Fakulta informatiky a informačných technológií
Slovenská technická univerzita

VRLab Radiant

Dokumentácia k riadeniu

(2. kontrolný bod)

Akademický rok: 2020/2021

Pedagogický vedúci: Ing. Juraj Vincúr

Členovia tímu (č.16): Bc. Patrik Tománek
Bc. Ľubomír Kurčák
Bc. Erik Paľa
Bc. Viktor Beňo

OBSAH

Úvod 5
1. Roly členov tímu a podiel práce 6
1.1. Roly členov tímu 6
1.2. Podiel práce na dokumentácií 7
2. Aplikácia manažmentu 8
2.1. Manažment agilného riadenia 8
2.2. Manažment komunikácie 9
2.3. Manažment dokumentácie 9
2.4. Manažment verzií 10
3. Sumarizácia šprintov 10
3.1. Šprint 1 (12.10. – 25.10.) 10
3.2. Šprint 2 (26.10 – 8.11.) 12
3.3. Šprint 3 (9.11. – 22.11.) 14
3.4. Šprint 4 (23.11. – 6.12.) 17
3.5. Šprint 5 (7.12. – 14.12.) 19
3.6. Šprint 6 (26.2. – 12.3.) 21
3.7. Šprint 7 (13.3. – 26.3.) 22
3.8. Šprint 8 (27.3. – 9.4.) 24
3.9. Šprint 9 (10.4. – 25.4.) 26
3.10. Šprint 10 (26.4. – 9.5.) 27
3.11. Šprint 11 (9.5. – 14.5.) 29
4. Globálna retrospektíva 30
4.1. Retrospektíva k šprintu 1 (26.10.) 30
4.2. Retrospektíva k šprintu 2 (9.11.) 31
4.3. Retrospektíva k šprintu 3 (23.11.) 31
4.4. Retrospektíva k šprintu 4 (7.12.) 32
4.5. Retrospektíva k šprintu 5 (14.12.) 32
4.6. Retrospektíva k šprintu 6 (12.3.) 33
4.7. Retrospektíva k šprintu 7 (26.3.) 33
4.8. Retrospektíva k šprintu 8 (9.4.) 34
4.9. Retrospektíva k šprintu 9 (23.4.) 34
4.10. Retrospektíva k šprintu 10 (7.5.) 35
4.11. Retrospektíva k šprintu 11 (14.5.) 35
Záver 36

Príloha A. Motivačný dokument

A.1. Tím

A.2. Motivácia

A.3. Zoradenie tém podľa priorít

A.4. Rozvrhy tímu

Príloha B. Prihláška na TP Cup

B.1. Kontakt

B.2. Tím

B.3. Motivácia

B.4. Náplň projektu

B.5. Ciele projektu

Príloha C. Metodiky

C.1. Metodika dokumentácie

C.2. Metodika agilného riadenia

C.3. Metodika komunikácie

C.4. Metodika verzovania

Príloha D. Zápisnice

D.1. Zápisnica č.1 (12.10.)

D.2. Zápisnica č.2 (19.10.)

D.3. Zápisnica č.3 (26.10.)

D.4. Zápisnica č.4 (2.11.)

D.5. Zápisnica č.5 (9.11.)

D.6. Zápisnica č.6 (16.11.)

D.7. Zápisnica č.7 (23.11.)

D.8. Zápisnica č.8 (30.11.)

D.9. Zápisnica č.9 (7.12.)

D.10. Zápisnica č.10 (14.12.)

D.11. Zápisnica č.11 (26.2.)

D.12. Zápisnica č.12 (6.3.)

D.13. Zápisnica č.13 (12.3.)

D.14. Zápisnica č.14 (19.3.)

D.15. Zápisnica č.15 (26.3.)

D.16. Zápisnica č.16 (6.4.)

D.17. Zápisnica č.17 (9.4.)

D.18. Zápisnica č.18 (16.4.)

D.19. Zápisnica č.19 (23.4.)

D.20. Zápisnica č.20 (30.4.)
D.21. Zápisnica č.21 (7.5.)
D.22. Zápisnica č.22 (14..)
Príloha E. Export evidencie úloh
E.1. Export úloh k šprintu 1
E.2. Export úloh k šprintu 2
E.3. Export úloh k šprintu 3
E.4. Export úloh k šprintu 4
E.5. Export úloh k šprintu 5
E.6. Export úloh k šprintu 6
E.7. Export úloh k šprintu 7
E.8. Export úloh k šprintu 8
E.9. Export úloh k šprintu 9
E.10. Export úloh k šprintu 10
E.11. Export úloh k šprintu 11
Príloha F. Webové sídlo projektu

Úvod

Tento dokument predstavuje dokumentáciu k riadeniu projektu v rámci predmetu Tímový projekt. Cieľom nášho projektu je vytvorenie laboratória vo virtuálnej realite pre distančné vzdelávanie, určené pre výučbu a simuláciu elektrických obvodov.

Súčasťou tohto dokumentu je predstavenie členov nášho tímu spolu s opisom ich úloh a podielu práce na jednotlivých častiach dokumentácie. Ďalej opisujeme jednotlivé manažmenty, ktoré sú potrebné pre zvýšenie efektívnosti a zlepšenie organizácie nášho tímu. Následne boli rozpísané šprinty nášho projektu počas zimného a letného semestra, pre každý šprint sme napísali naše ciele a aké výsledky sme dosiahli. Progres jednotlivých šprintov uvádzame vo forme *burndown* grafu. Poslednou sekciou dokumentu k riadeniu je globálna retrospektíva.

K dokumentu prikladáme prílohy ako motivačný dokument, prihlášku na TP Cup, metodiky nášho tímu, zápisnice zo stretnutí, automaticky generované exporty úloh zo šprintov z nástroje *Azure DevOps* a opis webového sídla projektu.

1. ROLY ČLENOV TÍMU A PODIEL PRÁCE

Autor kapitoly: Patrik Tománek

V tejto časti opíšeme roly jednotlivých členov tímu ako aj podiel celkovej práce na dokumentácií. Jednotlivé roly členov tímu sa môžu v budúcnosti zmeniť.

1.1. ROLY ČLENOV TÍMU

Ing. Juraj Vincúr

- Vedúci projektu VRLab
- Product Owner

Vytvára *product backlog* pre naše šprinty a usmerňuje nás k úspešnému dokončeniu nášho projektu.

Bc. Patrik Tománek

- Tímový líder
- Unity developer
- Web developer

Spolu so *Scrum Masterom* má na starosti udalosti spojené s agilným vývojom, ako napríklad plánovanie, retrospektívy, manažment tímových stretnutí a podobne. Pravidelne komunikuje s tímom na stave jednotlivých úloh spolu s riešením prípadných problémov s úlohami. Zaznamenáva každé stretnutie tímu s *Product Ownerom*, spisuje zápisnice, retrospektívy z týchto stretnutí.

Ako Unity developer má na starosti prácu na vývoji projektu a prototypov pre náš projekt. Má skúsenosti s vývojom v Unity, či už na univerzitných projektov, ako napríklad bakalárska práca, ale najmä na mimo-univerzitných projektov.

Ako Web developer má na starosti tvorbu a vylepšovanie našej webovej stránky tímu, ako aj pravidelné aktualizovanie obsahu tejto stránky, primárne dokumentovú časť.

Bc. Ľubomír Kurčák

- Scrum Master
- Unity developer
- Graphic designer

Má za úlohu zabezpečiť dodržiavanie pravidiel agilného vývoja v našom tíme. S tým súvisí aj administrácia nástroje *Azure DevOps*. Na každom stretnutí prezentuje progres tímu na príslušnom šprinte. Zdieľa jeho obrazovku počas celej doby stretnutia, na ktorej ukazuje spomínaný progres, ako aj náš celkový stav šprintu v *Azure DevOps*.

Ako Unity developer má na starosti prácu na vývoji projektu a prototypov pre náš projekt. Má skúsenosti s vývojom v Unity.

Ako Graphic designer má na starosti tvorbu modelov pre náš projekt. Má skúsenosti s prácou v Blender grafickom prostredí.

Bc. Tomáš Sabo (člen tímu počas zimného semestra)

- Deployment
- Unity Developer
- Web Developer

Ako Unity developer má na starosti prácu na vývoji projektu a prototypov pre náš projekt. Má skúsenosti s vývojom v Unity, ako napríklad tvorba aplikácie na generovanie UML diagramov v 3D priestore v prostredí Unity, počas jeho tímového projektu, na ktorom pracoval v roku 2018/2019.

Ako Web Developer pomáha so správou a vývojom našej webovej stránky, takisto kontroluje aktuálnosť dokumentovej časti našej stránky.

Ako Deployment má na starosti správu všetkých dokumentov ako aj verzií nášho projektu. Pred odovzdaním niektorých z častí, výsledný produkt najprv skontroluje a rozhoduje, či môže byť produkt odovzdaný alebo je potrebná jeho úprava.

Tomáš s nami už ďalej nepokračoval v letnom semestri, a preto všetka jeho práca zaznamenaná v tomto dokumente sa týka iba zimného semestra.

Bc. Erik Paľa

- Analytik
- Dokumentácia
- Tester

Ako analytik má na starosti analyzovať riešenia založene na breadboard prístupe. Takisto navrhuje vlastné riešenia, podľa ktorých sa následne vytvoria prototypy. Výstupom jeho analýzy sú takisto aj dokumentácie

Ako tester má na starosti testovanie všetkých prototypov, pričom po otestovaní vytvorí dokumentáciu, ktorú následne odošle na nás Discord server, aby si ju mohli Unity developeri prečítať a oboznámiť sa s problémami ich prototypu.

Bc. Viktor Beňo

- Analytik
- Dokumentácia
- Tester

Ako analytik má na starosti analyzovať riešenia založene na grid prístupe. Takisto navrhuje vlastné riešenia, podľa ktorých sa následne vytvoria prototypy. Výstupom jeho analýzy sú takisto aj dokumentácie

Ako tester má na starosti testovanie všetkých prototypov, pričom po otestovaní vytvorí dokumentáciu, ktorú následne odošle na nás Discord server, aby si ju mohli Unity developeri prečítať a oboznámiť sa s problémami ich prototypu.

1.2. Podiel práce na dokumentácií

Pri jednotlivých kapitolách dokumentu ako aj prílohách uvádzame meno autora, ktorý danú kapitolu, respektíve prílohu vypracoval. Pred dokončením dokumentov ich Erik Paľa skontroloval za účelom nájdenia akýchkoľvek chýb.

Túto podkapitolu sme rozdelili do 2 častí, na základe 2 dokumentov, ktoré máme v kontrolných bodov odovzdať.

Dokument k riadeniu – Dokument sa člení do niekoľkých častí ako je úvod, roly členov tímu a podiel práce, aplikácia manažmentu, sumarizácia šprintov, globálna retrospektíva a záver. Tieto časti vypracoval Patrik Tománek. K dokumentu sú takisto priložene prílohy ako Motivačný dokument, ktorý bol vytvorený všetkými členmi tímu: Patrik Tománek, Erik Paľa, Ľubomír Kurčák, Viktor Beňo a Tomáš Sabo. Ďalej prílohy ako prihláška na TP Cup, zápisnice a webové vytvoril Patrik Tománek. Prílohu export evidencie úloh vytvorili Ľubomír Kurčák a Patrik Tománek a na metodikách pracoval Tomáš Sabo.

Dokument k inžinierskemu dielu – Dokument sa člení do niekoľkých častí ako je úvod, globálne ciele pre zimný semester, celkový pohľad na systém, moduly systému, prototypy, záver a bibliografia.. Na tvorbe tohto dokumentu sa podieľali všetci členovia nášho tímu, autori jednotlivých kapitol alebo podkapitol sú uvedení k príslušnej časti.

Autori jednotlivých častí inžinierskeho diela sú uvedení v Tabuľka 1

Časť dokumentu	Autori
Úvod	Ľubomír Kurčák, Erik Paľa
Globálne ciele pre zimný semester	Ľubomír Kučák, Patrik Tománek
Globálne ciele pre letný semester	Erik Paľa
Celkový pohľad na systém	Ľubomír Kurčák, Patrik Tománek
Breadboard riešenie	Erik Paľa
Blokové (grid) riešenie	Viktor Beňo
Prekladač modelu na obvod pre breadboard riešenie	Erik Paľa
Prekladač modelu na obvod pre blokové (grid) riešenie	Viktor Beňo
Vyhodnotenie návrhov a riešení	Erik Paľa, Viktor Beňo
Synchronizačné frameworky	Erik Paľa, Viktor Beňo, Tomáš Sabo
Prototyp modelov súčiastok	Ľubomír Kurčák, Patrik Tománek, Viktor Beňo
Prototyp pre ovládanie a interakciu	Ľubomír Kurčák, Patrik Tománek
Prototyp pre podporu virtuálnej reality	Ľubomír Kurčák, Patrik Tománek, Viktor Beňo
VRLAB PROTOTYP	Patrik Tománek, Viktor Beňo
Záver	Ľubomír Kurčák, Erik Paľa

Tabuľka 1 - Autori jednotlivých častí inžinierskeho diela

2. APLIKÁCIA MANAŽMENTU

Autor kapitoly: Patrik Tománek

V tejto časti opisujeme funkcie manažmentov v jednotlivých častiach projektu. K jednotlivým častiam takisto prislúcha metodika, ktorou sa riadia všetci členovia tímu.

2.1. MANAŽMENT AGILNÉHO RIADENIA

Naše stretnutia s *Product Ownerom* prebiehajú pravidelne každý pondelok od 7:00 pomocou Google Meet. Na každom stretnutí Ľubomír Kurčák prezentoval jeho obrazovku, na ktorej ukazoval stav nášho šprintu, ako aj celý progres, ktorý sme vytvorili.

Na začiatku stretnutia, sme si prebehli stav nášho šprintu zaznamenaný v *Azure DevOps*. *Product Owner* prešiel všetky položky šprintu, pričom ku každej sme prezentovali náš progres, ktorý sme

doposiaľ vytvorili. Každý výstup bol vzájomne diskutovaný a prípadne požiadavky na zmeny boli hneď zapísané pod príslušnú úlohu. Pokiaľ riešiteľ niektorej z úloh narazil na problém, prebiehala diskusia za účelom vyriešenia tohto problému, aby sa mohlo naďalej pokračovať v práci na danej úlohe. Stretnutia v strede šprintu slúžili hlavne na doriešenie prípadných problémov a otázok. Stretnutia na začiatku šprintu *Product Owner* skontroloval výstup celého šprintu a potom sa prešlo na retrospektívu tohto šprintu a pokračovalo sa naplánovaním nového šprintu.

Retrospektíva prebiehala takou formou, že *Product Owner* sa opýtal, čo bolo dobré na šprinte a čo by sme mohli zmeniť. Všetci členovia tímu mali najprv čas na rozmyslenie a vyslovenie svojich odpovedí, pričom vždy hovoril iba jeden člen tímu, až dokým nedokončí všetko, čo chcel povedať a dá slovo ďalšiemu. Následne sa spoločne diskutovalo o všetkých odpovediach za účelom nájdenia riešenia pre vzniknuté problémy.

Na plánovanie šprintu využívame nástroj *Azure DevOps,* ktorý spravuje náš *Scrum Master* Ľubomír Kurčák. Na začiatku plánovania šprintu *Product Owner* definuje všetky *User Stories*, ktoré máme v danom šprinte vypracovať. Každá *User Story*, ešte predtým ako je nasadená do šprintu je ohodnotená formou *User* Points pomocou techniky *planning poker*. Po tom, ako sme dostali všetky *User Stories* sa prejde na tvorbu všetkých úloh pre každú *User Story.* Tieto úlohy sú následne rozdelené medzi jednotlivých členov tímu.

Detaily manažmentu agilného riadenia sú v **Metodika agilného riadenia.**

2.2. MANAŽMENT KOMUNIKÁCIE

Komunikácia sa skladá z 2 hlavných častí:

- Týždenná komunikácia
  - stretnutie s *Product Ownerom*
  - tímové stretnutie
- Pravidelná komunikácia
  - s *Product Ownerom*
  - s tímom

Týždenná komunikácia prebiehala vo forme hovoru a pravidelná komunikácia prebiehala v textovej forme. Pre týždennú komunikáciu s *Product Ownerom* používame Google Meet, a pre komunikáciu s tímom používame vlastný Discord server. Tento Discord server takisto slúži pre pravidelnú komunikáciu s tímom. Server obsahuje niekoľko kanálov, kde každý ma špeciálny účel. Pre pravidelnú komunikáciu s *Product Ownerom* sme vytvorili Slack server, kde sa takisto nachádza kanál integrácie s TFS.

Založili sme si takisto tímovú e-mailovú schránku. Táto schránka slúži na formálnu komunikáciu za celý tím. Každý s členov tímu majú nastavené preposielanie pošty ž tejto schránky na ich súkromné, vďaka tomu môžeme rýchlo reagovať na všetky správy.

Adresa e-mailovej schránky je: tp16.radiant@gmail.com

Detaily manažmentu komunikácie sú v **Metodika komunikácie**.

2.3. MANAŽMENT DOKUMENTÁCIE

Počas celého procesu vývoja vzniká príslušná dokumentácia. Určili sme si osobu, ktorá má na starosti udržiavať vytvorené dokumenty v jednotnej podobe. Všetky tieto dokumenty sú uložené na

web stránke nášho tímu. Na zálohovanie ako aj zdieľanie dokumentov medzi jednotlivými členmi tímu používame náš Discord server v príslušnom kanále. Takisto sa ku každému dokumentu môže vyjadriť člen s návrhom na jeho zmenu.

Dokumentácie vznikajú priebežne počas celého procesu tvorby projektu. Zápisnice vznikajú počas každého stretnutia. Retrospektívy vznikajú po každom šprinte, spolu s exportom úloh daného šprintu. Metodiky sú písane podľa našej potreby.

Detaily manažmentu dokumentácie sú v **Metodika dokumentácie**.

2.4. MANAŽMENT VERZIÍ

Na manažment verzií používame Unity funkcionalitu *Unity Collaborate*. Vďaka tejto funkcionality môžeme udržiavať všetky verzie nášho produktu na jednom mieste. Každý z členov tímu má prístup k tomuto miestu, kde môže vidieť všetky *commit* správy. Prečo sme sa rozhodli používať túto funkcionalite je fakt, že každá verzia produktu je zvlášť uložená a môže sa medzi nimi jednoducho prepínať v *Unity Editore* a teda z prostredia, v ktorom vyvíjame náš produkt. *Unity Collaborate* nám takisto umožňuje spoločnú prácu na projekte. Vždy keď na ňom pracuje viacero ľudí naraz, môžeme vidieť to, čo robia ostatní a tým pádom veľmi jednoducho predídeme kolíziam.

Detaily manažmentu verzií sú v **Metodika verzií.**

3. SUMARIZÁCIA ŠPRINTOV

Autor kapitoly: Patrik Tománek

3.1. ŠPRINT 1 (12.10. – 25.10.)

Alias šprintu: Apple

Prvý šprint nášho tímového projektu bol najmä o zoznamovaní sa s agilným vývojom Scrum. Boli nám vysvetlené všetky potrebné funkcionality prostredia *Azure DevOps*, ktoré budeme potrebovať a využívať počas celej práce na našom projekte.

Tímové stretnutia prebiehali pravidelne každý týždeň, výsledkom každého stretnutia bola zápisnica. Po tom ako nám bol vysvetlený priebeh šprintov sme začali s napĺňaním prvého šprintu. Pre všetky *User Stories* sme udelili *Story Points* pomocou techniky *planning poker.* Každú *User Story* sme mohli diskutovať s naším *Product Ownerom*, aby sme nemali žiadne nejasnosti počas práce. Po tom, ako sme dostali všetky *User Stories*, už bolo na nás aké úlohy si k jednotlivým *User Stories* vytvoríme. Najskôr po tom ako sme si vytvorili jednotlivé úlohy, sme si tieto úlohy medzi sebou rozdelili. Každú úlohu sme si takisto aj ohodnotili v počte hodín, koľko nám bude približne trvať jej vypracovanie.

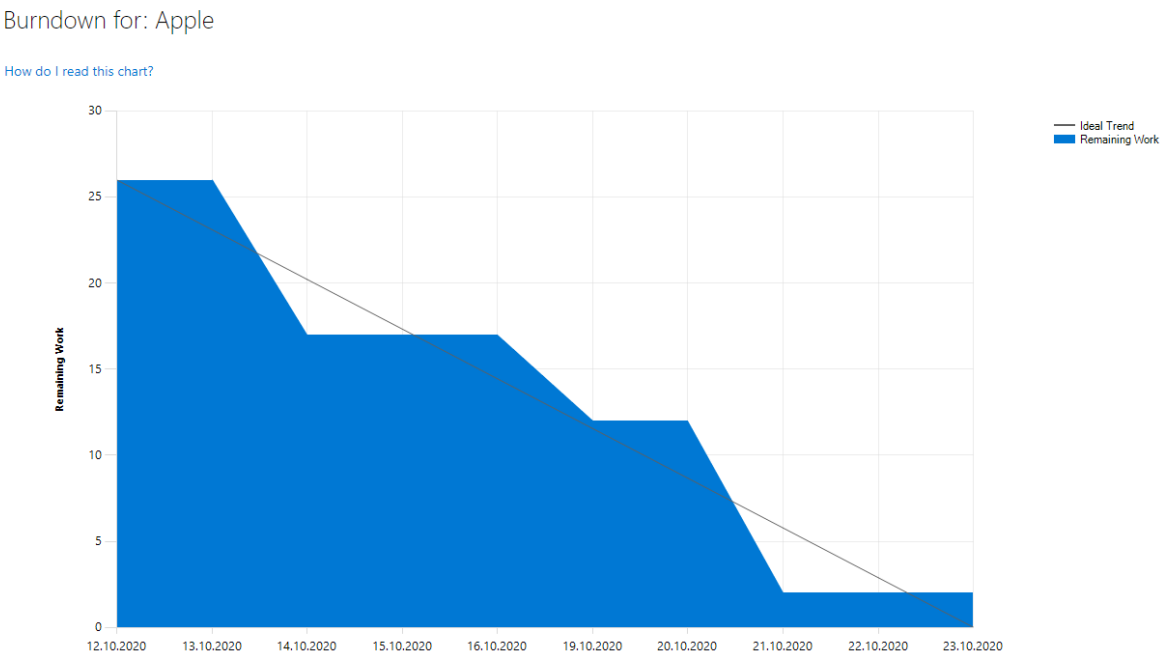
V prvom šprinte sme sa chceli hlavne zamerať na vytvorenie a nasadenie webovej stránky pre náš projekt. Táto úloha bola rozdelená medzi 3 členmi tímu: Patrik Tománek, Ľubomír Kurčák a Tomáš Sabo. Vypracovala sa iba 1 verzia stránky, pretože s ňou bol náš *Product Owner* hneď spokojný, a teda neboli potrebné ďalšie zmeny iné ako zmena textu na stránke.

Prvý šprint od nás takisto vyžadoval vypracovať analýzu na rôzne simulátory pre logické ako aj elektrické obvody, pretože v tomto čase sme nevedeli, akým smerom pôjdeme. Na tejto *User Story* pracovali naši dvaja analizátori Erik Paľa a Viktor Beňo. Erik mal za úlohu vypracovať analýzu pre elektrické obvody a Viktor pre logické obvody. Výsledkom analýzy mal byť dokument, ktorý mal

obsahovať link na daný simulátor, jeho popis, ukážku jeho funkčnosti, výhody a nevýhody daného simulátora.

Poslednou úlohou nášho šprintu bola prvotná integrácia vybraných simulátorov do Unity. Na tejto úlohe sa začalo pracovať až po dokončení predošlej úlohy, aby sme vedeli, ktoré simulátory budeme chcieť odskúšať. Po konzultácií simulátorov s naším *Product Ownerom*, sme sa dohodli, že simuláciu vypracujeme pre 2 simulátory. Jedným so simulátorom bol SpiceSharp a tým druhým bol hotový simulátor taktiež vytvorený v prostredí Unity za použitia VR. Integráciu mal na starosti Ľubomír Kurčák, ktorý integroval SpiceSharp do nášho Unity projektu a otestoval druhý už hotový projekt.

Na *Obrázok 1* je znázornený *Burndown chart* 1. šprintu. Dôvod prečo nám graf na konci trochu preteká nie je ten, že sme nedokončili jednu z úloh. Neuvedomili sme si to, že pokiaľ nastavíme koniec šprintu v deň nášho stretnutia s *Product Ownerom*, ktorý máme v pondelok, tak nám šprint končí ako keby v piatok o 23:59, pretože sa berú do úvahy len pracovné dni. Na víkend sme mali ešte naplánované stretnutie tímu, aby sme prediskutovali jednotlivé integrácie simulátorov v Unity, aby sme si spoločne vybrali cestu, ktorou pôjdeme.



Obrázok 1 - Burndown chart 1. šprintu

Prvý šprint sa skladal z 3 *User Stories:*

User Story: Vytvorenie webového sídla

Riešitelia: Ľubomír Kučák, Tomáš Sabo, Patrik Tománek

Story Points: 13

Stav: Uzavretý

Vytvorili sme webovú stránku s potrebným obsahom. Následne do nej budú pribúdať nové dokumenty.

User Story: Analýza simulátorov pre logické a elektrické obvody

Riešitelia: Viktor Beňo, Erik Paľa

Story Points: 8

Stav: Uzavretý

Analyzovali sme niekoľko simulátorov pre logické a elektrické obvody, pričom sme si ako najväčší potenciál určili SpiceSharp knižnicu pre simuláciu elektrických obvodov.

User Story: Prvá integrácia v Unity

Riešiteľ: Ľubomír Kurčák

Story Points: 5

Stav: Uzavretý

Integrovali sme 2 možné simulátory do nášho projektu, pričom SpiceSharp knižnica preukázala vysoký potenciál a druhý simulátor, resp. hotový projekt v Unity sme zamietli z dôvodu jeho nedostatočnej funkcionality.

3.2. Šprint 2 (26.10 – 8.11.)

Alias šprintu: Banana

Druhý šprint bol zameraný na dokončenie všetkých potrebných úloh aby sme mohli prejsť na prvú tvorbu prototypu v Unity, riešili sa takisto aj iné veci ako napríklad napísanie a odoslanie prihlášky do TP Cup ako aj nájdenie vhodného nástroju pre komunikáciu medzi tímom a *Product Ownerom* aj počas práce na šprintu.

Prihlášku do TP Cup si zobral na starosť Patrik Tománek, ktorý najprv spísal základný koncept, ktorý sme spoločne ohodnotili a upravili.

Ako nástroj na komunikáciu medzi tímom a *Product Ownerom*, sme sa rozhodovali medzi Discordom a Slackom, takisto bolo potrebné do vybraného nástroja vytvoriť integráciu s TFS. Túto *User Story* mali na starosti Viktor Beňo a Ľubomír Kurčák.

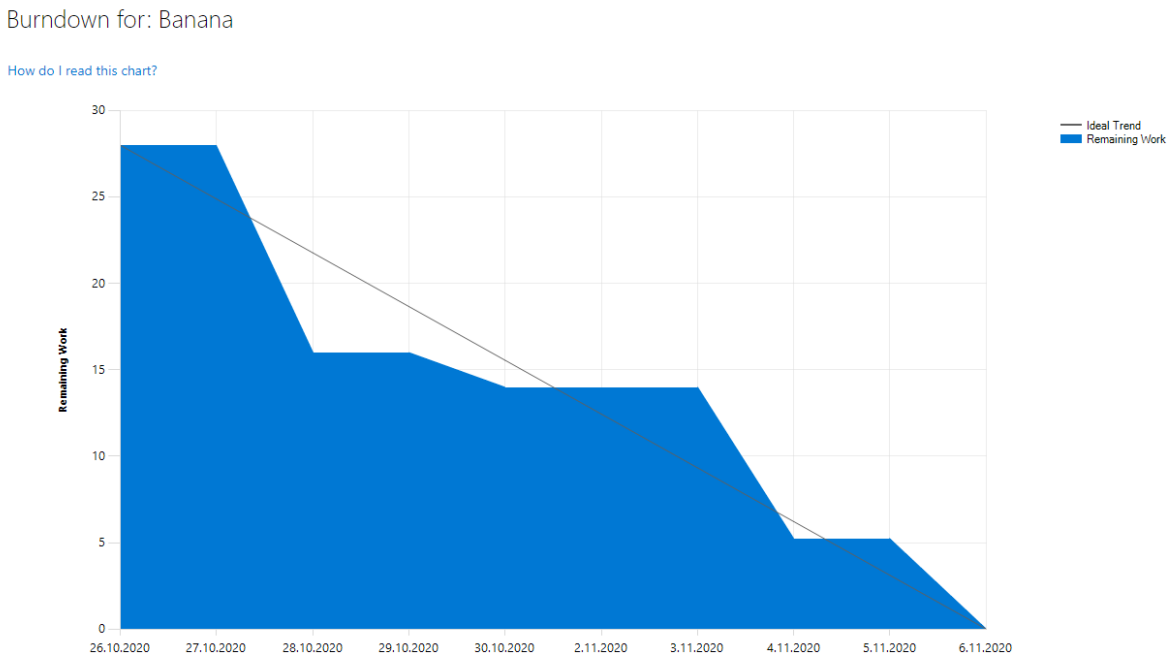
Čo sa týka dokončeniu všetkých potrebných úloh pred prvotnou tvorbou prototypu v Unity, tím začal s hľadaním 3D modelov pre elektrické a logické obvody. Síce sme sa po prvom šprinte dohodli, že sa vydáme smerom elektrických obvodov, pre istotu sme sa stále mali aspoň okrajovo venovať logickým obvodom v prípade, že by sme narazili na problémy. Túto *User Story* mali na starosti Ľubomír Kurčák a Patrik Tománek. Pri hľadaní modelov sa malo dbať celkový polycount modelov, pretože prototyp v budúcnosti môže byť integrovaný pre podporu Android zariadení. V takomto prípade je dôležité dodržať striktne daný maximálny počet polygónov, inak nebude možné náš projekt do takéhoto prostredia nasadiť. Sústredili sme sa na voľne dostupné modely s čo najnižším počtom polygónov. Všetky nádejne modely mali byť takisto otestované v Unity prostredí za účelom nájdenia rôznych chýb ako napríklad nesprávne tiene, priesvitnosť textúr atď.

Keďže náš projekt je určený pre mladých študentov, ako ďalšiu *User Story* sme mali na starosti vypracovanie analýzy existujúcich riešení z praxe, konkrétne tých fyzických (hmotných). Táto *User Story*

bola zameraná na inšpiráciu ako by mal náš finálny produkt vyzerať. Pracovali na nej Tomáš Sabo, Erik Paľa a Viktor Beňo. Viktor mal za úlohu analyzovať grid riešenia a Erik mal na starosti analýzu breadboard riešení.

Ako poslednú *User Story* sme mali vytvoriť návrh potencionálneho riešenia. Táto *User Story* bola závislá od výstupu predošlých 2 *User Stories,* konkrétne nájdenie 3D modelov a analýzy riešení z praxe. Následne Viktor Beňo a Erik Paľa vytvorili návrhy ako by mohol vyzerať prvotný prototyp nášho projektu. Viktor mal na starosti navrhnúť grid riešenie a Erik breadbordové riešenie.

Na *Obrázok 2* môžeme vidieť *Burndown chart* 2. šprintu. Tentokrát sme si dali pozor na dokončení všetkých úloh ešte v pracovných dňoch a nie cez víkend.



Obrázok 2 - Burndown chart 2. šprintu

Druhý šprint sa skladal z 5 *User Stories:*

User Story: Napísať prihlášku do TP Cup

Riešiteľ: Patrik Tománek

Story Points: 2

Stav: Uzavretý

Patrik najprv napísal počiatočný návrh, ktorý bol skontrolovaný a upravený tímom, aby mohol byť následne odovzdaný.

User Story: Nástroj na komunikáciu v tíme

Riešitelia: Viktor Beňo, Ľubomír Kurčák

Story Points: 3

Stav: Uzavretý

Viktor a Ľubo sa rozhodovali medzi použitím Slack alebo Discord nástroja. Napokon sa rozhodli pre použitie Slack nástroje, kvôli jednoduchšej integrácii s TFS.

User Story: Hľadanie 3D modelov pre elektrické/logické obvody

Riešitelia: Patrik Tománek, Ľubomír Kurčák

Story Points: 8

Stav: Uzavretý

Patrik pre modely určené na breadboard implementáciu našiel niekoľko voľne dostupných elektrických súčiastok, pričom pri ich testovaní v Unity sa našli viaceré chyby, ako napríklad priesvitné textúry alebo veľký nepomer k veľkostiam jednotlivých súčiastok medzi ostatnými. Ľubomírovi sa nepodarilo nájsť žiadne vyhovujúce modely pre grid implementáciu, pričom ale povedal, že nemá problém ich sám vytvoriť.

User Story: Analýza existujúcich riešení z praxe (fyzických)

Riešitelia: Tomáš Sabo, Viktor Beňo, Erik Paľa

Story Points: 8

Stav: Uzavretý

Analyzovali sme niekoľko reálnych hmotných riešení, pričom sme vypracovali výsledný dokument, v ktorom boli ku každému riešeniu spísané výhody a nevýhody takéhoto riešenia.

User Story: Návrh potencionálnych riešení

Riešitelia: Viktor Beňo, Erik Paľa

Story Points: 5

Stav: Uzavretý

Viktor navrhol riešenie pre grid a Erik pre breadboard. V návrhu sa nachádzali aj nákresy jednotlivých riešení spolu s rôznymi prístupmi, ako tieto nákresy docieliť v našom prototype.

3.3. Šprint 3 (9.11. – 22.11.)

Alias šprintu: Cranberry

Tretí šprint bol zameraný na tvorbu prvého prototypu v Unity. Po prvom a druhom šprinte sme sa dohodli, že naše riešenie bude pozostávať zo simulácie elektrických obvodov v gride. Pre simuláciu

správania elektrických obvodov budeme používať SpiceSharp knižnicu, ktorú sme už implementovali do Unity v prvom šprinte.

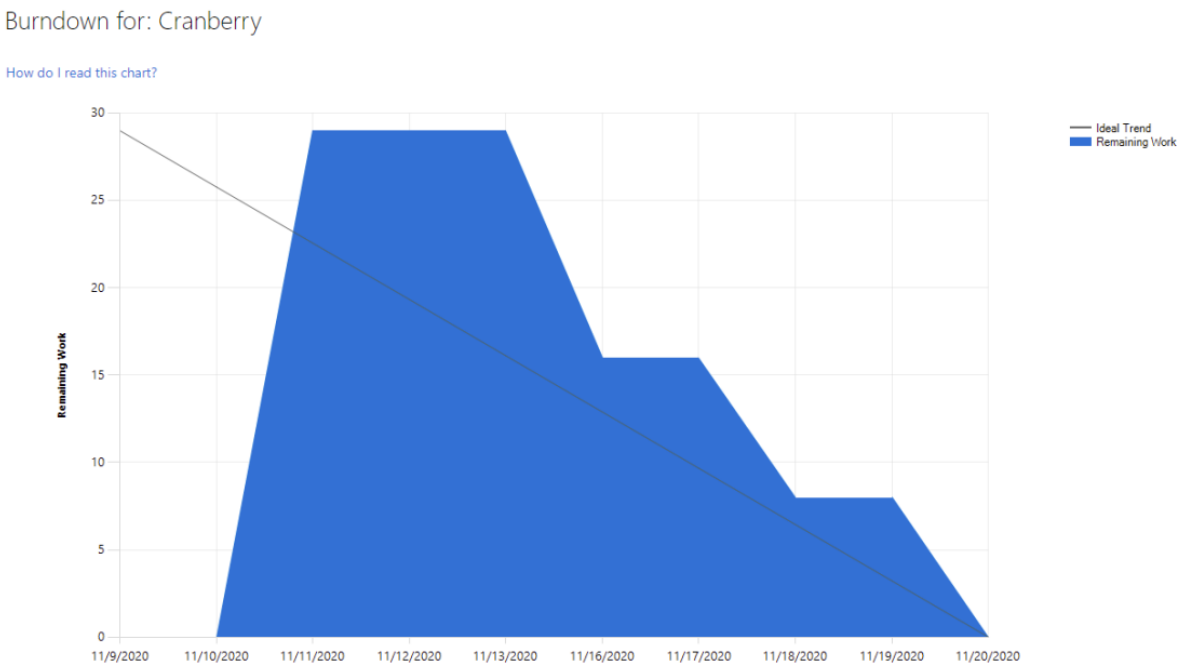
Prvá *User Story* bola zameraná na tvorbu modelov elektrických súčiastok vo forme kociek. Keďže v predošlom šprinte sa nám nepodarilo nájsť vyhovujúce modely pre naše riešenie, Ľubomír Kurčák sa rozhodol, že tieto modely namodeluje sám v 3D grafickom softvéri Blender.

Druhá *User Story* súvisela s prvou, pretože sme mali vytvoriť zoznam všetkých elektrických súčiastok, ktoré budeme v projekte potrebovať. Tieto el. súčiastky museli byť podporované SpiceSharp knižnicou. Túto *User Story* mali na starosti Viktor Beňo a Tomáš Sabo. Až potom mohol Ľubomír Kurčák vytvoriť modely vo forme kociek pre celý zoznam elektrických súčiastok.

Tretia *User Story* bola zameraná na tvorbu prototypu v Unity, konkrétne konštrukcia obvodov v gride. Bolo potrebné vytvoriť mriežku, pre detekciu kolízií s kockami. Takisto sa musela vytvoriť funkcionalita pre ovládanie hernej postavy hráčom, možnosť zobrať kocku zo zeme, otáčať ju v ruke alebo ju z ruky pustiť. Dostali sme za úlohu vytvoriť aj spôsob zvýrazniť kocky, na ktoré sa hráč pozerá, aby vedel, ktorú po stlačení príslušného tlačidla zdvihne. Takisto pokiaľ hráč drží v ruke kocku a stoji pri grid mriežke, je potrebné vytvoriť simuláciu danej kocky v hráčovej ruke, ako bude vyzerať po vložení do gridu. Takže hráč sa môže pozerať priamo do gridu na jednotlivé pozície, do ktorých môže kocka zapadnúť a bude vidieť simuláciu jeho kocky do tohto gridu bez toho, aby ju musel do nej najprv vložiť. Túto *User Story* si zobrali na starosť Patrik Tománek, Ľubomír Kurčák a Tomáš Sabo.

Posledná *User Story* bola zameraná na návrh realizácie pre SpiceSharp. Mali sme si premyslieť celý proces transformovania zloženia nášho grid riešenia až po vygenerovanie kódu pre SpiceSharp. Opätovne sme túto realizáciu mali rozdelenú do 2 častí, jedna časť bola realizácia pre grid riešenie, ktorú mali na starosti Viktor Beňo a druhá časť bola pre breadboard riešenie, ktorú mal na starosti Erik Paľa. Opísanie postupu pre breadboard riešenie bol už ale záložný plán, pokiaľ niekedy v budúcnosti pri práce na projekte zistíme, že grid riešenie nebola dobrá voľba. V takom prípade máme vypracovaný záložný plán pre breadboard riešenie. Súčasťou realizácií mal takisto byť aj obrázok skonštruovaného obvodu v grid/breadbord spolu s kódom, ktorý reprezentuje tento obvod ako vstup do SpiceSharp knižnice.

Na *Obrázok 3* môžeme vidieť *Burndown chart* 3. šprintu. Pri tvorbe tohto grafu sme zabudli zo začiatku inicializovať počet zostávajúcich hodín a inicializovali sme iba počet celkových hodín. Kvôli tomu sa nám cely graf posunul a takisto sa doňho nezaznačil prvý progres nášho šprintu. Našťastie sa tento problém týkal iba prvých 2 dňoch šprintu, kedy sme tento problém odstránili.



Obrázok 3 - Burndown chart 3. šprintu

Tretí šprint sa skladal zo 4 *User Stories:*

User Story: Vytvoriť modely kociek

Riešiteľ: Ľubomír Kurčak

Story Points: 13

Stav: Uzavretý

Ľubomír vytvoril všetky potrebné modely pre náš projekt, modely boli otestované, pričom sa na nich nenašli žiadne nedostatky.

User Story: Navrhnúť zoznam kociek

Riešitelia: Viktor Beňo, Tomáš Sabo

Story Points: 5

Stav: Uzavretý

Viktor a Tomáš vytvorili zoznam všetkých podporovaných elektrických súčiastok SpiceSharp knižnicou a vybrali z nich všetky potrebné pre realizáciu nášho projektu.

User Story: Konštrukcia obvodov v gride

Riešitelia: Patrik Tománek, Tomáš Sabo, Ľubomír Kurčák

Story Points: 15

Stav: Uzavretý

Bol vytvorený prvý prototyp v Unity prostredí, kde hráč mohol ovládať vlastnú postavu, mohol zdvihnúť ľubovoľnú kocku, ktorá reprezentovala elektrickú súčiastku. Túto kocku mohol následne v ruke rotovať alebo pustiť späť na zem. Hráč takisto mal vyznačenú kocku, na ktorú sa pozerá. Následne hráč mohol túto kocku vložiť do mriežky, ktorá jednak reagovala na to, na ktorú pozíciu sa hráč díva, v takom prípade sa simulovala kocka v jeho ruke do gridu, kedy mohol túto kocku doň vložiť stlačením príslušného tlačidla. Takisto každá pozícia v gride, určená pre kocku reagovala na kolízie s týmito kockami. To znamená, že pokiaľ hráč pustí kocku z ruky a kocka padne na grid, spadne do prvej možnej diery.

User Story: Návrh realizácie pre SpiceSharp

Riešitelia: Erik Paľa, Viktor Beňo

Story Points: 11

Stav: Uzavretý

Erik navrhol realizáciu breadboard riešenia pre SpiceSharp a Viktor pre grid riešenie. Návrh obsahoval celý postup od zloženia obvodu v breadboard/grid až po vygenerovanie kódu kompatibilný so SpiceSharp knižnicou. Obidvaja sa rozhodli ako ukážkový príklad vytvoriť obvod pre žiarovku. Návrhy takisto obsahovali obrázok ako takýto obvod vyzerá v danom riešení spolu s postupom, ako sa z daného riešenia vygeneruje kód pre SpiceSharp.

3.4. Šprint 4 (23.11. – 6.12.)

Alias šprintu: Date

Štvrtý šprint bol zameraný hlavne na nadstavbu nášho prvého prototypu. V tomto šprinte sme začali s integráciou *VR Interaction Framework* balíka do nášho projektu. Tento balík má za úlohu všetky elementy spojené v virtuálnou realitou. Takisto sme do vytvoreného prototypu implementovali nami vytvorené *SpiceSharp* demo. Posledná časť šprintu bola venovaná pokračovaní v tvorbe modelov pre náš projekt.

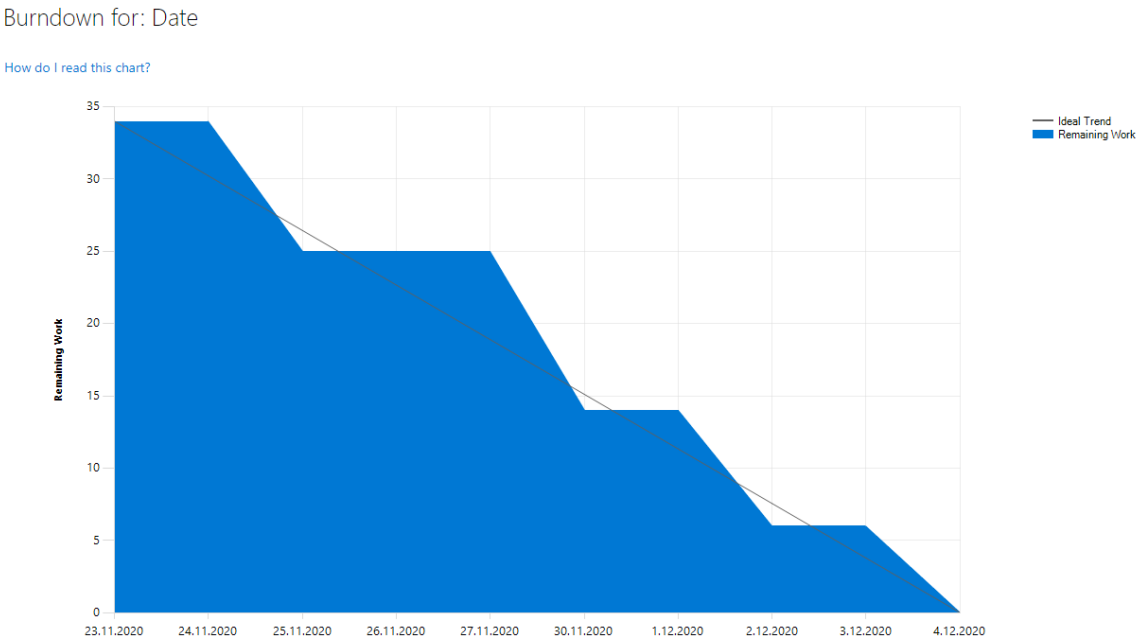
Prvá *User Story* bola zameraná na integráciu *VR Interaction Framework* balíka do nášho projektu. Súčasťou tejto integrácie bolo vytvorenie vlastnej scény s použitím objektov z *VR Interaction Framework*. Následne sme integrovali interakciu s týmito objektami, pričom sme okrem objektov ponúkaných v tomto balíku integrovali aj interakciu pre naše modely, konkrétne sme vytvorili prvý testový model, ktorý bude použitý v našom projektu tak, aby bol kompatibilný s VR. Posledná časť tejto *User Story* bola zameraná na porovnávaní nami vytvorených *snap zones* so *snap zones* z *VR Interaction Framework.*

Druhá *User Story* bola zameraná na tvorbu prvého *SpiceSharp* dema. Vytvorili sme skript, ktorý vedel meniť parametre elektrického obvodu počas jeho simulácie. Skontrolovali sme rozdiely medzi

statickou a prechodnou simuláciou obvodov. Posledná časť tejto *User Story* bola venovaná tvorbe vlastnej súčiastky v *SpiceSharp* knižnici, počas ktorej sme zistili, že tvorba vlastných súčiastok je omnoho náročnejšia než sa zdalo, a pred tým ako sa k nej dostaneme, je potrebné aby sme si naštudovali jadro *SpiceSharp* knižnice, konkrétne funkcionalitu matíc a matematického modelu.

Tretia *User Story* bola zameraná na rozšírení typov modelov o ďalšie súčiastky. Vytvorili sme niekoľko nových súčiastok, konkrétne sme mali vytvárať jednotlivé súčiastky ako 3D modely a reliéfové modely pre elektrické súčiastky.

Na Obrázok 4 môžeme vidieť *Burndown chart* 4. šprintu. Práca na tomto šprinte bola zo začiatku trošku pomalšia ako sme boli zvyknutí počas práce na predošlých šprintov. Našťastie sme sa ale po prvotných problémov vrátili k predošlému pracovnému tempu a šprint sme riadne dokončili.



Obrázok 4 - Burndown chart 4. šprintu

Štvrtý šprint sa skladal z 3 *User Stories:*

User Story: Vytvoriť scénu vo VR Interaction Framework

Riešiteľ: Patrik Tománek

Story Points: 8

Stav: Uzavretý

Patrik vytvoril scénu v Unity s objektami z *VR Interaction Framework*, otestoval funkcionalitu tohto balíka rôznymi interakciami so spomínanými objektami, ako aj vytvoril interakciu s našimi modelmi, ktoré budeme používať v našom projekte. Nakoniec porovnal funkcionalitu *snap zones* z *VR Interaction Framework so snap zones*, ktoré sme si sami v projekte vytvorili. Počas tohto testovania sme zistili, že lepšie bude používať funkcionalitu z *VR Interaction Framework*, pretože nemáme doposiaľ dostatočné skúsenosti s tvorbou VR hier a naše *snap zones* by mohli v budúcnosti vytvárať mnohé problémy.

Funkcionalita ponúknutá z *VR Interaction Framework* je pre náš projekt dostačujúca, aj keď bude potrebné čiastočné prispôsobenie, v respektíve nadstavba tejto funkcionality pre náš projekt.

User Story: Vytvoriť demo si SpiceSharp

Riešiteľ: Viktor Beňo, Erik Paľa, Tomáš Sabo

Story Points: 17

Stav: Uzavretý

Viktor, Erik a Tomáš vytvorili prvý demo skript, ktorí simuloval elektrický obvod za pomoci *SpiceSharp* knižnice v našom projekte. Tento skript bol takisto schopný meniť parametre elektrického obvodu za behu jeho simulácie. Takisto bolo potrebné odmeranie časov statických a prechodných simulácií, aby sme vedeli, ku ktorej simulácií sa máme v budúcnosti prikláňať, aby nedošlo k narušeniu optimalizácií nášho projektu. Pri meraní týchto časov sme ale zistili, že medzi tými časmi je minimálny rozdiel a preto prechodná simulácia bude pre náš projekt výhodnejšie v prípadoch, v ktorých potrebujeme meniť parametre elektrického obvodu za jeho chodu namiesto toho, aby sme ho celý vypli a zapli na novo. Takisto nie len, že rozdiel časov bol minimálny, ale takisto sme zistili, že *SpiceSharp* knižnica je časovo veľmi efektívna pre simuláciu elektrických obvodov. Jediný problém, na ktorý sme so *SpiceSharpom* narazili bol ten, že pre simuláciu elektrického obvodu sme museli v Unity spustiť nové vlákno, aby sa nenarušil priebeh hry, a teda hra nezačala sekať. Dohodli sme sa na tom, že všetky výpočty spojené so *SpiceSharpom* budú behať na vlastnom vlákne nezávisle od vlákna, ktoré je používané pre *Unity API*. Ako posledné sme mali vytvoriť vlastnú súčiastku v *SpiceSharp*, pretože táto knižnica neobsahuje všetky súčiastky, ktoré budeme v našom projekte potrebovať, ale poskytuje nám možnosť si tieto súčiastky vytvoriť. Počas tvorby vlastnej súčiastky sme ale zistili, že tvorba ďalších, zložitejších súčiastok bude omnoho náročnejšia ako naša prvá súčiastka. Preto bude potrebné sa najprv venovať dôkladnej analýzy jadra *SpiceSharp* knižnice, konkrétne bližšie pochopiť funkcionalitu matíc a matematického modelu.

User Story: Rozšíriť typy modelov o ďalšie súčiastky

Riešiteľ: Ľubomír Kurčák

Story Points: 7

Stav: Uzavretý

Ľubomír pokračoval v tvorbe modelov pre náš projekt. Konkrétne vytváral nové reliéfové ako aj 3D modely elektrických súčiastok pre náš projekt. V tomto šprinte po dokončení tejto úlohy sme sa s *Product Ownerom* dohodli, že už nebudeme ďalej vytvárať reliéfové modely a budeme sa zameriavať na tvorbu 3D modelov.

3.5. Šprint 5 (7.12. – 14.12.)

Alias šprintu: Elderberry

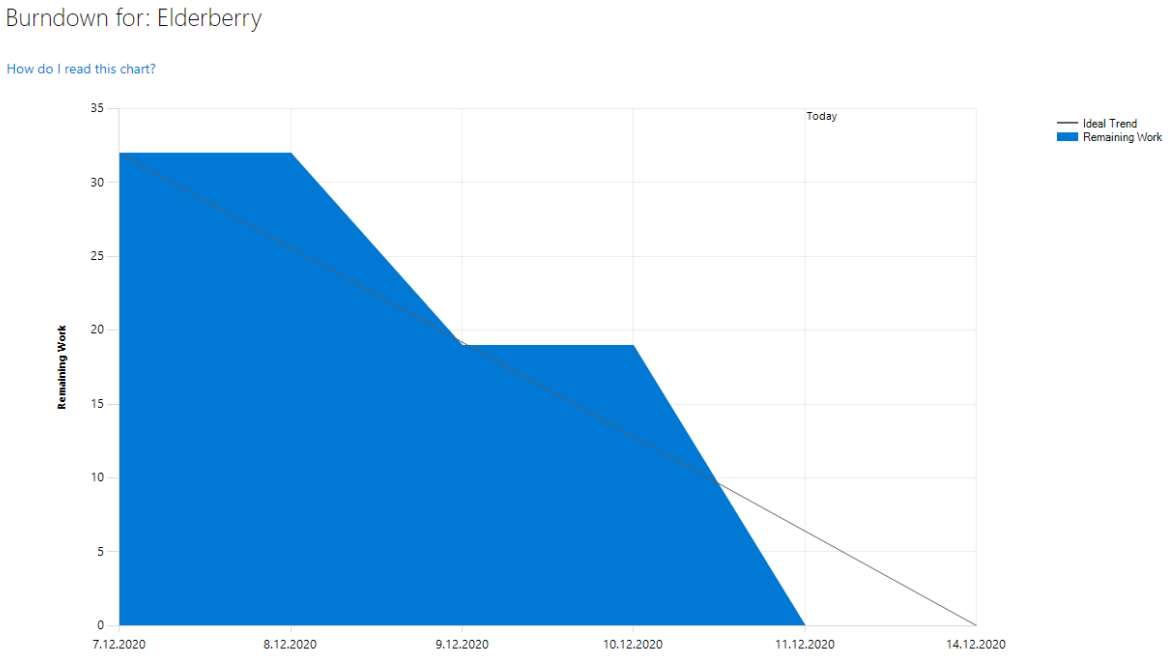
Piaty a posledný šprint zimného semestra bol venovaný analýze synchronizačných frameworkov, za účelom nájdenia najoptimálnejšieho frameworku pre náš kolaboratívny projekt.

Posledná časť šprintu sa zaoberala pokračovaním práce s VR Interaction Frameworkom (VRIF), a teda rozšírenie funkcionality virtuálnej reality v našom projekte. Táto úloha priamo nadväzovala na úlohu z predošlého šprintu. Keďže tento šprint trval iba jeden týždeň, vypracovávali sme iba tieto 2 *User Stories*.

Prvá *User Story* bola zameraná na analýzu synchronizačných frameworkov, konkrétne sme analyzovali frameworky ako: Photon Bolt, Smartfox Server, Photon Unity Networking 2. Následne bolo vytvorené vyhodnotenie týchto analýz za účelom nájdenia najoptimálnejšieho riešenia pre náš projekt. Okrem synchronizácie pohybu a prostredia našou úlohou bola takisto aj analýza synchronizácie *voice chat*u.

Druhá *User Story* bola zameraná na pokračovanie implementácie *VR Interaction Framework* do nášho projektu. Táto *User Story* priamo nadväzovala na predošlú *User Story* zo šprintu č.4. Do projektu boli implementované modifikované *Snap Zones,* pridaná integrácia s našími modelmi elektrických súčiastok. Posledná časť tejto *User Story* sa zaoberala kontrolou spojenia medzi jednotlivými elektrickými súčiastkami umiestenými v *grid boarde.* Na koniec vznikol 2. prototyp nášho projektu, ktorý podporoval virtuálnu realitu.

Na Obrázok 5 môžeme vidieť *Burndown chart 5. šprintu.* Keďže tento šprint trval iba jeden týždeň, počas ktorého sme vypracovali 2 *User Stories*, preto graf obsahuje iba 2 hlavné skoky.



Obrázok 5 - Burndown chart 5. šprintu

User Story: Analýza synchronizačných frameworkov

Riešiteľ: Ľubomír Kurčák, Viktor Beňo, Erik Paľa, Tomáš Sabo

Story Points: 8

Stav: Uzavretý

Ľubomír analyzoval SmartFox server, Viktor analyzoval Photon Bolt a Realtime Cloud, Tomáš analyzoval Photon Unity Networking 2. Erik následne vytvoril vyhodnotenie všetkých týchto analýz

frameworkov. Z tejto analýzy vyplývalo, že do nášho projektu budeme najskôr implementovať Photon Bolt, pretože sa javí ako najlepší kandidát spomedzi konkurencie.

User Story: VR Interaction Framework 2

Riešiteľ: Patrik Tománek

Story Points: 8

Stav: Uzavretý

Patrik pokračoval s implementovaním *VR Interaction Framework* funkcionalít do nášho projektu. Keďže sme sa v predošlom šprinte rozhodli o používaní *Snap Zones* funkcionality z tohto frameworku namiesto používania nami implementovanej funkcionality bolo potrebné danú funkcionalitu prispôsobiť nášmu projektu. Prispôsobenie sa týkalo pridania offsetu pre rotácie elektrických súčiastok po tom ako sú vložené do *grid boardu*. Integrovala sa podpora virtuálnej reality pre ďalšie modely elektrických súčiastok, ako aj prvá verzia kontroly spojenia medzi jednotlivými súčiastkami vloženými v *grid boarde*. Výsledkom tejto *User Story* bolo vytvorenie 2. prototypu nášho projektu, v ktorom bola podpora pre virtuálnu realitu.

3.6. Šprint 6 (26.2. – 12.3.)

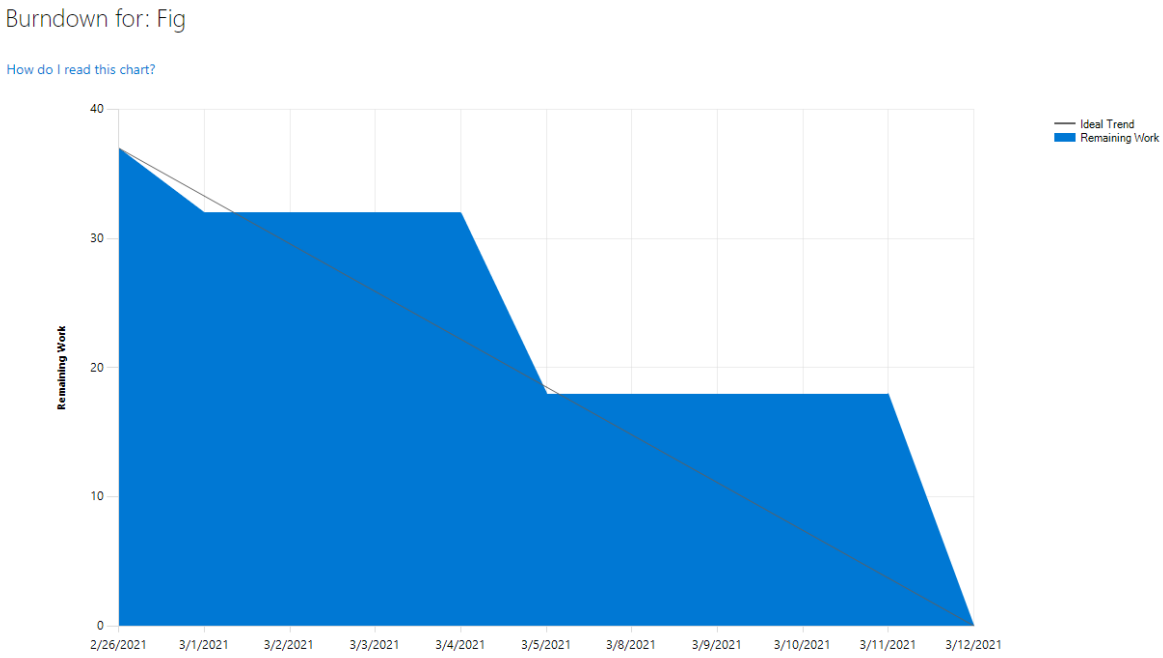
Alias šprintu: Fig

Šiesty a zároveň prvý šprint letného semestra bol venovaný integrácie *SpiceSharp-u* do nášho prototypu a zároveň vytvoreniu dokumentácie k existujúcim častiam prototypu.

Prvá časť šprintu sa zaoberala integráciou *SpiceSharp-u* do nášho prototypu, konkrétne sme najskôr mali analyzovať vstupy do *SpiceSharp-u* a následne sme mali implementovať prekladač obvodov simulácie vytvorených v prototype.

Druhá a posledná časť šprintu sa zaoberala tvorbou dokumentácie k už existujúcim častiam prototypu, konkrétne sme mali zdokumentovať jednotlivé súčiastky obsiahnuté v prototype, zdokumentovať taktiež ich 3D modely ako aj proces vytvárania konektivít medzi súčiastkami pri ich manipulácie v gride za účelom vytvorenia prvotnej používateľskej príručky.

Na Obrázok 6 - Burndown chart 6. šprintuObrázok 6 môžeme vidieť Burndown chart 6. šprintu. Počas letného semestra sme sa rozhodli vytvárať razantnejšie kroky počas implementácie, a teda stretávať sa menej často pričom ale naraz skontrolujeme väčšie progresy.



Obrázok 6 - Burndown chart 6. šprintu

User Story: Integrácia Unity Prototypu do SpiceSharp

Riešiteľ: Viktor Beňo, Patrik Tománek

Story Points: 8

Stav: Uzavretý

Viktor na začiatku analyzoval štruktúru vstupov do *SpiceSharp-u*, pričom vytvoril základnú štruktúru, ktorú následne Patrik implementoval, a teda vytvoril prekladač obvodov vytvorených v prototype na vstupy pre *SpiceSharp.*

User Story: Vytvorenie dokumentácie k existujúcim častiam prototypu

Riešiteľ: Erik Paľa, Ľubomír Kurčák, Patrik Tománek

Story Points: 8

Stav: Uzavretý

Účelom tejto úlohy bolo vytvorenie prvotnej verzie používateľskej príručky. Erik mal na starosti zdokumentovať všetky súčiastky v našom prototype, vytvorenie ich zoznamu spolu so stručným opisom jednotlivých podporovaných súčiastok. Následne Ľubomír vytvoril zoznam modelov spolu s ich dokumentáciou ako aj dokumentáciou tvorby jednotlivých modelov. Na koniec vytvoril používateľsky katalóg pre súčiastky. Na koniec Patrik vytvoril dokumentáciu konektivít súčiastok v rámci logickej reprezentácie elektrického obvodu.

3.7. Šprint 7 (13.3. – 26.3.)

Alias šprintu: Grapefruit

Siedmy šprint bol zložený z 4 častí, a to vypracovanie návrhu simulovaného vozidla, implementácia *SpiceSharpParser-a*, vytvorenie modelu autíčka a nakoniec analýza fyzikálnej simulácie vozidla.

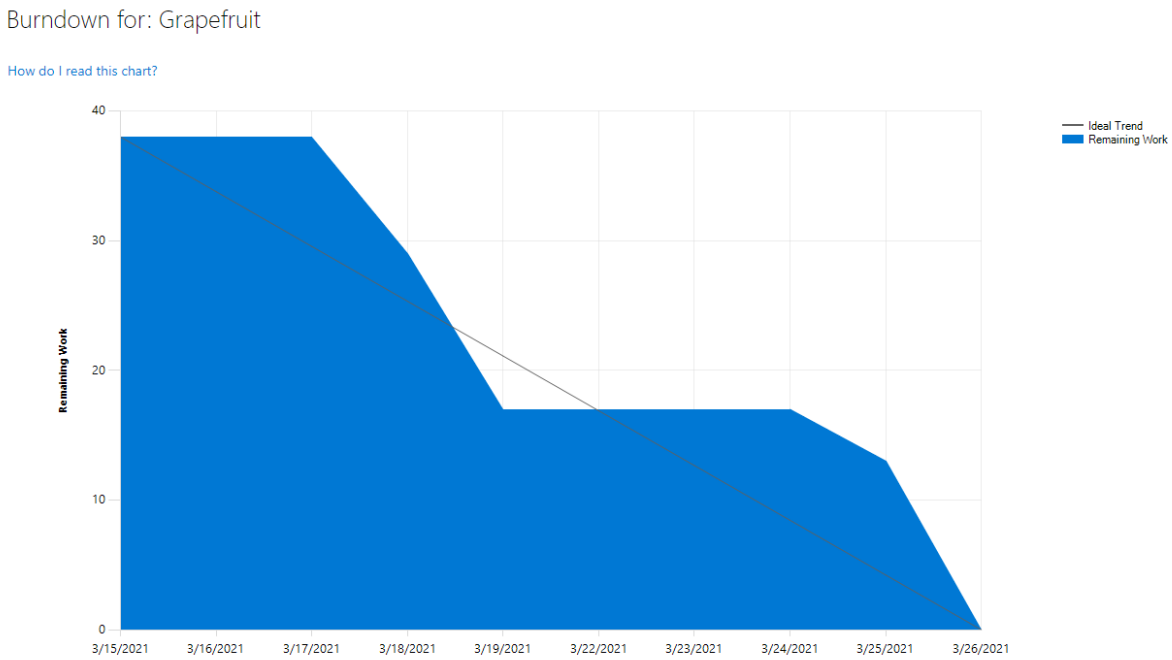
Prvá časť sa zaoberala vypracovaním návrhu simulovaného vozidla, konkrétne vytvorenie návrhu pre hlavnú štruktúru vozidla ako aj pre samostatné súčiastky.

Druhá časť sa zaoberala implementáciou *SpiceSharpParser-a* do nášho prototypu. Na začiatku bolo potrebná komunikácia s autorom tejto knižnice za účelom zistenia, či knižnica bude aktualizovaná a kompatibilná sú súčasnou verziou *SpiceSharp-u.* Ako posledné sme mali implementovať demo scénu, v ktorej bude ukázaná funkcionalita tejto knižnice.

Tretia časť bola vytvorená tvorby modelu autíčka. Najprv sme mali vytvoriť zoznam už existujúcich modelov a vytvorenia .FBX modelov pripravených na ich import.

Štvrtá a posledná časť tohto šprintu bola zameraná na analýzu fyzikálnych simulácií vozidla, konkrétne sme mali analyzovať doposiaľ vytvorené riešenia a vybrať to najvhodnejšie.

Na Obrázok 7 môžeme vidieť Burndown chart 7. šprintu, pokračovali sme v predošlom systéme, a teda menej stretnutí ale väčšie progresy.



Obrázok 7 - Burndown chart 7. šprintu

User Story: Vypracovanie návrhu simulovaného vozidla

Riešiteľ: Ľubomír Kurčák

Story Points: 5

Stav: Uzavretý

Ľubomír vytvoril návrh základnej konštrukcie vozidla s podporou pre modulárne súčiastky a následne vytvoriť návrh pre samotné súčiastky, ich kategorizáciou na časti, ktoré sú mechanické, resp. fyzické a časti, ktoré súvisia s elektrickým obvodom.

User Story: Implementácia Parsera SpiceSharp

Riešiteľ: Viktor Beňo, Patrik Tománek

Story Points: 13

Stav: Uzavretý

Na začiatku Viktor kontaktoval autora knižnice SpiceSharpParser za účelom zistenia, či má v záujme pokračovať a pracovať na danej knižnice. Problémom bol fakt, že súčasná verzia tejto knižnice nie je kompatibilná so súčasnou verziou SpiceSharp-u, ktorú používame v našom prototype. Na koniec autor knižnice odpísal a ubezpečil náš, že knižnicu v krátkej dobe aktualizuje. Po aktualizácií bola táto knižnica Viktorom integrovaná do projektu a Patrik vytvoril demo scénu, v ktorej bola zobrazená základná funkcionalita tejto knižnice.

User Story: Vytvorenie modelu autíčka

Riešiteľ: Ľubomír Kurčák

Story Points: 9

Stav: Uzavretý

Ľubomír našiel vhodné low poly modely a analyzoval ich výhody, resp. nevýhody z hľadiska vizuálneho spracovania a funkcionalít potrebných animácií. Následne vytvoril .FBX modely pripravených na import do prototypu.

User Story: Analýza fyzikálnej simulácie vozidla

Riešiteľ: Erik Paľa

Story Points: 8

Stav: Uzavretý

Erik analyzoval všetky nájdené riešenia ohľadom fyzikálnej simulácie vozidla, spísal ich výhody, resp. nevýhody vzhľadom na jednotlivé faktory.

3.8. Šprint 8 (27.3. – 9.4.)

Alias šprintu: Huckleberry

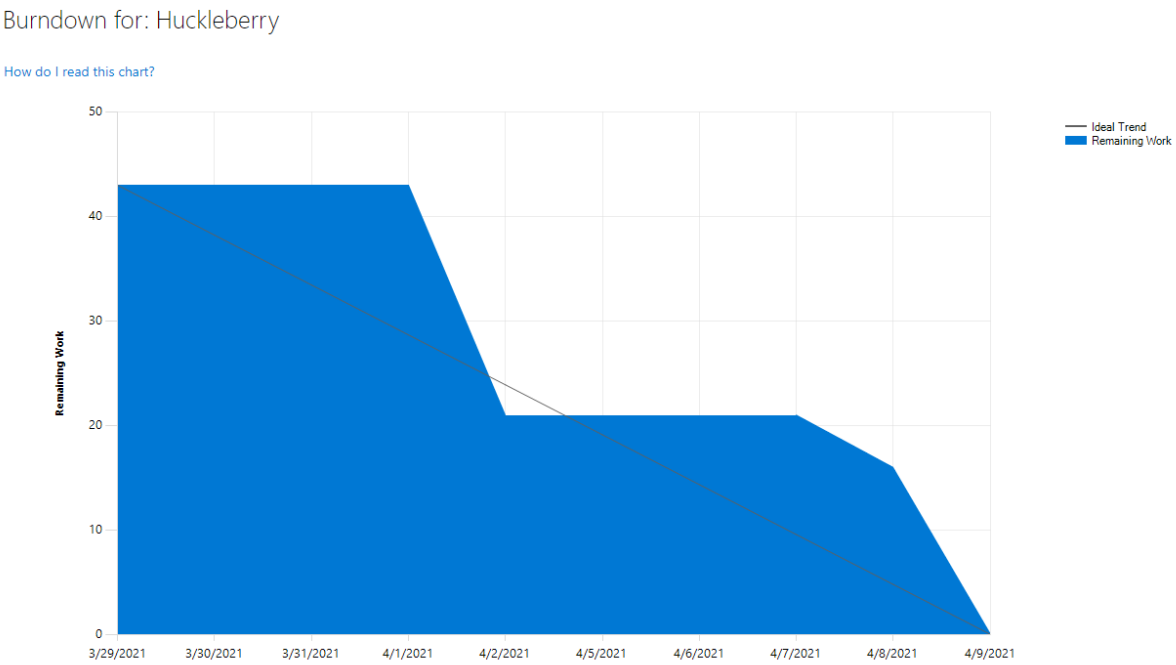
Osmy šprint bol venovaný návrhu scény pre modifikáciu vozidla, výpis Spice kódu pre súčiastky a vytvorenie príkladov pokročilejších obvodov.

Prvá časť sa zaoberala návrhom scény pre modifikáciu vozidla, konkrétne rozhodnúť o spôsobe samotnej modifikácie, vytvorenie návrhu opisu takejto scény spolu s grafickými ilustráciami ako aj schémami existujúcich obvodov.

Druhá časť sa zaoberala výpisom Spice kódu pre súčiastky v gridboarde počas simulácie. Mala sa vytvoriť interaktívna tabuľa, ktorá má prístup k interným Spice kódom simulovaných súčiastok.

Tretia a posledná časť šprintu sa zaoberala vytvorením príkladov pokročilejších obvodov, konkrétne sa jednalo o vytvorenie obvodu pre oscilátor, buzzer a silnejší, resp. slabší motor.

Na Obrázok 8 môžeme vidieť Burndown chart 8. šprintu, ktorý ma veľmi podobnú štruktúru ako predošlé.



Obrázok 8 - Burndown chart 8. šprintu

User Story: Návrh scény pre modifikáciu vozidla

Riešiteľ: Ľubomír Kurčák, Erik Paľa

Story Points: 8

Stav: Uzavretý

Ľubomír rozhodol o spôsobe modifikácie vozidla, vytvoril opis scény takého spôsobu spolu s grafickými ilustráciami. Erik Paľa nakoniec vytvoril schémy existujúcich obvodov, ktoré budú očakávané od používateľa

User Story: Výpis Spice kódu pre súčiastky

Riešiteľ: Patrik Tománek

Story Points: 5

Stav: Uzavretý

Patrik vytvoril interaktívnu tabuľu a vložil ju do nášho prototypu. Táto tabuľa mala prístup k interným Spice kódom simulovaných súčiastok, ktorý bude následne zobrazený používateľovi.

User Story: Vytvorenie príkladov pokročilejších obvodov

Riešiteľ: Viktor Beňo

Story points: 13

Stav: Uzavretý

Viktor vytvoril obvody pre oscilátor, buzzer a silnejší, resp. slabší motor, pričom otestoval ich funkčnosť. Silnejší motor bol ovládaný pomocou mosfetu a slabší motor pomocou obvodu.

3.9. Šprint 9 (10.4. – 25.4.)

Alias šprintu: Kiwi

Deviaty šprint bol venovaný vytvoreniu scény pre modifikáciu vozidla, modifikácie modelu vozidla, testovaniu oscilátorov a kondenzátorov *SpiceSharp-u* a doplnený modelov súčiastok.

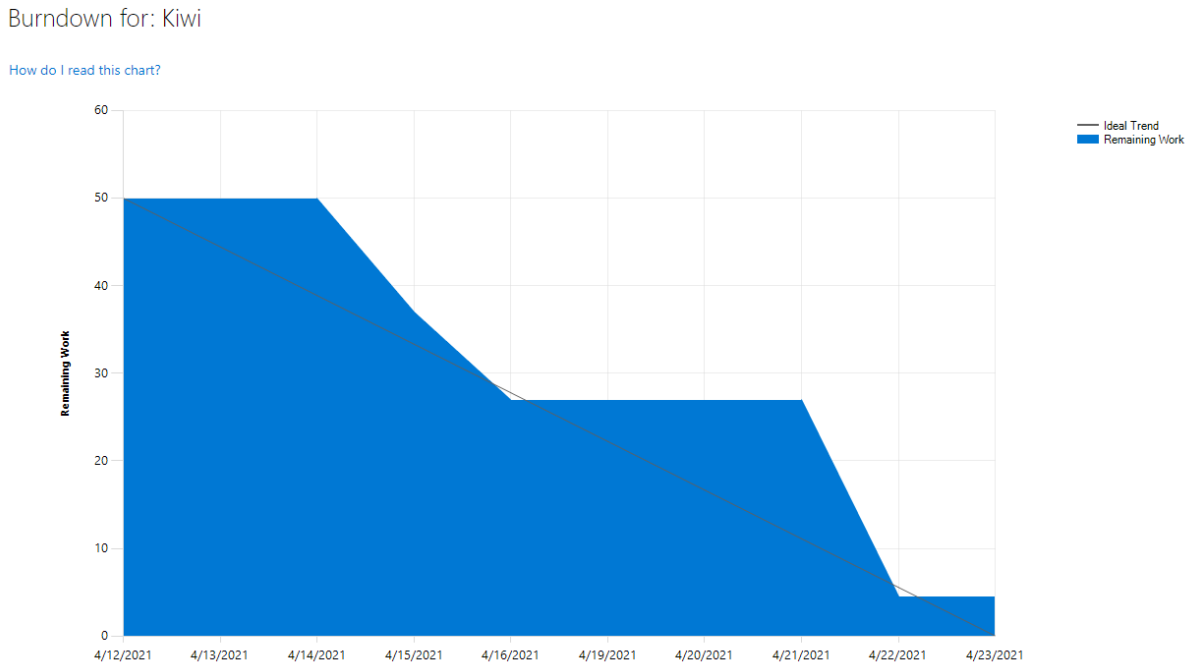
Na začiatku sa pokračovalo v úlohe z predošlého šprintu, konkrétne vytvorenie scény pre modifikáciu vozidla.

Druhá časť sa zaoberala samotnou modifikáciou vozidla. Mali sme analyzovať spracovanie animácií a animačných parametrov. Následne sme mali integrovať model spolu s fyzikálnou simuláciou a rozdeliť tento model na interaktívne a statické súčiastky.

Tretia časť sa zaoberala testovaním oscilátorov a kondenzátorov *SpiceSharp-u*. Najprv sme mali vytvoriť príklady kódov pre oscilátor a kondenzátor a vyskúšať tieto obvody v LTspice.

Štvrtá časť sa zaoberala doplnením modelov súčiastok, konkrétne sa jednalo o model batérie, cievky, kondenzátora, motora, reproduktora a oscilátora.

Na Obrázok 9 môžeme vidieť Burndown chart 9. šprintu. Dôvod prečo nie je dokončený bol ten, že sme zabudli do konca šprintu ukončiť jednu úlohu v Azure DevOps.



Obrázok 9 - Burndown chart 9. šprintu

User Story: Vytvorenie scény pre modifikáciu vozidla

Riešiteľ Patrik Tománek

Story Points: 11

Patrik vytvoril interaktívne komponenty vozidla a umožnil vkladanie týchto komponentov do gridboardu, pričom sa pre tento komponent načítal daný obvod.

User Story: Modifikácia modelu vozidla

Riešiteľ: Erik Paľa, Patrik Tománek

Story Points: 8

Erik na začiatku analyzoval spracovanie animácií a animačných parametrov používaného modelu vozidla a následne Patrik integroval na tento model fyzikálnu simuláciu a rozdelil ho na interaktívne a statické súčiastky

User Story: Testovanie oscilátorov a kondenzátorov *SpiceSharp-u*

Riešiteľ: Viktor Beňo, Erik Paľa

Story Points: 13

Viktor vytvoril príklady kódov pre oscilátor a kondenzátor a Erik tieto kódy otestoval a odsimuloval v LTspice.

User Story: Doplnenie modelov súčiastok

Riešiteľ: Ľubomír Kurčák

Story Points: 8

Ľubomír vytvoril modely pre batériu, cievku, kondenzátor, motor, reproduktor a oscilátor.

3.10. Šprint 10 (26.4. – 9.5.)

Alias šprintu: Ligonberry

Desiaty šprint bol venovaný vytvoreniu symbolických komponentov vozidla, integrácie modelu vozidla do scény pre jeho modifikáciu, vytvoreniu vzorových obvodov s použitím oscilátoru a vytvorenie osciloskopu.

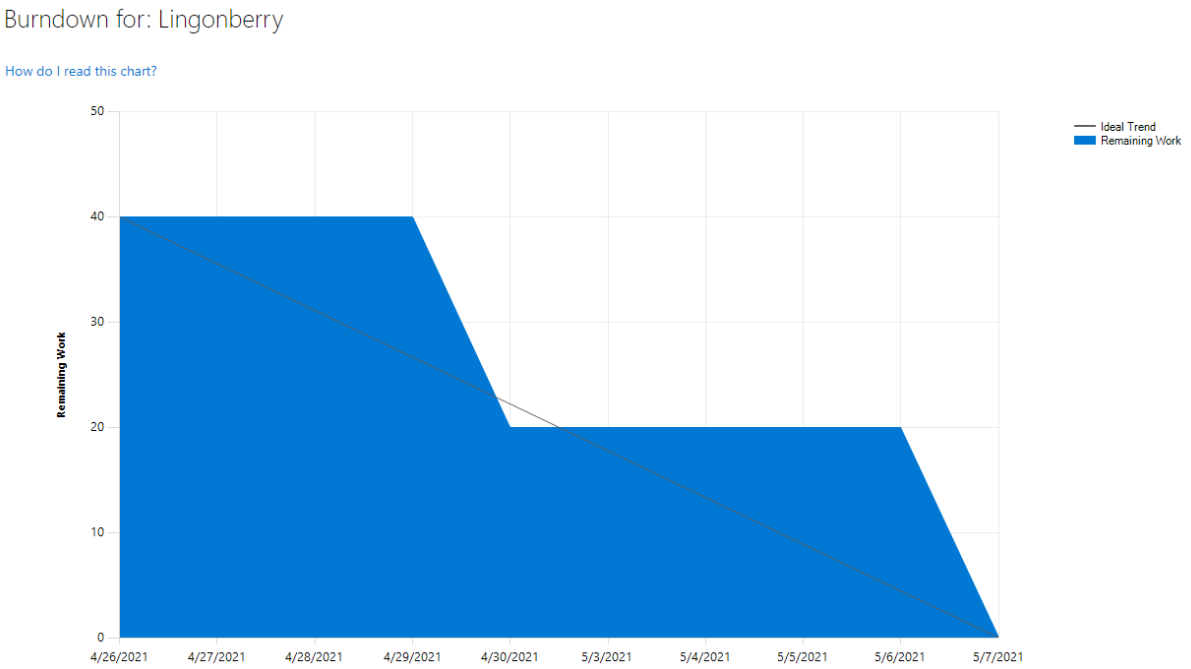
Prvá časť sa zaoberala tvorbou symbolických komponentov vozidla, ktoré budú používané ako súčiastky v gridboarde, konkrétne tvorbe symbolických súčiastok pre motor, svetlá a klaksón.

Druhá časť sa zaoberala integráciou modelu vozidla do scény pre jeho modifikáciu, náhradou kociek za reálne súčiastky vozidla a implementovaniu riešenia pre stratenie súčiastky.

Tretia časť sa zaoberala vytvorením vzorových obvodov s použitím oscilátoru, konkrétne sme mali vytvoriť obvody, ktoré následne otestujeme a vytvoríme pre tieto obvody schematický zápis.

Posledná časť sa zaoberala tvorbou osciloskopu, a teda vytvoreniu záznamu osciloskopických dát a následne zobrazenie týchto dát.

Na Obrázok 10 môžeme vidieť Burndown chart 10. šprintu, v tomto šprinte sme mali iba 2 veľké progresy, ktoré stačili na dokončenie šprintu.



Obrázok 10 - Burndown chart 10. šprintu

User Story: Vytvorenie symbolických komponentov vozidla

Riešiteľ: Ľubomír Kurčák

Story Points: 5

Ľubomír vytvoril symbolické komponenty pre motor, svetlá a klaksón. Tieto symbolické komponenty budú následne používané v gridboarde.

User Story: Integrácia modelu vozidla do scény pre modifikáciu

Riešiteľ: Patrik Tománek

Story Points: 13

Patrik integroval model vozidla do scény pre jeho modifikáciu, pričom nahradil kocky za reálne súčiastky tohto vozidla a implementoval riešenie stratenia súčiastky a jej návratu do vozidla.

User Story: Vytvorenie vzorových obvodov s použitím oscilátoru

Riešiteľ: Viktor Beňo, Erik Paľa

Story Points: 13

Viktor vytvoril ukážkové obvody s použitím oscilátoru, pričom tieto obvody otestoval a následne pre tieto obvody Erik vytvoril schematické zápisy.

User Story: Vytvorenie osciloskopu

Riešiteľ: Erik Paľa

Story Points: 8

Erik vytvoril záznam osciloskopických dát a následne tento záznam zobrazil v prototype.

3.11. Šprint 11 (9.5. – 14.5.)

Alias šprintu: Mango

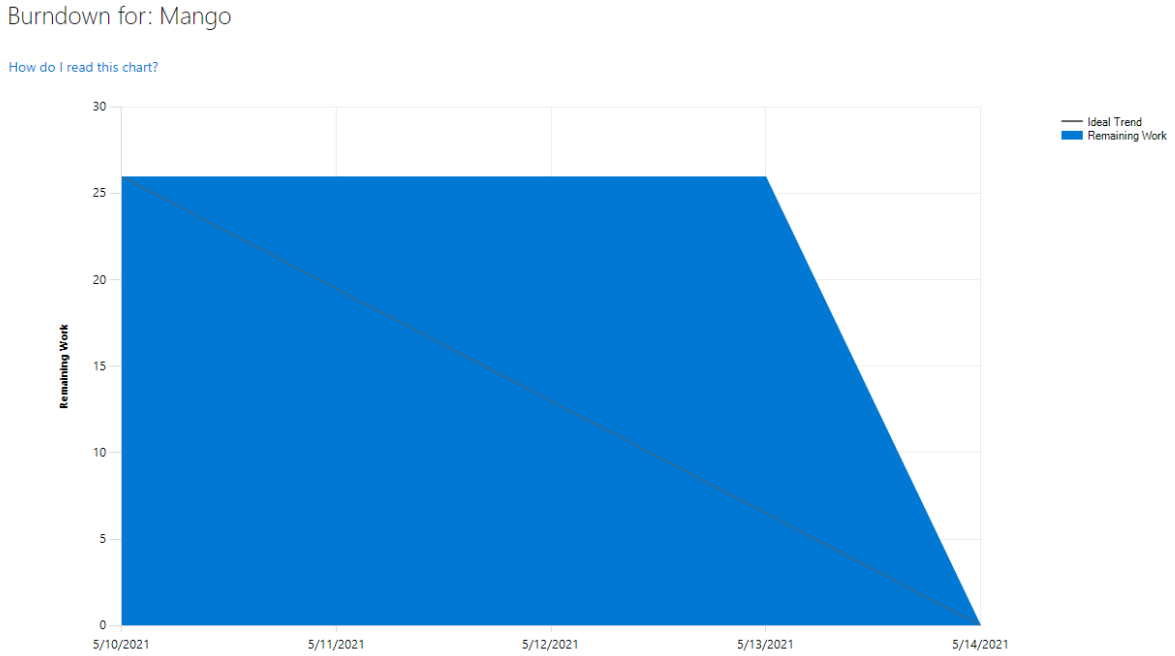
Jedenásty a zároveň posledný šprint tímového projektu bol venovaný posledným úpravám prototypu, konkrétne finalizácie úloh zapájania obvodov do vozidla, optimalizácie používateľského zážitku. Okrem posledných úprav sme mali za úlohu vytvoriť aj používateľskú príručku k nášmu prototypu.

Prvá časť sa zaoberala tvorbe používateľskej príručky. Táto príručka má obsahovať základné ovládanie, postup pri tvorení obvodov a ovládaní vozidla.

Druhá časť sa zaoberala finalizáciou úloh zapájania obvodov. Mali sme implementovať vybrané schémy pre motor, klaksón a svetlá do vozidla a otestovať korektnú funkcionalitu zapojených obvodov.

Tretia časť sa venovala samotnej optimalizácií používateľského zážitku, v tejto časti sa upravovali rôzne snap zones pre zjednodušenie interakcie nie len vo VR ale aj v desktop verzií, zjednodušenie manipulácie súčiastok vo vozidle a vypnutie nepotrebných kolízií.

Na Obrázok 11 môžeme vidieť Burndown chart 11. šprintu, na tento šprint sme mali iba jeden týždeň, preto sa na obrázku nachádza iba 1 veľký skok.



Obrázok 11 - Burndown chart 11. šprintu

User Story: Vytvoriť používateľskú príručku

Riešiteľ: Erik Paľa, Ľubomír Kurčák

Story points: 6

Erik vytvoril príručku pre základné ovládanie a tvorenie obvod a Ľubo vytvoril príručku pre ovládanie vozidla. Príručka tvorenia obvodov zahŕňala kompletný tutoriál ako meniť napätie, funkcionalita osciloskopu, vkladanie súčiastok do grid boardu, získanie súčiastok zo snap zones, pridanie nového modelu súčiastky, vybranie súčiastky z auta a načítanie obvodu pre túto súčiastku po jej vložení do grid boardu.

User Story: Finalizácia úloh zapájania obvodov

Riešiteľ: Viktor Beňo, Patrik Tománek

Story Points: 6

Viktor implementoval schémy pre motor, klaksón a svetlá vozidla do prototypu a Patrik testoval korektnú funkcionalitu týchto obvodov, našťastie všetko fungovalo ako má.

User Story: Optimalizácia používateľského zážitku

Riešiteľ: Patrik Tománek, Ľubomír Kurčák

Story Points: 5

Patrik a Ľubomír mali za úlohu optimalizovať celkový používateľský zážitok, preto upravovali niektoré detaily v prototype, ako napríklad úprava snap zones, zjednodušenie manipulácie súčiastok vo vozidle alebo vypnutie nepotrebných kolízií.

4. GLOBÁLNA RETROSPEKTÍVA

Autor kapitoly: Patrik Tománek

Pre lepšie pochopenie globálnej retrospektívy nášho tímu sme sa rozhodli rozdeliť túto kapitolu na retrospektívy k jednotlivým šprintom. Každá retrospektíva sa skladala z 3 častí, prvou časťou boli pozitíva šprintu, následne negatíva šprintu a celkové zhodnotenie s prípadnými dohodami na zmeny v pracovnom nasadení.

4.1. RETROSPEKTÍVA K ŠPRINTU 1 (26.10.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili

- Pracovali sme priebežne na úlohách, čiže už v polovici šprintu každý člen tímu prezentoval jeho progres na jemu pridelenej úlohe
- • Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli načas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Pracovné nasadenie na ďalšom šprinte ponecháme zatiaľ nezmenené po prvom šprinte

4.2. RETROSPEKTÍVA K ŠPRINTU 2 (9.11.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Úlohy, ktoré mali prioritu, že museli byť splnené do prvej polovici šprintu sa bez problémov splnili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli načas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Tento šprint sa principiálne podobal prvému šprintu, preto sme pracovné nasadenie ponechali rovnaké z toho dôvodu, že v 1. retrospektíve neboli nájdené žiadne negatíva
- Rozhodli sme sa opätovne pokračovať v rovnakom pracovnom nasadení bez zmien

4.3. RETROSPEKTÍVA K ŠPRINTU 3 (23.11.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Počas tohto šprintu sme mali vypracovať aj dokument k riadeniu a inžinierskemu dielu, pričom každý člen z tímu na ňom usilovne pracoval
- Všetky časti obidvoch dokumentov boli vzájomne diskutované a prípadne upravované
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom

- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli načas dokončené
- Dokument k riadeniu a inžinierskemu dielu bol načas dokončený

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- V tomto šprinte sme po prvý krát začali aj s implementáciou a nie len analýzou. Vytvárali sme prvý prototyp nášho riešenia, pričom celý proces prebehol bez problémov.
- Rozhodli sme sa nerobiť zmeny v pracovnom nasadení

4.4. RETROSPEKTÍVA K ŠPRINTU 4 (7.12.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli načas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- V tomto šprinte sme po prvý krát integrovali prvé SpiceSharp demo do nášho projektu, ako aj VR Interaction Framework. Nevyskytli sa žiadne značné problémy, jediné na čo sme prišli je, že tvorba vlastných súčiastok v SpiceSharp knižnici je náročnejšie než sa zdalo a budeme najskôr potrebovať bližšie pochopiť jadro SpiceSharpu
- Rozhodli sme sa nerobiť zmeny v pracovnom nasadení

4.5. RETROSPEKTÍVA K ŠPRINTU 5 (14.12.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách

- Šprint prebehol bez akýchkoľvek komplikácií, vzhľadom na to, že sme mali na to iba týždeň, a teda sme aj dostali menej úloh
- Všetky úlohy boli načas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- V tomto šprinte sme viac menej pokračovali v tom, s čím už máme skúsenosti, takže sa nevyskytli žiadne problémy
- Ďalší šprint začne až v letnom semestri tak dúfame, že pracovné nasadenie na začiatku letného semestra bude rovnaké ako počas celého zimného semestra

4.6. RETROSPEKTÍVA K ŠPRINTU 6 (12.3.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Aj po pauze sme sa nemali problém vrátiť do starého pracovného nasadenia a úspešne šprint dokončiť

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Nedošlo k žiadnej zmene pracovného nasadenia po prázdninách
- Počas tohto šprintu sme boli časovo dosť vyťažený, čo nakoniec vyústilo k nestihnutiu jednej User Story, čo ale pre nás nepredstavoval problém, pretože žiadna úloha nebola od tejto závislá, a teda sme ju len presunuli do nového šprintu

4.7. RETROSPEKTÍVA K ŠPRINTU 7 (26.3.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách

- Všetky úlohy boli včas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Počas tohto šprintu sme sa vrátili do pôvodného tempa, a teda všetky úlohy sa stihli včas dokončiť.

4.8. RETROSPEKTÍVA K ŠPRINTU 8 (9.4.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli včas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Pokračovali sme v klasickom tempe aj naďalej

4.9. RETROSPEKTÍVA K ŠPRINTU 9 (23.4.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Už sa blíži koniec semestra, a preto je potrebné, aby sme pokračovali v danom tempe aj naďalej a úspešne náš projekt dokončili

4.10. RETROSPEKTÍVA K ŠPRINTU 10 (7.5.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Čaká nás posledný týždňový šprint, a teda je potrebné dokončiť všetky potrebné časti

4.11. RETROSPEKTÍVA K ŠPRINTU 11 (14.5.)

POZITÍVA ŠPRINTU

- Hneď po zadelení úloh a vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Toto bol náš posledný šprint tímového projektu, už nás čaká len tvorba vstupov do semifinále TP Cupu

ZÁVER

Počas práce na prvých piatich šprintoch sme sa ako celý tím úspešne adaptovali na princípy agilného vývoja v metodike *Scrum*. Zo začiatku sme mali značné medzery pri tvorbe nového šprintu, konkrétne ohodnocovanie jednotlivých *User Stories* pomocou *Story Points* princípom *planning poker*. Tento problém sa nám ale podaril eliminovať už do 2. šprintu, pretože sme sa ako tím lepšie spoznali a „naladili“ sme sa na tú istú vlnu.

Náš počiatočný prístup k projektu sa pozitívne odrazil na prvej retrospektíve, kde sme sa jednohlasne zhodli na ponechaní súčasného pracovného nasadenia, vďaka čomu sme bez väčších problémov úspešne dokončili zvyšné šprinty zimného semestra.

Počas práca na šiestich šprintoch v letnom semestri sa nám podarilo udržiavať tempo, ktoré sme mali aj v zimnom semestri, a teda strata jedného člena nášho tímu nás nezasiahla. Počas letného semestra sme sa už naplno venovali tvorbe rôznych verzií nášho prototypu až dokedy sa nám nepodarilo vytvoriť finálny prototyp, ktorý spĺňa všetky projektové požiadavky a ešte niečo navyše.

PRÍLOHA A. MOTIVAČNÝ DOKUMENT

Autori prílohy: Patrik Tománek, Ľubomír Kurčák, Tomáš Sabo, Erik Paľa, Viktor Beňo

A.1. TÍM

Členovia: Patrik Tománek, Erik Paľa, Ľubomír Kurčák, Viktor Beňo, Tomáš Sabo

Doterajšie skúsenosti:

- Tvorba hier v Unity3D
  - Patrik, Ľubomír a Tomáš už majú skúsenosti s prácou v Unity3D.
- Práca v Unity3D ako univerzitný projekt
  - Patrik mal za úlohu vytvoriť interaktívnu webovú aktivitu, aby oboznámil jej používateľov s princípmi rôznych algoritmov a umelej inteligencie používanej pri profilovaní internetových používateľov, pričom sa rozhodol vypracovať tento projekt v Unity3D za použitia WebGL pluginu (https://randomxd.itch.io/how-tointernet).
  - Tomáš už pracoval na tímovom projekte v roku 2018/2019, jeho tím (Tím 20) vyvíjal aplikáciu na generovanie UML diagramov v 3D priestore v prostredí Unity3D.
- Tvorba responzívnej aplikácie použitím raytracing
- Práca s neurónovými sieťami
  - Viktor pracoval s neurónovými sieťami v rámci bakalárskej práce. Neurónová sieť bola vytvorená pomocou TensorFlow knižnice v jazyku Python.
- Technické skúsenosti relevantné pre tvorbu responzívnych aplikácií - nízka odozva vstupu (input lag), grafické programovanie - projekcia, shadery, optimalizácia draw-calls, networking pre hry - unreliable prenos, streaming, interpolácia.

A.2. MOTIVÁCIA

Téma: VR laboratórium pre dištančné vzdelávanie [VRLab]

Prečo: Rozhodli sme sa pre výber tejto témy, pretože nás zaujala z viacerých hľadísk:

- Virtuálna realita sa stáva čoraz populárnejšou, a teda aj dopyt pre projekty zamerané na VR. Oboznámenie sa s prácou s touto technológiou by predstavoval pozitívny prínos do budúcnosti všetkých členov tímu.
- Unikátna príležitosť získať hands-on skúsenosti pri tvorbe aplikácií a používaní VR hardvéru v univerzitnom prostredí a aplikovať získané poznatky v novej technologickej oblasti.
- Téma ponúka veľa možností zabudovania vlastných kreatívnych nápadov.
- Možnosť overiť široko škálovú uplatniteľnosť VR technológií vzhľadom na laického a rekreačného používateľa.

Čo vieme poskytnúť: Po tímovej diskusií sme našli niekoľko dôvodov, prečo práve my by sme mali byť zvolený pre túto tému:

- **Skúsenosti s platformou.** Keďže členovia tímu už pracovali s Unity3D, máme za to, že to značne urýchli prácu na projekte, a umožní členom, ktorí skúsenosti s Unity3D nemajú, aby boli produktívni od prvého dňa a odniesli si z projektu čo najviac.
- **Technické skúsenosti relevantné pre tvorbu responzívnych aplikácií.** Dôraz na nízku odozvu vstupu (input lag). Grafické programovanie, projekcia, shadery, používanie grafických driverov,

optimalizácia draw-calls. Networking pre hry - serializácia, unreliable prenos, streaming, interpolácia, server-client, peer-to-peer architektúry.

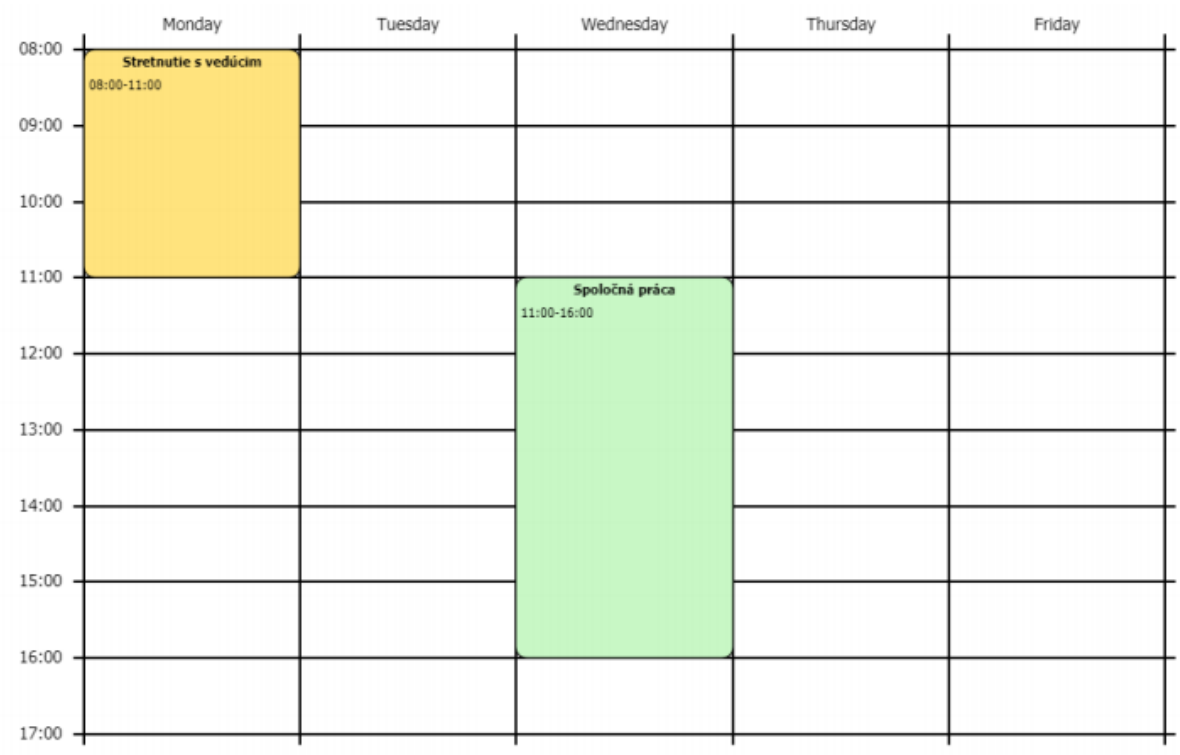
- **Filozofia dizajnu zameraného na používateľa.** Používateľ by mal vykonať čo najmenší počet krokov pred začatím používania produktu. Snaha neumožniť používateľovi "chytiť sa do pasce." Používateľské prostredie by malo byť prehľadné a interakcia by mala byť intuitívna a ohľaduplná k všetkým používateľom.
- **Entuziazmus a pocit zmysluplnosti.** Všetci sa tešíme z možnosti pracovať na tvorbe VR hry v Unity3D ako univerzitný projekt. Pri voľbe preferovanej témy sme sa ihneď zhodli a vidíme ju ako dobrú iniciatívu a príležitosť na osobný rozvoj.

A.3. Zoradenie tém podľa priorít

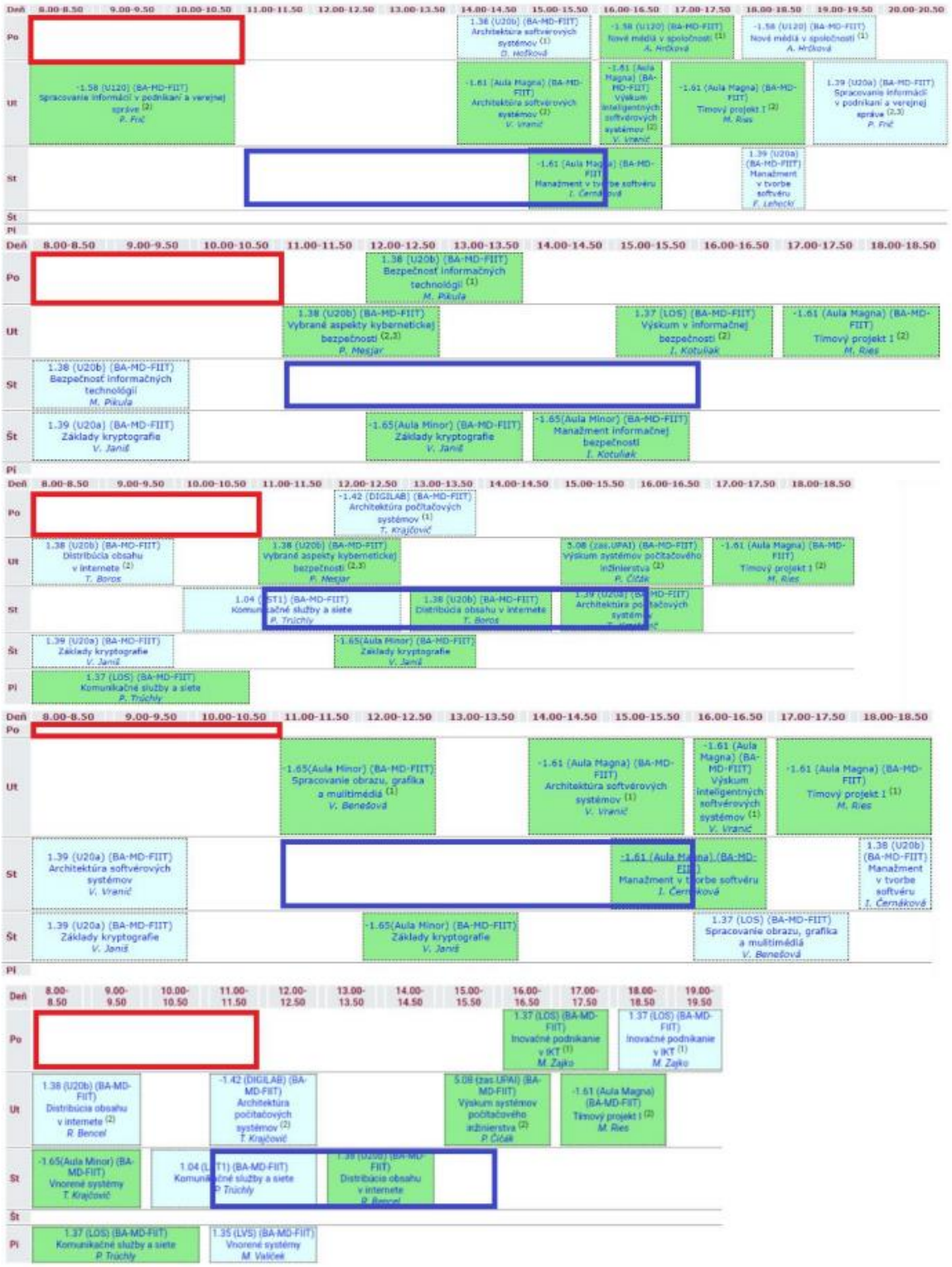
1. 17. VR laboratórium pre dištančné vzdelávanie [VRLab]
2. 3. Vizualizácia softvéru vo virtuálnej a rozšírenej realite [VizReal]
3. 10. Game-chain: Ako bezpečne vymieňať herné účty
4. 8. Automatické rozpoznávanie spektier [ARS]
5. 9. Vzdialené monitorovanie zdravotného stavu človeka pomocou E-Health
6. 5. Educational and coworking driven orchestration portal [EDUCO]
7. 6. Transformácia priestorov na bezpečné a inteligentné miesta na prácu [SmartSpace]
8. 4. Educational Content Engineering Hub - Databáza otázok, odpovedí, úloh a riešení [ECEH-DU]
9. 1. Blockchain platobné brány [BlockPay]
10. 19. Podporný informačný systém pre študijné oddelenie [CROSS-CHECK]
11. 15. Webový vyhľadávač podobnosti [AntiPlag]
12. 2. Webové IDE pre ASIC [ASICDE]

A.4. Rozvrhy tímu

- Martin Suchoň nám oznámil, že sa nebude zúčastňovať štúdia na fakulte, čím efektívne nebude súčasťou tímu
- Čas na stretnutie s vedúcim projektu by nám vyhovoval pondelky medzi 8:00 a 11:00
- Tímové stretnutia budú predbežne prebiehať v stredu medzi 11:00 a 16:00



Obrázok A.1 - Rozvrh pre predmet Tímový projekt



Obrázok A.2 - Rozvrhy členom tímu s vyznačenými časmi pre Tímový projekt

Červenou farbou je znázornený čas na stretnutie s vedúcim.

Modrou farbou je znázornený čas na spoločnú prácu, rozhodli sme sa pre vybratie väčšieho časového úseku, z dôvodu kolízií medzi jednotlivými členmi a ich cvičeniami/prednáškami, na základe toho sa budú aj rozdeľovať úlohy.

PRÍLOHA B. PRIHLÁŠKA NA TP CUP

Autor prílohy: Patrik Tománek

B.1. KONTAKT

tp16.radiant@gmail.com

B.2. TÍM

Náš tím sa skladá z 5 členov: Patrik Tománek, Ľubomír Kurčiak, Tomáš Sabo, Erik Paľa a Viktor Beňo. Pričom 3 členovia nášho tímu už majú rozsiahlejšie skúsenosti s Unity engine, a poslední 2 členovia sú vzdelaní v oblasti elektrických obvodov. Na základe toho sme si vytvorili projekt presne ušitý pre náš tím.

B.3. MOTIVÁCIA

Kvôli súčasnej pandémie koronavírusu trpia takmer všetky školy na Slovensku, obzvlášť základné alebo stredné školy. Pre tieto inštitúcie je v mnohých prípadoch takmer nemožné prejsť na online výučbu, čo ma obrovský dopad na vzdelanie mladej generácie. Medzi najproblematickejšie a zároveň veľmi dôležité oblasti výučby patria laboratórne cvičenia, ktoré väčšinou vyžadujú špeciálne vybavenie ako aj odborný dozor.

Našťastie ale v súčasnej dobe existujú viaceré riešenia ako prekonať mnohé problémy spojené s distančnou výučbou, konkrétne sa zameriavame na virtuálnu realitu. Vďaka virtuálnej realite sme schopní skonštruovať laboratórium, v ktorom v tom istom čase môže byť učiteľ aj jeho žiaci. Učiteľ žiakom vysvetlí preberanú látku a následne si žiaci môžu praktickými úlohami tieto vedomosti overiť a použiť v praxi, pričom nie je potrebné aby žiaci vlastnili akékoľvek vybavenie okrem VR headsetu, pretože všetky potrebné veci im budú vo virtuálnej realite k dispozícii.

B.4. NÁPLŇ PROJEKTU

V našom projekte sa budeme zameriavať na prácu s elektrickými obvodmi. Konkrétne učiteľ bude mať možnosť nahrať si jeho prezentáciu do VR prostredia, kde ju odprednáša žiakom. Témou laboratórneho cvičenia bude tvorba elektrických obvodov pre ovládanie modelu autíčka. Žiakom budú najskôr vysvetlené základy obvodov pre zapnutie svetla, spustenie smeroviek alebo trúbenie. Neskôr prejdú na zložitejšie koncepty ako tvorba motora, pridávanie alebo uberanie plynu, otáčanie kolies a brzdenie.

Žiaci si následne budú môcť zrealizovať vlastné elektrické obvody na základe nadobudnutých vedomostí, pričom každý žiak má k dispozícii vlastný model auta, na ktorom bude všetky skonštruované elektrické obvody následne testovať.

Keďže chceme aby náš projekt bol skutočne zaujímavý pre mladšiu generáciu. Žiaci si po dokončení tvorby ich elektrických obvodov a ich implementovanie do modelu auta, môžu zmerať svoje sily, resp. vedomosti voči ostatným žiakom. Konkrétne sa bude jednať o formu závodu, v ktorom sa ukáže, kto zvládol tvorbu elektrických obvodov najlepšie.

B.5. CIELE PROJEKTU

- Vytvorenie laboratórneho cvičenia pre elektrické obvody vo virtuálnej realite, ako aj desktope
- Možnosť nahranie prezentácie do VR prostredia z učiteľovho počítaču
- Vytvorenie prostredia pre tvorbu elektrických obvodov pomocou breadboardov
- Vytvorenie modelov a logiky pre všetky potrebné elektrické súčiastky
- Vytvorenie interaktívneho modelu autíčka:
    - Môže sa doňho nasadnúť
    - Môžu sa doňho implementovať všetky vytvorené el. obvody
    - Možnosť otestovať korektnosť el. obvodov zábavným a interaktívnym spôsobom na základe žiakovho inputu
    - Auto môže zapínať svetlá, smerovky, trúbiť, pridať a uberať plyn, meniť smer, použiť turbo
- Zábavné otestovanie výstupov všetkých žiakov spoločným závodom všetkých modelov áut na spoločnej závodnej trati

Projekt bude vytvorený pomocou Unity engine za použitia C# programovacieho jazyku. Hlavným cieľom bude implementácia pre prostredie virtuálnej reality, ale takisto aj obyčajný desktop. Veríme, že tento projekt uľahčí prechod škôl do distančnej výučby bez straty vzdelávacej hodnoty.

PRÍLOHA C. METODIKY

Autor prílohy: Tomáš Sabo

C.1. METODIKA DOKUMENTÁCIE

Dedikácia

Metodika opisuje zásady a postupy písania dokumentácie, ktoré zaužívame v tíme 16 Radiant. Je určená všetkým členom tímu.

Role

- **Člen tímu** - študent, ktorý je súčasťou tímu č. 16 Radiant na predmete Tímový projekt
- **Autor metodiky** - člen tímu, ktorý napísal danú metodiku
- **Odborník v oblasti** - člen tímu, ktorý ma najviac vedomostí v určitej oblasti
- **Scrum Master** - člen tímu, ktorý riadi tím a ktorý vytvára zápisnice počas stretnutia s product ownerom

Zápisnica

Zápisnica slúži na spísanie najdôležitejších udalostí zo stretnutia, tento dokument je taktiež veľmi dôležitý pre product ownera ako zdroj informácií. Zápisnicu spravidla spisuje Scrum Master alebo osoba, ktorá ho zastupuje. Avšak prispieť k obsahu zápisnice môže každý člen tímu.

Zápisnica musí obsahovať:

- **Zhrnutie zo stretnutia** - obsahuje hlavné body stretnutia, čo sa riešilo a robilo v očíslovaných odrážkach
- **Poznámky zo stretnutia** - štrukturovaný text, ktorý obsahuje všetky dôležité informácie zo stretnutia

Na *Obrázok C.1*, je možné vidieť príklad zápisnice.

ZÁPISNICA Č. 1

Tím č. 16 - Radiant

12.10.2020

ZHRNUTIE STRETNUTIA

1. Diskusia ohľadom výberu simulátora pre praktické laboratórium
2. Prvotné napĺňanie backlogu
3. Tvorba a spustenie prvého šprintu

POZNAMKY ZO STRETNUTIA

- Prvotné zoznamovanie sa so Scrum Poker, diskusia pri odhadovaní zložitosti jednotlivých User Tasks.
- Vypracovanie analýzy nájdených druhov simulácií so zameraním na elektrické, resp. logické obvody, primárne pre C# knižnice. Následne spísanie dokumentu s opísaním daného simulátora, uvedenie jeho výhod, resp. nevýhod.
- Vybranú simuláciu integrovať do Unity, vypracovanie jednoduchého príkladu s príslušným výstupom.
- Používať LTS verziu Unity (2019.4.12f)
- Vytvorenie webového sídla
    - Názov projektu a príslušný opis
    - Zoznam členov a ich roly
    - Počiatočné zameranie projektu
    - Vloženie zápisnice do dokumentového sídla

Obrázok C.1 – Zápisnica zo stretnutia

Retrospektíva

Retrospektíva slúži na zhodnotenie priebehu celého šprintu. retrospektíva sa vytvára pomocou dvoch otázok - Čo bolo dobré na šprinte a čo bolo zlé na šprinte, prípadne čo by sa mohlo zlepšiť. Retrospektívu vytvára člen tímu počas stretnutia alebo tesne po stretnutí na základe odpovedí jednotlivých členov tímu, následne sa spíše zhodnotenie celého šprintu.

Retrospektíva musí obsahovať:

- **Čo bolo dobré na šprinte** - spíšu sa pozitíva šprintu
- **Čo bolo zlé na šprinte** - spíšu sa negatíva šprintu, prípadne pripomienky
- **Zhodnotenie šprintu** - opíše sa celý výsledok šprintu, aké pravidlá chce tím dodržiavať alebo zmeniť v nasledujúcich šprintoch

Na *Obrázok C.2*, je možné vidieť príklad retrospektívy.

RETROSKEPTÍVA Č. 1

Tím č. 16 - Radiant

26.10.2020

POZITÍVA ŠPRINTU

- Hneď po zadelení úloha vytvorení šprintu product ownerom, sme si tieto úlohy medzi sebou rozdelili
- Pracovali sme priebežne na úlohách, čiže už v polovici šprintu každý člen tímu prezentoval jeho progres na jemu pridelenej úlohe
- Pravidelná komunikácia počas celého šprintu, nielen počas bloku určeného na spoločnú prácu alebo stretnutie s product ownerom
- Vzájomné pomáhanie si počas práce na úlohách
- Všetky úlohy boli načas dokončené

NEGATÍVA ŠPRINTU

Žiadne

ZHODNOTENIE

- Pracovné nasadenie na ďalšom šprinte ponecháme zatiaľ nezmenené po prvom šprinte

Obrázok C.2 – Retrospektíva šprintu

Dokument s exportom úloh

Na začiatku každého nového šprintu, po plánovaní úloh, člen tímu vyexportuje dokument, ktorý obsahuje aktuálny stav úloh z nástroja Azure DevOps. Tento export musí obsahovať všetky evidované úlohy, ktoré vzišli z našich stretnutí. Ku každej User Story sa uvádzajú dôležité informácie ako názov, popis, stav úlohy, zodpovedný riešiteľ a podobne.

Na *Obrázok C.3*, je možné vidieť príklad dokumentu s exportom úloh.

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15604	**Vytvorenie webového sídla**	**User Story**	Closed	Vytvorenie webového sídla * zapisnica * nazov projektu + popis * zoznam clenov timu + roly * na co sme sa nazaciatok projektu zamerali	13	Radiant\Apple		Bc. Ľubomír Kurčák
15609	Spojazdnenie servera cez virtualne stroje	Task	Closed			Radiant\Apple	4	Bc. Ľubomír Kurčák
15610	Nasadenie sablony	Task	Closed			Radiant\Apple	2	Tomas Sabo
15611	Naplnenie obsahu stranky	Task	Closed	Na stranke by mali byt aj obrazky		Radiant\Apple	4	Bc. Patrik Tománek
15603	**Analyza simulatorov pre logicke a elektricke obvody**	**User Story**	Closed	prioritne csharp libraries ked sa nenajdu tak aj ine jazyky Treba spisat aj dokument, vzdy linka, popis, sample, vyhody/nevyhody. Vsetko pojde do vysledneho dokumentu. Idealne nuget package * circuit blox	8	Radiant\Apple		Bc. Ľubomír Kurčák
15605	Analyza simulatorov pre logicke obvody	Task	Closed			Radiant\Apple	4	Bc. Viktor Beňo
15606	Analyza simulatorov pre elektricke obvody	Task	Closed			Radiant\Apple	4	Bc. Erik Paľa
15607	**Prva integracia v Unity**	**User Story**	Closed	Integrovanie vybranej kniznice na logicke/elektricke obvody do Unity Rozchodenie sample Napriklad jednoduchy obvod, a program vypise voltage v bode ("meram napatie medzi X Y) Verzia LTS Release 2019.4.12f1 Released: 7 October 2020 https://unity3d.com/unity/qa/lts-releases	5	Radiant\Apple		Bc. Ľubomír Kurčák
15608	Prva integracia v Unity	Task	Closed			Radiant\Apple	8	Bc. Ľubomír Kurčák

Obrázok C.3 – Export úloh

C.2. METODIKA AGILNÉHO RIADENIA

Dedikácia

Metodika opisuje princípy riadenia prostredníctvom Agile, táto metodika je zaužívaná v tíme č. 16 Radiant a je určená všetkým členom tímu.

Príprava

Scrum Master by mal ešte pred začiatkom stretnutia si pripraviť osnovu stretnutia, podľa ktorej stretnutie bude prebiehať. V prípade zmeny Scrum Mastera je potrebné sa dohodnúť s členmi tímu na pridelení pozície Scrum Mastera inému členovi ešte niekoľko dní pred začiatkom stretnutia.

Začiatok stretnutia

Na začiatku stretnutia si členovia pozrú v plánovacom kalendári ich úlohy na dnešné stretnutie. Ak niektoré úlohy nie sú pridelené, členovia sa rozhodnú na ktorej úlohe sa začne pracovať.

Tímový stand-up

Na začiatku každého stretnutia prebieha tímový stand-up, ktorý slúži na udržanie prehľadu o vývoji produktu a o zistení možných komplikácií.

Každý člen tímu musí odpovedať tri základné otázky:

- Čo som spravil na projekte od posledného stretnutia?
- Čo plánujem spraviť na projekte do budúceho stretnutia?
- Existuje nejaká prekážka, ktorá mi bráni splniť úlohy?

Priebeh stretnutia

Stretnutie prebieha cez konferenčný hovor, a členovia pracujú na svojich pridelených úlohách. Na konci stretnutia si každý člen poznačí svoj odpracovaný čas na úlohách do plánovacieho kalendára. Počas stretnutia je povinný jeden člen tímu zapisovať poznámky zo stretnutia a taktiež je tento člen zodpovedný za spísanie zápisnice ku danému dňu zo stretnutia. Po skončení stretnutia je potrebné ešte v ten deň spísať zhodnotenie stretnutia, toto zhodnotenie by malo byť odsúhlasené všetkými členmi tímu.

Product Owner

Stretnutie s product ownerom začína s pripomenutím úloh pre aktuálny šprint a skontrolovaním aktuálne rozpracovaných a dokončených úloh. Počas stretnutia s product ownerom sa robí backlog grooming.

User Story sú zoradené v Backlogu podľa priority, túto prioritu User Story určuje product owner počas stretnutia na základe jej komplexity a užitočnosti. Komplexitu User Story určujú členovia tímu, na určovanie komplexity User Story sa používa planning poker. Pri určovaní komplexity User Story prebieha komunikácia medzi členmi a planning poker sa opakuje kým sa členovia nedohodnú. Ak User Story nemá splnenú Definition of Ready, je možné ju rozdeliť na viaceré User Story. Každá User Story pred zaradením do šprintu musí byť rozdelená na Tasky. Do Taskov je potrebné definovať estimating hours, estimating hours by nemal byť väčší ako 8 hodín. Ak Task je časovo náročnejší ako 8 hodín je potrebné ho rozdeliť na menšie Tasky a rozložiť medzi viacerých členov tímu.

Právomoc Scrum Mastera

Scrum Master má právo zakázať témy diskusie alebo nejaký problém, ktorý aktuálne riešia členovia tímu. Scrum Master má právo vyhlásiť prestávku v rozsahu 10 až 30 minút, prestávky by mali byť odsúhlasené členmi tímu.

Po šprinte

Členovia tímu po šprinte spoločne oddiskutujú čo stihli, čo spôsobovalo problémy a čo by sa dalo zlepšiť.

Definitions

Definition of Done

Definition of Done je v tíme definovaný:

- Projekt bol odprezentovaný product ownerovi.
- Pre User Story sú splnené akceptačné kritériá product ownera.
- User Story bola schválená product ownerom.
- Kód obsahuje zmysluplné komentáre.
- Kód bol skontrolovaný metódou code review aspoň od jedného ďalšieho člena tímu.

Definition of Ready

Definition of Ready je v tíme definovaný:

- Pre User Story:
    - Musí byť zaradená do šprintu.
    - Musí mať zvolenú zodpovednú osobu.
    - Názov musí byť zrozumiteľný.
    - Komplexita User Story musí byť ohodnotená tímom a zapísaná ako Story points.
- Pre Task:
    - Musí byť zaradený do šprintu.
    - Musí mať zvoleného riešiteľa úlohy.
    - Musí obsahovať krátky zrozumiteľný popis.
    - Názov musí byť zrozumiteľný.
    - Náročnosť je odhadovaná počtom hodín v Original Estimate a Remaining.
    - Úloha musí odkazovať na User Story, pod ktorým sa nachádza.

C.3. METODIKA KOMUNIKÁCIE

Dedikácia

Metodika opisuje dohodnuté spôsoby komunikácie, ktoré zaužívame v tíme 16 Radiant.

Komunikačné nástroje

V našom tíme používame komunikačný nástroj Slack a Discord, každý člen tímu by mal byť pripojený na Slacku alebo Discorde vždy keď to je možné, ideálne mať nainštalovaný Slack alebo Discord aj v mobile, aby členovia tímu boli rýchlo dosiahnuteľný. V komunikačnom nástroji Slack náš tím používa niekoľko komunikačných kanálov pre potreby komunikácie v tíme. Počas stretnutia sme napojený na spoločnom konferenčnom hovore na komunikačnom nástroji Discord.

Aktuálne komunikačné kanály sú:

- **general** - všeobecný kanál na riešenie rôznych problémov ohľadom tímu
- **stretnutie** - tento komunikačný kanál sa používa na komunikáciu tímu počas stretnutia
- **tfs** - tu dostávame všetky notifikácie z TFS
- **status-report** - Každý večer medzi 20:00 a 24:00 do tohto komunikačného kanálu napíšu povinne všetci členovia status report čo za aktuálny deň spravili alebo alternatívne stačí svoj postup za deň oddiskutovať na Discorde s ostatnými členmi tímu. Prostredníctvom tohto reportu dávajú členovia tímu najavo ostatným členom na čom aktuálne pracujú a taktiež to slúži ako spätná väzba pre Scrum Mastera.
- **tp-cup** - komunikačný kanál, ktorý slúži výhradne pre TP Cup.

Spoločný email:

Okrem Slacku a Discordu používame spoločný e-mail, ku ktorému ma každý člen tímu prístup:

tp16.radiant@gmail.com

Vo výnimočných situáciach každý člen tímu má k dispozícii telefónny kontakt na každého člena tímu.

Komunikácia s product ownerom

S product ownerom komunikujeme cez nasledovné spôsoby:

- Počas stretnutia cez Google Meet.
- Slack
- E-mail - V prípade dôležitých e-mailov, ktoré nie sú priamo adresované product ownerovi, tak je zaužívané pridávať product ownera do CC.

C.4. METODIKA VERZOVANIA

Dedikácia

Metodika opisuje zásady a postupy verziovania, ktoré sú zaužívané v tíme 16 Radiant.

Zaužívané pojmy

Zdrojový kód a komentáre musia byť napísané v anglickom jazyku, ostatné jazyky sú zakázané.

- Commit - zmeny v repozitári
- Branch - samostatná vetva projektu, ktorá umožňuje paralelnú prácu v tíme
- Push - nahratie zmien z lokálneho repozitára do vzdialeného
- Pull - stiahnutie zmien do lokálneho repozitára zo vzdialeného
- Merge - spája dve alebo viacero úprav z vetiev do jednej

Repozitár

Náš tím má vytvorený vlastný repozitár na prácu na tímovom projekte, k tomuto repozitáru majú prístup všetci členovia tímu. Repozitár nášho tímu je súkromný a na jeho prístup je potrebné povolenie od správcu.

Branch

V našom tíme pri práci rozlišujeme dva hlavné druhy vetiev a to master a development.

Vetva master predstavuje hlavnú vetvu projektu, z tejto vetvy vznikajú ďalšie vetvy, ktoré obsahujú ďalšie funkčné časti projektu. Táto vetva musí byť vždy aktuálna a všetky jej pridané časti musia byť stabilné a funkčné, čiže do master vetvy sa môže pridávať iba funkčný, stabilný a otestovaný kód.

Vetva development predstavuje rozpracovanú vetvu projektu na ktorej sa aktuálne pracuje, táto vetva sa neustále mení, spravidla sa upravuje, spústa a testuje lokálne. V určitých momentoch sa vetva development pridáva do master vetvy, spravidla to je na konci šprintu. Pri takomto pridávaní do master vetvy, musí byť vetva plne funkčná, otestovaná a prejdená code review. Development vetva pri pridávaní do master vetvy nemôže obsahovať bugy a musí byť stabilná.

Členovia nášho tímu si môžu z development vetvy vytvárať ďalšie menšie vetvy, ktoré umožňujú paralelnú prácu na úlohách, tieto menšie vetvy predstavujú prácu na User Story. Poznámka: pred prácou je potrebné sa uistiť, že sa pracuje s aktuálnou verziou vetvy, v prípade, že nie je aktuálna, je potrebné ju aktualizovať.

Pomenovanie vetiev: náš tím sa riadi podľa konvencie gitflow, čiže názov vetvy dodržiava nasledovný formát: [vetva]/[názov]

Označenie vetiev môže byť:

- feature - časť produktu
- release - nasadenie otestovanej časti produktu
- hotfix - oprava chybnej časti kódu

Commit správy

Pri písaní správ pre commity je potrebné aby boli efektívne, čiže aby zahŕňali informačnú hodnotu a aby bolo z nej jasné čo daný commit robí. Je potrebné aby každý člen tímu pri písaní commit správ sa snažil aby správa bola stručná a aby vystihovala príslušné zmeny.

Preto je potrebné dodržiavať nasledovné pravidlá:

- Správa musí mať maximálne 50 znakov, v prípade jej prekročenia je potrebné commit rozdeliť.
- Správa začína veľkým písmenom.
- Pre ukončenie správ sa na konci nepíše bodka.
- Správa musí byť napísaná v slovenskom jazyku.

Príklad dobrej správy: "Oprava chyby pri vkladaní súčiastok do gridu"
Príklad zlej správy: "opravil som nejaké chyby"

Mergovanie vetiev

Pri spájaní vetiev je potrebné sa riadiť nasledujúcimi pravidlami:

- Člen tímu, ktorý ide spájať vetvy sa musí najprv uistiť, že sa nachádza vo vetve, ktorú ide spájať.
- Kód v danej vetve sa ešte raz skompiluje a uistí, že funguje správne
- V prípade, že sa nájdu nejaké problémy alebo chyby počas testovania, je potrebné túto chybu nahlásiť členovi tímu, ktorý na danej vetve pracoval a čo najrýchlejšie ju opraviť, bez opravy nemôžu sa vetvy spojiť.
- Po spojení je potrebné vyriešiť prípadne vzniknuté konflikty.

PRÍLOHA D. ZÁPISNICE

Autor prílohy: Patrik Tománek

D.1. ZÁPISNICA Č.1 (12.10.)

Zhrnutie Stretnutia

1. Diskusia ohľadom výberu simulátora pre praktické laboratórium
2. Prvotné napĺňanie backlogu
3. Tvorba a spustenie prvého šprintu

Poznámky zo stretnutia

- Prvotné zoznamovanie sa so Scrum Poker, diskusia pri odhadovaní zložitosti jednotlivých User Tasks.
- Vypracovanie analýzy nájdených druhov simulácií so zameraním na elektrické, resp. logické obvody, primárne preC# knižnice.Následne spísanie dokumentu s opísaním daného simulátora, uvedenie jeho výhod, resp. nevýhod.
- Vybranú simuláciu integrovať do Unity, vypracovanie jednoduchého príkladu s príslušným výstupom.
- Používať LTS verziu Unity (2019.4.12f).
- Vytvorenie webového sídla
    - Názov projektu a príslušný opis
    - Zoznam členov a ich roly
    - Počiatočné zameranie projektu
    - Vloženie zápisnice do dokumentového sídla

D.2. ZÁPISNICA Č.2 (19.10.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Analýza doposiaľ nájdených simulácií na elektrické, resp. logické obvody Viktorom a Erikom

Poznámky zo stretnutia

- Vytvoriť certifikát na HTTPS pomocou https://zerossl.com/
    - Pridanie certifikátu na našu stránku pomocou HTTP File upload metódy
- Analýza nájdených simulácií
    - https://github.com/SpiceSharp - otestovať (veľká priorita)
    - https://github.com/diegocastagne/VR – otestovať(stredná priorita)
    - https://m.scirp.org/papers/87762 – otestovať (malá priorita)
    - https://github.com/npvinhphat/SnapSim - netestovať
    - https://www.codeproject.com/Articles/24577/Circuit-Engine - netestovať
    - http://liblcs.sourceforge.net/ - netestovať
    - https://github.com/Devoney/LogicCircuit - netestovať
    - https://github.com/PhoenixGameStudios/DangrLib - netestovať
- Špecifikácie písania dokumentu analyzovaných simulácií

    - Vypísať výhody a nevýhody (rozsah a zrozumiteľnosť poskytnutej dokumentácie, poskytnuté príklady použitia, aktuálnosť simulácie)
    - Zameriavať sa na čo najaktuálnejšie riešenia, ideálne nech je ich repozitár udržiavaný
    - Najväčší potenciál: SpiceSharp
- Implementovať viacero simulácií do Unity (integrovanie základného príkladu použitia každej simulácie), minimálne implementovať SpiceSharp

D.3. ZÁPISNICA Č.3 (26.10.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte
5. Napísať prihlášku pre TP Cup

Poznámky zo stretnutia

- Odstrániť NuGet package zUnity a pridať do projektu SpiceSharppomocou dll súborov
- Upraviť hlavný text na stránke
- Pripraviť nástroj na komunikáciu v tíme sintegráciou TFS
- Nájsť 3D modely pre elektrické/logické obvody
    - **Priorita!** (dokončiť do prvej polovice šprintu)
    - Dávať pozor na počet polygónov (zdokumentovať ich počet), sústrediť sa na low-poly modely
    - Prehľadávať 3D modely na
        - CGTrader
        - Unity Asset Store
        - TurboSquid
    - Rozdeliť modely do kategórií (grid, káblové, ...)
    - Hľadať breadboardové aj blokové modely, môžu sa kombinovať
    - Otestovanie textúr v Unity, snaha zísť možné problémy (polycount, priesvitnosť textúr, zlé tiene, ...)
- Analyzovať existujúce riešenia z praxe
    - Porovnať výhody/nevýhodyčo sa týka implementácie do VR prostredia
    - Circuit Blox, Physics Blocks, Tactile IQube, a ďalšie
- Vytvoriť návrh potencionálnych riešení
    - Spísať viaceré možnosti, zamyslieť sa nad nimi, spísať ich výhody/nevýhody
        - Grid/Breadboard
        - STEM/blocks
    - Závislé od analýzy amodelov, ktoré nájdeme
- Rozhodnúť presný cieľ práce
    - Široký prototyp (napr. VR nadstavba SpiceSharpu)
    - Špecifický prípad (Breadboard s modelmi áut)

D.4. ZÁPISNICA Č.4 (2.11.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Diskusia ohľadomimplementácie breadboardu alebo grid/blokov.

Poznámky zo stretnutia

- Otestovať zadarmo dostupné modely elektrických súčiastok, breadboardu a grid/blokov v Unity
    - Analýza použiteľnosti textúr v projekte
- Nenašli sa žiadne postačujúce modely grid/blokov, budeme si ich musieť vytvoriť
- Momentálne je preferencia na grid/bloky namiesto integrácie breadboardu
- Pri analýze dbať aj na náročnosť riešenia, preferencia je vytvoriť projekt, ktorý je možné použiť vo výučbe na základných školách
- Vytvoriť jednoduché nákresy potencionálnych riešení

D.5. ZÁPISNICA Č.5 (9.11.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Napísať pre obidva návrhy riešení (grid, breadboard) ich spôsob realizácie pomocou SpiceSharp knižnice
    - Poriadne premyslieť celý proces logiky
    - Naviazať obrázok návrhu spolu s príslušnou ukážkou kódu (stačí uviesť príklad jednoduchej žiarovky)
    - Zdokumentovať celý proces od tvorby elektrického obvodu v grid/breadboard zapojení až po vygenerovanie kódu pre príslušný obvod
- Breadboard je záložné riešenie, prioritou sa stáva grid
    - Potrebné premyslieť, či budú uzly dané kockou alebo mriežkou
- Implementácia konštrukcie obvodu v gride
    - Controller agnostic
    - Vytvorenie mriežky, ktorá detekuje kolízie s vkladanými blokmi
    - Zarovnať rotáciu súčiastok iba na 1 osi pri vkladaní do mriežky
    - Vytvoriť Grabber
        - Pozícia určená myšou (použiť raycast)
        - Zvýrazniť, ktorý objekt bude zodvihnutý
        - Zvýrazniť, na ktorú pozíciu bude objekt vložený do mriežky
- Zozbierať všetky možné použiteľné súčiastky, ktoré sú podporované SpiceSharpom, vybrať všetky tie, ktoré budeme potrebovať v našom projekte
- Vytvoriť modely kociek
    - Vytvoriť všetky spojovacie kocky (priama, štvoritá, križovatka, atď.)
    - Vytvoriť všetky tie, ktoré sme si vybrali na základe analýzy SpiceSharp knižnice

D.6. Zápisnica č.6 (16.11.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Diskusia ohľadom vytvorenia SpiceSharp výstupu na základe gridu

Poznámky zo stretnutia

- Pridať modely pre voltmeter a ampérmeter
- Uvedenie parametrov pre súčiastky v gride
    - Vypísať parametre pomocou UI vo forme kontextového menu
- Ako VR avatary budeme používať https://readyplayer.me/
- Ako VR framework budeme používať https://assetstore.unity.com/packages/templates/systems/vrinteraction-framework-161066
- Možnosť jednoduchej tvorby 3D modelov pomocou https://www.tinkercad.com/, ale môžeme zostať aj pri práci v Blender
- Vytvoriť aj návrh transformácie súčiastok v gride do kódu, ktorý je kompatibilný s SpiceSharp knižnicou

D.7. Zápisnica č.7 (23.11.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- V 3. šprinte sme mali za úlohu vytvoriť modely kociek pre naše elektrické súčiastky. V tomto šprinte sme dostali za úlohu tieto modely rozšíriť o ďalšie typy
    - Vytvorenie mosfet diódy pre porovnanie do breadboardu
    - Zmeniť textúry na súčasne vytvorených modelov, za účelom lepšej viditeľnosti vo VR prostredí
- Vytvoriť demo so SpiceSharp
    - Vytvoriť skript, ktorý mení parametre elektrického obvodu v runtime
    - Vyhodnotenie časových analýz
        - Meranie času statickej analýzy – pri každej zmene elektrického obvodu sa tento obvod vypne a znovu zapne
        - Meranie času transient analýzy – pri každej zmene elektrického obvodu sa automaticky prepočíta obvod bez jeho vypnutia
    - Vytvoriť vlastnú súčiastku
        - V projekte budeme takisto potrebovať súčiastky, ktoré nie sú podporované SpiceSharp knižnicou
        - Vytvoriť vodič ako rozšírenie rezistoru
- Vytvoriť scénu vo VR Interaction Framework

- https://assetstore.unity.com/packages/templates/systems/vr-interaction-framework-161066
- Implementovať do scény objekty z tohto frameworku
- Vytvoriť interakciu s objektami
  - Grabber
  - Snapping
- Porovnať snapping z VR Interaction Frameworku s tým, ktorý sme si sami vytvorili
  - Otestovať ich použiteľnosť
  - Vybrať ten, ktorý je pre náš projekt vhodnejší
- Implementovať VR emulátori z VR Interaction Framework
  - Zabezpečiť funkcionalitu aj za použitia klávesnice a myší

D.8. Zápisnica č.8 (30.11.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Diskusia ohľadom tvorby SpiceSharp dema
3. Diskusia ohľadom VR Interaction Framework

Poznámky zo stretnutia

- Jedna z úloh v 4. šprinte je vytvorenie vylepšených reliéfových modelov, ktorá bola do prvej polovice hotová a ukázaná nášmu Product Ownerovi, s ktorým sme sa dohodli o ukončení tohto smeru vývoja, pričom budeme pokračovať v smere tvorby 3D modelov
- Pri tvorbe SpiceSharp dema sme doposiaľ vytvorené demo testovali iba v rámci prostredia Visual Studio, kde nás Product Owner nasmeroval na riziko vzniku rôznych problémov pri implementácií tohto dema do Unity
  - Lepšie je hneď testovať SpiceSharp funkcionalitu v rámci Unity, aby sa čo najskôr odhalili možné chyby kým nebude neskoro
  - V prípade zistenia fatálnych chýb je dôležité hľadať novú knižnicu pre simuláciu elektrických obvodov
- Pre VR Interaction Framework sme sa dohodli o vytvorení vlastnej scény v Unity, v ktorej sa budú nachádzať rôzne objekty z tohto frameworku
  - Otestovať funkcionalitu emulátoru obsiahnutého vo frameworku
  - Naviazať na tento emulátor náš vlastný skript pre desktop riešenie, ktorý obsahuje raycast funkcionalitu

D.9. Zápisnica č.9 (7.12.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácií 3. šprintu, konkrétne úlohy, ktorá sa týkala vytvorenia SpiceSharp dema v Unity, sme dostali pár usmernení, na ktorých budeme pracovať ďalej
- Zistilo sa, že tvorba nových komponentov je náročnejšia než sme zo začiatku očakávali
    - Je potrebné bližšie pochopiť jadro SpiceSharp, konkrétne funkcionalita matíc a matematického modelu
- Máme pokračovať s integráciou VR Interaction Framework do nášho projektu.
    - Vytvoriť rozšírenie pre snap zones od VR Interaction Framewrok potrebné pre náš projekt
    - Integrovať naše vytvorené modely, aby boli kompatibilné s VR Interaction Framework
    - Prvotná integrácia inputov a outputov uzlov grid boardu
        - Rozmyslieť si akú logiku budeme používať
        - Začať implementovať vybranú logiku do nášho prototypu
- Ďalej máme analyzovať synchronizačné frameworky pre náš projekt, ktorí bude kolaboratívny. Okrem synchronizácie pohybu je takisto potrebná aj implementácia voice chatu.
    - Photon Bolt
    - SmartFox Server
    - Realtime Cloud
    - PUN2
    - Vyhodnotenie analýz

D.10. ZÁPISNICA Č.10 (14.12.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu

Poznámky zo stretnutia

- Všetky úlohy boli riadne dokončené
- Jednalo sa o posledný šprint zimného semestra, takže sme nový šprint nezačínali.

D.11. ZÁPISNICA Č.11 (26.2.)

Zhrnutie Stretnutia

1. Rekapitulácia práce počas dokončení skúškového obdobia zimného semestra
2. Tvorba nového šprintu
3. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie našej práce počas skúškového obdobia zimného semestra, ako aj po jeho dokončení ešte pred začatím letného semestra sme diskutovali ohľadom súčasného stavu v akom sa projekt nachádza, aby sme mohli úspešne vytvoriť prvý šprint v tomto semestri
- V tomto šprinte máme už vytvoriť prvú úspešnú integráciu nášho Unity prototypu do SpiceSharp, a teda aby si používatelia mohli tvoriť vlastné obvody na základe poskytnutých súčiastok a mohli tento obvod aj otestovať, resp. simulovať

    - Najskôr musíme vykonať analýzu vstupov pre SpiceSharp knižnicu, aby sme na základe toho mohli do nášho prototypu implementovať transformáciu elektrického obvodu z hry na vstup pre SpiceSharp knižnicu
    - Na základe analýzy môžeme prejsť na implementáciu a tvorbu nového prototypu nášho projektu
- Máme zdokumentovať doposiaľ existujúce časti prototypu za účelom vytvorenia prvotnej verzie používateľskej príručky
    - Vytvoríme zoznam podporovaných súčiastok, v ktorom tieto súčiastky budú aj zdokumentované, a teda budú sa tam nachádzať informácie ohľadom jej funkcionality
    - Vytvoríme zoznam 3D modelov súčiastok, spolu s dokumentáciou ich tvorby ako aj vytvorenie používateľského katalógu pre súčiastky
    - Zdokumentujeme implementačnú logiku konektivity súčiastok v rámci logickej reprezentácie elektrického obvodu v našom prototype
- Máme vytvoriť návrh simulovaného vozidla
    - Vytvoríme návrh hlavnej konštrukcie vozidla s podporou pre modulárne súčiastky
    - Vytvoríme návrh jednotlivých súčiastok vozidla ako aj ich kategorizáciu
        - Súčiastky, ktoré sú len mechanické, resp. fyzikálne
        - Súčiastky, ktoré súvisia s tvorbou a simuláciou elektrických obvodov

D.12. ZÁPISNICA Č.12 (6.3.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Diskusia ohľadom tvorby vstupu pre SpiceSharp

Poznámky zo stretnutia

- Počas rekapitulácie jednotlivých úloh sme nenarazili na žiadne problémy okrem našej User Story Integrácia Unity Prototypu do SpiceSharp. Pri tejto User Story sme počas práce zistili, že jej riešenie je komplexnejšie než sme si mysleli, a preto sme sa rozhodli počkať do stretnutia, kde sme mohli ďalšiu prácu prediskutovať
    - Prvotný problém bol vo vytvorení vstupu, a teda nejakého reťazca z Unity prototypu do SpiceSharp simulátoru. Pri riešení tejto úlohy sme našli hneď niekoľko možných riešení a chceli sme ich najprv prediskutovať, kým sa do nejakého pustíme. Nakoniec sme vybrali 2 možné riešenia:
        - Primárnym riešením bude použitie SpiceSharpParser balíčka, ktorý na základe definovaných šablón je schopný vytvoriť vstup kompatibilný so SpiceSharp simulátorom. Tieto šablóny budú musieť byť avšak na začiatku vyplnené.
        - Ďalším možným riešením môže byť odoslanie poľa polí do SpiceSharp threadu, v ktorom sa toto pole spracuje a vytvorí si podľa neho svoj požadovaný vstup do simulácie.

D.13. ZÁPISNICA Č.13 (12.3.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme diskutovali ohľadom 2 vecí
    - Jedna User Story sa nestihla včas dokončiť, konkrétne vypracovanie návrhu simulovaného vozidla. Keďže žiadna úloha nebola od tejto závislá nič zlé sa nestalo, len sme ju presunuli do ďalšieho šprintu.
    - Nepodarilo sa nám integrovať SpiceSharpParser do nášho projektu, pričom sme tento problém na tomto stretnutí aj vyriešili.
- Pri retrospektíve sa nenašli žiadne problémy, dôvod nedokončenia úlohy bola časová vyťaženosť v danom týždni, a teda na vypracovanie úlohy už nezvýšil čas
- V tomto šprinte máme teda pokračovať v úlohe z predošlého týždňa, konkrétne vypracovanie návrhu simulovaného vozidla
- Máme sa úlohu implementovať SpiceSharpParser do projektu, pretože sa zistila a odstránila príčina, kvôli ktorej sme neboli schopní implementovať tento balíček do projektu
    - Najprv máme integrovať SpiceSharpParser do projektu
    - Vytvoriť demo scénu pre otestovanie funkčnosti tohto balíčka
    - Kontaktovať autora tohto balíčka ohľadom aktualizácie, pretože pre korektnú funkčnosť tohto balíčka je potrebná zmena verzie SpiceSharp knižnice na staršiu. Dôvod je taký, že SpiceSharpParser nie je kompatibilný s novou verziou SpiceSharp knižnice.
- Máme vytvoriť model autíčka. V prípade, že sa nám nepodarí nájsť vhodné textúry takéhoto autíčka, bude potrebné ich vytvorenie
    - Najprv máme vytvoriť zoznam existujúcich modelov
    - Následne vytvoríme .fbx modely, ktoré importujeme do projektu a odhalíme prípadne problémy s danými modelmi
- Máme analyzovať fyzikálne simulácie vozidla
    - Najprv máme nájsť existujúce riešenia rôznych fyzikálnych simulácií, ako napríklad aj arkádové simulácie
    - Následne tieto nájdené riešenia zanalyzujeme a vyhodnotíme ich výhody/nevýhody

D.14. ZÁPISNICA Č.14 (19.3.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte
2. Diskusia ohľadom SpiceSharpParser

Poznámky zo stretnutia

- Počas diskusie sme nenarazili na žiadne problémy pri prezentovaní doterajšieho progresu, a teda budeme pokračovať v práci
- Po kontaktovaní jedného ztvorcov SpiceSharpParser projektu sme nedostali žiadnu odpoveď, a preto sme sa rozhodli kontaktovať aj druhého tvorcu, aby sme sa mohli čím skôr dozvedieť, či sa na danom projekte ešte bude pracovať.

D.15. ZÁPISNICA Č.15 (26.3.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu

3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme nenarazili na žiaden problém s nejakou z úloh
- Pri retrospektíve sa nenašli žiadne problémy, všetky úlohy boli dokončené včas
- V tomto šprinte sa máme venovať návrhu scény pre modifikáciu vozidla
    - Najprv si máme definovať, akým spôsobom bude auto modifikované, niektoré z doposiaľ vymyslených možností sú:
        - Zmenšenina vozidla počas jeho modifikácie
        - Hráč bude môcť vkladať vytvorené obvody do snap zón vozidla, kedy sa tento obvod premení na jednu zo súčiastok, ako napríklad motor, svetlá, atď.
        - Hráč bude môcť vyberať komponenty z vozidla, položiť ich na stôl, kedy sa tento komponent premení na grid-board a následne vypracuje požadovaný elektrický obvod, ktorý opätovne vloží do vozidla už vo forme komponentu
        - A iné
    - Ďalej máme po návrhu vytvoriť aj grafický náčrt scény so všetkými hlavnými časťami, ako napríklad stôl, vytiahnutie súčiastky, spôsob priebehu interakcie so súčiastkami a gridboardom, atď.
    - Nakoniec máme pre tieto súčiastky vytvoriť aj prvotné verzie schém elektrických obvodov, ktorých tvorba sa bude očakávať od používateľa
    - Celý proces máme zdokumentovať
- Následne máme vypísať spice kód pre súčiastky v simulovanom obvode
    - Najprv je potrebné vytvoriť interaktívnu tabuľu v prostredí hry
    - Následne na tejto tabuli si budú môcť používatelia v prípade záujmu nechať vygenerovať spice kód pre nimi simulovaný obvod
- Ako poslednú úlohu máme vytvoriť príklady pokročilejších obvodov za použitia SpiceSharp a SpiceSharpParsera
    - Oscilátor
    - Buzzer
    - Zložitejšie riadenie motora vozidla ako len cez odpor, a teda na odpor bude používateľom poskytnutý pomalý motor, pričom si budú môcť vytvoriť rýchlejší motor za použitia mosfetu
    - A ďalšie z pohľadu vozidla

D.16. ZÁPISNICA Č.16 (6.4.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte

Poznámky zo stretnutia

- Dnešné stretnutie sme nemali v piatok ako obvykle (02.04.2021) kvôli sviatku, ale mali sme ho až dnes
- Odprezentovali sme doterajší progres a následne sme o ňom diskutovali
- Počas diskusie sa nenašli žiadne problémy s úlohami

D.17. ZÁPISNICA Č.17 (9.4.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme nenarazili na žiaden problém s nejakou z úloh
- Pri retrospektíve sa nenašli žiadne problémy, všetky úlohy boli dokončené včas
- V tomto šprinte sa máme venovať vytvoreniu scény pre modifikáciu vozidla
    - Určiť spôsob selektovania objektov vozidla
    - Určiť spôsob načítavania obvodu z príslušného objektu vozidla
    - Vytvoriť interaktívne komponenty vozidla
    - Implementovať vkladanie objektov vozidla do gridboardu a následne spustenie gridboardu pre daný objekt
- Ďalej máme modifikovať model vozidla
    - Vykonať analýzu a spracovať animácie a animačných parametrov z poskytnutého modelu vozidla
    - Integrácia modelu s fyzikálnou simuláciou
    - Rozdeliť model na interaktívne a neinteraktívne súčiastky
- Predposledná User Story je zameraná na testovanie oscilátorov a kondenzátorov SpiceSharp-u
    - Vytvoriť sample kód oscilátorov
    - Vytvoriť sample kód kondenzátorov
    - Vyskúšať vytvorené kódy aj v LTspice
- Doplnenie modelov súčiastiek
    - Okrem vytvorenia samotných modelov je potrebné pre každý model zvlášť vytvoriť aj sample v Unity, pričom je potrebné tento vytvorený obvod odfotiť spolu so vstupom pre SpiceSharpParser a výpisom výstupudanej simulácie
    - Vytvoriť model batérie, cievky, kondenzátora, motora, reproduktora, oscilátora

D.18. ZÁPISNICA Č.18 (16.4.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte

Poznámky zo stretnutia

- Počas diskusie sa neodhalili žiadne problémy s úlohami
- Dnešné stretnutie bolo z väčšej časti zamerané na editovanie prvotnej verzií nášho článku pre IIT SRC 2021, ako aj tvorby nášho plagátu

D.19. ZÁPISNICA Č.19 (23.4.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu

2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme nenarazili na žiadne zásadné problémy s nejakou z úloh
- Pri retrospektíve sa takisto nenašli žiadne problémy
- V tomto šprinte máme integrovať model vozidla do scény pre modifikáciu
    - Všetky časti vozidlá, ktoré súvisia s elektrickými obvodmi bude potrebné rozdeliť na samostatne časti, konkrétne motor a svetlá
    - Takisto pri vyťahovaní súčiastok z vozidla a vkladaní ich do príslušného miesta na gridboarde je dôležité zabezpečiť spôsob, ktorý bude vracať stratené súčiastky naspäť do vozdila
- Ďalej máme vytvoriť symbolické komponenty vozidla, ktoré budú používané v gridboarde počas tvorby obvodu pre daný komponent
    - Konkrétne sa jedná o vytvorenie symbolického komponentu pre motor, žiarovku a klaksón
- Predposledná úloha je venovaná tvorbe vzorových obvodov s použitím Oscilátora
    - Vytvorenie niekoľkých rôznych obvodov
    - Otestovanie týchto obvodov
    - Vytvorenie elektrických schém pre tieto obvody
- Na koniec máme vytvoriť osciloskop, a teda vytvorenie a následne zobrazenie osciloskopických dát zo simulácie

D.20. ZÁPISNICA Č.20 (30.4.)

Zhrnutie Stretnutia

1. Diskusia ohľadom jednotlivých úloh v šprinte

Poznámky zo stretnutia

- Počas diskusie sme narazili na problém pri tvorbe osciloskopu, problém bol v tom,že balíček, ktorý sme na tvorbu grafov používali mal priveľkú záťaž na výkon. Rozhodli sme sa preto implementovať iné riešenie, konkrétne otestovať Line Renderer, prípadne nájsť iné spôsoby vykresľovania grafov s minimálnym efektom na výkon.

D.21. ZÁPISNICA Č.21 (7.5.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Tvorba nového šprintu
4. Rozdelenie úloh v šprinte

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme nenarazili na žiadne zásadné problémy s nejakou z úloh

- Pri retrospektíve sa nenašli žiadne problémy
- Na stretnutí sme sa zamerali na tvorbu posledného týždňového šprintu, pričom sme si za úlohu mali vymyslieť úlohy sami, aby sme dokončili všetko čo potrebujeme
- Ako prvú úlohu sme si zvolili vytvorenie používateľskej príručky
    - Príručka základného ovládania
    - Príručka tvorby obvodov
        - Kompletný návod ako poskladať ukážkový obvod, menenie napätia, osciloskop, vkladanie súčiastok do gridu, získanie súčiastok zo snap zones, pridanie nového modelu, vybratie súčiastky z auta a jej vloženie do príslušného miesta pri gridboarde
    - Príručka ovládania vozidla
- Druhú úlohu sme si určili ako finalizáciu úloh zapájania obvodov
    - Implementácia schém pre motor, svetlá a klaksón do hry
    - Testovanie korektnej funkcionality zapojených obvodov
- Ako poslednú úlohu sme si určili optimalizáciu používateľského zážitku
    - Úprava snap zones
    - Zjednodušenie manipulácie súčiastok vo vozidle
    - Vypnutie nepotrebných kolízií

D.22. ZÁPISNICA Č.22 (14.5.)

Zhrnutie Stretnutia

1. Rekapitulácia šprintu
2. Retrospektíva šprintu
3. Ukončenie TP

Poznámky zo stretnutia

- Počas rekapitulácie predošlého šprintu sme nenarazili na žiadne zásadné problémy s nejakou z úloh
- Pri retrospektíve sa nenašli žiadne problémy
- Na stretnutí sme diskutovali ohľadom posledného jednotýždňového šprintu ako aj celkových výsledkov, ktoré sa nám podarilo dosiahnuť
- Nový šprint sme už nevytvárali, ale čaká nás ešte TP Cup, takže sa zameriavame na tvorbu požadovaných vstupov do semifinále

PRÍLOHA E. EXPORT EVIDENCIE ÚLOH

Autor prílohy: Ľubomír Kurčák, Patrik Tománek

E.1. EXPORT ÚLOH K ŠPRINTU 1

D	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15604	**Vytvorenie weboveho sidla**	**User Story**	Closed	Vytvorenie weboveho sidla * zapisnica * nazov projektu + popis * zoznam clenov timu + roly * na co sme sa na zaciatok projektu zamerali	13	Radiant\Apple		Bc. Ľubomír Kurčák
15609	Spojazdnenie serveru cez virtualne stroje	Task	Closed			Radiant\Apple	4	Bc. Ľubomír Kurčák
15610	Nasadenie sablony	Task	Closed			Radiant\Apple	2	Tomas Sabo
15611	Naplnenie obsahu stranky	Task	Closed	Na stranke by mali byt aj obrazky		Radiant\Apple	4	Bc. Patrik Tománek
15603	**Analyza simulatorov pre logicke a elektricke obvody**	**User Story**	Closed	prioritne csharp libraries ked sa nenajdu tak aj ine jazyky Treba spisat aj dokument, vzdy linka, popis, sample, vyhody/nevyhody. Vsetko pojde do vysledneho dokumentu. Idealne nuget package * circuit blox	8	Radiant\Apple		Bc. Ľubomír Kurčák
15605	Analyza simulatorov pre logicke obvody	Task	Closed			Radiant\Apple	4	Bc. Viktor Beňo

15606	Analyza simulatorov pre elektricke obvody	Task	Closed			Radiant\Apple	4	Bc. Erik Paľa
15607	**Prva integracia v Unity**	**User Story**	Closed	Integrovanie vybranej kniznice na logicke/elektricke obvody do Unity Rozchodenie sample Napriklad jednoduchy obvod, a program vypise voltage v bode ("meram napatie medzi X Y) Verzia LTS Release 2019.4.12f1 Released: 7 October 2020 https://unity3d.com/unity/qa/lts-releases	5	Radiant\Apple		Bc. Ľubomír Kurčák
15608	Prva integracia v Unity	Task	Closed			Radiant\Apple	8	Bc. Ľubomír Kurčák

E.2. Export úloh k šprintu 2

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15676	**Napisat prihlasku do TPCup**	**User Story**	Closed	prvotny navrh, nie je to zavazne kontakt na tim	2	Radiant\Banana		Bc. Ľubomír Kurčák
15677	Spisat zakladny koncept	Task	Closed			Radiant\Banana	1	Bc. Patrik Tománek
15678	Finalizacia a odoslanie prihlasky	Task	Closed			Radiant\Banana	1	Bc. Patrik Tománek
15674	**Nastroj na komunikaciu v time**	**User Story**	Closed	Slack/Discord integracia na TFS pridat zakaznika	3	Radiant\Banana		Bc. Ľubomír Kurčák
15679	Vybrat nastroj na timovu komunikaciu	Task	Closed			Radiant\Banana	1	Bc. Viktor Beňo
15680	Vytvorit integraciu s TFS	Task	Closed			Radiant\Banana	2	Bc. Ľubomír Kurčák
15698	Otestovanie prepojenia na TFS	Task	Closed			Radiant\Banana		Bc. Viktor Beňo
15612	**Hladanie 3D modelov pre elektricke/logicke obvody**	**User Story**	Closed	treba napisat polycount (prisny limit polygonov (limit 100 000 tris/vertices)) * dobre vyhladat low-poly treba vybrat co najskor aby sa stihli kupit/zistit ci su vhodne * CGTrader * Unity Asset Store * Turbo Squid * https://www.cgtrader.com/3d-models/science/medical/lab-props * https://www.cgtrader.com/3dmodels/electronics/computer/electronic-components-6aa15bc8-b166-4f6d-b9af-c5768d645c01 * breadboard example: https://www.cgtrader.com/3d-models/science/laboratory/electronical-project-board * kable v breadboard unity example: https://www.youtube.com/watch?v=YYrLBzmFEjE rozdelit do kategorii,	8	Radiant\Banana		Bc. Ľubomír Kurčák

				naparovat na existujuce Papers: * https://cecas.clemson.edu/hcsl/wp-content/uploads/2016/04/Interactive-Breadboard-Activity-Simulation.pdf				
15681	Hladanie modelov pre Breadboard	Task	Closed			Radiant\Banana	4	Bc. Patrik Tománek
15682	Hladanie modelov pre grid/bloky	Task	Closed			Radiant\Banana	4	Bc. Ľubomír Kurčák
15686	Otestovat modely v Unity	Task	Closed	poly count textury/priesvitnost tiene najst hociake chyby		Radiant\Banana	3	Bc. Ľubomír Kurčák
15672	**Analyza existujucich rieseni z praxe (fyzickych)**	**User Story**	Closed	Treba porovnat vyhody/nevyhody Napr. Grid lahsi na implementaciu * Circuit Blox * Physics Blocks * Tactile IQube https://www.tactiles.io/ * https://www.amazon.com/Teenii-Electricity-Experiment-Electromagnetism-Electronics/dp/B075KPZM5N * https://www.amazon.co.uk/Snap-Circuits-SC-110-Electronics-Exploration/dp/B07VPDWXST/ref=pd_sbs_21_7?_encoding=UTF8&pd_rd_i=B07VPDWXST&pd_rd_r=0678aa0c-03ea-4474-b18a-ae049869928b&pd_rd_w=HB9RC&pd_rd_wg=YxFvn&pf_rd_p=b9bf232d-9a8a-4c7d-aa9d-641c0995d3a2&pf_rd_r=N6EREKB257TVMAZF86MB&psc=1&refRID=N6EREKB257TVMAZF86MB * ... treba este pogooglit	5	Radiant\Banana		Bc. Ľubomír Kurčák
15683	Vyhladanie rieseni	Task	Closed			Radiant\Banana	2	Tomas Sabo
15684	Analyza grid rieseni	Task	Closed			Radiant\Banana	2	Bc. Viktor Beňo
15685	Analyza breadboard rieseni	Task	Closed			Radiant\Banana	2	Bc. Erik Paľa
15673	**Navrh potencionalnych rieseni**	**User Story**	Closed	Grid/Breadboard STEM/Blocks zalezi od analyzy/modelov, kt. najdeme	8	Radiant\Banana		Bc. Ľubomír Kurčák

15687	Navrh blokoveho riesenia	Task	Closed			Radiant\Banana	3	Bc. Viktor Beňo
15688	Navrh breadboard riesenia	Task	Closed			Radiant\Banana	3	Bc. Erik Paľa

E.3. Export úloh k šprintu 3

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15728	**Vytvorit modely kociek**	**User Story**	Closed	Minimalne: * vsetky spojovacie kocky (priama, stvorita krizovatka, etc) * na zaklade zvolenych kociek vytvorit dalsie modely	13	Radiant\Cranberry		Bc. Ľubomír Kurčák
15734	Vytvorenie reliefovych modelov	Task	Closed	3D mesh relief		Radiant\Cranberry	4	Bc. Ľubomír Kurčák
15735	Vytvorenie texturovanych modelov	Task	Closed			Radiant\Cranberry	2	Bc. Ľubomír Kurčák
15736	Vytvorenie modelov s 3D modelmi suciastok	Task	Closed			Radiant\Cranberry	8	Bc. Ľubomír Kurčák
15727	**Navrhnut zoznam kociek**	**User Story**	Closed	pozriet si ake veci su podporovane v spice: * diody * kable? * baterka * ziarovka * etc	5	Radiant\Cranberry		Bc. Ľubomír Kurčák
15732	Ziskat zoznam podporovanych suciastiek	Task	Closed			Radiant\Cranberry	3	Bc. Viktor Beňo
15733	Vyber suciastiek na implementaciu	Task	Closed	k vybranym + voltmeter, ampermeter, meter odporu		Radiant\Cranberry	4	Tomas Sabo
15726	**Konstrukcia obvodov v gride**	**User Story**	Closed	Controller agnostic Mriezka ktora detekuje koliziu Zarovnanie rotacie suciastiek do gridu Grabber - pozicia urcena mys + raycast ked je nieco v rayi tak to chytim Co grabnut ked koliduju dve (najrozumnejsie asi podla vzdialenosti) Snapovanie podla rotacie, pozicie v gride Check/highlight, ktory objekt, pozicia v mriezke je najblisie Rotacia (zatial) v 2D (ked to je na mriezke)	15	Radiant\Cranberry		Bc. Ľubomír Kurčák

15729	Mriezka ktora detekuje koliziu	Task	Closed	rovnomerne ;D		Radiant\Cranberry	8	Bc. Patrik Tománek
15730	Grabber	Task	Closed	controller agnostic pozicia raycast sposob vyberania objektov Rotacia (zatial) v 2D (ked to je na mriezke)		Radiant\Cranberry	6	Tomas Sabo
15731	Highlightovanie a ICP	Task	Closed			Radiant\Cranberry	2	Bc. Ľubomír Kurčák
15723	**Navrh realizacie pre SpiceSharp**	**User Story**	Closed	Pekne casti, samply - obrazok, premysleny proces, zlozenie v gride az po vygenerovanie kodu	11	Radiant\Cranberry		Bc. Ľubomír Kurčák
15724	Opisat postup pre breadboard	Task	Closed	Treba opisat jednoduchy obvod, napriklad so ziarovkou		Radiant\Cranberry	5	Bc. Erik Paľa
15725	Opisat postup pre blokove riesenie	Task	Closed	Treba opisat jednoduchy obvod, napriklad so ziarovkou		Radiant\Cranberry	5	Bc. Viktor Beňo

E.4. Export úloh k šprintu 4

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15763	**Vytvorit scenu vo VR Interaction Frameworku**	**User Story**	Closed	integracia snapzones, grabberov, etc	8	Radiant\Date		Bc. Ľubomír Kurčák
15769	Vytvorit scenu s objektmi z VR Framework	Task	Closed			Radiant\Date	4	Bc. Patrik Tománek
15770	Vytvorit interakcie s objektmi	Task	Closed			Radiant\Date	4	Bc. Patrik Tománek
15773	Integracia VR emulatorov	Task	Closed			Radiant\Date	4	Bc. Patrik Tománek
15774	Porovnanie snap zones VR Frameworku a custom snap zones	Task	Closed			Radiant\Date	2	Bc. Patrik Tománek
15762	**Vytvorit demo so SpiceSharp**	**User Story**	Closed	vytvorit skript, ktory vie menit parametre za behu simulacie transient analysis skontrolovat ako dlho trva vypocet pre staticku/transient analyzu, ziskat cas (trvanie) custom models vytvorit build vytvorit vlastnu suciastku - vytvorit vodic ako extension rezistoru https://spicesharp.github.io/SpiceSharp/articles/custom_components/custom_models.html https://spicesharp.github.io/SpiceSharp/articles/tutorials/changing_parameters/changing_parameters_during_simulation.html	17	Radiant\Date		Bc. Ľubomír Kurčák
15765	Zmena parametrov v runtime	Task	Closed			Radiant\Date	4	Bc. Viktor Beňo
15766	Meranie casu statickej analyzy	Task	Closed			Radiant\Date	2	Tomas Sabo

15767	Vytvorit vlastnu suciastku	Task	Closed	Spravit novu dll. Vytvorit novy build. Vyskusat v Unity. https://github.com/SpiceSharp/SpiceSharp/blob/master/SpiceSharpTest/Examples/CustomResistor/NonlinearResistor.cs		Radiant\Date	4	Bc. Erik Paľa
15768	Meranie casu tranzientnej analyzy	Task	Closed			Radiant\Date	2	Tomas Sabo
15775	Vyhodnotenie casovych analyz a vytvorenie konecneho navrhu pouzitia simulatora elektrich obvodov	Task	Closed			Radiant\Date	1	Bc. Ľubomír Kurčák
15761	**Rozsirit typy modelov o dalsie suciastky**	**User Story**	Closed	mosfet dioda, pre porovnanie do breadboardu	7	Radiant\Date		Bc. Ľubomír Kurčák
15771	Vytvorit rozsirenie 3D modelov	Task	Closed			Radiant\Date	4	Bc. Ľubomír Kurčák
15772	Vytvorit rozsirenie reliefovych modelov	Task	Closed			Radiant\Date	4	Bc. Ľubomír Kurčák

E.5. Export úloh k šprintu 5

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15764	**Analyza synchronizacnych frameworkov**	**User Story**	Closed	photon bolt/PUN2/realtime cloud bolt user friendly smartfox server podpora pre voice chat photon ma photon voice 2	8	Radiant\Elderberry		Bc. Ľubomír Kurčák
15792	Analyza Photon Bolt	Task	Closed			Radiant\Elderberry	4	Bc. Viktor Beňo
15793	Analyza Smartfox Server	Task	Closed			Radiant\Elderberry	4	Bc. Ľubomír Kurčák
15795	Analyza Realtime Cloud	Task	Closed			Radiant\Elderberry	4	Bc. Viktor Beňo
15796	Vyhodnotenie analyzy	Task	Closed			Radiant\Elderberry	2	Bc. Erik Paľa
15797	Analyza PUN2	Task	Closed			Radiant\Elderberry	4	Tomas Sabo
15788	**VR Interaction Framework 2**	**User Story**	Closed	* snap zones rotation offset * integracia s existujucimi objektami * integracia outputov a inputov (uzly)	8	Radiant\Elderberry		Bc. Ľubomír Kurčák
15789	Snap zones rotation offset	Task	Closed			Radiant\Elderberry	6	Bc. Patrik Tománek
15790	Integracia s nasimi modelmi el. suciastok	Task	Closed			Radiant\Elderberry	3	Bc. Patrik Tománek

15791	Outputy inputy uzlov kontrola spojenia	Task	Closed			Radiant\Elderberry	5	Bc. Patrik Tománek

E.6. EXPORT ÚLOH K ŠPRINTU 6

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15818	**Integracia Unity Prototypu do SpiceSharp**	**User Story**	Closed		8	Radiant\Fig		Bc. Ľubomír Kurčák
15822	Analyza vstupu do SpiceSharp	Task	Closed	Vykonat analyzu vstupov, a zabezpecit obvodovo-agnosticku simulaciu.		Radiant\Fig	6	Bc. Viktor Beňo
15823	Implementovat tranzientny prekladac obvodov simulacie do jazyku SpiceSharp	Task	Closed	Integrovat obvodovo-agnosticky diagnosticky prekladac obvodov simulacie do obvodoveho simulatoru v jazyku SpiceSharp		Radiant\Fig	7	Bc. Patrik Tománek
15819	**Vytvorenie dokumentacie k existujucim castiam prototypu**	**User Story**	Closed	Zdokumentovanie existujucich casti prototypu za ucelom vytvorenia prvotnej pouzivatelskej prirucky.	8	Radiant\Fig		Bc. Ľubomír Kurčák
15820	Zdokumentovanie jednotlivych suciastiek	Task	Closed	Vytvorenie zoznamu a dokumentacia podporovanych suciastok a vytvorenie pouzivatelskeho dokumentu pre suciastky.		Radiant\Fig	6	Bc. Erik Paľa
15824	Zdokumentovanie 3D modelov suciastok	Task	Closed	Vytvorenie zoznamu modelov a ich dokumentacia a dokumentacia ich tvorby a vytvorenie pouzivatelskeho katalogu pre suciastky.		Radiant\Fig	4	Bc. Ľubomír Kurčák

15825	Vytvorenie dokumentacie konektivity suciastok	Task	Closed	Zdokumentovanie konektivity suciastok v ramci logickej reprezentacie elektrickeho obvodu.		Radiant\Fig	4	Bc. Patrik Tom

E.7. Export úloh k šprintu 7

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15821	**Vypracovanie navrhu simulovaneho vozidla**	**User Story**	Closed	vytvorit prve sketche/navrhy/modely * z akych casti sa sklada * ktore casti su fyzikalne * ktore casti suvisia s obvodom * navrhy ako bude vyzerat: svetla, etc * vytvorit dokument s dopismy	5	Radiant\Grapefruit		Bc. Ľubomír Kurčák
15826	Vytvorenie navrhu suciastok vozidla	Task	Closed	Vytvorenie navrhu suciastok vozidla, a vytvorit kategorizaciu na casti, ktore su mechanicke/fyzikalne, a tie ktore suvisia s elektrickym obvodom.		Radiant\Grapefruit	6	Bc. Ľubomír Kurčák
15827	Vytvorenie navrhu vozidla	Task	Closed	Vytvorenie navrhu hlavnej konstrukcie vozidla s podporou pre modularne suciastky.		Radiant\Grapefruit	4	Bc. Ľubomír Kurčák
15843	**Implementacia Parsera SpiceSharp**	**User Story**	Closed	https://www.nuget.org/packages/SpiceSharp/2.8.0 https://www.nuget.org/packages/System.Linq.Expressions/4.3.0 https://www.nuget.org/packages/SpiceSharpBehavioral/1.1.5 https://www.nuget.org/packages/SpiceSharp-Parser/1.5.0 napisat autorovi	13	Radiant\Grapefruit		Bc. Ľubomír Kurčák
15846	Integrovat parser do projektu	Task	Closed	implicitne zahrnut parser.		Radiant\Grapefruit	4	Bc. Viktor Beňo
15847	Komunikacia s autorom SpiceSharpParser-u	Task	Closed	kontaktovat autora o nekompatibilnych verziach spicesharp a spicesharp parser skusit kontaktovat Svena z SpiceSharp teamu		Radiant\Grapefruit	2	Bc. Viktor Beňo
15848	Vytvorit SpiceSharpParser demo scenu	Task	Closed	Vytvorit custom scenu pre SpiceSharpParser na otestovanie funkcionality balicka.		Radiant\Grapefruit	2	Bc. Patrik Tománek

15844	**Vytvorenie modelu auticka**	**User Story**	Closed	treba (mozno): -paku -pedale -volant -klakson -laser turret -radio -svetla -brzdy (dvojite mozno)	9	Radiant\Grapefruit		Bc. Ľubomír Kurčák
15849	Vytvorit zoznam existujucich modelov	Task	Closed	Najst vhodne low poly modely a analyzovat ich vyhody/nevyhody z hladiska vizualneho spracovania a funkcionalit potrebnych animacii na tychto spominanych modeloch o 2 riadky vyssie.		Radiant\Grapefruit	6	Bc. Ľubomír Kurčák
15850	Vytvorenie .FBX modelov pripravenych na import	Task	Closed	Z vsetkych najdenych modelov treba vytvorit vhodne .fbx modely vo vhodnom formate, ktore maju spravne nastaveny origin kazdeho meshu tohto .fbx modelu.		Radiant\Grapefruit	4	Bc. Ľubomír Kurčák
15845	**Analyza fyzikalnej simulacie vozidla**	**User Story**	Closed	analyza rieseni pre fyziku vozidla, smart kart package f = ma elasticke kolizie, odpruzenie https://learn.unity.com/project/karting-template https://learn.unity.com/tutorial/karting-mod-smart-karts-training-guide?uv=2019.3&projectId=5c82b27cedbc2a0e8db0c728#5ec832feedbc2a31b5a891f4	8	Radiant\Grapefruit		Bc. Ľubomír Kurčák
15852	Analyzovat najdene riesenia	Task	Closed	Analyza vsetkych najdenych rieseni ohladom fyzikalnej simulacie vozidla, spisat ich vyhody/nevyhody vzhladom na jednotlive faktory tychto fyzikalnych simulacii spominanych vozidiel.		Radiant\Grapefruit	5	Bc. Erik Paľa
15851	Najst existujuce riesenia fyzikalnej simulacie vozidla	Task	Closed	Najst rozne sposoby fyzikalnej simulacie vozidla ako napriklad taketo sposoby: arkada, atd.		Radiant\Grapefruit	5	Bc. Erik Paľa

E.8. EXPORT ÚLOH K ŠPRINTU 8

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15901	**Navrh sceny pre modifikaciu vozidla**	**User Story**	Closed	moznost: zmensenina vozidla, suciastky zapadaju do snap zony	8	Radiant\Huckleberry		Bc. Ľubomír Kurčák
15905	Rozhodnut sposob modifikovania vozidla	Task	Closed	zmensenina alebo 1:1 skala a akym sposobom sa umiestnuju suciastky		Radiant\Huckleberry	4	Bc. Ľubomír Kurčák

15906	Vytvorenie navrhu opisu sceny	Task	Closed	Vytvorenie navrhu opisu sceny pre modifikaciu vozidla		Radiant\Huckleberry	4	Bc. Ľubomír Kurčák
15907	Vytvorenie grafickej ilustracie navrhu	Task	Closed	Vytvorenie grafickej ilustracie navrhu pre vozidla		Radiant\Huckleberry	3	Bc. Ľubomír Kurčák
15908	Vytvorenie schem existujucich obvodov	Task	Closed	Vytvorenie schem existujucich obvodov, priklady schem ocakavanych od usera.		Radiant\Huckleberry	4	Bc. Erik Paľa
15902	**Vypis Spice kodu pre suciastky**	**User Story**	Closed	vypis vstupu pre Spice Sharp Parser	5	Radiant\Huckleberry		Bc. Ľubomír Kurčák
15909	Vytvorenie interaktivnej tabule	Task	Closed	Vytvorenie interaktivnej tabule pre zobrazenie SPICE kodu.		Radiant\Huckleberry	6	Bc. Patrik Tománek
15910	Pristupit k internym Spice kodom simulovanych suciastok	Task	Closed	Pristupit k internym agnostickym Spice kodom simulovanych suciastok pre moznost neskorsieho vypisu v programe.		Radiant\Huckleberry	2	Bc. Patrik Tománek
15903	**Vytvorenie prikladov pokrocilejsich obvodov**	**User Story**	Closed	oscilator buzzer zlozitejsie riadenie motora ako len cez odpor - na odpor pomaly motor - rychly cez mosfet (tranzistor riadeny napatim)? https://www.infineon.com/dgdl/irlb3036pbf.pdf?fileId=5546d462533600a40153566033ea2589 a dalsie z pohladu auta	13	Radiant\Huckleberry		Bc. Ľubomír Kurčák
15911	Vytvorenie obvodu pre oscilator	Task	Closed	Vytvorenie obvodu pre oscilator a otestovat uskutocnitelnost		Radiant\Huckleberry	6	Bc. Viktor Beňo
15912	Vytvorenie obvodu pre buzzer	Task	Closed	Vytvorenie obvodu pre buzzer a otestovat uskutocnitelnost		Radiant\Huckleberry	6	Bc. Viktor Beňo
15913	Ovladanie slabsieho a silnejsieho motora	Task	Closed	Ovladanie slabsieho a silnejsieho motora. Slabsi motor - ovladany pomocou odporu. Silnejsi motor - ovladany pomocou mosfetu.		Radiant\Huckleberry	8	Bc. Viktor Beň

E.9. Export úloh k šprintu 9

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15904	**Vytvorenie sceny pre modifikaciu vozidla**	**User Story**	Closed	Vytvorenie sceny pre modifikaciu vozidla selektovanie objektov vozidla, vzdy sa njakou formou nacita grid	11	Radiant\Kiwi		Bc. Patrik Tománek
15929	Vytvorenie interaktivnych komponentov vozidla	Task	Closed			Radiant\Kiwi	4	Bc. Patrik Tománek
15930	Vkladanie suciastok do gridboardu a spustenie gridboardu pre dany komponent	Task	Closed			Radiant\Kiwi	4	Bc. Patrik Tománek
15926	**Modifikacia modelu vozidla**	**User Story**	Closed	Modifikacia modelu vozidla animacie zabacania, ci sa to da menit pomocou animacneho parametru pozriet sa na fyzikalnu simulaciu, ako zmodifikovat model aby fungoval s standardnym ovladacom vozidla su vsetky modely oddelene, ktore potrebujeme (napriklad na highlight paky, svetiel, volantu, etc)	8	Radiant\Kiwi		Bc. Patrik Tománek
15931	Analyza a spracovanie animacii a animacnych parametrov	Task	Closed			Radiant\Kiwi	2	Bc. Erik Paľa
15932	Integracia modelu s fyzikalnou simulaciou	Task	Closed			Radiant\Kiwi	3	Bc. Patrik Tománek
15933	Rozdelenie modelu na interaktivne suciastky	Task	Closed			Radiant\Kiwi	3	Bc. Patrik Tománek

15927	**Testovanie oscilatorov a kondenzatorov SpiceSharp**	**User Story**	Closed	Testovanie oscilatorov a kondenzatorov SpiceSharp samplace kodu zaznamenane, vzory funkcnych oscilatorov a kondenzatorov skusit vyuzit fora, skusit ci to v LTspice funguje	13	Radiant\Kiwi		Bc. Viktor Beňo
15934	Vytvorit sample kodu oscilatorov	Task	Closed			Radiant\Kiwi	8	Bc. Viktor Beňo
15935	Vytvorit sample kodu pouzitia kondenzatorov	Task	Closed			Radiant\Kiwi	8	Bc. Viktor Beňo
15936	Vyskusanie simulacie v LTspice	Task	Closed			Radiant\Kiwi	6	Bc. Erik Paľa
15928	**Doplnenie modelov suciastiek**	**User Story**	Closed	Doplnenie modelov suciastiek vytvorenie sample obvodov v Unity, staci odvodit a SpiceSharp vstup fotka obvodu v Unity vstup pre SpiceSharp Parser (to co hodi do okna) vypisat do konzoly a prekopirovat z konzoly vytvorenie sample obvodov v Unity, staci odvodit a SpiceSharp vstup fotka obvodu v Unity vstup pre SpiceSharp Parser (to co hodi do okna) vypisat do konzoly a prekopirovat z konzoly	8	Radiant\Kiwi		Bc. Ľubomír Kurčák
15937	Vytvorenie modelu baterie	Task	Closed			Radiant\Kiwi	2	Bc. Ľubomír Kurčák
15938	Vytvorit modelu cievky	Task	Closed			Radiant\Kiwi	2	Bc. Ľubomír Kurčák
15939	Vytvorenie modelu capacitoru	Task	Closed			Radiant\Kiwi	2	Bc. Ľubomír Kurčák
15940	Vytvorenie modelu motora	Task	Closed			Radiant\Kiwi	2	Bc. Ľubomír Kurčák
15941	Vytvorenie modelu reproduktoru	Task	Closed			Radiant\Kiwi	2	Bc. Ľubomír Kurčák
15942	Vytvorenie modelu oscilatoru	Task	Closed	https://www.google.com/imgres?imgurl=https%3A%2F%2F5.imimg.com%2Fdata5%2FJP%2FVL%2FMY-45609829%2F12-mhz-crystal-oscillator-quartz-crystal-500x500.png&imgrefurl=https%3A%2F%2Fwww.indiamart.com%2Fproddetail%2F12-		Radiant\Kiwi	2	Bc. Ľubomír Ku

				mhz-crystal-oscillator-quartz-crystal-18796453473.html&tbnid=iT_iWBRB5yaZAM&vet=12ahUKEwjoyaTT-4LwAhWDgM4BHTsHCjIQMygBegUIARCVAg..i&docid=TIIIPE_HgEuKMM&w=483&h=482&q=crystal%20oscillator&ved=2ahUKEwjoyaTT-4LwAhWDgM4BHTsHCjIQMygBegUIARCVAg				

E.10. EXPORT ÚLOH K ŠPRINTU 10

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
15969	**Vytvorenie symbolickych komponentov vozidla**	**User Story**	Closed	ked vlozis komponent do policka ides rrobit pre vozidlo suciatsku gridboard obvod a ked tam hodis tu ten tu vozidlo suciastku ze ides prenho robit komponent tak sa ti tam automaticky da na pevno ten komponent njaka zmensenina nech ti to nevykresluje taky pocet polygonov aby to bolo mensie aby si nezmensoval model	5	Radiant\Lingonberry		Bc. Ľubomír Kurčák
15970	Vytvorenie symbolickeho komponentu motoru	Task	Closed			Radiant\Lingonberry	4	Bc. Ľubomír Kurčák
15971	Vytvorenie symbolickeho komponentu ziarovky	Task	Closed			Radiant\Lingonberry	4	Bc. Ľubomír Kurčák
15972	Vytvorenie symbolickeho komponentu klaksonu	Task	Closed			Radiant\Lingonberry	4	Bc. Ľubomír Kurčák
15966	**Integracia modelu vozidla do sceny pre modifikaciu**	**User Story**	Closed	vytvorenie symbolickej kocky pre suciastku auta v gridboarde nahradit kocky za realne suciastky vyriesit stratenie suciastky a jej navrat do vozidla	13	Radiant\Lingonberry		Bc. Patrik Tománek
15973	Nahradit kocky za realne suciastky	Task	Closed			Radiant\Lingonberry	4	Bc. Patrik Tománek

15974	Implementovat riesenie stratenie suciastky a jej navrat do vozidla	Task	Closed			Radiant\Lingonberry	4	Bc. Patrik Tománek
15967	**Vytvorenie vzorovych obvodov s pouzitim Oscilatora**	**User Story**	Closed		13	Radiant\Lingonberry		Bc. Viktor Beňo
15975	Vytvorenie prikladoveho obvodov	Task	Closed			Radiant\Lingonberry	4	Bc. Viktor Beňo
15976	Otestovanie prikladoveho obvodov	Task	Closed			Radiant\Lingonberry	4	Bc. Viktor Beňo
15977	Vytvorenie elektrickych schem obvodov	Task	Closed			Radiant\Lingonberry	4	Bc. Erik Paľa
15968	**Vytvorenie osciloskopu**	**User Story**	Closed		8	Radiant\Lingonberry		Bc. Erik Paľa
15978	Vytvorenie zaznamu osciloskopickych dat	Task	Closed			Radiant\Lingonberry	4	Bc. Erik Paľa
15979	Zobrazenie zaznamu osciloskopickych dat	Task	Closed			Radiant\Lingonberry	4	Bc. Erik Paľa

E.11. EXPORT ÚLOH K ŠPRINTU 11

ID	Title	Work Item Type	State	Description	Story Points	Iteration Path	Original Estimate	Assigned To
16029	Vytvorit pouzivatelsku prirucku	User Story	Closed		6	Radiant\Mango		Bc. Erik Paľa
16032	Prirucka zakladneho ovladania	Task	Closed			Radiant\Mango	4	Bc. Erik Paľa
16033	Prirucka tvorenia obvodov	Task	Closed	tutorial ako poskladat prikladovy obvod vsetko od zaciatku menenie napatia osciloskop vkladanie suciastok do gridu ziskanie suciastok zo snap zon pridanie noveho modelu suciastky (SPICE kod z clipboardu) zoberie suciastku z auta a otvori sa gridboard chyti motor nabehne gridboard motora		Radiant\Mango	4	Bc. Erik Paľa
16035	Prirucka ovladania vozidla	Task	Closed			Radiant\Mango	4	Bc. Ľubomír Kurčák
16030	Finalizacia uloh zapajania obvodov	User Story	Closed		6	Radiant\Mango		Bc. Viktor Beňo
16039	Implementovanie schem do hry	Task	Closed			Radiant\Mango	4	Bc. Patrik Tománek
16040	Testovanie korektnej funkcionality zapojenych obvodov	Task	Closed			Radiant\Mango	4	Bc. Viktor Beňo
16031	Optimalizacia pouzivatelskeho zazitku	User Story	Closed		5	Radiant\Mango		Bc. Patrik Tománek
16036	Uprava snap zon suciastkoveho menu	Task	Closed			Radiant\Mango	2	Bc. Patrik Tománek

16037	Zjednodusenie manipulacie suciastok vo vozidle	Task	Closed			Radiant\Mango	2	Bc. Patrik Tománek
16038	Vypnutie nepotrebnych kolizii	Task	Closed			Radiant\Mango	2	Bc. Ľubomír Ku

PRÍLOHA F. WEBOVÉ SÍDLO PROJEKTU

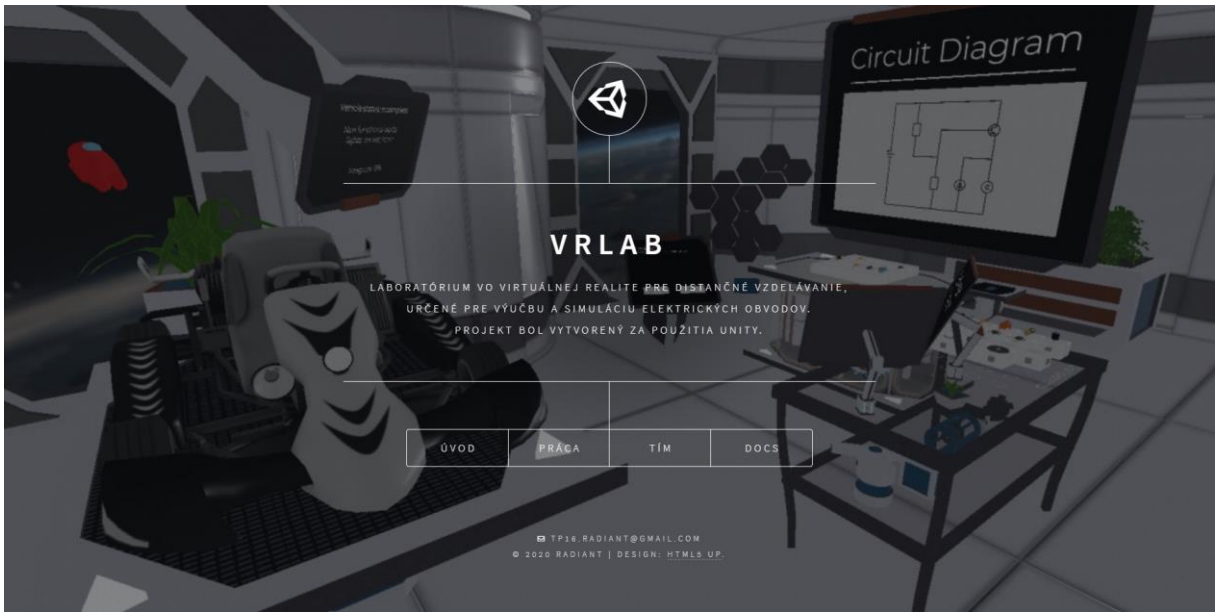
Autor prílohy: Patrik Tománek

Webová stránka nášho tímu sa nachádza na adresách:

- http://labss2.fiit.stuba.sk/TeamProject/2020/team16/

Webová stránka bola vytvorená v 1. šprinte. Pracoval na nej Patrik Tománek. Jedná sa o webstránku, na ktorej sa používateľ naviguje pomocou menu v strede obrazovky.

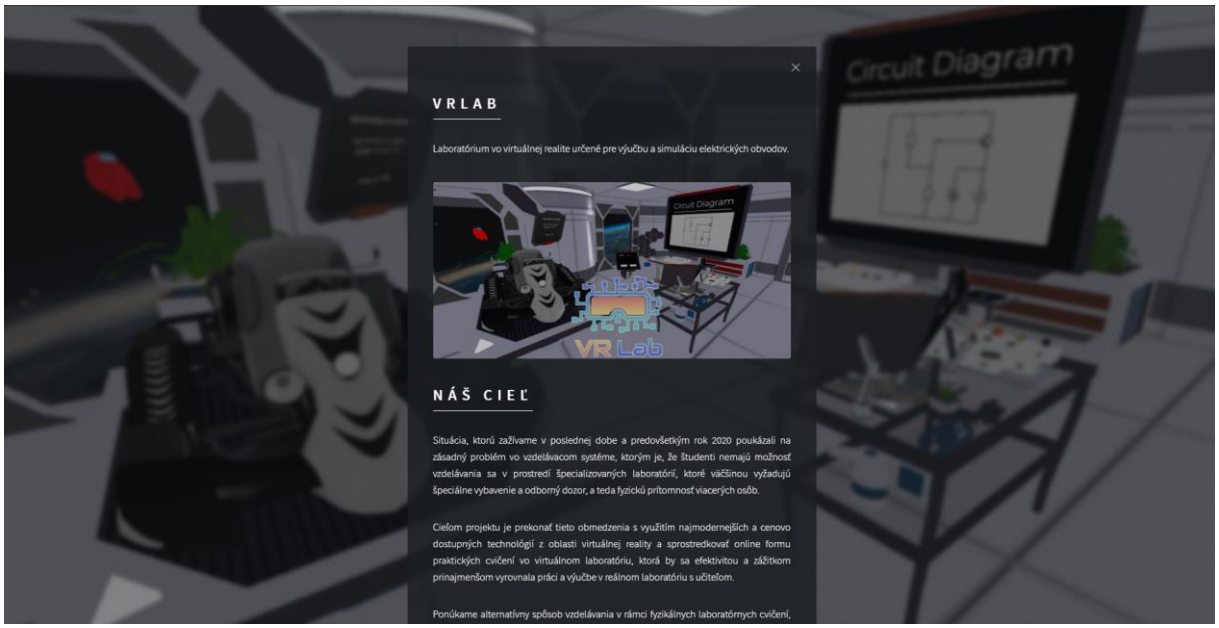
Na *Obrázok F.1* môžeme vidieť úvodnú stranu nášho webového sídla. Na samotnom spodku stránky sa nachádza kontakt na náš tím ako aj odkaz na voľne dostupné šablóny, s ktorými sme sa pri tvorbe stránky inšpirovali.



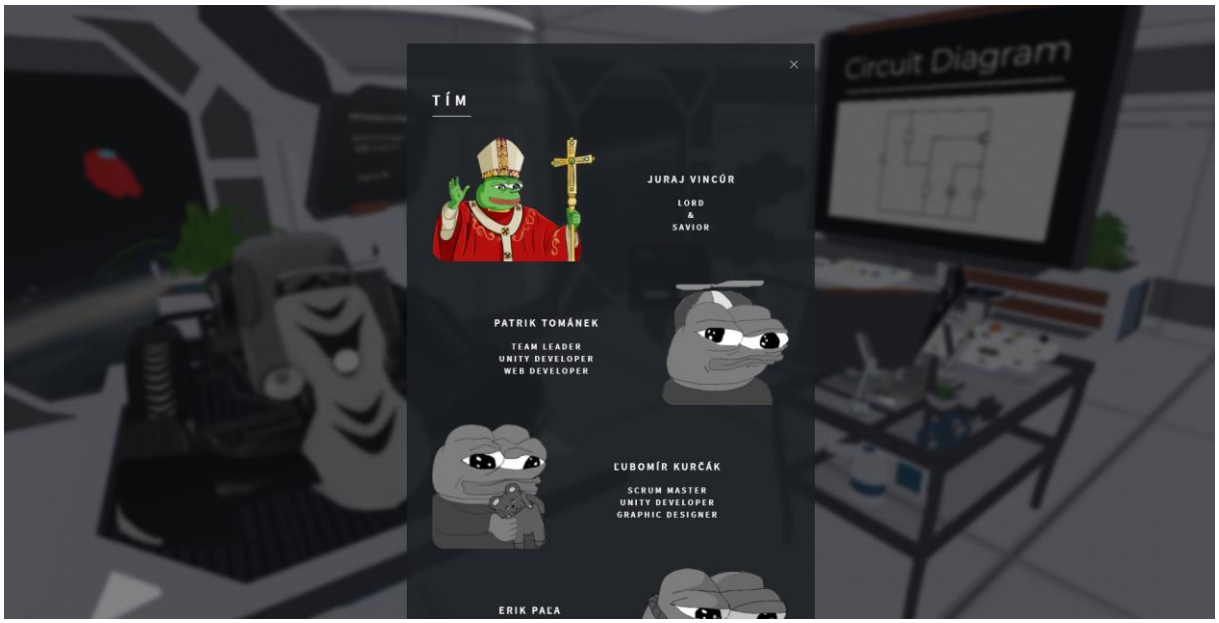
Obrázok F.1 – Úvodná strana

V strede sa nachádza názov projektu spolu s krátkym úvodom, čím sa náš projekt zaoberá. Pod úvodom môžeme vidieť jednotlivé sekcie na našej webovej stránke. Na *Obrázok F.2*, je zobrazený podrobnejší úvod, a teda charakterizácia nášho projektu, spolu s motiváciou, prečo sme sa preň rozhodli. Na *Obrázok F.3*, je uvedená naša stručná dokumentácia práce na projekte. Na *Obrázok F.4,* je výpis členov nášho tímu, na vrchu sa nachádza vedúci nášho projektu a následne ostatní členovia tímu. Ku každému členovi prislúcha obrázok, jeho meno a role. Obrázky členov tímu sa vyfarbia pokiaľ na nich používateľ posunie kurzor. Hlavným obsahom našej stránky sú dokumenty, ktoré sa pravidelne aktualizujú. Na Obrázok F.5, môžeme vidieť prvotnú stránku dokumentov, a teda jednotlivé zápisnice. Ďalej sa v tejto sekcií nachádzajú aj ostatné dokumenty ako retrospektívy, backlogy a ostatné dokumenty, medzi ktoré patria naše metodiky, analýzy a návrhy a iné.

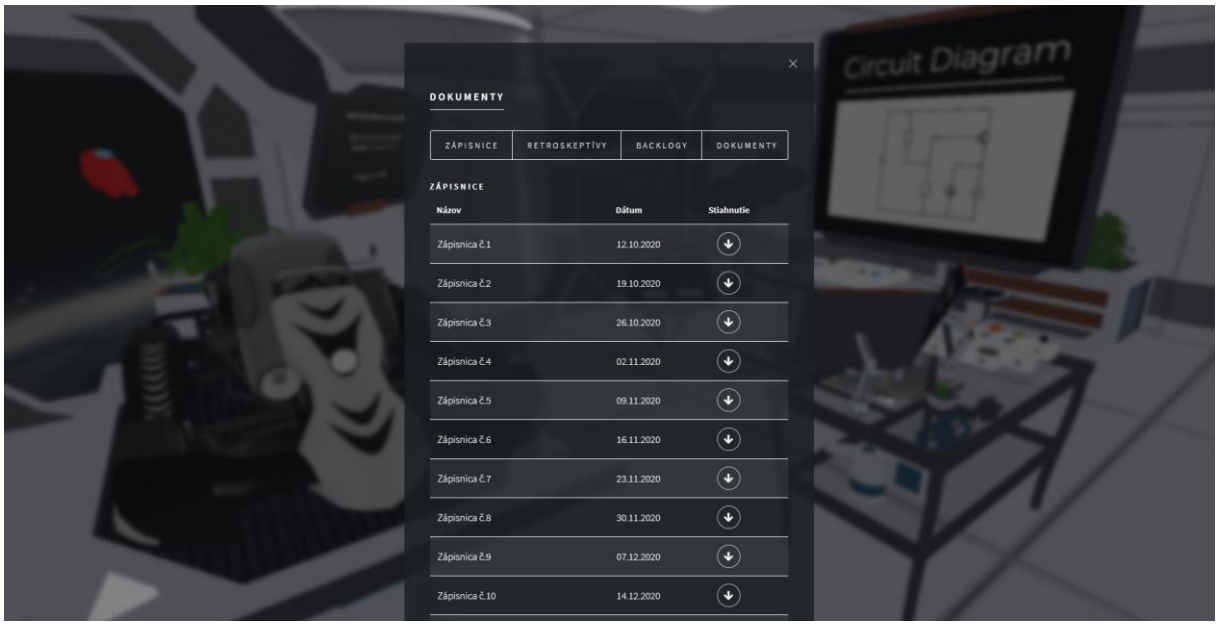
Obrázok F.2 - Charakterizácia a motivácia nášho projektu



Obrázok F.3 - Stručná dokumentácia práce na projekte



Obrázok F.4 - Členovia tímu (Spojenie 2 obrázkov do jedného)



Obrázok F.5 - Dokumentová sekcia stránky

Fakulta informatiky a informačných technológií
Slovenská technická univerzita

VRLab Radiant

Dokumentácia k inžinierskemu dielu

(2. kontrolný bod)

Akademický rok: 2020/2021

Pedagogický vedúci: Ing. Juraj Vincúr

Členovia tímu (č.16): Bc. Patrik Tománek
Bc. Ľubomír Kurčák
Bc. Erik Paľa
Bc. Viktor Beňo

OBSAH

1 Úvod 1
2 Globálne ciele pre zimný semester 2
3 Globálne ciele pre letný semester 2
4 Celkový pohľad na systém 2
5 Moduly systému 4
5.1 Breadboard riešenie 4
5.1.1 Analýza 4
5.1.2 Návrh 4
5.2 Blokové (Grid) riešenie 5
5.2.1 Analýza 5
5.2.2 Návrh 5
5.3 Prekladač modelu na obvod pre breadboard riešenie 7
5.3.1 Analýza 8
5.3.2 Návrh 8
5.4 Prekladač modelu na obvod pre blokové (grid) riešenie 9
5.4.1 Analýza 9
5.4.2 Návrh 10
5.5 Vyhodnotenie návrhov a riešení 12
5.6 Synchronizačné frameworky 12
5.6.1 Analýza 12
5.6.2 Porovnanie frameworkov v záujmových kategóriách 15
5.6.3 Zhodnotenie 16
5.7 Analýza možností pre simuláciu vozidla 16
5.7.1 Analýza 16
5.7.2 Zhodnotenie 17
6 Prototypy 17
6.1 Prototyp modelov súčiastok 17
6.1.1 Analýza 17
6.1.2 Návrh 17
6.1.3 Implementácia 19
6.2 Prototyp pre ovládanie a interakciu 20
6.2.1 Analýza 20
6.2.2 Návrh 20
6.2.3 Implementácia 21

6.2.4 Testovanie 23
6.3 Prototyp pre podporu virtuálnej reality 24
6.3.1 Analýza 24
6.3.2 Návrh 25
6.3.3 Implementácia 25
6.3.4 Testovanie 27
6.4 VRLAB PROTOTYP 27
6.4.1 Návrh 28
6.4.2 Implementácia 31
6.4.3 Testovanie 34
Záver 35

Príloha A. Používateľská príručka
Príloha B. Zdrojové kódy
Príloha C. Protokol z testovania
Príloha D. Generovaná technická dokumentácia

1 Úvod

Tento dokument predstavuje dokumentáciu k inžinierskemu dielu projektu v rámci predmetu Tímový projekt. Cieľom nášho projektu je vytvorenie laboratória vo virtuálnej realite pre dištančné vzdelávanie, určené pre výučbu a simuláciu elektrických obvodov.

Situácia, ktorú zažívame poslednej dobe a predovšetkým rok 2020 poukázali na zásadný problém vo vzdelávacom systéme, ktorým je, že študenti nemajú možnosť vzdelávania sa v prostredí špecializovaných laboratórií, ktoré väčšinou vyžadujú špeciálne vybavenie a odborný dozor, a teda fyzickú prítomnosť viacerých osôb.

Cieľom projektu je prekonať tieto obmedzenia s využitím najmodernejších a cenovo dostupných technológií z oblasti virtuálnej reality a sprostredkovať online formu praktických cvičení vo virtuálnom laboratóriu, ktorá by sa efektivitou a zážitkom prinajmenšom vyrovnala práci a výučbe v reálnom laboratóriu s učiteľom.

Ponúkame alternatívny spôsob vzdelávania v rámci fyzikálnych laboratórnych cvičení, kde môžu študenti nadobudnúť praktické skúsenosti, ktoré nevyžadujú ich fyzickú prítomnosť v špecializovanom laboratóriu. Náš projekt takisto poskytuje ochranu študentov, ako aj učiteľov pred možnými haváriami počas experimentovania, ktoré môžu mať za následok ako zdravotné problémy, tak aj materiálne straty. Veríme, že virtuálna realita má uplatnenie vo vzdelávaní, pretože rôzne experimenty sú bezpečnejšie a prístupnejšie bez potreby ďalších fyzických zdrojov okrem samotného VR headsetu.

Súčasťou tohto dokumentu je opísanie globálnych cieľov nášho projektu v rámci zimného semestra, celkový pohľad na vytváraný systém, prehľad doposiaľ vytvorených modulov systému a na koniec opis prototypu, ktorý sme vytvorili. Prototyp bol vytvorený v Unity a modely, s ktorými sme pracovali sme si vytvorili v grafickom prostredí Blender.

2 GLOBÁLNE CIELE PRE ZIMNÝ SEMESTER

Autor kapitoly: Ľubomír Kurčák, Patrik Tománek

Za zimný semester by sme chceli vytvoriť jednoduchý funkčný prototyp, ktorý nám napovie ďalší smer vývoja. Cieľom je vytvoriť program, v ktorom bude možné vytvárať elektrické obvody zložené z blokových súčiastok uložených do mriežky. Správanie vytvoreného obvodu bude simulované a výstupy budú použité na ovplyvnenie virtuálneho prostredia, napríklad rozsvietená žiarovka, krútiaci sa moment kolies, a iné.

3 GLOBÁLNE CIELE PRE LETNÝ SEMESTER

Autor kapitoly: Erik Paľa

Naším primárnym cieľom pre letný semester je vytvoriť finálnu verziu prototypu nášho projektu, v ktorom bude možné zostavovať a simulovať ľubovoľné elektrické obvody. Tento cieľ je možné rozdeliť na dva menšie ciele, z ktorých prvý a hlavný je zhotovenie prototypu aplikácie, v ktorej bude možné vytvoriť a simulovať rôzne elektrické obvody podľa potrieb používateľa. Druhým cieľom je implementácia scenára s elektrickým autíčkom, kde bude používateľ pre spojazdnenie autíčka musieť poskladať obvody pre jeho jednotlivé komponenty, z ktorých primárny je motor.

4 CELKOVÝ POHĽAD NA SYSTÉM

Autori kapitoly: Ľubomír Kurčák, Patrik Tománek

Celkový pohľad na systém sa dá opísať ako laboratórne virtuálne prostredie, v ktorom používatelia môžu vytvárať simulovaný elektrický obvod a experimentovať a testovať jeho správanie. Používatelia interagujú so skladacou mriežkou, do ktorej vkladajú elektronické súčiastky. Z mriežky so súčiastkami sa následne vygeneruje vstup pre simulátor elektrických obvodov. Tento krok je úmyselne oddelený, čo nám poskytuje nezávislosť od jednotlivých simulátorov, a takisto nám umožňuje jednoduchšiu rozšíriteľnosť pre nové súčiastky. Súčiastky v obvode sa dajú kategorizovať do troch skupín z pohľadu energie; zdroje, ako je napríklad baterka, manipulátory toku, ako rezistory a vodiče a spotrebiče energie, ako napríklad žiarovka, elektrický motor a ďalšie. Výstup zo simulátora nám vráti napätie v jednotlivých častiach obvodu, na základe ktorých vieme simulovať jeho správanie v hernej logike a zobraziť výstupy audiovizuálnou formou.

Mriežka simulovaného obvodu je dvojrozmerná a má rozmery m×n, ktoré sú parametrizovateľné. Je možné uvažovať o rozšírení do troch rozmerov, no túto variantu sme z hľadiska jednoduchosti použitia, tvorby a ladenia obvodu nezvolili. Je potrebné zvoliť také parametre veľkosti, ktoré umožňujú dostatočnú komplexnosť obvodovej logiky na splnenie úloh, či tvorbu zaujímavej funkcionality. Na druhú stranu, čím väčší obvod simulujeme, tým dlhšie budú výpočty trvať, čo môže zapríčiniť nízku obnovovaciu frekvenciu simulácie a zobrazenia a teda stratu interaktivity. Na každú pozíciu v mriežke vieme vložiť jednu súčiastku. Každá zo súčiastok môže byť pripojená ku každej susednej súčiastke v mriežke. Niektoré súčiastky môžu mať viac pripojovacích soketov ako iné. Napríklad súčiastka ako baterka má dva sokety, jeden ktorý predstavuje kladné a druhý záporné napatie. Ďalej napríklad rozvetvenie vodiča predstavuje trojvetvovú križovatku, a teda prislúchajúca súčiastka bude mať 3 sokety. Súčiastky je pri vkladaní možné orientovať do štyroch hlavných smerov mriežky. Jedná sa o

rotácie o 0°, 90°, 180° a 270°. Pri rotáciách sa pozície soketov menia, čím je možné vytvoriť z tej istej pozície prepojenia na iné súčiastky.

Používatelia vložia súčiastky do mriežky, ktoré následne prekladáme na vstup, ktorý vie spracovať simulátor elektrických obvodov. V tomto kroku je potrebné určiť susedenosť a prepojenia jednotlivých súčiastok. Tieto informácie sú určené pozíciou a orientáciou súčiastok. Susedné súčiastky, ktoré majú na spoločnej hrane obe voľný soket považujeme za spojené. Vygenerovaný opis obvodu dáme následne na vstup simulátoru. Výstup simulátoru opäť interpretujeme a použijeme ako výsledok hernej logiky, ktorý graficky vizualizujeme.

Používatelia môžu program ovládať buď pomocou nástrojov virtuálnej reality, alebo klasickou klávesnicou a myšou. Program je navrhnutý aby bol nezávislý od typu vstupu. Voľba predmetu na interakciu je vykonaná pretnutím lúču s najbližším predmetom k začiatku lúča. So zvoleným predmetom môže používateľ voľne manipulovať a umiestniť ho do mriežky. Niektoré súčiastky majú ďalšie funkcie, ktoré je možné ovládať. Používateľ môže použiť voltmeter na merania v ľubovoľných častiach v obvode.

Používatelia majú k dispozícií interaktívnu tabuľu, na ktorú si môžu písať ľubovoľné poznámky, ktoré potrebujú, pričom si môžu vybrať z niekoľko rôzne farebných fixiek.

Používatelia majú takisto k dispozícii vlastnú plne interaktívnu motokáru. Pokiaľ si budú chcieť na túto motokáru nasadnúť a otestovať ju jazdou po laboratóriu, najprv ale budú musieť skonštruovať obvody pre jednotlivé komponenty ako motor, klaksón a svetlá. Počas konštrukcie obvodu si budú môcť nastaviť ako intenzívne budú tieto svetlá, akou hlasitosťou a frekvenciou tónu bude klaksón hrať, alebo ako rýchly ich motor bude. K dispozícii im ponúkame aj ukážkové obvody pre jednotlivé časti, pokiaľ by si používatelia nevedeli tieto obvody zostrojiť sami. Ukážkové obvody sú k dispozícii vo forme nákresov na papieri, ktoré sú umiestnené na stole vedľa interaktívnej tabule. Tieto papiere sú takisto interaktívne, takže si ich môžu používatelia zobrať so sebou kam potrebujú.

5 MODULY SYSTÉMU

V tejto kapitole je rozpísaných niekoľko modulov, ktoré sú súčasťou nášho projektu. Každý modul je rozpísaný v samostatnej podkapitole spolu s autorom alebo autormi, ktorí na danom module pracovali.

5.1 BREADBOARD RIEŠENIE

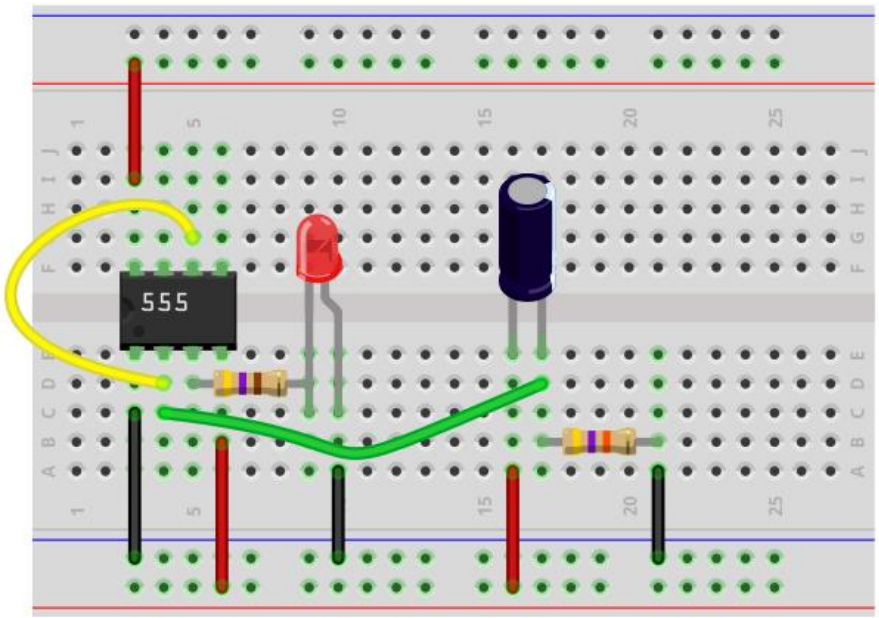
Autor podkapitoly: Erik Paľa

5.1.1 Analýza

Riešenie takéhoto typu by predstavovalo simuláciu reálneho zapojenia obvodu v realite. Toto riešenie umožňuje vytvárať komplexné zapojenia, kde používateľ nie je obmedzený počtom komponentov v uzle.

Výhodou tohto riešenia je to, že pre jednotlivé komponenty by boli použité ich reálne obrazy. Ďalšou veľkou výhodou tohto riešenia je taktiež možnosť vyskúšať si zapojenie zo simulátora v realite, kde reálny obvod by bolo možné zapojiť presne podľa zapojenia v simulátore.

Nevýhodou tohto riešenia je to, že v porovnaní s blokovým riešením je pre neskúseného používateľa oveľa zložitejšie na pochopenie. Ďalšou veľkou nevýhodou vysoká náročnosť na zhotovenie riadiacej logiky pre simuláciu takýchto zapojení. V porovnaní s blokovým riešením je nevýhodou aj to, že je potrebné riešiť pozície jednotlivých nožičiek pre každú súčiastku.



Obrázok 1 - Príklad breadboard zapojenia

5.1.2 Návrh

Základ tohto riešenia by predstavoval breadboard a pre jednotlivé komponenty by boli použité ich reálne obrazy, príklad breadboard zapojenia môžete vidieť na Obrázok 1

Samotná doska by bola rozdelená na jednotlivé uzly, kde v stredných radoch pinov jeden stĺpec predstavuje jeden uzol a pri vrchných a spodných radoch, ktoré sú oddelené od stredných, jeden riadok reprezentuje jeden uzol. Veľkosti uzlov je možné ľubovoľne zväčšiť tak, že vodičom spojíme 2 samostatné uzly.

Výhodou oproti blokovému riešeniu je, že nie je potrebné riešiť počet nožičiek, ktoré súčiastka má, nakoľko breadboard umožňuje zapojenie komponentov s ľubovoľným počtom nožičiek.

5.2 BLOKOVÉ (GRID) RIEŠENIE

Autor podkapitoly: Viktor Beňo

5.2.1 Analýza

Riešenie takéhoto typu by sa podobalo na stavebnice zapájania elektrického obvodu, v ktorom sú el. komponenty kocky(alebo iné vhodné tvary) zapájajúce sa do pripravenej podložky, príklad môžete vidieť na Obrázok 2. Na tejto podložke sú vyznačené miesta, na ktoré sa bloky stavebnice ukladajú a väčšinou sa jedná o mriežku s rozmermi bloku.

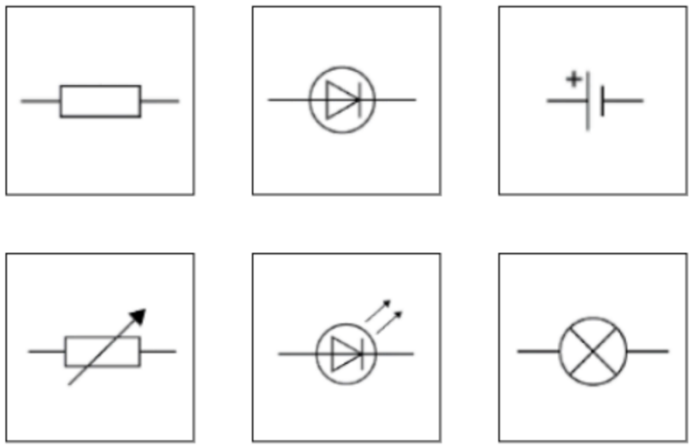
Elektrický obvod, zapojený s použitím týchto blokov pôsobí prehľadnejšie a je ľahší na pochopenie. Taktiež by bolo takéto riešenie jednoduchšie na implementáciu, zvlášť ak by boli použité na spájanie diskrétne bloky rovnakej veľkosti, ktoré sa spájajú magneticky.

Obrázok 2 - Produkt IQube ako príklad blokového riešenia

5.2.2 Návrh

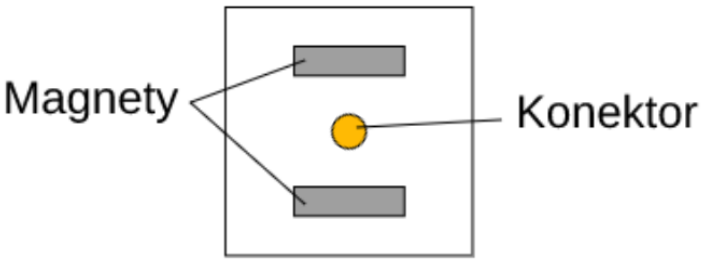
Použité by boli kocky predstavujúce elektrické komponenty ako hlavné stavebné bloky elektrického obvodu. Zapájali by sa do plochej dosky s mriežkovým vzorom pre pravidelné usporiadanie, ktoré bude pôsobiť prehľadne.

Kocky by mali na vrchnej strane znázornené schematické značky elektrických komponentov, ktoré prestavujú. Na Obrázok 3 je pohľad zhora pre šesť kociek elektrických komponentov.



Obrázok 3 - Návrh kociek blokového riešenia pri pohľade zhora

Kocky budú spájané magneticky pre jednoduchšie skladanie obvodov. Na bočných stranách, podľa typu elektrického komponentu, sa budú nachádzať dva magnety pre správne spojenie kociek a potrebný počet konektorov, príklad môžete vidieť na Obrázok 4.

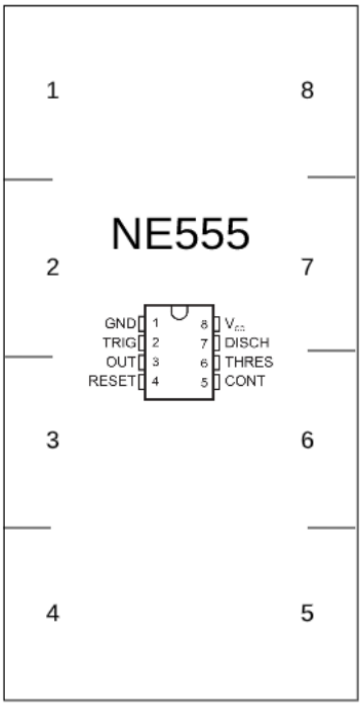


Obrázok 4 - Návrh kocky blokového riešenia pri pohľade z boku

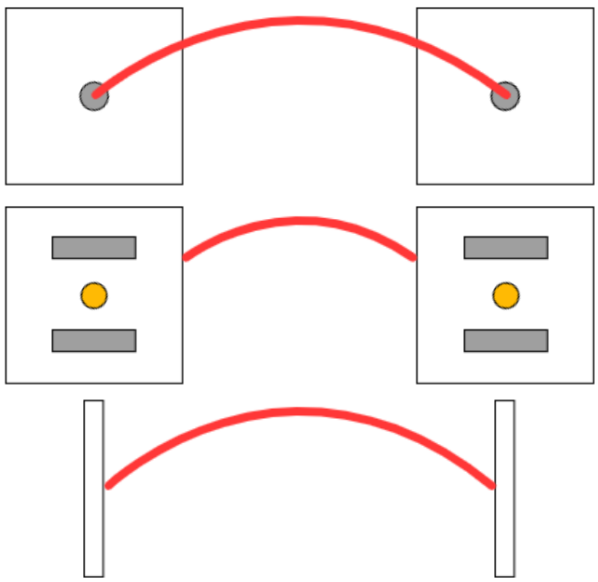
Pri súčiastkach s väčším počtom zapojení je najľahším riešením vytvorenie väčšieho bloku, kde pre každé potrebné zapojenie vstupu alebo výstupu elektrického komponentu bude použitá jedna

kocka. Takýto blok, zobrazený na Obrázok 5, sa dá aj redukovať na menší počet kociek, avšak výsledný menší blok môže pôsobiť neprehľadne.

Obrázok 5 - Návrh pre viac blokovú súčiastku



Kocky by sa mohli prepájať prípadne aj káblami, ktoré by sa tiež pripájali magneticky. Takýto príklad môžete vidieť na Obrázok 6.



Obrázok 6 - Návrh pre káblové prepojenie blokov

5.3 PREKLADAČ MODELU NA OBVOD PRE BREADBOARD RIEŠENIE

Autor podkapitoly: Erik Paľa

5.3.1 Analýza

Pri breadboard modeli bude treba riešiť zapojenie komponentov do uzlov. Pri breadboarde sú jednotlivé uzly reprezentované stĺpcami na vnútornej časti dosky a dvoma riadkami pinov na vrchu a na spode dosky.

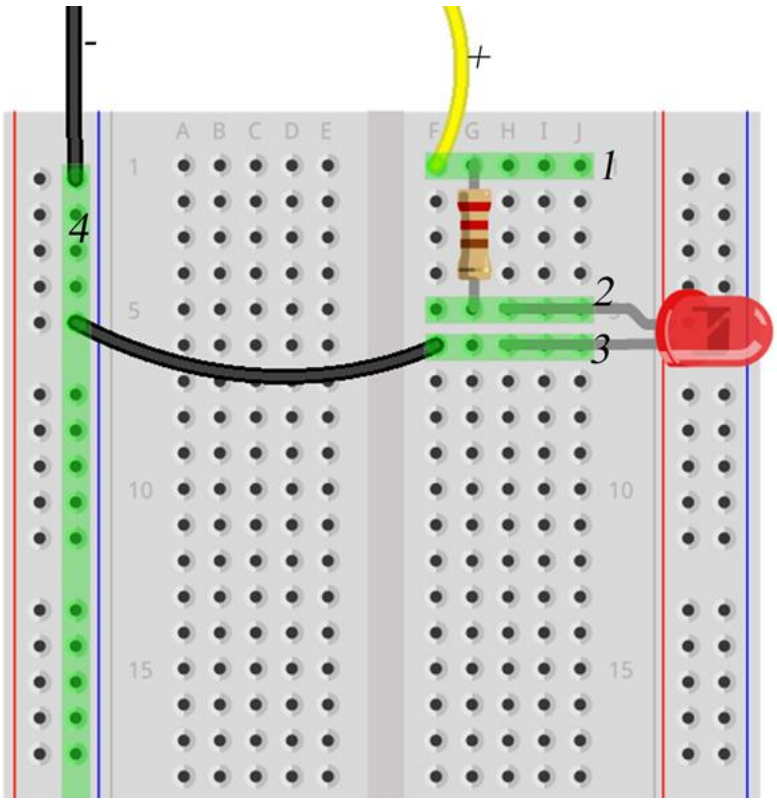
Pre tvorbu samotného obvodu poskytuje *SpiceSharp* triedu *Circuit*, do ktorej sa následne môžu pridávať jednotlivé súčiastky z knižnice. Pri pridávaní komponentov je potrebné nastaviť ich názov, hodnoty, uzly, v ktorých je súčiastka zapojená a prípadne iné parametre.

SpiceSharp má implementovanú funkcionalitu pre vytváranie uzlov (*nodes*), kde ak súčiastky majú uzol so spoločným názvom, považujú sa za spojené.

5.3.2 Návrh

V *SpiceSharp*-e sa súčiastky pripájajú k uzlom označeným ako "nodes", ktoré sú označené premennou typu *String*. Vzhľadom na to, že pri breadboarde predstavuje pri vnútorných riadkoch pinov každý stĺpec 1 uzol a pri vrchných a spodných dvoch radoch pinov je každý riadok 1 uzol, je potrebné nastaviť jednotlivé piny na doske tak, aby patrili uzlu, ktorý reprezentuje daný stĺpec alebo riadok.

Jednotlivé piny sú reprezentované dvoma hodnotami, kde písmeno reprezentuje riadok a číslo stĺpec v ktorom sa pin nachádza. Uzly je možné reprezentovať samostatnou sadou súradníc, kde sa pre vnútornú sadu už neriešia jednotlivé riadky ale táto časť dosky bude rozdelená len na vrchnú a spodnú polovicu. Čísla stĺpcov zostávajú zachované, nakoľko každý stĺpec predstavuje 1 uzol. Pre správnu funkcionalitu uzlov je teda potrebné nastaviť, ktoré riadky na doske sú v ktorej polovici a teda do ktorého uzla spadajú.



Obrázok 7 - Jednoduchý elektrický obvod pri breadboard riešení

Na Obrázok 7 môžeme vidieť jednotlivé uzly vyznačené zelenou farbou a označené číslami. Pri zostrojení jednoduchého obvodu, ktorý pozostáva zo zdroja, rezistora a diódy je pri pridávaní jednotlivých komponentov potrebné nastaviť pozíciu ich nožičiek korektne, aby obvod mohol fungovať.

Pri pridávaní rezistora je potrebné nastaviť mu hodnotu odporu, ktorú má v obvode predstavovať a polohu nožičiek, ktoré v tomto konkrétnom prípade sú pre prvú nožičku G1 a pre druhú nožičku G5, kde G1 spadá do prvého uzla a G5 do druhého uzla. Následne pri pridávaní diódy je potrebné dbať na jej otočenie. Pri jej pridaní do obvodu sa nastaví poloha nožičiek, kde prvá nožička (anóda) má súradnice H5 a druhá nožička (katóda) má súradnice H6. Nakoľko rezistor aj dióda majú nožičku v uzle 2 (stĺpec 5), sú považované za prepojené. Zdroj tohto obvodu je pripojený do uzla 1 a uzla 4. Môžeme si všimnúť, že ako bolo skôr v dokumente uvedené, celý spodný riadok predstavuje jeden uzol. Celý obvod sa uzavrie pridaním prepojenia medzi uzlom 3 a uzlom 4.

Pri vytváraní elektrického obvodu na breadboard-e je potrebné ošetriť, aby sa pri nastavovaní súradníc pre nožičky súčiastok nevyskytla situácia, že oba nožičky jednej súčiastky by sa nachádzali v rovnakom uzle.

5.4 Prekladač modelu na obvod pre blokové (grid) riešenie

Autor podkapitoly: Viktor Beňo

5.4.1 Analýza

Pri blokovom modeli bude treba riešiť spájanie blokov ako elektrických komponentov a ich pridanie do výsledného elektrického obvodu, ktorý bude simulovaný v *SpiceSharp*-e. Bloky sa budú dať rotovať takže sa bude musieť riešiť správne zapojenie súčiastok v obvode.

SpiceSharp poskytuje triedu *Circuit* pre vytvorenie elektrického obvodu, do ktorej sa následne môžu vkladať jednotlivé elektrické komponenty, ktoré knižnica poskytuje, príklad obvodu môžete vidieť na Obrázok 8a Pri vytváraní elektrických komponentov sa nastavuje ich názov, hodnoty elektrických veličín príslušné komponentu, uzly napojenia komponentu a prípadne iné parametre.

```
// Build the circuit
var ckt = new Circuit(
    new VoltageSource("V1", "in", "0", 1.0),
    new Resistor("R1", "in", "out", 1.0e4),
    new Resistor("R2", "out", "0", 2.0e4)
    );
```

Obrázok 8a - Vytvorenie obvodu s tromi el. komponentami v prostredí SpiceSharp

SpiceSharp rieši spájanie súčiastok v obvode cez uzly (nodes), ktoré má každá súčiastka a predstavujú názov uzla, s ktorým je táto súčiastka spojená. Ak majú napríklad komponenty nastavený uzol s rovnakým názvom, tak to znamená, že sú prepojené.

Tranzitná simulácia

Po vytvorení obvodu môžu byť spustené simulácie pre striedavý a jednosmerný prúd, ktoré simulujú obvod v jednom časovom okamihu. Pre simulovanie obvodu v čase poskytuje SpiceSharp tranzitné simulácie, ktoré po určenej veľkosti časového kroku dokážu simulovať zmeny v čase pre elektrické komponenty.

Zmena parametrov

Parametre súčiastok sa dajú meniť aj počas tranzitnej simulácie, ktorá má funkcie vykonávané v jednotlivých štádiách simulovania jedného časového kroku a ich volaním sa môžu pre tento krok napríklad v tomto prípade upraviť hodnoty súčiastky. Avšak nie sú odporúčané veľmi drastické zmeny parametrov počas simulácie, kvôli možným ťažkostiam pri hľadaní konvergencie obvodu alebo jeho nenájdenia. Je odporúčané dobre poznať tok simulovania pre správnu zmenu parametrov a taktiež štruktúru a použitie parametrov pri simulačných výpočtoch, ktoré treba podľa potreby v krokoch simulácie taktiež aktualizovať.

Vlastná súčiastka

Je možné doplniť SpiceSharp o vlastné komponenty, avšak je potrebné porozumieť ako simulátor počíta napätie v jednotlivých uzloch obvodu, čo je realizované modifikovanou uzlovou analýzou.

SpiceSharpParser

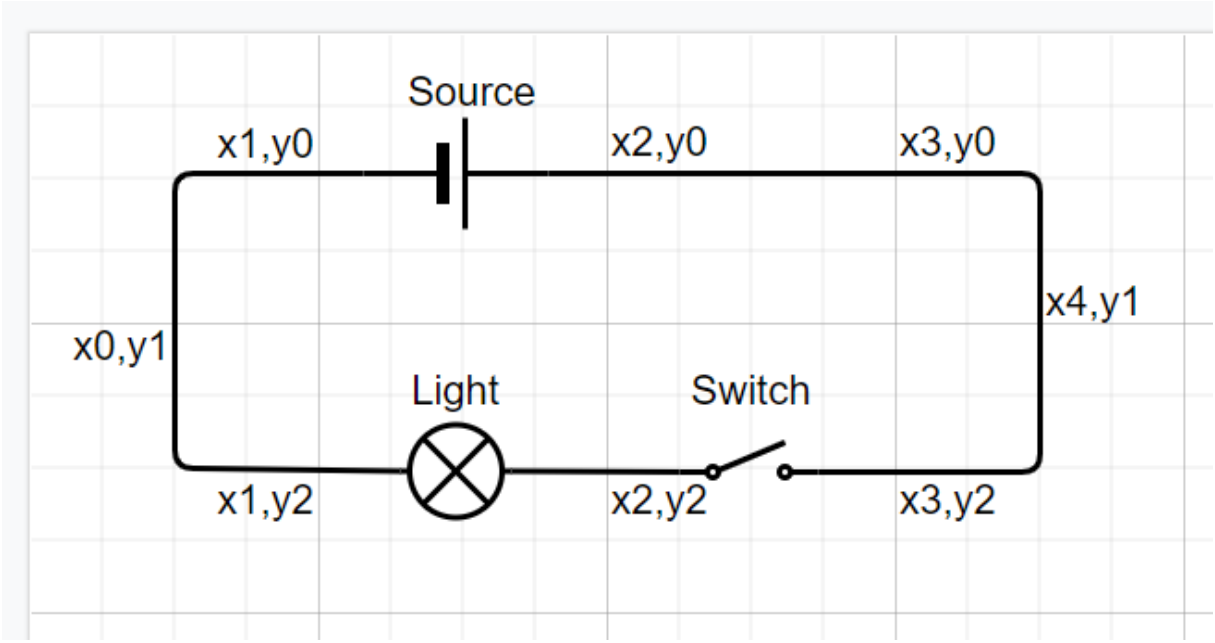
SpiceSharParser je knižnica vyvíjaná popri knižnici SpiceSharp a slúži na parsovanie SPICE netlist zápisu obvodov a ich prevod do SpiceSharp knižnice. Podporovaných je väčšina zápisov na základe PSpice netlist formátu. Na Obrázok 8b je príklad vytvorenia reťazca, ktorý predstavuje SPICE netlist zápis obvodu. V obvode je použitá dióda, ktorej model je v reťazci zapísaný vo formáte SPICE modelovej špeicfikácie.

```
var netlist = string.Join(Environment.NewLine,
    "Diode circuit",
    "D1 OUT 0 1N914",
    "V1 OUT 0 0",
    ".model 1N914 D(Is=2.52e-9 Rs=0.568 N=1.752 Cjo=4e-12 M=0.4 tt=20e-9)",
    ".DC V1 -1 1 10e-3",
    ".SAVE i(V1)",
    ".END");
```

Obrázok 8b - Vytvorenie SPICE netlist reťazca s dvoma el. komponentami pre SpiceSharpParser

5.4.2 Návrh

Keďže v *SpiceSharp*-e sa súčiastky pripájajú jednoducho k uzlom označeným ako “nodes”, ktoré sa označujú premennou typu *String*, pre mriežku, s ktorou budeme pracovať, si môžeme vytvoriť maticu alebo dvojrozmerné pole *Stringov*. V tejto matici budú uložené názvy pre jednotlivé uzly v obvode, kde sa dotýkajú dva bloky na mriežke. Na Obrázok 9 sú tieto názvy uzlov príkladovo označené (napríklad “x1,y0”).



Obrázok 9 - Jednoduchý elektrický obvod pri blokovom riešení

Na začiatku sa vytvorí v *SpiceSharp*-e nový objekt obvodu, do ktorého budú postupne pri akcii vloženia alebo odobratia bloku vkladané alebo odobraté komponenty elektrického obvodu, ktoré dané bloky predstavujú.

Pri vložení nového bloku na určené miesto v mriežke sa podľa typu el. komponentu, umiestnenia a rotácie bloku zistia uzly pre daný komponent. Ďalej sa už len tento komponent vytvorí s napojením na správne uzly a poskytnutými parametrami komponentu uložených v objekte bloku. Po vytvorení sa komponent pridá do obvodu.

Bloky predstavujúce križovatku (iba spájajú všetky susedné bloky) vytvoria nový uzol, s ktorým sa pomocou rezistorov pripoja všetky 4 uzly bloku. Podobne sa postupuje pri kríženiach typu "T".

V objekte bloku sa udržuje typ el. komponentu, jeho parametre(alebo pred vytvorený komponent) a napájacie uzly, ktoré môžu byť kódované *integer* polom, pričom budú použité hodnoty 0 a 1 pre určenie tých strán bloku, na ktorých je napájací uzol. Napríklad pre rezistor horizontálnej polohe by bolo takéto pole (0,1,0,1), pričom prvá hodnota by predstavovala spodnú stranu bloku a nasledujúce hodnoty ďalšie strany v smere hodinových ručičiek. Takéto kódovanie sa môže podľa potreby a implementácie upraviť.

Pri odobratí bloku z mriežky a teda z elektrického obvodu by sa podľa konkrétneho názvu komponentu tento komponent vyhľadal v obvode a z neho následne vymazal.

Pre súčiastky, ktoré budú mať premenlivé parametre (napríklad potenciometer) budú vytvorené funkcie zasahujúce do príslušných parametrov prebiehajúcej simulácie. Navrhuje sa použitie samostatného vlákna pre simuláciu obvodu, ku ktorému sa bude pristupovať z hlavného vlákna Unity. Tranzitná simulácia nepodporuje režim simulácie v nekonečnom časovom intervale, preto je najlepšie použiť čo najväčšiu hodnotu pre jej trvanie a v prípade potreby môžeme aktuálny stav simulácie uložiť a použiť ako začiatok novej simulácie. Simulácia sa dá zastaviť aj predčasne, takže jej ukončenie bude jednoduché.

Keďže tvorba vlastnej súčiastky nie je ľahký proces (najmä zložitejších), vlastné súčiastky, ktoré sú jednoduché sa môžu implementovať na strane Unity skriptov. Ak budú ale potrebné zložitejšie

súčiastky, podľa štruktúry a priebehu výpočtov v SpiceSharp sa môžu takéto súčiastky pridať do knižnice SpiceSharp.

Pre tvorbu obvodov sa môže použiť aj knižnica SpiceSharpParser. Pri tomto spôsobe by sa po vyskladaní obvodu a pri spustení simulácie zostavil reťazec SPICE netlist, ktorý by sa touto knižnicou previedol na SpiceSharp obvod. Týmto spôsobom môžeme uľahčiť pridávanie nových SPICE modelov pre súčiastky ako je dióda alebo tranzistor. Takýto SPICE model by stačilo vložiť do vytváraného netlist reťazca a priradiť súčiastky k týmto modelom. Namiesto skladania obvodu v prostredí SpiceSharp je potrebné prejsť blokmi v gridboarde a spísať výsledný reťazec, ktorý bude vstupom pre SpiceSharpParser.

5.5 Vyhodnotenie návrhov a riešení

Autori podkapitoly: Erik Paľa, Viktor Beňo

Vzhľadom na vysokú zložitosť riešenia tohto projektu pri použití breadboard-u sme sa rozhodli uprednostniť blokové riešenie, ktoré je prehľadnejšie a jednoduchšie na implementáciu.

Cieľom tohto blokového riešenia bude vytvoriť interaktívne prostredie, kde jednotlivé komponenty budú reprezentované 3D blokmi, ktoré bude možné uložiť na mriežku a tým zostrojiť elektrický obvod.

Jednotlivé komponenty budú reprezentované rôznymi typmi blokov. Napríklad štandardné súčiastky ako rezistor budú reprezentované ako kocka, na ktorej bude symbol rezistora. Aktívne súčiastky ako napríklad spínač alebo dióda budú reprezentované ich 3D modelmi.

Výhodou tohto riešenia je jeho prehľadnosť pre používateľa, ktorý nemusí pracne pripájať jednotlivé nožičky súčiastok do breadboard-u, ale stačí mu otočiť súčiastku ako potrebuje a umiestniť ju do mriežky, ktorá automaticky vytvorí uzly na miestach kde sa bloky spájajú.

Pre tvorbu obvodov je výhodnejšie použitie knižnice SpiceSharpParser. Poskladaný obvod by sa tak dal aj textovo zobraziť pre kontrolu zápisu a bol by poskytnutý aj jednoduchý spôsob pridávania nových SPICE modelov súčiastok pre špecifikovanie elektrických vlastností konkrétnych, reálnych súčiastok.

Obvod bude simulovaný plynulou tranzitnou simuláciou, ktorá bude bežať v samostatnom vlákne a bude podporovať zmenu parametrov počas behu simulácie. Jednoduchšie súčiastky budú pripravené pomocou skriptov na strane Unity a pri potrebe zložitejších súčiastok, tieto budú vytvorené a pridané do knižnice SpiceSharp.

5.6 Synchronizačné frameworky

Autori podkapitoly: Erik Paľa, Viktor Beňo

Keďže náš projekt ma byť vytvorený ako kolaboratívne prostredie, je potrebné zvoliť si adekvátny framework, ktorý nám poskytne potrebnú funkcionalitu vzhľadom na vývojové prostredie Unity, ako aj integrácia virtuálnej reality v našom projekte. Z toho dôvodu sme analyzovali niekoľko frameworkov a následne ich vyhodnotili.

5.6.1 Analýza

Mirror

Mirror predstavuje bezplatné, open source riešenie pre podporu synchronizácie používateľov. Tento framework je založený na modeli klient - server. Taktiež disponuje offline LAN podporou. Nepodporuje matchmaking pre používateľov, čo pre nás predstavuje veľkú nevýhodu. Výhodou však je že podporuje rôzne platformy. Nepodporuje však peer-to-peer spojenie.

Photon Bolt

Photon bolt je sieťové riešenie s bohatými funkcionalitami, ktoré bolo inšpirované sieťovými riešeniami viacerých hier. Umožňuje ľahko vytvoriť autoritatívnu hernú logiku so zabudovanými funkcionalitami, akými sú napríklad hitbox rewinding, lag compensation alebo client side prediction. Je založený na modeli klient - server a podporuje 4 hlavné architektúry, ktorými sú dedikované servery, klientom hostované servery, peer to peer a LAN/WLAN. Cena tohto framework-u sa odvíja od počtu aktuálnych používateľov. Môže byť použitý bezplatne alebo si za neho môže používateľ mesačne platiť. Zdrojový kód tohto framework-u je neprístupný. Podporuje offline mód, matchmaking a peer-to-peer spojenie. Offline LAN podpora a podpora priameho pripojenia pomocou IP sú platené funkcie. Rovnako ako predchádzajúci framework, aj tento je multi platformový.

Photon Unity Networking v2

Photon Unity Networking V2 je sieťové riešenie, ktoré sa najčastejšie využíva v hrách v ktorých sa nachádza málo hráčov a tieto hry trvajú krátko. Vývoj multiplayeru v tomto frameworku je jednoduchý, čo je jednou z hlavných výhod tohto frameworku. Jeden z hráčov sa vždy tvári ako host a ostatní hráči sú napojený cez relay servery. Používanie tohto frameworku je zadarmo. Jeho zdrojový kód je na vyššej úrovni voľne dostupný. Je postavený na modeli klient-klient. Tento framework podporuje offline mód, nedisponuje však offline LAN podporou. Taktiež podporuje matchmaking. Nepodporuje však peer-to-peer spojenia, ani priame pripojenie prostredníctvom IP adresy.

MLAPI

MLAPI je framework, ktorý obsahuje mnoho vlastností a mnoho ďalších je ešte vo vývoji. Komponenty tohto frameworku sú veľmi podobné frameworkom UNET a Mirror. vývoj tohto frameworku zabezpečuje jeden človek. Tento framework sa stále považuje za jeden z tých menej populárnejších, aj pre menší počet projektov, minimalistickú dokumentáciu a málo návodov. Vzhľadom na množstvo vlastností, ktorými tento framework disponuje je možné, že sa stane voľbou pre mnoho vývojárov. Využitie tohto framework-u je bezplatné a jeho kód je voľne dostupný. Je založený na modeli klient-server. Disponuje offline LAN podporou, podporou offline módu a technológiou multicast pre objavovanie LAN hier. Neumožňuje však matchmaking. V rámci konektivity umožňuje peer-to-peer spojenie a pripojenie cez IP adresu.

SmartFoxServer 2X

SmartFoxServer je multi-platformový client/server framework, ktorý je v porovnaní s ostatnými na nižšej úrovni abstrakcie so zameraním na vysokú výkonnosť, vhodnú pre masívne multiplayer online hry. Framework poskytuje veľkú voľnosť a granularitu pri návrhu sieťovej architektúry, ako aj samostatnej synchronizácie objektov.

Výhodou frameworku je, že podporuje veľké množstvo platforiem (Adobe Flash/Flex/Air, Unity, HTML5, iOS, Universal Windows Platform, Android, Java, C++ a iné), čiže v tomto ohľade umožňuje jednoduchú rozšíriteľnosť. Ďalšou výhodou je, že framework je dobre dokumentovaný, s množstvom príkladov aj vo formáte videí pre rôzne platformy, vrátane Unity. Tento má podporu pre offline LAN a taktiež offline mód. Ďalej disponuje aj technológiou multicast pre vyhľadávanie LAN hier a umožňuje priame pripojenie cez IP adresu. Neumožňuje však peer-to-peer spojenia.

Súčasťou tohto frameworku sú aj rôzne pokročilejšie funkcionality, ako je vzdialená administrácia serverov, cloud-hosting, či šifrovanie prenosového protokolu. Pre študentov a malé tímy je možné získať licenciu zadarmo, s obmedzením na 100 súčasných užívateľov. Hoci sú tieto funkcionality vo všeobecnosti žiadúce, a umožňujú veľmi slobodný vývoj, pre náš projekt sú často zbytočné. Vysoká úroveň výkonnosti pre nás nie je dôležitá, pretože budeme pracovať s malou scénou a obmedzeným počtom užívateľov. Nízka úroveň abstrakcie je pre nás nevýhodná, pretože vyžaduje viac developerského času a poskytuje menej funkcionalít pripravených na použitie. Jemná granularita a vysoká konfigurovateľnosť by v našej fáze vývoja mohli vytvoriť zbytočné trenie.

5.6.2 Porovnanie frameworkov v záujmových kategóriách

V Tabuľka 1 uvádzame porovnanie analyzovaných synchronizačných frameworkov vzhľadom na podstatné vlastnosti pre náš projekt.

	Mirror	**PUN 2**	**Bolt**	**MLAPI**	**SmartFoxServer 2X**
Sieťový model	Klient/Server	Klient/Klient	Klient/Server	Klient/Server	Klient / Authoritatívny Server
Integrácia s Unity	Mirror komponenty	wizard PhotonView components	Bolt Wizard Bolt komponenty	MLAPI components	DLL pokytuúca API
Podpora platforiem	Desktop, Web, Mobile, Console	Desktop, Web, Mobile, Console	Desktop, Mobile, Console	Desktop, Web, Mobile, Console	Desktop, Web, Mobile, Console
Hosting	Dedikovanýserver, klient, lokálna LAN / Wifi	Photon Cloud alebo Photon Server	Dedikovaný server, klient, P2P, lokálna LAN / Wifi	Dedikovaný server, lokálna LAN / Wifi	Dedikovaný server
LAN support	Áno	Nie	Za poplatok	Áno	Áno
Synchronizácia premenných	Syncvars, RPCs, Events	RPC, Events, OnSerializePhotonView	Bolt Wizard, Commands	NetworkedVar, Messaging System	Vstavané
Synchronizácia animácií	Áno	Áno	Áno	Nie	Nie
Synchronizovaný Timestamp	Áno	Áno	Nie	Áno	Nie
Serializácia vlastných typov	Áno	Áno	Áno	Áno	Áno
Manažment inštancovania sieťových objektov	Áno	Áno	Áno	Áno	Nie

Tabuľka 1 – Porovnanie synchronizačných frameworkov v záujmových kategóriách

5.6.3 Zhodnotenie

Analýza dostupných frameworkov pre prototypovanie multiplayer funkcionality v našom projekte nás doviedla k záveru, že pre naše potreby bude najvhodnejšie použiť framework photon bolt. Pre tento framework sme sa rozhodli kvôli jeho rozsiahlej funkcionalite a jednoduchej integrácii v prostredí Unity. Napriek tomu, že je tento framework platený, poskytuje aj bezplatnú funkcionalitu, ktorá nám bude na realizáciu projektu stačiť.

5.7 ANALÝZA MOŽNOSTÍ PRE SIMULÁCIU VOZIDLA

Autor kapitoly: Erik Paľa

5.7.1 Analýza

Karting Microgame

Tento balík v sebe spája viacero assetov, prostredníctvom ktorých je možné jednoducho vytvárať a simulovať jednoduché prostredie, v ktorom je umožnené jazdiť na motokárach.

Požiadavky:

- Najnovšia verzia unity hub
- Unity Editor 2020 LTS
- WebGL Build Support modul
- Share WebGL Game package

Tento balík je praktický aj z dôvodu, že obsahuje návody, ako si vytvoriť vlastnú simuláciu motokárovej trate a jazdy. Taktiež slúžil ako základ už pre viacero iných projektov, čo naznačuje, že je jednoduché s ním pracovať a prispôsobiť si ho pre vlastné potreby.

Karting Game

Tento projekt je vybudovaný na základe vyššie uvedeného balíka "Karting Microgame". Pri tomto prevedení je možné sledovať počet okruhov a časy pre okruhy. Vzhľadom na povahu nášho projektu by tieto funkcie mohli byť považované za zbytočné. Výhodou je, že tento projekt je low-poly. Nevýhodou je, že toto prevedenie nepodporuje online prepojenie, čo je vzhľadom na povahu nášho projektu problematické.

Karting

V tomto prípade sa jedná o tutoriál. ktorý krok po kroku vysvetľuje, ako vytvoriť jednoduchú simuláciu vozidiel na trati. Jedná sa o dvojdielny tutoriál, kde v prvom kroku je vytvorená samotná simulácia a v druhom kroku je k nej implementovaný multiplayer. Autori poskytujú na stránke tutoriálu aj SDK potrebné pre vytvorenie simulácie. To, že celú simuláciu je potrebné od základov vytvoriť v našom projekte však predstavuje veľkú nevýhodu, vzhľadom na to, že to nie je primárne zameranie nášho projektu a práca na tejto simulácii by konzumovala čas, potrebný pre implementáciu primárnych funkcionalít nášho projektu.

Kart Game

Tento projekt je ďalším z projektov, ktoré sú vybudované na "karting microgame". Vzhľadom na zámer využitia simulácie motokár v našom projekte však tento projekt disponuje veľkým množstvom funkcionalít, ktoré sú pre nás zbytočné.

Unity-Vehicle-Physics

Zámerom tohto projektu je implementácia simulácie fyzikálnych vlastností pre vozidlo. Pre simuláciu fyziky v tomto projekte je využitá funkcionalita prostredia Unity, ktorá umožňuje simuláciu fyziky. V tomto projekte je aplikovaná na jednoduché vozidlo a demonštruje reálne správanie vozidla na nerovnom teréne. Zároveň slúži ako ukážka simulácie správania tlmičov na kolesách vozidla.

5.7.2 Zhodnotenie

Po preskúmaní možností, ktoré by nám umožnili simuláciu motokáry v našom projekte sme sa rozhodli zvoliť karting microgame. Tento projekt disponuje všetkou funkcionalitou, ktorá je pre náš projekt potrebná. Ďalším dôvodom, prečo sme zvolili karting microgame je jeho jednoduchá aplikácia a modifikovateľnosť pre potreby nášho projektu

6 PROTOTYPY

Do konca 3. šprintu sme vytvorili náš prvý prototyp projektu zameraný na ovládanie a interakciu. Po dokončení 5. a teda posledného šprintu zimného semestra sme vytvorili druhý prototyp zameraný na podporu virtuálnej reality.

6.1 PROTOTYP MODELOV SÚČIASTOK

Autori podkapitoly: Ľubomír Kurčák, Patrik Tománek, Viktor Beňo

V projekte sú najdôležitejšie objekty jednotlivé súčiastky, z ktorých budú používatelia skladať výsledný obvod. Pri grafických požiadavkách na objekty identifikujeme niekoľko kompromisov. Modely musia byť dostatočne odlíšiteľné a naznačovať svoju funkciu, pričom zároveň musia byť zoskupené z vhodného počtu plôch aby nevytvárali príliš vysokú záťaž na grafickú kartu a tým pádom nespomaľovali rýchlosť obnovovania. Do úvahy sme vzali aj možnosť rozšíriteľnosti súčiastok užívateľom.

6.1.1 Analýza

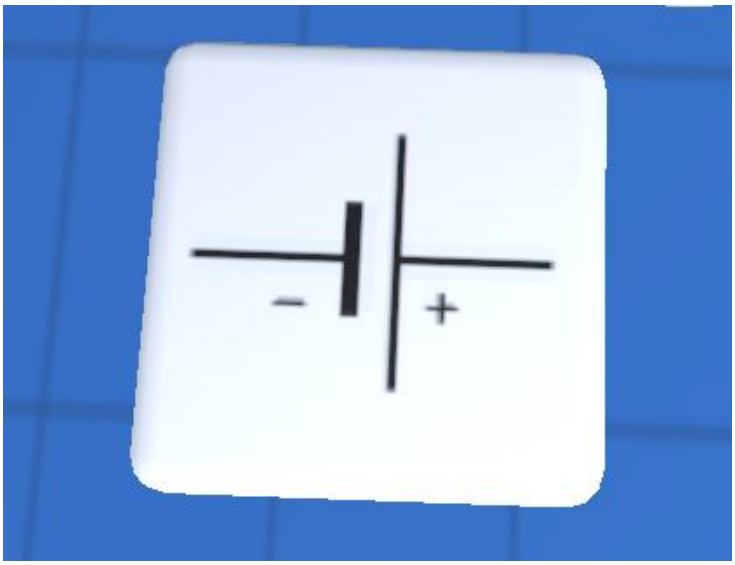
Základná požiadavka na modely súčiastok je, že musia zapadať do mriežky. Na základe modelov by mala byť rozpoznateľná ich funkcia, stav a tiež zapojenie, či kontakt so susediacimi súčiastkami. Ďalšie uvažované kritérium bola jednoduchosť rozšíriteľnosti sady súčiastok používateľmi. Na začiatku bol vykonaný prieskum existujúcich modelov, avšak nakoniec sme sa po uvážení rozhodli, že vytvorenie vlastných modelov nám umožní lepšie realizovať našu predstavu produktu.

6.1.2 Návrh

Uvažovali sme o troch hlavných typoch modelov: schematické modely, reliéfové modely a 3D modely. Každý prístup má iné výhody a stoja na rôznych pozíciách v rámci uvažovaných požiadaviek.

Schematické modely

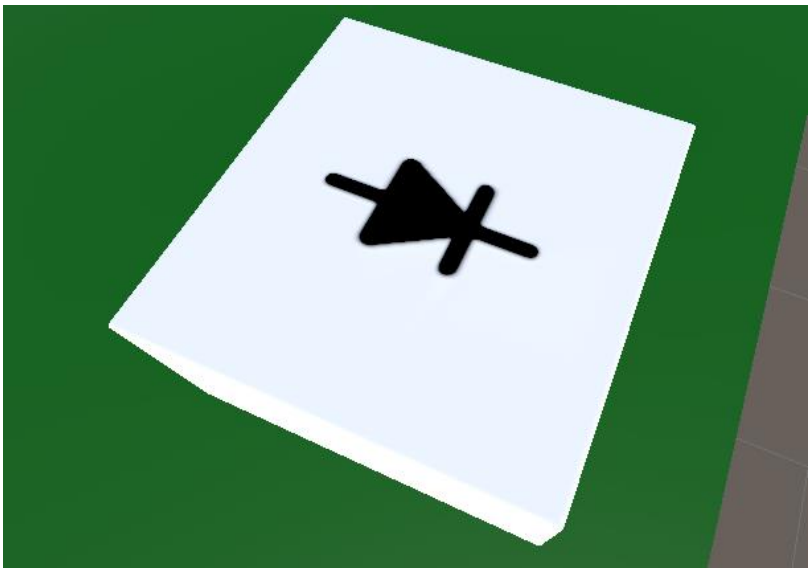
Schematické modely sú veľmi jednoduché, keďže všetky súčiastky používajú rovnaký model, pričom sa mení iba aplikovaná textúra. Tento typ modelov je vhodný kvôli jednoduchosti jeho implementácie. Tento typ modelov je takisto najjednoduchší na rozšírenie používateľom, kde používateľovi stačí okrem špecifikovania logiky určiť obrázok, ktorý bude na súčiastke zobrazený. Výhodou schematických modelov je aj veľmi nízka náročnosť na výkon, a teda nebude spôsobovať nízku obnovovaciu frekvenciu aj na pomalších zariadeniach. Nevýhodou tohto typu modelu je, že rasterové obrázky majú obmedzenú kvalitu, a pri blízkom pohľade bude možné vidieť pixelizáciu obrázkov schém. Na Obrázok 10 môžeme vidieť ukážku schematického spracovania.



Obrázok 10 - Model súčiastky batérie v schematickom spracovaní

Reliéfne modely

Reliéfne modely sú podobné schematickým modelov v tom, že tiež znázorňujú schematickú značku súčiastky. Rozdiel je v tom, že reliéfny model vystupuje z dvojrozmernej plochy do výšky. Dôsledkom toho je, že modely sú rozpoznateľné aj bez potreby textúry. Výhodou tohto spôsobu oproti čisto textúrovému prístupu schematických modelov je, že schematická značka má nekonečné rozlíšenie, keďže zakódovaním schémy do modelu je efektívne vektorizujeme. Nevýhodou je však nutnosť vytvorenia vlastného modelu pre každú súčiastku, čo znižuje rozšíriteľnosť tohto variantu. Príklad modelu súčiastky v reliéfovom spracovaní môžeme vidieť na Obrázok 11.



Obrázok 11 - Model súčiastky diódy v reliéfovom spracovaní

3D modely

Posledný uvažovaný typ modelov sú modely s trojrozmernými modelmi súčiastok. Výhodou týchto modelov je, že sú pútavejšie a ich funkcia je intuitívnejšia ako pri schematických modeloch. Pomocou trojrozmerných modelov je možné jednoduchšie a jednoznačnejšie indikovať stav súčiastok, ako sú napríklad stavy pre zasvietenú a vypnutú žiarovka, alebo zapnutý a vypnutý spínač. Nevýhodou je však nemožnosť rozšírenia 3D súčiastok, pretože je takmer nemožné očakávať od používateľov vytvorenie vlastných modelov kompatibilných s našou aplikáciou, a animácia stavu týchto súčiastok si vyžaduje explicitnú implementáciu v rámci aplikácie. Príklad modelu súčiastky v trojrozmernom spracovaní môžeme vidieť na Obrázok 12.

Obrázok 12 - Model súčiastky tranzistoru v trojrozmernom spracovaní

6.1.3 Implementácia

Modely boli tvorené použitím open source programu Blender. Modely exportované z Blenderu boli pridané do prostredia Unity, kde z nich boli vytvorené inštancovateľné objekty. Modely boli vyhotovené vo všetkých uvažovaných spracovaniach. Modely boli vytvorené spôsobom, aby boli čo

najviac znovu-použiteľné s ohľadom na možnosť rozšíriteľnosti používateľom. Štandardizované sú základný tvar bloku, a konektory k okolitým blokom, na základe typu súčiastky. Momentálne sú v projekte použité kombinované schematické a 3D modely. Možnosť schematických modelov zachovávame pre rozšíriteľnosť, avšak snažíme sa vytvoriť čo najväčší počet štandardných súčiastok v 3D verziách.

6.2 Prototyp pre ovládanie a interakciu

Autori podkapitoly: Ľubomír Kurčák, Patrik Tománek

Cieľom prvého prototypu bolo umožniť hráčovi pohybovať sa v 3D priestore, ako aj dovoliť hráčovi interagovať s elektrickými súčiastkami a grid boardom v tomto prostredí.

6.2.1 Analýza

Pohybovanie sa v 3D priestore ako aj interakcia s elektrickými súčiastkami a grid boardom sú kľúčové prípady použitia pre náš projekt. Počas našej analýzy bolo potrebné určiť si akým prístupom túto funkcionalitu implementujeme do nášho riešenia. Rozhodli sme sa tento prvý prototyp implementovať ako desktopové riešenie pre podporu klávesnice a myši.

Súčasťou finálneho prototypu sú taktiež modely pre jednotlivé elektrické súčiastky, ktoré sme si spoločne vybrali pre náš projekt. Tieto modely vytvoril Ľubomír Kurčák v grafickom prostredí Blender. Hlavný element rozhodovania pri výbere súčiastok, ktoré budú súčasťou nášho riešenia bol ten, že všetky súčiastky museli byť podporované knižnicou *SpiceSharp*, ktorú budeme používať na simuláciu správania elektrických obvodov v našom projekte. Po dokončení tvorby zoznamu podporovaných súčiastok sa mohlo prejsť na ich tvorbu.

Čo sa týka tvorby prototypu v Unity, rozhodli sme sa pre vytvorenie ukážkovej scény pre prvý prototyp, ktorá obsahovala prostredie, v ktorom sa hráč pohybuje. Takisto sa v tomto prostredí nachádzal grid board ako aj jednotlivé elektrické súčiastky, s ktorými mohol hráč interagovať. Funkcionalita jak pohybovania tak aj interakcie bola tvorená pre klávesnicu aj myš takým spôsobom, aby sa dané riešenie mohlo aplikovať aj na následnú integráciu VR do nášho projektu.

6.2.2 Návrh

Pri tvorbe prototypu v Unity, sme sa rozhodovali, akým smerom sa vyberieme. Nakoniec sme sa dohodli na finálnom návrhu.

Hráč bude reprezentovaný ako *GameObject* spolu s kamerou ako aj ďalším *GameObjectom*, ktorý bude reprezentovať jeho ruky. Hráč sa bude môcť pohybovať do všetkých smerom pomocou klávesnice a myšou bude ovládať smer akým sa hráč pozerá. Ruky hráča budú slúžiť ako kontajner pre všetky elektrické súčiastky, ktoré si vezme do rúk. Spôsob detekcie, akým hráč bude zdvíhať jednotlivé súčiastky, ako aj ich ukladanie do grid boardu bude zabezpečený pomocou *Raycast* funkcionality. Táto funkcia slúži na interakciu hráča s objektom, na ktorý sa hráč momentálne pozerá, pretože tento *Raycast* bude vysielaný zo stredu obrazovky hráča, a teda stredu danej kamery, ktorá je hráčovou súčasťou. *GameObject* hráč bude teda vo finále obsahovať 3 hlavné komponenty. Jeden komponent bude slúžiť ako kontrolér pre ovládanie, ďalší pre interakciu a posledný komponent bude kontrolérom pre hráčove ruky.

Elektrické súčiastky budú taktiež reprezentované ako *GameObject*. V tejto fáze prototypu sme neimplementovali logiku jednotlivých súčiastok, takže tieto súčiastky zatiaľ zohrávajú úlohu modelu, s ktorým hráč môže interagovať a vkladať ho do grid boardu. Tieto súčiastky budú súčasťou rodičovského kontajneru pokiaľ sa ich hráč nerozhodne zdvihnúť. Keďže hrá ma pôsobiť reálne, je

potrebné aby na tieto súčiastky bola naviazaná fyzika, ktorá bude zabezpečená *Rigidbody* komponentom, ktorý je súčasťou Unity engine. Ďalej každá súčiastka bude obsahovať komponent, ktorý hlavne bude mať za úlohu zisťovania stavu, v akom sa súčiastka nachádza. Stavy môžu byť rôzne, ako napríklad súčiastka je v rukách hráča, súčiastka je na zemi alebo súčiastka je uložená v grid boarde.

Grid board bude reprezentovaný ako priestor, do ktorého môže hráč vkladať elektrické súčiastky. Grid board bude obsahovať niekoľko priestorov určených pre spomínané súčiastky. Na každom z týchto miest bude naviazaný komponent, ktorý bude slúžiť ako kontrolér pre vloženie súčiastky do tohto priestoru. Okrem iného tento komponent bude takisto obsahovať informáciu, či je tento priestor voľný alebo doňho už nejaká súčiastka bola vložená. Opäť súčasťou tohto prototypu nie je implementovaná logika reálneho grid boardu, takže jeden komponent pre každý priestor bol zatiaľ dostačujúci pre pointu tohto prototypu. Takisto každý úložný priestor musí obsahovať vlastný *collider*, ktorý bude kontrolovať, či sa nejaká elektrická súčiastka nachádza v jeho prostredí a ak áno, túto súčiastku si automaticky pripne do svojho priestoru. Tento prístup bude obzvlášť potrebný pri integrácií VR prototypu, aby hráč mohol elektrickú súčiastku jednoducho priložiť pri ľubovoľný voľný priestor a súčiastka sa doňho automaticky vloží.

Interakcia hráča s elektrickými súčiastkami ako aj grid boardom má niekoľko rôznych funkcií. Čo sa týka interakcie so súčiastkami, hráč bude môcť ľubovoľnú súčiastku zdvihnúť zo zeme a zobrať ju do rúk, následne ju bude môcť v rukách otáčať okolo jednej osi, tento smer bude definovať ako bude súčiastka vložená do gridu (vertikálne/horizontálne), hráč takisto môže súčiastku z rúk pustiť naspäť na zem. Aby hráč vedel, s ktorou súčiastkou práve interaguje, a teda, ktorú súčiastku po stlačení príslušného tlačidla zodvihne do rúk, daná súčiastka bude hráčovi zvýraznená. Pri interakcií s grid boardom je potrebné, aby hráč vedel, do ktorého priestoru bude súčiastka uložená po stlačení príslušného tlačidla. Rozhodli sme sa do implementovať prístup simulácie. Táto simulácia bude simulovať súčiastku, ktorú hráč bude držať v ruke do priestoru, na ktorý sa hráč díva. Simulácia súčiastky bude presnou kópiou stavu súčiastky, ktorú hráč drží v ruke. To znamená, že pokiaľ bude hráč túto súčiastku v ruke rotovať, bude sa rotovať aj daná simulácia v priestore na grid boarde.

6.2.3 Implementácia

Hierarchia scény predstavovala *Canvas*, *Room (priestor, v ktorom sa hráč nachádza), Player (hráč).*

Canvas v prvom prototype slúžil iba ako ukazovateľ kam sa hráč pozerá. Tento ukazovateľ predstavoval zelenú guličku v strede obrazovky.

Room objekt bol zložený z niekoľkých častí. Jednou bola podlaha, po ktorej sa hráč mohol pohybovať, druhá časť predstavovala grid board a posledná časť bol spomínaný kontajner pre všetky elektrické súčiastky. V tomto kontajneri sa nachádzali všetky súčiastky, ktoré neboli vložené do grid boardu, alebo ich hráč nedržal v ruke, to znamená, že pokiaľ nejakú zo súčiastok hráč pustil na zem, automaticky sa zaradilo do tohto kontajneru.

Player bol objekt, ktorý predstavoval daného hráča. Na tomto objekte bola naviazaná kamera, ktorá hráčovi renderovala obraz a takisto súčasťou tohto objektu boli ruky hráča ako aj model samotného hráča.

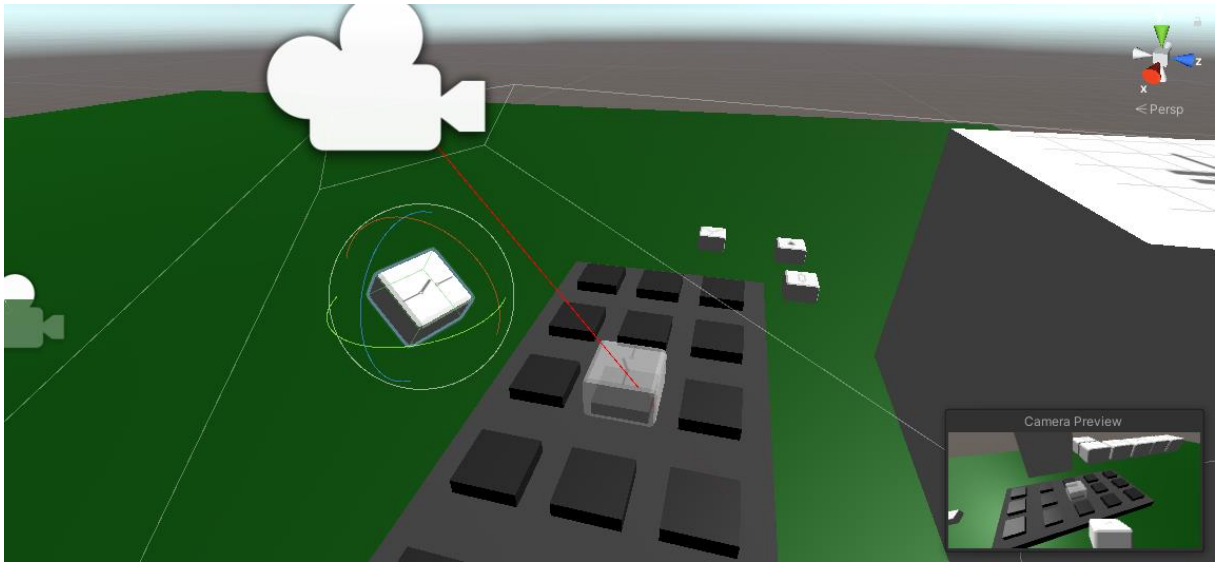
Následne vzniklo niekoľko komponentov, pričom každý komponent bol súčasťou príslušného *GameObject* objektu.

MovementController je komponent *Player* objektu, tento komponent je zodpovedný za pohybovanie sa hráča. Na základe hráčovho inputu tento kontrolér udáva objektu smer a rýchlosť,

akou sa bude pohybovať. Takisto je tu sledované, či hráč stlačil príslušne tlačidlo pre skok, v tom prípade *Player* objekt vyskočí v hre. Tento komponent takisto pracuje s Unity komponentom *CharacterController*, ktorý je takisto súčasťou *Player* objektu. *CharacterController* slúži na simuláciu pohybovania pomocou *Move(Vector3 motion)* funkcie. *Motion* je tvorený všetkými hráčovými inputmi. V prípade, že hráč skočil a teda je vo vzduchu, do tohto *motion* sa takisto počíta aj nami zadefinovaná gravitácia.

HandsController je ďalší komponent *Player* objektu, tento komponent slúži ako kontrolér hráčových rúk. Súčasťou tohto komponentu sú funkcie ako *pickupItem(GameObject item)*, ktorý zdvihne objekt, na ktorý sa hráč pozerá a vloží mu ho do rúk. Ďalej obsahuje funkciu *dropItem()*, ktorá pustí objekt, ktorý hráč práve drží v ruke na zem. Funkcie ako *GameObject getGrabbedItem()* a *removeItem()* primárne slúžia pre logiku naviazanú na grid boarde, kde tento grid board potrebuje zistiť aký objekt hráč drží v ruke a prípadne mu tento objekt z ruky odstrániť, čím sa priamo vloží do jedného z priestorov na grid boarde. *HandsController* takisto slúži na rotáciu príslušného objektu, ktorý hráč drží. Hráč môže objekt rotovať 2 smermi, ale vždy iba na 1 osi. Iné osi nie sú potrebné pretože táto rotácia hovorí o tom, akou stranou bude daná elektrická súčiastka vložená do grid boardu.

InteractionController je zatiaľ posledný komponent *Player* objektu. Tento komponent je zodpovedný za interakciu hráča s prostredím, v ktorom sa nachádza. Zaznamenáva hráčov input ako aj jeho zorné pole a reaguje naň. Hlavnou súčasťou komponentu je *UnityEngine.Physics.Raycast* funkcionalita, ktorá zabezpečuje zaznamenávanie akýchkoľvek kolízii hráča s objektami. Je to akýsi laser pointer, ktorý kontroluje, či tento laser preťal nejaký objekt. Ak je objekt preťatý, vráti nám tento objekt, aby sme s ním mohli ďalej pracovať. Na Obrázok 13 môžeme vidieť spomínaný *Raycast*, ktorý je vysielaný zo stredu kamery, a teda zorného poľa hráča. Keďže hráč drží v ruke elektrickú súčiastku a *Raycast* preťal jednu z voľných pozícií na grid boarde, táto súčiastka sa simuluje na danú pozíciu. V pravom dolnom rohu obrázku môžeme vidieť ako tento táto situácia vyzerá z pohľadu hráča.



Obrázok 13 - Ukážka Raycast funkcionality (červená čiara)

Raycast využívame na sledovanie hneď niekoľkých udalostí:

- Pokiaľ hráč stlačil ľavé tlačidlo na myši skontroluj či:
    - sa hráč pozerá na elektrickú súčiastku, v tom prípade ju hráč zodvihne, pokiaľ už nedrží v ruke inú súčiastku

    - sa hráč pozerá na voľné miesto na grid boarde, v tom prípade sa súčiastka uloží do daného miesta.
- Pokiaľ sa hráč pozerá na interaktívny objekt skontroluj či:
    - sa hráč pozerá na elektrickú súčiastku, pričom v ruke nedrží inú, v takom prípade zvýrazni túto súčiastku
    - sa hráč pozerá na voľné miesto na grid boarde, pričom v ruke drží elektrickú súčiastku, v takom prípade simuluj súčiastku na dané miesto

ItemController je komponent naviazaný na elektrické súčiastky. Komponent obsahuje informácie o danej súčiastke, ako napríklad či bola súčiastka vložená do grid boardu a ak áno, tento komponent si uloží *InsertController* komponent toho miesta, do ktorého bola vložená. Tento komponent takisto disponuje *private void OnTriggerEnter(Collider other)* funkcionalitou, ktorá je zodpovedná za sledovanie kolízií tohto objektu z ostatnými. Konkrétne sa sleduje, či je tento objekt v kolízii s voľným miestom na grid boarde a ak áno, tak sa do tohto miesta uloží. Súčasťou tohto komponentu je aj informácia o rodičovskom kontajnery, pretože každá súčiastka sa vždy nachádza v tomto kontajnery, výnimkou sú iba situácie, že táto súčiastka je uložená na grid boarde alebo ju hráč drží v ruke.

InsertController je posledný komponent 3. šprintu, ktorý má na starosti vkladanie elektrických súčiastok do voľných miest na grid boarde. Sú 2 typy, ako sa súčiastka môže dostať do grid boardu:

- Súčiastka bola do grid boardu vložená z rúk hráča po stlačení pravého tlačidla na myši
- Súčiastka zachytila kolíziu z voľným miestom na grid boarde a tým pádom sa do tohto miesta uložila automaticky

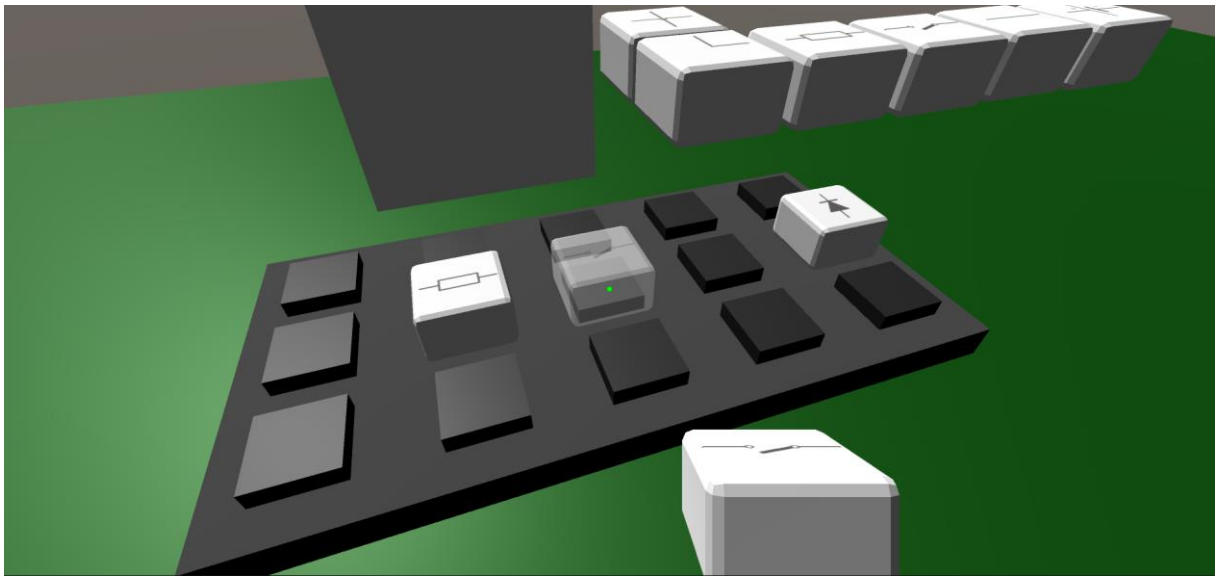
Následne pre oba prípady je použitá ďalšia metóda, ktorá zabezpečuje úspešne vloženie súčiastky do grid boardu. Súčasťou tejto metódy je takisto aj zaokrúhlenie *Z-osi* rotácie súčiastky k najbližšiemu násobku 90 stupňov, aby sa zachovalo správne usadenie súčiastky do grid boardu.

```
float itemCurrentRotationZ = Mathf.Round((insertedItem.transform.rotation.eulerAngles.z) / 90f) * 90f;
```

Poslednou časťou tohto komponentu je metóda, zodpovedná za vytiahnutie súčiastky z grid boardu, pokiaľ sa tak hráč rozhodol spraviť stlačením príslušného tlačidla.

6.2.4 Testovanie

Celé testovanie prvého prototypu prebiehalo v prostredí Unity, v ktorom naši testeri testovali všetky implementované funkcionality prototypu. Výsledok testovania nepredstavoval žiadne nájdené chyby v riešení a tým pádom testovanie prebehlo úspešne. Na Obrázok 14 môžeme vidieť finálnu podobu prvého prototypu.



Obrázok 14 - Ukážka prototypu pre ovládanie a interakciu

6.3 PROTOTYP PRE PODPORU VIRTUÁLNEJ REALITY

Autori podkapitoly: Ľubomír Kurčák, Patrik Tománek, Viktor Beňo

Cieľom druhého prototypu bola implementácia podpory pre virtuálnu realitu. Hráč sa teda mohol vďaka VR headsetu a ovládačmi pohybovať v 3D priestore ako aj interagovať s rôznymi objektami, a teda hlavne s elektrickými súčiastkami a grid boardom. Do tohto prototypu sme už takisto aj implementovali prvé demo so *SpiceSharp* knižnicou, ako aj kontrolu spojenia medzi jednotlivými elektrickými súčiastkami umiestnených v grid boarde.

6.3.1 Analýza

Podpora virtuálnej reality je takisto kľúčovým aspektom nášho projektu. Počas našej analýzy bolo potrebné nájsť vyhovujúci prístup, akým túto funkcionalitu implementujeme do nášho riešenia. Nakoniec sme sa rozhodli použiť *VR Interaction Framework* ako jadro funkcionalít virtuálnej reality.

Súčasťou tohto prototypu sú takisto aj nové modely pre ďalšie elektrické súčiastky, ktoré budeme v projekte používať. Takisto sa pre tieto súčiastky ako aj súčiastky, ktoré už boli vytvorené implementovala podpora ich interakcie vo virtuálnej realite. Tie modely opätovne vytvoril Ľubomír Kurčák. Ďalšia potrebná časť druhého prototypu bolo vytvorenie dema so *SpiceSharp* knižnicou. Vďaka tomu sme mohli začať simulovať už aj zložitejšie obvody v našom projekte. Toto demo vytvoril Viktor Beňo.

Čo sa týka tohto prototypu, rozhodli sme sa pre modifikovanie prvej ukážkovej scény, ktorá vznikla ako výstup z prvého prototypu. Do prostredia boli pridané nové elektrické súčiastky ako aj nové objekty, ako napríklad nový grid board. Druhý prototyp je primárne zameraný na podporu virtuálnej reality, to znamená, že sme zatiaľ neimplementovali novú funkcionalitu vytvorenú počas tohto prototypu pre desktopové riešenie, pričom sme ale celý kód tvorili agnostickým prístupom, a teda implementácia všetkých funkcionalít pre desktopové riešenie nebude predstavovať problém, len to momentálne nebolo naším cieľom.

6.3.2 Návrh

Pri tvorbe tohto prototypu sme sa museli najprv dohodnúť, aký framework použijeme, aby spĺňal všetky náležitosti potrebné pre náš projekt. Nakoniec sme sa dohodli na použitiu *VR Interaction Framework*.

VR Interaction Framework nám poskytuje širokú podporu rôznych funkcionalít pre virtuálnu realitu. Do nášho prostredia stačí keď vložíme *prefab* hráča vytvoreným týmto frameworkom a celá logika pohybu hráča nemusí byť samostatne implementovaná. Okrem iného tento framework ponúka aj množstvo iných objektov, ktoré sme takisto implementovali do našej novej scény. Jednalo sa o objekty ako napríklad stôl, aby sme vytvorili prvotný návrh laboratórneho prostredia. Takisto aj interakcia s jednotlivými objektami pomocou VR ovládačov je zabezpečená týmto frameworkom, pričom si ale konkrétnu funkcionalitu budeme musieť prispôsobiť pri práci s elektrickými súčiastkami ako aj ich vkladaním do grid boardu.

Prispôsobenie funkcionality *VR Interaction Frameworku* bude hlavne zahŕňať dodatočnú funkcionalitu v prípade, že hráč vloží elektrickú súčiastku do grid boardu. V takom prípade bude potrebné skontrolovať či sa v jeho okolí nachádza iná súčiastka, konkrétne či susedné súčiastky majú medzi sebou spojovníky, v takom prípade dôjde medzi nimi k spojeniu.

Spojenie medzi súčiastkami je dôležité pri transformácií elektrického obvodu na vstup pre *SpiceSharp* knižnicu. Každá elektrická súčiastka obsahuje niekoľko konektorov, tieto konektory sú následne napájané na iné konektory susedných súčiastok. Pokiaľ dôjde k spojeniu je potrebné vytvoriť identifikátor pre spojené konektory. Na základe tohto identifikátoru následne vie *SpiceSharp* knižnica spracovať daný obvod. Bližšie informácie sú uvedené v sekcií 3.4.

6.3.3 Implementácia

Do nášho prototypu sme integrovali *VR Interaction Framework* a následne sme vytvorili scénu za použitia objektov z tohto frameworku. Objekty sme pridávali do už vytvorenej scény z prvého prototypu.

Ako prvé sme do scény vložili *XR Rig Advanced* prefab z *VR Interaction Frameworku*, tento prefab zabezpečuje kompletnú logiku postavy, ktorá je ovládaná hráčom pomocou VR headsetu spolu s ovládačmi. Tento prefab takisto poskytuje možnosť zapnúť emulátor, v takom prípade sa dá testovať korektnosť nášho riešenia aj pomocou klávesnice a myši a nie je potrebné zapojenie VR. Vďaka tomu sa rapídne urýchľuje celý proces vývoja. Hráč sa teda pozerá cez VR headset na prostredie, v ktorom je umiestnený, pričom takisto aj vidí svoje ruky, ktoré ovláda pomocou VR ovládačov.

Následne sme pridali do scény nový grid board. *VR Interaction Framework* ponúka možnosť prichytávania objektov do určenej oblasti (snap zones). Túto funkcionalitu sme použili pre každé miesto v našom grid boarde, do ktorého hráč môže vkladať elektrické súčiastky. Túto funkcionalitu sme si ale museli následne aj prispôsobiť pre náš projekt. Jeden z problémov bol, že táto funkcionalita vkladá objekty do prostredia vždy v rovnakej rotácií, a teda *Vector3 (0,0,0)*. Tento problém bol ale vyriešený pridaním *Snap Zones Offset* komponentu pre každú elektrickú súčiastku. Tento komponent je zodpovedný za zmenu pozície alebo rotácia pre objekt, ktorí je vložený do nejakej snap zóny. Každá elektrická súčiastka má nastavenú *X rotation* offset na 270 stupňov, vďaka tomu bude každá súčiastka hlavnou častou hore. Ďalej je potrebné rotovať tento objekt tak, aby nikdy nebol nakrivo ale mohol byť iba otočený 4 rôznymi spôsobmi. Hráč si samozrejme môže daný objekt ľubovoľne rotovať v rukách, ale jakmile je súčiastka vložená do grid boardu, je potrebné ju rotovať do najbližšieho násobku 90.

Každá elektrická súčiastka takisto obsahuje niekoľko konektorov. Zatiaľ maximálny počet konektorov, ktoré môžu byť na súčiastku pridané sú 4: hore, vpravo, dole, vľavo. Konektory sú pre

každý elektrický komponent reprezentované ako *boolean*. Každý elektrický komponent ma na začiatku nastavené tieto konektory v *DefaultPins*, čo je *boolean array.* Následne každý komponent takisto obsahuje aj *boolean array CurrentPins*, ktoré hovorí, ako sú konektory rozmiestnené po umiestnení elektrickej súčiastky do grid boardu. V prípade, že sa súčiastka vloží do grid boardu s inou rotáciou ako je jej základná, zavolá sa funkcia *public void rotatePints(int count)*, kde *count* predstavuje počet, koľko krát sa pole *CurrentPins* shiftne do prava. Celá táto logika je obsiahnutá v *ComponentController* komponente, ktorý je súčasťou všetkých elektrických súčiastok.

Aby sa mohli medzi elektrickými súčiastkami kontrolovať spojenia, každá oblasť v grid boarde, do ktorej môže byť elektrická súčiastka vložená obsahuje 4 hodnoty pre každá stranu oblasti, takáto hodnota predstavuje *int*, kde 0 znamená, že súčiastka na danej strane nemá konektor, 1 znamená, že na nej konektor má ale nie je s ničím napojený a 2 znamená, že tam konektor je a zároveň je aj napojený na iný konektor. Zatiaľ sme ešte neimplementovali transformáciu nášho elektrického obvodu ako vstup do *SpiceSharp* knižnice. Keď ale začneme túto funkcionalitu implementovať do nášho projektu, jediné čo nám stačí bude premeniť *int* hodnotu 2 na referenciu tej súčiastky, na ktorú je daná súčiastka napojená. Už bude iba potrebné pre toto spojenie vytvoriť vlastný identifikátor a budeme schopný celý obvod pretvoriť ako vstup do *SpiceSharp* knižnice, ako už bolo vysvetlené v sekcií 3.4.

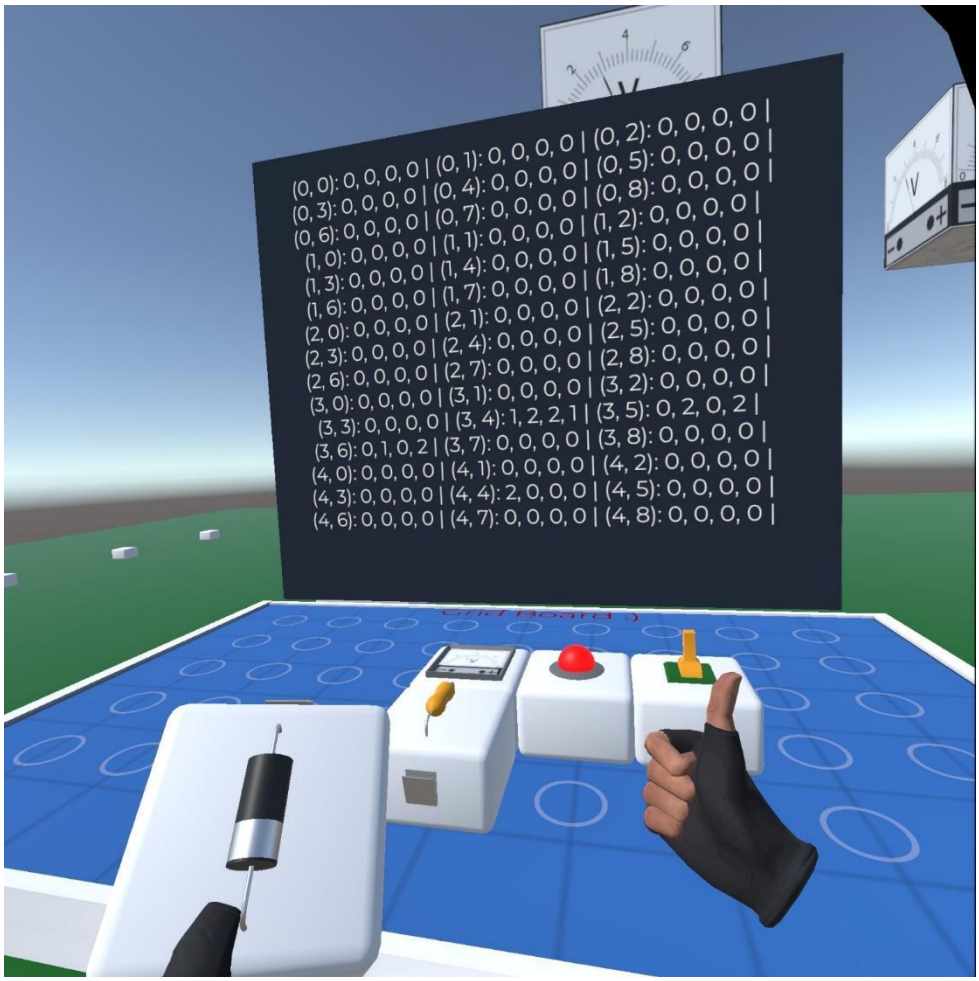
Čo sa týka samotného nového grid boardu, každá oblasť, do ktorej môže byť vložená elektrická súčiastka obsahuje komponent *gridboardPosition*, tento komponent obsahuje *Vector2 positionInGridboard,* ktorý hovorí o pozícií danej oblasti vzhľadom na celý grid board. Ďalej každá oblasť takisto obsahuje 4 *int* hodnoty *c_top, c_right, c_bottom, c_left*. Tieto hodnoty predstavujú konektory jednotlivých strán elektrickej súčiastky ako bolo spomenuté v odstavci vyššie. Samotný grid board obsahuje komponent *gridboardController,* ktorý hneď na po tom, ako sú všetky objekty v scéne inicializované zavolá metódu *setUpChilCoordinates(),* v ktorej sa pre každú jednu oblasť grid boardu nastaví *Vector2 positionInGridboard*, keďže *child* objekty gridboardu sú tvorenú oblasťami pre elektrické súčiastky. Ako ďalšiu metódu tento komponent obsahuje *checkForConnections(GameObject HeldItem, GameObject slot)*, ktorá po tom, ako je nejaká súčiastka vložená do grid boardu, kontroluje všetky možné spojenia tejto súčiastky s ostatnými v jej okolí. *GameObject HeldItem* predstavuje elektrickú súčiastku, ktorú hráč vložil do grid boardu a *GameObject slot* predstavuje oblasť, do ktorej ju hráč vložil. Následne sa z *GameObject HeldItem*, konkrétne z komponentu *ComponentController*, ktorý je súčasťou tohto *GameObject HeldItem*, zistí, kde všade má konektory. Podľa týchto konektorov sa skontrolujú susedné oblasti v gridboarde, či sa tam nachádza nejaká súčiastka, a ak áno, či ta súčiastka má susedný konektor, v takom prípade sa tieto 2 súčiastke spoja, a teda hodnota konektorov jednotlivých oblastí sa nastaví na 2. Posledná metóda *gridboardController* komponentu je *updateConnections(GameObject HeldItem, GameObject slot)*, jej funkcionalita je taká istá ako predošlá metóda len s tým rozdielom, že táto metóda je zavolaná v prípade, že hráč z grid boardu vytiahol nejakú súčiastku a preto je potrebné vyresetovať všetky konektory tejto súčiastky, ako aj aktualizovať konektory susedných súčiastok z 2 na 1. Vedľa gridboardu sa takisto nachádza aj tabuľa, ktorá vypíše všetky informácie o jednotlivých oblastiach gridboardu, konkrétne pozíciu oblasti v gridboardu a všetky konektory. Táto tabuľa je aktualizovaná zakaždým keď sa zavolá buď *checkForConnections*, alebo *updateConnections* metóda.

Poslednou časťou druhého prototypu bola implementácia dema so *SpiceSharp* knižnicou. Vytvorili sme 2 komponenty *CircuitBuilder* a *SpiceSharpController*. *SpiceSharpController* má za úlohu spustiť simuláciu nami vytvoreného obvodu. Táto simulácia sa spustí v prípade, že hráč stlačí v hre príslušné tlačidlo. Nami vytvorený obvod má bežať dovtedy, dokedy obvod nebude manuálne vypnutý, a preto je potrebné takúto simuláciu spustiť na vlastnom vlákne. Unity nepodporuje *multi-threading*, preto všetka funkcionalita, ktorá potrebuje využitie *Unity API* beží na hlavnom vlákne. Preto nie je

možné takúto simuláciu spustiť v rámci hlavného vlákna, pretože Unity bude spracovávať túto simuláciu až dokedy neskončí, počas čoho sa celá hra zasekne. Preto sme sa rozhodli pre všetky *SpiceSharp* obvody spúšťať tieto simulácie na samostatných vláknach. Tieto vlákna avšak slúžia čisto iba na *SpiceSharp* funkcionalitu. Keďže náš projekt takisto potrebuje možnosť meniť tieto bežiace simulácie za ich behu, komponent *CircuitBuilder* obsahuje metódu *public (string, double) changeResistance(double value)*, kde *value* predstavuje hodnotu, o ktorú sa zmení odpor našej simulácie. Táto funkcia je volaná opätovne po stlačení príslušného tlačidla v hre z hlavného vlákna *Unity API*, čo ale nepredstavuje žiaden problém na výkonnosť, pretože táto funkcia sa následne spracováva na našom *SpiceSharp* vlákne. Táto metóda takisto do hlavného vláka vráti *Tuple*, konkrétne *string status*, ktorý hovorí o stave simulácie a *double output* simulácie. Tento výstup predstavuje napätie v zdroji.

6.3.4 Testovanie

Celé testovanie prvého prototypu prebiehalo v prostredí Unity, tentokrát testovanie mohli uskutočniť iba naši *Unity developeri* Ľubomír Kurčák a Patrik Tománek, pretože mali k dispozícii VR headsety. Výsledok testovania nepredstavoval žiadne nájdené chyby v riešení a tým pádom testovanie prebehlo úspešne. Na Obrázok 15 môžeme vidieť finálnu podobu prvého prototypu.



Obrázok 15 - Ukážka prototypu pre podporu virtuálnej reality

6.4 VRLAB PROTOTYP

Autori podkapitoli: Patrik Tománek, Viktor Beňo

Cieľom posledného prototypu bola implementácia všetkej potrebnej funkcionality na základe špecifikácií nášho projektu. Korektne zabezpečenie funkcionalít *SpiceSharp-u* a *SpiceSharpParser-u* spolu s *VRIF*. Dokončovanie prototypu prebiehalo rozsiahlym testovaním, odstraňovaním chýb a zlepšovanie celkového používateľského zážitku ako aj samotnej optimalizácií výsledného prototypu.

6.4.1 Návrh

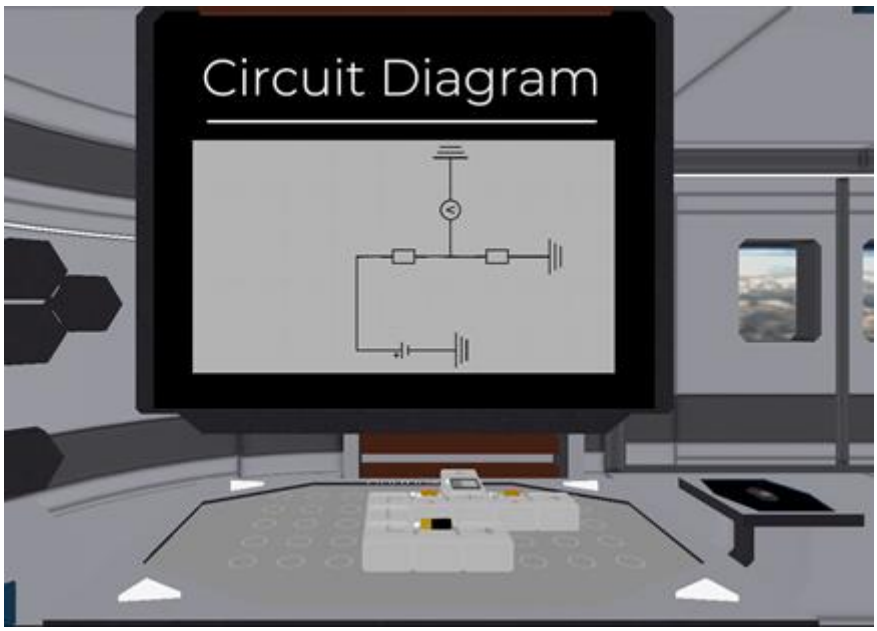
1) Tvorba elektrických obvodov

Na Obrázok 16 môžeme vidieť tvorbu elektrických obvodov v prototype, používatelia si môžu konštruovať elektrické obvody na stole pod obrazovkou, pričom sa im počas tvorby ich obvodov automaticky generuje obvodová schéma.

Na Obrázok 17 môžeme vidieť simuláciu elektrického obvodu, počas simulácie okrem numerického výsledku môže používateľ vidieť aj grafové zobrazenie. Takisto si používatelia môžu meniť hodnoty jednotlivých elektrických súčiastok v obvode ihneď počas simulácie.

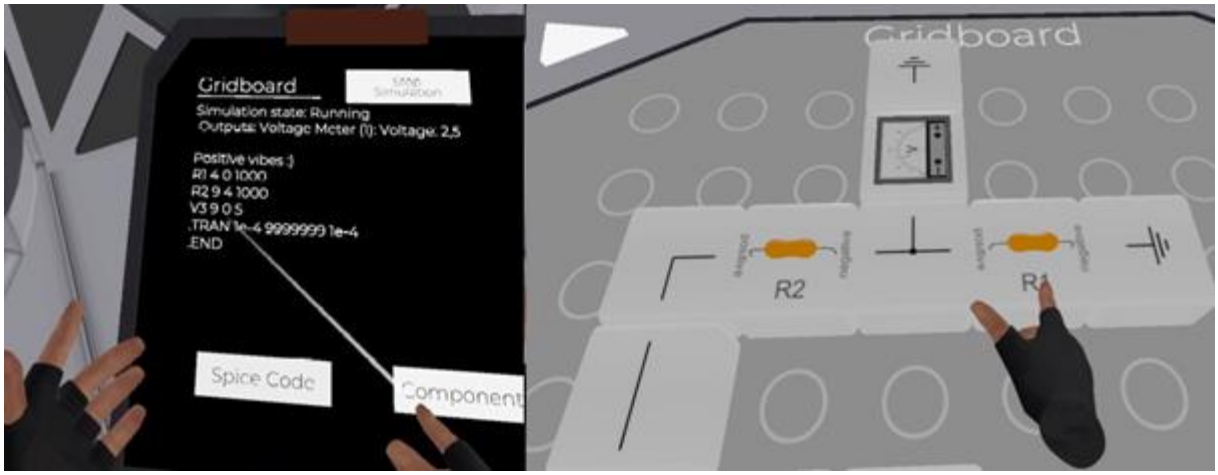
Na Obrázok 18 môžeme vidieť generovanie spice kódu a následnú úpravu komponentov v obvode. Okrem generovania obvodovej schémy, používateľom takisto poskytujeme možnosť generovania spice kódu pre ich simulovaný elektrický obvod. Spice kód takisto ako aj obvodová schéma, slúži na všeobecnú reprezentáciu elektrického obvodu. Používatelia môžu vidieť jednotlivé identifikátory vo vygenerovanom spice kóde (R1, R2, …) na nimi vytvorenom obvode aby vedeli, ktoré komponenty počas simulácie menia.

Projekt takisto obsahuje dynamické súčiastky, ktoré sa prispôsobujú výslednej simulácií. Používatelia majú možnosť simulácie svetielka, ktorého intenzita svietivosti sa mení na základe výstupu zo simulácie. Okrem svetielka im takisto ponúkame možnosť simulácie motorčeka alebo reproduktoru. Reproduktor na základe oscilačných vĺn generuje zvuk, pričom si používatelia môžu pomocou upravovania súčiastok za behu tejto simulácie upraviť hlasitosť alebo frekvenciu. Pri motorčeku si používatelia môžu prispôsobiť rýchlosť, akou sa bude točiť.



Obrázok 16 - Tvorba elektrických obvodov

Obrázok 17 - Simulácia elektrických obvodov



Obrázok 18 - Generovanie spice kódu

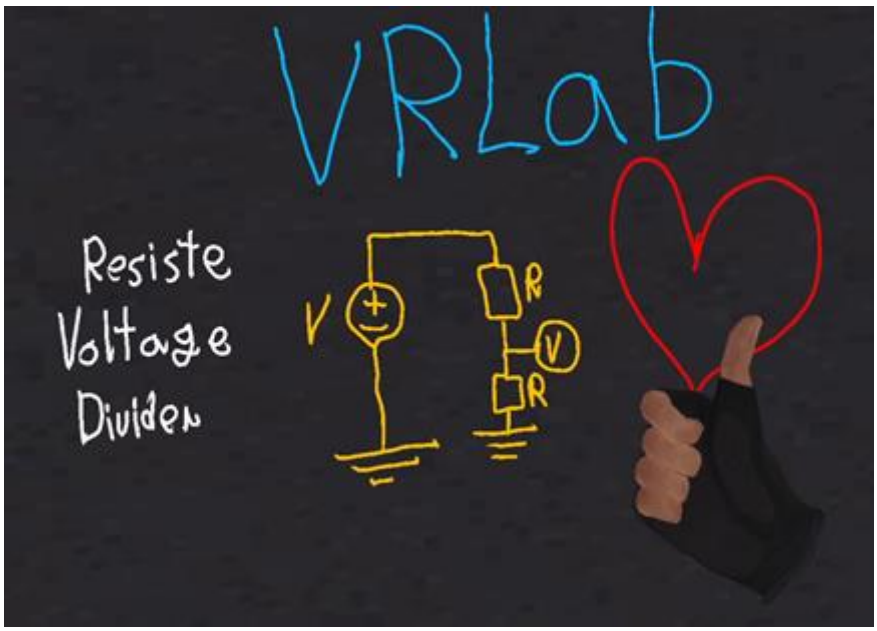
2) Úprava elektrických súčiastok

Na Obrázok 19 môžeme vidieť modifikáciu elektrických súčiastok. Používatelia si môžu upraviť jednotlivé elektrické súčiastky ako potrebujú. Môžu si napríklad nastaviť vlastnú hodnotu odporu pri resistore, pričom si môžu ešte aj vybrať metrický prefix (nano, micro, mili, …) pre nimi zvolenú hodnotu. Pokiaľ si používateľ avšak bude chcieť zmeniť celkový model súčiastky, povedzme diódy, stačí mu iba vyhľadať si špecifikácie požadovaného modelu vo forme spice výrazu, ktorý si skopíruje do systémového clipboardu a po stlačení príslušného tlačidla v hre sa mu do nej všetky modely automaticky nahrajú. Všetky modifikované súčiastky si hráč takisto môže uložiť, vďaka čomu môže tieto súčiastky používať bez potreby opätovnej úpravy hodnôt.

Obrázok 19 - Modifikácia elektrických súčiastok

3) Písanie na tabuľu

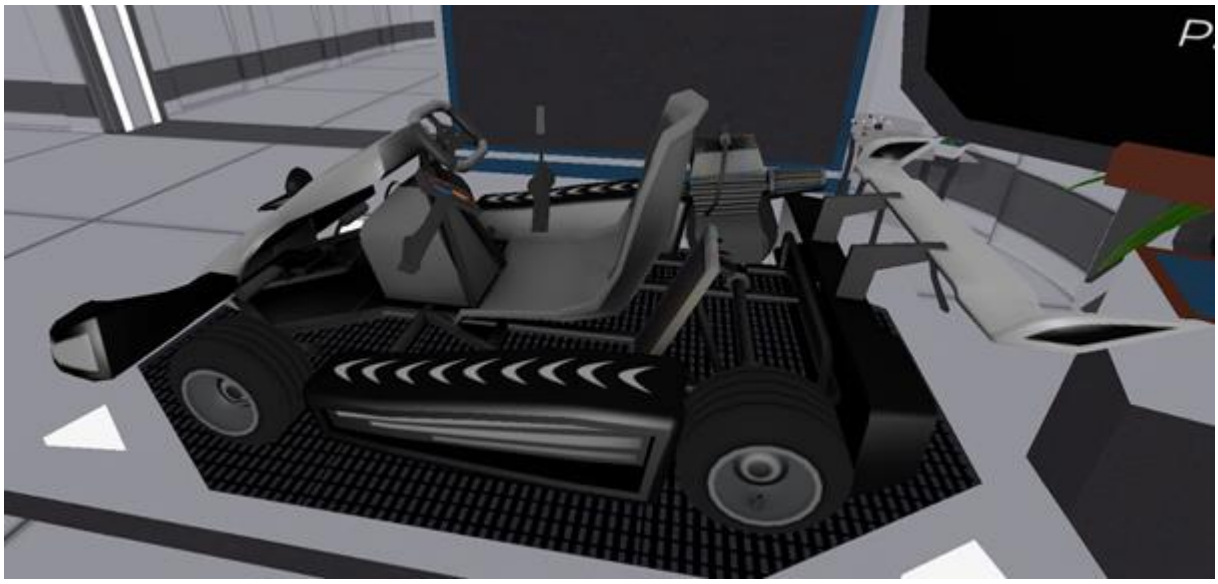
Na Obrázok 20 môžeme vidieť interaktívnu tabuľu. Používateľ si môže v prípade potreby písať akékoľvek poznámky na tabuľu, vedľa ktorej má k dispozícií niekoľko rôzne farebných fixiek.



Obrázok 20 - Interaktívna tabuľa

4) Konštrukcia obvodov pre jednotlivé komponenty motokáry

Na Obrázok 21 môžeme vidieť motokáru, pokiaľ si v nej bude chcieť zajazdiť, bude najprv avšak potrebné aby vytvoril elektrické obvody pre jednotlivé komponenty motokáry ako je motor, klaksón a svetlá. To ako rýchlo pôjde jeho motokára, akou frekvenciou a hlasitosťou bude jeho klaksón hrať alebo ako intenzívne budú jeho svetlá je čisto na používateľovi a jeho fantázií pri tvorbe obvodov. Tým menej zdatným v tvorbe elektrických obvodov poskytujeme obrázky ukážkových obvodových schém pre všetky 3 komponenty aj s požadovanými hodnotami alebo modelmi jednotlivých elektrických súčiastok.



Obrázok 21 - Motokára

5) Jazdenie na motokáre

Na Obrázok 22 môžeme vidieť motokáru pripravenú na testovaciu jazdu. Pokiaľ používateľ úspešne skonštruoval obvody pre všetky súčiastky, môže si do motokáry nasadnúť a otestovať ju vďaka plne interaktívnemu volantu, páky alebo palubnej dosky, z ktorej môže ovládať svetla a klaksón!

Obrázok 22 - Jazdenie v motokáre

6.4.2 Implementácia

Na začiatku sme sa venovali integrácie knižníc pre simuláciu elektrických obvodov *SpiceSharp* a *SpicesharpParser*, následne bolo potrebné vytvorenie spôsobu registrácií konektivít medzi jednotlivými komponentami počas ich vkladania do gridboardu. Každý gridboard obsahuje komponent *gridboardController*, ktorý je zodpovedný za detekciu konektivít všetkých súčiastok, ktoré boli doň

vložené. Tento komponent sa skladá z niekoľkých funkcií, ktoré budú detailne popísané nižšie. Názvy a atribúty týchto funkcií sú nasledovné:

- public void handleComponentInGridboard(GameObject component, GameObject slot, string InsertOrExtract)
- void checkForWireConnection(GameObject component, GameObject slot, int previousComponentDirection, List<KeyValuePair<ComponentController, int>> connectionIssuer)
- List<KeyValuePair<ComponentController, int>> finitializeConnection(List<KeyValuePair<ComponentController, int>> connectionIssuer, string InsertOrExtract)
- void connectComponents(List<KeyValuePair<ComponentController, int>> connectionIssuer)
- IEnumerator disconnectComponents(List<KeyValuePair<ComponentController, int>> connectionIssuer)
- void updateComponentAttributes(KeyValuePair<ComponentController, int> component)

Ako nultý krok každého gridboardu je nastavenie súradníc pre každý slot, do ktorého môže byť vložená súčiastka. Toto nastavenie sa vždy vykoná iba raz, a to vtedy, keď sa inicializuje komponent *gridboardController*.

Kedykoľvek sa do gridboardu vloží, resp. vyberie nejaká elektrická súčiastka, zavolá sa metóda, *handleComponentInGridboard*, kde *component* predstavuje vloženú súčiastku, *slot* predstavuje slot, do ktorého bola súčiastka vložená a *InsertOrExtract* hovorí o tom, či sme danú súčiastku do gridboardu vložili alebo ju z neho vybrali.

Dôležitou časťou tejto funkcie sú premenné:

- List<KeyValuePair<ComponentController, int>> connectionIssuer
- List<GameObject> connectionMultiWires
- List<KeyValuePair<ComponentController, int>> checkedWires

Premenná *connectionIssuer*, obsahuje všetky nájdené súčiastky, spolu s uzlom, na ktorom došlo k spojeniu počas kontroly konektivity ešte pred tým, ako boli tieto súčiastky medzi sebou spojené. Premenná *connectionMultiWires* obsahuje všetky káblové súčiastky, ktoré obsahujú viac ako 2 konce, a teda môžu spájať viac ako 2 komponenty. Premenná *checkedWires* obsahuje všetky káblové súčiastky spolu s uzlom, na ktorý sa nejaká súčiastka napojila, čo predchádza zacykleniu obvodu.

Po vložení súčiastky do gridboardu sa najprv táto súčiastka musí správne zrotovať, okrem rotácie objektu sa ale musia zrotovať aj jednotlivé uzly tejto súčiastky. Túto rotáciu zabezpečí funkcia *public void rotatePins(int count)*, kde count hovorí o tom, koľko sa má táto súčiastka zrotovať smerom doľava.

Následne sa determinuje či sa jedná o káblovú súčiastku. V takomto prípade sa najprv skontroluje, či sa jedná o normálny kábel, a teda kábel, ktorý môže niesť iba 1 prúd a vtedy sa zavolá funkcia *checkForWireConnection* a následne *finitializeConnection*. Pokiaľ sa ale jedná o kábel premostenia, čo je špeciálny prípad kedy cez 1 káblovú súčiastku môžu ísť súčasne 2 rozdielne prúdy. V takom prípade sa pre túto súčiastku zavolá funkcia *checkForWireConnection* 2-krát. Prvý krát po

vertikálnej dráhe kábla a druhý krát po horizontálnej dráhe. Pre obe tieto dráhy sa takisto aj samostatne zavolá funkcia *finitializeConnection*. Táto funkcia kontroluje, či sa v *connectionIssuer* nachádza viac ako 1 súčiastka, a podľa toho, či *InsertOrExtract* je rovný *insert* alebo *extract* zavolá funkciu *connectComponents()*, resp. *coroutine disconnectComponents*. Funkcia *connectComponents()* prebehne všetky súčiastky v *connectionIssuer*, a nastaví im unikátny identifikátor, vďaka ktorému sa vie, že tieto súčiastky sú na seba napojené. A následne tieto súčiastky vloží do *List<ComponentController> connectedComponents*. Na konci sa pre každú súčiastku ešte zavolá funkcia *updateComponentAttributes*. Táto funkcia má na starosti aktualizovanie atribútov súčiastky, ako sú napríklad: pozitívny uzol, negatívny uzol, atď. Coroutine *disconnectComponents* prebehne všetky súčiastky, odpojí ich od seba a odstráni ich z *connectedComponents*. Následne takisto zavolá funkciu *updateComponentAttributes.* Dôvod, prečo je toto Coroutine a nie funkcia je ten, že naše káblové súčiastky zahŕňajú káble s 3 alebo 4 uzlami. V takom prípade sa môžu z obvodov robiť „ostrovčeky". A po tom, ako sú všetky káble medzi sebou odpojené sa prebehne pole *connectionMultiWires*, ktoré hovorí o tom, koľkými káblami s 3 alebo 4 uzlami sa prešlo počas vyberania súčiastky z gridboardu a opätovne na tieto súčiastky zavolá funkciu *handleComponentInGridboard*, ktorý prebehne už iba tieto multi-káble a skontroluje, či sú všetky napojenia správne zaznamenané. Toto sa však ale môže vykonať až po tom, ako je súčiastka úspešne vybraná z gridboardu, a teda gridboard ju už ďalej neregistruje. Preto sa používa Coroutine spolu s *yield return new WaitForEndOfFrame()*, ktorá počká do konca framu, kedy už budú všetky operácie vzhľadom na vytiahnutie súčiastky dokončené a následne sa prebehne spomínaný *connectionMultiWires*.

Pokiaľ sa o káblovú súčiastku nejedná, a teda jedná sa o inú elektrickú súčiastku, v takom prípade sa skontroluje, či má táto súčiastka nejakých susedov. Konkrétne horného, pravého, dolného alebo ľavého suseda. Pokiaľ nejakého z týchto susedov má a zároveň je tento sused káblová súčiastka, pretože 2 súčiastky sa na seba napojiť nemôžu pokiaľ nie sú spojené káblom, tak sa do *connectionIssuer* pridá táto elektrická súčiastka a na susednú, resp. káblovú súčiastku sa zavolá metóda *checkForWireDetection* spolu s *finitializeConnection*. Keďže tieto susedné napojenia sa vždy kontrolujú vzhľadom na im príslušný uzol, pre každého suseda je tento proces zavolaný samostatne.

Funkcia *checkForWireDetection* je rekurentná funkcia, ktorá kontroluje všetky káble napojené na počiatočný kábel, na ktorý bola táto funkcia zavolaná a zisťuje, či sa našli aspoň 2 elektrické súčiastky, ktoré sú týmito káblami medzi sebou prepojené. Na začiatku sa však skontroluje, či sa tento kábel nenachádza v *checkedWires*, aby sa predišlo zacykleniu v rekurzíve. Následne sa skontroluje, či sa jedná o kábel s 3 alebo 4 uzlami, v takom prípade ho pridá do *connectionMultiWires*. Potom sa začnú kontrolovať všetci susedia tohto káblu, a teda horný, pravý, dolný a ľavý sused. Pokiaľ susedná súčiastka je ďalšia káblová súčiastka, opäť sa zavolá *checkForWireDetection* pre túto susednú káblovú súčiastku. Pokiaľ sa ale nejedná o káblovú súčiastku a teda jedná sa o inú elektrickú súčiastku, táto súčiastka sa vloží do *connectionIssuer*. Špeciálnym prípadom tejto *checkForWireDetection* funkcie sú premosťovacie káble. Keď sa kontrolujú iné káble tak sa vždy kontrolujú všetky smery, na ktorých má táto súčiastka uzol, pričom kábel premostenia má síce uzly na každej strane, no nie každým sa môže aj vyjsť. Preto v prípade, že sa jedná o premosťovací kábel sa kontroluje, či smer, na ktorom sa našla susedná súčiastka je smerom opačným, akým sa do tejto súčiastky, a teda premosťovacieho kábla vošlo.

6.4.3 Testovanie

Testovanie výsledného prototypu prebiehalo v prostredí Unity, prototyp testoval Patrik Tománek a Viktor Beňo. Okrem testovania našimi členmi tímu tento výsledný prototyp bol testovaní aj tímom č. 12 iPP. V dokumente prikladáme ich evaluáciu po testovaní nášho prototypu.

ZÁVER

Po prvých piatich šprintoch sa nám podarilo vypracovať dôležitú analýzu pre náš projekt. Na základe tejto analýzy sme si presne určili akú simuláciu budeme implementovať do nášho projektu. Konkrétne sa jednalo o implementáciu elektrických obvodov v blokovom (grid) riešení. Takisto sme sa ešte okrajovo venovali breadboard riešeniu, s ktorým vývojom nepokračujeme z viacerých dôvodov uvedených v dokumente. Pre simulovanie elektrického obvodu bola zvolená knižnica SpiceSharp, ktorá dokáže simulovať obvod v čase, umožňuje pridanie nových súčiastok a taktiež umožňuje zmenu parametrov počas simulácie. Na sieťovú synchronizáciu v Unity sme po analýze viacerých frameworkov zvolili framework Photon Bolt, ktorý poskytuje dostatok funkcionalít v bezplatnej verzii a je jednoduchý na integráciu v Unity.

Výsledkom všetkých analýz bol prvý funkčný prototyp nášho projektu, kde sme sa primárne venovali implementácií pohybu a interakcií s objektami, konkrétne interakcia elektrických súčiastok s mriežkou (gridom). Do konca zimného semestra sme vytvorili druhý funkčný prototyp, ktorého cieľom bola podpora virtuálnej reality pre náš projekt.

V letnom semestri sme sa následne zamerali na implementáciu finálneho protypu nášho projektu. V tomto prototype sme vytvorili laboratórium, ktoré prostredníctvom grid boardu umožňuje simuláciu ľubovoľných elektrických obvodov. Tieto simulácie sú realizované prostredníctvom Spice Sharp, čo je jazyk vyvinutý priamo na tento účel. Pre preklad obvodov z C# do spice kódu sme do projektu zahrnuli Spice Sharp parser, ktorý prekladá jednotlivé obvody do Spice kódu a umožňuje tak ich simuláciu.

V tomto semestri sme taktiež implementovali alternatívny scenár, k simulácii vlastných obvodov, ktorým je zapojenie a spojazdnenie motokáry, na ktorej je, po správnom zapojení a otestovaní obvodov, možné jazdiť po laboratóriu.

Ďalšími implementovanými funkcionalitami sú zmeny parametrov súčiastok v priebehu simulácie a implementácia modelov pre jednotlivé súčiastky s možnosťou pridania vlastných modelov.

Náš projekt ponúka v budúcnosti možnosti ďalšieho rozšírenia funkcionalít. Veľký potenciál predstavuje hlavne možnosť implementácie pripojenia viacerích používateľov do jedného laboratória, čo by umožnilo viesť pomocou tohto projektu celé vyučovacie hodiny zamerané na jednoduché elektrické obvody.

PRÍLOHA A. POUŽÍVATEĽSKÁ PRÍRUČKA

Autor kapitoly: Erik Paľa

VR Lab

Ovládanie PC

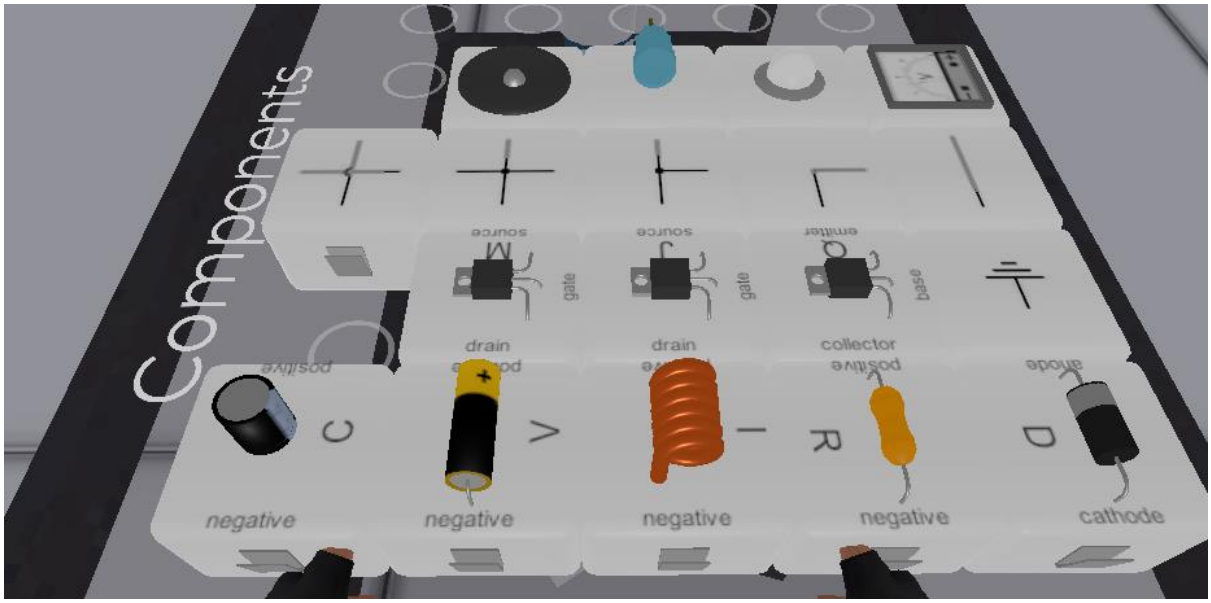
Základný pohyb	**Tlačidlo**
Vpred	**W**
Vzad	**S**
Vľavo	**A**
Vpravo	**D**
Hore	**L Alt**
Dole	**L Ctrl**
Úchop - ľavá ruka	**B**
Úchop - pravá ruka	**X**
Nastúpiť do motokáry	**R**

Aplikácia VR Lab poskytuje používateľovi rôzne možnosti interakcie. Hlavné 2 možnosti sú simulácia vlastných elektrických obvodov a modelovanie vozidla na základe preddefinovaných obvodov pre jednotlivé komponenty

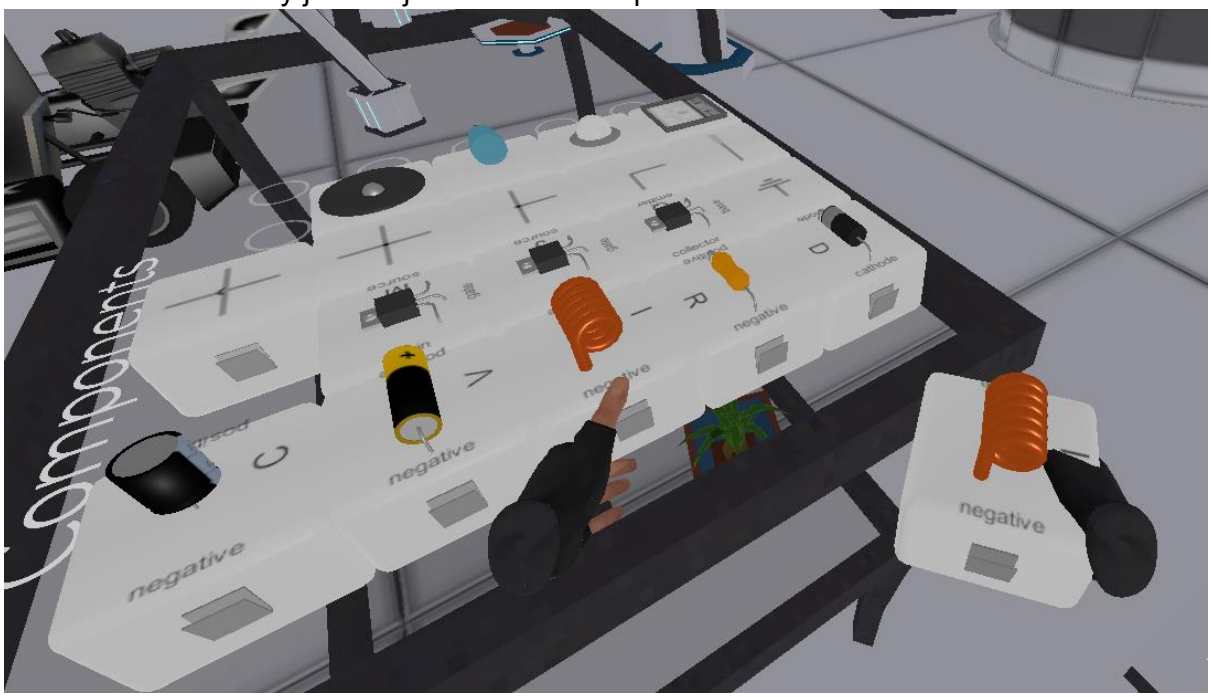
Simulácia vlastných obvodov

Výber súčiastok

Používateľ má k dispozícii stôl, na ktorom sú uložené jednotlivé súčiastky.

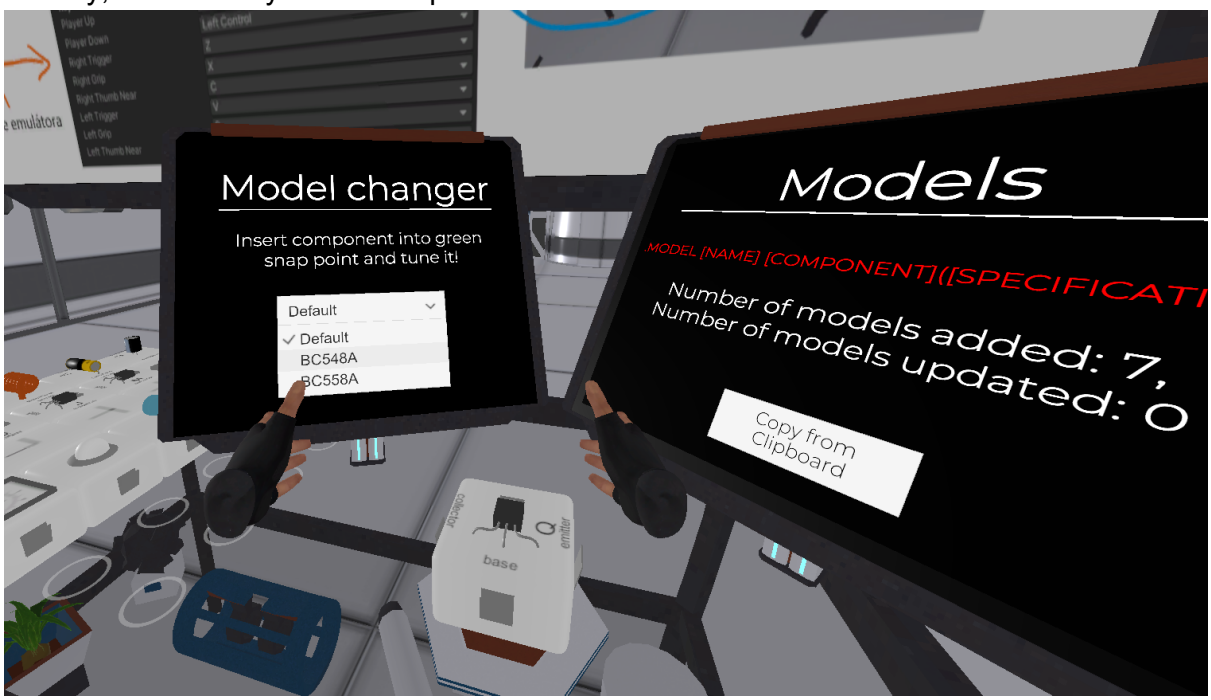


Pre získanie súčiastky ju stačí jednoducho uchopiť.

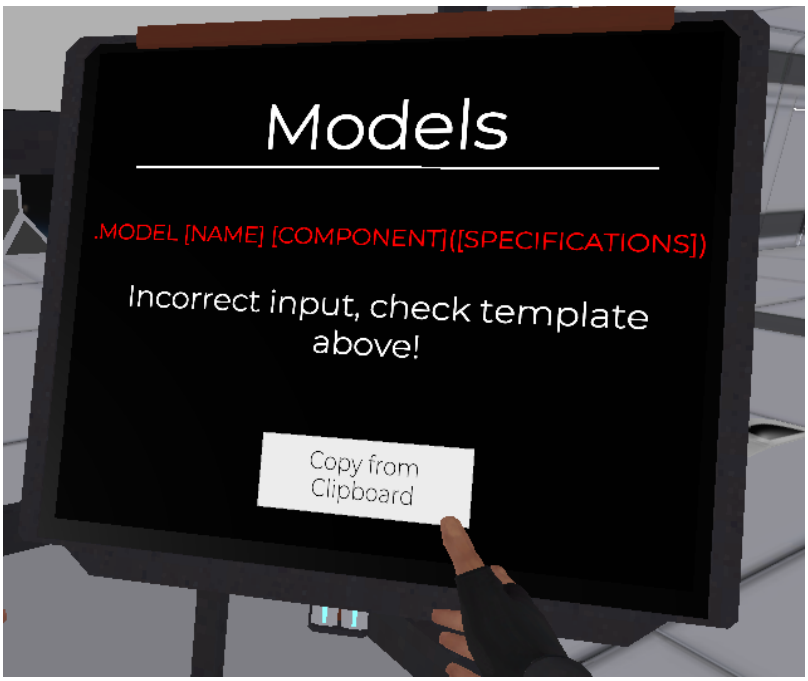


Nastavenie hodnôt súčiastok

Používateľ má k dispozícii stanicu, do ktorej uloží súčiastku a následne si pre ňu môže vybrať modely, ktoré sú v systéme k dispozícii.

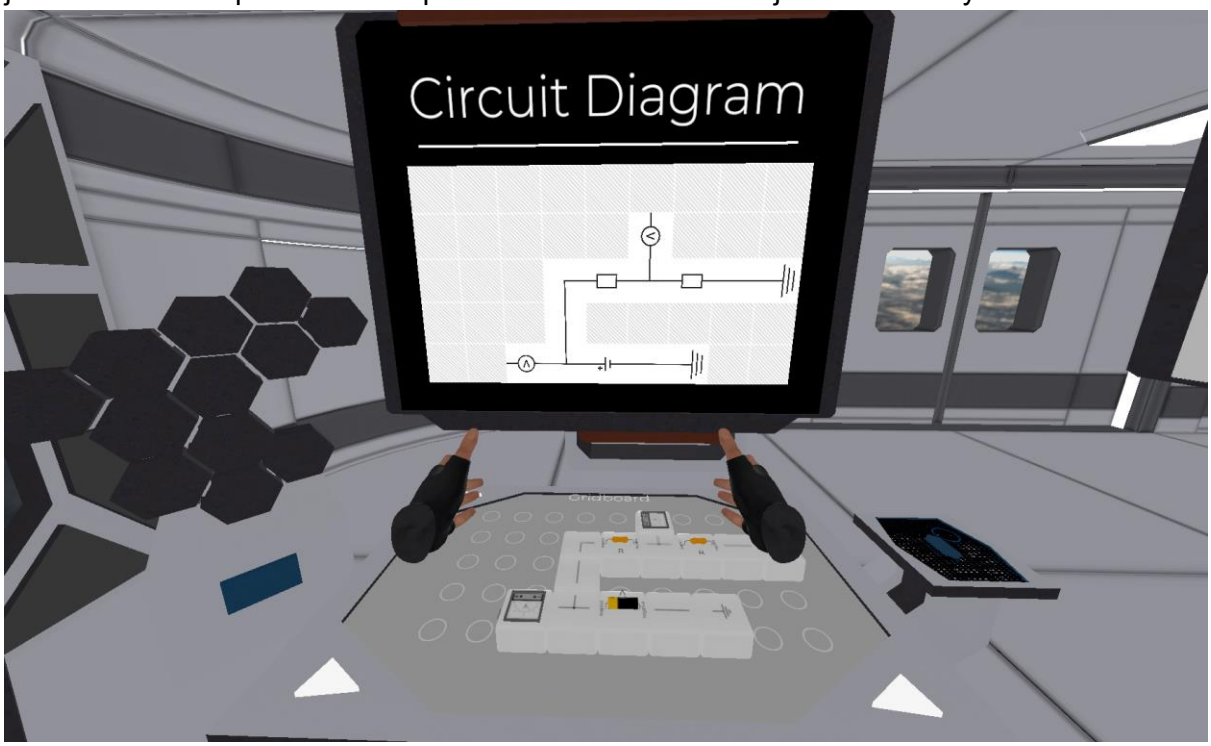


V prípade, že ani jeden z ponúkaných modelov nezodpovedá potrebám používateľa, môže si používateľ definovať nový model na základe jeho spice kódu, ktorý je dohľadateľný na internete. Po potvrdení výberu sa súčiastke nastavia parametre zodpovedajúce zvolenému modelu.



Umiestnenie komponentu

Po nastavení hodnôt súčiastky ju môže používateľ umiestniť na simulačný stôl. Súčiastku stačí jednoducho uchopiť z terminálu pre nastavenie a umiestniť ju na simulačný stôl.

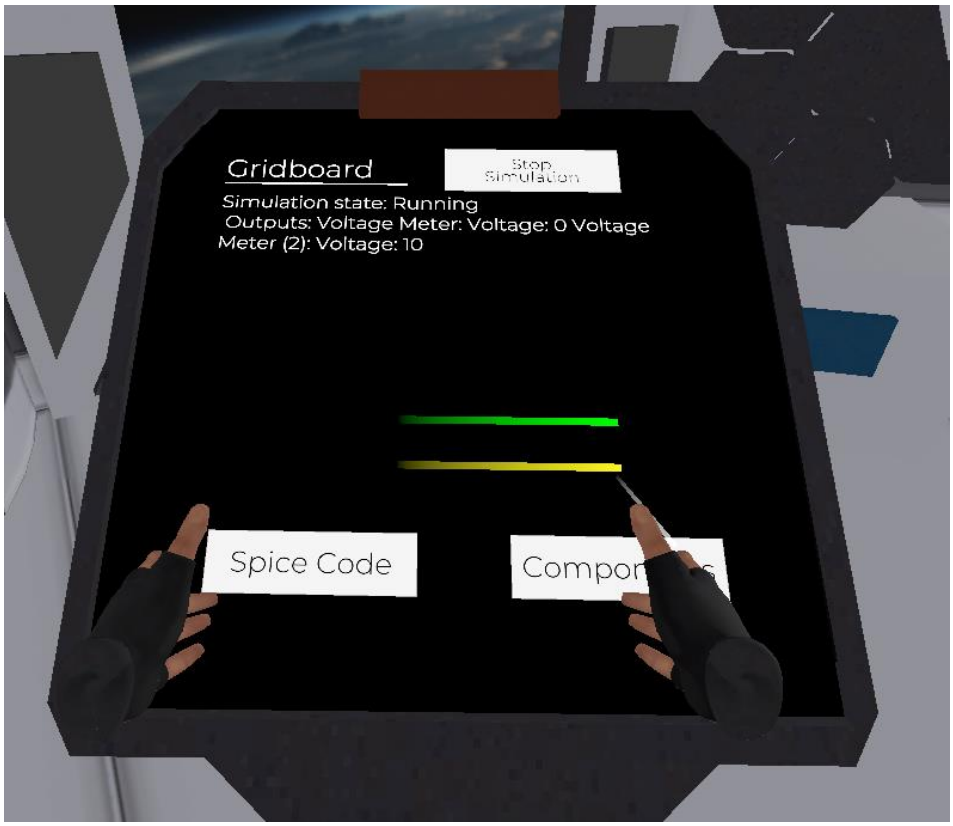


Pri umiestňovaní súčiastky sa na stole zvýraznia miesta kam je možné súčiastku umiestniť. Nad stolom sa zároveň zobrazuje aktuálna schéma obvodu, ktorý je na stole zapojený.

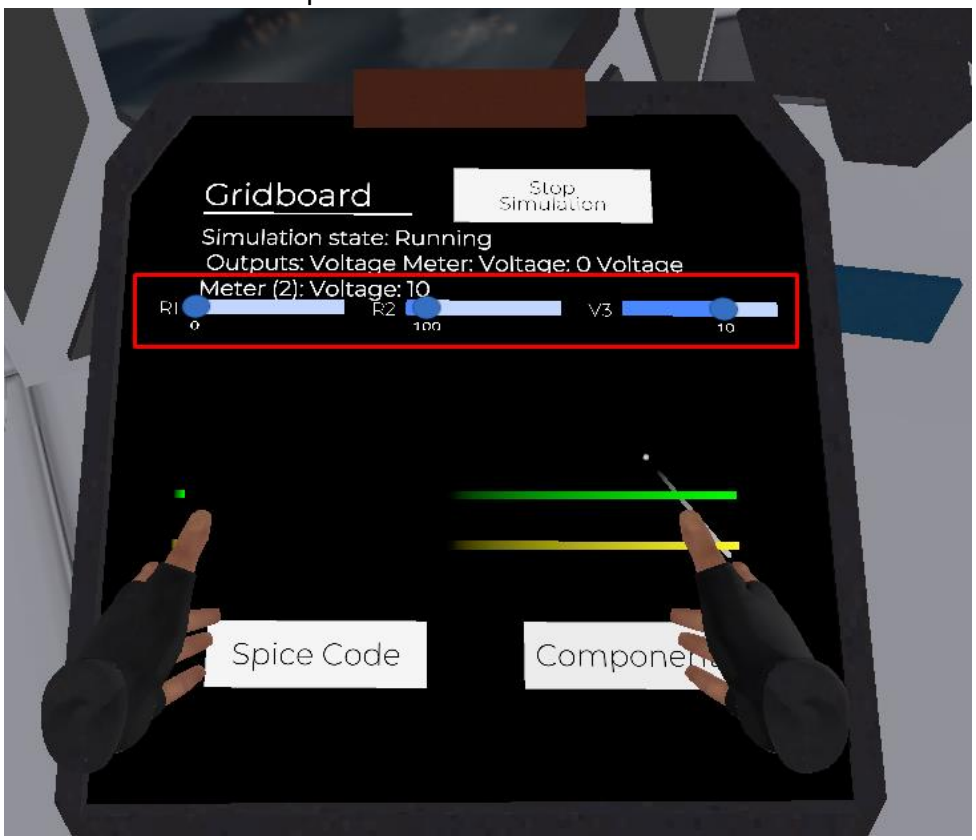
Simulácia obvodu

Po zapojení požadovaného obvodu môže používateľ spustiť simuláciu obvodu.

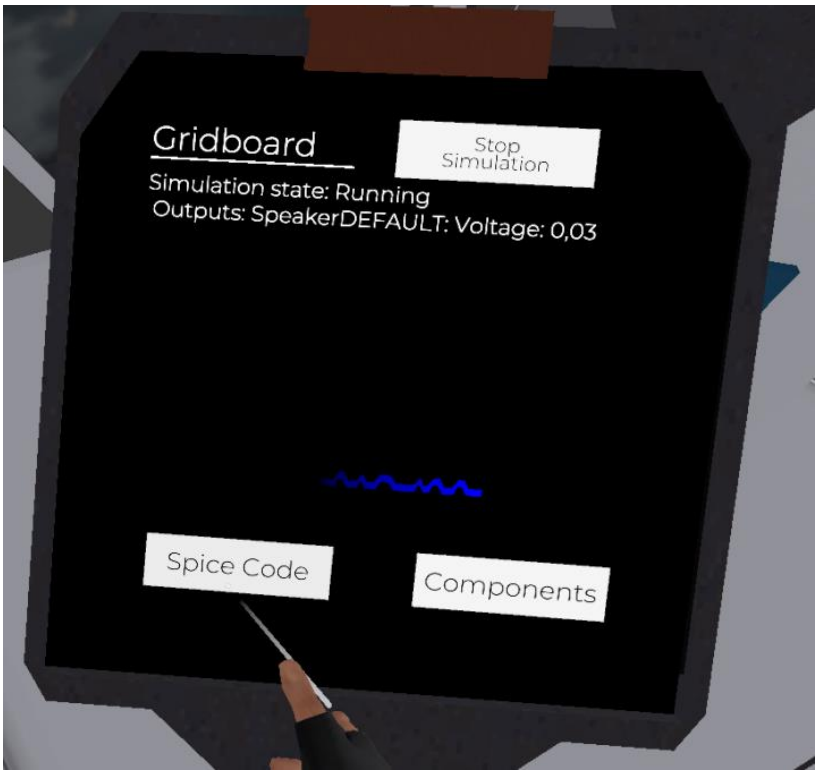
Na tomto paneli sa následne zobrazí výstup obvodu, podľa bodov, kam používateľ umiestnil meracie zariadenia.



Po stlačení tlačidla components sa pod výstupom zobrazia komponenty, ktorým je možné počas priebehu simulácie meniť parametre.



Tento terminál zároveň automaticky monitoruje oscilácie v obvode



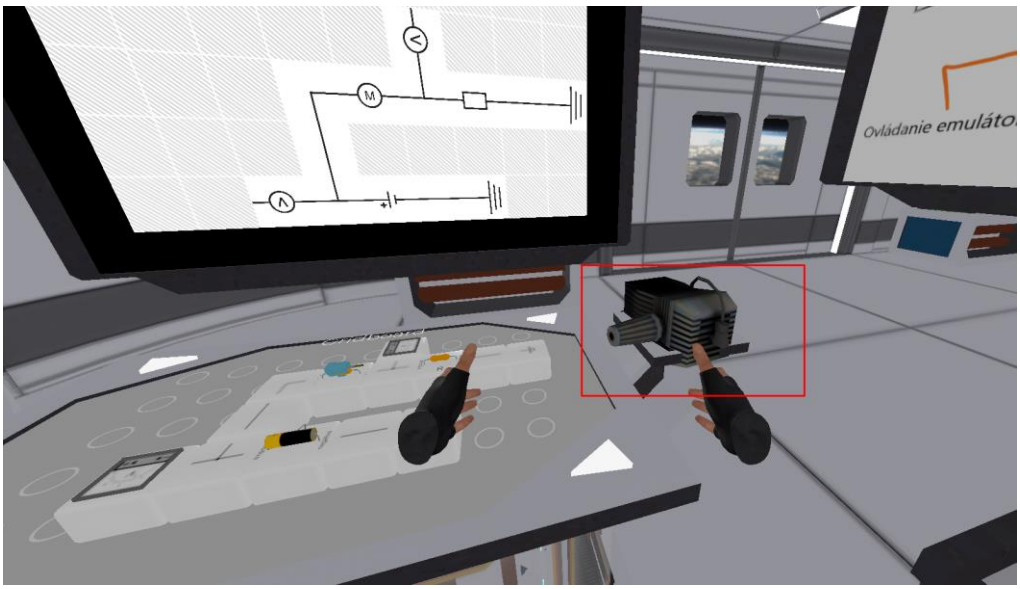
Modelovanie obvodov pre motokáru

VR Lab ponúka ako alternatívu k bežnému skladaniu obvodov možnosť, kde si používateľ môže pomocou vytvárania obvodov spojazdniť motokáru, na ktorej je mu potom umožnené jazdiť.

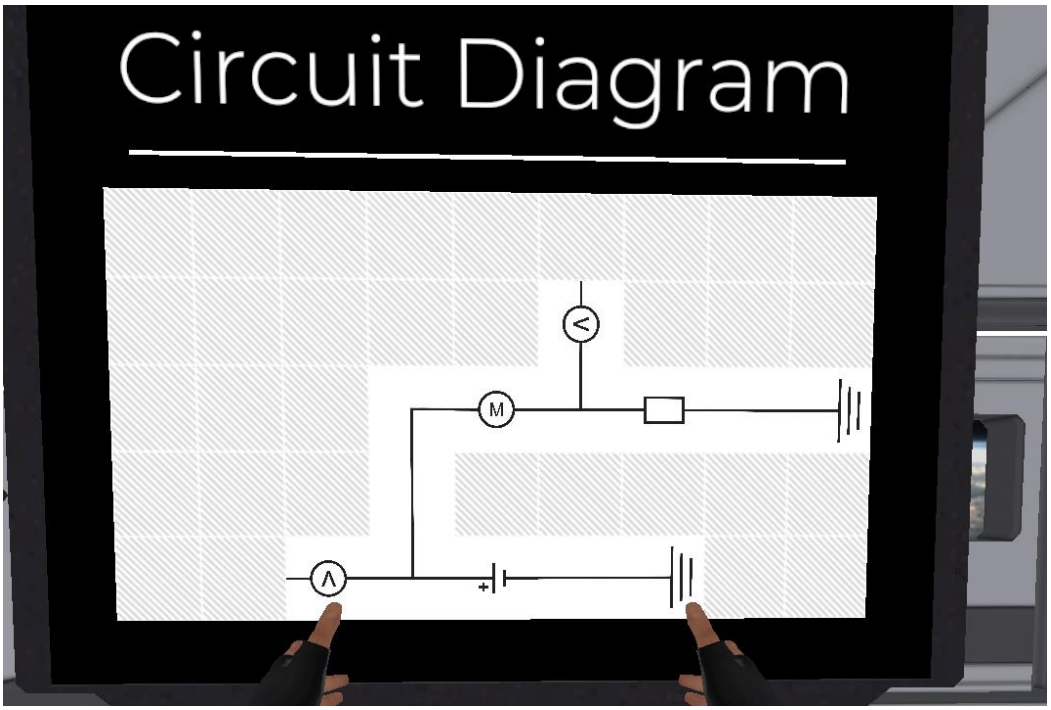
Obvody pre jednotlivé komponenty

Jednotlivé komponenty motokáry (napríklad motor) majú definované obvody, ktoré musí používateľ zapojiť aby bola súčiastka funkčná.
Obvody pre jednotlivé komponenty sa zapájajú na simulačnom stole. Z motokáry stačí požadovanú súčiastku jednoducho uchopiť a odniesť k stolu.

Pri stole je na tieto komponenty vyčlenený špeciálny podstavec, kam je možné súčiastku umiestniť.



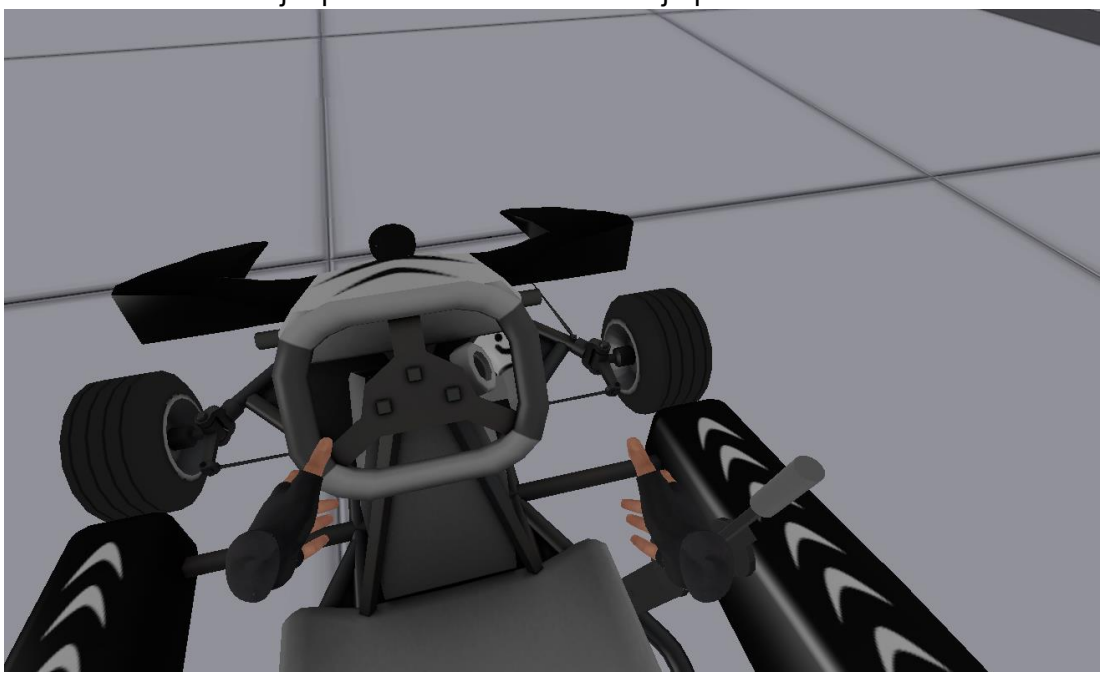
Po umiestnení komponentu sa preň načíta preddefinovaný obvod, ktorý je potrebné zapojiť, aby bola táto súčiastka funkčná.



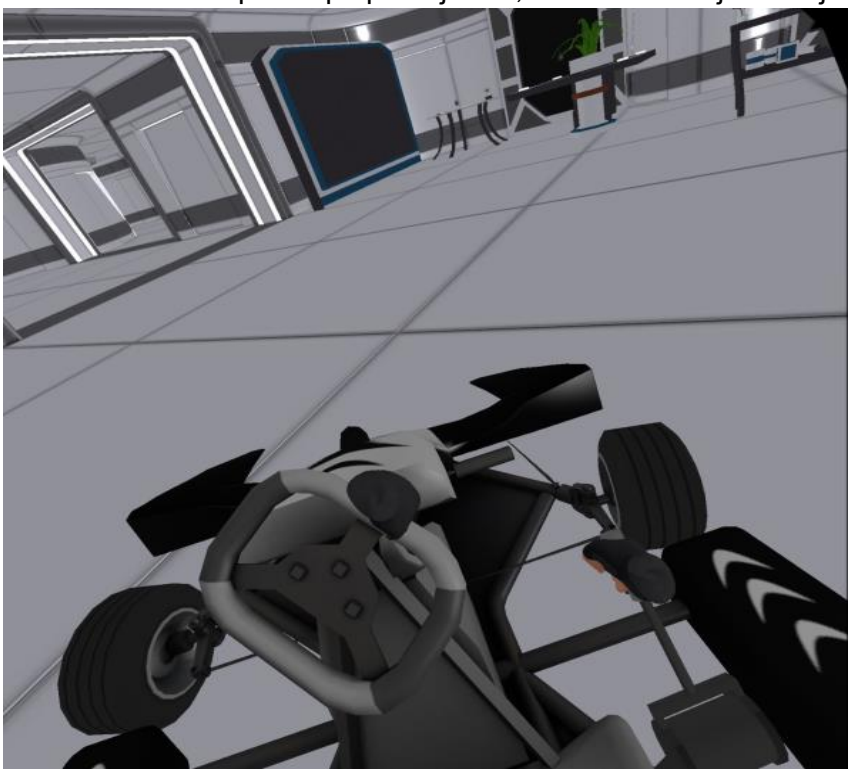
Po zapojení a odsimulovaní súčiastky ju používateľ jednoducho vráti späť na pôvodné miesto na motokáre.

Kompletná motokára

Po tom, čo používateľ zapojí všetky potrebné súčiastky pre chod motokáry, môže do nej nasadnúť a riadiť ju. pre nasadnutie do vozidla je potrebné stlačiť tlačidlo “R”.



Motokára sa ovláda volantom a pákou po pravej ruke, ktorou sa určuje smer jazdy.



PRÍLOHA B. ZDROJOVÉ KÓDY

Autor kapitoly: Patrik Tománek

Fakulta informatiky a informačných technológií
Slovenská technická univerzita

VRLab Radiant

Inštrukcie k zdrojovému kódu

Akademický rok: 2020/2021
Pedagogický vedúci: Ing. Juraj Vincúr
Členovia tímu (č.16): Bc. Patrik Tománek
Bc. Ľubomír Kurčák
Bc. Erik Paľa
Bc. Viktor Beňo

Project Build

Projekt je dostupný na Google Drive:
https://drive.google.com/file/d/1iYILRZR_vOqnWwkj2Zf9va8gTuxZhT-Z

Po stiahnutí a rozbalení stačí spustiť *VRLab.exe* a riadiť sa Prílohou A: Používateľská príručka.

GitHub

Celý projekt sme uložili na GitHub: https://github.com/patrikTomanek/VRLab

Inštrukcie k obsahu

Keďže sa na GitHub linku nachádza celý náš projekt, uvádzame stručné inštrukcie k nášmu projektu

Zdrojové kódy

Nami vytvorené kódy sa nachádzajú v:
https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Scripts

3D modely

Na tvorbu a úpravu modelov používame modelovací program Blender. Nami vytvorené 3D modely elektrických súčiastok sa nachádzajú v:

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Electronic%20Components

Textúry, tiene a materiály používané na modeloch sa nachádzajú v:

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Textures

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Shaders

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Materials

VR Interaction Framework

Tento framework obsahuje komplexnú funkcionalitu pre virtuálnu realitu

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/BNG%20Framework

SpiceSharp & SpiceSharpParser knižnica

Slúži na simuláciu elektrických obvodov

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/SpiceSharp

Ostatné

Zabezpečenie simulácie virtuálnej reality:

https://github.com/lubomirkurcak/vrlab/tree/master/Assets/XR

Audio súbory použité v hre:

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Audio

Rôzne modely, textúry a materiály stiahnuté z Unity Asset Store:

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Low%20Poly%20Kart%20With%20Player%2015

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/Sci-Fi%20Styled%20Modular%20Pack

https://github.com/patrikTomanek/VRLab/tree/main/VRLab/Assets/SkySerie%20Freebie

Inštrukcie k spusteniu

- Je potrebná inštalácia modelovacieho programu Blender (aktuálna verzia)
- Je potrebná inštalácia Unity Hub ○ Po vložení nášho projektu do Unity Hub Vám bude automaticky poskytnutá možnosť nainštalovať si verziu tohto projektu (pokiaľ Vám z nejakých dôvodov táto možnosť poskytnutá nebude, verzia nášho projektu je: 2019.4.12f1)
- **Testovanie virtuálnej reality**
    - **Emulátor**
        - na *XR Rig VRLab* aktivujte komponent *VR Emulator*.
    - **Oculus Quest 1**
        - Potrebný Virtual Desktop, prepojenie headsetu s počítačom
        - V Unity Editore -> Edit -> Project Settings -> XR Plub-in Managment -> Initialize XR on Startup

PRÍLOHA C. PROTOKOL Z TESTOVANIA

Autori kapitoly: Viktor Beňo, Patrik Tománek, tím č. 12

iPP (tím č. 12): Posudok na prototyp tímu č. 16: VRLab

Hodnotenie prototypu

1. **Ako náročné bolo vykonanie testovacej úlohy?** (ako náročná je orientácia v aplikácií, či sú zrozumiteľne oddelené jednotlivé časti aplikácie, ako náročná bola práca s UI, či bola aplikácia dostatočne intuitívna, atď.)

   Odpoveď: Testovacia úloha nebola náročná, orientácia v aplikácií bola jednoduchá. Po krátkej chvíli bolo jednoduché si zvyknúť na ovládanie. Práca s UI nebola náročná a bola viac menej intuitívna.

2. **Našli ste nejaké vážne nedostatky vo funkcionalite?** (či ste odhalili neprítomnosť nejakej funkcionality, o ktorej si myslíte, že je pre takúto aplikáciu dôležitá alebo niektorá z funkcionalít nebola dostatočná poprípade chybná, atď.)

   Odpoveď: Neodhalili sme žiadne nedostatky pri vykonávaní testovacieho scenáru.

3. **Vidíte v takejto aplikácií potenciál?** (či si viete predstaviť, že by výučba prebiehala takouto formou, či by dokázala takáto aplikácia zlepšiť a zefektívniť fyzikálne laboratórne cvičenia alebo si myslíte, že výučba v školských priestoroch bude stále efektívnejšia, atď.)

   Odpoveď: Vieme si predstaviť využitie tejto aplikácie v rámci predmetu fyziky, konkrétne zameranie na elektrické obvody.

Zhodnotenie prototypu

(akékoľvek pripomienky, ktoré vznikli počas testovania aplikácie ako napríklad pripomienky na vizuálne spracovanie alebo čokoľvek iné)

Odpoveď: Prototyp pôsobí zaujímavo a má potenciál. Žiadne pripomienky počas testovania nevznikli. Páčilo sa nám aj vizuálne spracovanie testovacieho prostredia.

PRÍLOHA D. GENEROVANÁ TECHNICKÁ DOKUMENTÁCIA

Autor kapitoly: Patrik Tománek

VRLab

Generated by Doxygen 1.8.20

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 amongus Class Reference . . . 7

4.1.1 Member Function Documentation . . . 7

4.1.1.1 spawnAmongus() . . . 8

4.1.1.2 Start() . . . 8

4.1.1.3 Update() . . . 8

4.1.2 Member Data Documentation . . . 8

4.1.2.1 ejectRotation . . . 8

4.1.2.2 ejectSpeed . . . 8

4.1.2.3 hexColors . . . 8

4.1.2.4 kindasus . . . 8

4.1.2.5 monkaLaugh . . . 9

4.1.2.6 moveXorZ . . . 9

4.1.2.7 spawnpoints . . . 9

4.2 CircuitBuilder Class Reference . . . 9

4.2.1 Member Function Documentation . . . 10

4.2.1.1 changeAttribute() . . . 10

4.2.1.2 changeResistance() . . . 10

4.2.1.3 getSpiceSharpOutputs() . . . 10

4.2.1.4 runSimulation() . . . 10

4.2.1.5 SpiceSharp_Viktor() . . . 10

4.2.1.6 TransformIntoSpiceSharpInput() . . . 11

4.2.2 Member Data Documentation . . . 11

4.2.2.1 inductor . . . 11

4.2.2.2 mutualInductance . . . 11

4.2.2.3 output . . . 11

4.2.2.4 outputs . . . 11

4.2.2.5 outputsResult . . . 11

4.2.2.6 resistance . . . 11

4.2.2.7 resistor . . . 12

4.2.2.8 simulation . . . 12

4.2.2.9 spiceCode . . . 12

4.2.2.10 string . . . 12

4.2.2.11 tb . . . 12

4.2.2.12 tran . . . 12
4.2.2.13 voltageSource . . . 12
4.3 ComponentController Class Reference . . . 13
4.3.1 Member Function Documentation . . . 14
4.3.1.1 getDefaultNameTag() . . . 14
4.3.1.2 rotatePins() . . . 14
4.3.1.3 Start() . . . 14
4.3.1.4 Update() . . . 14
4.3.2 Member Data Documentation . . . 14
4.3.2.1 Anode . . . 14
4.3.2.2 BaseNode . . . 15
4.3.2.3 BulkNode . . . 15
4.3.2.4 CapacitanceValue . . . 15
4.3.2.5 Cathode . . . 15
4.3.2.6 CollectorNode . . . 15
4.3.2.7 ComponentType . . . 15
4.3.2.8 Coupling . . . 15
4.3.2.9 currentNodesPositions . . . 15
4.3.2.10 Dc . . . 16
4.3.2.11 defaultNameTag . . . 16
4.3.2.12 DrainNode . . . 16
4.3.2.13 DynamicGameObject . . . 16
4.3.2.14 EmitterNode . . . 16
4.3.2.15 Flow . . . 16
4.3.2.16 GateNode . . . 16
4.3.2.17 Inductance . . . 16
4.3.2.18 InductorName1 . . . 17
4.3.2.19 InductorName2 . . . 17
4.3.2.20 Model . . . 17
4.3.2.21 NameTag . . . 17
4.3.2.22 NegativeControllingNode . . . 17
4.3.2.23 NegativeNode . . . 17
4.3.2.24 nodesPositions . . . 17
4.3.2.25 OutputType . . . 17
4.3.2.26 PositiveControllingNode . . . 18
4.3.2.27 PositiveNode . . . 18
4.3.2.28 Resistance . . . 18
4.3.2.29 SourceNode . . . 18
4.3.2.30 SubstrateNode . . . 18
4.3.2.31 valueUnit . . . 18
4.3.2.32 WireType . . . 18
4.4 ComponentHandler Class Reference . . . 19

4.4.1 Member Function Documentation . . . 19
4.4.1.1 OnTriggerEnter() . . . 19
4.4.1.2 Start() . . . 19
4.4.1.3 Update() . . . 19
4.4.2 Member Data Documentation . . . 19
4.4.2.1 body . . . 20
4.4.2.2 insertedItemIC . . . 20
4.4.2.3 isInserted . . . 20
4.4.2.4 parentContainer . . . 20
4.5 Dashboard Class Reference . . . 20
4.5.1 Member Function Documentation . . . 21
4.5.1.1 hornOff() . . . 21
4.5.1.2 hornOn() . . . 21
4.5.1.3 lightsOnOff() . . . 21
4.5.1.4 setUpMotor() . . . 21
4.5.1.5 Start() . . . 21
4.5.1.6 Update() . . . 21
4.5.2 Member Data Documentation . . . 22
4.5.2.1 engine . . . 22
4.5.2.2 horn . . . 22
4.5.2.3 lightbulb . . . 22
4.5.2.4 lights . . . 22
4.6 GoKartEnabler Class Reference . . . 22
4.6.1 Member Function Documentation . . . 23
4.6.1.1 positionPlayerToKart() . . . 23
4.6.1.2 removePlayerFromKart() . . . 23
4.6.1.3 Start() . . . 23
4.6.1.4 Update() . . . 23
4.6.2 Member Data Documentation . . . 23
4.6.2.1 AK . . . 23
4.6.2.2 Player . . . 24
4.6.2.3 PlayerController . . . 24
4.6.2.4 PlayerSittingPosition . . . 24
4.6.2.5 screenFader . . . 24
4.7 GoKartScreenFader Class Reference . . . 24
4.7.1 Member Function Documentation . . . 25
4.7.1.1 fadeOutJustInCaseSmileyFace() . . . 25
4.7.1.2 OnTriggerEnter() . . . 25
4.7.1.3 OnTriggerExit() . . . 25
4.7.2 Member Data Documentation . . . 25
4.7.2.1 screenFader . . . 25
4.8 GraphFast Class Reference . . . 25

4.8.1 Member Function Documentation . . . 26
4.8.1.1 createGraph() . . . 26
4.8.1.2 finishGraphsLine() . . . 26
4.8.1.3 hideGraph() . . . 26
4.8.1.4 resetGraph() . . . 27
4.8.1.5 showGraph() . . . 27
4.8.1.6 Start() . . . 27
4.8.1.7 Update() . . . 27
4.8.2 Member Data Documentation . . . 27
4.8.2.1 chart . . . 27
4.8.2.2 chosenColors . . . 27
4.8.2.3 colors . . . 27
4.8.2.4 finishingPath . . . 28
4.8.2.5 generatePoints . . . 28
4.8.2.6 graphCreated . . . 28
4.8.2.7 graphScalerX . . . 28
4.8.2.8 graphScalerY . . . 28
4.8.2.9 lineRenderer . . . 28
4.8.2.10 lineRenderers . . . 28
4.8.2.11 numericOutputs . . . 28
4.8.2.12 positionIndex . . . 29
4.8.2.13 TimerX . . . 29
4.9 gridboardController Class Reference . . . 29
4.9.1 Member Function Documentation . . . 30
4.9.1.1 checkForWireConnection() . . . 30
4.9.1.2 componentsRemovalCoroutine() . . . 30
4.9.1.3 connectComponents() . . . 30
4.9.1.4 createSpiceSharpInput() . . . 31
4.9.1.5 createSpiceSharpSimulation() . . . 31
4.9.1.6 disconnectComponents() . . . 31
4.9.1.7 finitializeConnection() . . . 31
4.9.1.8 handleComponentInGridboard() . . . 31
4.9.1.9 removeAllComponentsFromGridboard() . . . 31
4.9.1.10 setUpChildCoordinates() . . . 31
4.9.1.11 Start() . . . 32
4.9.1.12 Update() . . . 32
4.9.1.13 updateComponentAttributes() . . . 32
4.9.2 Member Data Documentation . . . 32
4.9.2.1 capacitorsNodesForIC . . . 32
4.9.2.2 checkedWires . . . 32
4.9.2.3 columns . . . 32
4.9.2.4 componentsCircuit . . . 32

4.9.2.5 componentsDynamicAttribute . . . 33
4.9.2.6 componentsOutput . . . 33
4.9.2.7 connectedComponents . . . 33
4.9.2.8 connectionIndexIdentifier . . . 33
4.9.2.9 connectionMultiWires . . . 33
4.9.2.10 deleteComponents . . . 33
4.9.2.11 gameobjectsOutput . . . 33
4.9.2.12 MC . . . 34
4.9.2.13 rows . . . 34
4.9.2.14 separatorsForComponentAttribute . . . 34
4.9.2.15 slots . . . 34
4.9.2.16 SpiceCode . . . 34
4.9.2.17 SSC . . . 34
4.9.2.18 tuningStation . . . 34
4.10 gridboardPosition Class Reference . . . 35
4.10.1 Member Data Documentation . . . 35
4.10.1.1 positionInGridboard . . . 35
4.11 HandsController Class Reference . . . 35
4.11.1 Member Function Documentation . . . 36
4.11.1.1 dropItem() . . . 36
4.11.1.2 getGrabbedItem() . . . 36
4.11.1.3 pickupItem() . . . 36
4.11.1.4 removeItem() . . . 36
4.11.1.5 Start() . . . 36
4.11.1.6 Update() . . . 36
4.11.2 Member Data Documentation . . . 37
4.11.2.1 currentPickedItem . . . 37
4.11.2.2 isGrabbed . . . 37
4.12 InsertController Class Reference . . . 37
4.12.1 Member Function Documentation . . . 38
4.12.1.1 insertItem() . . . 38
4.12.1.2 pullItem() . . . 38
4.12.1.3 saveItemToGrid() . . . 38
4.12.1.4 snapItem() . . . 38
4.12.1.5 Start() . . . 38
4.12.1.6 Update() . . . 38
4.12.2 Member Data Documentation . . . 38
4.12.2.1 insertedItem . . . 39
4.12.2.2 isUsed . . . 39
4.13 InteractionController Class Reference . . . 39
4.13.1 Member Function Documentation . . . 39
4.13.1.1 Start() . . . 40

4.13.1.2 Update() . . . 40
4.13.2 Member Data Documentation . . . 40
4.13.2.1 Hands . . . 40
4.13.2.2 HC . . . 40
4.13.2.3 highlightedItem . . . 40
4.13.2.4 invisibleMaterial . . . 40
4.13.2.5 outlinedMaterial . . . 40
4.13.2.6 pickupRange . . . 41
4.13.2.7 playerCamera . . . 41
4.13.2.8 simulatedItem . . . 41
4.14 ModelChanger Class Reference . . . 41
4.14.1 Member Function Documentation . . . 42
4.14.1.1 changeComponentsUnit() . . . 42
4.14.1.2 changeComponentsValue() . . . 42
4.14.1.3 changeModel() . . . 42
4.14.1.4 createModels() . . . 42
4.14.1.5 createModelsFromClipboard() . . . 43
4.14.1.6 extractModelFromTuningStation() . . . 43
4.14.1.7 getModelComponent() . . . 43
4.14.1.8 getModelSpecification() . . . 43
4.14.1.9 initializeValueChanger() . . . 43
4.14.1.10 insertModelIntoTuningStation() . . . 43
4.14.1.11 Start() . . . 43
4.14.1.12 Update() . . . 44
4.14.2 Member Data Documentation . . . 44
4.14.2.1 customModels . . . 44
4.14.2.2 dropdown . . . 44
4.14.2.3 insertedCC . . . 44
4.14.2.4 models . . . 44
4.14.2.5 unit . . . 44
4.14.2.6 valueChanger . . . 44
4.15 MovementController Class Reference . . . 45
4.15.1 Member Function Documentation . . . 45
4.15.1.1 Start() . . . 45
4.15.1.2 Update() . . . 45
4.15.2 Member Data Documentation . . . 45
4.15.2.1 CC . . . 46
4.15.2.2 gravity . . . 46
4.15.2.3 jumpSpeed . . . 46
4.15.2.4 lelel . . . 46
4.15.2.5 lookSpeed . . . 46
4.15.2.6 lookXLimit . . . 46

4.15.2.7 moveDirection . . . 46
4.15.2.8 movementSpeed . . . 46
4.15.2.9 playerCamera . . . 47
4.15.2.10 rotationX . . . 47
4.16 PlayerInteractions Class Reference . . . 47
4.16.1 Member Function Documentation . . . 47
4.16.1.1 Start() . . . 47
4.16.1.2 Update() . . . 47
4.16.2 Member Data Documentation . . . 48
4.16.2.1 isInKart . . . 48
4.17 Scheme Class Reference . . . 48
4.17.1 Member Function Documentation . . . 48
4.17.1.1 generateSchemeImages() . . . 49
4.17.1.2 resetSchemeImage() . . . 49
4.17.1.3 Start() . . . 49
4.17.1.4 Update() . . . 49
4.17.1.5 updateSchemeImage() . . . 49
4.17.2 Member Data Documentation . . . 49
4.17.2.1 components . . . 49
4.17.2.2 defaultSprite . . . 50
4.17.2.3 image . . . 50
4.17.2.4 images . . . 50
4.17.2.5 increaseX . . . 50
4.17.2.6 increaseY . . . 50
4.17.2.7 modelChanger . . . 50
4.17.2.8 symbols . . . 50
4.18 SpeedLever Class Reference . . . 51
4.18.1 Member Function Documentation . . . 51
4.18.1.1 Start() . . . 51
4.18.1.2 Update() . . . 51
4.18.2 Member Data Documentation . . . 51
4.18.2.1 SpeedLeverHinge . . . 51
4.18.2.2 SpeedLeverInput . . . 52
4.19 SpiceSharpController Class Reference . . . 52
4.19.1 Member Function Documentation . . . 53
4.19.1.1 changeDynamicAttribute() . . . 53
4.19.1.2 closeSpiceSharpInstance() . . . 53
4.19.1.3 createSpiceSharpInstance() . . . 53
4.19.1.4 initializeDynamicComponentsUI() . . . 53
4.19.1.5 resetDynamicOutputs() . . . 54
4.19.1.6 showOrHideDynamicComponents() . . . 54
4.19.1.7 showOrHideSpiceCode() . . . 54

4.19.1.8 Start() . . . 54
4.19.1.9 Update() . . . 54
4.19.1.10 validateOutput() . . . 54
4.19.2 Member Data Documentation . . . 54
4.19.2.1 CB . . . 54
4.19.2.2 ComponentsOutput . . . 55
4.19.2.3 defaultVehicleComponentIndex . . . 55
4.19.2.4 DynamicAttributes . . . 55
4.19.2.5 DynamicComponent . . . 55
4.19.2.6 DynamicComponents . . . 55
4.19.2.7 dynamicComponentsButton . . . 55
4.19.2.8 DynamicComponentsSliders . . . 55
4.19.2.9 GameObjectsOutput . . . 56
4.19.2.10 graph . . . 56
4.19.2.11 increasing . . . 56
4.19.2.12 isZero . . . 56
4.19.2.13 removeComponentsButton . . . 56
4.19.2.14 simulationOutput . . . 56
4.19.2.15 speakerCurrentPeak . . . 56
4.19.2.16 speakerFrequencyTimer . . . 56
4.19.2.17 spiceCode . . . 57
4.19.2.18 spiceCodeButton . . . 57
4.19.2.19 SpiceSharpThread . . . 57
4.19.2.20 startSimulationButton . . . 57
4.19.2.21 stopSimulationButton . . . 57
4.19.2.22 vehicleComponentInGridboard . . . 57
4.20 SteeringWheel Class Reference . . . 57
4.20.1 Member Function Documentation . . . 58
4.20.1.1 Start() . . . 58
4.20.1.2 Update() . . . 58
4.20.2 Member Data Documentation . . . 58
4.20.2.1 rotationMultiplier . . . 58
4.20.2.2 SteeringWheelHinge . . . 58
4.20.2.3 SteeringWheelInput . . . 59
4.20.2.4 TurningWheels . . . 59
4.21 vehicleComponent Class Reference . . . 59
4.21.1 Member Function Documentation . . . 60
4.21.1.1 loadGridboardComponentsFromVehicleComponent() . . . 60
4.21.1.2 processComponent() . . . 60
4.21.1.3 resetVehicleComponent() . . . 60
4.21.1.4 saveGridboardComponentsIntoVehicleComponent() . . . 60
4.21.1.5 Start() . . . 60

4.21.1.6 Update() . . . 60
4.21.2 Member Data Documentation . . . 61
4.21.2.1 defaultComponent . . . 61
4.21.2.2 defaultComponentIsSpawned . . . 61
4.21.2.3 defaultCompononentPosition . . . 61
4.21.2.4 defaultSlot . . . 61
4.21.2.5 defaultVehicleComponentPosition . . . 61
4.21.2.6 hornSound . . . 61
4.21.2.7 isCompleted . . . 61
4.21.2.8 isProcessed . . . 62
4.21.2.9 resetTimer . . . 62
4.21.2.10 spawnedComponent . . . 62
4.21.2.11 valueAfterCompletion . . . 62
4.21.2.12 VehicleComponent . . . 62
4.22 vehicleComponents Class Reference . . . 62
4.22.1 Member Function Documentation . . . 63
4.22.1.1 loadingComponents() . . . 63
4.22.1.2 loadSavedCircuitFromVehicleComponent() . . . 63
4.22.1.3 saveCircuitToVehicleComponent() . . . 63
4.22.1.4 Start() . . . 63
4.22.1.5 Update() . . . 63
4.22.2 Member Data Documentation . . . 64
4.22.2.1 componentsInGridboard . . . 64
4.23 vehicleEvaluation Class Reference . . . 64
4.23.1 Member Function Documentation . . . 64
4.23.1.1 activateGoKart() . . . 65
4.23.1.2 finishGoKart() . . . 65
4.23.1.3 processComponent() . . . 65
4.23.1.4 Start() . . . 65
4.23.1.5 Update() . . . 65
4.23.2 Member Data Documentation . . . 65
4.23.2.1 components . . . 65
4.23.2.2 lowerPlatform . . . 65
4.23.2.3 platform . . . 66
4.23.2.4 progress . . . 66
4.23.2.5 text . . . 66

5 File Documentation **67**

5.1 amongus.cs File Reference . . . 67
5.2 CircuitBuilder.cs File Reference . . . 67
5.3 ComponentController.cs File Reference . . . 67
5.3.1 Enumeration Type Documentation . . . 68

5.3.1.1 ComponentType . . . 68
5.3.1.2 OutputType . . . 68
5.3.1.3 WireType . . . 68
5.4 ComponentHandler.cs File Reference . . . 69
5.5 Dashboard.cs File Reference . . . 69
5.6 GoKartEnabler.cs File Reference . . . 69
5.7 GoKartScreenFader.cs File Reference . . . 69
5.8 GraphFast.cs File Reference . . . 69
5.9 GraphSample.cs File Reference . . . 70
5.10 gridboardController.cs File Reference . . . 70
5.11 gridboardPosition.cs File Reference . . . 70
5.12 HandsController.cs File Reference . . . 70
5.13 InsertController.cs File Reference . . . 70
5.14 InteractionController.cs File Reference . . . 70
5.15 ModelChanger.cs File Reference . . . 70
5.16 MovementController.cs File Reference . . . 71
5.17 PlayerInteractions.cs File Reference . . . 71
5.18 Scheme.cs File Reference . . . 71
5.19 SpeedLever.cs File Reference . . . 71
5.20 SpiceSharpController.cs File Reference . . . 71
5.21 SteeringWheel.cs File Reference . . . 71
5.22 vehicleComponent.cs File Reference . . . 71
5.22.1 Enumeration Type Documentation . . . 72
5.22.1.1 VehicleComponent . . . 72
5.23 vehicleComponents.cs File Reference . . . 72
5.24 vehicleEvaluation.cs File Reference . . . 72

Index **73**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

CircuitBuilder 9
MonoBehaviour
- amongus 7
- ComponentController 13
- ComponentHandler 19
- Dashboard 20
- GoKartEnabler 22
- GoKartScreenFader 24
- GraphFast 25
- gridboardController 29
- gridboardPosition 35
- HandsController 35
- InsertController 37
- InteractionController 39
- ModelChanger 41
- MovementController 45
- PlayerInteractions 47
- Scheme 48
- SpeedLever 51
- SpiceSharpController 52
- SteeringWheel 57
- vehicleComponent 59
- vehicleComponents 62
- vehicleEvaluation 64

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

amongus . . . 7
CircuitBuilder . . . 9
ComponentController . . . 13
ComponentHandler . . . 19
Dashboard . . . 20
GoKartEnabler . . . 22
GoKartScreenFader . . . 24
GraphFast . . . 25
gridboardController . . . 29
gridboardPosition . . . 35
HandsController . . . 35
InsertController . . . 37
InteractionController . . . 39
ModelChanger . . . 41
MovementController . . . 45
PlayerInteractions . . . 47
Scheme . . . 48
SpeedLever . . . 51
SpiceSharpController . . . 52
SteeringWheel . . . 57
vehicleComponent . . . 59
vehicleComponents . . . 62
vehicleEvaluation . . . 64

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

amongus.cs . . . 67
CircuitBuilder.cs . . . 67
ComponentController.cs . . . 67
ComponentHandler.cs . . . 69
Dashboard.cs . . . 69
GoKartEnabler.cs . . . 69
GoKartScreenFader.cs . . . 69
GraphFast.cs . . . 69
GraphSample.cs . . . 70
gridboardController.cs . . . 70
gridboardPosition.cs . . . 70
HandsController.cs . . . 70
InsertController.cs . . . 70
InteractionController.cs . . . 70
ModelChanger.cs . . . 70
MovementController.cs . . . 71
PlayerInteractions.cs . . . 71
Scheme.cs . . . 71
SpeedLever.cs . . . 71
SpiceSharpController.cs . . . 71
SteeringWheel.cs . . . 71
vehicleComponent.cs . . . 71
vehicleComponents.cs . . . 72
vehicleEvaluation.cs . . . 72

Chapter 4

Class Documentation

4.1 amongus Class Reference

Inheritance diagram for amongus:

Public Attributes

- GameObject monkaLaugh
- Transform[ ] spawnpoints

Private Member Functions

- void Start ()
- void Update ()
- void spawnAmongus ()

Private Attributes

- string[ ] hexColors = { "#c51111", "#132ed1", "#117f2d", "#ed54ba", "#ef7d0d", "#f5f557", "#3f474e", "#d6e0f0", "#6b2fbb", "#71491e", "#38fedc", "#50ef39" }
- GameObject kindasus
- int moveXorZ
- float ejectSpeed = 7f
- float ejectRotation = 200f

4.1.1 Member Function Documentation

4.1.1.1 spawnAmongus()

```
void amongus.spawnAmongus ( )  [private]
```

4.1.1.2 Start()

```
void amongus.Start ( )  [private]
```

4.1.1.3 Update()

```
void amongus.Update ( )  [private]
```

4.1.2 Member Data Documentation

4.1.2.1 ejectRotation

```
float amongus.ejectRotation = 200f  [private]
```

4.1.2.2 ejectSpeed

```
float amongus.ejectSpeed = 7f  [private]
```

4.1.2.3 hexColors

```
string [] amongus.hexColors = { "#c51111", "#132ed1", "#117f2d", "#ed54ba", "#ef7d0d", "#f5f557",
"#3f474e", "#d6e0f0", "#6b2fbb", "#71491e", "#38fedc", "#50ef39" }  [private]
```

4.1.2.4 kindasus

```
GameObject amongus.kindasus  [private]
```

4.1.2.5 monkaLaugh

```
GameObject amongus.monkaLaugh
```

4.1.2.6 moveXorZ

```
int amongus.moveXorZ  [private]
```

4.1.2.7 spawnpoints

```
Transform [] amongus.spawnpoints
```

The documentation for this class was generated from the following file:

- amongus.cs

4.2 CircuitBuilder Class Reference

Public Member Functions

- void TransformIntoSpiceSharpInput (string SpiceCode, List< List< string >> componentsOutput)
- void SpiceSharp_Viktor ()
- List< string > getSpiceSharpOutputs ()
- void changeAttribute (string componentName, double newValue)

Public Attributes

- double output
- double resistance
- Transient tran
- string spiceCode = ""
- string

Private Member Functions

- void runSimulation (string SpiceCode)
- double changeResistance (double value)

Private Attributes

- SpiceSharp.Components.Resistors.Parameters resistor = null
- SpiceSharp.Components.Inductors.Parameters inductor = null
- SpiceSharp.Components.MutualInductances.Parameters mutualInductance = null
- SpiceSharp.Components.CommonBehaviors.IndependentSourceParameters voltageSource = null
- SpiceSharp.Behaviors.ITemperatureBehavior tb = null
- Simulation simulation
- List< string > outputs = new List<string>()
- List< string > outputsResult = new List<string>()

4.2.1 Member Function Documentation

4.2.1.1 changeAttribute()

```
void CircuitBuilder.changeAttribute (
            string componentName,
            double newValue )
```

4.2.1.2 changeResistance()

```
double CircuitBuilder.changeResistance (
            double value )  [private]
```

4.2.1.3 getSpiceSharpOutputs()

```
List<string> CircuitBuilder.getSpiceSharpOutputs ( )
```

4.2.1.4 runSimulation()

```
void CircuitBuilder.runSimulation (
            string SpiceCode )  [private]
```

4.2.1.5 SpiceSharp_Viktor()

```
void CircuitBuilder.SpiceSharp_Viktor ( )
```

4.2.1.6 TransformIntoSpiceSharpInput()

```
void CircuitBuilder.TransformIntoSpiceSharpInput (
            string SpiceCode,
            List< List< string >> componentsOutput )
```

4.2.2 Member Data Documentation

4.2.2.1 inductor

```
SpiceSharp.Components.Inductors.Parameters CircuitBuilder.inductor = null  [private]
```

4.2.2.2 mutualInductance

```
SpiceSharp.Components.MutualInductances.Parameters CircuitBuilder.mutualInductance = null
[private]
```

4.2.2.3 output

```
double CircuitBuilder.output
```

4.2.2.4 outputs

```
List<string> CircuitBuilder.outputs = new List<string>()  [private]
```

4.2.2.5 outputsResult

```
List<string> CircuitBuilder.outputsResult = new List<string>()  [private]
```

4.2.2.6 resistance

```
double CircuitBuilder.resistance
```

4.2.2.7 resistor

```
SpiceSharp.Components.Resistors.Parameters CircuitBuilder.resistor = null  [private]
```

4.2.2.8 simulation

```
Simulation CircuitBuilder.simulation  [private]
```

4.2.2.9 spiceCode

```
string CircuitBuilder.spiceCode = ""
```

4.2.2.10 string

```
CircuitBuilder.string
```

4.2.2.11 tb

```
SpiceSharp.Behaviors.ITemperatureBehavior CircuitBuilder.tb = null  [private]
```

4.2.2.12 tran

```
Transient CircuitBuilder.tran
```

4.2.2.13 voltageSource

```
SpiceSharp.Components.CommonBehaviors.IndependentSourceParameters CircuitBuilder.voltageSource
= null  [private]
```

The documentation for this class was generated from the following file:

- CircuitBuilder.cs

4.3 ComponentController Class Reference

Inheritance diagram for ComponentController:

Public Member Functions

- void rotatePins (int count)
- string getDefaultNameTag ()

Public Attributes

- string[ ] nodesPositions
- string[ ] currentNodesPositions
- ComponentType ComponentType = ComponentType.Resistor
- string PositiveNode
- string NegativeNode
- string InductorName1
- string InductorName2
- string CollectorNode
- string BaseNode
- string EmitterNode
- string SubstrateNode
- string Model
- string Anode
- string Cathode
- string DrainNode
- string GateNode
- string SourceNode
- string BulkNode
- string PositiveControllingNode
- string NegativeControllingNode
- double Dc
- double CapacitanceValue
- double Inductance
- double Coupling
- double Resistance
- WireType WireType = WireType.deffinitelyNotAWire
- string Flow
- OutputType OutputType = OutputType.deffinitelyNotAnOutput
- Text NameTag
- GameObject DynamicGameObject
- string valueUnit

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- string defaultNameTag

4.3.1 Member Function Documentation

4.3.1.1 getDefaultNameTag()

```
string ComponentController.getDefaultNameTag ( )
```

4.3.1.2 rotatePins()

```
void ComponentController.rotatePins (
            int count )
```

4.3.1.3 Start()

```
void ComponentController.Start ( )  [private]
```

4.3.1.4 Update()

```
void ComponentController.Update ( )  [private]
```

4.3.2 Member Data Documentation

4.3.2.1 Anode

```
string ComponentController.Anode
```

4.3.2.2 BaseNode

```
string ComponentController.BaseNode
```

4.3.2.3 BulkNode

```
string ComponentController.BulkNode
```

4.3.2.4 CapacitanceValue

```
double ComponentController.CapacitanceValue
```

4.3.2.5 Cathode

```
string ComponentController.Cathode
```

4.3.2.6 CollectorNode

```
string ComponentController.CollectorNode
```

4.3.2.7 ComponentType

```
ComponentType ComponentController.ComponentType = ComponentType.Resistor
```

4.3.2.8 Coupling

```
double ComponentController.Coupling
```

4.3.2.9 currentNodesPositions

```
string [] ComponentController.currentNodesPositions
```

4.3.2.10 Dc

```
double ComponentController.Dc
```

4.3.2.11 defaultNameTag

```
string ComponentController.defaultNameTag  [private]
```

4.3.2.12 DrainNode

```
string ComponentController.DrainNode
```

4.3.2.13 DynamicGameObject

```
GameObject ComponentController.DynamicGameObject
```

4.3.2.14 EmitterNode

```
string ComponentController.EmitterNode
```

4.3.2.15 Flow

```
string ComponentController.Flow
```

4.3.2.16 GateNode

```
string ComponentController.GateNode
```

4.3.2.17 Inductance

```
double ComponentController.Inductance
```

4.3.2.18 InductorName1

```
string ComponentController.InductorName1
```

4.3.2.19 InductorName2

```
string ComponentController.InductorName2
```

4.3.2.20 Model

```
string ComponentController.Model
```

4.3.2.21 NameTag

```
Text ComponentController.NameTag
```

4.3.2.22 NegativeControllingNode

```
string ComponentController.NegativeControllingNode
```

4.3.2.23 NegativeNode

```
string ComponentController.NegativeNode
```

4.3.2.24 nodesPositions

```
string [] ComponentController.nodesPositions
```

4.3.2.25 OutputType

```
OutputType ComponentController.OutputType = OutputType.deffinitelyNotAnOutput
```

4.3.2.26 PositiveControllingNode

```
string ComponentController.PositiveControllingNode
```

4.3.2.27 PositiveNode

```
string ComponentController.PositiveNode
```

4.3.2.28 Resistance

```
double ComponentController.Resistance
```

4.3.2.29 SourceNode

```
string ComponentController.SourceNode
```

4.3.2.30 SubstrateNode

```
string ComponentController.SubstrateNode
```

4.3.2.31 valueUnit

```
string ComponentController.valueUnit
```

4.3.2.32 WireType

```
WireType ComponentController.WireType = WireType.deffinitelyNotAWire
```

The documentation for this class was generated from the following file:

- ComponentController.cs

4.4 ComponentHandler Class Reference

Inheritance diagram for ComponentHandler:

Public Attributes

- GameObject parentContainer
- bool isInserted
- InsertController insertedItemIC
- GameObject body

Private Member Functions

- void Start ()
- void Update ()
- void OnTriggerEnter (Collider other)

4.4.1 Member Function Documentation

4.4.1.1 OnTriggerEnter()

```
void ComponentHandler.OnTriggerEnter (
            Collider other )  [private]
```

4.4.1.2 Start()

```
void ComponentHandler.Start ( )  [private]
```

4.4.1.3 Update()

```
void ComponentHandler.Update ( )  [private]
```

4.4.2 Member Data Documentation

4.4.2.1 body

```
GameObject ComponentHandler.body
```

4.4.2.2 insertedItemIC

```
InsertController ComponentHandler.insertedItemIC
```

4.4.2.3 isInserted

```
bool ComponentHandler.isInserted
```

4.4.2.4 parentContainer

```
GameObject ComponentHandler.parentContainer
```

The documentation for this class was generated from the following file:

- ComponentHandler.cs

4.5 Dashboard Class Reference

Inheritance diagram for Dashboard:

Public Member Functions

- void lightsOnOff ()
- void hornOn ()
- void hornOff ()
- void setUpMotor ()

Public Attributes

- vehicleComponent horn
- vehicleComponent engine
- vehicleComponent lights
- MeshRenderer lightbulb

Private Member Functions

- void Start ()
- void Update ()

4.5.1 Member Function Documentation

4.5.1.1 hornOff()

```
void Dashboard.hornOff ( )
```

4.5.1.2 hornOn()

```
void Dashboard.hornOn ( )
```

4.5.1.3 lightsOnOff()

```
void Dashboard.lightsOnOff ( )
```

4.5.1.4 setUpMotor()

```
void Dashboard.setUpMotor ( )
```

4.5.1.5 Start()

```
void Dashboard.Start ( )  [private]
```

4.5.1.6 Update()

```
void Dashboard.Update ( )  [private]
```

4.5.2 Member Data Documentation

4.5.2.1 engine

```
vehicleComponent Dashboard.engine
```

4.5.2.2 horn

```
vehicleComponent Dashboard.horn
```

4.5.2.3 lightbulb

```
MeshRenderer Dashboard.lightbulb
```

4.5.2.4 lights

```
vehicleComponent Dashboard.lights
```

The documentation for this class was generated from the following file:

- Dashboard.cs

4.6 GoKartEnabler Class Reference

Inheritance diagram for GoKartEnabler:

Public Attributes

- GameObject PlayerSittingPosition
- GameObject Player
- CharacterController PlayerController
- GameObject screenFader

Private Member Functions

- void Start ()
- void Update ()
- void positionPlayerToKart ()
- void removePlayerFromKart ()

Private Attributes

- KartGame.KartSystems.ArcadeKart AK

4.6.1 Member Function Documentation

4.6.1.1 positionPlayerToKart()

```
void GoKartEnabler.positionPlayerToKart ( )  [private]
```

4.6.1.2 removePlayerFromKart()

```
void GoKartEnabler.removePlayerFromKart ( )  [private]
```

4.6.1.3 Start()

```
void GoKartEnabler.Start ( )  [private]
```

4.6.1.4 Update()

```
void GoKartEnabler.Update ( )  [private]
```

4.6.2 Member Data Documentation

4.6.2.1 AK

```
KartGame.KartSystems.ArcadeKart GoKartEnabler.AK  [private]
```

4.6.2.2 Player

```
GameObject GoKartEnabler.Player
```

4.6.2.3 PlayerController

```
CharacterController GoKartEnabler.PlayerController
```

4.6.2.4 PlayerSittingPosition

```
GameObject GoKartEnabler.PlayerSittingPosition
```

4.6.2.5 screenFader

```
GameObject GoKartEnabler.screenFader
```

The documentation for this class was generated from the following file:

- GoKartEnabler.cs

4.7 GoKartScreenFader Class Reference

Inheritance diagram for GoKartScreenFader:

Public Member Functions

- void fadeOutJustInCaseSmileyFace ()

Public Attributes

- BNG.ScreenFader screenFader

Private Member Functions

- void OnTriggerEnter (Collider other)
- void OnTriggerExit (Collider other)

4.7.1 Member Function Documentation

4.7.1.1 fadeOutJustInCaseSmileyFace()

```
void GoKartScreenFader.fadeOutJustInCaseSmileyFace ( )
```

4.7.1.2 OnTriggerEnter()

```
void GoKartScreenFader.OnTriggerEnter (
            Collider other )  [private]
```

4.7.1.3 OnTriggerExit()

```
void GoKartScreenFader.OnTriggerExit (
            Collider other )  [private]
```

4.7.2 Member Data Documentation

4.7.2.1 screenFader

```
BNG.ScreenFader GoKartScreenFader.screenFader
```

The documentation for this class was generated from the following file:

- GoKartScreenFader.cs

4.8 GraphFast Class Reference

Inheritance diagram for GraphFast:

Public Member Functions

- void createGraph (List< string > outputs)
- void resetGraph ()
- void showGraph ()
- void hideGraph ()

Public Attributes

- GameObject chart
- GameObject lineRenderer

Private Member Functions

- void Start ()
- void Update ()
- IEnumerator finishGraphsLine (LineRenderer lr)

Private Attributes

- bool graphCreated = false
- bool generatePoints = false
- float TimerX = 0
- List< float > numericOutputs = new List<float>()
- float graphScalerX = 200
- float graphScalerY = 15
- List< LineRenderer > lineRenderers = new List<LineRenderer>()
- int positionIndex = 0
- Color[ ] colors = { Color.blue, Color.magenta, Color.red, Color.yellow, Color.green, Color.cyan, Color.grey }
- List< Color > chosenColors = new List<Color>()
- bool finishingPath

4.8.1 Member Function Documentation

4.8.1.1 createGraph()

```
void GraphFast.createGraph (
            List< string > outputs )
```

4.8.1.2 finishGraphsLine()

```
IEnumerator GraphFast.finishGraphsLine (
            LineRenderer lr )  [private]
```

4.8.1.3 hideGraph()

```
void GraphFast.hideGraph ( )
```

4.8.1.4 resetGraph()

```
void GraphFast.resetGraph ( )
```

4.8.1.5 showGraph()

```
void GraphFast.showGraph ( )
```

4.8.1.6 Start()

```
void GraphFast.Start ( )  [private]
```

4.8.1.7 Update()

```
void GraphFast.Update ( )  [private]
```

4.8.2 Member Data Documentation

4.8.2.1 chart

```
GameObject GraphFast.chart
```

4.8.2.2 chosenColors

```
List<Color> GraphFast.chosenColors = new List<Color>()  [private]
```

4.8.2.3 colors

```
Color [] GraphFast.colors = { Color.blue, Color.magenta, Color.red, Color.yellow, Color.green,
Color.cyan, Color.grey }  [private]
```

4.8.2.4 finishingPath

```
bool GraphFast.finishingPath  [private]
```

4.8.2.5 generatePoints

```
bool GraphFast.generatePoints = false  [private]
```

4.8.2.6 graphCreated

```
bool GraphFast.graphCreated = false  [private]
```

4.8.2.7 graphScalerX

```
float GraphFast.graphScalerX = 200  [private]
```

4.8.2.8 graphScalerY

```
float GraphFast.graphScalerY = 15  [private]
```

4.8.2.9 lineRenderer

```
GameObject GraphFast.lineRenderer
```

4.8.2.10 lineRenderers

```
List<LineRenderer> GraphFast.lineRenderers = new List<LineRenderer>()  [private]
```

4.8.2.11 numericOutputs

```
List<float> GraphFast.numericOutputs = new List<float>()  [private]
```

4.8.2.12 positionIndex

```
int GraphFast.positionIndex = 0  [private]
```

4.8.2.13 TimerX

```
float GraphFast.TimerX = 0  [private]
```

The documentation for this class was generated from the following file:

- GraphFast.cs

4.9 gridboardController Class Reference

Inheritance diagram for gridboardController:

Public Member Functions

- void handleComponentInGridboard (GameObject component, GameObject slot, string InsertOrExtract)
- void createSpiceSharpSimulation ()
- void removeAllComponentsFromGridboard ()

Public Attributes

- int rows
- int columns
- GameObject[ ] slots
- GameObject tuningStation

Private Member Functions

- void Start ()
- void Update ()
- void setUpChildCoordinates ()
- void checkForWireConnection (GameObject component, GameObject slot, int previousComponentDirection, List< KeyValuePair< ComponentController, int >> connectionIssuer)
- List< KeyValuePair< ComponentController, int > > finitializeConnection (List< KeyValuePair< ComponentController, int >> connectionIssuer, string InsertOrExtract)
- void connectComponents (List< KeyValuePair< ComponentController, int >> connectionIssuer)
- IEnumerator disconnectComponents (List< KeyValuePair< ComponentController, int >> connectionIssuer)
- void updateComponentAttributes (KeyValuePair< ComponentController, int > component)
- void createSpiceSharpInput ()
- IEnumerator componentsRemovalCoroutine ()

Private Attributes

- int connectionIndexIdentifier = 0
- List< ComponentController > connectedComponents = new List<ComponentController>()
- List< GameObject > connectionMultiWires = new List<GameObject>()
- List< KeyValuePair< ComponentController, int > > checkedWires = new List<KeyValuePair<ComponentController, int>>()
- SpiceSharpController SSC
- string SpiceCode = ""
- List<(string SpiceCodeName, double attribute, string unit)> componentsDynamicAttribute = new List<(string SpiceCodeName, double attribute, string unit)>()
- List< ComponentController > componentsCircuit = new List<ComponentController>()
- List< List< string > > componentsOutput = new List<List<string>>()
- List< GameObject > gameobjectsOutput = new List<GameObject>()
- char[ ] separatorsForComponentAttribute = new char[ ] { ' ', '(', ')' }
- ModelChanger MC
- List<(string posNode, string negNode)> capacitorsNodesForIC = new List<(string posNode, string negNode)>()
- bool deleteComponents

4.9.1 Member Function Documentation

4.9.1.1 checkForWireConnection()

```
void gridboardController.checkForWireConnection (
            GameObject component,
            GameObject slot,
            int previousComponentDirection,
            List< KeyValuePair< ComponentController, int >> connectionIssuer )  [private]
```

4.9.1.2 componentsRemovalCoroutine()

```
IEnumerator gridboardController.componentsRemovalCoroutine ( )  [private]
```

4.9.1.3 connectComponents()

```
void gridboardController.connectComponents (
            List< KeyValuePair< ComponentController, int >> connectionIssuer )  [private]
```

4.9.1.4 createSpiceSharpInput()

```
void gridboardController.createSpiceSharpInput ( )  [private]
```

4.9.1.5 createSpiceSharpSimulation()

```
void gridboardController.createSpiceSharpSimulation ( )
```

4.9.1.6 disconnectComponents()

```
IEnumerator gridboardController.disconnectComponents (
            List< KeyValuePair< ComponentController, int >> connectionIssuer )  [private]
```

4.9.1.7 finitializeConnection()

```
List<KeyValuePair<ComponentController, int> > gridboardController.finitializeConnection (
            List< KeyValuePair< ComponentController, int >> connectionIssuer,
            string InsertOrExtract )  [private]
```

4.9.1.8 handleComponentInGridboard()

```
void gridboardController.handleComponentInGridboard (
            GameObject component,
            GameObject slot,
            string InsertOrExtract )
```

4.9.1.9 removeAllComponentsFromGridboard()

```
void gridboardController.removeAllComponentsFromGridboard ( )
```

4.9.1.10 setUpChildCoordinates()

```
void gridboardController.setUpChildCoordinates ( )  [private]
```

4.9.1.11 Start()

```
void gridboardController.Start ( )  [private]
```

4.9.1.12 Update()

```
void gridboardController.Update ( )  [private]
```

4.9.1.13 updateComponentAttributes()

```
void gridboardController.updateComponentAttributes (
            KeyValuePair< ComponentController, int > component )  [private]
```

4.9.2 Member Data Documentation

4.9.2.1 capacitorsNodesForIC

```
List<(string posNode, string negNode)> gridboardController.capacitorsNodesForIC = new List<(string
posNode, string negNode)>()  [private]
```

4.9.2.2 checkedWires

```
List<KeyValuePair<ComponentController, int> > gridboardController.checkedWires = new List<Key↩
ValuePair<ComponentController, int>>()  [private]
```

4.9.2.3 columns

```
int gridboardController.columns
```

4.9.2.4 componentsCircuit

```
List<ComponentController> gridboardController.componentsCircuit = new List<ComponentController>()
[private]
```

4.9.2.5 componentsDynamicAttribute

```
List<(string SpiceCodeName, double attribute, string unit)> gridboardController.components↩
DynamicAttribute = new List<(string SpiceCodeName, double attribute, string unit)>()  [private]
```

4.9.2.6 componentsOutput

```
List<List<string> > gridboardController.componentsOutput = new List<List<string>>()  [private]
```

4.9.2.7 connectedComponents

```
List<ComponentController> gridboardController.connectedComponents = new List<ComponentController>()
[private]
```

4.9.2.8 connectionIndexIdentifier

```
int gridboardController.connectionIndexIdentifier = 0  [private]
```

4.9.2.9 connectionMultiWires

```
List<GameObject> gridboardController.connectionMultiWires = new List<GameObject>()  [private]
```

4.9.2.10 deleteComponents

```
bool gridboardController.deleteComponents  [private]
```

4.9.2.11 gameobjectsOutput

```
List<GameObject> gridboardController.gameobjectsOutput = new List<GameObject>()  [private]
```

4.9.2.12 MC

```
ModelChanger gridboardController.MC  [private]
```

4.9.2.13 rows

```
int gridboardController.rows
```

4.9.2.14 separatorsForComponentAttribute

```
char [] gridboardController.separatorsForComponentAttribute = new char[] { ' ', '(', ')' }
[private]
```

4.9.2.15 slots

```
GameObject [] gridboardController.slots
```

4.9.2.16 SpiceCode

```
string gridboardController.SpiceCode = ""  [private]
```

4.9.2.17 SSC

```
SpiceSharpController gridboardController.SSC  [private]
```

4.9.2.18 tuningStation

```
GameObject gridboardController.tuningStation
```

The documentation for this class was generated from the following file:

- gridboardController.cs

4.10 gridboardPosition Class Reference

Inheritance diagram for gridboardPosition:

Public Attributes

- Vector2 positionInGridboard

4.10.1 Member Data Documentation

4.10.1.1 positionInGridboard

```
Vector2 gridboardPosition.positionInGridboard
```

The documentation for this class was generated from the following file:

- gridboardPosition.cs

4.11 HandsController Class Reference

Inheritance diagram for HandsController:

Public Member Functions

- void pickupItem (GameObject item)
- GameObject getGrabbedItem ()
- void removeItem ()
- void dropItem ()

Public Attributes

- bool isGrabbed = false

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- GameObject currentPickedItem

4.11.1 Member Function Documentation

4.11.1.1 dropItem()

```
void HandsController.dropItem ( )
```

4.11.1.2 getGrabbedItem()

```
GameObject HandsController.getGrabbedItem ( )
```

4.11.1.3 pickupItem()

```
void HandsController.pickupItem (
            GameObject item )
```

4.11.1.4 removeItem()

```
void HandsController.removeItem ( )
```

4.11.1.5 Start()

```
void HandsController.Start ( )  [private]
```

4.11.1.6 Update()

```
void HandsController.Update ( )  [private]
```

4.11.2 Member Data Documentation

4.11.2.1 currentPickedItem

```
GameObject HandsController.currentPickedItem  [private]
```

4.11.2.2 isGrabbed

```
bool HandsController.isGrabbed = false
```

The documentation for this class was generated from the following file:

- HandsController.cs

4.12 InsertController Class Reference

Inheritance diagram for InsertController:

Public Member Functions

- void insertItem (HandsController HC)
- void snapItem (GameObject item)
- void pullItem ()

Public Attributes

- bool isUsed = false

Private Member Functions

- void Start ()
- void Update ()
- void saveItemToGrid ()

Private Attributes

- GameObject insertedItem

4.12.1 Member Function Documentation

4.12.1.1 insertItem()

```
void InsertController.insertItem (
            HandsController HC )
```

4.12.1.2 pullItem()

```
void InsertController.pullItem ( )
```

4.12.1.3 saveItemToGrid()

```
void InsertController.saveItemToGrid ( )  [private]
```

4.12.1.4 snapItem()

```
void InsertController.snapItem (
            GameObject item )
```

4.12.1.5 Start()

```
void InsertController.Start ( )  [private]
```

4.12.1.6 Update()

```
void InsertController.Update ( )  [private]
```

4.12.2 Member Data Documentation

4.12.2.1 insertedItem

```
GameObject InsertController.insertedItem  [private]
```

4.12.2.2 isUsed

```
bool InsertController.isUsed = false
```

The documentation for this class was generated from the following file:

- InsertController.cs

4.13 InteractionController Class Reference

Inheritance diagram for InteractionController:

Public Attributes

- Camera playerCamera
- GameObject Hands
- float pickupRange = 2
- Material outlinedMaterial
- Material invisibleMaterial

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- HandsController HC
- GameObject simulatedItem
- GameObject highlightedItem

4.13.1 Member Function Documentation

4.13.1.1 Start()

```
void InteractionController.Start ( )  [private]
```

4.13.1.2 Update()

```
void InteractionController.Update ( )  [private]
```

4.13.2 Member Data Documentation

4.13.2.1 Hands

```
GameObject InteractionController.Hands
```

4.13.2.2 HC

```
HandsController InteractionController.HC  [private]
```

4.13.2.3 highlightedItem

```
GameObject InteractionController.highlightedItem  [private]
```

4.13.2.4 invisibleMaterial

```
Material InteractionController.invisibleMaterial
```

4.13.2.5 outlinedMaterial

```
Material InteractionController.outlinedMaterial
```

4.13.2.6 pickupRange

```
float InteractionController.pickupRange = 2
```

4.13.2.7 playerCamera

```
Camera InteractionController.playerCamera
```

4.13.2.8 simulatedItem

```
GameObject InteractionController.simulatedItem  [private]
```

The documentation for this class was generated from the following file:

- InteractionController.cs

4.14 ModelChanger Class Reference

Inheritance diagram for ModelChanger:

Public Member Functions

- void createModelsFromClipboard ()
- string getModelSpecification (string modelName)
- string getModelComponent (string modelName)
- void changeModel (int index)
- void insertModelIntoTuningStation (GameObject component)
- void extractModelFromTuningStation ()
- void changeComponentsValue (float newValue)
- void changeComponentsUnit (int index)

Public Attributes

- string models
- Dropdown dropdown
- GameObject valueChanger
- Dropdown unit

Private Member Functions

- void Start ()
- void Update ()
- void createModels (string input)
- void initializeValueChanger ()

Private Attributes

- List<(string component, string modelName, string specification)> customModels = new List<(string component, string modelName, string specification)>()
- ComponentController insertedCC

4.14.1 Member Function Documentation

4.14.1.1 changeComponentsUnit()

```
void ModelChanger.changeComponentsUnit (
            int index )
```

4.14.1.2 changeComponentsValue()

```
void ModelChanger.changeComponentsValue (
            float newValue )
```

4.14.1.3 changeModel()

```
void ModelChanger.changeModel (
            int index )
```

4.14.1.4 createModels()

```
void ModelChanger.createModels (
            string input )  [private]
```

4.14.1.5 createModelsFromClipboard()

```
void ModelChanger.createModelsFromClipboard ( )
```

4.14.1.6 extractModelFromTuningStation()

```
void ModelChanger.extractModelFromTuningStation ( )
```

4.14.1.7 getModelComponent()

```
string ModelChanger.getModelComponent (
            string modelName )
```

4.14.1.8 getModelSpecification()

```
string ModelChanger.getModelSpecification (
            string modelName )
```

4.14.1.9 initializeValueChanger()

```
void ModelChanger.initializeValueChanger ( )  [private]
```

4.14.1.10 insertModelIntoTuningStation()

```
void ModelChanger.insertModelIntoTuningStation (
            GameObject component )
```

4.14.1.11 Start()

```
void ModelChanger.Start ( )  [private]
```

4.14.1.12 Update()

```
void ModelChanger.Update ( )  [private]
```

4.14.2 Member Data Documentation

4.14.2.1 customModels

```
List<(string component, string modelName, string specification)> ModelChanger.customModels =
new List<(string component, string modelName, string specification)>()  [private]
```

4.14.2.2 dropdown

```
Dropdown ModelChanger.dropdown
```

4.14.2.3 insertedCC

```
ComponentController ModelChanger.insertedCC  [private]
```

4.14.2.4 models

```
string ModelChanger.models
```

4.14.2.5 unit

```
Dropdown ModelChanger.unit
```

4.14.2.6 valueChanger

```
GameObject ModelChanger.valueChanger
```

The documentation for this class was generated from the following file:

- ModelChanger.cs

4.15 MovementController Class Reference

Inheritance diagram for MovementController:

Public Attributes

- float movementSpeed = 5f
- float jumpSpeed = 8f
- float gravity = 20f
- Camera playerCamera
- bool lelel = true

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- CharacterController CC
- Vector3 moveDirection = Vector3.zero
- float rotationX = 0f
- float lookXLimit = 90f
- float lookSpeed = 2f

4.15.1 Member Function Documentation

4.15.1.1 Start()

```
void MovementController.Start ( )  [private]
```

4.15.1.2 Update()

```
void MovementController.Update ( )  [private]
```

4.15.2 Member Data Documentation

4.15.2.1 CC

```
CharacterController MovementController.CC  [private]
```

4.15.2.2 gravity

```
float MovementController.gravity = 20f
```

4.15.2.3 jumpSpeed

```
float MovementController.jumpSpeed = 8f
```

4.15.2.4 lelel

```
bool MovementController.lelel = true
```

4.15.2.5 lookSpeed

```
float MovementController.lookSpeed = 2f  [private]
```

4.15.2.6 lookXLimit

```
float MovementController.lookXLimit = 90f  [private]
```

4.15.2.7 moveDirection

```
Vector3 MovementController.moveDirection = Vector3.zero  [private]
```

4.15.2.8 movementSpeed

```
float MovementController.movementSpeed = 5f
```

4.15.2.9 playerCamera

```
Camera MovementController.playerCamera
```

4.15.2.10 rotationX

```
float MovementController.rotationX = 0f  [private]
```

The documentation for this class was generated from the following file:

- MovementController.cs

4.16 PlayerInteractions Class Reference

Inheritance diagram for PlayerInteractions:

Public Attributes

- bool isInKart

Private Member Functions

- void Start ()
- void Update ()

4.16.1 Member Function Documentation

4.16.1.1 Start()

```
void PlayerInteractions.Start ( )  [private]
```

4.16.1.2 Update()

```
void PlayerInteractions.Update ( )  [private]
```

4.16.2 Member Data Documentation

4.16.2.1 isInKart

```
bool PlayerInteractions.isInKart
```

The documentation for this class was generated from the following file:

- PlayerInteractions.cs

4.17 Scheme Class Reference

Inheritance diagram for Scheme:

Public Member Functions

- void generateSchemeImages (int rows, int columns)
- void updateSchemeImage (GameObject insertedComponent, int index)
- void resetSchemeImage (int index)

Public Attributes

- GameObject image
- List< GameObject > images = new List<GameObject>()
- Sprite[ ] symbols
- ModelChanger modelChanger

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- Sprite defaultSprite
- float increaseX = 100
- float increaseY = 100
- string[ ] components

4.17.1 Member Function Documentation

4.17.1.1 generateSchemeImages()

```
void Scheme.generateSchemeImages (
            int rows,
            int columns )
```

4.17.1.2 resetSchemeImage()

```
void Scheme.resetSchemeImage (
            int index )
```

4.17.1.3 Start()

```
void Scheme.Start ( )  [private]
```

4.17.1.4 Update()

```
void Scheme.Update ( )  [private]
```

4.17.1.5 updateSchemeImage()

```
void Scheme.updateSchemeImage (
            GameObject insertedComponent,
            int index )
```

4.17.2 Member Data Documentation

4.17.2.1 components

```
string [] Scheme.components  [private]
```

Initial value:

```
= { "Battery", "Capacitor", "Coil", "Diode", "Ground",
        "NJFET", "PJFET", "Lightbulb", "MosfetNMOS", "MosfetPMOS", "Motor", "Oscillator", "Resistor",
       "Speaker",
        "Switch", "TransistorNPN", "TransistorPNP", "Meter", "Bridge", "Crossing3", "Crossing", "Edge",
       "Line"}
```

4.17.2.2 defaultSprite

```
Sprite Scheme.defaultSprite  [private]
```

4.17.2.3 image

```
GameObject Scheme.image
```

4.17.2.4 images

```
List<GameObject> Scheme.images = new List<GameObject>()
```

4.17.2.5 increaseX

```
float Scheme.increaseX = 100  [private]
```

4.17.2.6 increaseY

```
float Scheme.increaseY = 100  [private]
```

4.17.2.7 modelChanger

```
ModelChanger Scheme.modelChanger
```

4.17.2.8 symbols

```
Sprite [] Scheme.symbols
```

The documentation for this class was generated from the following file:

- Scheme.cs

4.18 SpeedLever Class Reference

Inheritance diagram for SpeedLever:

Public Attributes

- GameObject SpeedLeverHinge
- float SpeedLeverInput

Private Member Functions

- void Start ()
- void Update ()

4.18.1 Member Function Documentation

4.18.1.1 Start()

```
void SpeedLever.Start ( )  [private]
```

4.18.1.2 Update()

```
void SpeedLever.Update ( )  [private]
```

4.18.2 Member Data Documentation

4.18.2.1 SpeedLeverHinge

```
GameObject SpeedLever.SpeedLeverHinge
```

4.18.2.2 SpeedLeverInput

```
float SpeedLever.SpeedLeverInput
```

The documentation for this class was generated from the following file:

- SpeedLever.cs

4.19 SpiceSharpController Class Reference

Inheritance diagram for SpiceSharpController:

Public Member Functions

- void createSpiceSharpInstance (string SpiceCode, List< List< string >> componentsOutput, List<(string SpiceCodeName, double attribute, string valueUnit)> componentsDynamicAttribute, List< GameObject > gameobjectsOutput)
- void showOrHideSpiceCode ()
- void showOrHideDynamicComponents ()
- void closeSpiceSharpInstance ()

Public Attributes

- Text simulationOutput
- Text spiceCode
- GameObject spiceCodeButton
- GameObject dynamicComponentsButton
- GameObject startSimulationButton
- GameObject removeComponentsButton
- GameObject stopSimulationButton
- GameObject DynamicAttributes
- GameObject DynamicComponent
- vehicleComponent vehicleComponentInGridboard

Private Member Functions

- void Start ()
- void Update ()
- void initializeDynamicComponentsUI ()
- void changeDynamicAttribute (int index, double value)
- void validateOutput (int index, string output)
- void resetDynamicOutputs ()

Private Attributes

- List<(string SpiceCodeName, double attribute, string unit)> DynamicComponents = new List<(string SpiceCodeName, double attribute, string unit)>()
- List< GameObject > DynamicComponentsSliders = new List<GameObject>()
- Thread SpiceSharpThread
- CircuitBuilder CB
- GraphFast graph
- int defaultVehicleComponentIndex = -1
- List< List< string > > ComponentsOutput = new List<List<string>>()
- List< GameObject > GameObjectsOutput = new List<GameObject>()
- float speakerCurrentPeak = 0
- bool increasing
- float speakerFrequencyTimer = 0
- bool isZero

4.19.1 Member Function Documentation

4.19.1.1 changeDynamicAttribute()

```
void SpiceSharpController.changeDynamicAttribute (
            int index,
            double value )  [private]
```

4.19.1.2 closeSpiceSharpInstance()

```
void SpiceSharpController.closeSpiceSharpInstance ( )
```

4.19.1.3 createSpiceSharpInstance()

```
void SpiceSharpController.createSpiceSharpInstance (
            string SpiceCode,
            List< List< string >> componentsOutput,
            List<(string SpiceCodeName, double attribute, string valueUnit)> componentsDynamicAttribute,
            List< GameObject > gameobjectsOutput )
```

4.19.1.4 initializeDynamicComponentsUI()

```
void SpiceSharpController.initializeDynamicComponentsUI ( )  [private]
```

4.19.1.5 resetDynamicOutputs()

```
void SpiceSharpController.resetDynamicOutputs ( )  [private]
```

4.19.1.6 showOrHideDynamicComponents()

```
void SpiceSharpController.showOrHideDynamicComponents ( )
```

4.19.1.7 showOrHideSpiceCode()

```
void SpiceSharpController.showOrHideSpiceCode ( )
```

4.19.1.8 Start()

```
void SpiceSharpController.Start ( )  [private]
```

4.19.1.9 Update()

```
void SpiceSharpController.Update ( )  [private]
```

4.19.1.10 validateOutput()

```
void SpiceSharpController.validateOutput (
            int index,
            string output )  [private]
```

4.19.2 Member Data Documentation

4.19.2.1 CB

```
CircuitBuilder SpiceSharpController.CB  [private]
```

4.19.2.2 ComponentsOutput

```
List<List<string> > SpiceSharpController.ComponentsOutput = new List<List<string>>()  [private]
```

4.19.2.3 defaultVehicleComponentIndex

```
int SpiceSharpController.defaultVehicleComponentIndex = -1  [private]
```

4.19.2.4 DynamicAttributes

```
GameObject SpiceSharpController.DynamicAttributes
```

4.19.2.5 DynamicComponent

```
GameObject SpiceSharpController.DynamicComponent
```

4.19.2.6 DynamicComponents

```
List<(string SpiceCodeName, double attribute, string unit)> SpiceSharpController.Dynamic↩
Components = new List<(string SpiceCodeName, double attribute, string unit)>()  [private]
```

4.19.2.7 dynamicComponentsButton

```
GameObject SpiceSharpController.dynamicComponentsButton
```

4.19.2.8 DynamicComponentsSliders

```
List<GameObject> SpiceSharpController.DynamicComponentsSliders = new List<GameObject>()  [private]
```

4.19.2.9 GameObjectsOutput

```
List<GameObject> SpiceSharpController.GameObjectsOutput = new List<GameObject>()  [private]
```

4.19.2.10 graph

```
GraphFast SpiceSharpController.graph  [private]
```

4.19.2.11 increasing

```
bool SpiceSharpController.increasing  [private]
```

4.19.2.12 isZero

```
bool SpiceSharpController.isZero  [private]
```

4.19.2.13 removeComponentsButton

```
GameObject SpiceSharpController.removeComponentsButton
```

4.19.2.14 simulationOutput

```
Text SpiceSharpController.simulationOutput
```

4.19.2.15 speakerCurrentPeak

```
float SpiceSharpController.speakerCurrentPeak = 0  [private]
```

4.19.2.16 speakerFrequencyTimer

```
float SpiceSharpController.speakerFrequencyTimer = 0  [private]
```

4.19.2.17 spiceCode

```
Text SpiceSharpController.spiceCode
```

4.19.2.18 spiceCodeButton

```
GameObject SpiceSharpController.spiceCodeButton
```

4.19.2.19 SpiceSharpThread

```
Thread SpiceSharpController.SpiceSharpThread  [private]
```

4.19.2.20 startSimulationButton

```
GameObject SpiceSharpController.startSimulationButton
```

4.19.2.21 stopSimulationButton

```
GameObject SpiceSharpController.stopSimulationButton
```

4.19.2.22 vehicleComponentInGridboard

```
vehicleComponent SpiceSharpController.vehicleComponentInGridboard
```

The documentation for this class was generated from the following file:

- SpiceSharpController.cs

4.20 SteeringWheel Class Reference

Inheritance diagram for SteeringWheel:

Public Attributes

- GameObject SteeringWheelHinge
- GameObject[ ] TurningWheels
- float SteeringWheelInput

Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- float rotationMultiplier = 0.25f

4.20.1 Member Function Documentation

4.20.1.1 Start()

```
void SteeringWheel.Start ( )  [private]
```

4.20.1.2 Update()

```
void SteeringWheel.Update ( )  [private]
```

4.20.2 Member Data Documentation

4.20.2.1 rotationMultiplier

```
float SteeringWheel.rotationMultiplier = 0.25f  [private]
```

4.20.2.2 SteeringWheelHinge

```
GameObject SteeringWheel.SteeringWheelHinge
```

4.20.2.3 SteeringWheelInput

```
float SteeringWheel.SteeringWheelInput
```

4.20.2.4 TurningWheels

```
GameObject [] SteeringWheel.TurningWheels
```

The documentation for this class was generated from the following file:

- SteeringWheel.cs

4.21 vehicleComponent Class Reference

Inheritance diagram for vehicleComponent:

Public Member Functions

- void saveGridboardComponentsIntoVehicleComponent (List< GameObject > gridboardComponents)
- int loadGridboardComponentsFromVehicleComponent ()

Public Attributes

- VehicleComponent VehicleComponent = VehicleComponent.Motor
- GameObject defaultComponent
- GameObject spawnedComponent
- bool defaultComponentIsSpawned
- Transform defaultCompononentPosition
- GameObject defaultSlot
- bool isCompleted
- float valueAfterCompletion = 0
- AudioSource hornSound

Private Member Functions

- void Start ()
- void Update ()
- void processComponent (string component)
- IEnumerator resetVehicleComponent ()

Private Attributes

- float resetTimer = 0
- Vector3 defaultVehicleComponentPosition
- bool isProcessed

4.21.1 Member Function Documentation

4.21.1.1 loadGridboardComponentsFromVehicleComponent()

```
int vehicleComponent.loadGridboardComponentsFromVehicleComponent ( )
```

4.21.1.2 processComponent()

```
void vehicleComponent.processComponent (
            string component )  [private]
```

4.21.1.3 resetVehicleComponent()

```
IEnumerator vehicleComponent.resetVehicleComponent ( )  [private]
```

4.21.1.4 saveGridboardComponentsIntoVehicleComponent()

```
void vehicleComponent.saveGridboardComponentsIntoVehicleComponent (
            List< GameObject > gridboardComponents )
```

4.21.1.5 Start()

```
void vehicleComponent.Start ( )  [private]
```

4.21.1.6 Update()

```
void vehicleComponent.Update ( )  [private]
```

4.21.2 Member Data Documentation

4.21.2.1 defaultComponent

```
GameObject vehicleComponent.defaultComponent
```

4.21.2.2 defaultComponentIsSpawned

```
bool vehicleComponent.defaultComponentIsSpawned
```

4.21.2.3 defaultCompononentPosition

```
Transform vehicleComponent.defaultCompononentPosition
```

4.21.2.4 defaultSlot

```
GameObject vehicleComponent.defaultSlot
```

4.21.2.5 defaultVehicleComponentPosition

```
Vector3 vehicleComponent.defaultVehicleComponentPosition  [private]
```

4.21.2.6 hornSound

```
AudioSource vehicleComponent.hornSound
```

4.21.2.7 isCompleted

```
bool vehicleComponent.isCompleted
```

4.21.2.8 isProcessed

```
bool vehicleComponent.isProcessed  [private]
```

4.21.2.9 resetTimer

```
float vehicleComponent.resetTimer = 0  [private]
```

4.21.2.10 spawnedComponent

```
GameObject vehicleComponent.spawnedComponent
```

4.21.2.11 valueAfterCompletion

```
float vehicleComponent.valueAfterCompletion = 0
```

4.21.2.12 VehicleComponent

```
VehicleComponent vehicleComponent.VehicleComponent = VehicleComponent.Motor
```

The documentation for this class was generated from the following file:

- vehicleComponent.cs

4.22 vehicleComponents Class Reference

Inheritance diagram for vehicleComponents:

Public Member Functions

- void saveCircuitToVehicleComponent (GameObject HeldItem)
- void loadSavedCircuitFromVehicleComponent (GameObject HeldItem)

Private Member Functions

- void Start ()
- void Update ()
- IEnumerator loadingComponents (GameObject HeldItem)

Private Attributes

- List< GameObject > componentsInGridboard = new List<GameObject>()

4.22.1 Member Function Documentation

4.22.1.1 loadingComponents()

```
IEnumerator vehicleComponents.loadingComponents (
            GameObject HeldItem )  [private]
```

4.22.1.2 loadSavedCircuitFromVehicleComponent()

```
void vehicleComponents.loadSavedCircuitFromVehicleComponent (
            GameObject HeldItem )
```

4.22.1.3 saveCircuitToVehicleComponent()

```
void vehicleComponents.saveCircuitToVehicleComponent (
            GameObject HeldItem )
```

4.22.1.4 Start()

```
void vehicleComponents.Start ( )  [private]
```

4.22.1.5 Update()

```
void vehicleComponents.Update ( )  [private]
```

4.22.2 Member Data Documentation

4.22.2.1 componentsInGridboard

```
List<GameObject> vehicleComponents.componentsInGridboard = new List<GameObject>()  [private]
```

The documentation for this class was generated from the following file:

- vehicleComponents.cs

4.23 vehicleEvaluation Class Reference

Inheritance diagram for vehicleEvaluation:

Public Member Functions

- void processComponent (string component)

Public Attributes

- Text text
- GameObject platform

Private Member Functions

- void Start ()
- void Update ()
- void finishGoKart ()
- void activateGoKart ()

Private Attributes

- List< string > components = new List<string>{"lights", "motor", "horn"}
- int progress = 0
- bool lowerPlatform

4.23.1 Member Function Documentation

4.23.1.1 activateGoKart()

```
void vehicleEvaluation.activateGoKart ( )  [private]
```

4.23.1.2 finishGoKart()

```
void vehicleEvaluation.finishGoKart ( )  [private]
```

4.23.1.3 processComponent()

```
void vehicleEvaluation.processComponent (
            string component )
```

4.23.1.4 Start()

```
void vehicleEvaluation.Start ( )  [private]
```

4.23.1.5 Update()

```
void vehicleEvaluation.Update ( )  [private]
```

4.23.2 Member Data Documentation

4.23.2.1 components

```
List<string> vehicleEvaluation.components = new List<string>{"lights", "motor", "horn"}  [private]
```

4.23.2.2 lowerPlatform

```
bool vehicleEvaluation.lowerPlatform  [private]
```

4.23.2.3 platform

```
GameObject vehicleEvaluation.platform
```

4.23.2.4 progress

```
int vehicleEvaluation.progress = 0  [private]
```

4.23.2.5 text

```
Text vehicleEvaluation.text
```

The documentation for this class was generated from the following file:

- vehicleEvaluation.cs

Chapter 5

File Documentation

5.1 amongus.cs File Reference

Classes

- class amongus

5.2 CircuitBuilder.cs File Reference

Classes

- class CircuitBuilder

5.3 ComponentController.cs File Reference

Classes

- class ComponentController

Enumerations

- enum ComponentType {
  ComponentType.Capacitor, ComponentType.Inductor, ComponentType.Mutual_Inductance, ComponentType.Resistor, ComponentType.BipolarJunctionTransistor, ComponentType.Diode, ComponentType.Junction_FET, ComponentType.Mosfet1,
  ComponentType.Voltage_Source, ComponentType.Switch, ComponentType.Output, ComponentType.Ground, ComponentType.Wire }
- enum WireType {
  WireType.deffinitelyNotAWire, WireType.Line, WireType.L_Shape, WireType.Trio,
  WireType.Quadro, WireType.Bridge }
- enum OutputType {
  OutputType.deffinitelyNotAnOutput, OutputType.Voltage_Meter, OutputType.Lightbulb, OutputType.LED,
  OutputType.Motor, OutputType.Horn }

5.3.1 Enumeration Type Documentation

5.3.1.1 ComponentType

`enum ComponentType [strong]`

Enumerator

Capacitor	
Inductor	
Mutual_Inductance	
Resistor	
BipolarJunctionTransistor	
Diode	
Junction_FET	
Mosfet1	
Voltage_Source	
Switch	
Output	
Ground	
Wire	

5.3.1.2 OutputType

`enum OutputType [strong]`

Enumerator

deffinitelyNotAnOutput	
Voltage_Meter	
Lightbulb	
LED	
Motor	
Horn	

5.3.1.3 WireType

`enum WireType [strong]`

Enumerator

deffinitelyNotAWire	
Line	
L_Shape	
Trio	
Quadro	
Bridge	

5.4 ComponentHandler.cs File Reference

Classes

- class ComponentHandler

5.5 Dashboard.cs File Reference

Classes

- class Dashboard

5.6 GoKartEnabler.cs File Reference

Classes

- class GoKartEnabler

5.7 GoKartScreenFader.cs File Reference

Classes

- class GoKartScreenFader

5.8 GraphFast.cs File Reference

Classes

- class GraphFast

5.9 GraphSample.cs File Reference

5.10 gridboardController.cs File Reference

Classes

- class gridboardController

5.11 gridboardPosition.cs File Reference

Classes

- class gridboardPosition

5.12 HandsController.cs File Reference

Classes

- class HandsController

5.13 InsertController.cs File Reference

Classes

- class InsertController

5.14 InteractionController.cs File Reference

Classes

- class InteractionController

5.15 ModelChanger.cs File Reference

Classes

- class ModelChanger

5.16 MovementController.cs File Reference

Classes

- class MovementController

5.17 PlayerInteractions.cs File Reference

Classes

- class PlayerInteractions

5.18 Scheme.cs File Reference

Classes

- class Scheme

5.19 SpeedLever.cs File Reference

Classes

- class SpeedLever

5.20 SpiceSharpController.cs File Reference

Classes

- class SpiceSharpController

5.21 SteeringWheel.cs File Reference

Classes

- class SteeringWheel

5.22 vehicleComponent.cs File Reference

Classes

- class vehicleComponent

Enumerations

- enum VehicleComponent { VehicleComponent.Motor, VehicleComponent.Light, VehicleComponent.Horn }

5.22.1 Enumeration Type Documentation

5.22.1.1 VehicleComponent

```
enum VehicleComponent  [strong]
```

Enumerator

Motor	
Light	
Horn	

5.23 vehicleComponents.cs File Reference

Classes

- class vehicleComponents

5.24 vehicleEvaluation.cs File Reference

Classes

- class vehicleEvaluation

Index

activateGoKart
    vehicleEvaluation, 64
AK
    GoKartEnabler, 23
amongus, 7
    ejectRotation, 8
    ejectSpeed, 8
    hexColors, 8
    kindasus, 8
    monkaLaugh, 8
    moveXorZ, 9
    spawnAmongus, 7
    spawnpoints, 9
    Start, 8
    Update, 8
amongus.cs, 67
Anode
    ComponentController, 14

BaseNode
    ComponentController, 14
BipolarJunctionTransistor
    ComponentController.cs, 68
body
    ComponentHandler, 19
Bridge
    ComponentController.cs, 69
BulkNode
    ComponentController, 15

CapacitanceValue
    ComponentController, 15
Capacitor
    ComponentController.cs, 68
capacitorsNodesForIC
    gridboardController, 32
Cathode
    ComponentController, 15
CB
    SpiceSharpController, 54
CC
    MovementController, 45
changeAttribute
    CircuitBuilder, 10
changeComponentsUnit
    ModelChanger, 42
changeComponentsValue
    ModelChanger, 42
changeDynamicAttribute
    SpiceSharpController, 53
changeModel
    ModelChanger, 42
changeResistance
    CircuitBuilder, 10
chart
    GraphFast, 27
checkedWires
    gridboardController, 32
checkForWireConnection
    gridboardController, 30
chosenColors
    GraphFast, 27
CircuitBuilder, 9
    changeAttribute, 10
    changeResistance, 10
    getSpiceSharpOutputs, 10
    inductor, 11
    mutualInductance, 11
    output, 11
    outputs, 11
    outputsResult, 11
    resistance, 11
    resistor, 11
    runSimulation, 10
    simulation, 12
    spiceCode, 12
    SpiceSharp_Viktor, 10
    string, 12
    tb, 12
    tran, 12
    TransformIntoSpiceSharpInput, 10
    voltageSource, 12
CircuitBuilder.cs, 67
closeSpiceSharpInstance
    SpiceSharpController, 53
CollectorNode
    ComponentController, 15
colors
    GraphFast, 27
columns
    gridboardController, 32
ComponentController, 13
    Anode, 14
    BaseNode, 14
    BulkNode, 15
    CapacitanceValue, 15
    Cathode, 15
    CollectorNode, 15
    ComponentType, 15

Coupling, 15
currentNodesPositions, 15
Dc, 15
defaultNameTag, 16
DrainNode, 16
DynamicGameObject, 16
EmitterNode, 16
Flow, 16
GateNode, 16
getDefaultNameTag, 14
Inductance, 16
InductorName1, 16
InductorName2, 17
Model, 17
NameTag, 17
NegativeControllingNode, 17
NegativeNode, 17
nodesPositions, 17
OutputType, 17
PositiveControllingNode, 17
PositiveNode, 18
Resistance, 18
rotatePins, 14
SourceNode, 18
Start, 14
SubstrateNode, 18
Update, 14
valueUnit, 18
WireType, 18
ComponentController.cs, 67
BipolarJunctionTransistor, 68
Bridge, 69
Capacitor, 68
ComponentType, 68
deffinitelyNotAnOutput, 68
deffinitelyNotAWire, 69
Diode, 68
Ground, 68
Horn, 68
Inductor, 68
Junction_FET, 68
L_Shape, 69
LED, 68
Lightbulb, 68
Line, 69
Mosfet1, 68
Motor, 68
Mutual_Inductance, 68
Output, 68
OutputType, 68
Quadro, 69
Resistor, 68
Switch, 68
Trio, 69
Voltage_Meter, 68
Voltage_Source, 68
Wire, 68
WireType, 68
ComponentHandler, 19
body, 19
insertedItemIC, 20
isInserted, 20
OnTriggerEnter, 19
parentContainer, 20
Start, 19
Update, 19
ComponentHandler.cs, 69
components
Scheme, 49
vehicleEvaluation, 65
componentsCircuit
gridboardController, 32
componentsDynamicAttribute
gridboardController, 32
componentsInGridboard
vehicleComponents, 64
ComponentsOutput
SpiceSharpController, 54
componentsOutput
gridboardController, 33
componentsRemovalCoroutine
gridboardController, 30
ComponentType
ComponentController, 15
ComponentController.cs, 68
connectComponents
gridboardController, 30
connectedComponents
gridboardController, 33
connectionIndexIdentifier
gridboardController, 33
connectionMultiWires
gridboardController, 33
Coupling
ComponentController, 15
createGraph
GraphFast, 26
createModels
ModelChanger, 42
createModelsFromClipboard
ModelChanger, 42
createSpiceSharpInput
gridboardController, 30
createSpiceSharpInstance
SpiceSharpController, 53
createSpiceSharpSimulation
gridboardController, 31
currentNodesPositions
ComponentController, 15
currentPickedItem
HandsController, 37
customModels
ModelChanger, 44

Dashboard, 20
engine, 22
horn, 22

hornOff, 21
hornOn, 21
lightbulb, 22
lights, 22
lightsOnOff, 21
setUpMotor, 21
Start, 21
Update, 21
Dashboard.cs, 69
Dc
ComponentController, 15
defaultComponent
vehicleComponent, 61
defaultComponentIsSpawned
vehicleComponent, 61
defaultCompononentPosition
vehicleComponent, 61
defaultNameTag
ComponentController, 16
defaultSlot
vehicleComponent, 61
defaultSprite
Scheme, 49
defaultVehicleComponentIndex
SpiceSharpController, 55
defaultVehicleComponentPosition
vehicleComponent, 61
deffinitelyNotAnOutput
ComponentController.cs, 68
deffinitelyNotAWire
ComponentController.cs, 69
deleteComponents
gridboardController, 33
Diode
ComponentController.cs, 68
disconnectComponents
gridboardController, 31
DrainNode
ComponentController, 16
dropdown
ModelChanger, 44
dropItem
HandsController, 36
DynamicAttributes
SpiceSharpController, 55
DynamicComponent
SpiceSharpController, 55
DynamicComponents
SpiceSharpController, 55
dynamicComponentsButton
SpiceSharpController, 55
DynamicComponentsSliders
SpiceSharpController, 55
DynamicGameObject
ComponentController, 16

ejectRotation
amongus, 8
ejectSpeed
amongus, 8
EmitterNode
ComponentController, 16
engine
Dashboard, 22
extractModelFromTuningStation
ModelChanger, 43

fadeOutJustInCaseSmileyFace
GoKartScreenFader, 25
finishGoKart
vehicleEvaluation, 65
finishGraphsLine
GraphFast, 26
finishingPath
GraphFast, 27
finitializeConnection
gridboardController, 31
Flow
ComponentController, 16

GameObjectsOutput
SpiceSharpController, 55
gameobjectsOutput
gridboardController, 33
GateNode
ComponentController, 16
generatePoints
GraphFast, 28
generateSchemeImages
Scheme, 48
getDefaultNameTag
ComponentController, 14
getGrabbedItem
HandsController, 36
getModelComponent
ModelChanger, 43
getModelSpecification
ModelChanger, 43
getSpiceSharpOutputs
CircuitBuilder, 10
GoKartEnabler, 22
AK, 23
Player, 23
PlayerController, 24
PlayerSittingPosition, 24
positionPlayerToKart, 23
removePlayerFromKart, 23
screenFader, 24
Start, 23
Update, 23
GoKartEnabler.cs, 69
GoKartScreenFader, 24
fadeOutJustInCaseSmileyFace, 25
OnTriggerEnter, 25
OnTriggerExit, 25
screenFader, 25
GoKartScreenFader.cs, 69
graph

SpiceSharpController, 56
graphCreated
GraphFast, 28
GraphFast, 25
chart, 27
chosenColors, 27
colors, 27
createGraph, 26
finishGraphsLine, 26
finishingPath, 27
generatePoints, 28
graphCreated, 28
graphScalerX, 28
graphScalerY, 28
hideGraph, 26
lineRenderer, 28
lineRenderers, 28
numericOutputs, 28
positionIndex, 28
resetGraph, 26
showGraph, 27
Start, 27
TimerX, 29
Update, 27
GraphFast.cs, 69
GraphSample.cs, 70
graphScalerX
GraphFast, 28
graphScalerY
GraphFast, 28
gravity
MovementController, 46
gridboardController, 29
capacitorsNodesForIC, 32
checkedWires, 32
checkForWireConnection, 30
columns, 32
componentsCircuit, 32
componentsDynamicAttribute, 32
componentsOutput, 33
componentsRemovalCoroutine, 30
connectComponents, 30
connectedComponents, 33
connectionIndexIdentifier, 33
connectionMultiWires, 33
createSpiceSharpInput, 30
createSpiceSharpSimulation, 31
deleteComponents, 33
disconnectComponents, 31
finitializeConnection, 31
gameobjectsOutput, 33
handleComponentInGridboard, 31
MC, 33
removeAllComponentsFromGridboard, 31
rows, 34
separatorsForComponentAttribute, 34
setUpChildCoordinates, 31
slots, 34
SpiceCode, 34
SSC, 34
Start, 31
tuningStation, 34
Update, 32
updateComponentAttributes, 32
gridboardController.cs, 70
gridboardPosition, 35
positionInGridboard, 35
gridboardPosition.cs, 70
Ground
ComponentController.cs, 68

handleComponentInGridboard
gridboardController, 31
Hands
InteractionController, 40
HandsController, 35
currentPickedItem, 37
dropItem, 36
getGrabbedItem, 36
isGrabbed, 37
pickupItem, 36
removeItem, 36
Start, 36
Update, 36
HandsController.cs, 70
HC
InteractionController, 40
hexColors
amongus, 8
hideGraph
GraphFast, 26
highlightedItem
InteractionController, 40
Horn
ComponentController.cs, 68
vehicleComponent.cs, 72
horn
Dashboard, 22
hornOff
Dashboard, 21
hornOn
Dashboard, 21
hornSound
vehicleComponent, 61

image
Scheme, 50
images
Scheme, 50
increaseX
Scheme, 50
increaseY
Scheme, 50
increasing
SpiceSharpController, 56
Inductance
ComponentController, 16

Inductor
ComponentController.cs, 68
inductor
CircuitBuilder, 11
InductorName1
ComponentController, 16
InductorName2
ComponentController, 17
initializeDynamicComponentsUI
SpiceSharpController, 53
initializeValueChanger
ModelChanger, 43
InsertController, 37
insertedItem, 38
insertItem, 38
isUsed, 39
pullItem, 38
saveItemToGrid, 38
snapItem, 38
Start, 38
Update, 38
InsertController.cs, 70
insertedCC
ModelChanger, 44
insertedItem
InsertController, 38
insertedItemIC
ComponentHandler, 20
insertItem
InsertController, 38
insertModelIntoTuningStation
ModelChanger, 43
InteractionController, 39
Hands, 40
HC, 40
highlightedItem, 40
invisibleMaterial, 40
outlinedMaterial, 40
pickupRange, 40
playerCamera, 41
simulatedItem, 41
Start, 39
Update, 40
InteractionController.cs, 70
invisibleMaterial
InteractionController, 40
isCompleted
vehicleComponent, 61
isGrabbed
HandsController, 37
isInKart
PlayerInteractions, 48
isInserted
ComponentHandler, 20
isProcessed
vehicleComponent, 61
isUsed
InsertController, 39
isZero
SpiceSharpController, 56

jumpSpeed
MovementController, 46
Junction_FET
ComponentController.cs, 68

kindasus
amongus, 8

L_Shape
ComponentController.cs, 69
LED
ComponentController.cs, 68
lelel
MovementController, 46
Light
vehicleComponent.cs, 72
Lightbulb
ComponentController.cs, 68
lightbulb
Dashboard, 22
lights
Dashboard, 22
lightsOnOff
Dashboard, 21
Line
ComponentController.cs, 69
lineRenderer
GraphFast, 28
lineRenderers
GraphFast, 28
loadGridboardComponentsFromVehicleComponent
vehicleComponent, 60
loadingComponents
vehicleComponents, 63
loadSavedCircuitFromVehicleComponent
vehicleComponents, 63
lookSpeed
MovementController, 46
lookXLimit
MovementController, 46
lowerPlatform
vehicleEvaluation, 65

MC
gridboardController, 33
Model
ComponentController, 17
ModelChanger, 41
changeComponentsUnit, 42
changeComponentsValue, 42
changeModel, 42
createModels, 42
createModelsFromClipboard, 42
customModels, 44
dropdown, 44
extractModelFromTuningStation, 43

getModelComponent, 43
getModelSpecification, 43
initializeValueChanger, 43
insertedCC, 44
insertModelIntoTuningStation, 43
models, 44
Start, 43
unit, 44
Update, 43
valueChanger, 44
modelChanger
Scheme, 50
ModelChanger.cs, 70
models
ModelChanger, 44
monkaLaugh
amongus, 8
Mosfet1
ComponentController.cs, 68
Motor
ComponentController.cs, 68
vehicleComponent.cs, 72
moveDirection
MovementController, 46
MovementController, 45
CC, 45
gravity, 46
jumpSpeed, 46
lelel, 46
lookSpeed, 46
lookXLimit, 46
moveDirection, 46
movementSpeed, 46
playerCamera, 46
rotationX, 47
Start, 45
Update, 45
MovementController.cs, 71
movementSpeed
MovementController, 46
moveXorZ
amongus, 9
Mutual_Inductance
ComponentController.cs, 68
mutualInductance
CircuitBuilder, 11

NameTag
ComponentController, 17
NegativeControllingNode
ComponentController, 17
NegativeNode
ComponentController, 17
nodesPositions
ComponentController, 17
numericOutputs
GraphFast, 28

OnTriggerEnter
ComponentHandler, 19
GoKartScreenFader, 25
OnTriggerExit
GoKartScreenFader, 25
outlinedMaterial
InteractionController, 40
Output
ComponentController.cs, 68
output
CircuitBuilder, 11
outputs
CircuitBuilder, 11
outputsResult
CircuitBuilder, 11
OutputType
ComponentController, 17
ComponentController.cs, 68

parentContainer
ComponentHandler, 20
pickupItem
HandsController, 36
pickupRange
InteractionController, 40
platform
vehicleEvaluation, 65
Player
GoKartEnabler, 23
playerCamera
InteractionController, 41
MovementController, 46
PlayerController
GoKartEnabler, 24
PlayerInteractions, 47
isInKart, 48
Start, 47
Update, 47
PlayerInteractions.cs, 71
PlayerSittingPosition
GoKartEnabler, 24
positionIndex
GraphFast, 28
positionInGridboard
gridboardPosition, 35
positionPlayerToKart
GoKartEnabler, 23
PositiveControllingNode
ComponentController, 17
PositiveNode
ComponentController, 18
processComponent
vehicleComponent, 60
vehicleEvaluation, 65
progress
vehicleEvaluation, 66
pullItem
InsertController, 38

Quadro

ComponentController.cs, 69

removeAllComponentsFromGridboard
gridboardController, 31
removeComponentsButton
SpiceSharpController, 56
removeItem
HandsController, 36
removePlayerFromKart
GoKartEnabler, 23
resetDynamicOutputs
SpiceSharpController, 53
resetGraph
GraphFast, 26
resetSchemeImage
Scheme, 49
resetTimer
vehicleComponent, 62
resetVehicleComponent
vehicleComponent, 60
Resistance
ComponentController, 18
resistance
CircuitBuilder, 11
Resistor
ComponentController.cs, 68
resistor
CircuitBuilder, 11
rotatePins
ComponentController, 14
rotationMultiplier
SteeringWheel, 58
rotationX
MovementController, 47
rows
gridboardController, 34
runSimulation
CircuitBuilder, 10

saveCircuitToVehicleComponent
vehicleComponents, 63
saveGridboardComponentsIntoVehicleComponent
vehicleComponent, 60
saveItemToGrid
InsertController, 38
Scheme, 48
components, 49
defaultSprite, 49
generateSchemeImages, 48
image, 50
images, 50
increaseX, 50
increaseY, 50
modelChanger, 50
resetSchemeImage, 49
Start, 49
symbols, 50
Update, 49
updateSchemeImage, 49

Scheme.cs, 71
screenFader
GoKartEnabler, 24
GoKartScreenFader, 25
separatorsForComponentAttribute
gridboardController, 34
setUpChildCoordinates
gridboardController, 31
setUpMotor
Dashboard, 21
showGraph
GraphFast, 27
showOrHideDynamicComponents
SpiceSharpController, 54
showOrHideSpiceCode
SpiceSharpController, 54
simulatedItem
InteractionController, 41
simulation
CircuitBuilder, 12
simulationOutput
SpiceSharpController, 56
slots
gridboardController, 34
snapItem
InsertController, 38
SourceNode
ComponentController, 18
spawnAmongus
amongus, 7
spawnedComponent
vehicleComponent, 62
spawnpoints
amongus, 9
speakerCurrentPeak
SpiceSharpController, 56
speakerFrequencyTimer
SpiceSharpController, 56
SpeedLever, 51
SpeedLeverHinge, 51
SpeedLeverInput, 51
Start, 51
Update, 51
SpeedLever.cs, 71
SpeedLeverHinge
SpeedLever, 51
SpeedLeverInput
SpeedLever, 51
SpiceCode
gridboardController, 34
spiceCode
CircuitBuilder, 12
SpiceSharpController, 56
spiceCodeButton
SpiceSharpController, 57
SpiceSharp_Viktor
CircuitBuilder, 10
SpiceSharpController, 52

CB, 54
changeDynamicAttribute, 53
closeSpiceSharpInstance, 53
ComponentsOutput, 54
createSpiceSharpInstance, 53
defaultVehicleComponentIndex, 55
DynamicAttributes, 55
DynamicComponent, 55
DynamicComponents, 55
dynamicComponentsButton, 55
DynamicComponentsSliders, 55
GameObjectsOutput, 55
graph, 56
increasing, 56
initializeDynamicComponentsUI, 53
isZero, 56
removeComponentsButton, 56
resetDynamicOutputs, 53
showOrHideDynamicComponents, 54
showOrHideSpiceCode, 54
simulationOutput, 56
speakerCurrentPeak, 56
speakerFrequencyTimer, 56
spiceCode, 56
spiceCodeButton, 57
SpiceSharpThread, 57
Start, 54
startSimulationButton, 57
stopSimulationButton, 57
Update, 54
validateOutput, 54
vehicleComponentInGridboard, 57
SpiceSharpController.cs, 71
SpiceSharpThread
SpiceSharpController, 57
SSC
gridboardController, 34
Start
amongus, 8
ComponentController, 14
ComponentHandler, 19
Dashboard, 21
GoKartEnabler, 23
GraphFast, 27
gridboardController, 31
HandsController, 36
InsertController, 38
InteractionController, 39
ModelChanger, 43
MovementController, 45
PlayerInteractions, 47
Scheme, 49
SpeedLever, 51
SpiceSharpController, 54
SteeringWheel, 58
vehicleComponent, 60
vehicleComponents, 63
vehicleEvaluation, 65
startSimulationButton
SpiceSharpController, 57
SteeringWheel, 57
rotationMultiplier, 58
Start, 58
SteeringWheelHinge, 58
SteeringWheelInput, 58
TurningWheels, 59
Update, 58
SteeringWheel.cs, 71
SteeringWheelHinge
SteeringWheel, 58
SteeringWheelInput
SteeringWheel, 58
stopSimulationButton
SpiceSharpController, 57
string
CircuitBuilder, 12
SubstrateNode
ComponentController, 18
Switch
ComponentController.cs, 68
symbols
Scheme, 50

tb
CircuitBuilder, 12
text
vehicleEvaluation, 66
TimerX
GraphFast, 29
tran
CircuitBuilder, 12
TransformIntoSpiceSharpInput
CircuitBuilder, 10
Trio
ComponentController.cs, 69
tuningStation
gridboardController, 34
TurningWheels
SteeringWheel, 59

unit
ModelChanger, 44
Update
amongus, 8
ComponentController, 14
ComponentHandler, 19
Dashboard, 21
GoKartEnabler, 23
GraphFast, 27
gridboardController, 32
HandsController, 36
InsertController, 38
InteractionController, 40
ModelChanger, 43
MovementController, 45
PlayerInteractions, 47
Scheme, 49

SpeedLever, 51
SpiceSharpController, 54
SteeringWheel, 58
vehicleComponent, 60
vehicleComponents, 63
vehicleEvaluation, 65
updateComponentAttributes
gridboardController, 32
updateSchemeImage
Scheme, 49

validateOutput
SpiceSharpController, 54
valueAfterCompletion
vehicleComponent, 62
valueChanger
ModelChanger, 44
valueUnit
ComponentController, 18
VehicleComponent
vehicleComponent, 62
vehicleComponent.cs, 72
vehicleComponent, 59
defaultComponent, 61
defaultComponentIsSpawned, 61
defaultCompononentPosition, 61
defaultSlot, 61
defaultVehicleComponentPosition, 61
hornSound, 61
isCompleted, 61
isProcessed, 61
loadGridboardComponentsFromVehicleCompo-
nent, 60
processComponent, 60
resetTimer, 62
resetVehicleComponent, 60
saveGridboardComponentsIntoVehicleComponent,
60
spawnedComponent, 62
Start, 60
Update, 60
valueAfterCompletion, 62
VehicleComponent, 62
vehicleComponent.cs, 71
Horn, 72
Light, 72
Motor, 72
VehicleComponent, 72
vehicleComponentInGridboard
SpiceSharpController, 57
vehicleComponents, 62
componentsInGridboard, 64
loadingComponents, 63
loadSavedCircuitFromVehicleComponent, 63
saveCircuitToVehicleComponent, 63
Start, 63
Update, 63
vehicleComponents.cs, 72
vehicleEvaluation, 64
activateGoKart, 64
components, 65
finishGoKart, 65
lowerPlatform, 65
platform, 65
processComponent, 65
progress, 66
Start, 65
text, 66
Update, 65
vehicleEvaluation.cs, 72
Voltage_Meter
ComponentController.cs, 68
Voltage_Source
ComponentController.cs, 68
voltageSource
CircuitBuilder, 12

Wire
ComponentController.cs, 68
WireType
ComponentController, 18
ComponentController.cs, 68