknite na tlačidlo *Zrušiť* alebo *Uložiť*.

Jazyk je možné nastaviť podobným spôsobom, tlačidlom v hornom pravom rohu nástroja.