STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12: Tvorba učebních pomůcek, didaktická technologie

3D Simulátor obvodů

Jakub Seidl
Praha

Praha 17. 03. 2019

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 12: Tvorba učebních pomůcek, didaktická technologie

3D Simulátor obvodů

3D Circuit Simulator

Autoři: Jakub Seidl
Škola: PORG – gymnázium a základní škola, o. p. s.
Kraj: Praha
Konzultant: Mgr. Tomáš Fabián, Ing. Libor Seidl, CSc.

Praha 17. 03. 2019

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

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

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

V Praze dne 17. 03. 2019 ……

Jakub Seidl

Poděkování

Chtěl bych poděkovat našemu profesorovi fyziky panu Tomáši Fabiánovi, který mi poskytnul cenné rady při návrhu programu, paní profesorce Magdaléně Janichové, která mi pomohla s přípravou práce pro SOČ, svým spolužákům za spolupráci a pomoc při testování, svým rodičům a všem, kteří mě během vývoje programu podporovali.

Anotace

Tato práce se zabývá vývojem počítačové aplikace pro simulaci elektrických obvodů a jejich názornou 3D vizualizaci pomocí analogie s vodou. Cílem bylo ukázat a vysvětlit jevy probíhající v elektrických obvodech méně abstraktním způsobem. Výsledkem mé práce je program využívající herní engine Unity, kde je možné nakreslit obvod pomocí 13 základních součástek, simulovat jeho chování a vizualizovat obvod ve 3D tak, jak by fungoval, kdyby byla místo elektřiny použita voda.

Klíčová slova

Obvod – Simulátor – 3D – Unity

Annotation

This work describes the development of a computer application which can simulate electronic circuits and visualize them in 3D using the analogy with water. The goal was to show and explain the behaviour of circuits in a less abstract way. The result is a computer program build in the Unity game engine where the user can build a circuit using 13 basic circuit parts, simulate the behaviour of the circuit, and see in 3D how the circuit would work if water were used instead of electricity.

Keywords

Circuits – Simulator – 3D – Unity

Obsah

1 Úvod

Když jsme se začali ve škole učit o elektrických obvodech, zjistil jsem, že je pro některé mé spolužáky abstraktní koncept napětí a proudu těžko pochopitelný. Chtěl jsem, aby jen nehádali odpovědi do testů a nesnažili se jen zapamatovat si Kirchhoffovy zákony, aniž by je pochopili. Proto jsem svým spolužákům chtěl fungování napětí a proudu v obvodech vysvětlit názorně a pochopitelně a tak, abych je nezahltil abstraktními a špatně pochopitelnými čísly – tak, aby si uměli obvod lépe představit.

Rozhodl jsem se pro často používanou analogii k napětí a proudu – vodu. Místo, abych nakreslil jen obvod s obtížně uchopitelnými čísly pro napětí, odpor a proud, můžu obvod ukázat jako soustavu trubek s vodou a součástek fungujících na bázi vody. Napětí si je možné představit jako tlak vody, který lze následně ukázat jako výšku hladiny ve skleněném válci, a proud si lze představit jako objem vody, který proteče skrz danou trubku za daný čas.

Protože jsem nechtěl udělat jen ukázku jednoho obvodu nebo model jen jednoho případu, a protože jsem se zrovna začínal učit v herním enginu Unity, rozhodl jsem se, že se pokusím naprogramovat aplikaci, kde by bylo možné nakreslit elektrický obvod a následně ho automaticky vizualizovat ve 3D již zmíněným způsobem.

Mým cílem bylo tedy vytvořit program pro simulaci a analogickou 3D vizualizaci elektrických obvodů, který by zároveň uměl simulovat proud v reálném čase v krocích (simulace nebude předem vypočítaná), aby bylo možné sledovat tok proudu v jednotlivých součástkách a vodičích a aby bylo možné do obvodu během simulace zasahovat. Uživatel bude mít možnost 3D modelem volně procházet. Dával jsem přednost názornosti před přesností a rychlostí, protože už existuje spoustu jiných rychlejších simulátorů, kterým ale právě chybí názorné vysvětlení funkce; rychlost a přesnost jsem musel obětovat kvůli krokové simulaci.

Vývoj programu se skládal ze tří částí. První jsem musel udělat funkční nadstavbu pro program, prostředí pro kreslení obvodů ve 2D a systém pro jejich následnou vizualizaci ve 3D, včetně ovládání pohledu. Druhou velkou částí bylo uživatelské rozhraní, návod a intuitivní ovládání. Třetí částí bylo přidání samotného obsahu – součástek. V tom spočívalo nakreslení 2D symbolů pro součástky, vytvoření 3D modelů fungujících na bázi vody a vytvoření algoritmů pro simulaci dané součástky (pro 2D i 3D).

2 VÝVOJ PROGRAMU

2.1 Nápad a návrh

Samotnému vývoji programu předcházel návrh 3D interpretace elektrických obvodů. Jak jsem již zmínil, napětí si lze představit jako tlak vody a proud jako objem vody protékající trubkou. Rozhodl jsem se zanedbat odpor propojovacích vodičů, takže relativní hladina napětí bude jednotná pro každý uzel (vzájemně propojené dráty). V každém uzlu tedy může být ve vizualizaci umístěný skleněný válec napojený na trubky, v němž bude hladina vody představovat napětí vůči relativní nule. Během simulace se také v trubkách bude pohybovat textura vody ukazující proud.

Výpočet bude probíhat iteračně v reálném čase. Na začátku každého kroku se vychází z napětí v jednotlivých uzlech. Z rozdílů napětí mezi uzly a hodnot odporu součástek zapojených mezi uzly se spočítá velikost proudu, tedy velikost náboje, který se přesune mezi uzly. Změny náboje se projeví změnou výšky vodní hladiny v jednotlivých válcích (změnou uzlového napětí). Objem vodního sloupce v jednotlivých uzlech odpovídá elektrické kapacitě uzlu vzhledem k zemi.

Velikost kroku simulace určuje rychlost a přesnost simulace – malá velikost kroku by znamenala sice přesnou, ale pomalou simulaci; velký krok by simulaci urychlil, ale zato v některých případech způsobuje nestabilitu nebo nepřesnost (vysvětleno dále).

2.2 Pokusná verze 1

Protože jsem se začal učit pracovat v Unity[5] a programovat v jazyce C#[6] jen půl roku před tím, než jsem dostal tento nápad, neměl jsem ještě skoro žádné zkušenosti s vytvářením hotové aplikace. Chtěl jsem tedy nejprve naprogramovat pilotní verzi programu, abych mimo jiné zjistil případný zájem o projekt u spolužáků. První verze obsahovala neposuvnou mřížku omezené velikosti a jen 4 součástky: zdroj stejnosměrného proudu, rezistor, žárovku a spínač. Součástky šlo spojovat jednoduchým drátem do paralelního i sériového zapojení, na žárovce a rezistoru bylo možné nastavit velikost odporu, na zdroji velikost napětí a na spínači bylo možné ovládat stav sepnutí. Program také uměl obvod vizualizovat ve 3D, kde si následně uživatel mohl volně prohlížet obvod, vodu tekoucí v drátech. Ostatní funkce programu jako změna rychlosti simulace, ukládání a načítání obvodů, návod a další součástky nebyly ještě přidány.

Program jsem prezentoval ve třídě s úspěchem, a proto jsem začal vyvíjet druhou verzi simulátoru jako téma na školní malé maturity z fyziky (semestrální práce na konci kvarty osmiletého gymnázia).

2.3 Verze 2

Druhou verzi programu jsem začal vytvářet znova úplně od základu. Předchozí verze totiž nebyla programována dost udržovatelně a mnoho systémů bylo potřeba úplně přepracovat.

Druhá verze už měla pohyblivou neomezeně velkou mřížku, podporu ukládání a načítání obvodů, vestavěnou uživatelskou příručku, mnohem více součástek a vylepšené uživatelské rozhraní. Implementoval jsem také přesnější simulaci a přidal jsem několik funkčně úplně odlišných součástek, například voltmetr a ampérmetr. Po dokončení semestrální práce jsem program vystavoval na Maker Faire 2018 v Praze.

2.4 Pokračování vývoje

Po dokončení semestrální práce jsem ale program pro velký zájem ostatních neopustil a dále jsem se jej snažil stabilizovat a přidávat některé další funkce. Protože běžně při programování pracuji v angličtině, byl do té doby v angličtině i celý návod – ten jsem přeložil do češtiny a přidal možnost přepnutí jazyka návodu (program se nejprve pokusí nastavit výchozí jazyk podle jazyka systému). Program začal být také využíván v kvartě jako pomůcka k výuce obvodů, takže mi studenti nahlásili další chyby, na které narazili, abych je mohl opravit.

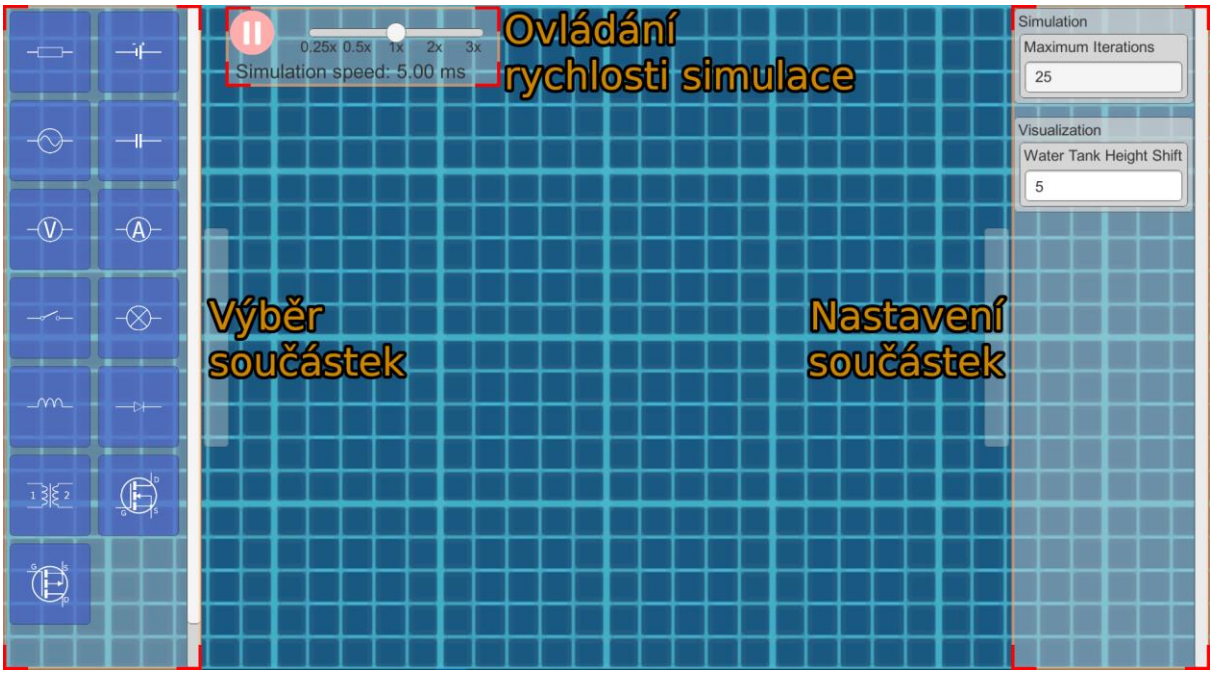
3 KONCEPTY PROGRAMU

3.1 Uživatelské rozhraní

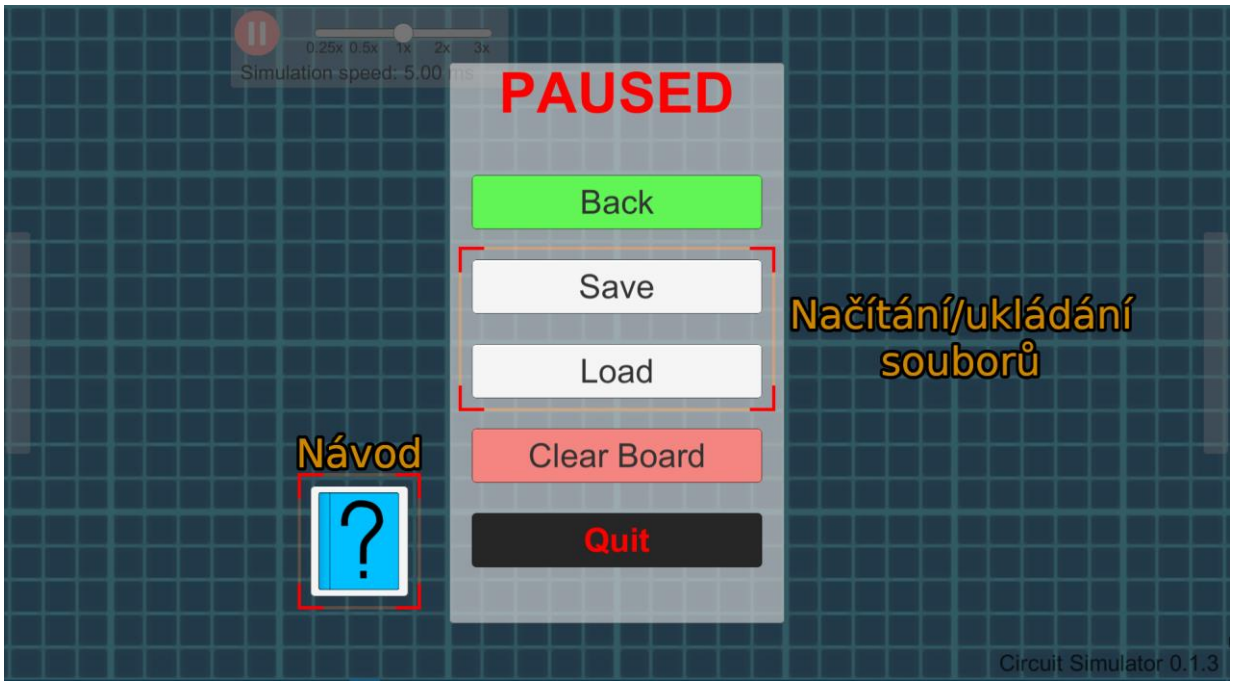
Uživatelské rozhraní programu se skládá z několika částí. Prvky související se samotnou úpravou obvodů jsou na hlavní obrazovce [Obr. 1] a ostatní se nacházejí v menu, které se vyvolá při pauze [Obr. 2].

Na hlavní obrazovce je vlevo umístěný výběr součástek, které je možné umístit na mřížku. Zatím je implementovaných 13 různých typů součástek, ale program byl navrhován tak, aby bylo možno další doplnit. Součástky se musí nejprve uchopit kliknutím na jejich ikonu, následně je možné uchopenou součástku otáčet klávesami R po směru a Shift + R proti směru hodinových ručiček, součástka se umístí na desku dalším kliknutím myši. Pokládání součástky je také možné zrušit kliknutím pravého tlačítka myši. Na pravé straně je okno, kde je možné vidět a nastavit vlastnosti vybraných součástek (napětí, proud, odpor…), popř. nastavení simulace jako celku, pokud není vybraná žádná součástka. Třetí okno vlevo nahoře je určeno k ovládání rychlosti simulace. Simulaci je možné zrychlit až na trojnásobek a zpomalit až na čtvrtinu normální rychlosti, nebo je ji nožné úplně zastavit pomocí červeného tlačítka pauzy. Údaj *Simulation speed* určuje, kolik milisekund program odsimuluje za jednu sekundu reálného času – základní rychlost 5 ms/s je tedy 200krát zpomalená simulace oproti realitě.

V menu při vyvolání pauzy jsou k dispozici ostatní funkce programu, některé nepřímo související s úpravou obvodů. Tlačítkem *Back* se menu zavře. Tlačítky *Load* a *Save* se načte nebo uloží obvod do souboru. Tlačítko *Clear Board* odstraní všechny součástky na desce. A tlačítko *Quit* program ukončí; program je také možno ukončit kliknutím na křížek.

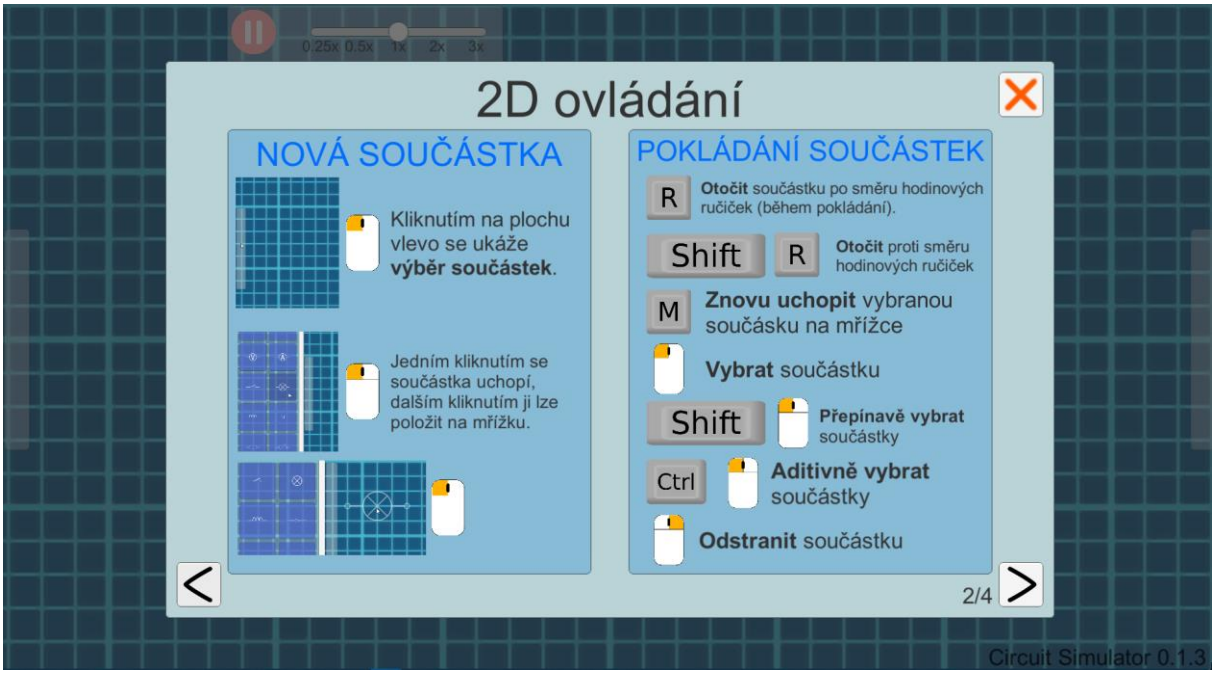


Obr. 1: Popis prvků v programu



Obr. 2: Popis hlavních prvků v pauzovém menu

Kliknutím na obrázek modré knížky se otevře návod. V návodu je popsané ovládání ve 2D prostředí [Obr. 3], 3D prostředí a každá součástka má popsané všechny vlastnosti, které jsou vidět v nastavení součástek. Návody jsou dostupné v češtině i angličtině.



Obr. 3: Ukázka 2. stránky návodu pro 2D ovládání

3.2 Ovládání ve 2D

Snažil jsem se, aby bylo ovládání intuitivní a jednotné pro celý program. Levé tlačítko myši by mělo sloužit k obecné interakci, pravé tlačítko k odstraňování objektů a prostřední tlačítko k posuvu mřížky. S mřížkou je možné pohybovat třemi způsoby: posouváním levým tlačítkem myši, posouváním prostředním tlačítkem myši a přibližováním nebo oddalováním kolečkem myši. Posuv prostředním tlačítkem funguje v každé situaci, protože nemá nastavenou žádnou jinou funkci. Kolečkem myši se také ovládá pouze přiblížení mřížky. Pohled se přibližuje vždy směrem ke kurzoru.

Levé tlačítko už má kromě posuvu mřížky přidělenou funkci pokládání a vybírání součástek a kreslení drátů, tyto akce lze provést pouze kliknutím bez pohybu kurzoru; pohnutím kurzoru po zmáčknutí levého tlačítka se začne mřížka posouvat a ostatní funkce se dočasně zakážou, dokud uživatel levé tlačítko nepustí. Levé tlačítko myši se také používá pro obecnou interakci s ostatními prvky mimo mřížku – pomocí něho je možné uchopit součástky z levého okna výběru součástek, označit hodnoty v pravém okně nastavení součástek a klikat na všechna ostatní tlačítka.

Základní funkci pravého tlačítka myši je odebírání součástek a propojujících drátů. Kliknutím na položenou součástku se součástka odebere, kliknutím na drát se označený drát smaže a kliknutím s uchopenou součástkou se pokládaná součástka také odstraní.

Ovládání na klávesnici jsem částečně převzal z jiných programů a částečně nastavil podle anglických názvů akcí. Zkratky R a Shift + R jsou pojmenované podle *Rotate*, M podle *Move*

a funkce Ctrl a Shift jsou inspirovány ostatními programy (v každém programu tyto klávesy fungují jinak, ale funkce Shift je převážně jednotná).

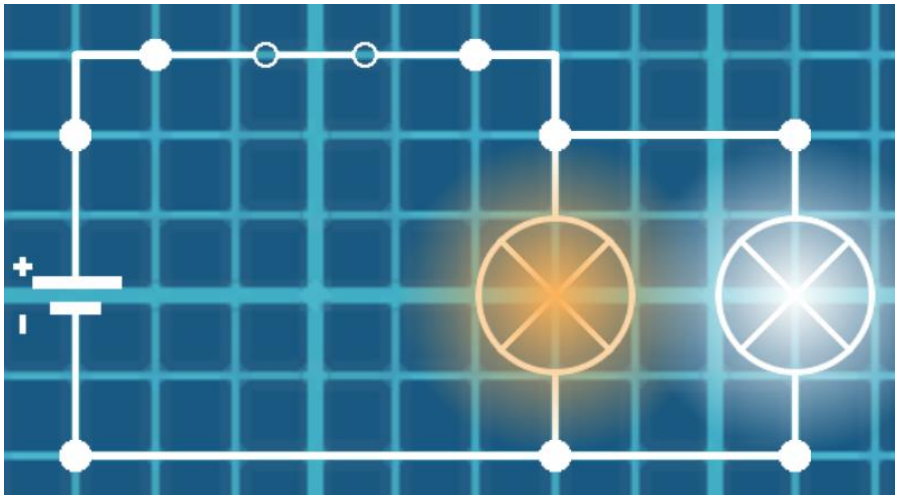
3.3 Ovládání ve 3D

Ovládání ve 3D je převážně inspirováno ovládáním pohledu přímo v prostředí Unity Editoru. Stejně jako v Unity, pohyb pohledu se aktivuje držením pravého tlačítka. Následně je možné pohybovat myší, aby se pohled otáčel, a používat klávesy W A S D a Q E pro pohyb dopředu, doleva, dozadu, doprava a dolů, nahoru. Pohyb lze dále zrychlit nebo zpomalit klávesami Shift nebo Ctrl. Tyto klávesy (zejména WASD) jsou využívány ve velké většině 3D programů a her, takže pro uživatele, kteří mají předchozí zkušenost s prací např. ve 3D modelovacích programech nebo s hraním her, je ovládání velmi intuitivní. WASD se upřednostňuje před šipkami, protože se tyto klávesy nacházejí v podobné formaci, ale na rozdíl od šipek jsou na levé straně klávesnice (což je při držení myši v pravé ruce pohodlnější), a zároveň od nich lze jednoduše dosáhnout na mezerník, Shift a Ctrl.

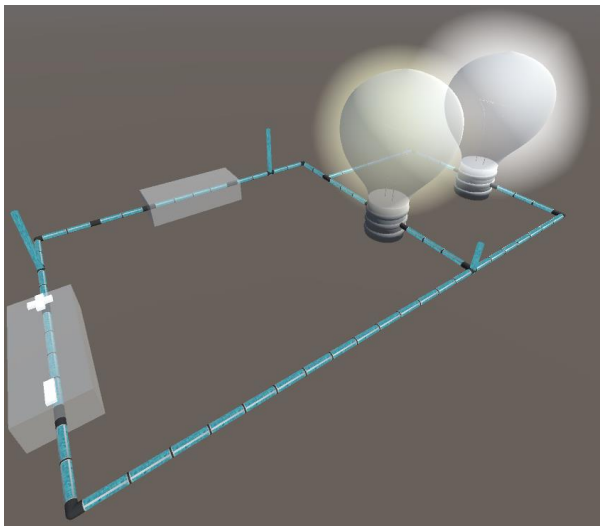
Podobně jako ve 2D lze 3D součástky označit a upravit jejich vlastnosti, ale na rozdíl od 2D v tomto pohledu není možné součástky přidávat, přesouvat, přepojovat, ani odstranit.

3.4 Zobrazení ve 2D

Cílem 2D zobrazení je simulovat kreslení obvodu na papír nebo tabuli, takže zde nejsou vidět napětí ani proudy ve drátech a součástky používají konvenční symboly[7]. Jedinými výjimkami je žárovka [Obr. 4], která zároveň ve 2D svítí podle svého napětí a teploty, voltmetr a ampérmetr, kde je možné povolit zobrazování grafu přímo na součástce ve 2D.



Obr. 4: Sériově zapojené žárovky ve 2D pohledu. Jedna z žárovek má vyšší konstrukční napětí, takže svítí méně.

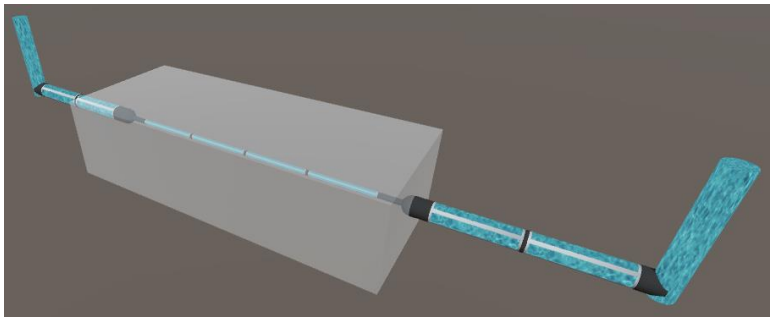

Obr. 5: Obvod z [Obr. 4] ve 3D

3.5 Zobrazení ve 3D

Na rozdíl od 2D by měla být 3D vizualizace názorná a podle ní by měli být studenti schopni pochopit analogii mezi elektřinou a vodou. Součástky jsem se tedy pokusil vymodelovat tak, jak by vypadaly, kdyby fungovaly na bázi vody. Zároveň jsem se pokusil zachovat tvar součástky podle konvenčních symbolů (např. 3D model rezistoru je obsažen v kvádru o stejných dimenzích jako 2D symbol). Bohužel, protože nemám dost zkušeností s 3D modelováním nebo protože jsem nenašel vhodnou analogii s vodou, mají některé součástky jen jednoduchý nebo symbolický model.

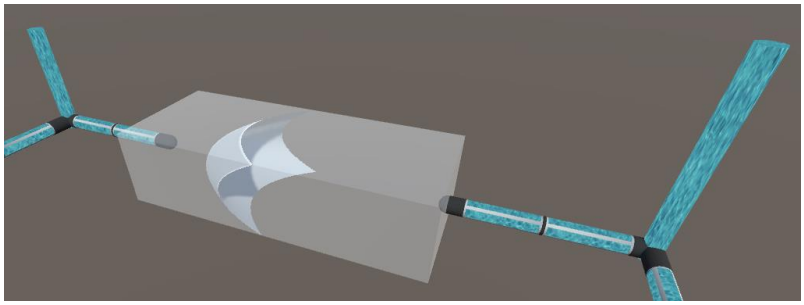
3.5.1 3D rezistor

3D model rezistoru [Obr. 6] je kapilára, která má tok vody brzdit. Průměr modelu kapiláry se také upravuje podle hodnoty odporu nastavené na rezistoru.


Obr. 6: 3D model rezistoru

3.5.2 3D kondenzátor

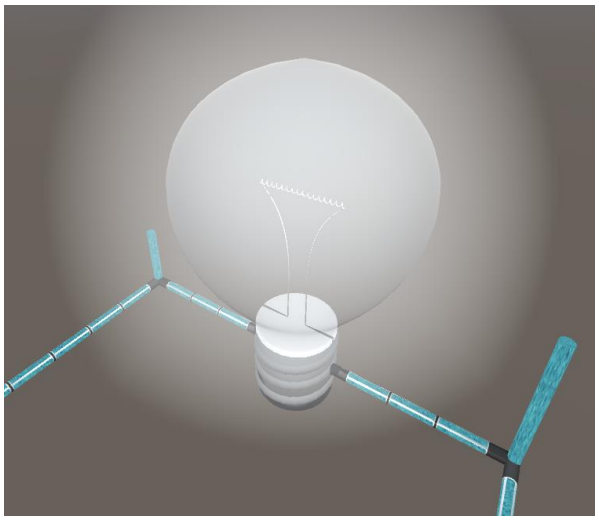
3D model kondenzátoru [Obr. 7] představuje nádrž s vodou a membránou uprostřed. Napětí kondenzátoru se zobrazuje jako prohnutí membrány na danou stranu. Pokud se kondenzátor odpojí od vnějšího zdroje napětí a obě strany se spojí obvodem, membrána se začne narovnávat a „vytlačuje“ tak vodu z jedné strany na druhou, dokud se membrána nenarovná.



Obr. 7: 3D model kondenzátoru s napětím 3 V

3.5.3 3D Žárovka

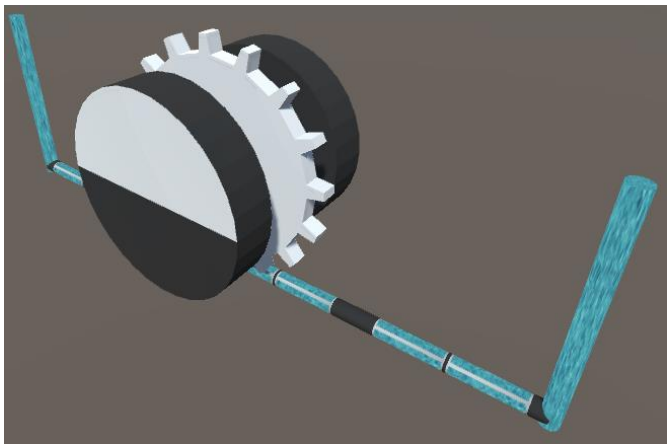
3D model této součástky [Obr. 8] je relativně jednoduchý, skládá se z trubek připojených na patici žárovky. Žárovka má průhledné sklo a vlákno, které mění barvu a září podle teploty. Chování vlákna je vymodelováno podle wolframu. Pokud se vlákno přehřeje (teplota se přiblíží bodu tání), poruší se a přestane vodit.



Obr. 8: 3D model žárovky svítící na plný výkon

3.5.4 3D cívka (induktor)

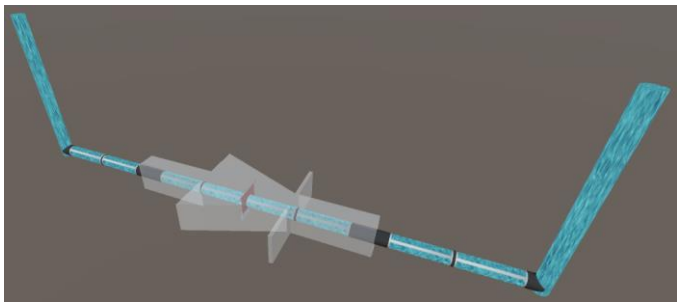
3D cívka [Obr. 9] má úplně jiný vzhled než ve 2D. Chování cívky si je možné představit jako setrvačník poháněný vodou. Velikost setrvačníku ve 3D se mění podle hodnoty indukčnosti.



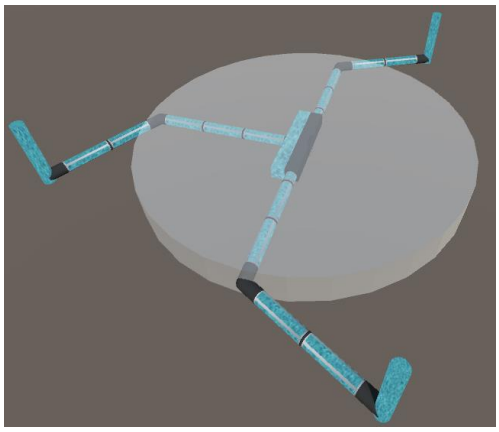
Obr. 9: 3D model cívky

3.5.5 3D dioda

Dioda je vymodelovaná jako trubka s červenou jednosměrnou záklopkou [Obr. 11], která umožňuje průtok jen jedním směrem (a v případě průrazu i druhým). Obal 3D modelu diody zachovává tvar 2D schematické značky. Chová se podle Shockleyovy rovnice[1][3].



Obr. 11: 3D model diody



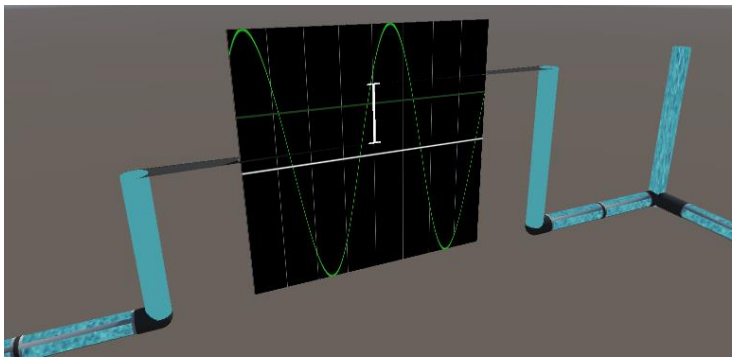
Obr. 10: 3D model NMOS tranzistoru; svorky odshora dolů: drain, gate, source

3.5.6 3D tranzistory

V programu jsou dostupné dva druhy tranzistorů: NMOS a PMOS. Jsou vymodelovány jako trubka s uzávěrem, který je ovládaný vedlejším přítokem (nábojem na hradle) [Obr. 10]. Ovládání průtoku pro NMOS a PMOS se chová opačně, fyzikálně logicky se chová jen PMOS (přítokem vody na hradlo se odsune uzávěr a hlavní kanál se přiškrtí, odtokem se uzávěr vrátí zpět).

3.5.7 3D voltmetr

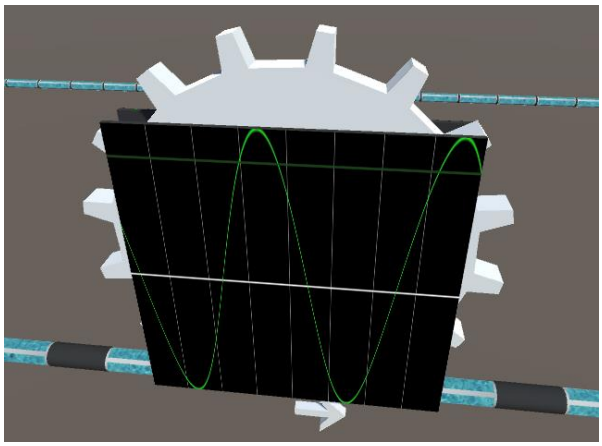
Součástka voltmetru [Obr. 12] je reprezentována dvěma vodními válci umístěnými naproti sobě s ukazateli výšky hladiny. Mezi ukazateli je bílá kóta ukazující rozdíl hladin, tedy napětí. Mezi vodními válci je také umístěn graf měřeného napětí, kam se zapisuje rozdíl obou vodních hladin. Zobrazením kóty mezi ukazateli by mělo být názorné, že se jedná pouze o rozdíl hladin, takže pokud se sníží hladiny na obou stranách stejně, napětí neklesne.



Obr. 12: 3D model voltmetru zaznamenávající střídavé napětí

3.5.8 3D ampérmetr

Ampérmetr je reprezentován vodním kolem poháněným proudem. Na kole je také přichycená obrazovka s grafem zaznamenaného proudu. Pod vodním kolem je navíc model šipky zvýrazňující směr a rychlost toku proudu – z něho by mělo být názorné, že záleží na velikosti a směru toku proudu, nikoliv například době trvání toku.



Obr. 13: 3D model ampérmetru zaznamenávající střídavý proud

4 FUNKCE PROGRAMU, SIMULACE A VÝPOČTY

Aby program mohl ukazovat okamžitý proud v každé součástce a každém drátu a zároveň aby zachoval interaktivitu, tedy možnost zasahovat do obvodu za chodu, musí simulace probíhat po krocích v reálném čase.

4.1 Základní princip simulace

Během každého kroku simulace se postupně aktualizuje stav všech součástek a následně se vyrovnají nastřádané náboje v drátech. Aktualizace součástek probíhá tak, že každá součástka má k dispozici momentální stav napětí na každé svorce (případně si může ukládat vlastní interní hodnoty) a podle těchto hodnot odebere nebo přidá svorkám nějaký náboj. Jako příklad, rezistor si vypočítá proud podle napětí, vynásobením proudu velikostí kroku získá náboj a tento náboj odečte od jedné svorky a přičte ke druhé. Z nastřádaného náboje se na konci kroku vypočítá změna napětí na každém uzlu.

Velikost kroku ovlivňuje rychlost a přesnost simulace. Větší krok umožňuje odsimulovat více času za menší počet cyklů, ale výpočty mohou začít ztrácet na přesnosti. Simulace nedává dobré výsledky zejména tehdy, když se v obvodu vyskytují prvky, u kterých dochází k přechodovým jevům kratším, než je velikost simulačního kroku. Sériový odpor žádné součástky by tedy neměl být příliš malý, stanovil jsem 1 Ω jako minimum. Pokud bychom chtěli tuto hodnotu snížit, je to možné zkrácením simulačního kroku, čímž se ovšem výpočet prodlouží, a nebo zvětšením virtuálních kapacit uzlů.

4.2 Implementace simulace

Aby mohla simulace probíhat přesouváním náboje, každý drát má malou virtuální kapacitu, takže se chová, jako by byl připojený k zemi malým kondenzátorem. Tuto kapacitu si lze ve 3D představit jako průřez vodního válce. Podobně jako velikost kroku tato kapacita ovlivňuje přesnost a stabilitu simulace. Větší kapacita sice zajišťuje stabilitu, ale může vyvolat nechtěné vedlejší efekty při stabilizaci, zejména u střídavého proudu. Ty se projevují například tak, že skrz drát může protékat chvíli malý proud, i když není obvod uzavřený, protože se naplňuje kapacita drátu; podle analogie s vodou to znamená, že se naplňuje vodní sloupec.

Původní implementace měla jen jeden krok simulace v každém cyklu – každá součástka jednou vypočítala náboj, který přesunula, a následně se napětí na všech drátech zprůměrovalo. Tato metoda ale způsobovala nestabilitu, zejména v obvodech se střídavým proudem, a kroky by musely být velmi malé. Využil jsem tedy metodu Runge-Kutta[2]. Výpočet přesunu náboje proběhne pro každou součástku celkem čtyřikrát každý cyklus, pokaždé s počátečními hodnotami z předchozího kroku, a tyto hodnoty jsou následně zprůměrovány podle vzorce:

$$\frac{x_0+2x_1+2x_2+x_3}{6} \tag{1}$$

Tato metoda dává přesnější výsledky a je mnohem rychlejší než více výpočtů s menšími kroky. Po vypočítání zprůměrovaného náboje pro přesunutí každá součástka upraví náboj na svých svorkách. Protože se součástky aktualizují postupně, mohly by dříve aktualizované součástky ovlivnit součástky aktualizované později, takže se náboj mění v odděleném zásobníku každého konektoru. Jakmile se dokončí aktualizace součástek, zjistí se celková změna náboje v drátu, ze které se vypočítá pomocí virtuální kapacity drátu celková změna napětí.

Za ideálních podmínek proběhne simulace 25krát za snímek a program stihne 60 snímků za sekundu, takže simulace proběhne 1500krát za sekundu. Pokud je obvod moc náročný na simulaci nebo pokud hardware nestíhá, simulace se může začít zpomalovat. Případné zpomalení se odráží v hodnotě *Simulation speed* [Obr. 1], která průměruje odsimulovaný čas za několik posledních snímků. Zpomalení se projevuje častěji, pokud je simulace urychlena.

```
public override void Update(int rkStep)
{
    voltage.Value = part[connector2].Voltage - part[connector1].Voltage; // Voltage delta

    float curr = voltage / resistance;

    rkCurrent[rkStep] = curr; // Writing the current to be averaged using the Runge-Kutta method

    if (rkStep == 3) // Steps are 0 to 3, step 3 is the last
    {
        // Get the processed current using the formula (a + 2b + 3c + d) / 6, where a, b, c, d are currents from previous steps
        current.Value = rkCurrent;
    }

    // Move charge from one connector to the other. The moved charge gets buffered until all circuit parts are processed
    part[connector1].chargeBuffer += curr * GameManager.Instance.CurrentTimeStep;
    part[connector2].chargeBuffer -= curr * GameManager.Instance.CurrentTimeStep;
}
```

Obr. 14: Ukázka kódu rezistoru, dokumentace v angličtině

Na [Obr. 14] je ukázka kódu, který aktualizuje stav součástky rezistoru. Tato funkce je zavolána čtyřikrát (s hodnotami 0 až 3) – jednou pro každý krok Runge-Kuttovy metody. Tato funkce jednoduše vypočítá proud podle Ohmova zákona (napětí / odpor) a zaznamená postupně čtyři různé hodnoty (mezi jednotlivými kroky Runge-Kuttovy metody se napětí na svorkách mění). Tyto čtyři hodnoty jsou v posledním kroku zprůměrovány a zapsány do vlastnosti „Proud“ („Current“), kterou uživatel může sledovat při označení součástky. Na konci každého kroku je také na svorkách změněn zásobník změny náboje podle proudu a velikosti kroku (změna je aplikována vždy po skončení kroku pro všechny součástky naráz).

4.3 Ostatní systémy

Kromě systému simulujícího obvody a systému vizualizujícího obvod ve 3D se program skládá ještě z několika dalších klíčových systémů. Jedním z nich je například samotné posouvání mřížky. Mřížka musí být nekonečná (omezená jen maximální velikostí čísel na počítači), ale pokrýt celý prostor opakujícím se 2D obrázkem mřížky by nebylo jednoduché, a navíc by možná vedlo k dalším problémům. Mít zafixovaný pohled a pohybovat všemi součástkami najednou by taky nebylo vhodné a mohlo by program značně zpomalovat. Tyto dva problémy jsem vyřešil tak, že místo mřížkou uživatel pohybuje samotným pohledem, a mřížka vždy pokrývá pouze to, co je vidět – mřížka je 2D objekt s opakující se texturou a mění svoji velikost a pozici vždy tak, aby pokrývala celý 2D pohled. Její okraje je možné zahlédnout během přechodu do 3D, kdy se mřížka postupně zprůhlední.

Přechod do 3D bylo také náročné vyřešit. Chtěl jsem, aby to byl plynulý zpomalený pohyb a aby se pohled přemístil tam, kam uživatel očekává. Nejprve jsem přechod nastavil tak, aby se plynule otáčel okolo středu obrazovky při přechodu tam i zpět, ale tam se objevilo několik problémů, když uživatel přešel zpět do 2D ještě během animace do 3D. Dalším pokusem bylo přemístit pohled uživatele směrem k ukazateli myši a nasměřovat ho směrem ke středu obrazovky; pohled se taky už vždy vracel na stejné místo, kde byl ve 2D naposledy. To se ale ukázalo jako neintuitivní, protože uživatel většinou kurzor při přechodu nechával na náhodném místě na obrazovce, takže pohled se vždy přemístil náhodně a pro uživatele neočekávaně. Nakonec jsem se rozhodl pohled při přechodu do 3D přemístit do středu 2D pohledu a nasměrovat ho na místo kurzoru. To se ukázalo jako asi nejlepší řešení, protože se pohled vždy přesouval na stejné místo nezávisle na pozici myši, a pokud uživatel o funkci nasměrování za myší věděl, mohl jí využít.

4.4 Problémy při vývoji

Protože toto byl můj první velký projekt v Unity a neměl jsem tolik zkušeností na začátku, narazil jsem při vývoji na spoustu problémů. Hned první byl problém obrazovek s různým rozlišením. Já mám jen obrazovku s poměry stran 16:9, ale ve škole jsou také obrazovky s poměry 4:3 a 16:10, takže se na nich některé prvky zobrazovaly na špatných místech. Musel jsem tedy přizpůsobit uživatelské rozhraní všem těmto poměrům, s čímž jsem neměl žádné zkušenosti. Neočekávaným problémem také byla záhadná chyba, kdy mi na některých počítačích nešly načíst uložené soubory. Zjistil jsem nakonec, že se na počítačích s anglickým Windows (např. na mém počítači) převádějí desetinná čísla na text s tečkou, zatímco český Windows očekává desetinnou čárku – formát čísla tedy nepozná a nahlásí chybu. Řešení naštěstí nebylo těžké, protože funkce na převod čísel na text a zpět má parametr, který zajistí, že program vždy použije a očekává tečku v desetinných číslech.

Program se také někdy vůbec nedařilo spustit na některých školních počítačích přímo z disku, ale běžel normálně z flash disku nebo z externího úložiště.

Naštěstí jsem velmi často nebyl první, kdo řešil většinu obecných problémů, na které jsem narazil. Tam mi pomohly zejména weby answers.unity.com, stackoverflow.com a komunita na reddit.com/r/Unity3D. Dokončením tohoto projektu jsem se naučil s většinou takových problémů počítat a předcházet jim, takže by další verze měla být robustnější a rychlejší.

Problémy nebyly jen s vývojem programu, ale i při uživatelském testování studenty. Ověřil jsem si, že i když se snažím podchytit všechny chyby, někteří uživatelé se přesto chovají tak, jak jsem neočekával, a najdou novou chybu. Ovládání také bohužel nebylo plánováno s možností přenastavení kláves, což byl problém zejména ve 3D pohybu pro ty, co neměli žádné zkušenosti s ovládáním WASD ve 3D programech nebo hrách.

Určité potíže nastaly při simulaci rozsáhlejších obvodů s větším počtem uzlů. Šíření elektrického náboje v těchto obvodech trvá déle (jednotky milisekund), což neodpovídá fyzikální realitě. Tento jev se projevil například v úloze, kde bylo zapojeno několik desítek žárovek v sérii. Po uzavření obvodu se napětí na žárovkách rozložilo zprvu nerovnoměrně,

takže první žárovka praskla. To je způsobeno tím, že dráty mají moc velkou virtuální kapacitu oproti fyzikální realitě (zhruba o 5-6 řádů, mikrofarady namísto obvyklých desítek pikofaradů). Tento problém se pokusím zmírnit v příští verzi.

5 PROGRAM V PRAXI

Důležitou částí vývoje každé aplikace je zkoušení a zpětná vazba uživatelů. Během testovací fáze programu ještě před dokončením jsem program zkoušel několikrát se svými spolužáky. Takto jsem získal zkušenost, že právě testování programu je nerychlejší způsob, jak najít chyby a nedostatky programu, protože se většina uživatelů chová neočekávaně. Protože já už jsem s programem zvyklý pracovat a vím, jak se s ním pracuje, na většinu takových chyb bych nikdy nepřišel. Program musí být dost robustní, aby zvládl i neočekávané chování uživatele, např. zapojení drátu na špatné místo, nečekané odstranění součástky, moc vysoké a nízké hodnoty apod. Bohužel se mi některé chyby nepovedlo reprodukovat doma (a občas nešly reprodukovat znova samotným uživatelem), ale i tak jsem díky tomuto zkoušení opravil mnoho nedostatků.

Po prezentaci tohoto programu jako semestrální práci jsem nepřestal ve vývoji. Kromě dalších opravených chyb jsem udělal ještě několik malých úprav, jako například změna barvy mřížky kvůli kontrastu nebo překlad návodu do češtiny (celý program byl původně v angličtině). Pro účely práce na SOČ jsem provedl ještě jedno zkoušení s ostatními studenty. Vytvořil jsem čtyři nedokončené obvody se zadáním, abych zjistil, jak se studentům bude s programem pracovat. Výsledky dopadly mnohem lépe než během předchozích zkoušení, největším problémem pro uživatele byla instalace a spuštění programu. Program zatím není bohužel úplně podporován na Apple produktech, protože nemám doma potřebný hardware, ani na linuxových systémech, takže některým studentům nešel simulátor doma spustit vůbec.

6 POKRAČOVÁNÍ, DALŠÍ VERZE

Tento program byl můj první a také jediný dokončený projekt v Unity, takže jsem díky němu získal mnoho nových zkušeností. Upevnil jsem si vlastní systém programování a naučil jsem se myslet napřed tak, aby byl program připravený na budoucí úpravy nebo změny. Bohužel, protože jsem tyto zkušenosti získával až během programování, program je sice funkční, ale není napsaný a navržený tak, aby ho bylo jednoduché udržovat. Během vývoje jsem často objevil nová řešení nebo přístupy k problémům, takže různé části programu jsou psané různými způsoby. Simulátor se také většinou chová stabilně jen když s ním pracuji já, zatímco ostatní často naráží na nedostatky a chyby, které se občas dají opravit jen restartem aplikace.

Proto jsem se rozhodl, že opravovat tuto verzi by zabralo mnohem déle, než kdybych vytvořil třetí verzi simulátoru obvodů. Od dokončení tohoto projektu jsem přešel na trochu jiný způsob programování, objevil jsem nové nástroje a začal využívat úplně nový systém uživatelského vstupu. Také se už snažím program strukturovat lépe a s cílem, aby přidání nové součástky netrvalo déle, než je potřeba k vytvoření 2D ikony, 3D modelu a implementování vlastního chování. Také jsem se poučil ze zkušenosti uživatelů a došel jsem ke kompromisu, že není nutné, aby program simuloval a zároveň umožňoval přidávání, přesouvání a odstraňování

součástek – uživatel by nejprve nakreslil celý obvod ve 2D, který by se následně zpracoval, což by vedlo k optimalizaci a urychlení simulace.

Rád bych z třetí verze programu udělal plnohodnotnou aplikaci, ne jenom mřížku s elektrickými obvody. Mám také v plánu přidat podporu pro úkoly, například kde by student měl vypočítat potřebný odpor na dané součástce. Výhoda práce ve virtuálním prostředí je, že není potřeba zacházet s opravdovými součástkami, takže přehřátá žárovka nebo zkrat zdroje nejsou problém, který by působil škodu. Plánuji možnost pro uživatele vytvořit si vlastní moduly z již existujících součástek – například usměrňovač proudu nebo síťový zdroj s vyhlazováním. Chtěl bych také umožnit obvod v programu nasimulovat napřed a přehrát v reálné rychlosti – uživatel by si mohl napřed nastavit, kdy se změní některé hodnoty na součástkách (např. sepnutí spínače v daný čas), a program by simulaci provedl přesněji a rychleji, ale nebyla by interaktivní (bylo by ale možné sledovat například grafy napětí nebo proudu).

7 ZÁVĚR

Původním záměrem projektu bylo pomoci studentům pochopit a lépe si představit, jak funguje napětí a proud v elektrických obvodech. Nechtěl jsem vytvořit přesný simulátor – ve světě už existuje spousta specializovaných programů pro přesnou simulaci a zkoumání obvodů – ale chtěl jsem, aby si studenti mohli abstraktní elektřinu představit jako něco, co není tak abstraktní a s čím mají více zkušeností. Během testování programu a vysvětlování fungování obvodů některým svým spolužákům jsem se setkal s několika mylnými představami, které jsem se snažil uvádět na pravou míru. Například ve čtvrtém úkolu, který jsem chtěl po studentech vyřešit, byl kondenzátor připojený k žárovce, a chtěl jsem po studentech, aby vysvětlili, proč se po uzavření obvodu začne napětí na kondenzátoru snižovat. Nebylo to tím, že by kondenzátor žárovku nabíjel. Kondenzátor se vybíjí, protože je obvod uzavřený, a proud procházející žárovkou napětí meze elektrodami kondenzátoru vyrovnává a žárovku po tuto dobu rozsvěcí. Proud se také postupně zmenšoval s klesajícím napětím kondenzátoru. Tato situace je možná moc abstraktní na pochopení pouze s elektřinou, ale myslím, že 3D model obvodu, kde kondenzátor je nádrž s napjatou membránou, která se snaží vyrovnat, je mnohem pochopitelnější a představitelnější. Nesnažím se o to, aby studenti správně vypočítali potřebné odpory rezistorů a proudy tekoucí skrz různá složitá paralelní a sériová zapojení. Chci jen studentům ukázat jednoduchý obvod a položit otázku tak, aby zodpovězením pochopili chování obvodu nebo dané součástky. V tomto si myslím, že program uspěl.

Jak už jsem ale zmínil, program je nedostatečně robustní a v rukou nezkušeného uživatele se zatím chová nestabilně. Proto lépe slouží jen jako nástroj k názorné ukázce než úplně funkční simulátor. Pracuji již tedy na třetí verzi tohoto programu, která bude robustnější, stabilnější, rychlejší a rozsáhlejší. Během vývoje programu jsem nasbíral spoustu nových zkušeností a upevnil jsem si vlastní postup práce, takže třetí verze bude už mnohem lépe vnitřně strukturovaná.

Mám také názor, že by simulátor měl být spíše navržený jako hra než jako čistě výukový program. Díky hrám jsem pochopil spoustu náročných témat a konceptů. Ve hře Kerbal Space Program jsem rychle pochopil orbitální mechaniku, proč rakety startují směrem na východ a co to je Hohmannova elipsa. Hra Factorio, kde je cílem postavit továrnu, mě donutila přemýšlet efektivně, počítat poměry pro výrobu potřebných součástek a plánovat strukturovaně. Hra War Thunder, což je simulátor tankových, leteckých a námořních bitev, mě naučila, jak náročné je jen vzletět s letadlem, a že je ještě mnohem těžší přistát. Na rozdíl od striktně výukových programů jsem tyto zkušenosti a znalosti získal, protože mě hra bavila hrát, a abych byl úspěšný, musel jsem tyto koncepty ovládat. Proto bych se pokusil třetí verzi simulátoru navrhnout spíše jako hru a přidat různé interaktivní součástky, aby mohl student sám experimentovat – například součástka reproduktoru, která by vydávala zvuk podle protékajícího proudu jako opravdový reproduktor.

Tento simulátor původně začal jako můj vlastní malý vedlejší projekt, abych pomohl spolužákům pochopit, jak funguje napětí a proud. Nakonec se z toho ale stala semestrální práce z fyziky, a potom dokonce práce na SOČ, což mě mile překvapilo a motivovalo k další práci. Chci proto na tomto projektu dále pokračovat, aby se ze simulátoru jednou stala užitečná pomůcka pro vysvětlování funkce elektrických obvodů.

V příloze přihlášky SOČ je zadání 4 úkolů, které jsem zadával studentům, a 4 videa s řešením. Program je možné si stáhnout na webu, odkaz je také v příloze. Úkoly jsou také dostupné po stažení programu, v programu jsou v nabídce Load umístěny ve složce ukoly.

8 ZDROJE A LITERATURA

1. Shockley diode equation. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2019-03-01]. Dostupné z: https://en.wikipedia.org/wiki/Shockley_diode_equation
2. BUBENÍK, František, Ivana PULTAROVÁ a Milan PULTAR. *Matematické vzorce a metody*. Vyd. 2., přeprac. Praha: České vysoké učení technické, 1997. ISBN 80-01-01643-9. Strany 231–232.
3. HORÁK, Zdeněk a František KRUPKA. *Fyzika: příručka pro vysoké školy technického směru.* 3. vyd. Praha: SNTL - Nakladatelství technické literatury, 1981. Strany 970–975.
4. VEDRAL, Josef a Jan FISCHER. *Elektronické obvody pro měřící techniku.* Vyd. 2. Praha: Vydavatelství ČVUT, 2004. ISBN 80-01-02966-2.
5. Dokumentace Unity. *Unity – Scripting API* [online]. [cit. 2019-03-01]. Dostupné z: https://docs.unity3d.com/ScriptReference/
6. Dokumentace C#. *C# Guide | Microsoft Docs* [online]. [cit. 2019-03-01]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/
7. Předloha pro schématické značky součástek. Electrical Symbols | Electronic Symbols | Schematic symbols [online]. [cit. 2019-03-01]. Dostupné z: https://www.rapidtables.com/electric/electrical_symbols.html

9 SEZNAM OBRÁZKŮ A TABULEK