Mendelova univerzita v Brně
Provozně ekonomická fakulta

Interaktivní 3D modely točivých elektrických strojů

Bakalářská práce

Vedoucí práce:
Ing. Dita Dlabolová

Martin Šimeček

Brno 2016

Rád bych poděkoval Ing. Ditě Dlabolové za vedení této bakalářské práce, její věcné komentáře a připomínky.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Interaktivní 3D modely točivých elektrických strojů**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno 17. května 2016 .....

Abstract

Šimeček, M. Interactive 3D models of rotating electrical machines. Bachelor thesis. Brno 2016.

The Bachelor thesis deals with the creation of interactive application for educational purposes showing the induction and PMDC motors using Blender and Python scripting language. Describes the different Blender tools, process of modeling and creation of executable programs.

Keywords

Blender, 3D model, Python, Blender Game Engine, interactive application, electric motors

Abstrakt

Šimeček, M. Interaktivní 3D modely točivých elektrických strojů. Bakalářská práce. Brno 2016.

Práce se zabývá tvorbou interaktivní aplikace pro studijní účely zobrazující indukční a stejnosměrný elektromotor za použití programu Blender a skriptovacího jazyka Python. Popisuje různé nástroje Blenderu, modelování a tvorbu spustitelných programů.

Klíčová slova

Blender, 3D model, Python, Blender Game Engine, interaktivní aplikace, elektromotory

Obsah

1 Úvod a cíl práce

1.1 Úvod

Za vynálezce prvního jednoduchého elektromotoru je považován Michael Faraday, který se při svých pokusech v roce 1821 inspiroval dánským fyzikem Hansem Christianem Ørstedem a jeho zjištěním, že elektrické a magnetické síly jsou navzájem propojeny. Ørsted to demonstroval na pokusu, kdy položil vodič, kterým procházel elektrický proud, na kompas a jeho ručička se začala vychylovat. Tím de facto položil základy systematického studia elektromagnetizmu (Hans Christian Oersted, 2016).

Michael Faraday, fascinován tímto zjištěním, vytvořil kontrukci, která je dnes nazývaná homopolární motor. I přesto, že tento první elektrický motor neměl žádné uplatnění v praxi, Michael Faraday tímto pokusem umožnil vývoj všem následujícím elektromotorům (První elektrický motor - Faradayův motor, 2016).

Dnes známe nejrůznější typy elektrických motorů. Dělí se podle druhu napájení, kontrukce, použití nebo podle velikosti. Jejich využití je obrovské. Od ručních kuchyňských mixérů,vrtaček nebo fénů na vlasy až po elektromobily, či elektrické lokomotivy. Život bez nich si už neumíme ani představit.

1.2 Cíl práce

Při výuce některých elektrotechnických předmětů, které nás provází celým studiem (např. Základy elektrotechniky, Automatizační technika), tvoří značnou část probírané látky právě elektromotory. Při výuce jsou používány pouze obrázky nebo jednoduché animace, které v mnoha případech nejsou přiliš názorné.

Cílem mojí práce je návrh a vytvoření aplikace pro studijní účely, která bude srozumitelně ukazovat, jak jednotlivé části motorů pracují. Bude umožňovat demontáž jednotlivých částí motoru (i za chodu) a zobrazení jejich funkčnosti. Její následné využití bude při výuce výše zmíněných předmětů.

Aplikace bude vytvořená v programu Blender, za použití Blender Game Enginu a bude zobrazovat modely stejnosměrného a střídavého asynchronního motoru. U stejnosměrného motoru bude možná změna vstupního napětí, čímž se bude řídit počet otáček. Asynchronní motor bude řízen změnou počtu pólových dvojic, tedy vinutím na statoru motoru.

2 Teoretický základ

2.1 3D grafika

3D grafika je příbuzná vektorové 2D grafice. Také pracuje se souřadnicemi bodů a informacemi o úsečkách, křivkách a plochách, ale data jsou uložena ve trojrozměrném souřadnicovém systému. Z těchto trojrozměrných dat reprezentujících tělesa je potom renderován 2D obrázek. Rendering je tvorba reálného obrazu na základě počítačového modelu, nejčastěji 3D. Rendering obsahuje v závislosti na softwaru mnoho parametrů a nastavení, kterými lze ovlivnit konečný vzhled scény.Jde o odvětví počítačové grafiky. Zabývá se tvorbou obrazů, napodobující reálný svět. Je to zvláštní způsob vizualizace dat. Data jsou parametry popisující reálný nebo imaginární svět s jeho objekty a jejich vlastnostmi. Úkolem syntézy obrazu je z tohoto počítačového modelu vytvořit obraz, který je pokud možno nerozeznatelný od fotografie definovaného objektu v reálném světě. Syntéza obrazu je odvětví, které významně ovlivňuje tvorbu počítačových her, programů pro tvorbu grafiky a animaci a programů CAD/-CAM. Realistické počítačové obrazy nacházejí uplatnění při tvorbě filmových efektů, architektuře, vojenství, při simulaci fyzikálních jevů nebo projektech pracující s virtuální realitou. Různými technikami se dají ve 3D grafice vytvořit velmi realisticky vypadající obrázky díky věrné simulaci světelných a optických jevů jako jsou stíny, odrazy, lom světla či kaustika. Pokročilé vývojové nástroje umožňují i realistické animace včetně pohybů oděvu, vlasů, vodní hladiny a simulace fyzikálních jevů jako je gravitace a odrazy (Počítačová grafika, 2015).

2.2 3D souřadný systém

V Blenderu je každý objekt znázorněn pomocí třech koordinátů a tím je umístěn v prosotru. Jednotlivé souřadnice leží na 3 navzájem kolmých přímkách, pojmenovaných X,Y,Z. Hodnota jednotlivých souřadnic udává vzdálenost objektu od počátku neboli průsečíků všech tří os, tedy bodu [0,0,0]. Globální systém je vždy jediný pro celou scénu. V souřadném systému v programu Blender udává X-ová přímka (červená) šířku, Y-ová (zelená) hloubku a Z-ová (modrá) výšku objektu. Kromě globálního souřadného systému můžeme souřadný systém přepnout i na lokální, kde se zároveň při rotaci objektů rotuje i se samotnými osami. V kombinaci s různými možnostmi nastavení fyzického typu objektů (například použitím typu charakter), dokážeme díky lokálním souřadnicím udržet směr pohybu vpřed (Blender dokumentace, 2016).

2.3 3D modelování

Pojmem 3D modelování se rozumí proces tvarování a vytváření 3D modelu, který může být reprezentován několika způsoby. Modely mohou být vytvořeny na počítači člověkem pomocí modelovacího nástroje, podle dat získaných měřicím přístrojem

z reálného světa nebo na základě počítačové simulace (Počítačová 3D grafika, 2015). Při vytváření libovolné 3D scény se prochází čtyřmi kroky (3D počítačová grafika, 2016):

1. Modelování
2. Osvětlení
3. Vytváření materiálů
4. Renderování

Modelování je prvním krokem, jehož výstupem je 3D model, který však ještě není otexturován ani nasvícen. Každý model se skládá z těchto částí:

- Bod
- Hrana
- Plocha

Bod, neboli vertex je bodem v prostoru (souřadného systému), z něhož se následně skládají další prvky. Dva spojené vertex dávají hranu, více hran plochu. Změnou jejich hodnot (velikost, rotace, posun) postupně vytváříme model, podle našich představ.

2.4 Reprezentace těles

Reprezentaci těles můžeme rozdělit na objemovou, tahovou a povrchovou. U objemové (výčtové) se prostor skládá z jednotlivých buněk, nazývané voxely. Samotná reprezentace je dána právě obsazeností voxelů, kvůli čemuž není příliš přesná a používá se hlavně jako pomocná struktura. Při tahové reprezentaci těles (šablonování) se nové objekty vytváří při pohybu (tažení) původního objektu po trajektorii popsanou určitou funkcí, kde zároveň dochází k transformaci (Modelování pevných těles, 2016).

Povrchová reprezentace se dále dělí na několik podtypů (Hranové reprezentace, 2016):

1. Hranová reprezentace (drátový model), kde se zobrazují pouze vrcholy a hrany.
2. Plošková reprezentace, rozšíření hranové reprezentace o plochy (stěny).
3. Strukturovaná plošková reprezentace, někdy nazývaná okřídlená hrana. Těleso je tvořeno třemi seznamy, na nejnižší úrovni je seznam vrcholů, na střední úrovni seznam okřídlených hran (uloženy informace o okolních elementech) a na nejvyšší úrovni seznam ploch.

2.5 Vývoj Blenderu

Historie

Začátky Blenderu můžeme datovat do roku 1988, kdy Ton Roosendaal založil animační studio NeoGeo. V roce 1995 se rozhodl, že je potřeba nová sada 3D nástrojů pro animaci a začal vyvíjet software, který je dnes známý pod názvem Blender. O dva roky později založil další firmu, kterou pojmenoval NaN (Not a Number), která se zabývala rozvojem Blenderu. I přes původní úspěchy se tento projekt nepodařilo udržet a tak Ton Roosendaal v roce 2002 založil nekomerční organizaci Blender Foundation, jejíž hlavním úkolem bylo rozvíjet Blender jako open source. K tomu však bylo zapotřebí odkoupit původní zdrojový kód od investorů společnosti NaN. To se povedlo během prvních 7 týdnů pomocí sbírky a kampaně „Free Blender“ ”. Od té doby se Blender šíří jako open source software (Blender dokumentace, 2015).

Blender dnes

Blender je open-source software pro modelování a vykreslování 3D počítačové grafiky a animací s využitím různých technik (např. sledování paprsku, radiosita, scanline rendering, GI). Vlastní interface je vykreslován pomocí knihovny OpenGL, která umožňuje nejen hardwarovou akceleraci vykreslování 2D a 3D objektů, ale především snadnou přenositelnost na všechny podporované platformy (např. GNU/Linux, Microsoft Windows, Mac OS X...) (Software pro tvorbu 3D modelu, 2015).

2.6 Free software

Podmínky „free softwaru“ dané Free Software Foundation (zakladatele GNU Projectu a tvůrce GNU General Public License) je myšleno „free jako svobodný“ spíše než „zdarma“. Free software je v tomto smyslu software, který je volně k užití, kopírování, modifikování, redistribuování a to bez omezení. Opakem tohoto, je licencování většiny komerčních softwarových balíčků, které vám dovolí nahrát software jen na jeden počítač, neumožňuje si vytvářet kopie a nikdy neuvidíte zdrojový kod (Blender dokumentace, 2015).

2.7 Blender a konkurence

Při vytváření 3D modelů a scén je na výběr z velkého množství softwaru, který můžeme používat. Od placených, profesionálních aplikací až po ty jednodušší, pro nenáročné uživatele. Z této nepřeberné škály programů jsem se rozhodl vybrat zástupce, kteří patří mezi nejpoužívanější v různých oborech, kde se počítačové 3D grafika používá.

3ds Max

Tento software vyvíjený firmou Autodesk je světově nejrozšířenější software pro tvorbu vizuálních efektů, počítačových her nebo animací postav. Byl použit v řadě úspěšných hollywoodských filmů, jako například: Lara Croft: Tomb Raider, Alice In Wonderland, Star Wars, Avatar a mnoho dalších. Je zde také možnost již tak pokročilý software dále rozšiřovat pomocí knihoven, modulů a doplňků (CAD studio, 2016). Cena licence je hodně závislá na verzi a době podpory, kterou si s ní zakoupíme. Např. nejnovější verze (2016) se pohybuje okolo 6000 Kč/měsíc.Vzhledem k tomu, že 3ds Max je šířen pouze jako placená licence, používá se především ve větších profesionálních studiích. Hlavním kladem je pak stálá podpora vývojářů a rychlé odstranění chyb, které s sebou každá nová verze přináší (3ds Max, 2016).

Cinema 4D

Stejně jako 3ds Max se jedná o komerční software šířený pouze pod placenou licencí. Oproti 3ds Max má jednodušší a pro mnohé uživatele přijatelnější rozhraní. Cena licence se liší, podle verze, která se pořídí. Výhodou je i to, že se dají následně dokoupit tzv. moduly a díky tomu si přesně zvolit, co se potřebuje a co ne. Naříklad modul Hair, pro tvorbu vlasů (Cinema4D - Hair, 2016), MOCCA modul, pro tvorbu 3D postav a jejich animaci a mnoho dalších. Na výběr je několik řešení, nejlevnější: Cinema 4D Prime stojí kolem 12000 Kč oproti tomu kompletní balík nástrojů, pojmenovaný Cinema 4D Studio má cenu až přes 80000 Kč. Díky tomu, že se jedná o placený software je zde opět ze strany vývojářů zaručená podpora (Cinema4D, 2016).

Rhinoceros 3D

Další zástupce komerčního 3D softwaru, tentokrát od společnosti Robert McNeel & Associates, vyvíjený od roku 1992, původně jako nadstavba AutoCadu. Cena je závislá na licenci a počtu doplňků např. Brazil - renderovací engine (Rhinoceros 3D - Brazil, 2016) nebo Penguin - stylizace (Rhinoceros 3D - Penguin, 2016), které si lze se softwarem objednat, samotný program pro komerční využití stojí kolem 20000 Kč. Cena klesá, při využití slevy pro učitele a studenty, zhruba na 5000 Kč nebo při přechodu z předešlé verze. Oproti výše zmíněným, plně profesionálním programům, má omezenější prostředky při modelování i tvorbě animací, což nemusí být nutně na škodu. Díky těmto vlastnostem je více přehledný pro začínající uživatele a studenty a v kombinaci s možností zakoupení školní a studentské licence se stává kvalitním řešením pro školy a studenty za rozumnou cenu (Rhinoceros 3D, 2016).

SketchUp

Původním vývojářem tohoto softwaru byla firma @Last Software, v roce 2006 ale došlo k akvizici této společnosti Googlem a dále byl SketchUp vyvíjen jako plugin

do aplikace Google Earth. V roce 2012 SketchUp odkoupila další firma, Trimble Navigation, která ho vyvíjí dodnes. Motto tohoto programu je: „snadno a rychle“. Vývojáří tvrdí, že práce s tímto programem je snadná jako práce s tužkou a papírem (SketchUp - Historie, 2016). SketchUp sice nabízí možnost vytváření animací ale oproti aplikacím, které jsem zmiňoval výše je jeho primární použití jiné. Díky jeho jednoduchosti je oblíben hlavně u architektů, pro tvorbu prezentací svých projektů nebo inženýrů a kontruktérů pro tvorbu modelů do 3D tiskáren. Dnes jsou dostupné 2 verze, SketchUp Make, která je zcela zdarma a SketchUp Pro, která je rozšířena o exportování modelu do více formátů, možnost tvorby animací ve vyšším rozlišení nebo třeba podporu přes email. V neposlední řadě bych zmínil, že tento program dodnes těží z toho, že byl několik let vyvíjen takovým gigantem, jako je Google a je s jeho službami dost propojený. Nejvíce patrné bude spojení na Google Maps a Google Earth, což vysvětluje jeho oblíbnost právě u architektů (SketchUp, 2016).

Blender

Blender je rozumným kompromisem mezi těmito dvěma extrémy. Jako freeware je zdarma, má obrovskou komunitu uživatelů, kteří ho neustále posouvají dál a zároveň solidní podporu od vývojářů. Součástí Blenderu je GameEngine, ve kterém se dají vytvářet interaktivní aplikace, je možné dopsání vlastních skriptů v jazyce Python nebo stažení dalších rozšíření ve formě knihoven (tzv. addonů)(Blender dokumentace, 2015).

Shrnutí

V této kapitole jsem zmínil několik programů, které se často používají při tvorbě 3D modelů a animací v různých pracovních oborech. Snažil jsem se do ní zahrnout jak složitější, profesionální, placený software, tak freeware, který se v mnohem případech hodně liší. Tím jsem chtěl ukázat, jak široká nabídka v této oblasti je a dost záleží na preferencích samotného vývojáře, v čem se mu konkrétní projekt bude vytvářet nejlépe. Například při vytváření interaktivní aplikace je zapotřebí kromě samotného programu pro vytváření modelů i engine(jádro), na kterém aplikace poběží. Někteří výrobci poskytují kromě možnosti exportu modelů pro jejich použití v enginech třetích stran i svůj vlastní, například Autodesk nabízí engine pojmenovaný Stingray (AutoDesk - Stingray, 2016) a Blender Blender Game Engine.

Program	**Vývojář**	**Cena**	**Vlastní engine**
3ds Max	AutoDesk	6 000 Kč/měsíc	Stingray
Cinema4D	Maxon	12000 Kč	-
Rhinoceros 3D	Robert McNeel & Associates	20000 Kč	-
SketchUp	Trimble Navigation	freeware	-
Blender	Blender Foundation	freeware	BGE

2.8 Popis Blenderu

Uživatelské rozhraní Blenderu pracuje s tzv. editory, kde každý z nich slouží k zobrazování a práci s jinými daty. Například editor Timeline slouží pro práci s jednoduchými animacemi, Text Editor pro psaní textových souborů (nejvíce používané pro skripty) nebo 3D View pro 3D zobrazení modelu. Pro urychlení práce může být v jeden okamžik zobrazeno více editorů a toto nastavení rozložení obrazovky může být uloženo pro pozdější použití jako tzv. layout. K dispozici je od vývojářů předpřipravených 9 layoutů, které jsou vhodné k určité činnosti ale je možné je měnit podle svým vlastních preferencí (Blender dokumentace - obrazovky, 2016).

Původní layouty

- 3D View Full
- Animation
- Compositing
- Default
- Game Logic
- Motion Tracking
- Scripting
- UV Editing
- Video Editing

2.9 Editory

3D View

Tento editor se používá pro práci se samotným 3D objektem. Slouží například k přepínání jednotlivých módů, které určují, jak s objektem pracuje (objektový, editační, tzv. „sculpting“ a další), změně vykreslování modelu objektu (drátový, plošková, texturovaný, materiál), zobrazování pouze vybraných vrstev scény, změně manipulátorů (posun, rotace, zvětšení). V tomto editoru se provádí i samotná úprava objektů nebo jeho částí (pozice, rotace, barva, přiřazení materiálů). (Blender dokumentace - 3DView, 2016).

Timeline

Timeline editor slouží hlavně ke kontrole nebo tvorbě jednoduchých animací. Zobrazuje časovou přímku, na kterou lze vkládat klíčové snímky, kdy dojde ke změně konkrétníího atributu objektu v animaci (př. změna pozice objektu na snímku 50).

Také slouží k nastavení počátečního a koncového snímku animace a pro její přehrání (Blender dokumentace - Timeline, 2016).

Dope Sheet

Dope Sheet je využiván hlavně tehdy, pokud je zapotřebí spravovat více animací najednou. Obsahuje podrobnější informace o animaci (pojmenování, přehledně zobrazené údaje o všech měněných atributech objektů), umožňuje také změnit více klíčových snímků zároveň nebo vkládat značky, které označují důležitá místa v animaci (Blender dokumentace - Dope Sheet, 2016).

Logic Editor

Logic editor je nástroj, pro tvorbu „logiky hry“ v Blenderu a jeho Game Enginu. Probíhá zde nastavení vlastností zvolených objektů nebo změna jejich chování. Tento editor se skládá ze třech menších součástí tzv. „logic bricks“.

- Senzor
- Controller
- Aktuátor

Interakce v aplikaci se získá kombinací všech těchto prvků. **Senzor** čeká, dokud nepřijde akce, která spustí novou část programu. Typickým senzorem může být například standardní vstup z klávesnice, myši nebo změna proměnné. Dalším blokem je Controller neboli **logický člen**, který slouží k seskupování více senzorů a jejich následném vyhodnocení. Jeho výstupem je logická hodnota pravda/nepravda podle pravdivostních tabulek jednotlivých členů (AND, OR, NAND, NOR, XOR,..). Mimo klasických logických spojek je zde i možnost Python, tzn. napsání vlastního kontroleru pomocí skriptu v jazyce Python. Pokud se logický člen vyhodnotí jako pravda, tzn. senzory zaznamenaly akci, která má vést ke změně chodu v pogramu, pak se volá poslední součást, tzv. aktuátor neboli **akční člen**. Ten už provede samotnou změnu, například nastavení viditelnosti nebo pohyb objektu, přehrání zvuku, spuštění animace, nastavení kamery nebo změna scény (Blender dokumentace - Logic Editor, 2016).

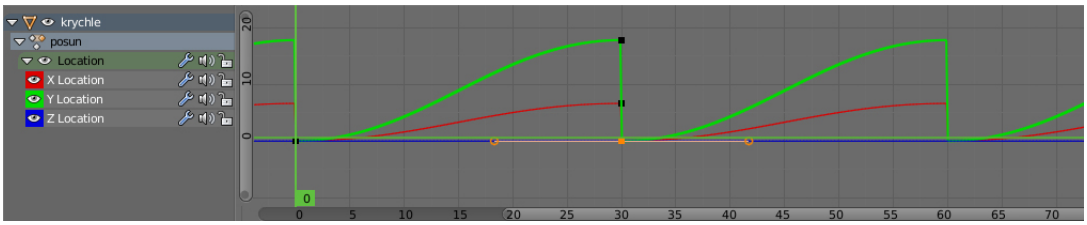
Textový editor

Textový editor slouží k vytváření textových dokumentů v prostředí Blenderu. Nejčastější využití tohoto editoru je psaní vlastních skriptů. Při vytváření nového dokumentu lze použít nabízenou možnost vytvoření dokumentu ze šablony, čímž uživateli dost usnadňuje práci. Ten totiž nemusí začátek skriptu, který bývá pro většinu z nich totožný (importování blender game enginu api do skriptu, nastavení hodnot sensorů)

psát neustále dokola. Editor také nabízí možnost pro zvýraznění syntaxe nebo zobrazení čísel řádků, výhodné pro debugování napsaného kódu. (Blender dokumentace - Text Editor, 2016)

Graph editor

Graf editor je nejsilnější nástroj pro správu animací. Po otevření tohoto editoru se zobrazí na ose X osa časová (jednotlivé snímky) a na ose Y hodnoty proměných, které měníme, například při posunu objektu se na ose Y zobrazuje velikost jednotlivých os v souřadném systému (zde nazývané kanály). Možností je i přidání vlastních značek (tzv. markery) pro označení důležitých oblastí v animaci a jejich popsání pro zjednodušení další práce nebo práci v kolektivu. Graf editor obsahuje 2 módy, prvním je F-Curves, který umožňuje editovat jednotlivé kanály pomocí funkčních křivek, například jejich linearizaci nebo cyklení. Druhým módem je tzv. „Drivers“ mód, který dovoluje upravovat animaci pomocí změny velikosti jednotlivých atributů (Blender dokumentace - Graph Editor, 2016).



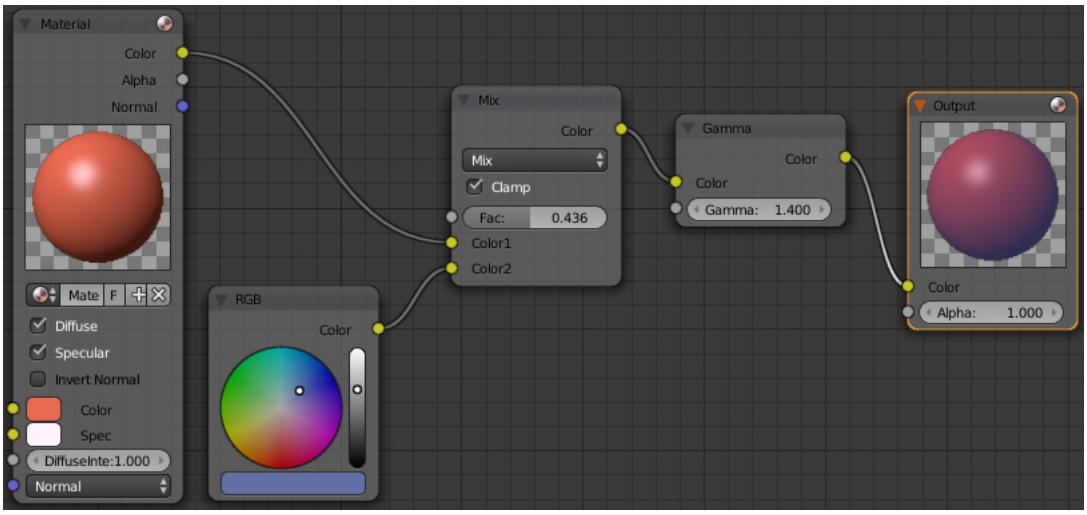
Obrázek 1: Graf editor zobrazující animaci posunu objektu krychle cyklenou po 30 snímcích.

UV/Image editor

Jak již samotný název napovídá, jedná se o editor pro správu obrázků. Používaný je zejména pro texturování modelů a UV mapování. Při procesu texturování lze využít možností nanášení textur (tzv. mapování) pomocí předem vytvořených funkcí. UV mapování ovšem poskytuje naprostou kontrolu nad tím, kam přesně se 2D textura na 3D objekt vloží. Princip této techniky spočívá ve správném rozložení vymodelovaného objektu na jednotlivé plošky nebo větší části, které se mají texturovat, této části se říká unwrapping neboli rozbalení modelu. Jednotlivé plochy a hrany se označí za důležité (Blender funkce Mark Seam) podle kterých se následně objekt rozloží. Při zobrazení v UV/Image editoru se automaticky zobrazí označený, již rozbalený objekt a lze přejít k poslední části, kterou je samotné texturování. UV/Image editor umožňuje takto rozbalený objekt vyexportovat jako obrázek v různých formátech, což značně usnadňuje tvorbu nebo úpravu samotné textury a její následné vložení zpět, kde přesně zapadne na správné místo. UV označení tedy slouží jako pomocný souřadný systém v 2D prostoru obrázku (Blender dokumentace - UV/Image Editor, 2016).

Node editor

Tento nástroj se používá k vytváření materiálů. Celý editor pracuje s uzly, které se řetězí za sebe a tím vzniká výsledný materiál. Jednotlivé uzly mohou být například **vstup**: RGB barva (popis barvy pomocí červené (R = red), zelené (G = green) a modré (B = blue) složky barvy), již vytvořený materiál nebo třeba textura, dále **barva** což je uzel sloužící pro správu barvy, například mixování více barev nebo změna gamma korekce, další možností je **vektor**, který slouží hlavně k mapování textur (volba geometrie objektu) různé druhy **konvertorů** a samotný **výstup** materiálu (Blender dokumentace - Node Editor, 2016).



Obrázek 2: Vytvoření jednoduchého metalicého materiálu v node editoru.

Python console

Díky tomuto nástroji a modulu bpy (zkratka pro Blender Python modul) lze Blender ovládat pomocí příkazové řádky konzole. Nemusí se tedy využívat jenom grafické rohraní ale pomocí příkazů vytvářet a modifikovat objekty nebo volat modifikátory. Například příkaz v konzoli: „bpy.context.object.location.xyz = (1, 2, 3)“ změní lokaci objektu na souřadnice X=1, Y=2, Z=3 v souřadném systému (Blender dokumentace - Python console, 2016).

Properties editor

Tento editor slouží k nastavení jednotlivých vlastností projektu a je tvořen několika záložkami. **Render** umožňuje nastavit požadavky na výsledné renderování. Při použití Blender Game Enginu to znamená hlavně nastavení rozlišení pro naši výslednou aplikaci či volba pro fullscreen režim, změna stínování. Přes záložku **Scene** lze spravovat vlastnosti týkající se scény, například jednotky, ve kterých se pracuje, správa barev nebo přepínání jednotlivých vrstev scény. Další sekcí je **World**, kde se

nastavuje okolí (svět), například barva horizontu, možnost přidání efektu mlhy nebo síla gravitace. Další možností je **Object**, kde se spravují vlastnosti objektů. Jejich pojmenování, transformace, umístění do vrstvy scény. **Modifikátory** pro přidání nového modifikátoru a jeho nastavení. **Materiály** a **Textury** pro správu parametrů vzhledu objektu a některých jeho vlastností, jako barva objektu, barevný odraz světla, velikost třecí síly materiálu nebo jeho elastičnosti. **Physics** záložka umožňuje nastavit fyzikální vlastnosti objektu, zdali bude ve scéně pasivní, statický nebo bude dynamický, bude to pohybující se charakter, který má specifické nastavení (možnost nastavit výšku kroku, pro překonávání objektů, které se uplatní například na schodech nebo nastavení síly skoku a rychlosti v pádu (v kombinací s gravitací nastavenou pro celý svět)) (Blender dokumentace - Properties editor, 2016).

Další

Je zde celá řada dalších editorů, pro správu videí, klipů, výpis z konzole, prohlížeč souborů pro ukládání/otevírání souborů nebo NLA (Non-Linear Action) editor pro efektivní správu akcí v animaci.

2.10 Modelování v Blenderu

V Blenderu jsou dva hlavní módy, ve kterých se dá pracovat a přepínat mezi nimi. Objektový, ve kterém se manipuluje s celými objekty (kamerami, světly, křivkami, textem,...) a editační neboli podobjektovým, kde se spravují jednotlivé části objektů (body, hrany, plochy). Celý proces modelování je v Blenderu usnadněn pomocí funkcí, které se nazvývají modifikátory. Pomáhají vytvářet modely, které by byly příliš náročné při manuální úpravě ojektu. Existuje jich celá řada, od booleanovských operací (průnik, součet, rozdíl objektů), přes triangulaci (převedení objektů na trojúhelníky) až po kolizi (simulace kolizí objektů) a jsou rozděleny do 4 sekcí (Blender dokumentace - modifikátory, 2016):

1. Úprava
2. Vytváření
3. Deformace
4. Simulace

2.11 Engine

V Blenderu jsou k dispozici celkem 3 enginy:

- Blender Render
- Cycles Render
- Blender Game

Konvenční

První dva jmenované, tedy: **Blender Render** a **Cycles Render** jsou konvenční. Používají se pro tvorbu 2D obrázků nebo animací. Určují hlavní prvky scény, jako zpracování světel, kamer nebo materiálů. Od původního Blender Render Enginu, se dnes již upouští, jeho oficiální vývoj je zastaven ovšem stále se používá, zejména pro renderování základních 2D obrázků (Blender dokumentace - Blender Render Engine, 2016). Druhý jmenovaný, Cycles Render Engine má mnohem komplikovanější zpracování světel, z tohoto důvodu je náročnější na hardware (možnost renderování pomocí GPU) ale jeho výsledky bývají mnohem realističtější (Blender dokumentace - Cycles Render Engine, 2016).

Herní

Poslední možností je **Blender Game Engine** (BGE), což je nástroj, pro tvorbu real-time projektů, interaktivních aplikací nebo her. Byl napsán v jazyce C++ a nabízí širokou škálu funkcí. Hlavní rozdíl mezi BGE a konvenčními enginy je způsob zpracování obrazů a animací. Při použití konvenčních enginů nemohou být ani obrazy, ani animace po renderování dále upravovány. Blender Game Engine ovšem renderuje scénu kontinuálně (real-time), díky čemuž je možné zajistit interakci s uživatelem a dynamičnost v aplikaci. Základem práce v tomto enginu je tzv. Logický editor (Blender dokumentace - Blender Game Engine, 2016).

Externí

Je i možnost používat Blender pouze k modelování objektů ale poté pro tvorbu aplikace použít jiný engine, do kterého si modely lze importovat. Blender umí v něm vytvořené objekty exportovat do široké škály formátů. Od formátu .3ds, který vyvinula forma Autodesk, Inc. pro svůj výše zmíněný software 3D Studio Max, přes .fbx formát, který umí importovat například Unity Engine (Unity dokumentace, 2016) nebo hodně oblíbený Unreal Engine pro tvobu her (Unreal Engine dokumentace, 2016), až po univerzální .obj formát vyvinutý firmou Wavefront Technologies (Blender dokumentace - import/export, 2016).

2.12 Spustitelný program

Výsledná aplikace se dá uložit i jako spustitelný program, záleží na operačním systému, na kterém v aktuální okamžik Blender běží a podle něj se vytvoří konkrétní soubor. Vzhledem k tomu, že Blender je vyvíjen a publikován nejen pro MS Windows ale i Linux a Mac OS, jde vytořit aplikace pro všechny tyto platformy. Samotný export probíhá pomocí oficiálních addonů, které je nejdříve potřeba povolit v uživatelském nastavení Blenderu. Jmenují se **Game Engine Publishing** a **Game Engine Runtime**. První jmenovaný umožňuje uložit dodatečné informace o pub-

likaci, jako verzi aplikace nebo platformu, pro kterou je vyvíjena, druhý addon se stará přímo o export do spustitelné podoby.

2.13 Elektrické stroje

Popis

Elektrickým strojem v nejobecnějším smyslu rozumíme zařízení pro přeměnu energie. Elektrický stroj přeměňuje elektrickou enerigii působením elektromagnetické indukce. Zkoumání procesů přeměny elektrické energie je založeno na obecných zákonitostech změny elektromagnetického pole, jakožto zvláštní formy existence hmoty (Hradil F., Škyřík J., 1993).

Rozdělení strojů

Podle způsobu přeměny energie dělíme elektrické stroje na:

1. Generátory, které mění mechanickou energii na energii elektrickou.
2. Motory, které mění elektrickou energii na energii mechanickou.
3. Měniče, které mění jen parametry elektrické energie.

Podle druhu proudu jsou stroje na proud:

1. stejnosměrný
2. střídavý
   - jednofázový
   - vícefázový, nejčastěji trojfázový

Podle toho, zde má stroj pohyblivé části či ne, rozeznáváme:

1. elektrické stroje točivé
2. elektrické stroje netočivé

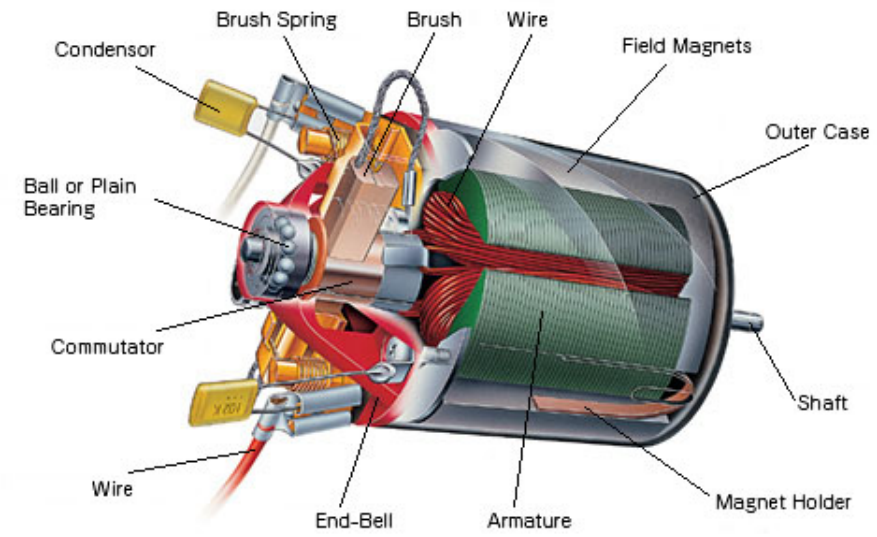
3 Metodika

3.1 Výběr vzorů

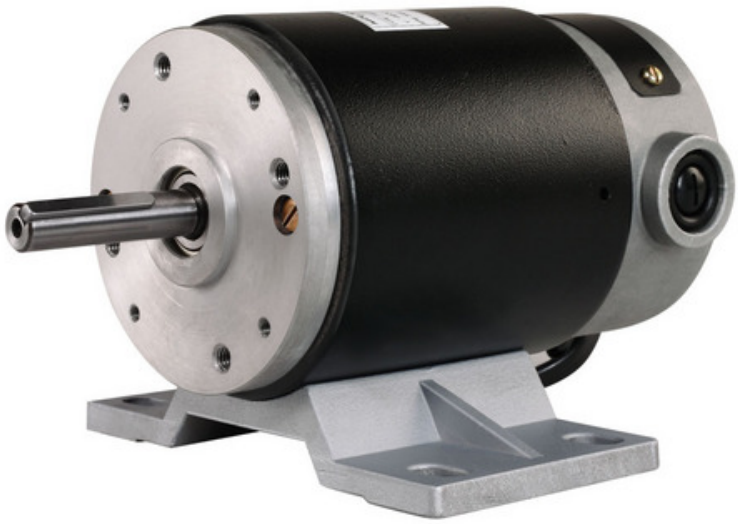
Aplikace bude zobrazovat 2 typy elektromotorů. První, DC motor (DC - direct current), tedy motor napájený pomocí stejnosměrného napětí. Druhým pak bude motor na střídavé napětí (AC - alternating current), které se dále dělí podle typu konstrukce na synchronní a asynchronní. Vzhledem k tomu, že asynchronní motor je díky svým vlastnostem, ke kterým patří například jednodušší konstrukce a s tím spojená i vyšší spolehlivost, levnější výrobní cena a snadná údržba, je nejvíce rozšířeným a využívaným motorem, bude aplikace zobrazovat právě tento typ motoru, který je někdy také nazývaný jako indukční.

Stejnosměrný

Jako vzor pro první část programu byl s přihlédnutím k faktu, že tato aplikace bude mít hlavně studijní charakter, zvolen konstrukčně nejjednodušší motor, tzv. „PMDC“ neboli Permanen Magnet DC Motor, tedy motor, který jako stator používá permanentní magnety, místo sekundárního vinutí. Tím se značně zvýší přehlednost tohoto modelu a pozornost studentů se bude upínat spíše k částem, které jsou pro funkci tohoto motoru elementární, jako například komutátor (součástka sloužící pro přepínání směru proudu tekoucí do rotoru). Aplikace bude umožňovat pomocí vlastního grafického uživatelského rozhraní změnu velikost vstupního napětí a tím měnit otáčky motoru, jeho úplné zastavení, demontáž a přepínání úhlů pohledů (kamer) scény (Basic Motor Types, 2016).



Obrázek 3: Zobrazení vnitřní konstrukce PMDC motoru (Electrical Motor Images, 2016).



Obrázek 4: Zobrazení DC motoru firmy Agni motors.

Střídavý asynchronní

Pro druhou část aplikace byl zvolen jako vzor třífázový indukční elektromotor s klecovou konstrukcí rotoru („kotva nakrátko“), protože je to jeho nejběžnější a nejpoužívanější konstrukce. Aplikace bude umožňovat nahlédnutí do části motoru nazývaná svorkovnice, kde se dá jednoduchým způsobem změnit zapojení motoru mezi stavy troúhelník/hvězda. Dále bude umožnovat změnu počtu pólových dvojic a tím řídit otáčky motoru, jeho úplné zastavení a demontáž (AC induction motors, 2016).

Obrázek 5: Zobrazení konstrukce AC motoru firmy Leeson Electric.

3.2 Uživatelské rozhraní

Každý z modelů bude mít kromě své vlastní scény, ve které bude umístěn i druhou, ve které se budou nacházet pouze prvky, pro jeho ovládání. Z důvodu odlišnosti ovládání obou motorů, bude muset mít každý model svoji vlastní a to i přesto, že část ovládání bude stejná. Společná část obou ovládacích scén bude čtveřice tlačítek pro změnu kamery, další dvojice tlačítek pro demontáž a sestavení motoru a tlačítko pro navrácení zpět do hlavní nabídky programu, tedy na výběr typu motoru. Část pro řízení počtu otáček bude u stejnosměrného motoru tvořena **scrollbarem**, v případě střídavého motoru dvěmi **radio buttony**. Prvky se vytvotvoří a upraví jako Mesh objekty, v případě tlačítek Plane, tedy plochy, radio buttony poté jako vyplněné kružnice. Jednodušší funkcionalita, kterou budou mít tlačítka pro změnu kamery, ukončení aplikace, permanentní překrytí pohledu kamery a scény s ovládáním, bude vytvořena pomocí logického editoru. Složitější, například ovládání vlastního scrollbaru, radio buttonů nebo spouštění a pozastavování animace pak budou obstarávat skripty napsané přímo v Blenderu a jeho Text editoru.

4 Vlastní práce

4.1 Scény

Tato aplikace se celkem skládá z pěti scén. První z nich je hlavní nabídka, kde si uživatel může vybrat typ motoru, který se mu má zobrazit. Obsahem dalších dvou scén je zobrazení samotných modelů motorů a v posledních 2 scénách se nachází uživatelské rozhraní pro jejich ovládání. Vzhledem k tomu, že má aplikace umožňovat různé pohledy na modely a jejich přepínání, řešením bylo vytvoření jedné scény pro každý model, která bude umožňovat jejich ovládání a její následné překrytí pro každou kameru na hlavní scéně.

Hlavní scéna

Tato jednoduchá scéna slouží pro výběr aplikace, která se má spustit a obsahuje pouze 3 tlačítka. Jedno tlačítko pro volbu stejnosměrného motoru, druhé pro střídavého a poslední, třetí tlačítko slouží pro ukončení celé aplikace.

Scéna stejnosměrného motoru

Scéna pro stejnosměrný motor zobrazuje čtyři kamery, mezi kterými se dá přepínat, dvě herní lampy (jedna směrem dolů, pro hlavní osvětlení, druhá směrem vzhůru, pro dojem odrazu plochy stolu), jednu bodovou lampu pro realističnost modelu (herní lampy nevrhají stíny, z toho důvodu se používá kombinace různých zdrojů světla, aby byla aplikace realističtější). Dále se v této scéně nachází plocha, která má představovat stůl, na kterém motor leží, a samotný motor. Ten se skládá z 9 částí. V kompletním sestaveném modelu může uživatel vidět: **plášť** motoru, který je tvořen 2 oddělenými částmi, **hřídel** motoru a dvě **ložiska**, která tyto dvě části spojuje. Uvnitř motoru lze pozorovat dva permanentní magnety, které tvoří **stator** motoru, **držák na uhlíkové kartáčky** pro správnou funkci **komutátoru**, **rotor s vinutím** a **větrák** pro chlazení motoru připevněný k hřídeli.

GUI pro stejnosměrný motor

Scéna obsahující 4 tlačítka pro změnu kamery, z toho poslední, čtvrtá možnost se povolí až po dostatečné demontáži motoru a slouží pro bližší pohled na komutátor. Dále 2 tlačítka pro odebírání jednotlivých částí motoru a jejich zpětnému navrácení, tlačítko START/STOP pro spuštění a vypínání motoru, které spouští a vypíná animaci hřídele a jejich potomků, což jsou rotor s vinutím, komutátor a větrák. Poté obsahuje vlastní „scrollbar“, kterým uživatel mění hodnoty vstupního napětí a tím i rychlost otáčení motoru (animace hřídele). Dalšími prvky GUI scény jsou 2 textové výstupy, které zobrazují číselné vyjádření vstupního napětí ve voltech a rychlost motoru uváděnou v otáčkách za minutu. Posledním prvkem je tlačítko pro vypnutí aplikace.

Scéna střídavého motoru

Scéna zobrazující střídavý motor, se skládá z odnímatelného **pláště** motoru, rozděleného na 2 části, **svorkovnice**, která slouží k přepínání zapojení motoru hvězda-trjúelník, **hřídele**, **statoru**, tentokrát v podobě jednotlivých statorových plíšků, s drážkami pro **statorové vinutí**, **rotor** v podobě klecové kotvy, **ložiska**, které spojují plášť a hřídel motoru a **větrák** pro jeho chlazení. Dále jsou k dispozici čtyři kamery, které slouží k přepínaní pohledů na motor pod různými úhly, dále tři lampy, z čehož dvě jsou herní a jedna bodová pro realistické stínování celé scény a plocha, která představuje stůl.

GUI pro střídavý motor

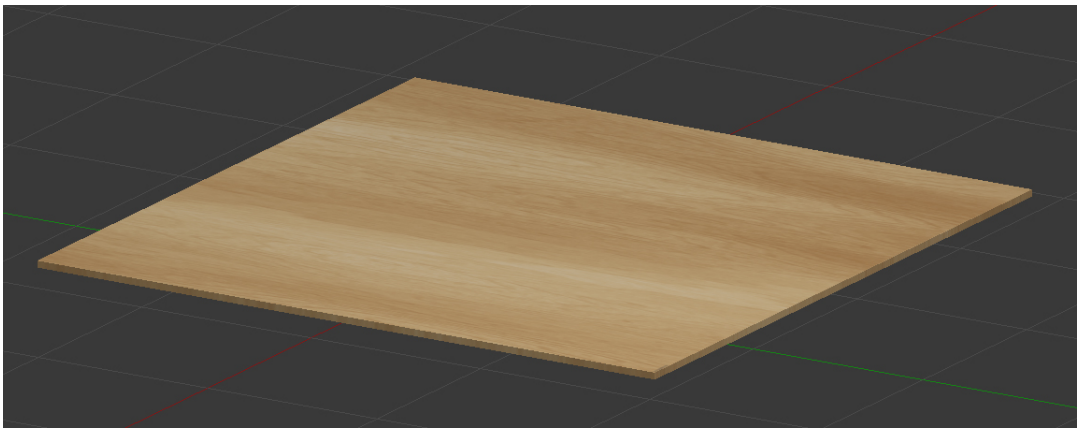
Tato scéna slouží k ovládání střídavého motoru. Obsahuje čtyři tlačítka pro ovládání všech kamer hlavní scény zobrazující střídavý motor, další dvě tlačítka pro rozebrání a seskládání zpět některých částí motoru, tlačítko START/STOP pro spuštění a vypnutí motoru, tedy animace otáčení hřídele, dále zde nalezneme dva vlastní „radio buttony“, pro přepínání počtu pólových dvojic a tedy ovládání rychlosti motoru a posledním prvkem této GUI scény je tlačítko pro vypnutí celé aplikace.

4.2 Stejnosměrný motor

Model desky stolu

Tento nejjednodušší model celé scény se vytvoří ze základního tvaru plochy. Přidání plochy do scény lze docílit pomocí klávesové zkratky SHIFT-A nebo zvolením v 3D View editoru funkce Add, z nabízených sekcí je plocha pod položkou Mesh, kde se nachází možnost Plane. Poté se musí nabízený tvar transformovat na požadovanou velikost. To lze docílit pomocí klávesové zkratky S v objektovém nebo editačním režimu 3D View editoru nebo pod položkou Object→Transform→Scale. Velikost objektu se mění táhnutím myši (směrem od středu objektu docílíme zvětšování, směrem k centru objektu změnšování). Pokud je zapotřebí transformovat objekt pouze v určitém směru, lze již při tranformaci zvolit osu, pro kterou má transformace platit (X, Y, Z) a objekt se bude transformovat jenom jejím směrem. Nyní se musí nastavit výška desky stolu, k tomu se nejvíce hodí použít modifikátor Solidify. V Properties editoru je možnost Modifikátory, jejíž podsekcí je přidání modifikátoru, a zvolení jeho typu. Po zvolení možnosti Solidify lze nastavit tloušťku objektu pod atributem Thickness, potvrzení modifikátoru probíhá po stisknutí volby Apply (Blender dokumentace - Solidify, 2016). Poslední částí je natexturování tohoto objektu. Přidání materiálu probíhá v Properties editoru pod položkou Material →New, kde si lze nový materiál pojmenovat, zvolit odrazivosti světla, jeho barvu a další parametry. Textura se nanáší ve své vlastní sekci: Texture, která se nachází hned vedle nabídky s materiály. Nabídkou Texture →New →Image →Open lze objekt otexturovat konkrétním obrázkem. V nabídce Mapping se ještě musí nastavit

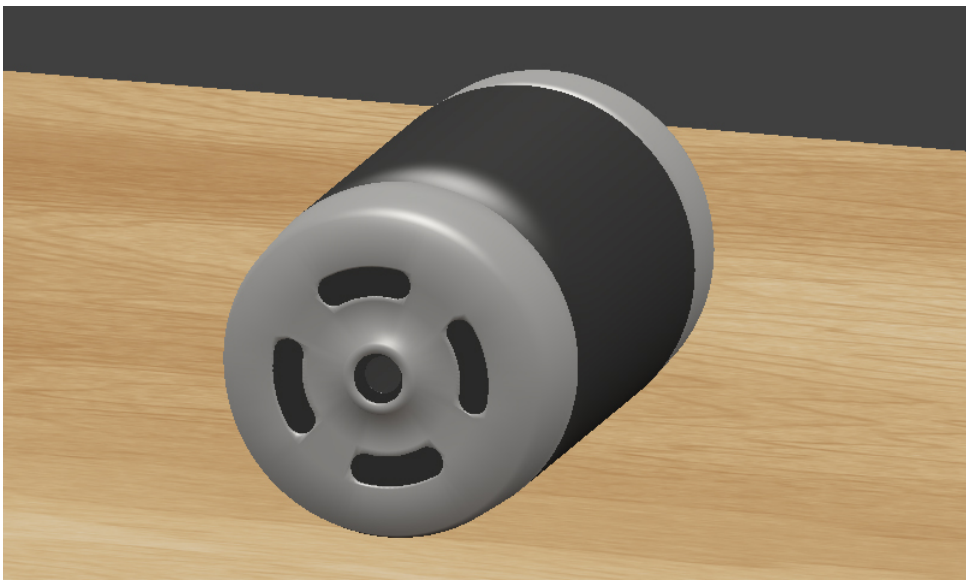
tvar objektu tak, aby byl vjem textury reálný. V takto jednoduchém tvaru objektu postačí základní Object metoda nanášení.



Obrázek 6: Ukázka objektu stolu.

Model pláště motoru

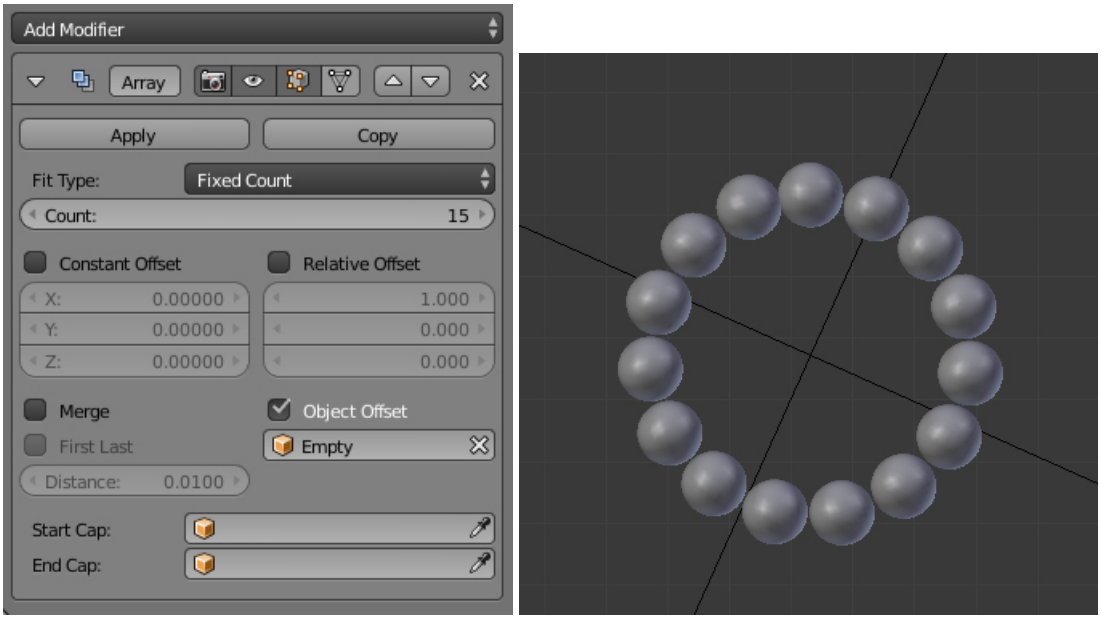
Tento model ponese základní tvar válce, který je přpravený v sekci Mesh. Při vytváření válce: SHIFT-A →Mesh →Cylinder, lze zvolit typ válce, který se má vytvořit. Možností Nothing se vytvoří válec, který má svoje podstavy nevyplněné. Objektu lze transformovat jeho velikost (viz výše) a také rotaci. K tomu slouží klávesová zkratka R při vybraném objektu. Stejně jako u velikosti i u rotace umožňuje vybrat pouze jednu z os souřadného systému, která se má do rotace aplikovat (X, Y, Z). Modifikátorem Solidify lze opět změnit tloušťku tohoto objektu tak, aby odpovídal realitě. Čela pláště motoru jsou tvořena dvěma kružnicemi. SHIFT-A →Mesh →Circle lze vytvořit práve kružnici. Přepnutím do editačního módu se zprástupňuje další možnost, Extude. Klávesová zkratka E nebo volba Mesh →Extrude slouží k vytažení ploch, hran či jednotlivých vertexů. Další důležitou funkcí pro vytváření objektů je Faces, klávesová zkratka F. Po označení hran tato funkce vytvoří plochu mezi nimi, v případě označení dvou vertexů se mezi nimi vytvoří hrana. Těmito funkcemi v kombinaci s hladkým stínováním a modifikátorem Subdivision Surface, pracující na principu rozdělování vytvořených ploch na menší úseky (libovolně volitelný parametr počtu nových ploch), který tedy slouží zejména pro vytváření hladkých ploch objektu (Blender dokumentace - Subdivision Surface, 2016) a posledním modifikátoru Boolean, který ze dvou objektů booleanovských operací sjednocení, průniku a rozdílu, vytvoří nový objekt lze vymodelovat celý plášť motoru (Blender dokumentace - Boolean, 2016).



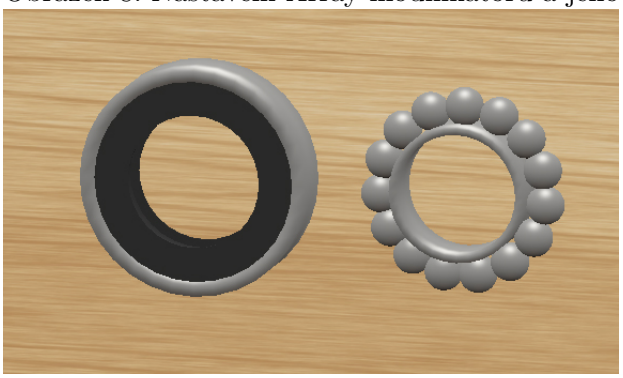
Obrázek 7: Ukázka objektu pláště stejnosměrného motoru.

Model ložiska

Většinu části ložiska vytvoříme pomocí akcí zmíněných výše, hlavní části tvoří dva krátké válce, čelní stěny ložiska jsou pak dvě plošky, oříznuté pomocí Boolean modifikátoru válcem o velikosti hřídele. Kuličky ložiska byly vytvořeny následovně. Vytvoření koule je možné přes SHIFT-A →Mesh →UV Sphere. Pomocí modifikátoru Array lze vytvořit pole stejných objektů, které vůči sobě budou rotovány kolem středu ložiska. Středem ložiska však musí být objekt, k tomuto účelu se nejvíce hodí Empty objekt, který se po vytvoření ložiska může vymazat. CTLR →Empty →Plain Axis tedy vytvoří prázdný objekt os, kteý bude vstupovat do Array modifikátoru aplikovaného na první objekt kuličky ložiska. Poté pomocí rotace Empty objektu modifikátor Array vytvoří předem daný počet kopií objektu, na který byl aplikován a posune je vůči sobě o velikost rotace, kterou mu udává Empty objekt(Blender dokumentace - Array, 2016).



Obrázek 8: Nastaveni Array modifikatoru a jeho aplikace.

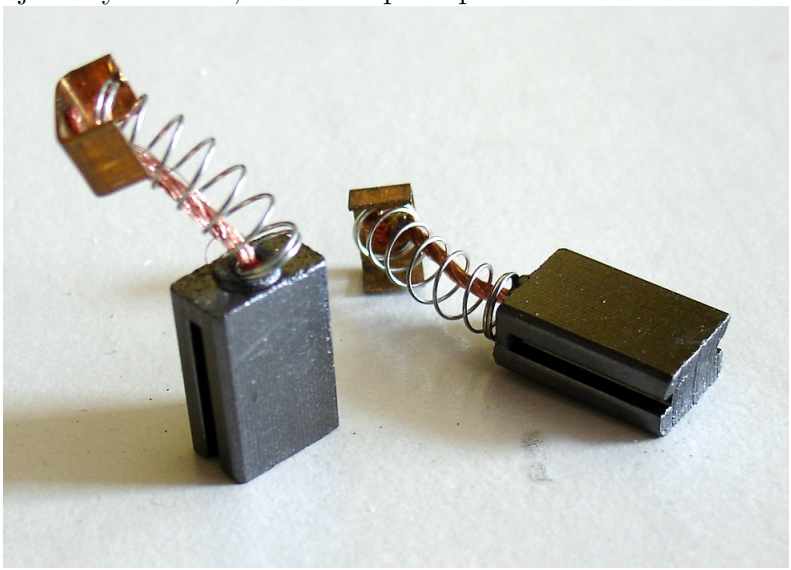


Obrázek 9: Celkové ložisko.

Model držáku s kartáčky

Z tvaru krychle pomocí funkcí Extrude a Faces, jde vytvořit základní tvar držáku na kartáčky. Poté je nutné vytvořit přívodní kabeláž, k tomuto učelu lze použít 2 křivky. Pomocí Bézierovy křivky (SHIFT-A →Curve →Bezier) a změny jejich řídících bodů lze upravovat její tvar do požadované podoby. Poté se vytvoří další křivka, tentokrát cyklická (SHIFT-A →Curve →Circle), která bude použita jako průmer kabeláže a naváže se na první křivku, která určí tvar jednotlivých kabelů. Navázání se provádí pomocí funkce data v Properties editoru v sekci Bevel Object. Aby se takto vytvořené objekty vykreslovaly (zatím to jsou stále křivky), musí se převést na mesh objekty. To lze udělat pomocí funkce Object →Convert to (klávesová zkratka ALT-C)→Mesh from Curve/Meta/Surf/Text.

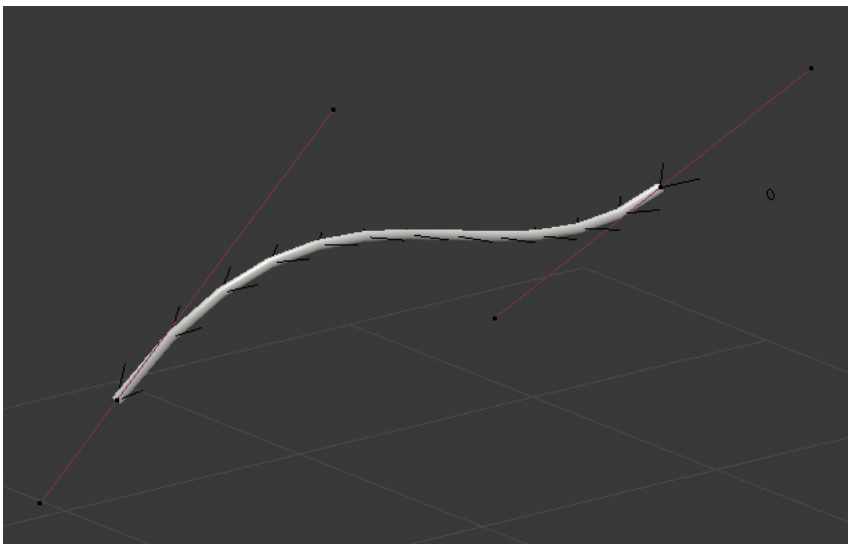
Další část jsou samotné uhlíkové kartáčky, které přivádí napětí na komutátor. Tato součástka je tvořena malou spirálou, která tlačí kartáčky ke komutátor a tím udržuje stálý kontakt, dokud se postupně neobrousí a nemusí se vyměnit.



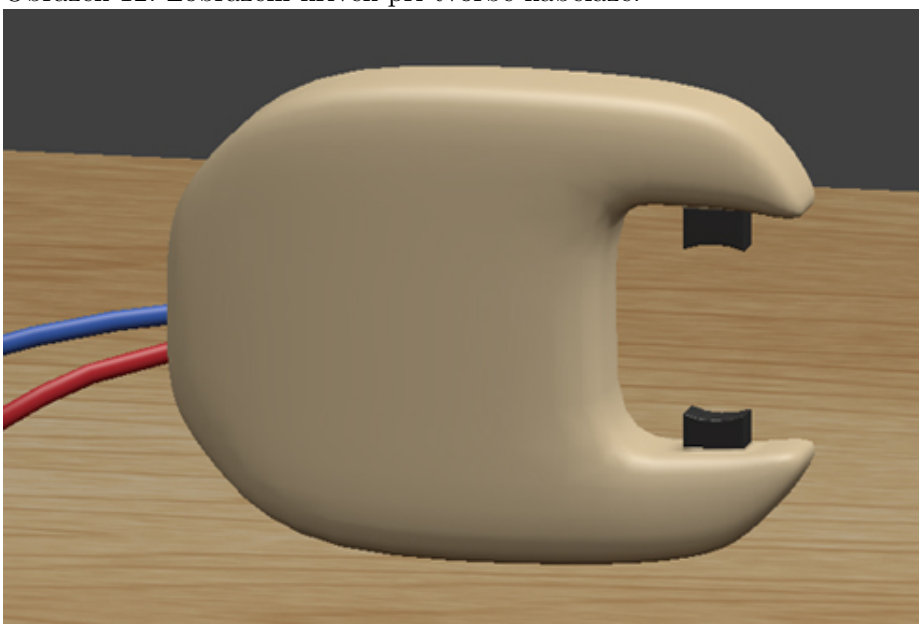
Obrázek 10: Reálné uhlíkové kartáčky.

Spirála jde v Blenderu vytvořit pomocí modifikátoru Screw. Ten se aplikuje na tvar kružnice a nastaví se osa, na které závity objektu vytvářet, kolik závitů se vytvoří nebo o úhel sklonu vzniklé spirály. Spojením takto vytvořené spirály se samotnými kartáčky lze uskutečnit pomocí funkce Join, která se nachází v sekci Object→Join(klávesová zkratka CTRL-J) (Blender dokumentace - Screw, 2016).

Obrázek 11: Model kartáčků.



Obrázek 12: Zobrazení křivek při tvorbě kabeláže.



Obrázek 13: Celkový držák s kabeláží a kartáčky uvnitř.

Model hřídele a rotoru

Hřídel je součást válcovitého tvaru, o různých průměrech, kterou lze vytvořit pomocí funkcí Extrude (E) a Scale (s) základního tvaru válce. Rotor stejnosměrného motoru se skládá z jednotlivých rotorových plíšků. Ty byly vytvořeny ze tvaru vyplněné kružnice, na kterou byl aplikován modifikátor Boolean, kterým se pomocí operace rozdílu a objektu ve tvaru kvádru odstranila vnitřní část rotoru. Dále byl použit modifikátor Solidify, kterým lze nastavit tloušťku objektu (jednoho rotorovoého plíšku) a následně byl použit poslední z modifikátorů, tentokrát Array, pro

vytvoření dalších kopií tohoto objektu, posunuté v požadovaném směru. Na rotoru se nachází také rotorové vinutí, tedy cívky, které byly vytvořeny pomocí křivek a jejich znásobením modifikátorem Array. Poslední část, která je připevněna k hřídeli je větrák.

Model komutátoru

Komutátor je součástka spojená s hřídelí, která se stará o správný tok elektrické energie do rotorového vinutí. Podle typu konstrukce se skláá z několika dotykových ploch. V programu Blender byla tato plocha vytvořena pomocí základního tvaru krychle, která se upravila funkcemi Extrude (E), Scale (S) a modifikátory Subdivison Surface pro hladký povrch, a Array, pro vytvoření kopií těchto ploch kolem osy hřídele.

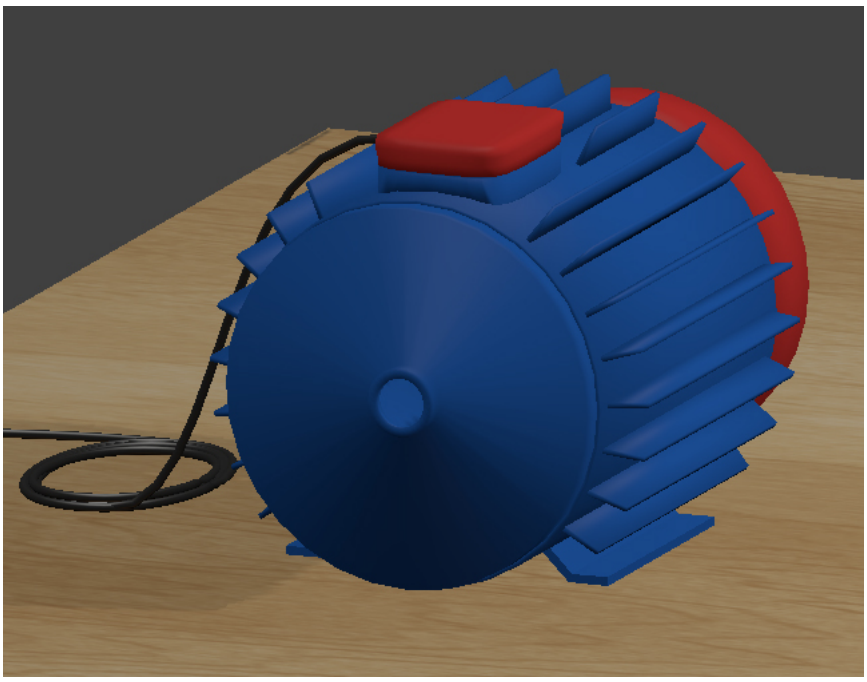
4.3 Střídavý motor

Společné prvky

Některé prvky jsou pro oba modely stejné. Například stejně jako v prvním případě stejnosměrného motoru, i tento motor bude ležet na modelu desky stolu. Blender umožňuje kopírování objektů jak mezi jednotlivými scénami, tak celými aplikacemi. To probíhá při otevření obou aplikací, pomocí klávesových zkratek CTLR-C (kopírování) CTRL-V(vložení). Dále lze objekt normálně upravovat a to jak ručně, tak i za pomoci modifikátorů. Kromě desky stolu je to i model ložiska, hřídel a větrák.

Model pláště motoru

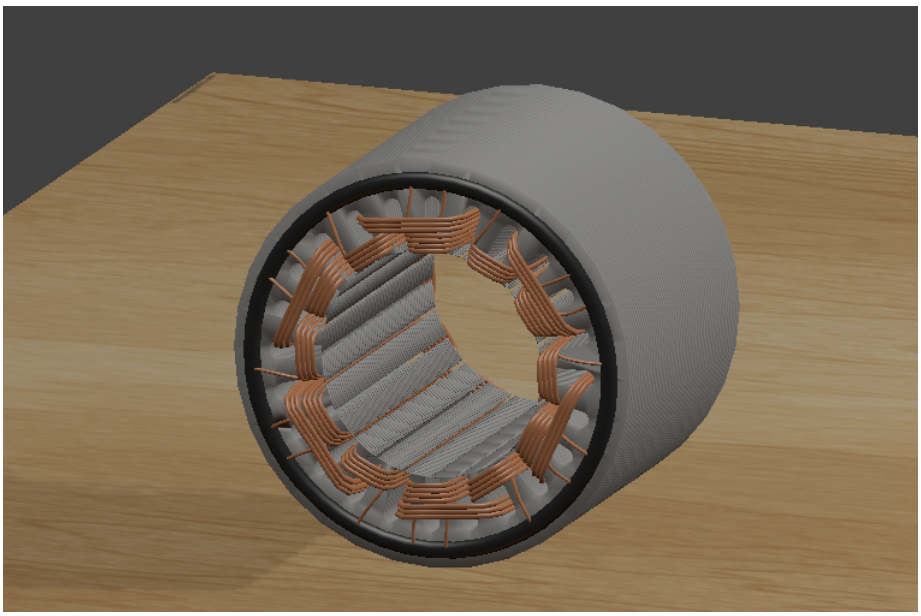
I v tomto případě, stejně jako u stejnosměrného motoru, bude prostřední část pláště tvořena válcem, který bude mít prázdné podstavy, krajní části (přední a zadní čelo) budou vytvořena z kružnice, která je upravena přes funkce Extrude (E), Scale (S) a modifikátory Subdivision Surface a Solidify. Rozdílem oproti pláště PMDC motoru je tzv. svorkovnice a žebrování, pro chlazení motoru. Žebrování lze vytvořit ze základního tvaru plochy (SHIFT-A →Plane). Té nastavíme tloušťku pomocí modifikátoru Solidify a druhým modifikátorem, Array, vytvoříme kopie tohoto jednoho žebra, konkrétně rotací kolem objektu prázné osy uprostřed modelu pláště motoru. Svorkovnice byla vytvořena jako kvádr, který byl tranformován na správnou velikost a místo, a spojen s celým páštěm funkcí Join.



Obrázek 14: Ukázka objektu pláště střídavého motoru.

Model statoru s vinutím

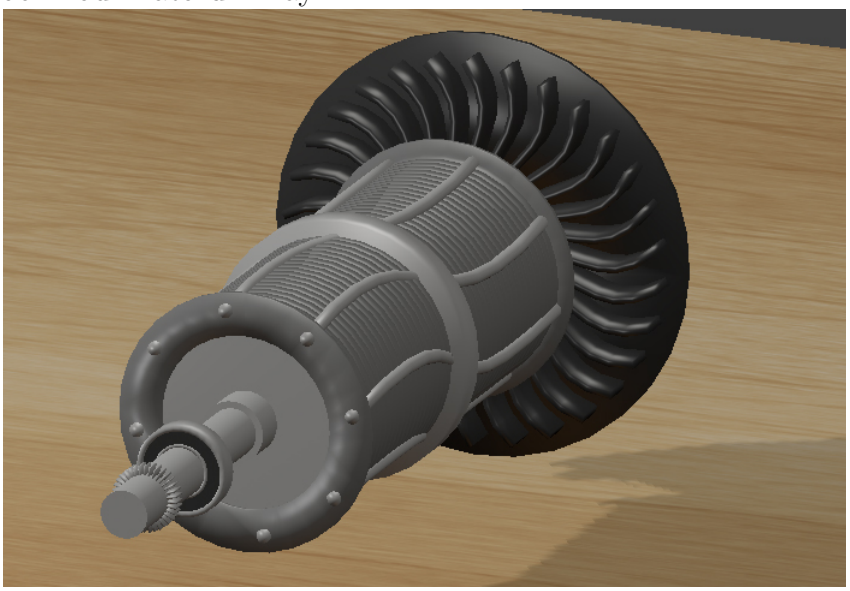
Stator motoru je v tomto případě tvořen statorovými plíšky. Nejefektivnější způsob je vytvořit jeden plíšek a poté pomocí modifikátoru Array provést jeho znásobení po určité ose. Samotný plíšek byl vytvořen z tvaru vyplněné kružnice, modifikátoru Solidify, pro nastavení jeho tloušťky a Boolean pro vyříznutí tvaru rotorové drážky uvnitř modelu. Na statoru se nachází 2 vinutí. První, tvořeno 6 cívkami (2 cívky = 1 pár pro každou z fází) a druhé, tvořeno 12 cívkami (4 cívky = 2 páry pro každou z fází). Samotné cívky byly vytvořeny pomocí kružnicových přímek.



Obrázek 15: Ukázka statoru s vinutím indukčního motoru.

Model rotoru

Tento model je tvořen tzv. „klecí“, která představuje kostru rotoru. Klec byla vymodelována pomocí kružnice, upravenou funkcemi pro transformaci velikosti a modifikátory Solidify a Boolean. Vodivé spojení bylo vytvořeno pomocí přímky a znásobené pomocí modifikátoru Array.



Obrázek 16: Model rotoru indukčního motoru.

4.4 Animace hřídele

Prvním důležitým krokem při vytváření animace motoru je spárovat všechny potřebné části. V případě PMDC motoru to jsou: větrák, rotor, rotorové vinutí, komutátor a samotná hřídel, pro střídavý motor pak: větrák, celá klec s rotorovými plechy a hřídel. Spárování probíhá pomocí funkce Parent (klávesová zkratka CTRL-P), kdy je prvně zapotřebí vybrat objekt, který má být potomkem a následně rodičovský objekt. Při takto spojených objektech postačí vytvořit pouze jednu animaci, v tomto případě hřídele, která je rodičovským objektem všech dalších částí a animace se přenese i na všechny ostatní potomky. Základní animaci se může vytvářet v několika editorech (3D View editor, Timeline editor, Dope Sheet editor, Graph edior). Na snímku 0 se zvolí první klíčový snímek přes funkci Insert KeyFrame (klávesová zkratka I), kde uživatel dostane na výběr, o jaký snímek se jedná (lokace, rotace, velikost). V případě, že uživatel animaci vytváří v 3D View editoru, se pomocí kurzorových kláves přesune na sníměk 50, pomocí funkce Rotate (R) se orotuje hřídel i se svými potomky o požadovaný úhel (360°) a v animaci se takto upravenému objektu opět uloží jeho souřadnice, pomocí funkce Insert KeyFrame. Další důležitá úprava této animace se odehrává v Graph Editoru, který nabízí průběh animace zlinearizovat. Tím docílíme toho, že na začátku animace se hřídel nebude zrychlovat a ke konci zase zpomalovat.

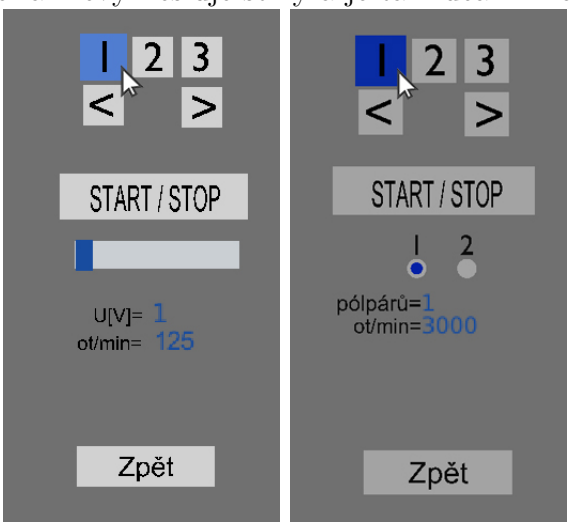
4.5 Materiály

Všem objektům ve scéně byly přidány materiály, nastavení barvy materiálu, odrazivost světla, správné stínování nebo průhlednost materiálu. Možnost přidání textur. Vlastní materiály mají i prvky ovládání, kde se musí nastavit tak, aby nezasahovaly do hlavní scény, kterou překrývají.

4.6 Ovládání

V této aplikaci je ovládání motoru členěno kompletně zvlášť. To hlavně z toho důvodu, že je na hlavní scéně více kamer a na každé z nich je možnost ovládání modelu povolena. Ovládání má tedy svoji vlastní scénu, která překrývá každou z kamer. Je nevhodné jednotlivé prvky grafického uživatelského rohzraní zobrazovat v perspektivě, proto je v této scéně pouze jedna ortografická kamera, směřující na čtveřici tlačítek pro přepínání pohledů kamer, další dvojici tlačítek pro rozebrání a skládání motoru, tlačítko pro spuštění animace, dvojici textových výstupů , v případě stejnosměrného motoru pro zobrazení otáček hřídele a vstupního napětí, střídavého motoru pro zobrazení počtu pólových dvojic a otáček, scrollbar pro změnu vstupního napětí PMDC motoru a dva radio buttony pro změnu počtu pólových dvojic indukčního motoru. Posledním prvkem je tlačítko „Zpět“ pro navrácení aplikace do fáze výběru typu motoru, tedy do hlavní nabídky. Samotné prvky jsou tvořeny jako Mesh objekty (plochy, vyplněné kružnice). Pro správné zobrazení prvků ovládání je

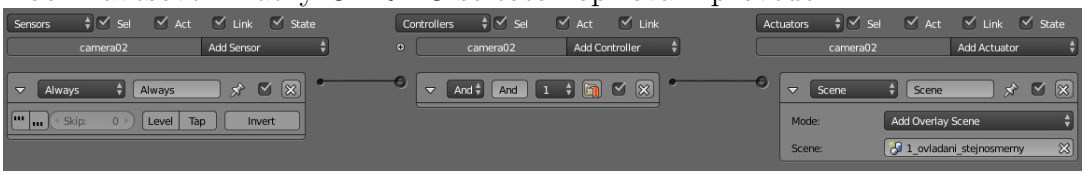
důležitá také volba správného materiálu, hlavně vlastnost „Shadeless“, diky které materiál nevykresluje stíny a je tak ideální hlavně pro tyto elementy.



Obrázek 17: Výřez scény s uživatelským ovládáním.

4.7 Logika hry

Vytváření interakce se provádí v editoru, který se nazývá Logic Editor. V této části se pracuje s 3 základními prkvy (senzory, logické členy, akční členy). Na hlavní scéně je tento editor použit pouze pro překrývní scén uživatelského rozhraní se samotným modelem a také pro změnu viditelnosti jednotlivých prvků. Jako senzor, do akce pro překrývání scén, vstupuje always, který je přes logický člen AND spojen se Scene aktuátorem, který překrývá kameru scénou s ovládáním (možnost Add Overlay Scene). Prvky: větrák, oba permanentní magnety, držák, ložiska a plášť poté mají jako senzor zprávu, kterou vysílá prvek ve scéně s ovládáním, o jejich viditelnosti. I tento senzor je propojen přes logický člen AND s aktuátorem, tentokrát Visibility, kterým je nastavena viditelnost objektu. Vzhledem k tomu, že mají stejný způsob zpracování (čekají na zprávu Show/Hide a podle toho změní svoji viditelnost), lze tento úsek zkopírovat do všech modelů. To se dělá tak, že se zvolí první objekt, od kterého se má kopírovat, pak druhý, do kterého se má obsah logického editoru kopírovat a pomocí klávesové zkratky CTRL-C se toto kopírování provede.



Obrázek 18: Ukázka logic editoru při překrývání kamery další scénou.

Ve scéně s ovládacím rozhraním se tento editor využívá více. Slouží například k interakci tlačítek pro přepínání kamer. To probíhá přes senzory mouseover (zjištění, zdali se ukazatel myši nachází na objektu tlačítka), které vyvolává animaci

pro zvýraznění objektu a usnadňuje orientaci uživatele, spolu se senzorem myši pro kliknutí na levé tlačíko myši pak vyvolá akci message, která zašle zprávu do hlavní scény na nastavení kamery.

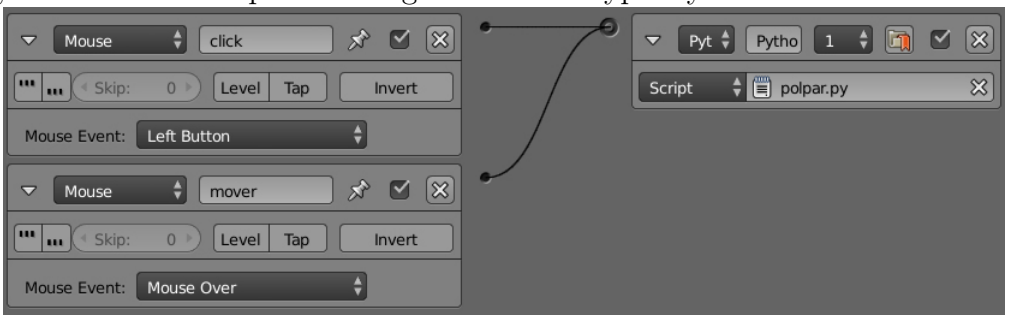
Logic editoru dále využívají i tlačítka na demontáž a sestavování motoru. Obsahují 2 proměnné, jednu na počítání vnoření úrovně, na které se model zrovna nachází a druhou pro ovládání 4 možnosti kamery. Podle velikosti číselné proměnné „uroven“, tedy stavu urovně rozebrání motoru zasílají zprávy jednotlivým komponentám modelu s komentářem, zdali prvek skrýt, či zobrazit. Další proměnná, tentokrát typu Boolean (logická datový typ True/False), určuje, zdali zobrazit možnost 4. pohledu na model.

Tlačítko pro ukončení aplikace se skládá ze 2 senzorů (mouseover a kliknutí na levé tlačítko myši), které je opět přes logický član AND spojen s aktuátorem, tentorkát Game, s vybranou možností Quit.

Posledními částmi této aplikace jsou tlačítka, jejichž funkčnost by bylo příliš složité a neefektivní vytvářet pomocí Logic Editoru samotného a vyplatí se u nich použít vlastní skript, v jazyce Python. Těmito prvky jsou: START/STOP tlačítko, pro spuštění a zastavení animace, samotný slidebar, který se skládá ze 2 kvádrů a tímto skriptem probíhá jeho ovládání nebo skript pro správu samotné animace, který řeší, na jakém snimku byla animace pozastavena a kde v ní dál pokračovat tak, aby v případě znovuspuštění nedocházelo k opakování určité část znovu.

4.8 Skripty

Pro psaní skriptů lze využít Text editor, který dovoluje i zobrazení zvýraznění syntaxe nebo popis řádků kódu, pro snadnější debugování. Další silný nástrojem pro psaní skriptů Blender poskytuje v okně systémové konzole, která se dá spustit přímo v Info editoru (Window→Toggle System Console). Do konzole se vypisují jednotlivé chybové hlášení volaných skriptů s popisem chyby a udaným číslem řádku, kde k chybě došlo, což vede k urychlení celého procesu psaní skriptů a jejich ladění. Skripty se následně volají pomocí Logic Editoru, kdy se danému objektu zvolí senzory, které budou vstupovat do logického členu typu Python.



Obrázek 19: Ukázka použití skriptu s dvěma senzory polpar.py v Logic Editoru.

Skript pro ovládání vertikálního „scrollbaru“

```
import bge
  #bge modul
def main():
  #main metoda
    cont = bge.logic.getCurrentController()
      #aktualni controller (skript)
    own = cont.owner
      #vlastnik controlleru (objekt)
    click = cont.sensors["click"]
      #senzor "clik"
    sceneMotor = bge.logic.getSceneList()[0]
      #scena s motorem
    hridel = sceneMotor.objects["hridel"]
      #objekt "hridel"
    scene = bge.logic.getCurrentScene()
      #aktualni scena
    barObj = scene.objects["bar"]
      #objekt "barObj"
    moverBar = barObj.sensors["mouseOver"]
      #senzor "overBar"
    pos = moverBar.hitPosition[0];
      #promenna pos (X-ova pozice)
    textObj = scene.objects["Text"]
      #objekt "Text"
    otObj = scene.objects["ot"]
      #objekt "ot"
    empty = scene.objects["Empty"]
      #objekt "Empty"

    const_speed = round((pos-2)*24);
      #vypocet rychlosti
    if moverBar.positive and click.positive:
      #ukazatel mysi nad objektem a zaroven klik
        own.worldPosition[0] = pos;
         #nastaveni X-ove souradnice
        textObj.text = str(const_speed)
         #nastaveni napeti
        otObj.text = str((const_speed)*125)
         #nastaveni otacek
        hridel.stopAction(0)
         #zastaveni animace pri pohybu slideru
```

```
42          hridel["startFrame"] = hridel.getActionFrame(0)
43           #ulozeni snimku, kde byla animace zastavena
44          empty["speed"] = (const_speed*0.16)
45           #ulozeni zmeny rychlosti
46
47 main()
48  #volani metody main
```

Skript pro ovládání „radio buttonu“

```
1  import bge
2    #bge modul
3
4  def main():
5    #metoda main
6       cont = bge.logic.getCurrentController()
7         #aktualni controller (skript)
8       own = cont.owner
9         #vlastnik controlleru (objekt)
10      click = cont.sensors['click']
11        #senzor "click"
12      mover = cont.sensors['mover']
13        #senzor mover
14      sceneMotor = bge.logic.getSceneList()[0]
15        #scena s motorem
16      scene = bge.logic.getCurrentScene()
17        #aktualni scena
18      button = scene.objects["button"]
19        #objekt "button"
20      polpar = scene.objects["polpary"]
21        #objekt "polpary" (text)
22      otacky = scene.objects["otacky"]
23        #objekt "otacky" (text)
24      empty = scene.objects["Empty"]
25        #objekt "Empty"
26      hridel = sceneMotor.objects["hridel"]
27        #objekt "hridel"
28
29      if click.positive and mover.positive:
30        #ukazatel mysi nad objektem a zaroven klik
31          hridel["startFrame"] = hridel.getActionFrame(0)
32           #prvni frame dalsi animace
33          button.position = own.position
34           #pozice buttonu
```

```
35          if str(own) == "one":
36              polpar.text = "1"
37              otacky.text = str(3000)
38              empty["speed"] = 2
39               #nastaveni textu a rychlosti animace
40          else:
41              polpar.text = "2"
42              otacky.text = str(1500)
43              empty["speed"] = 1
44               #nastaveni textu a rychlosti animace
45
46 main()
47   #volani metody main
```

Skript pro správu animace

```
1  import bge
2    #bge modul
3
4  def main():
5    #main metoda
6
7      cont = bge.logic.getCurrentController()
8        #aktualni controller (skript)
9      own = cont.owner
10       #vlastnik controlleru (objekt)
11     scene = bge.logic.getCurrentScene()
12       #aktualni scéna
13     sceneMotor = bge.logic.getSceneList()[0]
14       #scena s motorem
15     hridel = sceneMotor.objects["hridel"]
16       #objekt hridel
17     startFrame = hridel["startFrame"]
18       #promenna startFrame objektu hridel
19
20     start = own["start"]
21       #promenna start
22     speed = own["speed"]
23       #promenna speed
24
25
26     if start:
27             hridel.playAction("hridelAction",
28                 startFrame, startFrame+50,
```

```
29                      play_mode = bge.logic.KX_ACTIONACT_PLAY,
30                      speed = speed)
31                  #start animace od aktualniho snimku
32      else:
33          hridel.stopAction()
34                  # zastaveni animace
35
36
37  main()
```

Skript pro tlačítko START/STOP

```
1   import bge
2    #bge modul
3
4   def main():
5    #metoda main
6       cont = bge.logic.getCurrentController()
7         #aktualni controller (skript)
8       own = cont.owner
9         #vlastnik controlleru (objekt)
10      mover = own.sensors["mouseOver"]
11        #senzor "mouseOver"
12      click = own.sensors["click"]
13        #senzor "click"
14      sceneMotor = bge.logic.getSceneList()[0]
15        #scena s motorem
16      hridel = sceneMotor.objects["hridel"]
17        #objekt "hridel"
18      scene = bge.logic.getCurrentScene()
19        #aktualni scena
20      empty = scene.objects["Empty"]
21        #objekt "Empty"
22      text = scene.objects["start/stop"]
23        #objekt start/stop
24
25      if mover.positive and click.positive:
26        #mouseover a zaroven klik
27          empty["start"] = not empty["start"]
28            #zmena boolean hodnoty start v Empty
29          own["start"] = not own["start"]
30            #zmena boolean hodnoty v objektu start/stop
31          if empty["start"] == False:
32            #pokud je false
```

```
33              hridel["startFrame"] = hridel.getActionFrame(0)
34                #uchovani hodnoty zacatku animace
35
36 main()
37  #volani metody main
```

Skript pro zobrazení nápovědy k ovládacím prvkům

```
1  import bge
2    #bge modul
3
4  def main():
5    #main metoda
6       cont = bge.logic.getCurrentController()
7           #aktualni controller (skript)
8       own = cont.owner
9           #vlastnik controlleru (objekt)
10      scene = bge.logic.getCurrentScene()
11          #aktualni scena
12
13      mover = cont.sensors["mover_help"]
14           #senzor "mover_help"
15      help = scene.objects["help"]
16            #objekt "help" (text)
17
18      if mover.positive:
19         #ukazatel mysi nad objektem
20          help.visible = True;
21               #nastaveni viditelnosti
22          if str(own) =="tlacitko_1":
23              help.text = "kamera 1"
24          elif str(own) =="tlacitko_2":
25              help.text = "kamera 2"
26          elif str(own) =="tlacitko_3":
27              help.text = "kamera 3"
28          elif str(own) =="tlacitko_4":
29              if own["skryto"]==False:
30                  help.text = "kamera 4"
31              if own["skryto"] == True:
32                  help.visible = False;
33          elif str(own) =="start":
34              if own["start"]==False:
35                  help.text = "spustit motor"
36              elif own["start"]==True:
```

```
37                  help.text = "zastavit motor"
38          elif str(own) =="konec":
39              help.text = "hlavní menu"
40          elif str(own) =="tlacitko_<":
41              help.text = "řpidat díl"
42          elif str(own) =="tlacitko_>":
43              help.text = "odebrat díl"
44          elif str(own) =="slider" or str(own) =="bar":
45              help.text = "vstupní ěnaptí"
46               #popis jednotlivych tlacitek
47      else:
48          help.visible = False;
49              #nastaveni viditelnosti
50
51 main()
```

5 Diskuze

5.1 Přínos

Aplikace bude využívaná při výuce elektrotechnických předmětů, zejména základy elektrotechniky (ZEL) a automatizační technika (ATE). Oproti statickým prezentacím či jednoduchým animacím, které se běžně používají umožňuje studentům změnou vstupních hodnot (vstupní napětí stejnosměrného motoru, počet pólových dvojic na statoru indukčního motoru) ovládat motor a tak spolu s možnou demontáží dílů motoru napomáhá k lepšímu pochopení problematiky.

5.2 Rozšíření

Jako další rozšíření této aplikace se nabízí implementace dalších strojů a to jak jiných typů elektromotorů (krokový, synchronní), tak třeba i generátorů, které se taktéž řadí mezi točivé elektrické stroje (alternátory, dynama), jejichž funkcí je přeměna jiného druhu energie na energii elektrickou. Další vylepšení aplikace by mohlo spočívat v možnosti zobrazení popisů s krátkým vysvětlením součástek motoru, které by se zobrazily například po rozkrytí pláště motoru, zobrazení směru toku elektrického proudu, zobrazení magnetického pole nebo možnost přidání zátěže o určité hmotnosti na hřídel.

Co se povedlo

- vytvoření modelů
- vytvoření materiálů/textur
- vytvoření základní interakce pomocí Logic Editoru / Python skriptů

Co se nezdařilo

Jako nedostatky této aplikace bych zdůraznil hlavně méně kvalitní materiály, které byly aplikovány na modely. To bylo způsobeno hlavně složitější prací v Node Editoru, kde se materiály vytváří. V případě tvorby 2D renderů, ať už při práci v Blender Render Enginu nebo Cycles Enginu se dá využít širší nabídky tohto editoru, například „mirror“ efekt pro správný odraz světla pro vizualizaci kovového povrchu. V případě BGE, tedy při tvorbě real-time aplikací, by tato operace byla příliš výpočetně náročná a tak se tento problém musí řešit pomocí dalších skriptů s importovanou knihovnou VideoTexture. Další slabší stránkou aplikace je delší doba prvního spuštění projektu se scénami samotných motorů. Tento problém může být dán tím, že je aplikace rozdělena do několika samostatných projektů (pro její lepší čitelnost) nebo kvůli horší implementací operace „Spuštění hry“ přímo v Blender Game Enginu.

6 Závěr

Cílem této práce bylo vytvořit interaktivní aplikaci pro studijní účely, která umí zobrazovat modely stejnosměrného (PMDC) a střídavého (třífázový asynchronní) elektromotoru. Interakce s modelem spočívá ve změně scén, kamer, ovládání motoru, jeho rychlosti (u stejnosměrného motoru pomocí velikosti vstupního napětí, u střídavého počtem pólových dvojic).

6.1 Nástroje

Jako vývojový nástroj byl vybrán Blender, hlavně díky jeho dostupnosti, šíří se jako open-source software a také kvůli tomu, že má vlastní Game Engine, který dokáže renderovat objekty v reálném čase a díky tomu se v něm mohou vytvářet interaktivní aplikace.

V Blenderu se pracuje v tzv. editorech, krátký výčet těch nejpoužívanějších při tvorbě této aplikace:

- 3D View Editor - zobrazení a práce se samotným modelem
- Timeline - tvorba animací
- Logic Editor - vytváření interakce (logiky aplikace)
- Text Editor - psaní skriptů
- UV/Image Editor - texturování/mapování textur
- Node Editor - tvorba materiálů

6.2 Postup práce

V první fázi samotné práce došlo k rozdělení aplikace do jednotlivých scén, celkově jich má aplikace pět, první scénou je hlavní nabídka, kde si uživatel vybere typ zobrazovaného motoru. Na dalších dvou scénách se nachází modely motorů a do posledních dvou scén je umístěno uživatelské rozhraní pro ovládání modelů. Ovládání je umístěno v samostatných scénách z toho důvodu, že je na scénách s motory možnost přepínání více kamer a ovládání je dostupné na každé z nich. Řešením tohoto problému je právě vytvoření samostatných scén a jejich správné překrývání pomocí Logického editoru.

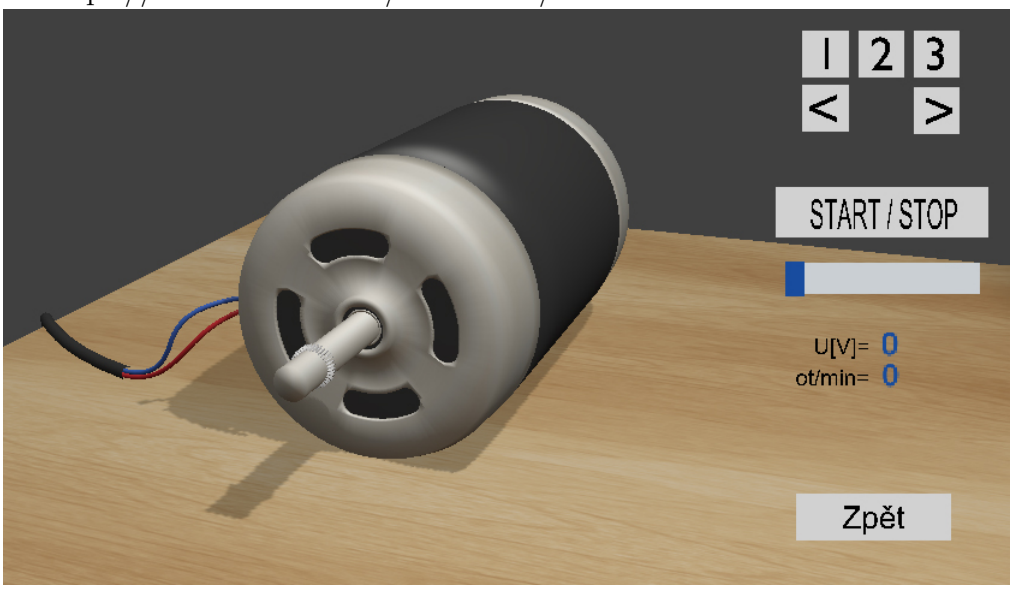
V další fázi došlo k vytvoření modelů. To v Blenderu probíhá dvěma způsoby. Kromě ručního modelování, jako je rotace, změna velikosti, vytažení, hran, vertexů nebo celých ploch je v Blenderu možnost modelovat pomocí modifikátorů. Modifikátory jsou nedestruktivní operace pro tvorbu složitějších tvarů a dělí se do 4 kategorií. V této aplikaci byly využity hlavně modifikátory Array (znásobení prvků), Subdivision Surface (hladký povrch), Screw (vytvoření pružinek pro kartáčky komutátoru) a Boolean (operace sjednocení, rozdílu a průniku objektů).

Dále byly vytvořeny materiály povrchů objektů, a vytvořily se animace pro oba motory pomocí Timeline, Dope Sheet a Graph editorů.

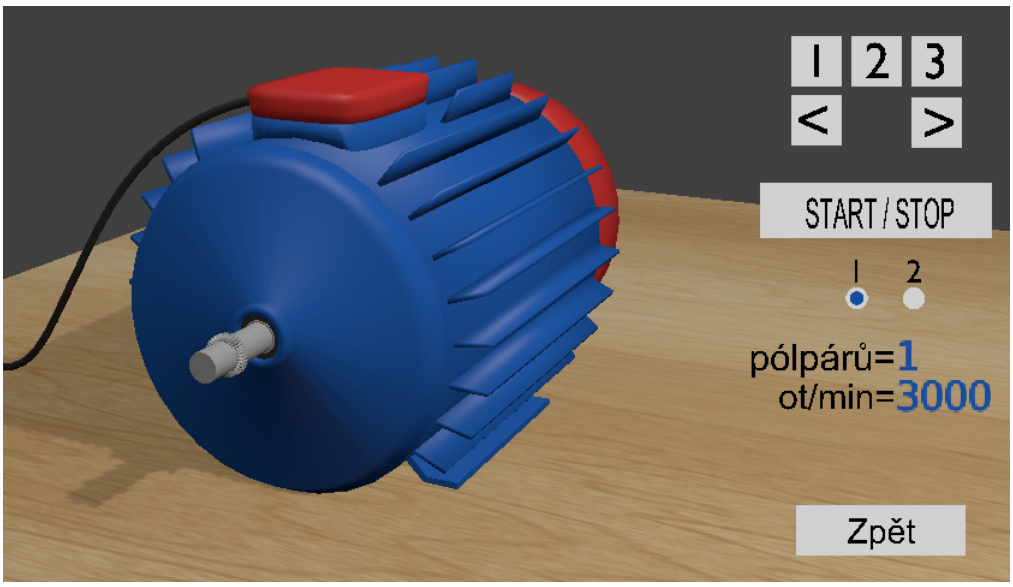
Poslední fází byla tvorba interakce mezi prvky scén. Některým jednodušším, například přepínání kamer, postačoval Logic Editor, složitější, zejména prvky uživatelského grafického rozhraní pak byly naprogramovány vlastními skripty v Textovém editoru pomocí jazyka Python.

6.3 Výsledky práce

Konečný stav aplikace umožňuje zobrazení obou motorů, stejnosměrný/střídavý, jejich ovládacích prvků a vzájemná interakce. Aplikace je dostupná ke stažení na adrese: https://akela.mendelu.cz/~xsimece2/



Obrázek 20: Celkový pohled na model PMDC motoru.



Obrázek 21: Celkový pohled na model indukčního motoru.

7 Reference

AUTODESK INC. *AutoDesk - Stingray*. AutoDesk - Stingray. [online]. 2016 [cit. 2016-04-17]. Dostupné z: http://www.autodesk.com/products/stingray/overview.

AUTODESK INC. *3ds Max*. AutoDesk. [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.autodesk.com/products/3ds-max/overview.

BLENDER *Blender dokumentace - import/export*. Blender. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://www.blender.org/manual/data_system/files/import_export.html.

BLENDER *Blender dokumentace - obrazovky*. Blender. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://www.blender.org/manual/interface/screens.html.

BLENDER *Blender dokumentace - 3DView*. Blender. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://www.blender.org/manual/editors/3dview/index.html.

BLENDER *Blender dokumentace - Timeline*. Blender. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://www.blender.org/manual/editors/timeline.html.

BLENDER *Blender dokumentace - Dope Sheet*. Blender. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://www.blender.org/manual/editors/dope_sheet/introduction.html.

BLENDER *Blender dokumentace - Logic Editor*. Blender. [online]. 2016 [cit. 2016-4-7]. Dostupné z: https://www.blender.org/manual/editors/logic_editor.html.

BLENDER *Blender dokumentace - Text Editor*. Blender. [online]. 2016 [cit. 2016-4-7]. Dostupné z: https://www.blender.org/manual/editors/text_editor.html.

BLENDER *Blender dokumentace - Graph Editor*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/editors/graph_editor/introduction.html.

BLENDER *Blender dokumentace - Graph Editor*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/editors/graph_editor/introduction.html.

BLENDER *Blender dokumentace - modifikátory*. Blender. [online]. 2016 [cit. 2016-3-28]. Dostupné z: https://wiki.blender.org/index.php/Doc:CZ/2.6/Manual/Modifiers.

BLENDER *Blender dokumentace - Blender Render Engine*. Blender. [online]. 2016 [cit. 2016-4-4]. Dostupné z: https://www.blender.org/manual/render/blender_render/index.html.

BLENDER *Blender dokumentace - Cycles Render Engine*. Blender. [online]. 2016 [cit. 2016-4-4]. Dostupné z: https://www.blender.org/manual/fr/render/cycles/index.html.

BLENDER *Blender dokumentace - Blender Game Engine*. Blender. [online]. 2016 [cit. 2016-4-4]. Dostupné z: https://www.blender.org/manual/game_engine/introduction.html.

BLENDER *Blender dokumentace - UV/Image Editor*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/editors/uv_image/uv_editing.

BLENDER *Blender dokumentace - Node Editor*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/editors/node_editor/introduction.html.

BLENDER *Blender dokumentace - Python console*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/editors/python_console.html.

BLENDER *Blender dokumentace - Properties editor*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/de/editors/properties/introduction.html.

BLENDER *Blender dokumentace - Solidify*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/modeling/modifiers/generate/solidify.html.

BLENDER *Blender dokumentace - Subdivision Surface*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/modeling/modifiers/generate/subsurf.html.

BLENDER *Blender dokumentace - Boolean*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/modeling/modifiers/generate/booleans.html.

BLENDER *Blender dokumentace - Array*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/modeling/modifiers/generate/array.html.

BLENDER *Blender dokumentace - Screw*. Blender. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://www.blender.org/manual/modeling/modifiers/generate/screw.html.

BLENDER *Blender dokumentace*. Blender. [online]. 2015 [cit. 2015-12-20]. Dostupné z: http://wiki.blender.org/index.php/Doc:CZ/2.4/Manual/Introduction.

BUREŠ, JIŘÍ. *Hans Christian Oersted*. Converter. [online]. 2016 [cit. 2015-3-25]. Dostupné z: http://www.converter.cz/fyzici/oersted.htm.

CAD STUDIO A.S. *CAD studio*. CAD Studio. [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.cadstudio.cz/3dsmax.

Dannhoferová Jana *Hranové reprezentace*. Mendelu. [online]. 2016 [cit. 2016-3-27]. https://is.mendelu.cz/eknihovna/opory/index.pl?cast=761;d_back=1;lang=cz.

Digital Media s.r.o. *Cinema4D*. Cinema4D . [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.cinema4d.cz/.

Digital Media s.r.o. *Cinema4D - Hair*. Cinema4D . [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.cinema4d.cz/produkty/maxon/cinema-4d-studio/popis/hair.aspx.

Dillard Ted *Basic Motor Types*. Wordpress. [online]. 2016 [cit. 2016-5-6]. Dostupné z: https://evmc2.wordpress.com/2014/12/04/basic-motor-types-pmdc-bldc-ac-induction-and-synchronous-and-series-dc/.

Epic Games, Inc. *Unreal Engine dokumentace*. Unreal Engine. [online]. 2016 [cit. 2016-4-17]. Dostupné z: https://wiki.unrealengine.com/Static_Mesh_from_Blender.

Futur3D *Software pro tvorbu 3D modelu*. Futur3D. [online]. 2015 [cit. 2015-12-20]. Dostupné z: http://www.futur3d.net/3d-software.

HRADIL, František a Jan ŠKYŘÍK. *Silnoproudá elektrotechnika*. 1. Praha: Státní pedagogické nakladatelství, 1993. ISBN 80-7157-045-1.

MAXON Computer *Cinema4D - Game Development*. Cinema4D - Game Development . [online]. 2016 [cit. 2016-04-13]. Dostupné z: http://www.maxon.net/products/workflow-integration/game-development.html.

McNeel *Rhinoceros 3D - Penguin*. Rhinoceros3D - Penguin . [online]. 2016 [cit. 2016-04-16]. Dostupné z: http://wiki.mcneel.com/penguin/home.

McNeel *Rhinoceros 3D - Brazil*. Rhinoceros3D - Brazil . [online]. 2016 [cit. 2016-04-16]. Dostupné z: http://wiki.mcneel.com/brazil/home.

Murphy Jim *AC induction motors*. MachineDesign. [online]. 2016 [cit. 2016-5-6]. Dostupné z: http://machinedesign.com/motorsdrives/whats-difference-between-ac-induction-permanent-magnet-and-servomotor-technologies.

Robert McNeel Associates *Rhinoceros 3D*. Rhinoceros3D. [online]. 2016 [cit. 2016-04-16]. Dostupné z: http://www.rhino3d.com/.

Strachota Pavel *Modelování pevných těles*. ČVUT. [online]. 2016 [cit. 2016-3-26]. Dostupné z: http://saint-paul.fjfi.cvut.cz/base/sites/default/files/POGR/POGR2/05.modelovani_teles.pdf.

Takáč, Marcel *3D počítačová grafika*. Ostrovskeho. [online]. 2016 [cit. 2016-3-26]. Dostupné z: http://ostrovskeho.sk/ucivo/data/sse/3D_grafika_teoria_01.pdf.

Trimble Navigation *SketchUp*. SketchUp . [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.sketchup.com/.

Tušš, Milo3 a Žaneta Pavlištíková *SketchUp - Historie*. Earch . [online]. 2016 [cit. 2016-04-09]. Dostupné z: http://www.earch.cz/cs/it/sketchup-pro-1-dil-historie-vzniku.

Unity Technologies *Unity dokumentace*. Unity. [online]. 2016 [cit. 2016-4-16]. Dostupné z: http://docs.unity3d.com/Manual/HOWTO-ImportObjectBlender.html.

Vysoká škola báňská *Počítačová grafika*. Vysoká škola báňská — Technická univerzita Ostrava. [online]. 2015 [cit. 2015-12-20]. Dostupné z: http://homen.vsb.cz/~ska74/HGF/uvod.htm.

Wikipedia *Počítačová 3D grafika*. Wikipedia. [online]. 2015 [cit. 2015-12-20]. Dostupné z: https://goo.gl/lmGggg.

Wikibooks *Blender dokumentace*. Blender. [online]. 2016 [cit. 2016-3-25]. Dostupné z: https://goo.gl/pS5shb.

Wonderful Engineering *Electrical Motor Images*. Wonderful Engineering. [online]. 2016 [cit. 2016-5-6]. Dostupné z: http://cdn.wonderfulengineering.com/wp-content/uploads/2014/04/electric-motor-4.jpg.

Fyzmatik *První elektrický motor - Faradayův motor*. Fyzmatik. [online]. 2016 [cit. 2015-3-26]. Dostupné z: http://fyzmatik.pise.cz/1541-prvni-elektricky-motor-faradayuv-motor.html.