МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Програмний комплекс симуляції бойової обстановки. Захищена клієнтська частина.»

Завідувач
випускаючої кафедри **Довбиш А.С.**

Керівник роботи **Ларик Т. В.**

Студента групи КБ – 61 **Яскевич Б.О.**

СУМИ 2020

ЗМІСТ

ВСТУП..... 3

1 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ СИМУЛЯТОРА ТА ПРОБЛЕМИ РОЗРОБКИ СИСТЕМИ СИМУЛЯЦІЇ ВЕДЕННЯ ВОГНЮ З АРТИЛЕРІЇ 4

1.1 Проблеми розробки системи симуляції ведення..... 4
вогню з артилерії..... 4

1.2 Огляд існуючих програмних рішень..... 8

1.2 Постановка задачі 14

2 ВИБІР МЕТОДУ РІШЕНЬ ПОСТАВЛЕНИХ ЗАДАЧ 18

2.1 Вибір рушія для створення симулятора..... 18

2.2 Вибір програмного забезпечення для створення Інтерфесу
..... 19

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМЛЕКСУ СИМУЛЯЦІЇ БОЙОВОЇ ОБСТАНОВКИ..... 20

3.1 Проектування прототипу інтерфейсів Керівника, Командира батареї, Старшого офіцера батареї, Командира гармати, Розвідника та Розвідника-далекомірника 20

3.2 Інтерфейс..... 26

3.3 Ландшафт 32

3.4 Моделі цілей 34

3.5 Реалізація Алгоритмів 35

ВИСНОВКИ 39

СПИСОК ЛІТЕРАТУРИ 40

ДОДАТОК..... 41

ВСТУП

Комп'ютерні ігри - це програми, які були розроблені задля розваг людей на комп'ютері. Такі ігри отримали феноменально потужний розвиток і за останні тридцять років вони мають величезний попит у споживачів. Багато людей грають в різного роду шутери, MMORPG, RPG, стратегії, симулятори та інші жанри комп'ютерних ігор. Чи не менше людей готові віддавати гроші за такого роду розваги. Це весело і цікаво. З розвитком комп'ютерної електроніки і покращенням програмного і апаратного забезпечення, ігри стають кращими і реалістичнішими, все більше наближаючись за якістю візуалізації до реального світу. Можна випускати першокласних фахівців використовуючи симулятори для проходження тестів, які максимально наближені до реального життя.

Simulator («Симулятор») – окремий жанр відеоігор, в якому велика увага приділяється точності імітації реальної дії. Авто симулятор - симулятор гонок, підвид, який використовується для навчання водіїв і гонщиків. Існують авіа-симулятори, мото-симулятори, симулятори підводних човнів. Окремий напрямок - спортивні симулятори. У нашому випадку розробляється Артилерійський симулятор. Який призначений для навчання артилеристів.

Робота присвячена розробці інтерфейсу, локацій, моделей транспорту і деяких елементів для розрахунку дальності до об'єкта та розробки декількох алгоритмів випадково вибору. Зокрема розробки Артилерійського симулятора на сучасному кросс-платформенному движку для створення ігор і додатків.

Для нормальної роботи користувача з додатком потрібен інтуїтивно зрозумілий і привабливий інтерфейс. Так само однією з поставлених завдань було розробити три різних види ландшафту: море, рівнини і гори. Потрібно реалізувати пристрій Далекомір, який показує скільки метрів до цілі та декілька алгоритмів випадкового вибору значень показників.

1 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ СИМУЛЯТОРА ТА ПРОБЛЕМИ РОЗРОБКИ СИСТЕМИ СИМУЛЯЦІЇ ВЕДЕННЯ ВОГНЮ З АРТИЛЕРІЇ

1.1 Проблеми розробки системи симуляції ведення вогню з артилерії

У зв'язку з поступовим переходом армії на новий спосіб комплектування, скороченням терміну військової служби, а також недостатність фінансування перед Збройними силами України гостро постало питання навчання особового складу з використанням навчально-тренувальних комплексів. З цією метою розпочато виконання комплексу заходів з розвитку навчальної матеріально-технічної бази Збройних сил. Зокрема, розпочато впровадження в процес навчання особового складу новітніх тренажерів, лазерних імітаторів стрільби, навчально-тренувальних комплексів, заснованих на використанні комп'ютерних технологій.

Дійсно, жодна зі складних і дорогих військово-технічних систем не зможе ефективно функціонувати без добре навченого персоналу. при підготовці фахівців для роботи па таких системах виникає ряд проблем. По-перше, безпосереднє навчання на реальній бойовій техніці і в умовах, наближених до бойових, нерідко стає неможливим в силу економічних причин. По-друге, деякі фрагменти бойової роботи розрахунків для безлічі можливих ситуацій важко відтворювати. При цьому, чим більш досконалим виявляється озброєння, тим важче його використовувати в навчальних цілях.

Для вирішення зазначених проблем слід переглянути ставлення до тренажерів і імітаційним комплексам. Вони повинні перейти з категорії допоміжного, другорядного обладнання в одну з категорій, що визначають боєздатність сучасної армії, і стати інструментом підготовки військ і розробки військової техніки.

Інтерес до тренажерних систем виник давно, проте з кінця 1990-х років він отримав новий розвиток. Яким би досконалим не був сучасний зразок або

комплекс озброєння, які б засоби автоматизації в ньому не були присутні, в будь-якому випадку людина є найважливішою і визначальною складовою цієї системи, причому цю складову необхідно відповідно готувати і навчати.

В даний час рівень підготовки особового складу багато в чому залежить і від рівня початкового навчання та підготовки, одержуваних в першу чергу з використанням технічних засобів. З огляду на дуже вже недовгий термін «корисною» служби сучасного захисника вітчизни, ця проблема стоїть досить гостро.

Другою складовою тренажерного буму стало бурхливий розвиток і впровадження комп'ютерних та інформаційних технологій, які в поєднанні з наявною науково-технічної, технологічної та виробничої базою забезпечили розробку і створення тренажерів нового покоління. це покоління тренажерів побудовано па базі персональних комп'ютерів, обладнано системою імітації обстановки, яка враховує спеціалізацію об'єкта навчання, і автоматизованої системою об'єктивного контролю в реальному масштабі часу.

Другою складових тренажерного буму стало Бурхливий розвиток и Впровадження комп'ютерних та інформаційних технологій, які в поєднанні з наявною науково-технічною, технологічною та виробничої базою забезпечує розробку і створення тренажерів нового покоління. Це покоління тренажерів побудували на базі персональних комп'ютерів, оснащених системою імітації обстановки, яка враховує спеціалізацію об'єкта навчання, і автоматизованої системою об'єктивного контролю в реальному часовому масштабі.

Багатогранність і складність процесів комп'ютерного навчання, забезпечення потреби в наявності високоефективних засобів відображення інформації не дозволяє обмежитися тільки ЄОМ. Необхідний цілий комплекс обчислювальних станцій і електронних пристроїв, об'єднаних в локальні обчислювальні мережі, в сукупності що представляють собою комп'ютерно тренажерний комплекс або комп'ютерну тренажерно-навчальну систему.

Як правило, такі комплекси або системи підрозділяються на дві групи:

- комп'ютерні системи навчання та інтелектуального тренаж, призначені для забезпечення теоретичного і перед тренажерного етапу підготовки;
- тренажери, що забезпечують формування професійних навичок.

Перша група з достатньою точністю повторює реальний зразок техніки, зовнішній вигляд її елементів, апаратури тощо. Використовуючи їх, ті, яких навчають військовослужбовці можуть виробляти певні маніпуляції, т. е. здійснювати роботу на віртуальному тренажері, як і на реальному зразку озброєння в повному обсязі, без обмежень. Друга група дозволяє викладачеві або інструктору моделювати для учнів різні ситуації бойової обстановки і відстежувати їх дії, фіксувати їх помилки, гнучко управляти їх діями в реальному часі, нарощувати складність завдань. Слід зазначити, що поділ комп'ютерних тренажерів на зазначені вище групи певною мірою умовне, оскільки кожен з них може бути переорієнтований на рішення інших завдань.

Ще однією проблемою є широкий спектр завдань професійної підготовки фахівців ракетних військ і артилерії, який повинен вирішувати комп'ютерний тренажерний комплекс, а саме:

- можливість вивчення призначення, пристрої та принципу дії як окремих механізмів, вузлів і агрегатів різних зразків, так і систем озброєння в цілому, включаючи і боєприпаси;
- можливість моделювання органів (пультів) управління різних систем озброєння па одних і тих же апаратних засобах автоматизованого робочого місця учня;

можливість надання кого навчають всього комплексу інформації (візуальної, аудіо і т.д.), змінною в реальному масштабі часу;

Вести дебати щодо необхідності впровадження тренажерно-навчальних систем для підготовки фахівців ракетних військ і артилерії. Однак в даній си-

туації насамперед необхідно зробити вибір продовжувати навчання «по-старому», на реальній техніці із застосуванням реальних боєприпасів і величезними матеріальними і фінансовими витратами, або, використовуючи навчальні комп'ютерні системи, абсолютно відмовитися від застосування реальних боєприпасів в ході бойової (індивідуальної) підготовки розрахунків, підвищити ефективність їх підготовки, зберегти технічний ресурс і без того зношеної матеріальної частини та уникнути значних фінансових і тимчасових витрат.

Більшість країн пострадянського простору рішуче йдуть шляхом тренажерів. Зокрема, в Росії вже кілька років успішно використовується навчальна система «КОНКУРС-ТМ», що дозволяє готувати операторів однойменних установок ПТУР. заняття за допомогою даних навчальних систем проводяться в спеціалізованих комп'ютерних класах і розраховані на 120-годинну індивідуальну підготовку військовослужбовця. Керівниками занять, як правило, є командири взводів і сержанти. системи також дозволяють займатися самостійно. Після здачі комплексного іспиту військовослужбовець допускається до практичних (тренажерним) стрільб.

Таким чином, широке і активне впровадження навчально-тренувальних комплексів, заснованих на використанні комп'ютерних технологій, в інтересах підготовки фахівців ракетних військ і артилерії дозволить:

- відтворювати процеси експлуатації для безлічі ситуацій, реалізація яких на реальних зразках озброєння є недоцільною або неприпустимою;
- економити ресурс техніки і витрачаються матеріальні засоби на її експлуатацію;
- подолати психологічну боязнь учнів на початковому етапі освоєння нової техніки;

1.2 Огляд існуючих програмних рішень

Огляд інструментів для розробки додатка

Unity

Unity - це міжплатформенний ігровий рушій, розроблений Unity Technologies, вперше оголошений та випущений у червні 2005 року на Всесвітній конференції розробників Apple Inc. як ігровий рушій Mac OS X. Станом на 2018 рік, рушій був розширений для підтримки понад 25 платформ. Цей інструмент може бути використаний для створення тривимірної або двовимірної, віртуальної реальності та ігор з доповненою реальністю, а також моделюванню. Даний рушій був прийнятий у галузях, що не входять до відеоігор, таких як кіно, автомобілебудування, архітектура, інженерія та будівництво.

Можна створювати інтернет-додатків за допомогою спеціалізованого інструменту для браузера Unity. Додатки, що були створені за допомогою цього рушія, підтримують такі графічні інтерфейси як DirectX та OpenGL.

Unity є рушієм багато-платформенним. Редактор Unity підтримується в Windows та macOS, з версією редактора, доступною для платформи Linux, хоча і в експериментальній стадії, а сам двигун на даний момент підтримує побудову ігор для більш ніж 25 різних платформ, включаючи мобільні та настільні комп'ютери , консолі та віртуальна реальність. Платформи включають iOS, Magic Leap, Android, Facebook Gameroom, ARKit від Apple, WebGL, Tizen, Windows, універсальна платформа Windows, 3DS, Cardboard, Steam VR, Mac, Oculus RiftGoogle Cardboard, PlayStation Vita, Samsung Smart TV, PlayStation 4, Oculus RiftGoogle Facebook Gameroom, Gear VR, змішана реальність Windows , Daydream, Android TV, LinuxNintendo Switch, Fire OS, ARKit від Apple, Google ARCore, PlayStation VR, Vuforia, Xbox One та Magic Leap.

Редактор Unity має доволі простий і комфортний інтерфейс, який можна легко налаштовувати, що складається з багатьох вікон, завдяки чому мо-

жна налагоджувати умови гри прямо в Unity редакторі. Рушій може підтримувати три мови програмування: JavaScript (модифікація), C #. Проект в Unity поділяється на деякі сцени — різні файли які знаходяться окремо, та містять свої ігрові умови.

Приклад роботи Unity(мал. 1)

Рисунок 1 – Інтерфейс Unity

Переваги:

- C #. Дана мова високорівнева і дозволяє програмісту комфортно розробляти ігри. Це суттєвий момент, тому що на відміну від інших движків, де використовується мова C ++, в C # є багато елементів і прийомів, які вже реалізовані, і програмісту потрібно тільки використовувати їх.
- Кросс-платформенність, це один і той же код, написаний на движку Unity, з мінімальними змінами може бути перенесений на різні платформи (PC, Mac, Android, iOS, Web, ігрові консолі). Це величезний плюс, що скорочує час для розробки програмного додатка в кілька разів.

- Хороше Community. Це означає, що у різних функцій движка є чіткий опис з прикладами на головному сайті рушія, звернутися до якого можна в будь-який момент. Або якщо є якась не зрозуміла до кінця річ, служба підтримки обов'язково відповість на виниклі питання.

Недоліки:

- Вартість Pro-версії. Для рядового або новачка фрілансера 4500 $ за ліцензію (Pro-ліцензія Unity + iOS Pro + Android Pro) для того, щоб можна було скористатися всіма «плюшками» Pro-версії і публікуватися в iOS- і Android-маркетах, досить дорого. Але недавно на GDC в Сан-Франциско був реліз Unity 5, за який не потрібно абсолютно точно платити і він є безкоштовним, без відрахувань з отриманого доходу. У версії Personal Edition недоступні тільки додаткові сервіси, такі як Unity Analytics Pro, Unity Cloud Build Pro і деякі інші. Дана версія найбільше підійде початківцям розробникам і невеликим компаніям. Якщо все-таки є необхідність покупки Pro-версії, то Unity пропонує її за 75 $ в місяць.

Unreal Engine

Unreal Engine - це ігровий рушій, розроблений компанією Epic Games, вперше продемонстрований у 1998 році шутер від першої особи Unreal. Хоча спочатку розроблений для шутерів від першої особи, він з успіхом застосовується в багатьох інших жанрах, включаючи платформи, бойові ігри, MMORPG та інші RPG. Написаний на C ++, Unreal Engine відрізняється високим ступенем портативності, підтримуючи широкий спектр платформ. Останній реліз - Unreal Engine 4, який запустився в 2014 році за моделлю підписки. З 2015 року її можна завантажити безкоштовно, її вихідний код доступний на GitHub. Epic дозволяє використовувати його в комерційних продуктах на основі роялті, зазвичай просять розробників 5% доходу від продажів, хоча з успіхом Fortnite, який став пробним для Unreal Engine for Epic, Epic ві-

дмовляється від цієї плати розробникам, які публікують свої ігри через магазин Epic Games Store. Компанія Epic оголосила, що Unreal Engine 5 буде випущений до кінця 2021 року.

Unreal Engine першого покоління був розроблений Тімом Свіні, засновником Epic Games. Створивши інструменти для редагування умовно-популярних ігор ZZT та Jill of the Jungle. Свіні почав писати рушій у 1995 році для виготовлення гри, яка згодом стане шутером від першої особи, відомою як Unreal. Після років розвитку він дебютував з випуском гри в 1998 році, хоча MicroProse та Legend Entertainment отримали доступ до технології набагато раніше, ліцензуючи її в 1996 році.

Unreal був відзначений своїми технічними нововведеннями, але Свіні в інтерв'ю Eurogamer визнав, що багато аспектів гри не заполіровані, посилаючись на скарги на її високі системні вимоги та проблеми з онлайн-ігровим процесом. Розвиток Unreal Tournament дозволило Epic вирішити ці моменти, включаючи декілька вдосконалень двигуна призначені для підвищення продуктивності та мережевого коду при одночасному вдосконаленні штучного інтелекту в ботах для відображення координації в командному середовищі. Крім того, що він доступний в Microsoft Windows, Mac, Linux і Unix, рушій-переносився через Unreal Турнір до PlayStation 2 і, за допомогою секретного рівня, до Dreamcast.

На даний момент вже вийшло 4 версії цього рушія, але Epic Games вже анонсували 5 версію, яка буде доволі реалістичною.

Приклад роботи Unreal Engine(мал. 2)

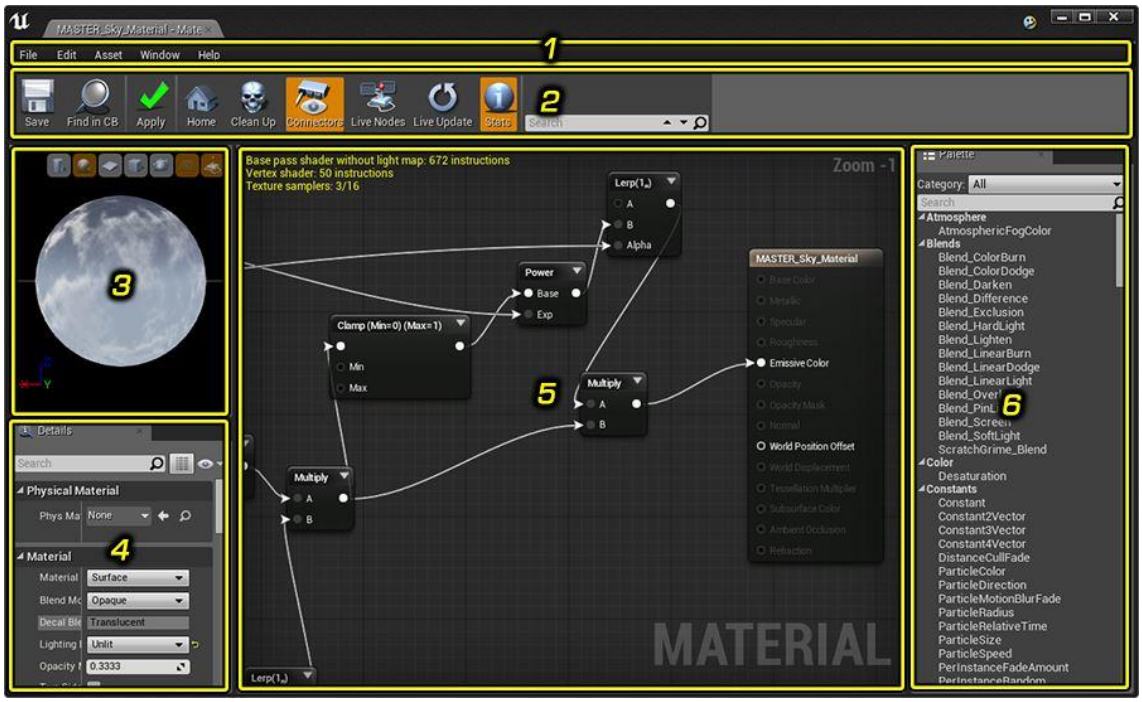


Рисунок 2 – Інтерфейс Unreal Engine

Пояснення до рисунку 2:

1. Menu Bar - Меню з функціоналом, пресуще редактору матеріалів.
2. Toolbar (Тулбар) - панель з необхідними або найбільш важливими інструментами роботи з матеріалами.
3. Viewport (вьюпорте) - Вікно з попередню оплату переглядом вашого матеріалу на примітиві чи моделлю.
4. Details - Список параметрів матеріалу або обраного блоку.
5. Graph (Графік) - Показує мережу виразів (блоків) вашого матеріалу. На даній панелі проводиться основна робота над матеріалом.
6. Pallete - Список всіх доступних виразів.

Переваги:

- В Unreal Engine 4 використовується мова програмування C ++.

- Повна версія Unreal Engine 4 буде безкоштовною для вас, поки дохід від гри становить менше $ 3 000 на квартал, якщо ж більше вам доведеться ділитися 5% від своїх доходів;
- Відкритий код;

Недоліки:

- В магазині додатки набагато менше ассетів;

Microsoft Visio

Microsoft Visio - це програмне забезпечення для малювання різних діаграм: блок-схем, організаційних діаграм, планів будівель, поверхових планів, діаграм потоків даних, технологічних схем, моделювання бізнес-процесів, swim lane блок-схем, 3D-карт.

Можливості Visio:

- Графічне оформлення схем. За допомогою засобів Visio можна задати різні ефекти для фігур на схемах процесів, вибрати теми оформлення схем, змінювати фігури, зберігаючи макети схем і метадані фігур;
- Спільна робота над схемами. Використовуючи веб-браузер можна організувати загальний доступ до перегляду схем. При додатковій установці SharePoint Server і Microsoft Lync 2013 у користувачів з'являється можливість коментувати схеми, здійснювати спільну роботу з ними і обмінюватися повідомлення;
- Взаємозв'язок схем з наборами даних. Кожну фігуру зі схеми можна пов'язати з набором даних з Excel, SharePoint, служби SharePoint Business Connectivity Services і SQL Server. Для наочного уявлення даних можна використовувати велику кількість графіків і колірних схем;
- Створення схем за допомогою стандартних нотацій. Для перевірки коректності створюваних схем в Visio вбудовані правила, що дозволяють контролювати правильність застосування елементів.

Ці правила задані для стандартних нотацій, таких як BPMN. При необхідності, такі правила можна задавати самостійно;

1.2 Постановка задачі

Інтерфейс — межа розділу двох систем, пристроїв або програм, певна їх характеристика, характеристиками з'єднання, сигналів обміну і тому подібне. Сукупність уніфікованих програмних і технічних правил і засобів, що забезпечують взаємодію програм і приладів в обчислювальній. Поняття інтерфейсу поширюється і на системи, які не є обчислювальними або інформаційними.

Призначений для користувача інтерфейс - інтерфейс, що забезпечує передачу інформації між користувачем-людиною і програмно-апаратними компонентами комп'ютерної системи.

Засоби і методи інтерфейсу користувача:

Засоби

- вивід інформації з пристрою для користувача - весь можливий діапазон впливів на організм людини (слухових, зорових, нюхових, тактильних, і т. д.) - екрани (проектори, дисплеї,) і динаміки, лампочки, сирени і зумери або наприклад вібромотор і т.д. і т.п;
- введення будь яким способом інформації або команд людини в пристрій - багато різних датчиків для контролю стану користувача – перемикачі, кнопки, датчики положення і руху, сервоприводи, потенціометри, жести особою і руками, або знімання мозкової активності користувача;

При наявності різних засобів введення, інтерфейси поділяються на типи - голосовий, жестова, брейн, і т.д., також цілком можливі варіанти коли використовуються декілька типів інтерфейсів.

Методи

- це набір деяких правил, заснованих розробниками приладу, згідно з якими сукупність впливу користувача повинно привести до певної реакції пристрою і виконання потрібного завдання - так званий логічний інтерфейс;

Безпека

Одним з основних напрямків досліджень в області забезпечення безпеки користувальницьких інтерфейсів, і, зокрема, візуальних інтерфейсів користувача, є розробка моделей інформаційної безпеки за умови комплексного обліку інформаційних, функціональних, психофізіологічних і екологічних аспектів безпеки. Це пов'язано, перш за все, з включенням інформаційного фактора до складу факторів середовища систем людина-комп'ютер і інформаційним характером майже всіх що відбуваються в області поширення ІП процесів.

Найменш розробленими областями проблематики захисту інформації в системі людина-комп'ютер (СЛК) відповідають такі загрози, як:

- спотворення сприймається користувачем інформації за рахунок її зашумлення джерелами середовища на робочому місці користувача;
- втрата або спотворення сприймається користувачем інформації через фізичну, семантичної або синтаксичної неузгодженості її уявлення користувачеві;
- спотворення уявлень користувача про реальний стан об'єкта управління за рахунок прихованих інформаційних впливів і неадекватне прийняття ним рішень в процесі вирішення завдань в рамках СЛК.

Інтерфейс

Для нормальної навігації користувача в програмі. Потрібен якісний і зрозумілий інтерфейс. Для того щоб його розробити слід спочатку спроектувати, а потім перенести симулятор. У даній програмі потрібно реалізувати інтерфейс взаємодії клієнта з симулятором. Також необхідно створити три форми на головному екрані: «Початок гри», «Налаштування» и «Вихід».

«Початок гри» відповідає за відкриття вікна в якому потрібно ввести ір сервера і ім'я гравця. Після цього після цей гравець вибирає свою роль і підключається до сесії.

«Налаштування» відповідає за відкриття вікна в якому можна регулювати гучність в головному меню і в самій грі.

«Вихів» відповідає за вихід з программи.

Також варто спроектувати та перенести в Unity такі інтерфейси:

- Інтерфейс Керівника **«Адміна»** який буде виставляти умови бойових дій;
- Інтерфейс Керівника батареї, той хто буде виконувати бойове завдання;
- Інтерфейс Розвідника-далекомірника;
- Інтерфейс Старшого офіцера батареї;

Ландшафт

У симуляторі так само повинен бути вибір ландшафту (карти) ведення бойових дій. Потрібно створити три місцевості з різними бойовими умовами: рівнини, гірська місцевість і морські умови.

Моделі цілей

Потрібно створити моделі цілей як для наземної місцевості, так і морської.

Далекомір

Для знаходження відстані до цілі потрібно створити пристрій «Далекомір», який буде показувати відстань в метрах.

Далекомір - пристрій, призначений для визначення відстані від спостерігача до об'єкта. Використовується в геодезії, для наведення на різкість в фотографії, в прицільних пристосуваннях зброї, систем бомбометання і т.д.

Алгоритм визначення кутомірів по ОТН, ЗТН, НТН

Даний алгоритм потрібен для знаходження випадкових значень для різних точок наведення.

Алгоритм визначення інтервалу, уступу та перевищення гармати відносно основної гармати

Даний алгоритм визначає інтервал от ступу та перевищення гармати відносно основної, якщо використовується більше одної гармати.

Алгоритм визначення кутів гребня

Даний алгоритм потрібен для знаходження випадкових значень кутів гребня.

2 ВИБІР МЕТОДУ РІШЕНЬ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Вибір рушія для створення симулятора

Для розробки гри був обраний Unity3d, бо Unity вважається не просто движком, а більше ідеальним місцем для розробки ігор для ПК. У програмі є такі засоби, які використовуються для розробки ПО. Ну а завдяки простому і зрозумілому оформлення створення ігор стає дійсно простим і комфортним. Завдяки мультіплатформеру створення ігор стає ще і універсальним.

Unity дозволяє розробляти гри без спеціальних навичок для цього. Розробнику потрібно створити об'єкт, а потім створити засоби, в яких він буде існувати. Програма дає можливість створювати і відразу розставляти об'єкти в реальному часі, при цьому можна протестувати результат. У движка є ще один плюс - величезний список Ассет і плагінів. Вони допоможуть прискорити процес створення гри. Також є можливість їх експорту і імпорту, додаючи відразу цілі рівні. Тут не доведеться проводити ніяких дій, що стосуються програмування. У вас вийде створити власний контент, завантажити його і можливо навіть заробити на розробці унікальних деталей.

Третім плюсом програми - підтримка відразу безлічі платформ, API. Ігри, створені в програмі, підійдуть для проектів на різних платформах. Також Unity працює з DirectX і OpenGL. Вона чудово поєднується з різними системами рендеринга і ще масу різних технологій. За допомогою движка можна створити найрізноманітніші ефекти. Існує думка, що цей движок створювався для інді-ігор і не може видати красиву реалістичну картинку не вірно. Творці движка випустили спеціально відео, яке показує на що він здатний, і результати вражають.

І що найголовніше - програма безкоштовна, так що цей факт відкриває масу можливостей для незалежних розробників. Хоча тут є і деякі обмеження. На безкоштовній версії перед запуском іграшки демонструватиметься логотип движка. Крім того, проект на безкоштовній версії не повинен принести розробнику більше ніж 100 тис. Доларів на рік. Ну а платна версія коштує

всього-то 125 доларів на місяць, так що і цей варіант не стане марнотратним навіть для початківців розробників.

2.2 Вибір програмного забезпечення для створення Інтерфесу

Для проектування був вибраний програмний продукт Microsoft Visio. Тому що він має такі переваги як:

- Швидкий початок роботи, при запуску пропонується набір вбудованих схем і підказки по контексту, що допомагає користувачеві швидко створити, відредагувати і закінчити схему;
- Продуктивна робота, Visio містить безліч фігур, які відповідають різним галузевим стандартам, включаючи BPMN 2.0, UML 2.4 і IEEE. Дана перевага допоможе з дотриманням всіх вимог побудувати організаційну діаграму, задокументувати бізнес-процес, намалювати сучасний план поверху або швидко замалювати блок-схему з дошки;
- Проста зв'язок схем з даними, можна пов'язати схеми з такими джерелами даних, як Excel, Active Directory, SharePoint і SQL Server. Схеми, пов'язані з даними, можуть оновлюватися автоматично і відображати різні іконки, символи і кольори для відображення змін лежать в основі їх даних;

3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМЛЕКСУ СИМУЛЯЦІЇ БОЙОВОЇ ОБСТАНОВКИ

3.1 Проектування прототипу інтерфейсів Керівника, Командира батареї, Старшого офіцера батареї, Командира гармати, Розвідника та Розвідника-далекомірника

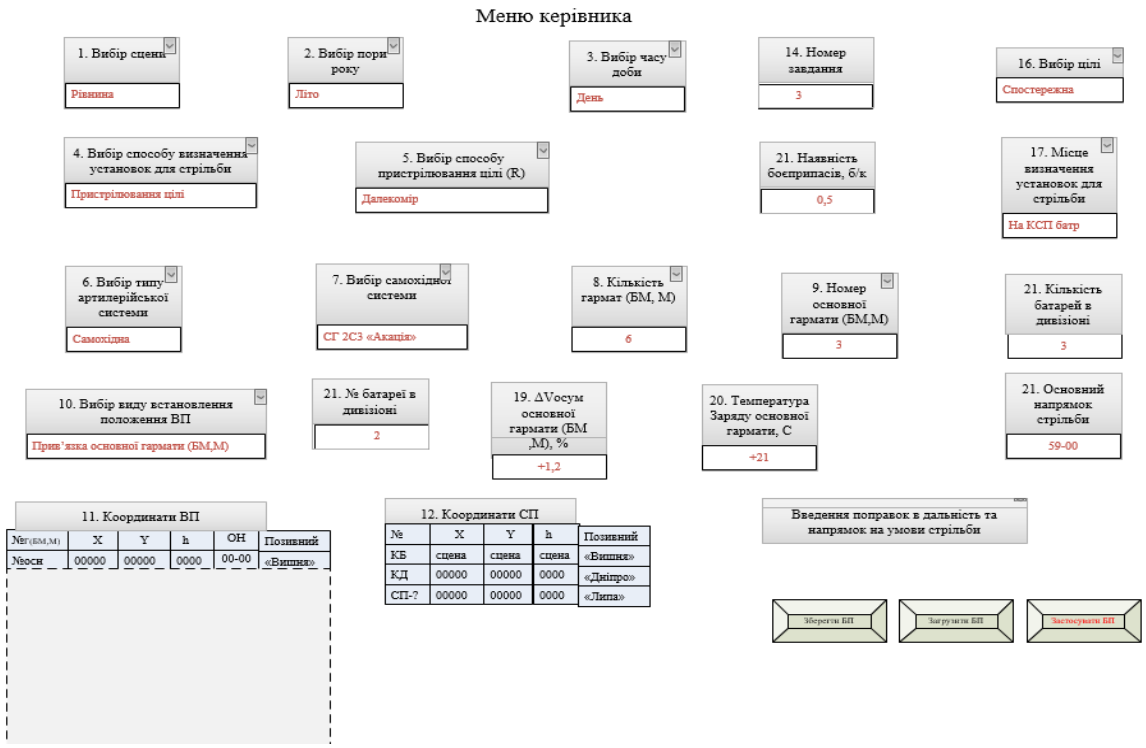


Рисунок 3 - Інтерфейс Керівника(прототип)

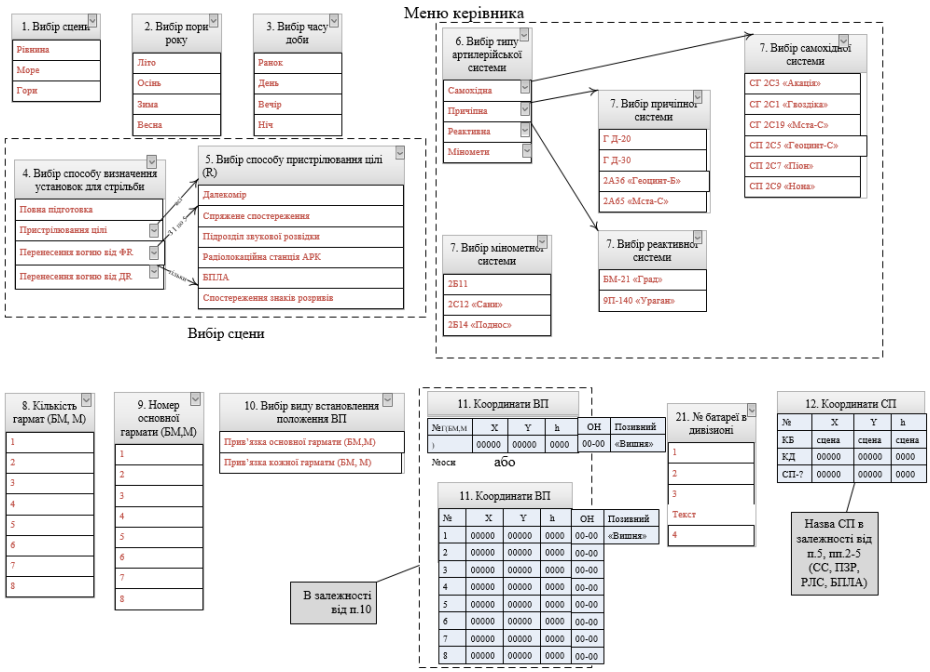


Рисунок 3.1 - Пояснення до інтерфейсу Керівника(прототип)

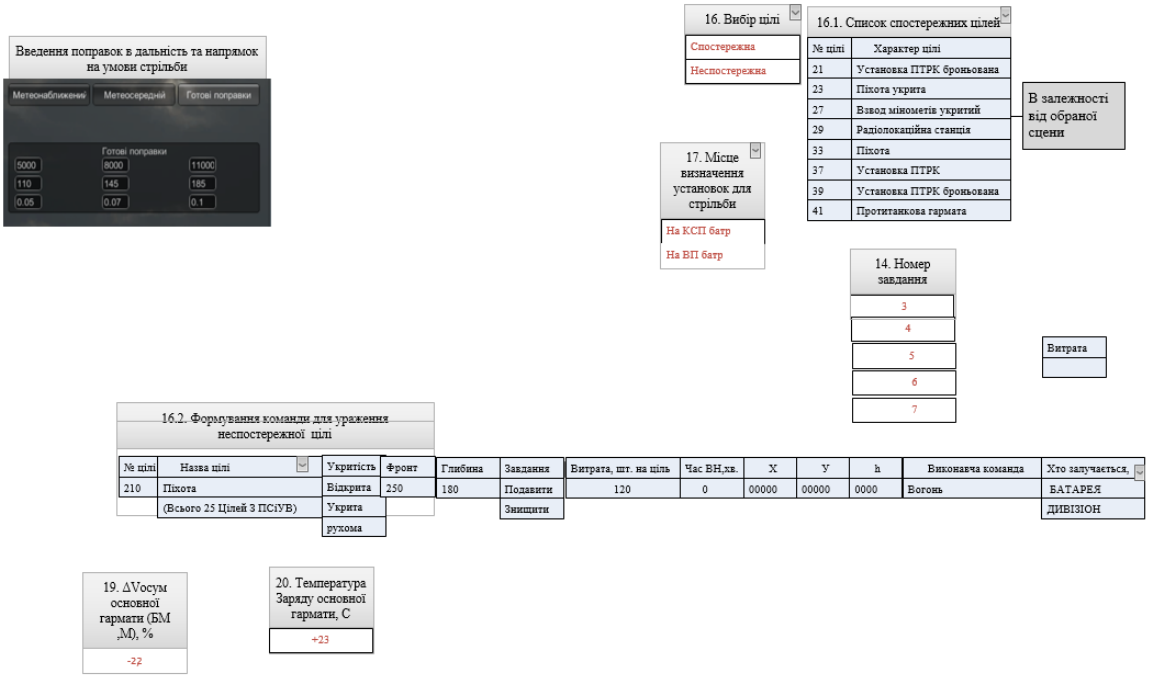


Рисунок 3.2 - Пояснення до інтерфейсу Керівника(прототип)

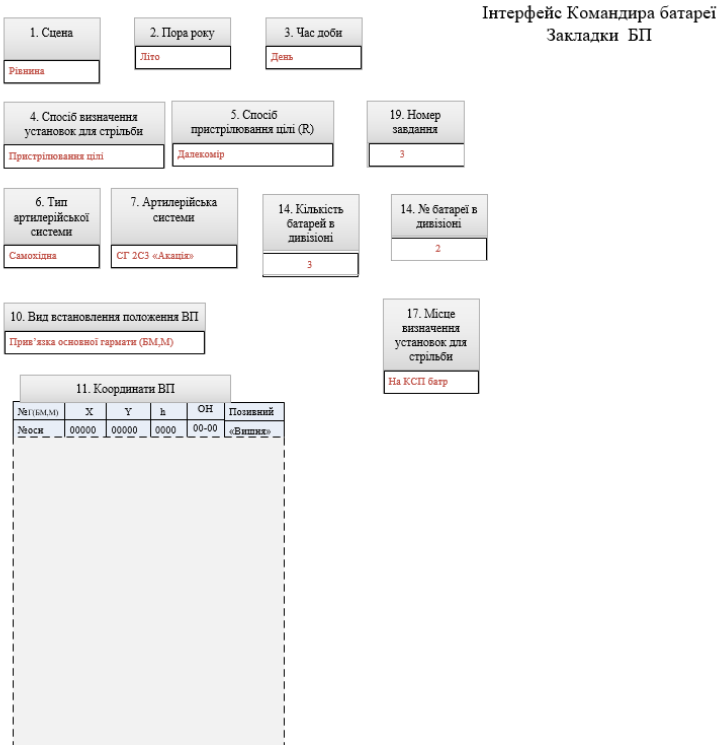


Рисунок 4 - Інтерфейс Командира батареї(прототип)

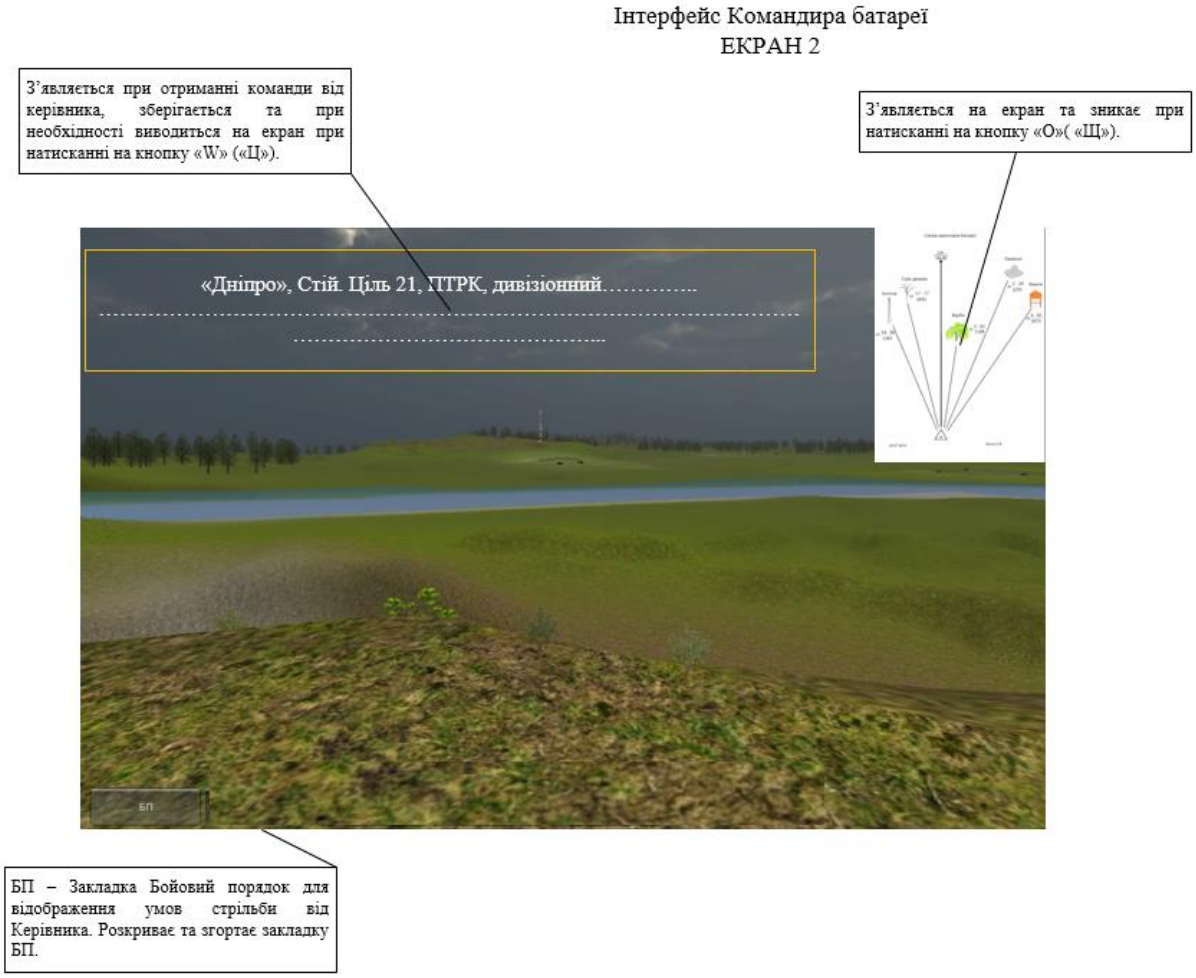


Рисунок 4.1 - Інтерфейс Командира батареї на мапі(прототип)

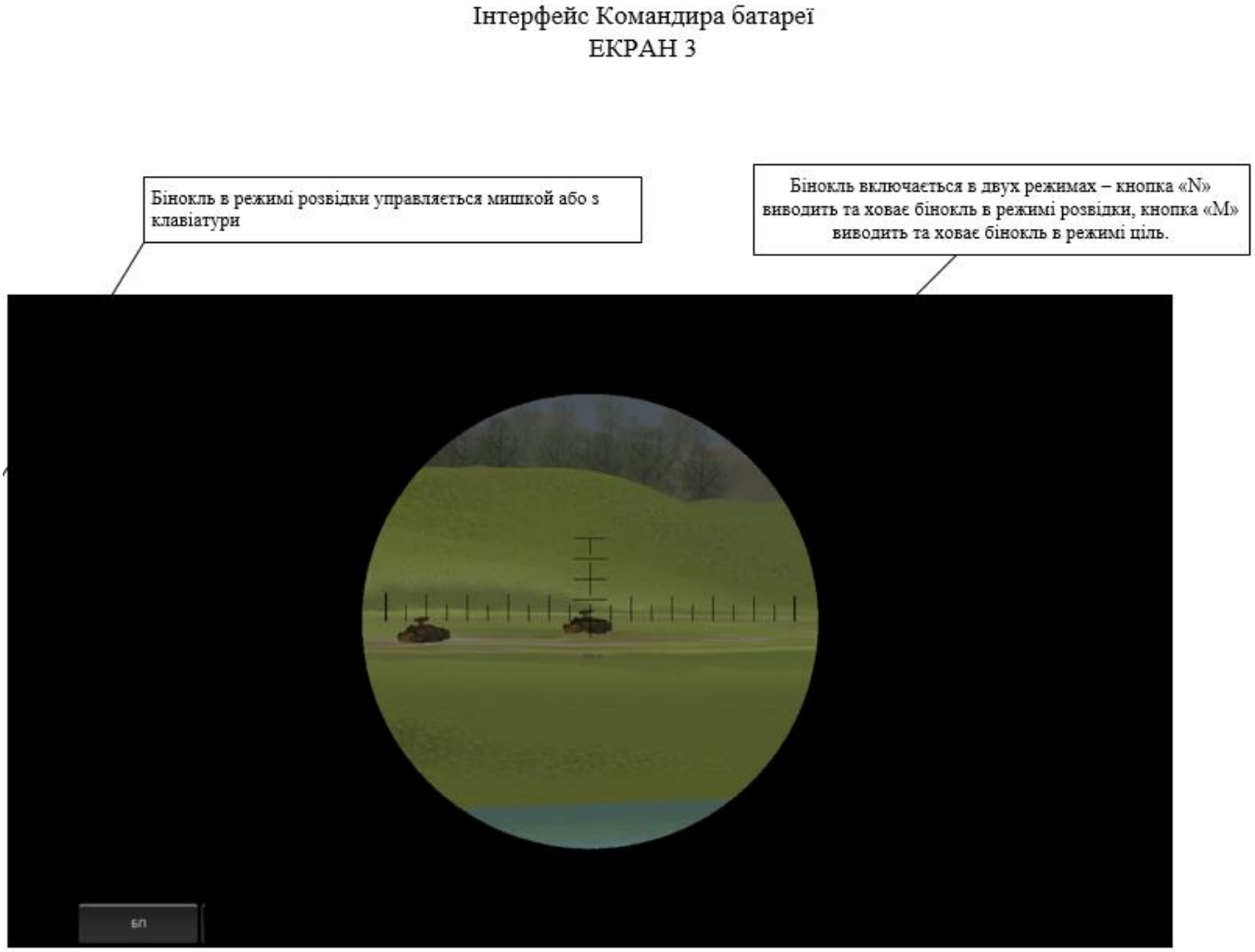


Рисунок 4.2 - Інтерфейс Командира батареї на мапі з Далекоміром(прототип)

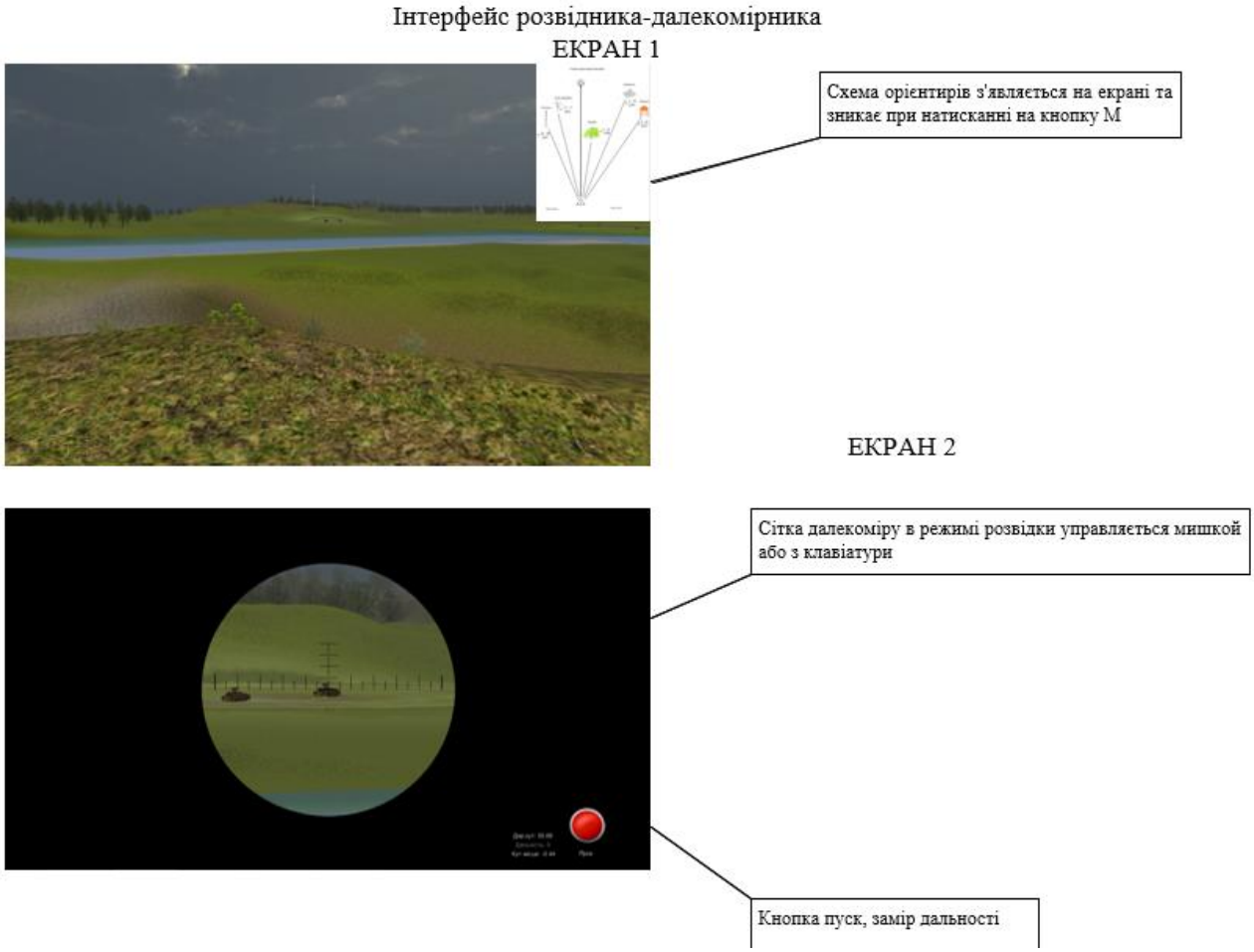


Рисунок 5 - Інтерфейс Розвідника-далекомірника(прототип)

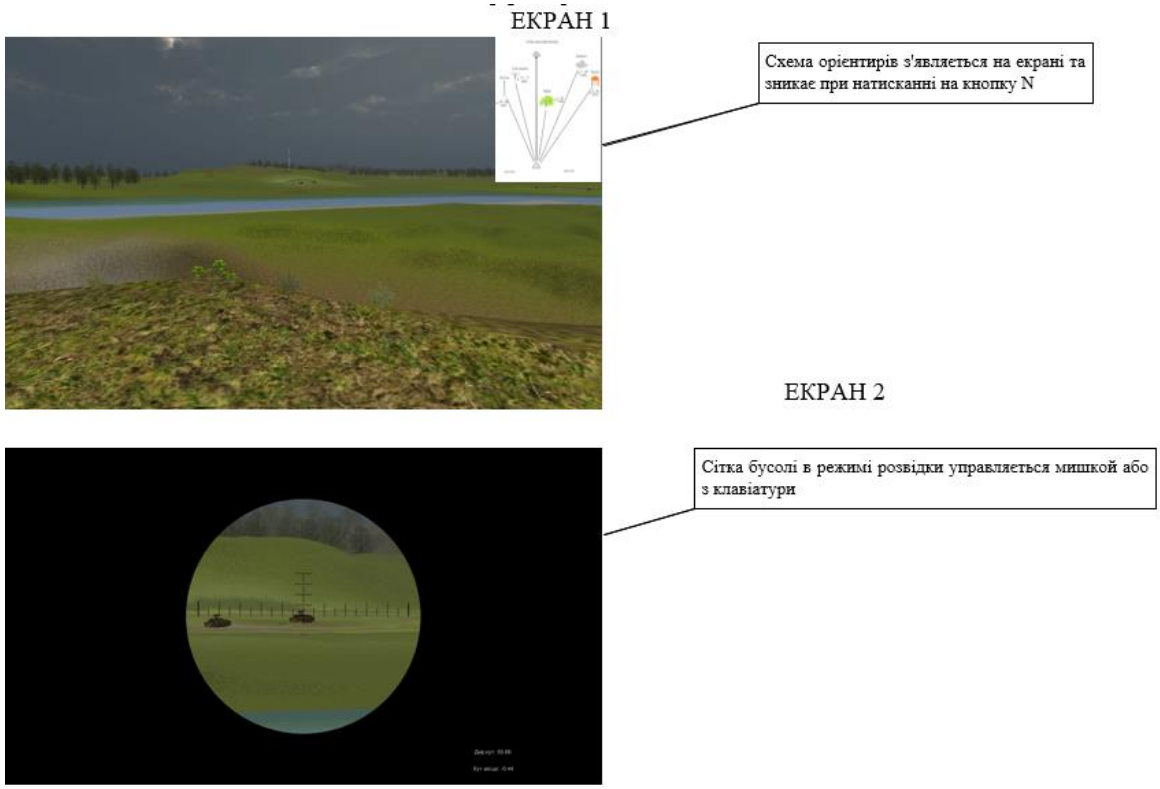


Рисунок 6 - Інтерфейс Розвідника (прототип)

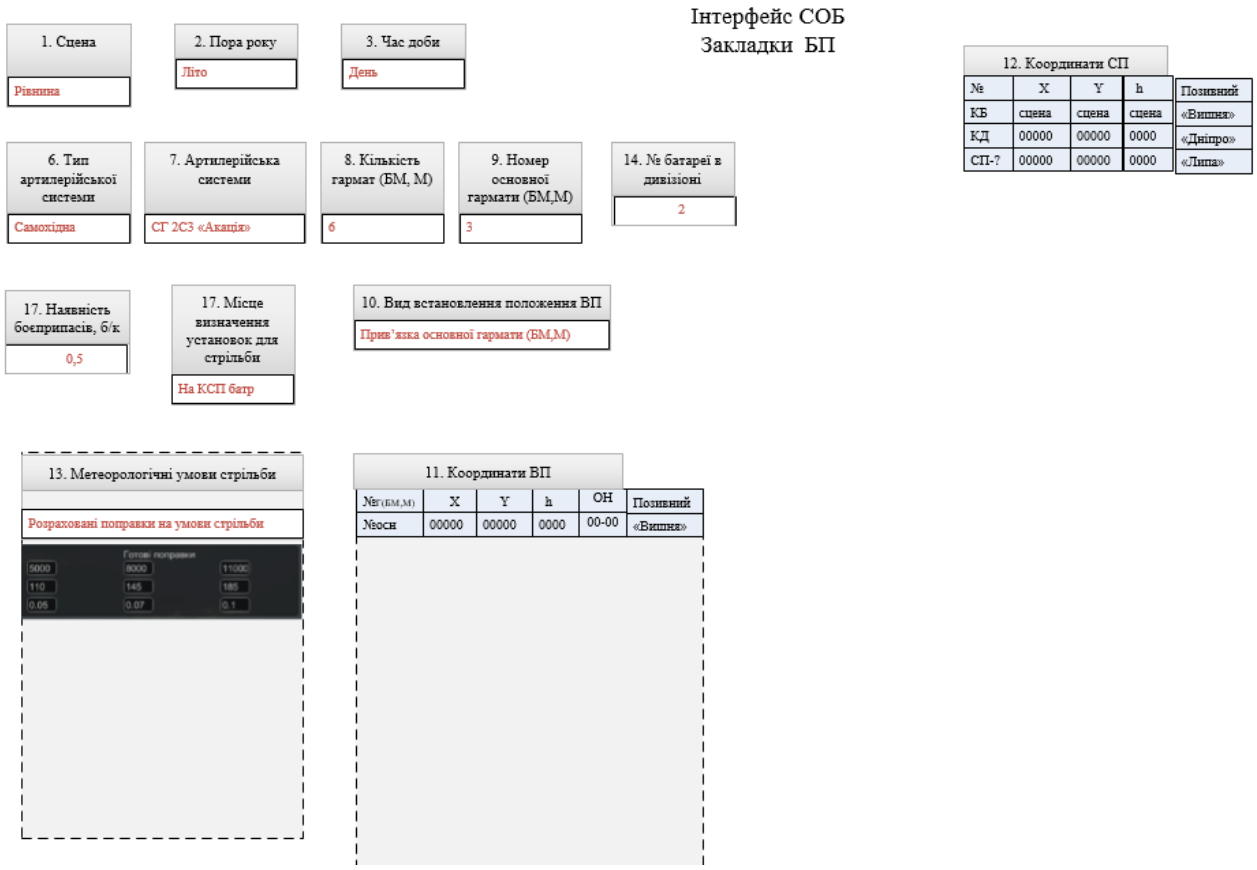


Рисунок 6 - Інтерфейс Старшого офіцера батареї (прототип)

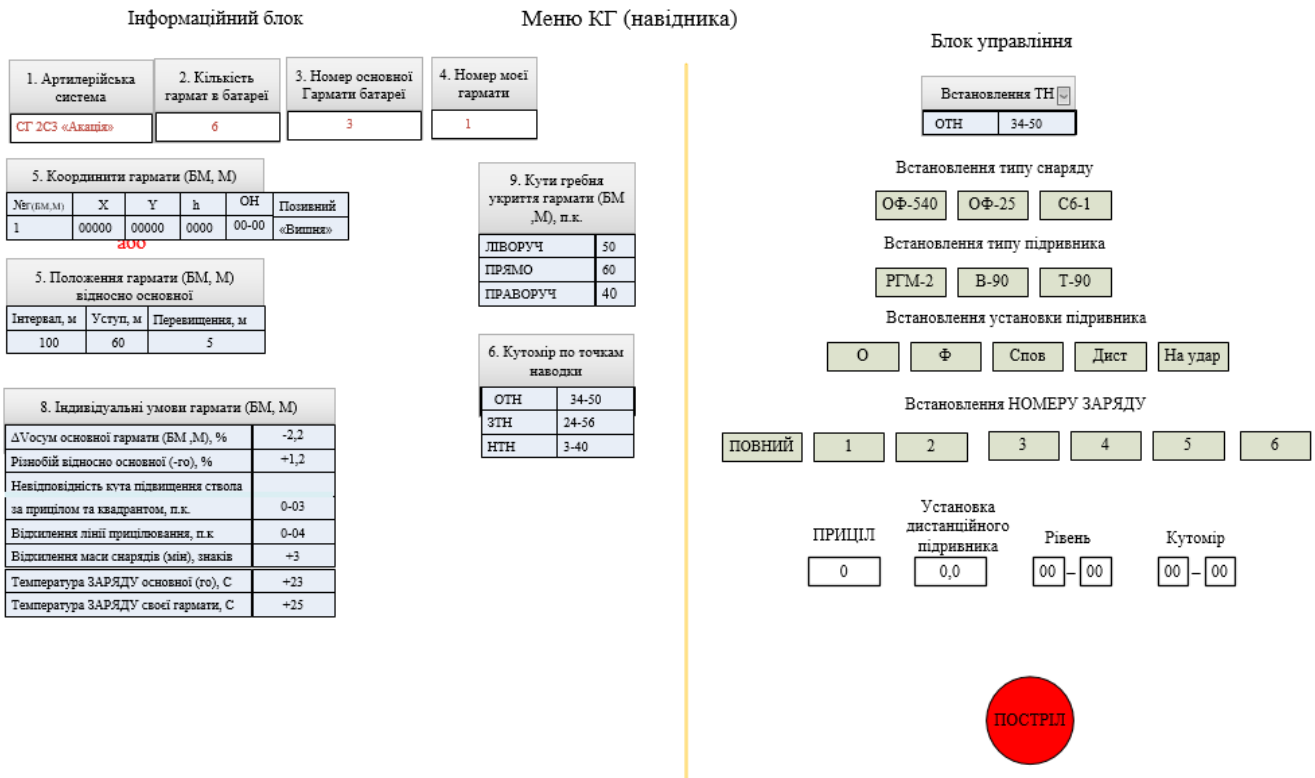


Рисунок 7 - Інтерфейс Командира гармати(прототип)

3.2 Інтерфейс

Для розробки інтерфейсу використовували стандартні можливості Unity3d. Багато необхідних речей вже вбудовані в Unity, тому розроблення інтерфейсу в ньому досить просте завдання, якщо розумієш, як що працює.

Що б додаток нормально функціонувало я створив такі кнопки як: «Початок гри», «Налаштування і Вихід».

Рисунок 8 - Початок гри

Рисунок 9 – Вікно підключення до серверу

Рисунок 10 – Налаштування

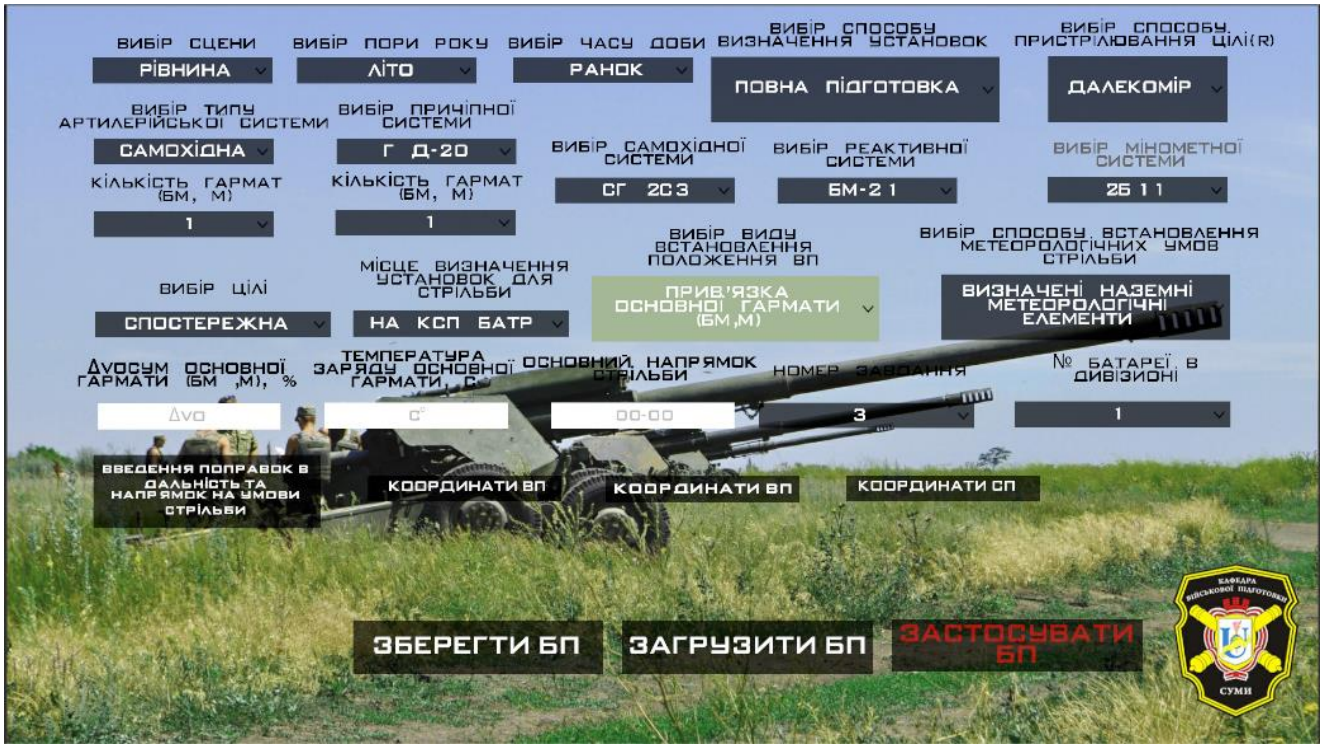


Рисунок 11 - Інтерфейс керівника

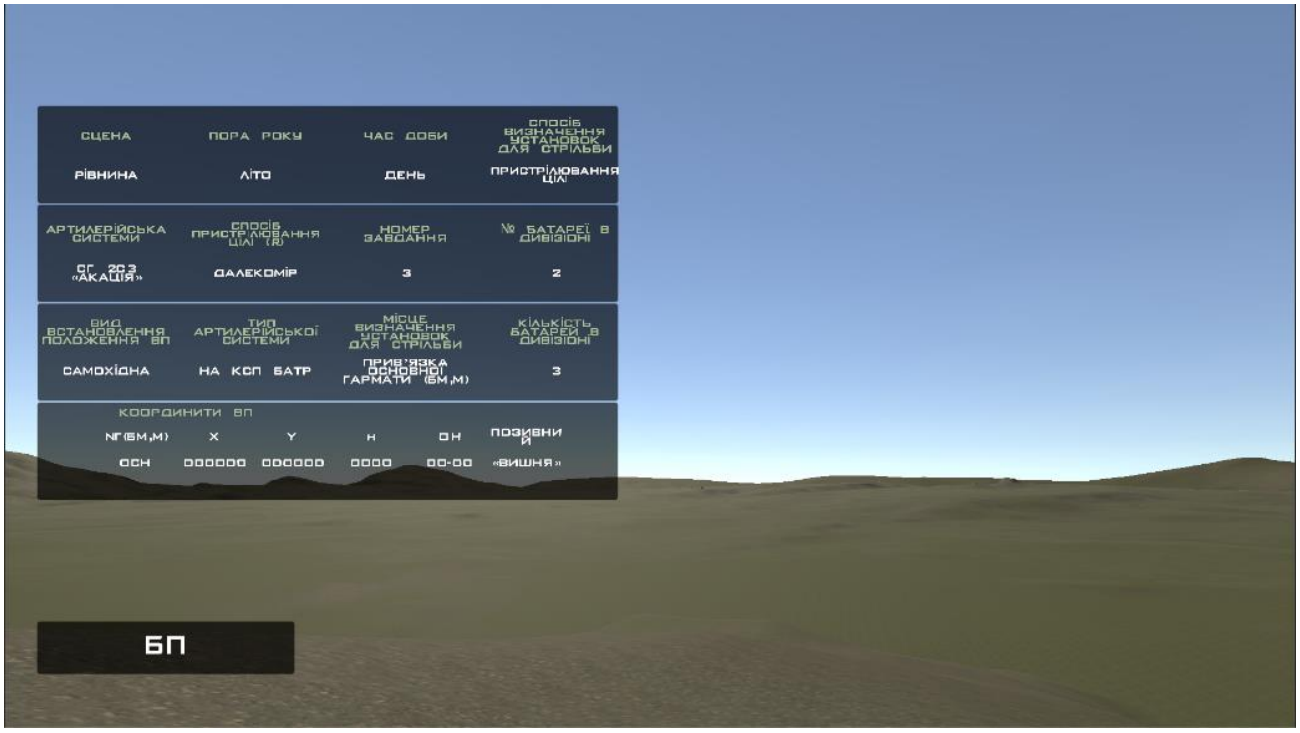


Рисунок 12 - Інтерфейс Командира батареї

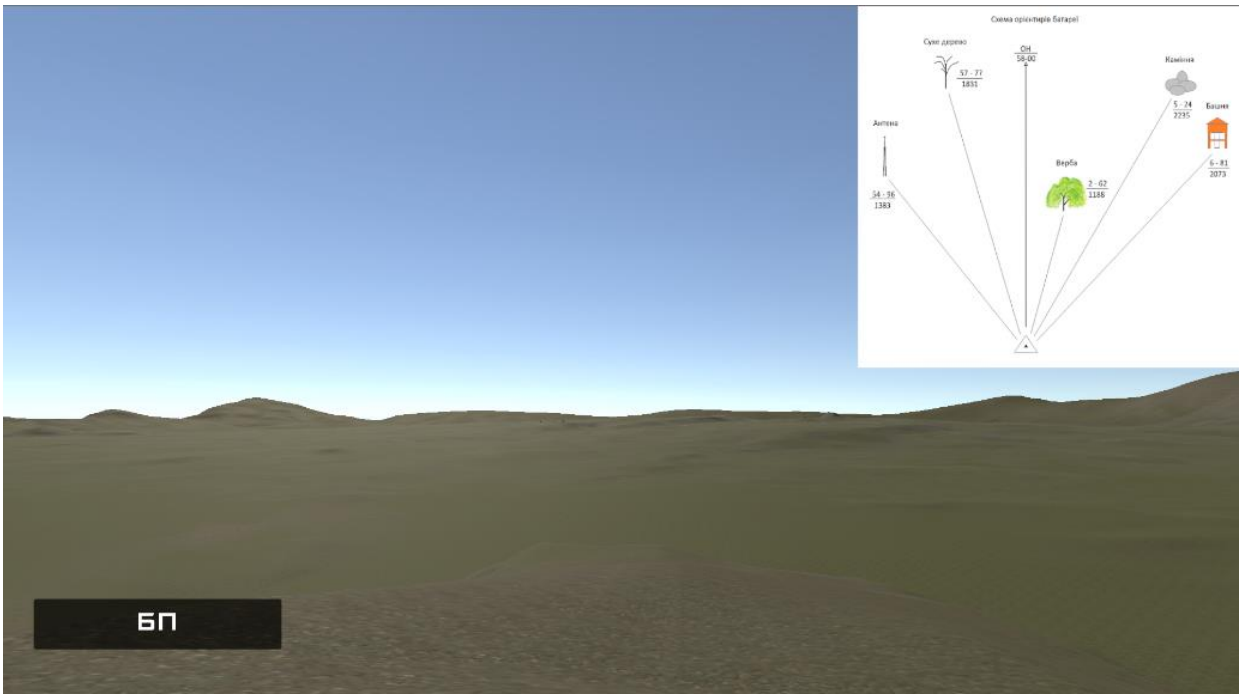


Рисунок 12.1 - Інтерфейс Командира батареї на мапі

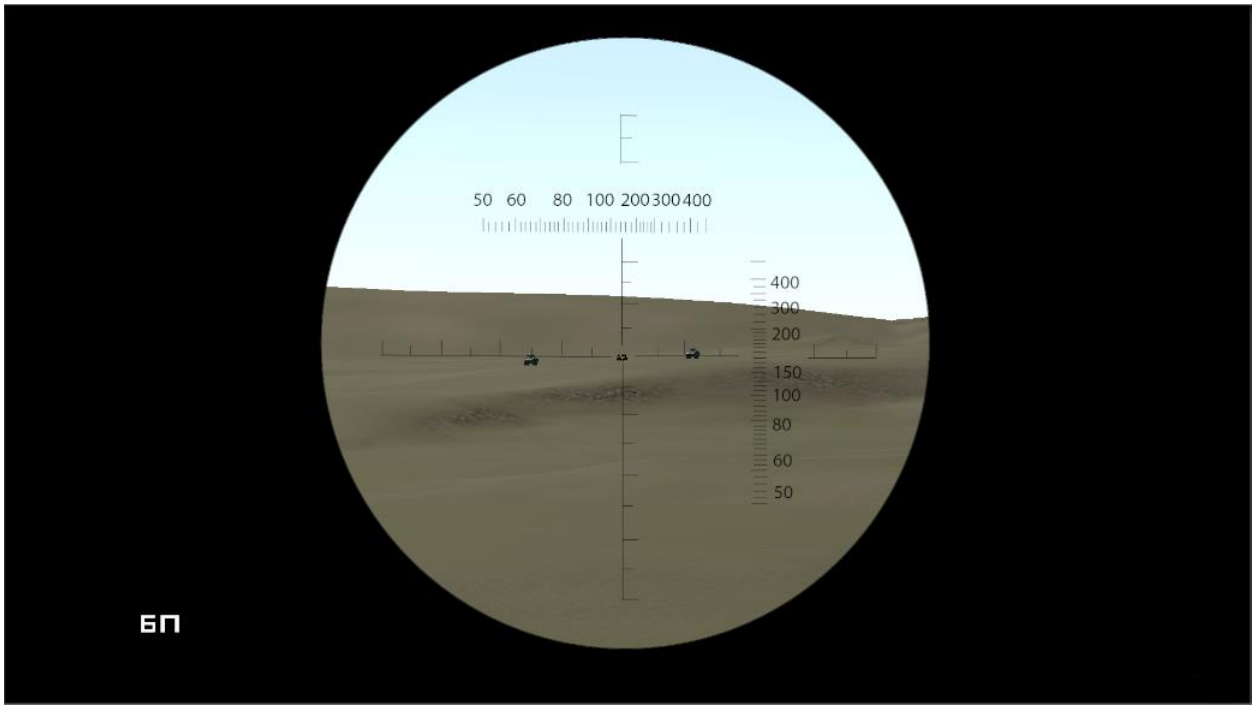


Рисунок 12.2 - Інтерфейс Командира батареї з Бусолью

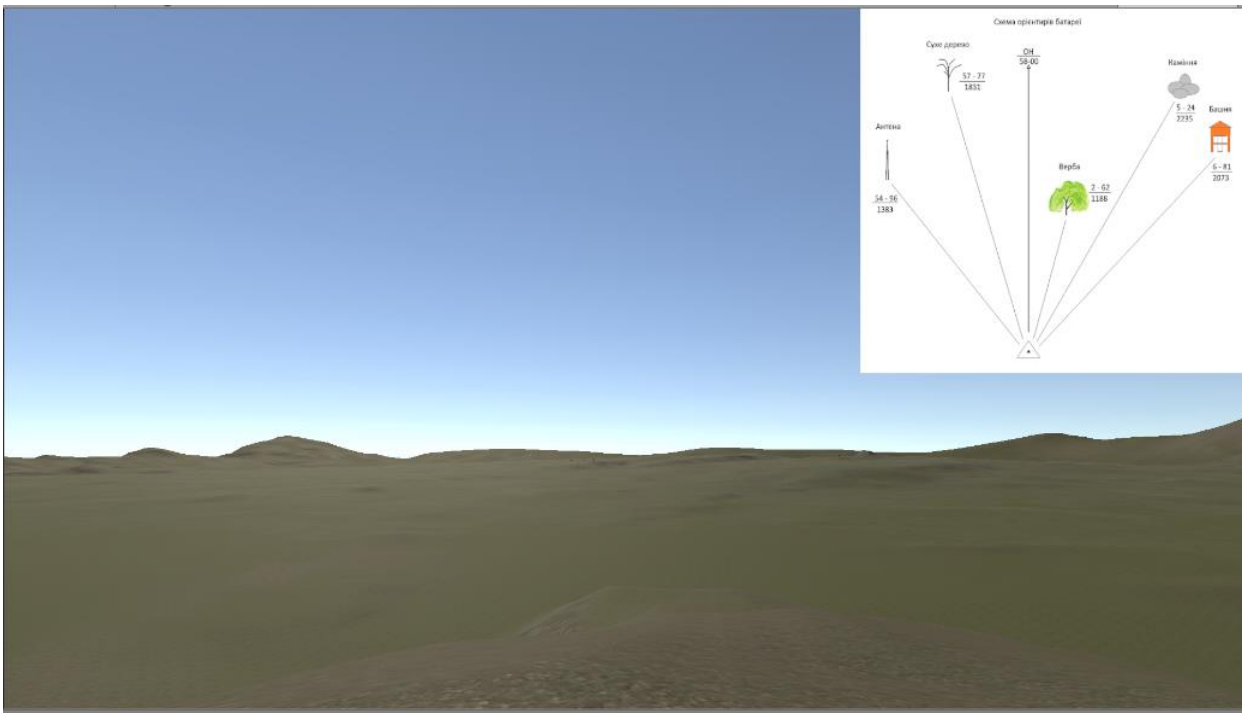


Рисунок 13 - Інтерфейс Розвідника далекоміра

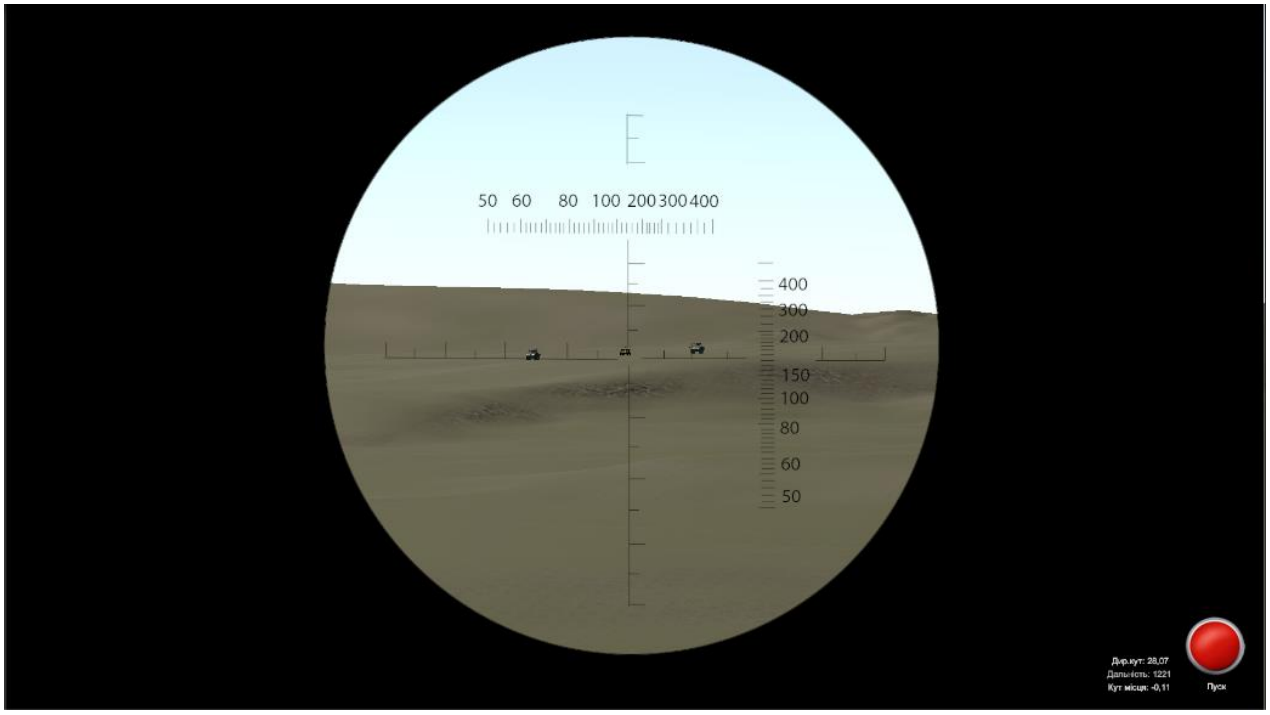


Рисунок 13.1 - Інтерфейс розвідника-далекомірника з далекоміром

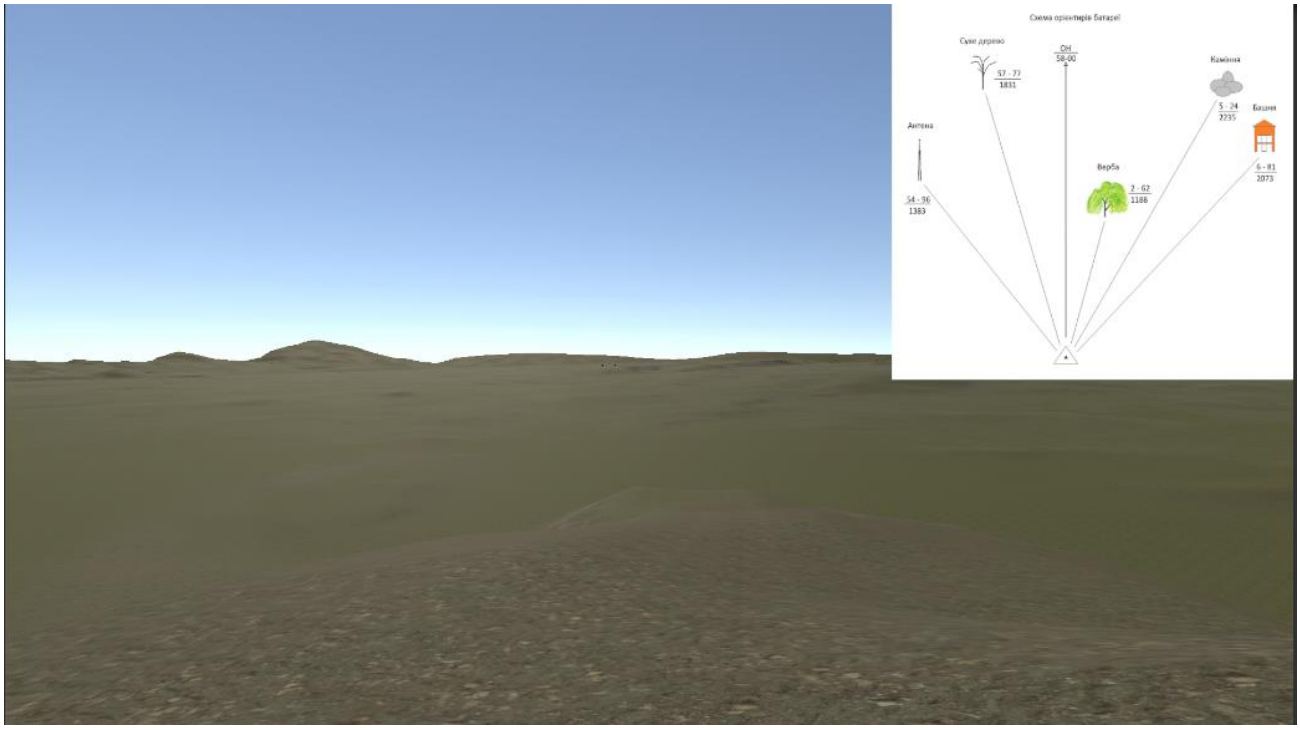


Рисунок 14 - Інтерфейс Розвідника

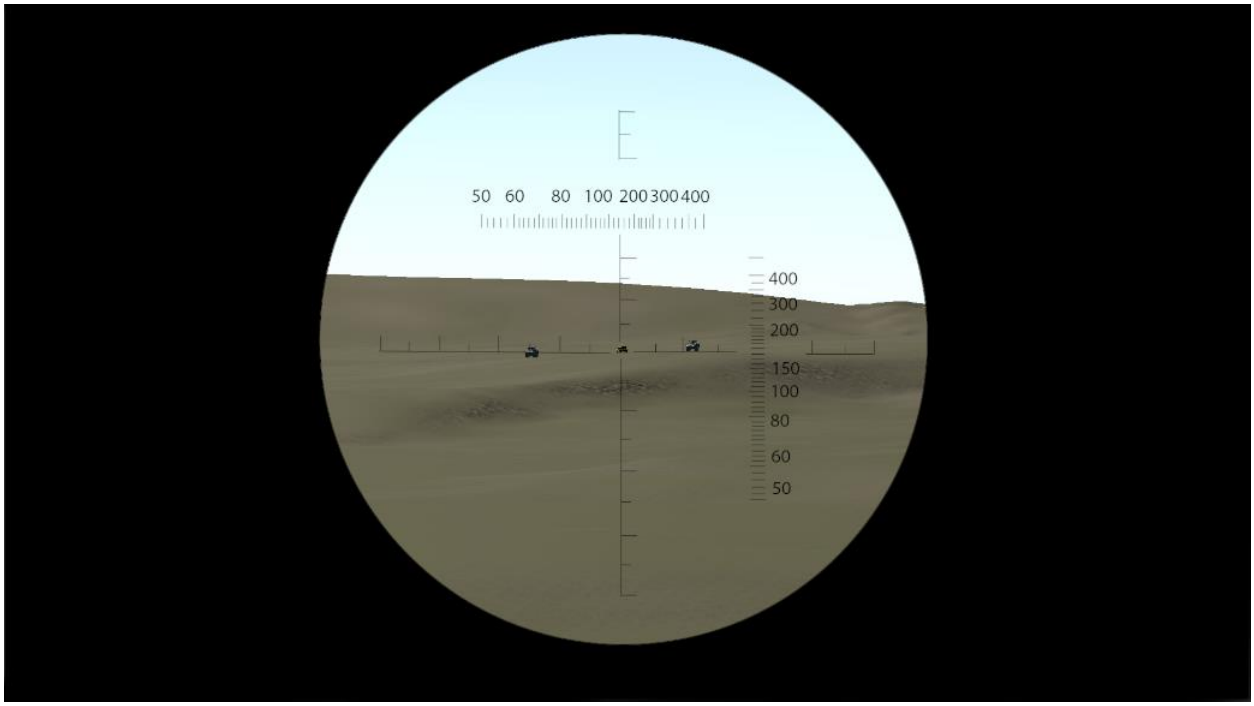


Рисунок 14.1 - Інтерфейс Розвідника з Бусолью

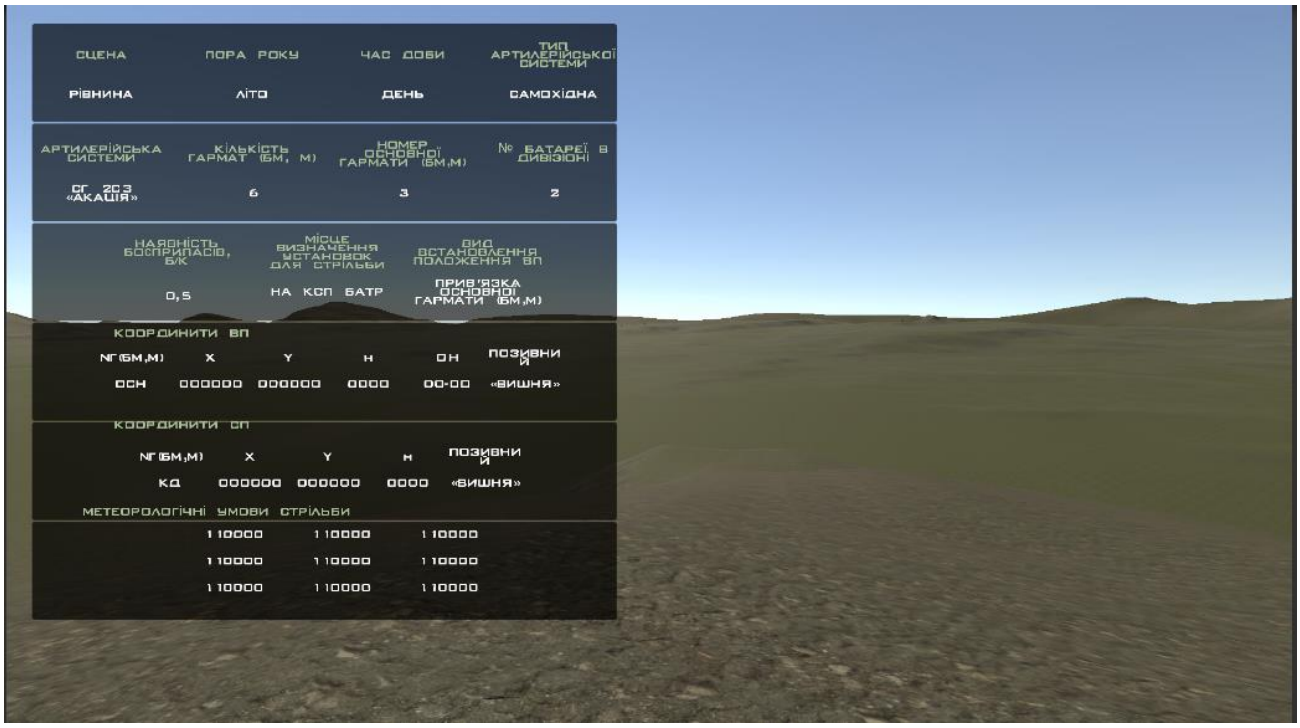


Рисунок 15 - Інтерфейс Старшого офіцера батареї

Рисунок 16 - Інтерфейс Керівника Гармати

3.3 Ландшафт

Ландшафт можна створити за допомогою вбудованих можливостей Unity3d, але це не практично, є безліч ассетів для зручної роботи з інструментами позиціонування ландшафту. Приклади: WordCreator, Gaia, MapMagic. Для розробки ландшафту я використовувасв зручний асет Gaia, у нього зрозумілий інтерфейс і безліч можливостей для автоматичної генерації світу.

Gaia дозволяє:

- Можна зручно розставляти пріоритети, і виділяти більше часу на більш важливі речі, так як багато інструментів для редагування ландшафту вже є в ассет.

- Зручно створювати та формувати місцевість за смаком користувача.

- Основні ресурси можна тратити на насадження та заселення місцевості, а не на побудову рельєфу.

Так само для генерації води був використаний ассет AQUAS.

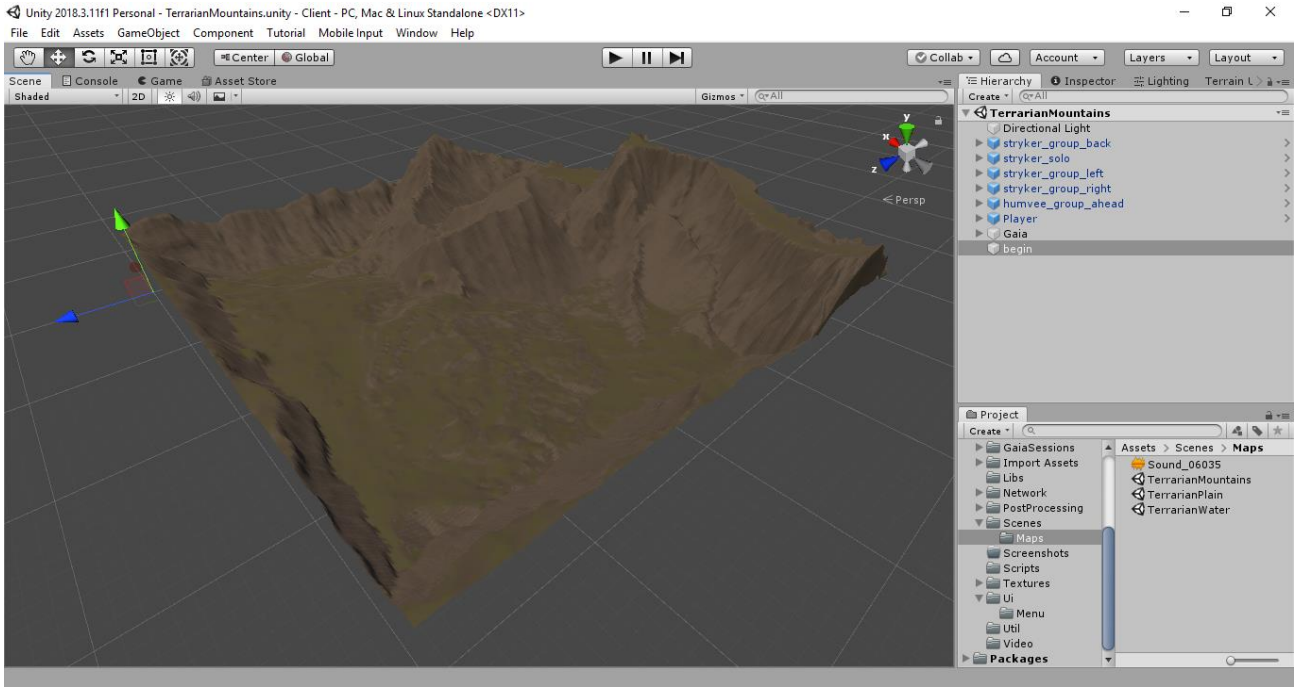


Рисунок 17 – Гірський масив

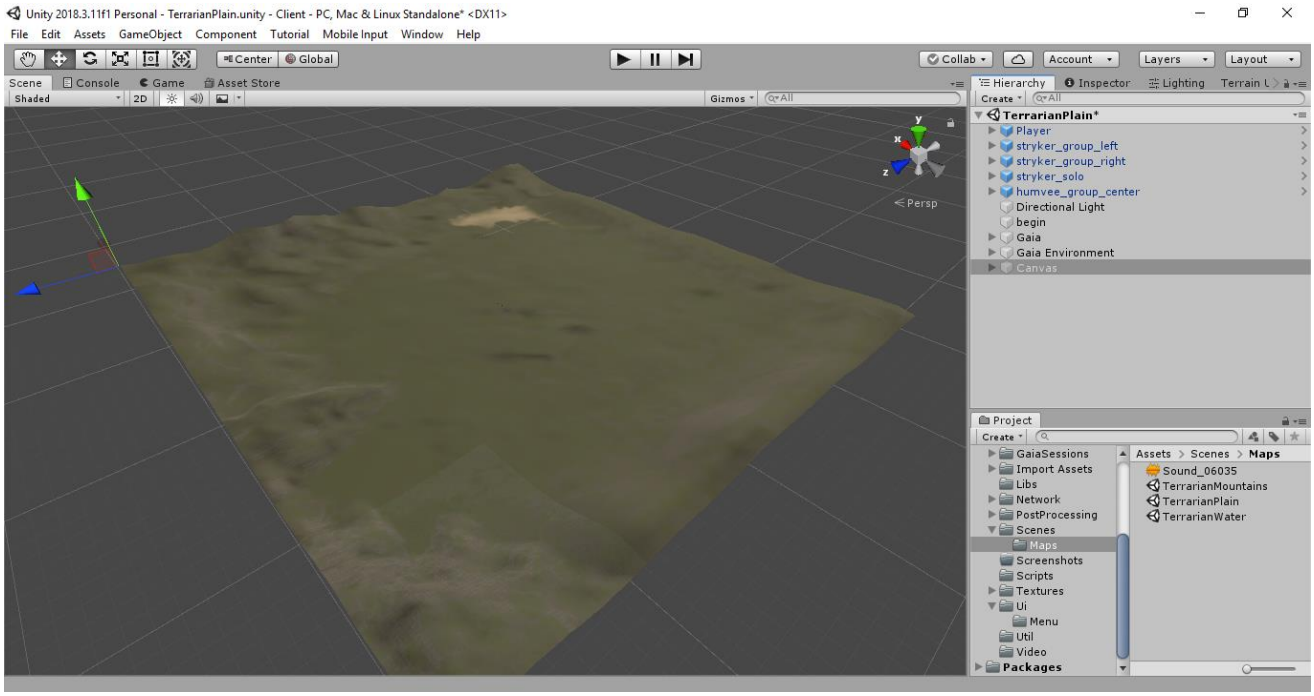


Рисунок 18 – Рівнини

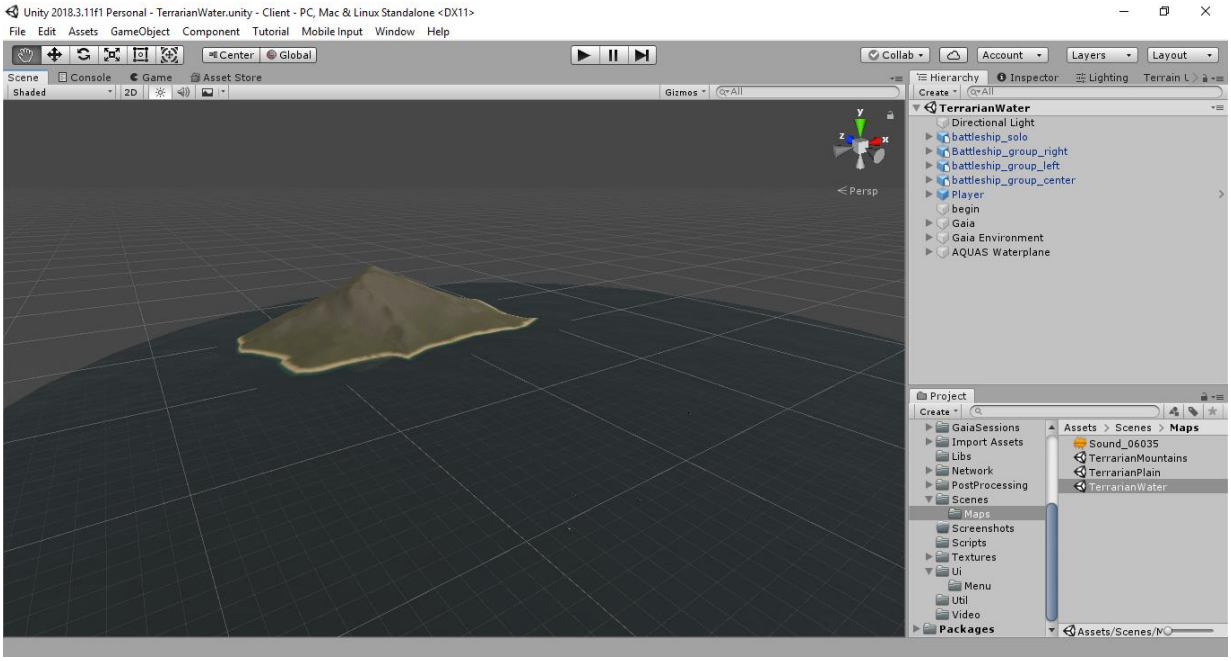


Рисунок 19 – Море

3.4 Моделі цілей

Для вирішення цьої задачі було прийнято рішення використовувати готові ассети(безкоштовні) моделей транспорту.Таке рішення було прийнято із-за того що небуло в наявності потрібних потужностей для рендеренгу моделей.

Ассети які використовувались: military vehicles і battleships.

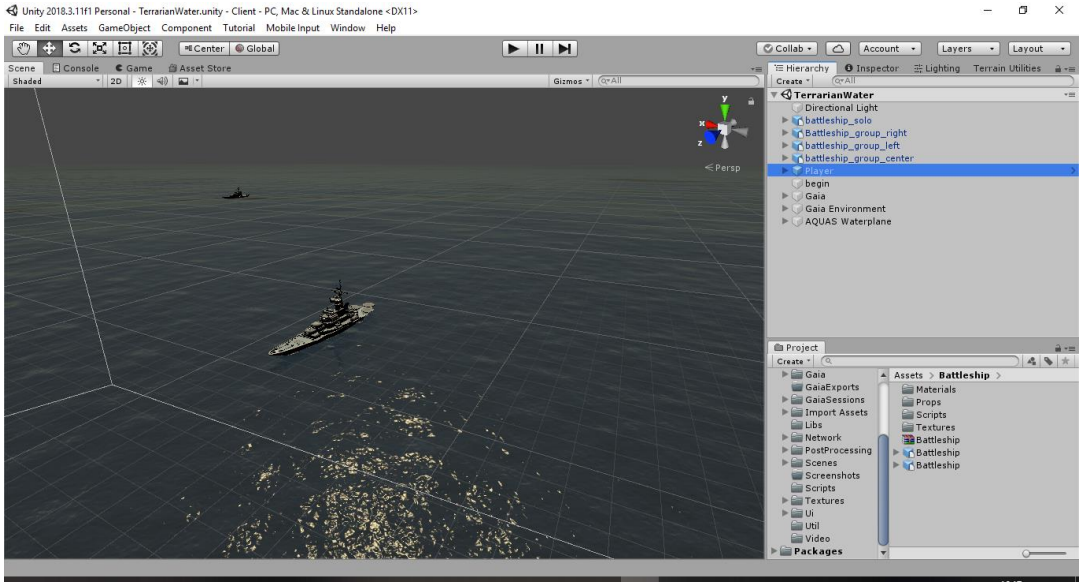


Рисунок 20 – Моделі кораблів

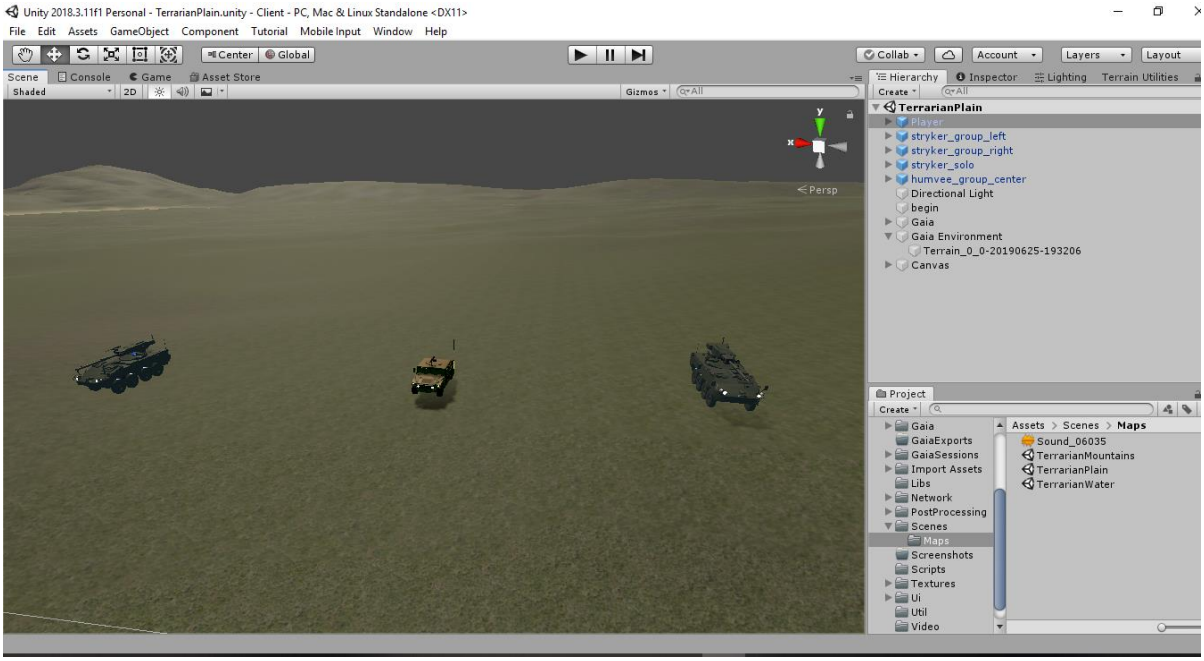


Рисунок 21 – Моделі наземного транспорту

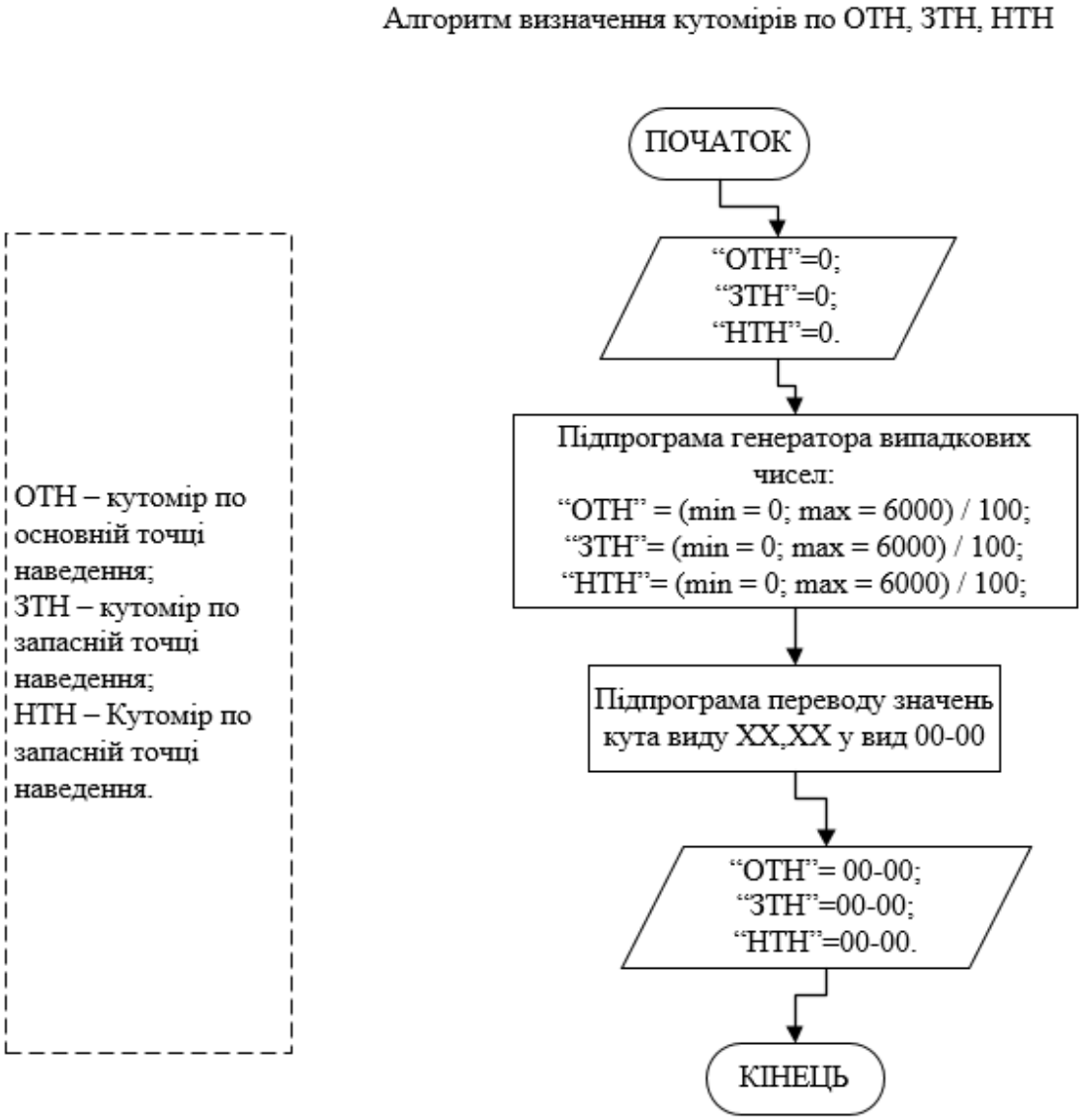
3.5 Реалізація Алгоритмів

Далекомір

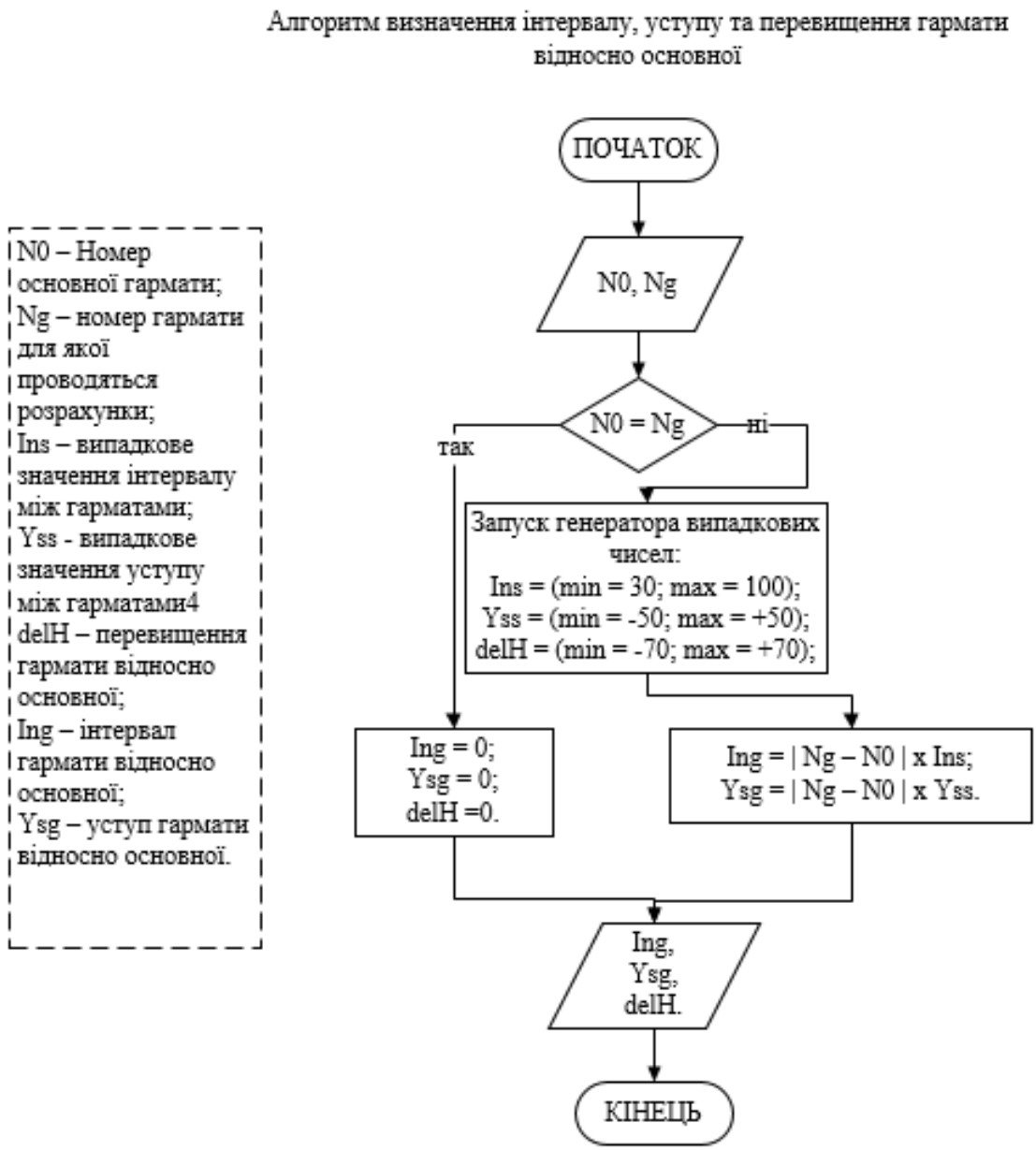
У далекоміра активному типі принцип роботи полягає в вимірюванні часу, протягом якого проходить сигнал, який далекомір послав на об'єкт. Оскільки швидкість поширення сигналу (звуку чи світла) вже відома можна дізнатися відстань до об'єкта. Вимірювання відстаней пасивними далекоміром ґрунтується на визначенні висоти рівнобедреного трикутника, наприклад, із відомої сторони протилежного гострого кута (так званого паралактичного кута). Одна з величин, або зазвичай є постійною, а інша - змінна (вимірювальною). За цими ознаками відрізняють далекомір з постійною основою та далекомір з постійним кутом .

В Unity було реалізовано далекомір за допомогою так званих Raycast, це промінь, який при натисканні на кнопку «Пуск» або праву кнопку миші, вистрілює в напрямок на який повернута камеро з кодом і за допомогою функції «distance» вимірює дальність до об’єкта.

Алгоритм визначення кутомірів по ОТН, ЗТН, НТН



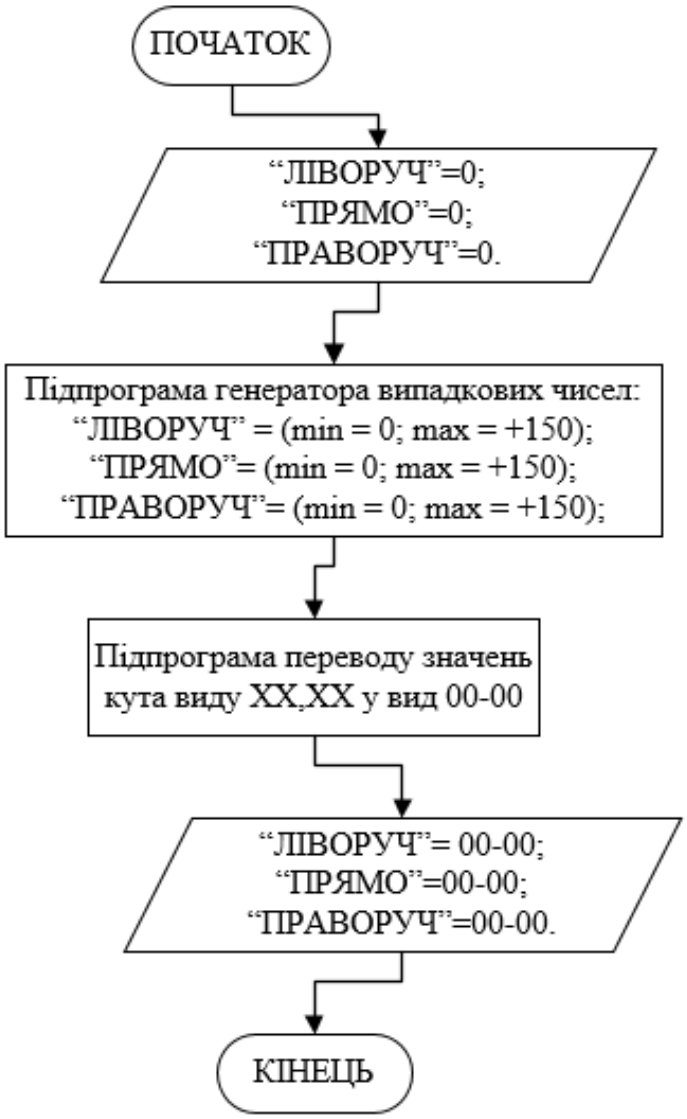
Алгоритм визначення інтервалу, уступу та перевищення гармати відносно основної гармати



Алгоритм визначення кутів гребня

Алгоритм визначення кутів гребня укриття

ЛІВОРУЧ – кут укриття вліво;
ПРЯМО – кут укриття прямо;
ПРАВОРУЧ – кут укриття вправо.



ВИСНОВКИ

Дана робота була присвячена розробці артилерійського симулятора ведення вогню. Такі симулятори зможуть дозволити надавати практичні знання майбутнім артилеристам без витрати великих коштів за боєприпаси, що дає можливість мінімізувати витрати, які будуть іти на здобування практичних навичок військових, без погіршення якості практики. До того ж що в симуляції зроблена помилка не призведе до негативного результатів її вплив буде мати менші наслідки ніж у реальному житті. Менша травматичність та менший ризик для здоров’я людини – ще один позитивний аспект у розробці артилерійського симулятора ведення вогню.

Програма була створена для навчальних цілей для підготовки артилеристів, і може бути розгорнута на локальних серверах військової кафедри.

СПИСОК ЛІТЕРАТУРИ

1. Троелсен Э., Джепикс Ф. Microsoft Visual C#. Подробное руководство. 8-е издание - М.: Издательский дом "Вильямс", 2016. – 1328 с.
2. Скит Д. C# in Depth - М.: Издательский дом "Вильямс", 2019. – 608 с.
3. Рихтер Д. CLR via C# - Питер, 2019. – 896 с.
4. Frederick P.B. The Mythical Man-Month. - Addison-Wesley, 1975. – 336 с.
5. Фримен Э., Робсон Э. Паттерны/шаблоны проектирования - Питер, 2015. – 656 с.
6. Robert C., Agile M. Software Development Principles Patterns and Practices. - Person Education Limited, 2002. – 529 с.
7. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. - Addison-Wesley, 1994. -395 с.
8. Gibson J. Introduction to Game Design, Prototyping, and Development - Addison-Wesley, 2014. – 944 с.
9. Хокинг Д. Unity в действии – Питер, 2016. – 352 с
10. Donzallaz P., Figueroa H., Harduin L., Jover C. Unity Manul https://docs.unity3d.com/ru/current/Manual/index.html

ДОДАТОК

Код до Дальноміру

```
using System;
using UnityEngine;

public class Dalnomer : MonoBehaviour
{
   public GUIStyle BStyle;
    public bool isEnabled=false;
    public float SavedDistance;
    public GameObject TargetTexture;
    private bool _canBeUsed;
    public float delenie_dalnomera = 0;
    public float ugol_mesta = 0;
    private float dd;
    private float M; //Угол места цели (ориентира)


     private void Update()
     {
        if (Input.GetKeyDown(KeyCode.N))
         {
            ToogleDalnomer();
        }

        /*float ygol = Mathf.Rad2Deg*2*Mathf.Acos(Camera.main.transform.rotation.w);
//Дирекционный угол в градусах
          float ugol = Mathf.Rad2Deg*2*Mathf.Asin(Camera.main.transform.rotation.x); //угол
места цели (по высоте) в градусах.*/

        var eulerAngle = Camera.main.transform.eulerAngles;
        var ygol = eulerAngle.y;
        var ugol = eulerAngle.x;
        dd = ygol/6; //Перевод градусов в деления угломера
        M = ugol/6;

        /*if (Camera.main.transform.rotation.y<0) //Проверка условия прирощения по Y
        {
        dd = 60 - dd; //Расчет дирекционного угла на цель (ориентир)
        }*/
        if (M>30)
            {
        M=M-60;
            }
         if (M<30)
            {
        M=M*(-1);
            }


        if(Input.GetKeyDown(KeyCode.M))
          {
            ToogleDalnomer();
            //find where is setting TargetName
            String targetNumber = GameObje.-
t.Find("Transport").GetComponent<TypeOfShootingTransporter>().targetNumber.ToString();
//Поиск цели
            transform.LookAt(GameObject.Find(targetNumber).transform.position);
//Направление бинокля в цель
```

```
          if (isEnabled == true)
          {
              GetComponent<MouseLook>().enabled = false; //Движение мыши в дальномере за-
прещено
          }
          else
          {
              GetComponent<MouseLook>().enabled = true;
          }
          }

        if (Input.GetKeyDown(KeyCode.Escape) && (isEnabled = true))
        {
           ToogleDalnomer();
       }

        if (isEnabled)
       {

          if(Input.GetMouseButton(0))
          {
              _canBeUsed = true;
          }

          if(Input.GetMouseButton(1))
          {
              if (_canBeUsed)
                 {
                     SavedDistance = GetDistance();
                 delenie_dalnomera = dd;
                 ugol_mesta=M;
              }
          }
       }
   }

   public void ToogleDalnomer()
   {
       if (isEnabled)
         {
           Camera.main.fieldOfView = 60;//Поле зрения бинокля 60 градусов
           isEnabled = false;
         }
         else
         {
           Camera.main.fieldOfView = 9.0f; //Поле зрения бинокля (позволяет согласовать
сетку с углами на местности)
           isEnabled = true;
         }
         //TargetTexture.active  = isEnabled;
         gameObject.SetActive(isEnabled);

       _canBeUsed = false;
   }

    private void OnGUI()
    {
        if (isEnabled)
        {
            string str = "Дальність: " + SavedDistance;
            if (SavedDistance <= 0)
            {
                str = "Дальність: 0";
            }
```

```
            GUI.Box(new Rect(Screen.width-250, Screen.height - 70, 120, 40), str);
            GUI.Box(new Rect(Screen.width -250, Screen.height - 90, 120, 40),
                    "Дир.кут: " + Math.Round(dd, 2));
          GUI.Box(new Rect(Screen.width -250, Screen.height - 50, 120, 40),
                  "Кут місця: " + Math.Round (M, 2));

            if (GUI.Button(new Rect(Screen.width - 120, Screen.height - 300, 100, 100),
"", BStyle))
            {
                _canBeUsed = true;
            }

            if (GUI.Button(new Rect(Screen.width - 120, Screen.height - 150, 100, 100),
"", BStyle))
            {
             if (_canBeUsed)
                {
                    SavedDistance = GetDistance();
                delenie_dalnomera = dd;
                ugol_mesta = M;
             }
            }

            //GUI.Label(new Rect(Screen.width - 120 + 10, Screen.height - 200, 100, 30),
"Розрахувати");
            GUI.Label(new Rect(Screen.width - 120 + 32, Screen.height - 50, 100, 30),
"Пуск");

       }
    }

    private float GetDistance()
    {
        RaycastHit hit;
        Vector3 fwd = Camera.main.transform.TransformDirection(Vector3.forward);
        if (Physics.Raycast(transform.position, fwd, out hit, 10000.0f))
        {
            float distanceToTarget = hit.distance;
            return (int)(distanceToTarget);
        }
        return 0f;
    }
}
```

Код до алгоритму визначення кутомірів по OTH, ZTH, HTH

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

public class Goniometer : MonoBehaviour
{
    void Start()
    {
        RandomAngle(0, 0, 0);
    }



    // Update is called once per frame
    void Update()
    {

    }

    private void RandomAngle(float OTH = 0, float ZTH = 0, float HTH = 0)
    {
        OTH = UnityEngine.Random.Range(0.0f, 6000.0f);
        ZTH = UnityEngine.Random.Range(0.0f, 6000.0f);
        HTH = UnityEngine.Random.Range(0.0f, 6000.0f);
        TranformToFormat(OTH, ZTH, HTH);
    }

    private void TranformToFormat(float OTH, float ZTH, float HTH)
    {
        string strOTH = System.Convert.ToString(OTH);
        string strZTH = System.Convert.ToString(ZTH);
        string strHTH = System.Convert.ToString(HTH);
        string[] partsOTH = strOTH.Split(',');
        string[] partsZTH = strZTH.Split(',');
        string[] partsHTH = strHTH.Split(',');
        Debug.Log("OTH: " + partsOTH[0].Substring(0, 2) + "-" + partsOTH[1].Substring(0,
2));
        Debug.Log("ZTH: " + partsZTH[0].Substring(0, 2) + "-" + partsZTH[1].Substring(0,
2));
        Debug.Log("HTH: " + partsHTH[0].Substring(0, 2) + "-" + partsHTH[1].Substring(0,
2));
    }
      }
```

Код до алгоритму визначення кутів гребня

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

public class ShelterAngle : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        RandomAngle(0, 0, 0);
    }



    // Update is called once per frame
    void Update()
    {

    }

    private void RandomAngle(float left = 0, float right = 0, float straight = 0)
    {
        left = UnityEngine.Random.Range(0.0f, 150.0f);
        right = UnityEngine.Random.Range(0.0f, 150.0f);
        straight = UnityEngine.Random.Range(0.0f, 150.0f);
        TranformToFormat(left, right, straight);
    }

    private void TranformToFormat(float left, float right, float straight)
    {
        string strLeft = System.Convert.ToString(left);
        string strRight = System.Convert.ToString(right);
        string strStraight = System.Convert.ToString(straight);
        string[] partsLeft = strLeft.Split(',');
        string[] partsRight = strRight.Split(',');
        string[] partsStraight = strStraight.Split(',');
        Debug.Log("Left: " + partsLeft[0] + "-" + partsLeft[1].Substring(0,2));
        Debug.Log("Right: " + partsRight[0] + "-" + partsRight[1].Substring(0, 2));
        Debug.Log("Straight: " + partsStraight[0] + "-" + partsStraight[1].Substring(0,
2));
    }
      }
```

Код до Алгоритму визначення інтервалу, уступу та перевищення гармати відносно основної гармати

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

public class LedgeInterval : MonoBehaviour
{
    private float NO = 0;
    private float Ng = 0;
    private float Ing = 0;
    private float Ins = 0;
    private float Yss = 0;
    private float Ysg = 0;
    private float delH = 0;
    // Start is called before the first frame update
    void Start()
    {
        ConditionEquality(NO, Ng);
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void ConditionEquality(float NO, float Ng)
    {
        if(NO != Ng)
        {
            RandomAngle();
        } else
        {
            Ing = 0;
            Ysg = 0;
            delH = 0;
            Output(Ing, Ysg, delH);
        }

    }

    private void RandomAngle(float Ins = 0, float Yss = 0, float delH = 0)
    {
        Ins = UnityEngine.Random.Range(30.0f, 100.0f);
        Yss = UnityEngine.Random.Range(-50.0f, 50.0f);
        delH = UnityEngine.Random.Range(-70.0f, 70.0f);
        Calculation(Ing, Yss, delH);
    }

    private void Calculation(float Ins, float Yss, float delH)
    {
        Ing = Mathf.Abs(Ng - NO) * Ins;
        Ysg = Mathf.Abs(Ng - NO) * Yss;
        Output(Ing, Ysg, delH);
    }
```

```
    private void Output(float Ing, float Ysg, float delH)
    {
        Debug.Log("Ing: " + Ing);
        Debug.Log("Ysg: " + Ysg);
        Debug.Log("delH: " + delH);
    }
}
```

Код до вікна Рисунок 9

```
using Assets.Network;
using KVP_Api.Network;
using System;
using UnityEngine;
using UnityEngine.UI;

namespace Assets.Ui.Menu.StartMenu
{
    public class ConnectMenu : MonoBehaviour
    {
        public InputField name;
        public InputField host;

        public GameObject preCanvas;
        public GameObject postCanvas;
        public GameObject menu;

        public void Connect()
        {
            Debug.Log("Init client");
            new Client(new ConnectableAddress(host.text.Split(':')[0],
                int.Parse(host.text.Split(':')[1])), name.text);

            menu.SetActive(false);
            preCanvas.SetActive(false);
            postCanvas.SetActive(!postCanvas.activeSelf);
        }
    }
}
```

Код до вікна Рисунку 10 і для форм «Налаштування» і «Вихід»

```
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Video;

namespace Assets.Ui.Menu
{
    public class MainMenu : MonoBehaviour
    {
        public GameObject settingsPlane;
        public GameObject startGamePlane;
        public Slider volumeGame;
        public Slider volumeMainMenu;
        public VideoPlayer videoOnMenu;

        void Update()
        {
            changeMainMenuVolume(volumeMainMenu.value);
            changeVolume(volumeGame.value);
        }

        private void changeMainMenuVolume(float newVolume)
        {
            videoOnMenu.SetDirectAudioVolume(0, newVolume);
        }

        private void changeVolume(float newVolume)
        {
            PlayerPrefs.SetFloat("volume", newVolume);
            AudioListener.volume = PlayerPrefs.GetFloat("volume");
        }

        public void showSettings()
        {
            if (settingsPlane != null)
            {
                startGamePlane.SetActive(false);
                settingsPlane.SetActive(!settingsPlane.activeSelf);
            }
        }

        public void showStartGame()
        {
            if (startGamePlane != null)
            {
                settingsPlane.SetActive(false);
                startGamePlane.SetActive(!startGamePlane.activeSelf);
            }
        }

        public void exitGame()
        {
            Application.Quit();
        }
    }
}
```