МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Херсонський державний університет

Факультет комп'ютерних наук, фізики та математики

Кафедра інформатики, програмної інженерії та економічної кібернетики

3D ОБ'ЄКТИ ВІРТУАЛЬНИХ ЛАБОРАТОРНИХ РОБІТ З ТЕМИ «ЕЛЕКТРИЧНІ ЯВИЩА. ЕЛЕКТРИЧНИЙ СТРУМ»

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти "бакалавр"

Виконав: студент 4 курсу
Спеціальності 121 Інженерія програмного забезпечення
Освітньо-професійної програми
«Інженерія програмного забезпечення»
першого (бакалаврського) рівня освіти
Соболь Микола Вікторович
Керівники кандидат фізико-математичних наук, доцент
Кравцов Геннадій Михайлович
доктор педагогічних наук, професор
Співаковський Олександр Володимирович
Рецензент кандидат педагогічних наук, доцент Гончаренко Тетяна Леонідівна

Херсон – 2020

ЗМІСТ

ВСТУП

Наше сьогодення характерне своєю мінливістю. Безупинний розвиток інформаційних технологій зробило будь-яку інформацію доступною для кожного, в будь-якій точці світі. Тепер, маючи з'єднання з глобальною мережею, можна отримати, змінити, доповнити будь-яке знання. Тому не дивно, що світ отримую нові технології в таких гігантських обсягах, а темпи їх отримання можна сміливо назвати реактивними. Більш того, іноді суспільство навіть не встигає осягнути впливу цього надшвидкого руху інформації на світ і на нього саме включно. В деякому роді, цей розвиток можна назвати рекурсивним, тобто зав'язаним на самому собі: чим швидше і більше з'являється інформація, тим більше прискорення та обсяги появи нової.

Цей надшвидкий ріст інформації створив парадокс її розвитку, що заведено називати «Інформаційним вибухом». Щоб зрозуміти масштаб цього явища, достатньо подивитись на статистику створення нових знань за останні 30 років[1]. Так наприклад, згідно з цими дослідженнями, людство у 2002 році виробило майже 18 * 108 байтів інформації або ж 18 Ексабайт (Ексабайт – міра вимірювання обсягу інформації). До того ж, за попередні до цього п'ять років було вироблено інформації більше, ніж за всю відому історію, а щорічний приріст обсягу нової становить приблизно 30 відсотків.

Природно, що такий швидкий приріст створив інфляцію знань. У світі, де знання можуть втратити актуальність не за роки, а місяці, дні, або навіть години, стало життєво необхідним вміти пристосовуватись до актуальної інформації, а головне — володіти вмінням швидко навчатись.

Достатньо швидко стало розуміло, що класичні системи навчання мало підходять для таких цілей. Тому потрібно було знайти рішення, що дозволило б надати відповідь на подібний запит сучасного ринку праці.

На щастя, вихід був знайдений напрочуд швидко. Звичайно йде річ про дистанційне навчання.

Актуальність теми.

Як вже було сказано вище, наш світ став дуже стрімким і мінливим. Тому й змінились вимоги до сучасних професій, а й відповідно, до спеціалістів цих професій. Так якщо раніше, навчання спеціаліста було комплексним і під час нього він отримував величезний обсяг інформації, з різних напрямків, а сама його спеціалізація містила більший обсяг універсальності. Так в приклад можна привести різних цехових майстрів і інженерів минулого та позаминулого століть. Цеховий робітник майже завжди знав як працювати з усім приладдям в цеху, а методи роботи з цими інструментами й самі ці інструменти іноді не змінювались за всю його робочу кар'єру. Часто все це переростало в деякого роду «традиції» та консервативність у ведені роботи, що дуже уповільнювало модернізацію виробництва та створювало супротив зі сторони таких робітників, у випадку вимоги підвищення їх кваліфікації, тобто зміни встановленого ладу.

Тепер же, сучасний ринок праці диктує зовсім нові правила і вимоги. Так зараз, консервативність в роботі не тільки не в потребі, а часто може зламати кар'єру робітника. Нинішні вимоги до спеціаліста – це гнучкість в навчанні та вміння швидко пристосовуватись до нових реалій ринку, нових технологій та інструментів. Більш цінується не кількість отриманих знань, а вміння їх використати, адаптувати до своєї роботи, а за потреби – робити це зі всією новою інформацією. Не меншою мірою ринок потребує працівників, що прагнуть і спроможні до навчання і підвищення кваліфікації. Для працедавців вже давно стало повсякденним витрачення значних коштів на перспективних працівників, що можливо зараз і не мають потрібної кваліфікації, але вході навчання отримають її, що звичайно, в перспективі, принесе більше користі для обох сторін.

Робітник, що не прагне до навчання і підвищення власної кваліфікації, швидко опиниться на узбіччі життя, бо не є перспективним спеціалістом. Доки він не буде прагнути цього, його участь бути низько

кваліфікованим та непотрібним працівником, а відповідно його робота буде низько оплачуваною й сам робітник, в такому випадку, навряд чи може сподіватись на якісь гарантії власного добробуту.

Відповідно, щоб пристосуватись до таких швидко змінюючися вимог, необхідно було мати можливість швидко навчатись. Зрозуміло, що традиційне стаціонарне навчання мало підходило для таких цілей. Особливо, якщо потрібно було навчитись не всій спеціальності в цілому, а лише деяким її вузьких напрямках. Окрім того класичне навчання потребує, зазвичай, повної концентрації на самому навчанні, перебування на місці стаціонарного навчання, а також не має можливості модульного навчання, коли учень може вибрати окремі розділи знань, опановування якими його цікавить.

В більшості випадків, людина, що вже працює має мало вільного часу, який вона може витратити на навчання, а тим паче змінювати своє місце перебування заради цього. Тому дистанційне навчання є напрочуд ефективним розв'язання цих проблем, адже в його основі лежать самі ті переваги, що необхідні для сучасного, швидкого навчання, а саме:

- Модульність
- Можлива асинхронність
- Можна швидко змінити чи доповнити програму навчання
- Не потребує стаціонарного перебування в місці навчання
- Не потребує постійної концентрації на предметі навчання
- В більшості випадків невибагливе до матеріальної бази учня
- Дозволяє суттєво знизити витрата на матеріально базу для закладу навчання
- Дозволяє зменшити навантаження на викладачів

Водночас, вибрана автором тема « 3D об’єкти віртуальних лабораторних робіт з теми «Електричні явища. Електричний струм»» є напрочуд актуальною, якщо розглядати її на фоні потреби в засобах дистанційного навчання. Окрім того, оскільки тема має відношення до

природничих наук, а саме фізики, то тема дипломної роботи стає ще більш актуальною для нашого часу. І на це є декілька причин:

- Сучасна молодь, а саме учні шкіл, мало зацікавлені в вивченні природничих наук, оскільки це досить вибагливі до концентрації дисципліни. Проблемою є ще й те, що більшість навчальних програм морально застарілі, мало наочні та не ставлять за свою ціль зацікавити учня в поглибленні власних знань з наданого матеріалу, а лише сухого викладають інформацію, переобтяжують матеріал формулами та іноді не зовсім зрозумілими для дітей трактуваннями фізичних процесів.

- Хоча більшість таких навчальних програм передбачають наочну презентацію матеріалу, однак через скрутне становище матеріальної бази в багатьох навчальних закладах середньої освіти, така діяльність як проведення лабораторних робіт з використанням повноцінного приладдя, реагентів, матеріалів та інших необхідних інструментів, є рідкістю, окрім найбільш примітивних і не потребуючих спеціальної підготовки варіантах.

- Не слід забувати й про збереження безпеки під час таких занять. Лабораторні роботи з природничих дисциплін завжди потребують надання високого рівня підтримки техніки безпеки, адже зазвичай йде мова про використання потенційно небезпечних реагентів або використання електричного струму та приладів, що живляться ним.

- Немало важливим фактором є обмеженість в часі при виконанні цих робіт. Зазвичай це не більше 20 – 30 хвилин, тобто період уроку, що не дозволяє зайнятись комплексним, складним експериментом, що займав би тривалий проміжок часу

- Також слід зазначити, що безпосередньо вибрана тема, що зв’язана з електричним струмом є досить актуально. Адже це явище не просто оточує нас кожного дня, а вже стало невід'ємною частиною нашого життя. Тому необхідно вчити дітей безпечного користування струмом, що в першу чергу, передбачає розуміння самого явища та принципів його дії.

Адже більшість травматичних випадків стаються саме через незнання природи явища.

- Окрім того популяризація вивчення цього розділу фізики та облегшення його вивчення через наочність та індивідуальний підхід до кожного учня, створення інтерактивних та цікавих програм, призведе до з’яви спеціалістів в тих технічних галузях, де вони гостро необхідні. Так, наприклад, в наш час існує низка проблем з створенням ефективних акумуляторів та швидких і водночас надійних способів їх зарядки. І це не кажучи про проблему пошуку не вуглецевих джерел енергії. Можливо, з’ява великої кількості нових спеціалістів в цих сферах дозволить розвинути відповідні галузі.

Якщо проаналізувати саму структуру поняття «віртуальних лабораторій», то стає зрозуміло, що вона вирішує більшість вказаних вище проблем, що стосуються саме способу навчання.

В підсумку можна зауважити, що створення програмного модуля віртуальних робіт нам допоможе розв'язати такі проблеми як:

- Безпека. Оскільки всі процеси відбуваються симуляційним чином, то буде неможливо, щоб вони завдали шкоди учневі.

- Час. Робота з віртуальною лабораторією мало прив’язано до реального часу, адже кожний процес симуляції можна тимчасово зупинити, прискорити чи сповільнити за власним бажанням. Окрім того, до колись початої роботи можна повернутись, звичайно якщо в модулі є така можливість.

- Контроль і оцінювання. Дистанційне навчання дає можливість уникнути постійного контролю над учнем. Більшість модулів віртуальних лабораторій дозволяють записати, зберегти та надалі відтворити послідовність дій учня та отримані ним результати.

- Матеріальна база. Одна з головних проблем традиційного навчання відходить на задній план, адже для виконання віртуальної лабораторної роботи потрібен лише один інструмент – комп’ютер.

- Наочність. Учень зможе без ризику розглянути процеси, що досліджуються та необхідні для них інструменти.

Об'єкт дослідження : віртуальна лабораторія з теми «Електричні явища. Електричний струм», створена з огляду на систему дистанційного навчання та засобами Unity 3D.

Предмет дослідження : бібліотека програмних 3D об’єктів віртуальних лабораторних робіт з теми «Електричні явища. Електричний струм».

Мета дослідження : розробити бібліотеку програмних модулів віртуальних лабораторних робіт.

Завдання дослідження :

1) Змоделювати програмні 3D об’єкти віртуальних лабораторних робіт.

2) Спроєктувати класи та бібліотеки програмного модуля.

3) Розробити бібліотеку програмних модулів.

Гіпотеза. Програмні засоби рушію Unity3D дозволять ефективно та зручно спроєктувати, змоделювати і створити фізичні об'єкти, що будуть симулювати необхідні фізичні явища, для віртуальних лабораторій з фізики.

Методи дослідження.

- теоретичні: аналіз, синтез.

- емпіричні: спостереження, порівняння

Огляд літератури

В першу чергу, приступаючи до проєктування класів 3D об’єктів, нам потрібно зібрати та дослідити потрібні матеріали з тем, що нам дозволять створити необхідний програмний продукт. А саме з таких тем:

1) Віртуальні лабораторії. Віртуальні лабораторії, як вид дистанційного навчання.

2) Основи та принципи роботи явища, що ми будем симулювати, а саме «Електричні явища. Електричний струм».

3) Особливості проєктування класів об’єктів, що симулюють роботу вище вказаних явищ.

4) Засоби створення для створення 3D об’єктів, що надає рушій Unity 3D.

5) Вже реалізовані матеріали з цієї або подібних тем.

Короткий огляд матеріалів з вище вказаних тем (дані про точне розміщення матеріалів вказано в розділі «Джерела») :

1) Матеріали статті «Дистанційне навчання» з Вікіпедії [2], що описують основні принципи поняття «Дистанційне навчання»

2) Матеріали статті «Unity» з Вікіпедії [3], що описує рушій Unity 3D.

3) Матеріали статті «Мультимедийная виртуальная лаборатория по физике в системе дистанционного обучения» [4], що описує основи концепції «Віртуальна лабораторія», особливості її проєктування та її класів для симуляції фізичних процесів, спираючись на засоби Unity 3D.

4) Матеріали підручника «Фізика : підручник для 8 класів. Бар'яхтар» [5], що містить опис фізичних явищ, особливості взаємодій фізичних об’єктів, що ми будемо симулювати та еталонні задачі (лабораторні роботи), які допоможуть нам протестувати наш програмний продукт.

5) Матеріали статті « About One Approach to Building Systems for Testing Physical Knowledge» [6], що містить опис особливостей моделювання об’єктів симуляції електричних явищ.

6) Матеріали офіційного посібника з Unity 3D [7], що дозволяють нам дослідити доступні засоби Unity 3D, необхідні нам для досягнення поставлених задач дослідження.

Огляд деяких вже представлених схожих програмних продуктів

LabVIEW

LabVIEW – платформа та середовище розробки для графічної мови програмування «G» (хоча зазвичай її назву опускають, маючи на увазі, зазвичай, LabVIEW) створена National Instruments (США)[8].

Головне призначення цього продукту – автоматизація обчислювального та вимірювального лабораторного приладдя.

Основні переваги LabVIEW[9]:

- графічне програмування, що значно спрощує роботу з середовищем для не програмістів
- можливість працювати з великою кількістю приладів через значну базу вже вбудованих в продукт драйверів
- велика кількість віртуальних приладів, що надаються з продуктом
- зручний інтерфейс
- велика кількість вже вбудованих функцій для різних операцій: збору даних, обчислень, генерації та аналізу сигналів тощо.
- незалежне існування створених програм від платформи в повноцінному середовищі, плюс можливість паралельного виконання команд
- значне ком’юніті продукту та величезна кількість супровідної документації

Однак з цим існує деяка кількість значних недоліків[10]:

- LabView є власницьким продуктом National Instrument, тобто потребує платної активації. Звичайно, є безкоштовна версія, але вона має досить обмежений час роботи.
- Потребує для роботи встановленого середовища з відповідними бібліотеками
- Сумнівна мова програмування
- Мало сумісний з іншими платформами, окрім Windows

Отже, стає зрозумілим, що LabView не є досить актуальним рішенням для створення віртуальних лабораторій.

JMCAD

JMCAD[11] – це програмний комплекс створений Юрієм Михайловським, основне призначення якого симуляція і моделювання складних динамічних систем.

Найбільше широке своє використання знайшов в навчальному процесі, моделюванні різноманітних явищ у фізиці, електротехніці, в динаміці машин і механізмів.

Основними його перевагами можна зазначити:

- Відкритий код, написаний на Java
- Реалізація обміну даними з зовнішніми програмами
- Кросплатформність
- Простота в побудові складних моделей
- Ефективні численні методи
- Велика кількість навчальної документації, що супроводжується прикладами [12]

Однак, цей програмний продукт мало нам підходить через надмірну універсальність, адже в дослідженні треба реалізувати моделювання симуляції вузькоспеціалізованої сфери фізики.

Flash застосунки та інші

В мережі Інтернет часто можна зустріти приклади «інтерактивних» віртуальних лабораторних робіт створених за допомогою технології Flash або малих застосунків. Однак більшість таких продуктів, насправді, не містять ніякої інтерактивності, а вся послідовність дій в такій лабораторній роботі суворо визначена. В деяких випадках ви можете змінити деякі числові параметри фізичного явища чи приладу, але не більше. Тим паче, про яку-небудь фізичну взаємодію об’єктів таких

лабораторних, на кшталт, зіткнення колізій, симуляції пружності чи рідини, взагалі не йде мови.

Тому такі продукти, певною мірою, можуть виступати як демонстрації фізичного явища, але повністю не компетентні, як засоби віртуальної лабораторії та фізичної симуляції.

Приклад такого програмного продукту вказано в посиланні, в розділі «Джерела» [13]

Наукова новизна отриманих результатів.

Отже, враховуючи розділ про актуальність роботи та аналіз літератури з теми нашого дослідження, можна сміливо констатувати, що отримані результати дослідження мають певну наукову новизну. І на це вказує декілька факторів, а саме:

- Попри присутність багатьох програмних продуктів, що дозволяють реалізувати симуляцію фізичних процесів, в більшості вони є універсальними та широко спеціалізованими, а отже потребують наявності повного програмного комплексу, виключаючи використання окремого і самостійного модуля.
- Більшість таких продуктів є платними
- Деякі продукти, як FLASH застосунки, лише імітують симуляцію фізичних процесів.
- Висока актуальність технології Unity 3D

Враховуючи вище вказані причини, можна дійти висновку, що отримані результати є актуальними та мають наукову новизну на фоні інших представлених програмних продуктів.

Структура роботи

Робота складається з 2 розділів. У першому розділі описуються основи технології моделювання 3D об’єктів та технології Unity для створення цих об’єктів, а також особливості фізичного явища, що

розглядає дослідження. У другому розділі описано, як використовуючи технології Unity та характерні риси фізичного явища, вказані в попередньому розділі, спроєктувати класи фізичних об’єктів та бібліотеку цих фізичних об’єктів, що можна використати в створенні віртуальної лабораторної роботи. Також в цьому розділі описують особливості моделювання класів об’єктів та програмних підходів, приводяться приклади коду цих класів та описання деяких нюансів їх структури та роботи.

РОЗДІЛ 1

МОДЕЛЮВАННЯ ПРОГРАМНИХ 3D ОБ'ЄКТІВ. ТЕХНОЛОГІЇ UNITY 3D. ОСНОВИ ТЕМИ «ЕЛЕКТРИЧНІ ЯВИЩА ТА ЕЛЕКТРИЧНИЙ СТРУМ»

1.1. Моделювання програмних 3D об'єктів

В першу чергу, нам треба дати визначення 3D об'єктів та 3D моделювання, а вже потім перейти до опису особливостей цих технологій та їх використання.

Отже:

- 3D модель або об'єкт – це об'ємне цифрове зображення необхідного об'єкту, як існуючого в реальності, так і вигаданого.
- 3D моделювання – це процес розробки будь-якої тривимірної поверхні, через математичне явлення, тобто використовуючи математичні закони. Зазвичай, подібна робота виконується в спеціалізованому програмному забезпеченні, а результат цієї праці, відповідно, називають 3D – моделлю.
- Програмне забезпечення 3D моделювання – є частиною такого класу програмних продуктів, як інструменти для відтворення та створення комп'ютерної 3D графіки.

Окрім того, модель потрібно якось показати або використати певним чином. Існує декілька прикладів цього. Наприклад:

- Двовимірне зображення моделі, за допомогою процесу 3D – візуалізації. Тобто виведення звичного нам в повсякденному житті «плоского» зображення на пристрій виводу – екран, проєктор, тощо.
- Використати в комп'ютерному моделюванні. Наприклад, при симуляції фізичних явищ, рендеру відео чи створенні ігрових продуктів.
- Відтворити 3D модель у вигляді фізично наявного об'єкта. В цьому допоможуть пристрої 3D друку, наприклад 3D принтер.

Підходи до створення модель можна розділити на два: автоматичний та, відповідно, ручний.

Автоматичне створення базується на генерації за допомогою математичних законів, користувачу тут відведена найменша роль – керування параметрами генерації.

Ручне створення відштовхується від протилежної ідеї – повний контроль при створенні, з мінімумом генерації. Найближчий аналог цього підходу – пластичні мистецтва, на кшталт ліплення, скульптури або різьби по різному матеріалу. Природно, що саме такий підхід дозволяє створити більш деталізовані моделі чім згенеровані.

Також можна відокремити метод створення 3D моделей за допомогою фотограмметрія [14] – перехідного варіанту між ручним та автоматичними підходами. В цьому методі використовується створенні 3D відбитки об’єктів методом 3D сканування або зіставлення декількох фотознімків об’єкту з різних ракурсів. Попри надвисоку деталізацію створених таким чином моделей, існує багато недоліків. Це й висока вартість інструментів, необхідність в спеціалістах, що працюють з цією технологією, а головне – неможливість коректного використання створених моделей без доопрацювання, адже об’єкти при такому підході зберігають світлові властивості зображення на момент сканування: тіні, блиск і так далі. Останнє робить неможливим використання таких моделей у випадку динамічної симуляції освітлення без відповідного налаштування. Тому зазвичай технологія фотограмметрії використовується для створення або малих за розміром об’єктів при «ідеальному» скануванні або статичних сцен – 3D зліпків, панорам тощо.

Існує можливість також поділити 3D моделі на дві категорії за їх формою та об’ємом:

- Solid-моделі (або твердотілі)[15] - це моделі, що повністю відтворюють об’єм об’єкту, тобто повторюють його обсяг, не допускаючи порожнин всередині, навіть якщо їх непомітно. Звичайно, такі моделі

більш реалістичні, особливо при симуляції деяких фізичних явищ (особливо світлових), однак їх створення супроводжується більшими складнощами. Найчастіше твердотілі моделі застосовуються при не візуальному моделюванні, на кшталт, медичного та інженерного моделювання, або в інших спеціалізованих візуальних додатках, таких як трасування променів.

- Shell-моделі (або моделі оболонки) – це моделі, що відкидають повне відтворення обсягу об’єкта, а є лише зображенням його поверхні, наприклад, межею. З такими моделями значно легше працювати ніж з твердотілими. Ця категорія моделей – найбільш розповсюджена, саме через простоту створення і роботи з ними. Більша частина моделей комп’ютерної графіки, що ми бачимо в кіно та іграх, є саме моделями оболонки.

Як ми знаємо, зовнішній вигляд об’єкту залежить в більшій своїй частині саме від його зовнішньої поверхні, що дозволяє використати цю властивість в комп’ютерній графіці. Стає зрозумілим, що двовимірні поверхні — чудова аналогія для об’єктів, які використовуються в 3D графіці. Оскільки поверхні не являються кінцевим, дискретним цифровим наближенням, то треба шукати інші, схожі уявлення.

Чудову підпадають під необхідне визначення полігональні сітки (а також деякі інші підвиди поверхонь). Саме полігональні сітки є найбільш поширеним зараз уявленням.

Також не буде зайвим згадати про такий процес перетворення уявлень об’єкта як теселяція.

Тесаляція[16] - це спосіб заміщення площини об’єкту використовуючи примітиви – найпростіші в обробці і представлені фігури: трикутники, квадрати, сфери, тощо. Всі примітиви на поверхні створюють між собою зв’язок – сітку.

Найбільш популярними примітивами при теселяції є трикутники, що також називають полігонами. Саме трикутники є найбільш зручними для

відтворення плоскої поверхні, адже містять в собі все необхідне для її відтворення – три вершини та відстань між ними. Сітку примітивів, що складається виключно з полігонів, називають полігональною сіткою.

Існує три найбільш популярних способи представлення 3D моделі:

- Полігональне моделювання[17] – об’єкт подається у вигляді власних поверхонь, що побудовані за допомогою багатокутників (полігонів). Тобто поверхні існують як точки в 3D просторі (вершини), що з’єднані відрізками. Таким чином вершини утворюють полігональну сітку. Більшість сучасних 3D моделей використовують саме цей спосіб представлення. І на це є дві основні причини: простота й гнучкість при роботі з цими моделями та швидкість їх опрацювання комп’ютерами. Однак, звичайно, є й недоліки. Через те, що багатокутники є лише плоскими, то вони можуть лише наближатись до криволінійних поверхонь, імітуючи їх. Хоча на практиці, при використанні в ігровому продукті чи відео, достатня кількість полігонів при деталізації моделі, робить імітацію скривлення поверхні об’єкту непомітним людському оку.
- Криволінійне моделювання – тут поверхню визначають не вершини, а криві, що знаходяться під регулюванням зважених контрольних точок. Збільшення ваги точки призведе до того, що крива буде сильніше тягнути саме до неї й навпаки зменшення ваг до меншого скривлення. Також цей метод підтримує різні види кривих, а саме неоднорідні раціональні B-сплайн (NURBS)[18], шліци, сплайни[19] та геометричні примітиви[20]. Певною мірою цей метод схожий на фізичну природу гравітації, де вага об’єкту визначає скривлення навколишнього простору-часу, на кшталт, кривих і контрольних точок.
- Цифрове ліплення [21] – наймолодший метод, і водночас один з перспективних. У своїй суті повторює реальну послідовність створення матеріального тривимірного об'єкта, але електронними засобами у вигляді віртуального уявлення. Більшість створених цим методом

моделей відносяться до твердотілих (Solid), а загальна поверхня задається уточненням декількох поверхонь, що накладаються одна на одну. Це дуже схоже на реальну ліпку де тонкі нашарування глини чи гіпсу утворюють загальний об’єм фігури. Також завдяки використанню 32-бітної карт зображень отриману можна триангулювати, тобто провести теселяцію з використанням трикутників, що дозволить створити полігональну сітку. Це дасть змогу додати деталізації об’єкту та перевести його в формат більш зручний для роботи комп’ютера.

Етап моделювання складається з аналізу питань. Які об’єкти необхідні? Якого вони повинні бути розміру, обсягу? Який загальний рівень деталізації та які технології треба використати?

Різноманітність таких створило безліч відповідей на них. Однак їх всі можна сформулювати за категоріями. Такі категорії й породили різноманіття методів моделювання.

Ось деякі приклади цих методів:

- Неявні поверхні
- Конструктивна стереометрія
- Дроблення поверхонь

Саме моделювання створюються за допомогою відповідних програмних продуктів. Наприклад:

- 3ds Max
- Maya
- Blender
- Lightwave
- Modo

Для моделювання складних фізичних матеріалів, як рідини, сипучі частинки чи газоподібних речовин, існує декілька підходів. Наприклад, симуляція через систему частинок, що схоже на підхід криволінійного моделювання, адже й тут використовується принцип контрольних точок,

за винятком того, що в системі частинок, кожна частинка є сама по собі контрольною точкою і її вага діє відносно інших точок. Схожим чином працює система вокселів.

Також можна не симулювати частинки, а імітувати, наприклад, використовуючи спрайти – плоскі зображення.

Окрім самого моделювання об’єктів, стоїть ще питання їх кінцевої візуалізації, тобто виведення. Зазвичай тривимірна графіка працює в області віртуального простору, а вже він отримує своє відображення на плоскому двовимірному зображенні дисплею чи аркуші паперу.

Слід зауважити, що в нас час цілком існують способи візуалізації тривимірного зображення в об’ємному вигляді. Для цього використовуються спеціалізовані пристрої візуалізації, як стерео окуляри, шоломи віртуальної реальності, 3D дисплеї, а в останній час і прототипи голограм – об’ємних зображень світлом, які не потребують поверхні виводу («висять» в повітрі). Однак всі наведені приклади представляють об’ємні характеристики досить умовно і примітивно, а технології імітація відчутності на дотик є ще мало продуктивними, складними та потребують дорогих інструментів. В наш час, останнє існує лише як відокремлений засіб сфери віртуальних розваг.

Тому найбільш розповсюджений підхід візуалізації у двовимірному просторі. Для отримання тривимірного зображення на плоскій поверхні потрібні такі кроки:

- Моделювання – створення тривимірної моделі сцени і об’єктів в ній.
- Текстурування – налаштування властивостей матеріалу моделей через растрові чи процедурні текстури
- Освітлення – встановлення та налаштування джерел світла
- Анімація (за потребою) – створення алгоритмів руху об’єктів в сцені

- Динамічна симуляція (за потребою) – автоматичний розрахунок взаємодії системи частинок та інших симуляцій фізичних явищ

- Рендеринг (або ж візуалізація) – побудова проєкції моделі сцени та її об’єктів відповідно до вибраної фізичної моделі

- Композитінг (або ж компонування) – постопрацювання отриманого зображення з минулого кроку

- Виведення кінцевого зображення на пристрій виведення – монітор або принтер

Текстурування – це проєктування процедурних або растрових текстур на поверхні тривимірного об'єкта. Зазвичай для цього використовується карти UV-координат[22], де кожній вершині моделі ставиться у відповідність певна координата на двовимірному просторі текстури.

Освітлення[23][24] – це процес створення та налаштування віртуальних джерел світла. Налаштування містить точку створення, кут нахилу та інтенсивність випромінювання. Слід зауважити, що інтенсивність може приймати від’ємні значення. Це проявляється в тому, що такі джерела будуть не породжувати, а відбирати світло в зоні радіусі своєї дії.

Як правило використовуються такі типи джерел освітлення:

- Omni light (Point light) – світло розповсюджується рівномірно в певному радіусі

- Spot light – світло має конічну форму контуру. Схоже на освітлення прожектором

- Directional light – джерело світла, що має певний заданий напрямок

- Area light (Plane light) – джерелом світла виступає певна поверхня, тобто відіграє роль світлового порталу

- Photometric – тип освітлення, що базується на передачі об’єктам фізично коректних параметрів матеріалу, таких як коефіцієнт поглинання, температура напруження поверхні тощо.

Слід зазначити, що кількість доступних типів і видів джерел освітлення може сильно варіюватись залежно від програмного засобу, що використовується. Наприклад, можуть бути певні обмеження на обсяг джерел світла, їхню форму тощо.

Анімація (комп'ютерна анімація)[25] – процес створення рухомих зображень, за допомогою комп'ютерних засобів.

Саме анімація є одним з головних втілень покликання комп'ютерної графіки – надання руху, тобто анімації, тривимірному об'єкту або створення імітації цього руху серед інших тривимірних об'єктів (наприклад, коли рухається задній фон сцени створюючи ілюзію руху об'єкта на передньому плані).

Майже всі програмні засоби для роботи з тривимірної комп'ютерною графікою мають широкі можливості роботи з анімаціями.

На етапі створення анімації математична (векторна) просторова модель перетворюється в плоске (растрове) зображення - кадр. Якщо за ціль стоїть створити відео, то рендериться певна послідовність цих кадрів. Зображення на екрані буде представлена як матриця точок, кожна з яких має принаймні три числових параметри – світлової інтенсивності за RGB-каналом, тобто інтенсивності червоного (R), зеленого (G) та синього (B) кольорів. Таким чином, тривимірна векторна структура даних, за допомогою рендеренгу, перетворюється в плоску матрицю пікселів. Цей крок майже завжди вимагає значних потужностей комп'ютера, особливо якщо за ціль стоїть притримування високого рівня деталізації.

Рендеринг[26] – процес отримання зображення, використовуючи певну модель побудови та засобами комп'ютерної графіки.

Зараз є декілька популярних технологій рендеренга. До того ж, вони часто комбінують елементи одне одного.

Наприклад:

- Z-буферизація[27] – алгоритм, що спирається на глибину елементів зображення при створенні 3D-зображень.

- Сканлайн (scanline)[28] – або ж RayCasting («відкидання променю») – алгоритм, що розраховує колір кожної точки зображення спираючись на побудову променю з точки зору камери до першого перетину з поверхнею. Колір отриманого пікселя буде таким же, як і колір перетнутої поверхні. Іноді колір може враховувати і освітлення цієї поверхні.

- Трасування променів (Raytracing)[29] – майже аналогічний сканлайну, але має певні відмінності. По-перше, колір пікселя уточнюється шляхом побудови додаткових променів – відбитих, переломлених. По-друге, використовується лише зворотне трасування погляду. Попри високу якість отримуваного зображення алгоритм є дуже неефективним та потребує значних технічних потужностей. Однак в останній час ця технологія отримує швидке поширення завдяки впровадженню нового покоління відеокарт з виділеними нейромережевими блоками обчислення, що дозволяють перенести всю роботу з обчислення освітлення саме на них[30].

- Глобальне освітлення (Global illumination, radiosity)[31] – алгоритм, що враховує не тільки пряме світло, що надходить безпосередньо від джерела, але й також відбиті промені світла від інших поверхонь моделей сцени. Цей тип освітлення потребує значно складніших алгоритмів обчислювання, а відповідно і потужностей, але й надає більший ступінь реалізму освітлення.

Слід також зауважити, що межа між видами трасування променів в наш майже стерлась. Причиною цього є гібридне використання способів рендеренгу в сучасних засобах роботи з комп’ютерною графікою.

Один з найпростіших алгоритм рендеренга в такому випадку буде сформульований, приблизно, так:

- Побудувати контури моделей на екрані комп’ютера за допомогою проєкції

- Створити ілюзію матеріалів, з яких виготовлені ці моделі

- Розрахувати спотворення цих об’єктів при взаємодії з прозорими середовищами

Моделювання вільного простору сцени містить кілька категорій об’єктів:

- Геометрія – полігональна сітка, сітка кривих, тощо.
- Матеріали – властивості поверхонь моделей.
- Джерела світла – налаштування та створення точок освітлення в сцені.
- Віртуальні камери – локація точки проєкції.
- Сили впливу – налаштування динамічних деформацій об’єктів. Головне призначення – анімації.
- Постефекти – ефекти, що можуть імітувати певні атмосферні явища: хмари, туман, дим, полум’я та інші.

Завдання тривимірного моделювання полягає саме в тому, щоб описати вище вказані об’єкти та розмістити їх в сцені за допомогою геометричних перетворень. При цьому вони повинні відповідати вимогах кінцевого зображення.

1.2. Технології Unity 3d

Unity 3D – це кросплатформний інструмент для розробки двовимірних та тривимірних додатків чи ігор. Створений в 2005 році розробником Unity Technologies.

На даний час — це один з найперспективніших інструментів для розробки власних або комерційних продуктів. Завдяки своїм особливостям Unity має низький поріг входу для початківців та безліч можливостей для вже маючих досвід с схожими продуктами. Велика кількість переваг рушію зробила його одним з самих популярних на ринку створення додатків.

До переваг цього Unity 3D можна віднести:

- Велика кількість платформ, що підтримуються

- Зручний інтерфейс взаємодії. Можна працювати з ресурсами через Drag&Drop. Є можливість налаштувати під себе більшу частину інтерфейсу.

- Внутрішня мова програмування c#, що є досить потужнім та популярним інструментом.

- Зручна система скриптингу.

- Ігровий рушій і середовище є одним цілим, що дозволяє тестувати додатки прямо в редакторі

- Підтримка майже всіх форматів файлів, що пов’язані з розробкою додатків

- Вбудована підтримка мережі та створення WEB-застосунків

- Можливість спільної розробки через Asset Server

- Величезне та активне ком’юніті

- Велика кількість документації та навчальних матеріалів

- Повністю безкоштовний для малих проєктів та навчальних проєктів

Однак існують і недоліки:

- Мало підходить для великих проектів

- Не так популярний серед розробників великомасштабних , як інші рушії, тому більшість пропрієтарних технологій комп’ютерної графіки імплементуються в нових версіях з значним запізненням від конкурентів.

- Потребує відповідального ставлення від програміста на етапі оптимізації продукту

- Багато модулів рушію є дещо застарілим

Однак в рамках теми дослідження нас цікавить саме такі переваги:

- Безкоштовність. Дає можливість нам експериментувати з власним продуктом не думаючи про ліцензію та надалі спокійно використовувати кінцеву програму за власним розсудом, наприклад, як навчальний матеріал

- Зручна та потужна мова програмування. Завдяки c# можна швидко створювати необхідні нам скрипти об’єктів. Враховуючи зрозумілий

синтаксис мови та безліч інструментів роботи з кодом, розробка стає досить ефективною та не потребує постійного мікроменеджменту власного коду

- Зручний, гнучкий та зрозумілий інтерфейс редактору . Дозволяє повністю налаштувати засоби керування під власні вподобання та потреби. До того ж, візуальна робота з об’єктами значно прискорює процес розробки

- Швидке тестування. Завдяки вбудованому в редактор середовищу нам не потрібно постійно збирати проєкт, щоб протестувати власний додаток. Це дозволяє зосередить на розробці та без остраху експериментувати з проектом, не боячись отримати не працюючий проєкт на виході.

- Зручна система ієрархії скриптів та об’єктів. Ми можемо створювати цілі бібліотеки власних скриптів та об’єктів, групуючи за власним розсудом. До того ж, вбудовування скритів в об’єкти через мінімум дій.

- Можливість швидко змінити платформу продукту. Наприклад, якщо ми захочемо прибрати потребу встановлення нашого продукту на носій користувача і розгорнути проєкт як WEB-додаток, то це займе мінімум часу. Це можливо зробити через вбудовану зміну платформи. Однак не слід забувати, що можливі деяка не сумісність на новій платформі, а отже потребує тестування після зміни.

- Велика кількість прикладів реалізації симуляції фізичних об’єктів та явищ. Проаналізувавши їх ми можемо вибрати найбільш компетентні для нашого дослідження та на їхній основі створити власні рішення.

Також слід детальніше описати технічні можливості деяких аспектів Unity, з якими ми будемо безпосередньо працювати.

Скриптова система – головний елемент роботи з об’єктами та в програмному продукті в цілому. Саме через неї ми визначаємо наповнення та властивості сцени, параметри цих об’єктів, їх поведінку, а

також умови їх існування та видалення/зникнення. Завдяки скриптам ми можемо реалізувати все, що нам необхідно для нашого дослідження.

Тому важливо знати можливості і особливості вбудованої скриптової системи Unity 3D. А вона містить такі технічні елементи:

- Скриптова система рушія реалізована через Mono[32] (відкрита реалізація технології .Net Framework)
- Декілька мов для використання, наприклад, власна розробка UnityScript (подібна до JavaScript), більш популярну c# або Boo. Однак, в нових версіях існує офіційна підтримка C# , JavaScript та UnityScript.
- Інтегроване середовище MonoDevelop[33], що дозволяє працювати зі скриптами не маючи додаткових інструментів.
- Починаючи з версії 5.2 з'являється вбудована можливість працювати зі скриптами через Visual Studio
- Зручна та наочна взаємодія зі скриптами в внутрішньому редакторі. Ми можемо зв'язувати скрипти з об'єктами, працюючи з ними, як зі звичайними об'єктами. Тобто, наприклад, переміщати їх через Drag&Drop, працювати з контекстним меню мишкою тощо. Також оголошую перемінні в скриптах, як Serialized, можемо їх редагувати напряму з редактора або просто візуально відстежувати їх зміну.

Рендеринг – ця складова напряму впливає на відображення сцени, її деталізацію та реалістичність. В наш час, коли потужність комп'ютерів дозволяють працювати зі складними алгоритмами опрацювання графіки, вимоги до технологій рушіїв відповідно зросли, а якість зображення рендеру сцени з кожним роком зростає.

Рушій Unity 3D підтримує такі технології рендеру:

- Графічний рушій використовує всі основні гілки розвитку технологій рендеру: DirectX[34](Windows), OpenGL[35] (Mac, Windows, Linux), OpenGL ES[36] (Andorid та iOS) та власне спеціально розроблене API для консолі Wii

- Підтримує новітні алгоритми сучасної комп'ютерної графіки, як bump mapping, paralllax mapping, reflection mapping, SSAO. Також мцстить останні розробки для роботи з тінями (один з найважчих елементів графіки для опрацювання комп'ютерами): динамічні тіні, shadow maps, render-to-texture. Також наявні різні алгоритми для роботи з повноекранними ефектами постопрацювання (post-processing).

- Unity підтримує інтеграцію з більшістю програмних продуктів для роботи з графікою: 3ds Max, Maya, Blender, zBrush, modo, Adobe Photoshop та багато інших. В процесі роботи над ігровим проєктом можна легко імпортувати файли цих програм та працювати з їх налаштуванням через інтерфейс Unity

- Шейдери в Unity реалізовані за допомогою ShaderLab, тобто є підтримка шейдерних програм написаних на GLSL або Cg. Рушій також дозволяє користуватись декількома варіантами реалізації шейдеру, що дає можливість визначати найкращий його варіант для різних відеокарт[37].

- Unity має вбудовану підтримку фізичного рушія Nvidia PhysX, підтримку симуляції одягу та систем ray casts та шарів зіткнення

Проєкти в Unity мають приблизно однакову структуру (технічну), що дозволяє уніфікувати роботу.

Проєкти мають приблизну таку структуру:

- Проєкти діляться на сцени (або ж рівні) – кожна з яких має свій набір скриптів, об'єктів та своє налаштування

- Сцени, своєю чергою, можуть містити як звичайні ігрові об'єкти (моделі), так і порожні – тобто з відсутньою моделлю.

- Об'єкти діляться на компоненти або набори компонентів, з якими надалі подальшому і взаємодіють скрипти

- У кожного об'єкта є ім'я (дозволяється існування декількох об'єктів з одним ім'ям), за потребою йому можна надати тег(тобто мітку), а також задається шар на якому він відображається

- У кожного об’єкта на сцені є обов’язковий компонент Transform – він задає координати розташування, повороту і розміру. Компонент працює у всіх трьох осях (x, y, z).

Отже, беручи в розгляд всі вказані вище параметри, переваги та недоліки, можна з впевненість сказати, що Unity дає можливість нам реалізувати програмний продукт нашого дослідження. Він містить всі необхідні нам компоненти для роботи з логікою об’єктів, графічні технології для візуалізації фізичних явищ, безкоштовний та зручний в використанні, що робить його ідеальним вибором для маленьких проєктів, на кшталт нашого дослідження.

1.3. Основи теми «Електричні явища та електричний струм»

Дослідження автора, окрім реалізації об’єктів для віртуальної лабораторної роботи, має за цілі показати на прикладі окремого фізичного явища особливості проєктування класів об’єктів. Саме унікальні природні властивості диктують особливий підхід до структури та принципів взаємодії.

Сфера природи, що розглядається в дослідженні – це електричні явища та два найбільш характерні його розділи: по-перше, електричний заряд, по-друге, електричний струм.

Однак, ми не можемо продемонструвати симуляції цих явищ, спираючись на саму ж симуляцію. Стає зрозуміли, що для повноцінної демонстрації треба реалізувати низку об’єктів, що взаємодіють з цими фізичними явищами й змінюють певним чином свої параметри. Для цього нам потрібно спиратись на вже наявні приклади таких взаємодій, що були перевірені, достатньо прості в повторенні та потребують роботи лише з самим явищем. Інакше кажучи на класичні експерименти зі струмом.

Класичні експерименти зі струмом – це достатньо прості елементи наукового дослідження, що наочно демонструють дії явища, тому вони

майже завжди наявні в навчальній програмі. Саме тому, доцільно в дослідженні спиратись на приклади з навчальних матеріалів. А враховуючи розрахунок на використання в навчальній сфері, то можна внести уточнення на найпростіший і зрозумілий вид навчального матеріалу – шкільного.

В шкільному курсі фізиці, електричні явища починають розглядати відразу після теплових – тісно пов'язаних з електричними й основи розуміння перших необхідне для розуміння других, однак це не стоїть за ціль дослідження. Треба зауважити, що ця частина курсу зазвичай прилягає на восьмий рік навчання (тобто 8 клас), а одночасно в курсі хімії розглядаються особливості будови атома й основні хімічні зв'язки. Враховуючи, зо згідно з програмою учні майже одночасно, паралельно стикаються з цими явищами та обидва курси доповнюють один одного, що є досить доцільним та логічним рішенням формування навчального плану.

Експерименти в курсі фізики представлені відповідними лабораторними роботами.

Спираючись на типовий підручник з фізики за 8 клас («Фізика. 8 клас . Бар`яхтар», 2016 рік), було вибрано три лабораторних роботи, що демонструють явища електричного струму та цілком підходять для аналізу характеристик симуляції:

- Лабораторна робота № 1 "Вимірювання опору провідника за допомогою амперметра та вольтметра. "
- Лабораторна робота № 2 "Дослідження електричного кола з послідовним з'єднанням провідників. "
- Лабораторна робота № 3 "Дослідження електричного кола з паралельним з'єднанням провідників. "

Умови робіт (умови робіт є дослівним цитуванням тексту умов з підручника):

Лабораторна робота №1

Тема: Вимірювання опору провідника за допомогою амперметра та вольтметра.

Мета: навчитися визначати опір провідника за допомогою амперметра та вольтметра; переконатися на досліді в тому, що опір провідника не залежить від сили струму в ньому та напруги на його кінцях.

Обладнання: джерело струму, резистор, повзунковий реостат, амперметр, вольтметр, ключ, з'єднувальні проводи.

Експеримент.

1. Складіть електричне коло за поданою схемою.

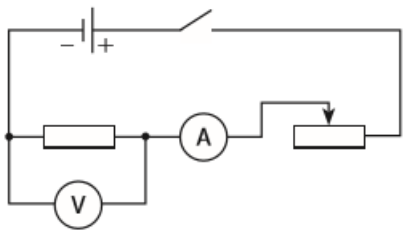


Рис. 1.1

2. Розташуйте повзунок реостата на середині обмотки.
3. Замкніть коло і виміряйте напругу на кінцях резистора та силу струму в ньому.
4. Плавно пересуваючи повзунок реостата, збільшіть силу струму в колі. Запишіть покази вольтметра та амперметра.
5. Пересуньте повзунок реостата у протилежний бік і ще двічі виміряйте напругу та силу струму.

Опрацювання результатів експерименту

1. Обчисліть опір резистора для кожного випадку.
2. Результати обчислень занесіть до таблиці.

Лабораторна робота №2

Тема: Дослідження електричного кола з послідовним з'єднанням провідників.

Мета: експериментально перевірити, що в разі послідовного з'єднання двох провідників справджуються співвідношення: I = I1 = I2; U = U1 + U2 ; R = R1 + R2 .

Обладнання: джерело струму, вольтметр, амперметр, ключ, два резистори, з'єднувальні проводи.

Експеримент

Дослід 1.

Порівняння сили струму в різних ділянках кола, яке містить послідовне з'єднання провідників.

1. Складіть електричне коло за накресленою вами схемою.
2. Виміряйте силу струму, увімкнувши амперметр спочатку між джерелом струму і першим резистором (I1), потім між ключем і другим резистором (I2), а потім між ключем і джерелом струму (I). Накресліть схеми відповідних електричних кіл.

Дослід 2.

Порівняння загальної напруги на ділянці кола, яка складаєтьсяз послідовно з'єднаних резисторів, і суми напруги на окремих резисторах.

1. У колі, складеномудля проведення досліду 1, виміряйте напругу спочатку на першому резисторі (U1), потім на другому резисторі (U2), а потім на обох резисторах (U). Накресліть схеми відповідних електричних кіл.
2. Результати вимірювань занесіть до табл. 2. Закінчіть заповнення таблиці та зробіть висновок.

Лабораторна робота №3

Тема: Дослідження електричного кола з паралельним з'єднанням провідників**.**

Мета: експериментально перевірити, що сила струму в нерозгалуженій частині кола дорівнює сумі сил струмів у відгалуженнях; довести, що загальний опір провідників, з'єднаних паралельно, менший за опір кожного з них.

Обладнання: джерело струму, вольтметр, амперметр, ключ, дві електричні лампи на підставках, з'єднувальні проводи.

Експеримент

1. Зберіть електричне коло за накресленою вами схемою.
2. Виміряйте силу струму I, що проходить у нерозгалуженій частині кола, потім силу струму I1, який тече в лампі 1, та силу струму I2, який тече в лампі 2.
3. Виміряйте напругу U на лампах.
4. Накресліть схеми відповідних електричних кіл.

Отже, маючи умови робіт ми можемо провести їх аналіз та виділити необхідні умови до реалізації класу об'єктів:

1) Реалізувати такі прилади:
   a. Джерело струму
   b. Провідник
   c. Амперметр
   d. Повзунковий реостат
   e. Ключ
   f. Вольтметр
   g. З'єднувальні проводи
   h. Лампа
2) Реалізувати такі характеристики для приладів:
   a. Джерело струму:
      i. Тип джерела (за необхідністю)

ii. Сила струму (I) – константна властивість джерела

iii. Напруга (U)

iv. Потужність P

v. Модель джерела

b. Провідник:

i. Опір провідника

ii. Суму опору при різник типах з’єднання

c. Амперметр:

i. Сила вхідного сигналу

d. Повзунковий реостат:

i. Сила вхідного сигналу

ii. Коефіцієнт вихідного сигналу

iii. Сила вихідного сигналу

e. Ключ:

i. Стан кола

f. Вольтметр:

i. Сила вхідного сигналу

g. Проводи:

i. Сила сигналу

h. Лампа:

i. Сила вхідного сигналу

ii. Сила достатнього сигналу

3) Реалізувати такі взаємодії:

a. Джерело:

i. Джерело – Провід

ii. Замикання джерела

b. Провідник:

i. Провідник – провід

c. Амперметр:

i. Амперметр –провід

ii. Опрацювання та виведення сигналу на шкалу

d. Повзунковий реостат:

i. Реостат – провід

ii. Отримання сигналу через провід

iii. Формація вихідного сигналу згідно з коефіцієнтом вихідного сигналу (шкала поділок)

iv. Виведення вихідного сигналу до наступного проводу

e. Ключ:

i. Ключ – провід

ii. Ключ вимикає/закриває все коло, тобто діє на все джерело

f. Вольтметр:

i. Вольтметр – провід

ii. Опрацювання та виведення сигналу на шкалу

g. Проводи:

i. Вказано в інших приладах

h. Лампа:

i. Лампа – проводи

ii. Ввімкнення лампи при достатньому рівні сигналу

4) Реалізувати такі явища:

a. Подача струму

b. Замикання/вимикання кола

c. Послідовне та паралельне з’єднання провідників

d. Формування параметрів струму, згідно з об’єктами в колі

Тепер сформувавши вимоги до об’єктів класу та симуляції, можна приступити до останнього пункту аналізу – вивчення безпосередньо явища та його особливостей поведінки.

Електричні явища та технології побудовані на принципах їх дії вже давно стали невід’ємною частиною сучасного світу, технологій, науки та нашого повсякденного життя. Саме розуміння природи цих явищ дозволило людству добитись таких технологічних висот. Зараз важко навіть уявити світ без використання електрики, адже саме вона живить майже всі сучасні прилади, в тому чи іншому вигляді. Електричні прилади зробили наше повсякденне життя легшим, підвищило його якісний рівень. Електрика оточує нас повсюди і стала життєво необхідною в кожному аспекті сучасного суспільства:

- Освітлення
- Транспорт
- Приготування їжі
- Різні види зв’язку
- Промисловість та виробництво
- Військова справа
- Наука
- Мистецтво

Вплив на суспільство має настільки значний вплив, що часто змінювало розвиток людство в несподіваних аспектах. Наприклад, електричне освітлення будучи більш дешевим і безпечним за гас та дерево, збільшило тривалість нашого біологічного світлового дня. Тепер є велика кількість людей, що активні в другій половині дня, безліч професій активних вночі, а великі міста, як іноді кажуть, ніколи «не

сплять». Іншим прикладом впливу є такі події, як індустріальна революція (до речі, окрім введення автоматизації праці, електрика збільшивши світловий день, дала змогу збільшити тривалість робочих змін), з'ява інтернету і персональних комп'ютерів, з'ява доступних засобів персонального мережевого зв'язку.

Вклад цього явища в розвиток людства, як технології просто неоціненний.

Так що ж таке «електрика»?

Електрика – це явище природи, що базується на існуванні, русі та взаємодії електричних зарядів[38].

Електричні явища були відомі людству ще в давнини. Так деякі аспекти електрики були відомі древнім грекам, фінікійцям, жителям Межиріччя. Вони знали про такі властивості, як магнетизм та електризація, хоча розуміння загальної природи цих явищ ще не було.

Можна виділити декілька основних вісей в історії дослідження та використання електрики:

- Фалес Мілетський, в 600-х роках до н.е., описує властивості електризації та магнетизму, на прикладі натирання бурштину та притягування ним різних маленьких об'єктів[39]. Однак він сам ще не розуміє різницю між двома явищами, вважаючи за одне.

- Вільям Ґілберт в 1600 році, спираючись на свої дослідження, розмежовує явища магнетизму та електризування. Саме він вводить поняття electricus – бурштиноподібний та робить висновок, що Земля є магнітом [40]

- Середина 17-го століття – Отто фон Ґеріке винаходить електростатичний генератор

- Стівен Ґрей [41] передає електрику за допомогою зволожених ниток, започатковуючи дослідження струмів, закладені основи поділу матеріалів на провідники та діелектрики

- Шарль Дюфе відкриває явища різнойменних зарядів, а також поділив речовини на провідники і ізолятори[42]

- 1752 рік – Бенджамін Франклін встановлює електричну природу блискавки

- 1791 – Луїджі Гальвані відкриває біоелектрику

- 1800 – Алессандро Вольта створює першу батарею – Вольтів стовп

- 1820 – Андре-Марі Ампер відкриває зв'язок між електрикою та магнетизмом

- 1821 – Майкл Фарадей вигадує електродвигун

- 1827 – Георг Ом встановлює математичний закон, що описує струм в електричному колі

- 19 століття — відкриття електромагнітної індукції Фарадеєм робить використання електрики можливим у великих масштабах, що призводить до великої кількості винаходів в цей час

- Кінець 19-го століття – електрика стає провідною силою в другій промисловій революції

- Наш час

Отже, тепер розглянемо основні поняття і визначення явищ нашого дослідження:

Атом будь-якої речовини містить ядро, довкола якого рухаються на своїх орбітах електрони. Вони в свою чергу скріплені завдяки ***електромагнітній взаємодії***.

Електричний заряд (q) — фізична величина, що вказує на властивість тіл та частинок вступати взаємодіяти згідно з законів електромагнітної взаємодії.

$$[q] = 1 \text{ Кл.}$$

Електризація — це процес набуття макроскопічними тілами електричного заряду

Електричний заряд буває двох типів – **позитивний** та **негативний**.

Негативно заряджені частинки відштовхуються, позитивно – навпаки.

Електричний струм — це векторно упорядкований рух заряджених частинок.

Електрична провідність — властивість речовини проводите крізь себе певну кількість електричного струму.

Враховуючи провідність речовини, її можна віднести до **провідників**, **діелектриків** чи **напівпровідників**.

Провідники — речовини, що пропускають значну кількість струму.

Діелектрики — речовини, що пропускають мізерну кількість струму.

Деякі речовини, наприклад, германій відносять до **напівпровідників**. Це матеріали, що змінюють властивість до пропуску струму, залежно від своєї температури.

В нашій віртуальній лабораторії ми будемо використовувати в основному електричні прилади. Тому ми повинні основні поняття побудови таких приладів та їх схематичне зображення.

Кожен електричний прилад має набір обов'язкових елементів. В першу чергу – джерело струму. Наприклад батарея. Батарея має дві точки виводу, тобто полюси. Полюс, на якому накопичується надлишок позитивного заряду, позначають знаком «+».

По-друге, необхідний сам споживач. Джерело струму та споживач з'єднані за допомогою проводів.

І по-третє, вимикачі кола – ключі. Якщо коло замкнуте, то струм йде, живлячи споживачі.

З'єднані елементами проводки в певному порядку джерела струму, споживачі та ключі складають електричне коло.

Електрична схема — це схематичне зображення, що описує побудову електричного кола, його елементів та тип їхнього з'єднання.

.

Деякі умовні позначення, застосовувані на схемах

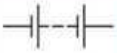
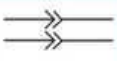
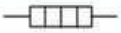
Елемент електричного кола	Умовне позначення
Гальванічний елемент або акумулятор	
Батарея гальванічних елементів або акумуляторів	
З'єднання проводів	
Резистор	
Електричний дзвінок	
Штепсельна розетка	
Перетин проводів (без з'єднання)	
Затискачі для під'єднання якого-небудь приладу	
Ключ	
Електрична лампа	
Нагрівальний елемент	
Запобіжник	

Рис.1.2

Сила струму — це фізична величина, що вказують на швидкість проходження частинок – носіїв стуму, які проходять через поперечний переріз провідника.

$$I = \frac{q}{t}$$

Силу струму вимірюють величиною, що названа в честь французького фізика Андре-Марі Ампера - *амперами* :

[I] = 1 А.

Для вимірювання сили струму використовують прилад, що має відповідну назву - **амперметр**.

На електричних схемах цей прилад позначають латинською літерою A .

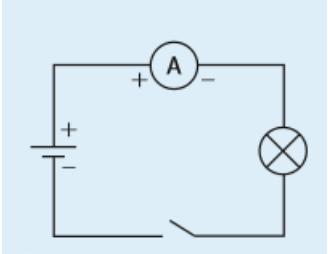


Рис.1.3 Приклад приєднання амперметра до кола

Електрична напруга — це кількісна характеристика кола, що вказує яку роботу виконав електричний при перенесені частинки з зарядом в 1 Кл між двома точками цього кола:

$$U = \frac{A}{q}$$

Напругу вимірюють величиною, що названа в честь італійського фізика та фізіолога, засновника електродинаміки, Алессандро Вольта - *вольтами* :

$$[U] = 1\text{В}$$

Для вимірювання сили струму використовують прилад, що має відповідну назву - **вольтметр**.

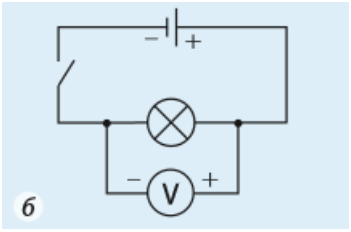


Рис.1.4 Приклад приєднання вольтметра до кола

Закон Ома

Сила струму в колі між двома його точками дорівнює прямо пропорційному відношенню напруги цього кола на його опору

$$I = \frac{U}{R}$$

Електричний опір — це характеристика, що вказує на спроможність провідника в колі чинити супротив дії електричному струму, залежно від матеріалу виготовлення цього провідника.

Опір вимірюють величиною, що названа в честь німецького фізика Георга Ома - *омами* :

[R] = 1 Ом

Існує певна особливість руху струм через провідник, матеріалом якого є один з видів металу. Так як метали мають особливо щільну кристалічну сітку, то частинки, вигляді вільних електронів, проходячи через цей провідник, зустрічають значно більший супротив речовини. Це виражено в формулі:

$$R = p\frac{l}{S}$$

Питомий опір речовини — це властивість речовини, що вказує її електричні властивості. Зазвичай ця характеристика дорівнює опору виготовленого з цієї речови провідника завдовжки n м і площею поперечного перерізу n м2.

Одиниця питомого опору — **ом-метр**:

$$[p] = 1\text{Ом} \cdot \text{м}$$

Реостат — пристрій що дозволяє регулювати силу струму в колі. Принцип його дій спирається на зміну опору самого реостату.

Послідовним з’єднання провідників – це коли в колі відсутні розгалуження віток проводів, а елементи розташовані послідовно один за одним.

Від’єднання будь-якого елементу з такого з’єднання призведе до розімкнення кола. Елементи в такому колі перестають працювати. Також це означає, що сила стума спільна для всіх об’єктів та дорівнює силі струм самого кола

$$I = I_1 = I_2$$

Це справедливо і для напруги

$$U = U_1 = U_2 = \ldots = U_n$$

Якщо є відгалуження, то сума її сили струму її гілок рівна силі струму на нерозгалуженій частині:

$$I = I_1 + I_2 + \ldots I_n$$

У загальному випадку опір R ділянки кола, яка складається з n паралельно з’єднаних провідників, можна обчислити, скориставшись формулою:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} + \ldots + \frac{1}{R_т}$$

Висновок

В цьому розділі ми розглянули весь необхідний нам матеріал для моделювання класів об’єктів електричних явищ. Ця інформація містила пояснення дії вивчаємого фізичного явища, опис технологій створення 3D об’єктів та доступі технології Unity, який ми будемо використовувати для реалізації. Отже, маючи весь необхідний матеріал, проаналізувавши його, ми можемо приступати до наступного розділу, де будемо безпосередньо розробляти класи об’єктів.

РОЗДІЛ 2
РОЗРОБЛЕННЯ ТА ВИКОРИСТАННЯ БІБЛІОТЕКИ ПРОГРАМНИХ МОДУЛІВ ВІРТУАЛЬНИХ ЛАБОРАТОРНИХ РОБІТ

2.1. Визначення структури та особливостей класів

По-перше, на основі попередньо наданого матеріалу, визначимо найбільш характерні вимоги, що диктує фізичне явище:

- Електричне коло завжди має замикатись
- Заряджені частинки завжди рухаються від полюсу «Плюс» до полюсу «Мінус»
- Рух частинок важко безпосередньо візуалізувати, дотримуючись достовірної симуляції явища.
- Однак рух частинок можна візуалізувати або штучними візуальними симулякрами або через безпосередній результат взаємодії з об’єктами кола. Наприклад, зміна показників вольтметра і амперметра або загорання ламп тощо.

Отже, що це дає нам в плані проектування? Ми можемо уточнити деякі вимоги до класів:

- Мати об’єкти полюсів. Наприклад розділити джерело на два об’єкти «плюс» та «мінус» або одне, але з обробкою колізій різних мешів стінки об’єктів.
- Рух частинок можна реалізувати через зміну значень змінних об’єктів під час обробки колізій
- Реалізувавши інтерфейси обробки колізій та передачі даних через них, можна спроектувати будь який новий об’єкт кола

Також стоїть питання з’єднання елементів кола:

- Як і в фізичному аспекті – через введення об’єктів типу «Провід». Саме проводи будуть з’єднувати елементи та характеризувати їх поведінку.

- Доцільніше в класі проводів не обробляти взаємодію з кожним об’єктом, а лише тримати в ньому значення параметрів, що будуть зберігатись для роботи з іншими об’єктами.

- Так як частинки рухаються завжди від «плюсу» до «мінусу», то для найпростішої симуляції кола не потрібно тримати значення вхідного та вихідного сигналів, а лише вхідного. Але якщо потрібно створити симуляцію руху заряду через металевий провідник, то треба враховувати, що струм в них протікає в протилежному напрямку.

Виділимо певні спільні нюанси для всіх об’єктів:

- На етапі початкового проектування, доцільніше використовувати не повноцінні моделі об’єктів, а фізичні примітиви з стандартної бібліотеки Unity, на кшталт кубу, сфера тощо. Ці фігури мають найбільш передбачувану поведінку при симуляції фізики та з ними зручніше працювати. Краще використовувати примітиви, що формою нагадують моделі об’єктів, що вони симулюють. Тобто, амперметр краще представляти у вигляді прямокутника.

- Всі створені об’єкти повинні з’єднуватись один з одним за допомогою проводів. Саме вони забезпечують роботу симуляції кола та передають всю важливу інформацію від одного елемента до іншого

- Всі об’єкти взаємодіють між собою за допомогою власних колізій. Якщо об’єкт не доторкається до жодного з інших об’єктів, то його існування не враховується в колі.

Для зручності типізації об’єктів їх можна розділити за допомогою певного умовного поділу. Автор в своєму дослідженні використовує такий поділ:

- Джерела (Sources) – об’єкти джерела. Є першою необхідною частиною кола. Залежно від реалізації, джерело може існувати як два окремі об’єкти («Source+», «Source-») або як один (якщо різні меші мають різні варіанти колізій)
- Метри (Meters) – вимірювальні прилади, на кшталт, вольтметра та амперметра. Не повинні ніяк не впливати на коло, а лише виводити значення сили сигналу, що отримали
- Проводи (wires) – з’єднувальні частини коли. Виконують транспортну функції в симуляції струму. Не мають власних взаємодій. Зберігають в собі значення, що передаються з одного об’єкта до іншого
- Об’єкти (objects) – об’єкти, що мають в собі певну поведінку при встановленні в колу. Наприклад, лампа, що вмикається, якщо отримую певну силу сигналу. Об’єкти можуть впливати на деякі значення в колі.

Отже можна сформулювати певну загальну схему для симуляції кола:

1) Від джерела «Плюс» йде значення про силу сигналу та запит на замкнення кола та інші можливі параметри
2) Джерело «Плюс» з’єднуючись з проводом передає йому параметри сили сигналу та запит на замкнення
3) Об’єкт проводу зберігає ці значення в власні змінні
4) Провід з’єднуючись з іншим об’єктом, використовуючи взаємодії самого об’єкту, може передати йому значення власних перемінних.
5) Об’єкт опрацьовує власні взаємодії на основі отриманих даних та передає їх далі по колу, використовуючи наступний провід, перезаписуючи вже його параметри
6) В випадку, якщо будь який об’єкт кола перестає взаємодіяти з проводом, викликається функція, що повертає значення змінних цього проводу стандартних

7) Останній провід повинен з’єднуватись з джерелом «Мінус»

8) Якщо джерело «Мінус» отримує значення змінної від проводу, що вказує на послідовне з’єднання всіх об’єктів в колі, починаючи з джерела «Плюс», то струм подається на джерело «Плюс».

2.2. Класи об’єктів. Сцена симуляції

Перейдемо до безпосередньо класів об’єктів, їх коду та взаємодії один з одним.

Спочатку треба почати не з об’єктів фізичного явища, а допоміжних об’єктів сцени:

- Камера - Камера допомагає нам проводити навігацію в симуляції, розглядати її з різних кутів та взаємодіяти з об’єктами в сцені за допомогою мишки.

- В симуляції автора об’єкт камера має назву «Main Camera» та містить три скріпти:

- CameraControl.cs’ – скрипт, що дозволяє керувати переміщенням камери за допомогою клавіш «WASD». Код скрипту з поясненнями наведений в додатку «Код(А.1).»

- Скрипт «MouseLook.cs» - дозволяє управляти кутом камери використовуючи мишку. Код скрипту з поясненнями наведений в додатку «Код(А.2).»

Перейдемо безпосередньо до об’єктів явища:

- «Source» - об’єкт джерело, що симулює позитивну сторону. Код скрипту з поясненнями наведений в додатку «Код(А.3).»

- «Wire»– з’єднує в колі об’єкти, передаючи інформації одних до інших. Не містить власних процедур. Код скрипту з поясненнями наведений в додатку «Код (А.4).

- «Meter» - клас об’єктів, що працює з отриманими даними через проводи. Інтерфейс взаємодії з проводами цього класу дозволяє створити будь який інший клас об’єктів, наслідуючи його методи. Код скрипту з поясненнями наведений в додатку «Код (А.5).

- «SourceM» - клас джерела «Мінус». Обробляє замкнення факт замкнення кола та подачі струму. Код скрипту з поясненнями наведений в додатку «Код (А.6).

- Реалізація таких об’єктів, як вольтметр чи амперметр суттєво не відрізняється від реалізації об’єкту «Meter», тому тут вони не будуть приведені.

- Такі об’єкти, як «Ключ» можуть існувати як дублюючі окремих функцій окремих об’єктів. Так той же ключ виконує проміжну роль проводу по передачі запиту на закриття кола.

- Враховуючи, що в класі «Meter» ми реалізували інтерфейс зв’язку з об’єктів з параметрами кола та їх подальшої передачі, то тепер можливо створити на його основі будь який новий похідний клас. Наприклад, лампи, що буде відрізнятись від «Meter» лише однією функцією перевірки на достатню силу вхідного сигналу і вмикати джерело світла при «True»

Принцип роботи явища колізій:

1) Початковий стан:

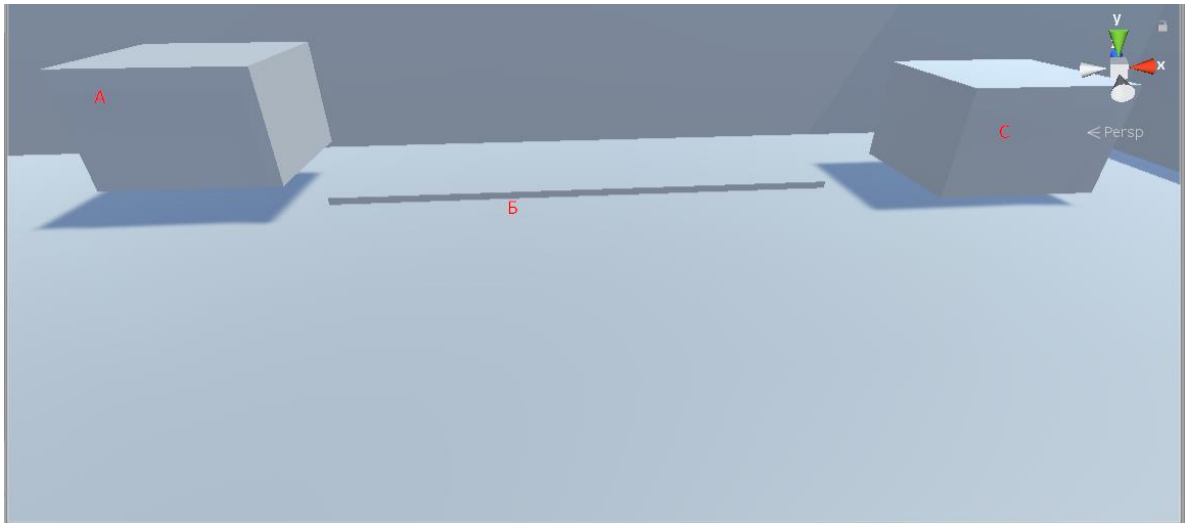


Рис.2.1 Початковий стан сцени

- А – Об’єкт «Meter»

- Б – Об’єкт «Wire»

- С – Об’єкт «Source+»

Як видно, об’єкт не доторкаються і відповідно ніяк не взаємодіють один з одним. На це вказують і їхні параметри:

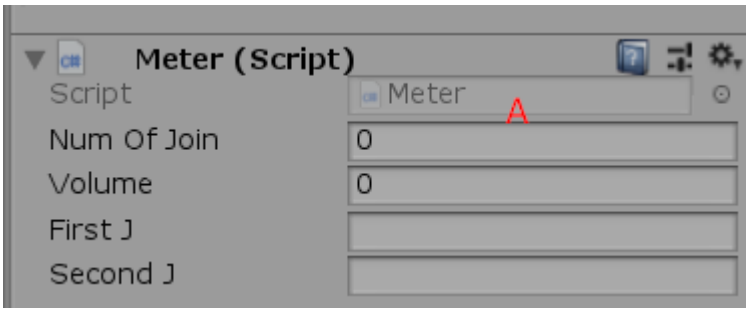


Рис.2.2 Початковий стан «Meter»

Рис.2.3 Початковий стан «Wire»

2) Тепер «доторкнемось» «Wire» до «Source+»:

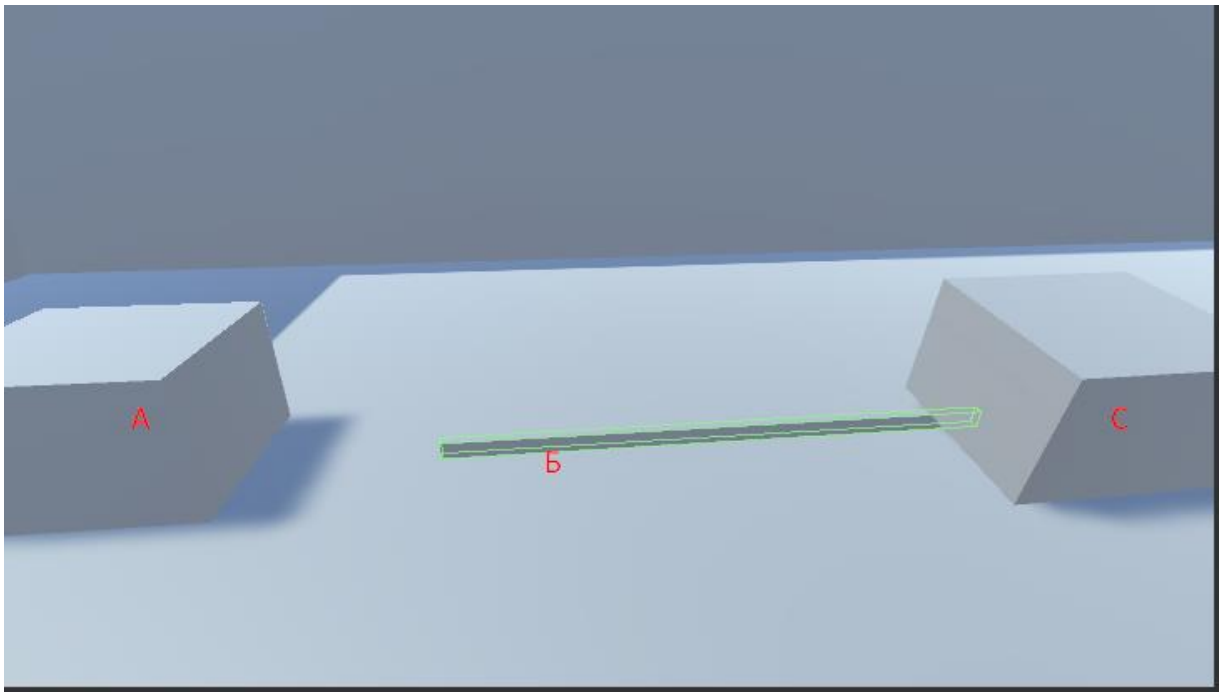


Рис.2.4 Дія колізії «Wire»

Перевіримо параметри «Wire»:

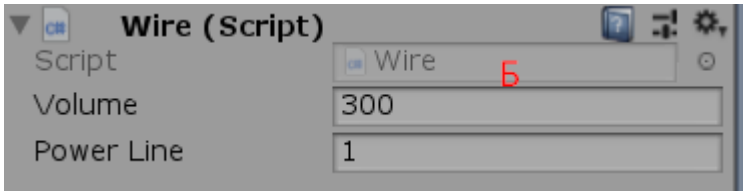


Рис.2.5 Зміна параметрів «Wire»

І дійсно, провід отримує від джерела значення параметрів сили сигналу та запит на замикання кола

3) Таким же чином додаємо в коло об’єкт «Meter»:

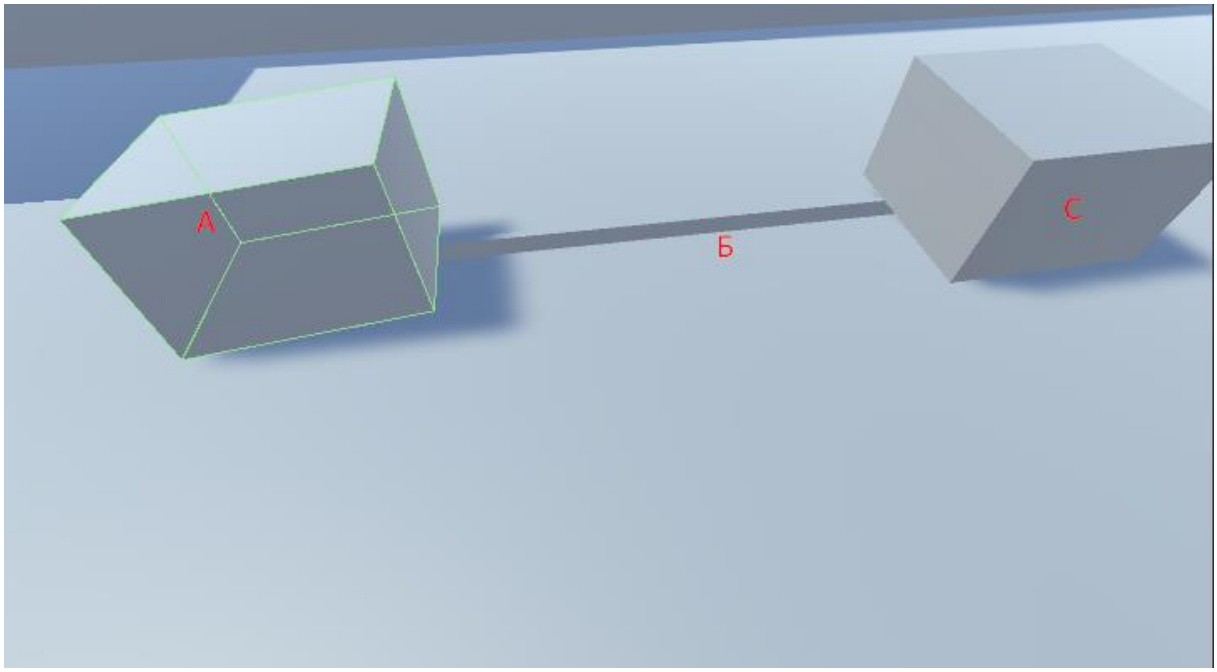


Рис.2.6 Дія колізії «Meter»

Перевіримо значення його параметрів:

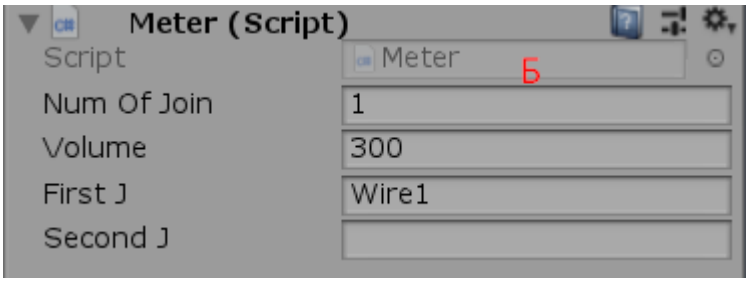


Рис.2.7 Зміна параметрів «Meter»

Вони змінились згідно з проектованими властивостями: вказано дійсно 1 число з’єднань з проводом, передана сила сигналу, та назва цього проводу.

4) Тепер вилучимо провід з кола (Рис.2.8):

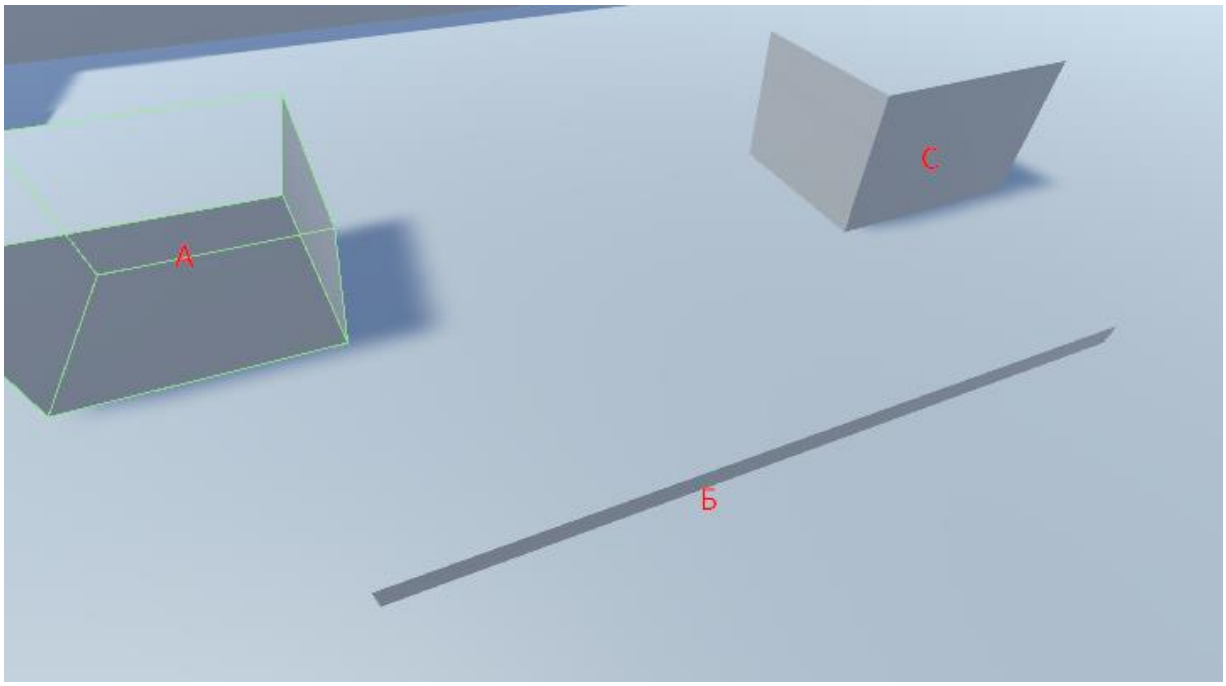


Рис.2.8 Вилучення об’єкта «Wire»

Перевіримо параметри:

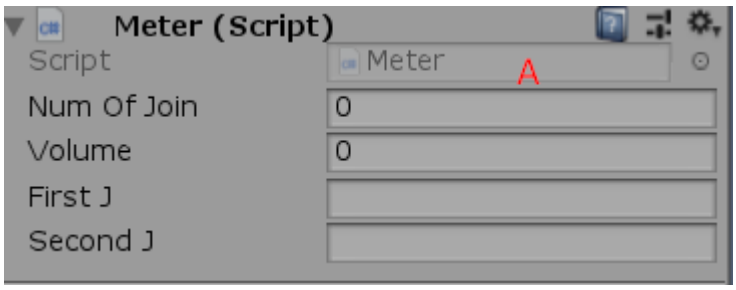


Рис.2.9 Ануляція параметрів «Meter»

Рис.2.10 Ануляція параметрів «Wire»

Отже, ми можемо сказати, що наша симуляція взаємодії через колізії працює.

Однак сцена та об’єкти виконують лише демонстраційну роль в сцені. Для них ще існують певні вимоги візуального характеру.

Так, наприклад, замість примітивів краще використовувати повноцінні 3D моделі потрібних об’єктів.

Приклади такої візуалізації:

Рис.2.11 Вольтметр

Рис.2.12 Амперметр

Рис.2.13 Провід

Також немало важливу роль відіграє візуалізація самої сцени:

Рис.2.14 Приклад візуалізації сцени

Приклад візуалізації сцени. Сцени існує в певному оточенні (кімнаті), а не в Empty Space, має поверхню да розміщення на ній об’єктів. Рендер додає в сцену тіні від наявного джерела світла, а матеріали поводять себе відповідно до заданих параметрів. Наприклад, як деревина моделі столу, що дає мало блиску, а поглинає більшу частину світла. Саме такі деталі візуалізації допомагають покращити сприйняття навчального матеріалу та зосередитись на роботі.

ВИСНОВКИ

Враховуючи актуальність теми дистанційного навчання, різні технології, що реалізують види цього будуть напрочуд актуальні в наш час впродовж тривалого періоду. Саме тому це дослідження вкладає певну, хоч і малу, частину в розвиток цього напряму.

Окрім того, як виявилось в ході вивчення матеріалу з теми «Електричні явища», майже не існує реалізацій віртуальних лабораторій саме з неї. Це також вказує на доречність проведеної роботи.

Також слід зазначити, що технології, які надає рушій Unity 3D, разом з мовою C#, створюють просте та потужне рішення для створення подібних проектів. Саме цей рушій, на думку автора, найбільш підходить для реалізації малих спеціалізованих віртуальних лабораторій. Завдяки своїй гнучкості та підтримці новітній технологій в 3D графіці, він є найперспективнішим засобом підтримки дистанційного навчання, саме в цьому напрямку. Враховуючи, що проект Unity можна розгорнути майже на будь-якій платформі, наприклад, як веб-додаток, що дозволяє скоротити вимоги до матеріальної бази учня майже до мінімуму, рушій є чудовим рішенням не тільки для розробника, а й для тих хто навчається.

Отже, ціль і результат проведеної роботи в дослідженні є напрочуд актуальним в наш час та достатньо доцільним з вибраної теми. Створені інтерфейси взаємодії можуть бути використані не тільки при симуляції електричних явищ, а в будь якій схожій за структурою роботі

Підсумовуючи вище сказане, віртуальні лабораторні роботи є перспективним напрямом дистанційного навчання, а рушій Unity дозволяє цілком розкрити їх потенціал на повну.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lyman P. How much information [Електронний ресурс] / P. Lyman, H. R. Varian // Release of the University of California – Режим доступу: https://web.archive.org/web/20180219162428/http://chnm.gmu.edu/digitalhistory/links/pdf/preserving/8_5a.pdf.

2. Вільна енциклопедія «Вікіпедія» - Дистанційне навчання [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Дистанційне_навчання

3. Вільна енциклопедія «Вікіпедія» - Unity (рушій гри) [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Unity_(рушій_гри)

4. Мультимедийная виртуальная лаборатория по физике в системе дистанционного обучения / Е. О. Козловский, Г. М. Кравцов // Інформаційні технології в освіті. - 2014. - Вип. 18. - С. 80-89. Режим доступу: http://nbuv.gov.ua/UJRN/itvo_2014_18_11

5. Фізика : підруч. для 8 кл. загальноосвіт. навч. закл. / [В. Г. Бар'яхтар, Ф. Я. Божинова, С. О. Дов гий, О. О. Кірюхіна] ; за ред. В. Г. Бар'яхтара, С. О. Довгого. — Х. : Вид-во «Ранок», 2016. — 240 с. : іл., фот. ; ISBN 978-617-09-2855-9

6. Lvov M. About One Approach to Building Systems for Testing Physical Knowledge [Електронний ресурс] / M. Lvov, S. Kuzmenkov, H. Kravtsov // ICTERI 2019. Part I: 4th International Workshop on Professional Retraining and Life-Long Learning using ICT: Person-oriented Approach (3L-Person 2019).. – 2019. – Режим доступу: http://ceur-ws.org/Vol-2393/paper_265.pdf.

7. Unity – Manual [Електронний ресурс] – Режим доступу: https://docs.unity3d.com/Manual/index.html

8. LabView – About:Product preview [Електронний ресурс] – Режим доступу: https://www.ni.com/ru-ru/shop/labview.html

9; 10. Вільна енциклопедія «Вікіпедія» - LabVIEW [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/LabVIEW

11. JMCAD– Home [Електронний ресурс] – Режим доступу: http://jmcad.sourceforge.net/index_us.shtml

12. JMCAD– Files [Електронний ресурс] – Режим доступу: https://sourceforge.net/projects/jmcad/files/

13. Виртуальные лаборатории и технические симуляторы– Виртуальная лаборатория общей физики [Електронний ресурс] – Режим доступу: https://www.sunspire.ru/products/physics2d/

14. Вільна енциклопедія «Вікіпедія» - Фотограмметрія [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Фотограмметрія

15. Вільна енциклопедія «Вікіпедія» - Моделювання твердих тіл [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Моделювання_твердих_тіл

16. Вільна енциклопедія «Вікіпедія» - Теселяція [Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/Теселяція

17. Вільна енциклопедія «Вікіпедія» - Полігональне моделювання [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Полігональне_моделювання

18. Вільна енциклопедія «Вікіпедія» - NURBS [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/NURBS

19. Вільна енциклопедія «Вікіпедія» - Сплайн[Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Сплайн

20. Вільна енциклопедія «Вікіпедія» - Геометричний примітив [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Геометричний_примітив

21. Вільна енциклопедія «Вікіпедія» - Комп’ютерна структура [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Комп'ютерна_скульптура

22. Вільна енциклопедія «Вікіпедія» - Текстура (тривимірна графіка) [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Текстура_(тривимірна_графіка)

23. Вільна енциклопедія «Вікіпедія» - Освітлення (комп’ютерна графіка)[Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/Освітлення_(комп'ютерна_графіка)

24. Unity – Manual: Lighting Overview [Електронний ресурс] – Режим доступу: https://docs.unity3d.com/ru/current/Manual/LightingOverview.html

25. Вільна енциклопедія «Вікіпедія» - Комп’ютерна анімація [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Комп'ютерна_анімація

26. Вільна енциклопедія «Вікіпедія» - Рендеринг [Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/Рендеринг

27. Вільна енциклопедія «Вікіпедія» - Z-буферизація[Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Z-буферизація

28. Вільна енциклопедія «Вікіпедія» - Ray casting[Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Ray_casting

29. Вільна енциклопедія «Вікіпедія» - Трасування променів[Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/Трасування_променів

30. Вільна енциклопедія «Вікіпедія» - GeForce 20[Електронний ресурс]. – Режим доступу:https://ru.wikipedia.org/wiki/GeForce_20

31. Вільна енциклопедія «Вікіпедія» - Глобальне освітлення [Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/Глобальне_освітлення

32. Mono – Home [Електронний ресурс] – Режим доступу: https://www.mono-project.com/

33. MonoDevelop – Home [Електронний ресурс] – Режим доступу: https://www.monodevelop.com/

34. Вільна енциклопедія «Вікіпедія» - DirectX[Електронний ресурс]. – Режим доступу:https://uk.wikipedia.org/wiki/DirectX

35. Вільна енциклопедія «Вікіпедія» - OpenGL [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/OpenGL

36. Вільна енциклопедія «Вікіпедія» - OpenGL ES [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/OpenGL_ES

37. Unity – Manual: Shaders[Електронний ресурс] – Режим доступу: https://docs.unity3d.com/Manual/Shaders.html

38. Літ.: Тамм И. Е. Основы теории электричества. 9-е изд. Москва, 1976; Калашников С. Г. Электричество. Москва, 1977; Сивухин Д. В. Общий курс физики. Т. 3. Москва, 1977; A. Morely, E. Hughes. Principles of Electricity. 5th ed. Harlow, 1994; J. Bird. Electrical and Electronic Principles and Technology. 3rd ed. Elsevier, 2007.

39. Thales. All Things are Full of God [Електронний ресурс] / Thales – Режим доступу: https://www.iep.utm.edu/thales/#H7.

40. Гильберт У. О магните, магнитных телах и большом магните — Земле. М.: АН СССР, 1956.- 412 с.

41. Храмов Ю. А. Грей Стефен (Gray Stephen) // Физики: Биографический справочник / Под ред. А. И. Ахиезера. — Изд. 2-е, испр. и дополн. — М.: Наука, 1983. — С. 91. — 400 с. — 200 000 экз. (в пер.)

42. Храмов Ю. А. Дюфе Шарль Франсуа (DUFAY Charles Francois de Cisternay) // Физики: Биографический справочник / Под ред. А. И. Ахиезера. — Изд. 2-е, испр. и дополн. — М.: Наука, 1983. — С. 109. — 400

ДОДАТКИ

Додаток А

Код (А.1):

```
using System.Collections;// Підключаємо необхідні бібліотеки Unity
using System.Collections.Generic;
using UnityEngine;

public class CameraControl : MonoBehaviour /*клас CameraControl наслідує методи і влстивості класу
MonoBehavior
- одного з основних класів в Unity*/
{
    public int CameraMoveSpeed = 4; //Параметр, що відповідає за швидкість руху
    private Vector3  MousePos;// Змінна для маніпулювання координатами розміщення камери


public GameObject obj; //Посилання на об'єкт, що буде використовувати скріпт

void Update(){ // Update - функція, що буде опрацьовуватись кожне оновлення кадру
     MousePos = Input.mousePosition;
    if(Input.GetKey(KeyCode.W))
                                obj.transform.Translate(Vector3.forward * CameraMoveSpeed *
Time.deltaTime);//рух камери вперед
     if(Input.GetKey(KeyCode.S))
                                obj.transform.Translate(Vector3.back * CameraMoveSpeed *
Time.deltaTime);//рух камери назад
     if(Input.GetKey(KeyCode.A))
                                obj.transform.Translate(Vector3.left * CameraMoveSpeed *
Time.deltaTime);//рух камери вбік ліворуч
     if(Input.GetKey(KeyCode.D))
                                obj.transform.Translate(Vector3.right * CameraMoveSpeed *
Time.deltaTime);/*рух камери вбік
            праворуч*/
    }
}
```

Код (А.2):

```
using UnityEngine;
using System.Collections;
public class MouseLook : MonoBehaviour {
 public enum RotationAxes {
 MouseXAndY = 0,
 MouseX = 1,
 MouseY = 2
 }
 public RotationAxes axes = RotationAxes.MouseXAndY;
 public float sensitivityHor = 9.0f;
 public float sensitivityVert = 9.0f;
 public float minimumVert = -45.0f;
 public float maximumVert = 45.0f;
 private float _rotationX = 0;
 void Start() {
 Rigidbody body = GetComponent<Rigidbody>();
 if (body != null)
 body.freezeRotation = true;
 }
 void Update() {
 if (axes == RotationAxes.MouseX) {
 transform.Rotate(0, Input.GetAxis("Mouse X") * sensitivityHor, 0);
 }
else if (axes == RotationAxes.MouseY) {
 _rotationX -= Input.GetAxis("Mouse Y") * sensitivityVert;
 _rotationX = Mathf.Clamp(_rotationX, minimumVert, maximumVert);
 float rotationY = transform.localEulerAngles.y;
 transform.localEulerAngles = new Vector3(_rotationX, rotationY, 0);
 }
 else {
 _rotationX -= Input.GetAxis("Mouse Y") * sensitivityVert;
 _rotationX = Mathf.Clamp(_rotationX, minimumVert, maximumVert);
 float delta = Input.GetAxis("Mouse X") * sensitivityHor;
 float rotationY = transform.localEulerAngles.y + delta;
 transform.localEulerAngles = new Vector3(_rotationX, rotationY, 0);
 }
 }
}
```

Код (А.3):

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Source : MonoBehaviour
{
    [SerializeField]/*Параметр вказує на те, що змінні внизу можуть бути доступні з редактору через
    інтерфейс
    */
    public int customV = 300; //Значення сили сигналу, що передається
     public int  power = 0;// Вказує на те, що струм йде
    public int inVol = 0;//Запит на замкнення кола


void Start(){//функція запускається кожного разу при запускі симуляції
        StartCoroutine (CheckCicle());//Запуск корутіни, що перевіряє замкнутість кола
    }
    IEnumerator CheckCicle(){//Означення перевіряючої корутіни
        while (true){
        var tempV = GameObject.Find("SourceM");//Шукаємо в нашій симуляції об'єкт джерела Мінус
        var tempS = tempV.GetComponent <SourceM>();//Отримуємо дані об'єкта "SourceM", тобто джерела Мінус
        if (tempS.startP == 1){//Якщо коло замкнуто
            power = customV;//Струм подається з силоб вказаною в CustomV
            }
        else if (tempS.startP == 0){
            power = 0;//Якщо коло розімкнуте - струм зникає
        }
        yield return new WaitForSeconds (1.1f);}//перевірка буде викликатись черз кожну 1.1 сек
    }
   private void OnCollisionStay(Collision col){//Процедура входу в колізію об'єкта Wires

       if (col.gameObject.tag =="Wire"){
            string ntf = col.gameObject.name;
            var tempVol = GameObject.Find (ntf);
            var tempScrpt = tempVol.GetComponent<Wire>();//передаємо дані на провід
            tempScrpt.volume = inVol; //Передаємо запит на замкнення коло в провід
            tempScrpt.powerLine = 1; //Передаємо стан кола далі
        }

    }
    if (col.gameObject.tag =="Wire"){
            string ntf = col.gameObject.name;
            var tempVol = GameObject.Find (ntf);
            var tempScrpt = tempVol.GetComponent<Wire>();//Передаємо дані на провід
            tempScrpt.volume = 0;//Зкидуємо значення силу сигналу, що передає провід
            tempScrpt.powerLine = 0;//Передаємо запит на розімкнення кола
        }
}

}
```

Код (А.4):

```
public class Wire : MonoBehaviour
{

    [SerializeField]//Означає, що ми маємо доступ до змінних з редактору (для тестування)
    public int volume = 0; //Змінна, що вказує на силу сигналу
    public int powerLine = 0;//Змінна, що вказує на стан зімкнення кола

}
```

Код (А.5):

Частина 1

```
public class Meter : MonoBehaviour
{


    [SerializeField]//Означає, що ми маємо доступ до змінних з редактору (для тестування)
    private int numOfJoin = 0;//Кількість поточних з'єднаннь (колізій з проводами)

    public int Volume = 0; //Значення силу сигналу, що отримується
    public string firstJ = "";//Ім'я об'єкту першого з'єднання
    public string secondJ = "";//Ім'я об'єкту другого з'єднання

private void OnCollisionStay(Collision col){//Процедура входу в колізію об'єкта Wires

       if ((col.gameObject.tag =="Wire") & (numOfJoin == 0)){//Обробка першого входу об'єкту
            string ntf = col.gameObject.name;
            var tempVol = GameObject.Find (ntf);
            var tempScrpt = tempVol.GetComponent<Wire>();
            if ((tempScrpt.volume >0)&(tempScrpt.powerLine == 1)){//Значення змінних, якщо провід передає
не порожні данні
                Volume = tempScrpt.volume;
                numOfJoin+=1;
                firstJ = ntf;
            }
            if (tempScrpt.volume == 0){//Якщо провід нічого не передає
                tempScrpt.volume =Volume;
                secondJ = ntf;
                numOfJoin+=1;

            }
       }
        if ((col.gameObject.tag =="Wire")&(numOfJoin==1)){//Обробка другого входу об'єкту
             var tempVolS = GameObject.Find (stf);
             var tempScrptS = tempVolS.GetComponent<Wire>();
             if ((tempScrptS.volume == 0)& (stf !=firstJ)){//Це об'єкт другий та нічого не передає
                secondJ =stf;
                tempScrptS.volume = Volume;//передаємо йому власні значення
                tempScrptS.powerLine = 1;
                numOfJoin+=1;
             }
```

Код (А.5):

Частина 2

```
        private void OnCollisionExit(Collision col){//Опрцювання різних комбінацій виходу проводів

            if ((col.gameObject.tag =="Wire") & (numOfJoin == 1)&(col.gameObject.name ==firstJ)){
                 string ntf = col.gameObject.name;
                 var tempVol = GameObject.Find (ntf);
                 var tempScrpt = tempVol.GetComponent<Wire>();
                 Volume = 0;
                 firstJ = "";
                 numOfJoin-=1;
             }
            if ((col.gameObject.tag == "Wire") & (numOfJoin == 1)&(col.gameObject.name == secondJ)){
                 string ntf = col.gameObject.name;
                 var tempVol = GameObject.Find (ntf);
                 var tempScrpt = tempVol.GetComponent<Wire>();
                 tempScrpt.volume = 0;
                 secondJ = "";
                 numOfJoin-=1;
             }
            if ((col.gameObject.tag =="Wire") & (numOfJoin == 2)&(col.gameObject.name ==firstJ)){
                 Volume = 0;
                 firstJ = "";
                 numOfJoin-=1;
              }
            if ((col.gameObject.tag == "Wire") & (numOfJoin == 2)&(col.gameObject.name == secondJ)){
                 string ntf = col.gameObject.name;
                 var tempVol = GameObject.Find (ntf);
                 var tempScrpt = tempVol.GetComponent<Wire>();
                 tempScrpt.volume = 0;
                secondJ = "";
                 numOfJoin-=1;
             }

      }
      }
```

Код (А.6):

Частина 1

```
{
    [SerializeField]//Означає, що ми маємо доступ до змінних з редактору (для тестування)
   public int inputS = 0;//Стан замкнення кола
   public int startP = 0;//Команда на ввімкнення струму


  void Start  (){
       StartCoroutine(checkInput());//Запуск корутіни
   }
      IEnumerator checkInput(){//Корутіна, що перевіряє стан джерела
       while (true){
       if (inputS == 1){//Коли сигнал про замнутість кола дійшов
           startP =1;//Подаємо команду на запуск струму
       }
       else if (inputS == 0){//Немає сигналу
           startP =0;//Немає струму
       };

       yield return new WaitForSeconds (1.1f);//Виконуємо корутіну кожні 1.1 секунди
       }
     }
```

Код (А.6):

Частина 2

```
private void OnCollisionStay(Collision col){//Процедура входу в колізію об'єкта Wires

        if (col.gameObject.tag =="Wire"){
             string ntf = col.gameObject.name;
             var tempVol = GameObject.Find (ntf);
             var tempScrpt = tempVol.GetComponent<Wire>();//Отримуємо дані з проводу
             if (tempScrpt.powerLine == 1){//Якщо є сигнал про замикання кола
                 inputS = 1;//Коло замкнулось
             }
             else if (tempScrpt.powerLine == 0){//Сигналу про замкнення немає
                 inputS =0;//Струм не подаєтсья
             }
         }

    }
private void OnCollisionStay(Collision col){//Процедура входу в колізію об'єкта Wires

        if (col.gameObject.tag =="Wire"){
             string ntf = col.gameObject.name;
             var tempVol = GameObject.Find (ntf);
             var tempScrpt = tempVol.GetComponent<Wire>();//Отримуємо дані з проводу
             if (tempScrpt.powerLine == 1){//Якщо є сигнал про замикання кола
                 inputS = 1;//Коло замкнулось
             }
             else if (tempScrpt.powerLine == 0){//Сигналу про замкнення немає
                 inputS =0;//Струм не подаєтсья
             }
         }

    }
```

Додаток Б

КОДЕКС АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ
ЗДОБУВАЧА ВИЩОЇ ОСВІТИ ХЕРСОНСЬКОГО
ДЕРЖАВНОГО УНІВЕРСИТЕТУ

Я, Соболь Микола Вікторович,
учасник(ця) освітнього процесу Херсонського державного університету, **УСВІДОМЛЮЮ**, що академічна доброчесність – це фундаментальна етична цінність усієї академічної спільноти світу.

ЗАЯВЛЯЮ, що у своїй освітній і науковій діяльності **ЗОБОВ'ЯЗУЮСЯ**:

– дотримуватися:

- вимог законодавства України та внутрішніх нормативних документів університету, зокрема Статуту Університету;
- принципів та правил академічної доброчесності;
- нульової толерантності до академічного плагіату;
- моральних норм та правил етичної поведінки;
- толерантного ставлення до інших;
- дотримуватися високого рівня культури спілкування;

– надавати згоду на:

- безпосередню перевірку курсових, кваліфікаційних робіт тощо на ознаки наявності академічного плагіату за допомогою спеціалізованих програмних продуктів;
- оброблення, збереження й розміщення кваліфікаційних робіт у відкритому доступі в інституційному репозитарії;
- використання робіт для перевірки на ознаки наявності академічного плагіату в інших роботах виключно з метою виявлення можливих ознак академічного плагіату;

– самостійно виконувати навчальні завдання, завдання поточного й підсумкового контролю результатів навчання;

– надавати достовірну інформацію щодо результатів власної навчальної (наукової, творчої) діяльності, використаних методик досліджень та джерел інформації;

– не використовувати результати досліджень інших авторів без використання покликань на їхню роботу;

– своєю діяльністю сприяти збереженню та примноженню традицій університету, формуванню його позитивного іміджу;

– не чинити правопорушень і не сприяти їхньому скоєнню іншими особами;

– підтримувати атмосферу довіри, взаємної відповідальності та співпраці в освітньому середовищі;

– поважати честь, гідність та особисту недоторканність особи, незважаючи на її стать, вік, матеріальний стан, соціальне становище, расову належність, релігійні й політичні переконання;

– не дискримінувати людей на підставі академічного статусу, а також за національною, расовою, статевою чи іншою належністю;

– відповідально ставитися до своїх обов'язків, вчасно та сумлінно виконувати необхідні навчальні та науково-дослідницькі завдання;

– запобігати виникненню у своїй діяльності конфлікту інтересів, зокрема не використовувати службових і родинних зв'язків з метою отримання нечесної переваги в навчальній, науковій і трудовій діяльності;

– не брати участі в будь-якій діяльності, пов'язаній із обманом, нечесністю, списуванням, фабрикацією;

– не підроблювати документи;

– не поширювати неправдиву та компрометуючу інформацію про інших здобувачів вищої освіти, викладачів і співробітників;

– не отримувати і не пропонувати винагород за несправедливе отримання будь-яких переваг або здійснення впливу на зміну отриманої академічної оцінки;

– не залякувати й не проявляти агресії та насильства проти інших, сексуальні домагання;

– не завдавати шкоди матеріальним цінностям, матеріально-технічній базі університету та особистій власності інших студентів та/або працівників;

– не використовувати без дозволу ректорату (деканату) символіки університету в заходах, не пов'язаних з діяльністю університету;

– не здійснювати і не заохочувати будь-яких спроб, спрямованих на те, щоб за допомогою нечесних і негідних методів досягати власних корисних цілей;

– не завдавати загрози власному здоров'ю або безпеці іншим студентам та/або працівникам.

УСВІДОМЛЮЮ, що відповідно до чинного законодавства у разі недотримання Кодексу академічної доброчесності буду нести академічну та/або інші види відповідальності й до мене можуть бути застосовані заходи дисциплінарного характеру за порушення принципів академічної доброчесності.

23.04.20		Микола Соболь
(дата)	(підпис)	(ім'я, прізвище)