**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ**

**КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

КУРСОВА РОБОТА

на тему:

«Генерування лабіринту і пошук виходу з нього»

з предмету «Об’єктно-орієнтовне програмування С#»

студента 2 курсу групи ПД-22

кафедри інженерії програмного забезпечення

Чернодуб Владислав

Викладач

аспірант, старший викладач кафедри

інженерії програмного забезпечення,

Гребенюк Віктор Вікторович

оцінка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ 2023

Зміст

[**Вступ** 3](#_Toc135177321)

[**1** **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ** 4](#_Toc135177322)

[**1.1** **Історія створення та використання лабіринтів** 4](#_Toc135177323)

[**1.2** **Існуючі алгоритми побудови лабіринтів** 5](#_Toc135177324)

[**2** **Обгрунтування вибору середовища** 6](#_Toc135177325)

[**2.1** **Вибір мови програмування** 6](#_Toc135177326)

[2.1.1 Історія мови програмування. 6](#_Toc135177327)

[2.1.2 Особливості мови 8](#_Toc135177328)

[2.1.3 Порівняння з конкурентами 10](#_Toc135177329)

[2.1.4 Версії мови 11](#_Toc135177330)

[**2.2** **Вибір середовища розробки для програмування** 13](#_Toc135177331)

[2.2.1 Історія середовища 13](#_Toc135177332)

[**2.3** **Вибір середовища для графічної частини** 15](#_Toc135177333)

[2.3.1 Порівняння з конкурентами 16](#_Toc135177334)

[**2.4** **Використані технології** 17](#_Toc135177335)

[**3.1** **Розробка додатку** 18](#_Toc135177336)

[**3.2**  **Короткий обзор додатку** 25](#_Toc135177337)

[**3.3** **Перспективи проекту** 27](#_Toc135177338)

[**ВИСНОВКИ** 29](#_Toc135177339)

[**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ** 30](#_Toc135177340)

[**ДОДАТОК А** 34](#_Toc135177341)

# **Вступ**

Генерування лабіринту та пошук виходу з нього є актуальними проблемами у сфері комп'ютерних наук. Ця тема викликає значний інтерес серед дослідників та розробників, оскільки лабіринти використовуються у багатьох сферах, включаючи графіку, штучний інтелект, комп'ютерні ігри та навіть в реальному світі, наприклад, для планування маршрутів у робототехніці.

Мета даної курсової роботи полягає в розробці програмного рішення на мові програмування C#, яке здатне генерувати лабіринти різних розмірів та знаходити шлях до виходу з них. Це дозволить нам не лише отримати розуміння алгоритмів та підходів до генерації лабіринтів, але й навчитися використовувати їх для розв'язання практичних завдань.

Розробка відбувалася у Visual Studio Community 19 та Unity2020, мовою програмування C# та платформою .NET 5.

# **АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ**

* 1. **Історія створення та використання лабіринтів**

Історія створення та використання лабіринтів почалась зі стародавніх часів. Лабіринти використовувалися в різних культурах та цивілізаціях з різних причин, включаючи релігійні обряди, розваги та навіть захист.

Одним з найвідоміших прикладів лабіринтів є Мінотавровий лабіринт, створений у грецькій міфології. Згідно з міфом, цей складний лабіринт був створений Дедалом для ув'язнення Мінотавра. Легенда Мінотавра та його лабіринту стала джерелом натхнення для багатьох мистецьких творів та літератури.

Протягом історії, лабіринти також використовувалися у багатьох інших культурах. Наприклад, древні римляни будували лабіринти в садах своїх вілл для розваг та релаксації. У середньовіччі лабіринти були часто включені до дизайну церковних садів, символізуючи шлях до духовної мудрості та богоспівної мети.

У сучасному світі, лабіринти залишаються популярними як розважальні та педагогічні елементи. Вони зустрічаються в лабіринтових парках та розважальних комплексах, де вони створюються для задоволення людей та розвитку логічного мислення. Також лабіринти використовуються в комп'ютерних іграх, де гравець повинен пройти складний лабіринт, розв'язуючи головоломки та уникати ворогів.

У сфері комп'ютерних наук, генерація лабіринтів та пошук виходу з них мають широке застосуван

ня. Вони використовуються для створення ігрових рівнів, симуляцій, штучного інтелекту та багатьох інших областей. Розробка ефективних алгоритмів для генерації лабіринтів та пошуку виходу є важливим завданням для розвитку цих областей та забезпечення цікавої та викликової гри.

* 1. **Існуючі алгоритми побудови лабіринтів**

Перші алгорити, які використовувались для створення лабіринтів є:

* Алгоритм "Recursive Backtracking" (рекурсивне відступання): Цей алгоритм, також відомий як "Depth-First Search" (пошук у глибину), базується на рекурсивному проходженні по сітці та випадковому знищенні стін, щоб створити лабіринт. Він гарантує, що у лабіринті буде шлях від початку до кінця, але може породити лабіринти з великою кількістю вузьких проходів.
* Алгоритм "Prim's Algorithm" (алгоритм Пріма): Цей алгоритм використовує поняття мінімального каркасного дерева для створення лабіринту. Він розпочинає з одної комірки і додає по одній стінці за раз, вибираючи найкоротший зв'язок між вже існуючими комірками. Цей алгоритм гарантує, що утворений лабіринт буде з'єднаним та без циклів.
* Алгоритм "Kruskal's Algorithm" (алгоритм Крускала): Цей алгоритм також використовує поняття мінімального каркасного дерева. Він випадковим чином обирає стінки та з'єднує комірки, доки всі комірки не будуть з'єднані в одне дерево. Цей алгоритм також гарантує з'єднаність лабіринту та відсутність циклів.

Ці алгоритми є базовими та широко використовуються для створення лабіринтів. Згодом були розроблені й інші алгоритми, що розширили можливості генерації лабіринтів та надали більше контролю над їх складністю та структурою. Для своєї роботи був вибраний перший алгоритм.

# **Обгрунтування вибору середовища**

* 1. **Вибір мови програмування**

Для виконання даної курсової роботи було обрано мову програмування C# [5], яка використовує платформу .NET [5].

C# (вимовляється «Сі-Шарп») - це мова програмування, призначена для розробки найрізноманітніших додатків, призначених для виконання в середовищі .NET Core [5]. Мова C# проста, типу безпечна і об'єктно-орієнтована. Завдяки безлічі нововведень C# забезпечує можливість швидкої розробки додатків, але при цьому зберігає виразність і елегантність, властиву мовам C.

Visual C# - це реалізація мови C# корпорацією Microsoft [6]. Підтримка Visual C# в Visual Studio забезпечується за допомогою повнофункціонального редактора коду, компілятора, шаблонів проектів, конструкторів, майстрів коду, потужного і зручного відладчика і багатьох інших засобів. Бібліотека класів .NET Core надає доступ до багатьох служб операційної системи і другим корисним, правильним класам, що істотно прискорює цикл розробки.

### Історія мови програмування.

С# живе за принципом «будь-яка сутність є об'єкт». Її відносять до об'єктно-орієнтованих, а точніше об'єктним, мов програмування. «Мова заснований на суворої компонентної архітектури і реалізує передові механізми забезпечення безпеки коду» - так прийнято характеризувати її. Однак скептики сумніваються як мінімум в її безпеці.

Прихильники C# називають її самим мультипарадигменною, універсальною, просунутою і зручною у використанні мовою програмування. З огляду на той факт, що за нею стоїть платформа Microsoft.NET, число таких прихильників досить велике.

Далекі предки C# з'явилися ще в 60-х роках. Все почалося з появи мови B, який в 1969 році був створений колективом розробників з Технологічного інституту Массачусетсу (англ. MIT) [7]. Головним автором B є Кен Томпсон. Тоді команда працювала над операційною системою UNIX [8]. Вже існуючий мову PL/I, що застосовувався в той час для мейнфреймів виробництва компанії IBM [9], був досить громіздким і менше підходив для поставленого завдання. Тому вчені вирішили створити нову мову, який і отримав назву Bash. Він є типовим представником ранніх імперативних мов програмування.

Після Bash, як це не дивно, пішов С, який був винайдений в 1972 році. Основою для нової мови послужив сам Bash.

Творцями C були Кен Томпсон і Денис Рітчі, які працювали в дослідницькій лабораторії компанії AT&T (англ. AT & T Bell Telephone Laboratories). У 1971 році Рітчі почав створювати розширену версію Bash. Спочатку він назвав її NBash (New Bash), але коли мова стала сильно відрізнятися від Bash, назву змінили на C. Bash розширився за рахунок явного використання типів, структур і ряду нових операцій.

З приводу виникнення мови Сі Пітер Мойлан в своїй книзі «The case against C» писав: «Це мала бути мова, здатний обійти деякі жорсткі правила, вбудовані в більшість мов високого рівня і забезпечують їх надійність. Це мала бути така мова, який дозволив би робити те, що до нього можна було реалізувати тільки на асемблері або на рівні машинного коду».

У 1984 році Б’ярн Страуструп (Bell Labs) виступив з проектом мови С++ [10]. Коли Страуструп займався дослідженнями в фірмі, йому треба було написати кілька імітаційних програм для моделювання розподілених обчислень. SIMULA-67 - об'єктно-орієнтована мова - міг би стати ідеальним інструментом для вирішення подібних завдань, якби не його порівняно низька швидкість виконання програм.

Так була створена мова програмування С++, спочатку отримав назву «Сі з класами» (C with classes). Назва «С++» придумав Рік Месчітті. «++» - це оператор інкремента в С, який ніби натякає на те, що мова С++, щось більше, ніж просто С.

Microsoft вирішила відзначити Міленіум випуском нових програмних продуктів. До 2000 року компанія підготувала промислові версії нових компонентних технологій і рішень в області обміну повідомленнями та даними, а також створення Internet-додатків (COM +, ASP +, ADO +, SOAP, Biztalk Framework) [11]. На підтримку цих нововведень Microsoft випустила інструментарій для розробки додатків - платформу .NET. Вона також об'єднувала «під одним дахом» кілька мов програмування, що було в новинку для того часу.

Ще одним нововведенням платформи .NET була технологія активних серверних сторінок ASP.NET (Active Server Page) [12]. З її допомогою можна було відносно швидко розробити веб-додатки, які взаємодіють з базами даних.

Мова програмування C# був створений спеціально для ASP.NET. На C# повністю була написана і сама ASP.NET.

Назва «Сі Шарп» (від англ. Sharp - дієз) несе «сакральний» сенс. Знак «#» (в музичній нотації читається як «дієз») означає підвищення висоти звуку на півтону. З іншого боку, назва «C#» виходить шляхом наступної «еволюційного ланцюжка»: C → C ++ → C ++++ (C#), так як символ «#» можна скласти з 4-х знаків «+».

C# підтримує всі три «стовпи» об'єктно-орієнтованого програмування: інкапсуляцію, успадкування і поліморфізм. Крім того, в ньому була реалізована автоматична «прибирання сміття», обробки виключень, динамічне зв'язування.

### Особливості мови

При створенні C#, його автори враховували досягнення багатьох інших мов програмування: C++, C, Java [13], SmallTalk [14], Delphi [15], Visual Basic [16] і т.д. Треба зауважити що у зв'язку з тим, що C# розроблявся з чистого аркуша, у його авторів була можливість (якою вони явно скористалися), залишити в минулому всі незручні і неприємні особливості (існуючі, як правило, для забезпечення сумісності), будь-якого з попередніх йому мов . В результаті вийшов дійсно простий, зручний і сучасну мову, за потужністю не поступається С++, але істотно підвищує продуктивність розробок.

Дуже часто можна простежити такий зв'язок - чим більше мова захищена і стійка до помилок, тим менше продуктивність програм, написаних на ній. Наприклад розглянемо дві крайності - очевидно це Assembler [17] і Java. У першому випадку ви можете досягти фантастичною швидкості своєї програми, але вам доведеться дуже довго змушувати її працювати правильно не на вашому комп'ютері. У випадку ж з Java - ви отримуєте захищеність, незалежність від платформи, але, на жаль, швидкість вашої програми навряд чи сумісна з існуючим уявленням про швидкість, наприклад, будь-якого окремого клієнтського застосування (звичайно існують застереження - JIT компіляція і інше). Розглянемо C++ з цієї точки зору - на мій погляд співвідношення в швидкості і захищеності близько до бажаного результату, але на основі власного досвіду програмування я можу з упевненістю сказати, що практично завжди краще понести незначну втрату в продуктивності програми і придбати таку зручну особливість, як «прибирання сміття», яка не тільки звільняє вас від стомлюючої обов'язки керувати пам'яттю вручну, але і допомагає уникнути вам багатьох потенційних помилок у вашому додатку. Насправді скоро «збірка сміття», та й будь-які інші кроки до усунення потенційних помилок стану відмінними рисами сучасної мови. У C#, як в поза сумнівом сучасній мові, також існують характерні особливості для обходу можливих помилок. Наприклад, крім згаданої вище «збірки сміття», там всі змінні автоматично не започатковано середовищем і мають типовий захищеністю, що дозволяє уникнути невизначених ситуацій у разі, якщо програміст забуде ініціалізувати змінну в об'єкті або спробує провести неприпустиме перетворення типів. Також в C# були вжиті заходи для виключення помилок при оновленні програмного забезпечення. Зміна коду, в такій ситуації, може непередбачувано змінити суть самої програми. Щоб допомогти розробникам боротися з цією проблемою C# включає в себе підтримку сумісності версій (versioning). Зокрема, на відміну від C++ і Java, якщо метод класу був змінений, це повинно бути спеціально обумовлено. Це дозволяє обійти помилки в коді і забезпечити гнучку сумісність версій. Також новою особливістю є native підтримка інтерфейсів і успадкування інтерфейсів. Дані можливості дозволяють розробляти складні системи і розвивати їх з часом.

У C# була уніфікована система типів, тепер ви можете розглядати кожен тип як об'єкт. Незважаючи на те, використовуєте ви клас, структуру, масив або вбудований тип, ви можете звертатися до нього як до об'єкта. Об'єкти зібрані в простори імен (namespaces), які дозволяють програмно звертатися до будь-чого. Це означає що замість списку включаються файлів заголовків в своїй програмі ви повинні написати які простори імен, щоб отримати доступ до об'єктів і класів всередині них, ви хочете використовувати. У C# вираз using дозволяє вам не писати кожен раз назву простору імен, коли ви використовуєте клас з нього. Наприклад, простір імен System містить декілька класів, в тому числі і Console. І ви можете писати або назва простору імен перед кожним зверненням до класу, або використовувати using як це було показано в прикладі вище.

Сучасність C# виявляється і в нових кроках до полегшення процесу налагодження програми. Традиційним засобом для налагодження програм на стадії розробки в C++ є маркування великих частин коду директивами #ifdef і т.д. У C#, використовуючи атрибути, орієнтовані на умовні слова, ви можете куди швидше писати налагоджувати код.

### Порівняння з конкурентами

Як і Java, C# спочатку призначався для веб-розробки, і приблизно 75% його синтаксичних можливостей аналогічні мови програмування Java. Його також називають «очищеної версією Java». 10% запозичено з C++, а 5% - з Visual Basic. І близько 10% C# - це реалізація власних ідей розробників.

Незважаючи на досить істотні відмінності між компонентної об'єктної моделлю COM (основного стандарту Microsoft для компонентного проектування і реалізації програмного забезпечення) і моделлю Java, мови програмування має досить багато спільного.

Єдине середовище виконання програм заснована на використанні проміжного мови IL (Intermediate Language - проміжна мова), виконуючого майже ту ж роль, що і байт-код віртуальної машини Java. Використовувані в рамках технології .NET компілятори з різних мов транслюють програми в IL-код. Так само, як і байт-код Java, IL-код [18] є команди гіпотетичної стекової обчислювальної машини. Але є і різниця в пристрої і використанні IL.

По-перше, на відміну від JVM [19], IL не прив'язаний до однієї мови програмування. У складі, попередніх версій Microsoft.NET є компілятори мов С++, С#, Visual Basic. Незалежні розробники можуть додавати інші мови, створюючи свої компілятори в IL-код.

По-друге, IL призначений не для програмної інтерпретації, а для подальшої компіляції в машинний код. Це дозволяє досягти істотно більшої швидкодії програм. Містять IL-код файли несуть достатньо інформації для роботи оптимізуючого компілятора.

Кік Редек з Microsoft вважає С# більш складною мовою, ніж Java. На його думку, «Мова Java була побудована таким чином, щоб уберегти розробника від пострілу у власну ногу.» (англ. «Java was built to keep a developer from shooting himself in the foot.»), а «С# був побудований так, щоб дати розробникові пістолет, але залишити його на запобіжнику.» (англ.« C# was built to give the developer a gun but leave the safety turned on»).

### Версії мови

Робота над C# почалася в грудні 1998 року. Проект отримав кодову назву COOL (C-style Object Oriented Language) [20].

Перша бета-версія C# 1.0 побачила світ влітку 2000 року, а в лютому 2002 разом з Microsoft Visual Studio вийшла остаточна версія мови. Так як C# поєднує кращі сторони попередніх їй популярних мов програмування у вигляді C, Java і C ++, програмістам легко здійснити перехід на C#, ґрунтуючись на знаннях будь-якого з перерахованих мов.

У 2005 році вийшов остаточний реліз версії C# 2.0. Це зміцнило його позиції на ринку. Додавання нових можливостей, таких як анонімні методи, узагальнення, часткові і параметризовані типи значно розширили можливості застосування C#.

Версія 3.0, що вийшла в 2008 році, дозволила C# вирватися вперед в «гонці озброєнь» мов програмування. Серед нововведень C# 3.0 - мова інтегрованих запитів (LINQ) [21]; неявно типізовані змінні і методи розширення; Lambda-вирази, які призначають реалізацію коду делегатам за допомогою нового, більш простого синтаксису. Особливо «відзначилися» Lambda-вирази разом з мовою інтегрованих запитів.

У 2010 році була випущена версія C# 4.0. Її головним доповненням до попередніх версій стали іменовані і необов'язкові аргументи. Перші дають можливість прив'язки аргументу і параметра на ім'я, а другі дозволяють вказати аргумент, який використовується за умовчанням, для кожного параметра. Не менш важливе нововведення - тип dynamic. Він дозволяє здійснювати перевірку щодо відповідності типів об'єктів не на етапі компіляції, а безпосередньо під час виконання програми.

Паралельно з'явилися нововведення в .NET Framework 4.0 - бібліотека розпаралелювання завдань (TPL) [22] і паралельний варіант мови інтегрованих запитів (PLNQ) [23]. Їх підтримка дозволяє здійснити паралельність виконання коду в комп'ютерах з багатоядерними процесорами або декількома одноядерними.

Версія C# 5.0 з'явилася в 2012 році. У ній було реалізовано зовсім небагато нововведень:

* асинхронні методи;
* операція отримання відомостей про зухвалого об'єкті.

Шоста версія була випущена через три роки після виходу п'ятої версії - в 2015. Її основні нововведення:

* в C# 6.0 можна форматувати властивості зі значеннями. Це допоможе уникнути помилки з null і порожніми значеннями властивості;
* кожен день нам доводиться стикатися з конкатенацією рядків. Хтось в основному використовує оператор "+", хтось - метод string.Format(). Але проблеми з ним всім відомі: при занадто великій кількості параметрів важко розуміти, що означають кожне число - {1}, {2}, {3}. В C# 6.0 придумали нову можливість, яка повинна об'єднати переваги обох методів;
* в C# 6.0 властивості і методи можна визначати через лямбда-вирази. Це сильно зменшує кількість коду;

Також Microsoft покращила новий компілятор в плані продуктивності.

У 2016 році стало відомо про нововведення підготовлюваної версії С # 7.0:

* бінарні літерали
* локальні функції
* Зіставлення зі зразком (Pattern matching)
* Тепер можна використовувати різноманітні засоби для зіставлення.
* Умови і використання об'єктів в перемикачах

Маленька революція для розробників. Тепер switch майже нічим не обмежений. Можна використовувати зіставлення.

Найостанніша версія мови програмування C# - це 9 версія, котра вийшла у листопаді 2020 року разом з платформою .NET 5.

## **Вибір середовища розробки для програмування**

Для програмування на мові C# більш за все підходить нативне для Windows середовище Visual Studio 2019. Visual Studio - це інтегроване середовище розробки (IDE) з широкими можливостями для створення додатків для Windows, Linux, Android, iOS та MacOS, а також сучасних веб-додатків і хмарних служб. Професійні інструменти та служби для розробки, призначені для індивідуальних розробників або невеликих груп. Ця інтегрована середовище спрощує виконання складних завдань, що дозволить вам приділяти більше часу і сил на досягнення своїх безпосередніх цілей.

### Історія середовища

Свою історію Visual Studio починає в 1997 році під кодовим ім’ям “Boston”, саме тоді була створена перша версії даного середовища разом з різними засобами розробки на різних мовах. Існувало два типи ліцензій, Professional та Enterprice і була підтримка Visual Basic 5.0, Visual C++ 5.0 Visual J++ 1.1, Visual FoxPro 5.0. Основною перевагою було включення в збірку нової на той час технології ASP.

Visual Studio 6.0 – створена в червні 1998 році під кодовою назвою “Aspen” і була останнім продуктом від Microsoft, який працював на Win9X. Даний випуск був основним для створення додатків на Windows до появи .NET Framework.

Visual Studio .NET випущена в 2002 році під назвою «Rainier», стала першою IDE яка базувалася на .NET платформі, тобто був створений єдиний стандарт для компілювання багатьох мов, серед яких були: С++, С#, VB.NET, J++, J#, F#.

Квітень 2003 запам’ятався тим, що був анонсований NET Framework 1.1 та саме Visual Studio 2003 була одним з додатків створених на цій платформі. Саме на цій платформі пізніше була випущена Microsoft Visio [27].

У Visual Studio 2005 наприкінці жовтня 2005 року вирішили проекспериментувати і випустили express версія з обмеженим функціоналом, але пізніше зробили висновок, що користувачі не оцінили поділ та повернулися до старої схеми, яка використовується й по нині. Через півроку вийшло оновлення з підтримкою усіх баз даних до 2005 року, можливість вибору розрядності операційної системи для певного проекту, WCF [28], WPF [29], LINQ та найбільш очікуваний .NET Framework 3.5

Visual Studio 2008 вийшла у листопаді 2007 року з’явилася можливість взаємодії між VS та MS Office, зокрема можна було створювати розширення. У веб частині з’явився функціонал з підтримкою AJAX/JSON [30] та підтримкою REST архітектури [31].

В 2010, 2012, 2013 були лише незначні виправлення пов’язані з різними вікнами біндингу.

Підтримка фреймворку Unity [32], UWP [33], С++ стандарту 14 та .NET Framework 4.6 можна було побачити у Visual Studio 2015, яка вийшла у світ 20 червня 2015 року.

Вже у Visual Studio 2017 року почали орієнтуватися на спільноту, яка використовує дану IDE, тому проводила бета-тестування серед звичайних програмістів. Основна зміна відбулася у інтерфейсі інсталятора та зовнішньому вигляді самої середи розробки, почалася підтримка платформи .NET Core 1.1-2.2 лише 7 березня 2017 року.

Як і минулого року почали з тестування, перша preview версія Visual Studio 2019 вийшла у грудні 2018 року, мала 5 етапів. Release Candidate був визначений у березні. На цей раз зміни були пов’язані також з дизайном, також було виправлена велика кількість помилок з безпекою, була впроваджена підтримка .NET Core 3.

## **Вибір середовища для графічної частини**

Для графічної частини додатку використовувався Unity. Тому, що Unity - це потужний та популярний інтегрований розвитоковий середовище (IDE), яке широко використовується для розробки ігор, віртуальної реальності (VR), доповненої реальності (AR) та інтерактивних додатків. Ось кілька причин, чому Unity є відмінним вибором для реалізації проекту з генерування лабіринту:

1. Мультиплатформеність: Unity надає підтримку для різних платформ, включаючи ПК, консолі, мобільні пристрої та навіть віртуальну реальність. Це означає, що ви можете створювати свій проект і запускати його на різних пристроях без необхідності переписування коду або відновлення всього проекту.
2. Зручність використання: Unity має дружній інтерфейс та широкий спектр інструментів, які полегшують розробку ігор та взаємодію з компонентами проекту. Ви можете використовувати графічний інтерфейс для створення та налаштування об'єктів, а також використовувати мови програмування, такі як C#, для розробки бізнес-логіки та алгоритмів.
3. Велика спільнота та ресурси: Unity має широку та активну спільноту розробників, яка пропонує безліч онлайн-ресурсів, документацію, плагіни та готові рішення для розв'язання різних завдань. Ви можете знайти підтримку, поради та розв'язки проблем у спільноті Unity, що значно спростить вашу роботу.
4. Графічні можливості: Unity має потужні графічні можливості, що дозволяють створювати вражаючі візуальні ефекти, освітлення та анімацію. Ви зможете візуалізувати ваш лабіринт, створити привабливе оточення та забезпечити захоплюючий геймплей.
5. Розширюваність: Unity дозволяє використовувати сторонні плагіни та інструменти для розширення функціональності вашого проекту. Ви можете легко інтегрувати сторонні бібліотеки.

Unity - це потужний та гнучкий інструмент для розробки ігор та інтерактивних додатків, який надає мені всі необхідні можливості для реалізації вашого проекту з генерування лабіринту.

### Порівняння з конкурентами

Unity є одним з найпопулярніших інструментів для розробки ігор, але на ринку також присутні інші конкуренти. Давайте порівняємо Unity з деякими з них:

1. Unreal Engine: Unreal Engine від Epic Games є одним з основних конкурентів Unity. Він також є потужним інтегрованим розвитковим середовищем, яке надає широкі можливості для розробки ігор у 2D та 3D. Unreal Engine відомий своїми фотореалістичними графічними можливостями та великою кількістю готових ресурсів. Unity, з іншого боку, володіє більш простим інтерфейсом та зручнішою навігацією, що робить його більш доступним для новачків. Вибір між Unity та Unreal Engine залежить від ваших особистих потреб та вимог проекту.
2. Godot Engine: Godot Engine є відкритим джерелом і також є популярним інструментом для розробки ігор. Він відрізняється своєю легкістю використання, гнучкістю та низькими вимогами до апаратного забезпечення. У порівнянні з Unity, Godot Engine має меншу спільноту розробників та менше готових ресурсів, але він забезпечує більшу відкритість та контроль над кодом.
3. CryEngine: CryEngine, розроблений компанією Crytek, відомий своїми високоякісними графічними можливостями та фізичною симуляцією. Він широко використовується для створення реалістичних ігрових світів. Проте, в порівнянні з Unity, CryEngine може бути складнішим у використанні та вимагати більше ресурсів для розробки.

Кожен з цих інструментів має свої особливості, переваги та недоліки. Проте для реалізації цього проєкта я обрав саму Unity. Через ряд переваг, котрі були озвучені вище.

**ПРЕДСТАВЛЕННЯ РЕЗУЛЬТАТІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ**

## **Використані технології**

Під час виконання курсової роботи було використано наступні технології/бібліотеки:

* .NET 5;
* ASP.NET Core 5.0;
* Unity Navigation System;
* Random Number Generation;
* Unity Editor Scripting;
* GitHub 14.7.43 [46].

## **Розробка додатку**

На самому початку маємо визначити алгоритм та спроектувати його в VS:Code. Тому для початку створимо головну модель, щоб можна було успадкувати деякі поля, котрі будуть присутні у всіх моделях

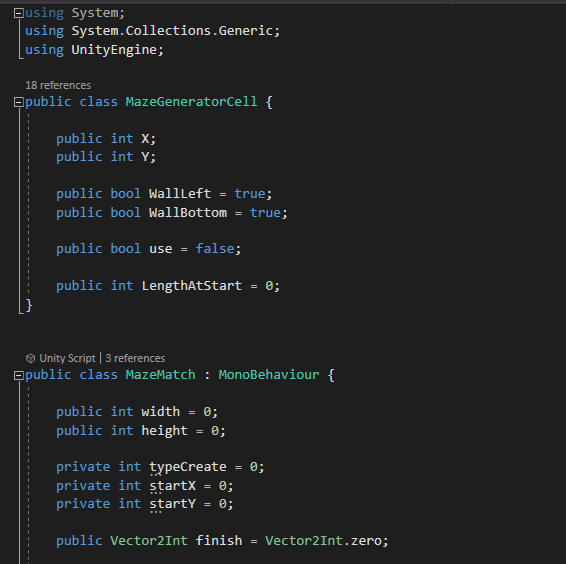


Рисунок 4.1 – Базова модель

Так, базова модель готова, тепер можна створювати об’єкти які будуть використовуватися головною моделлю, почнемо з 1 клітинки(див. Рис. 4.2).

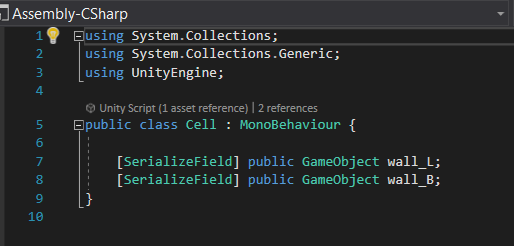


Рисунок 4.2 – Модель 1 клітинки

Продовжимо моделлю системи створення лабіринту(див. Рис. 4.3).

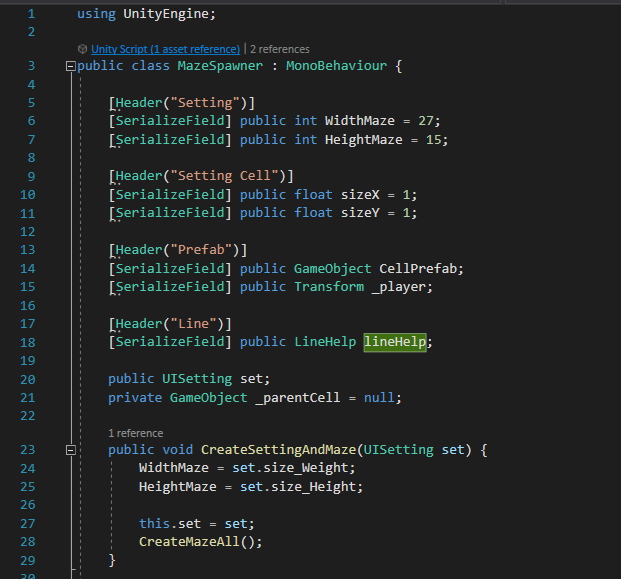
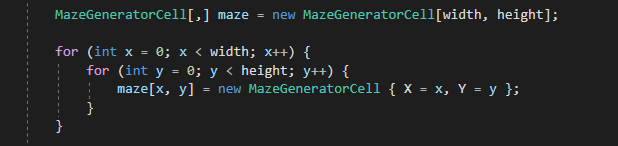


Рисунок 4.3 – Модель виклику генератора

Оскільки ми уже можемо викликати генерацію лабіринта, то проведемо його створення і заповним одинаковими клітинками(Див: Рис.4.4 та Рис. 4.5)

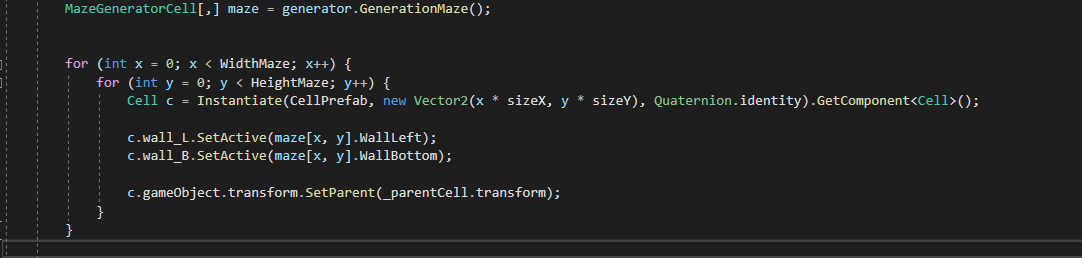
Рисунок 4.4 – Створення масиву і його заповнення(Математично)

Рисунок 4.5 – Виводимо лабіринт в редактор Unity

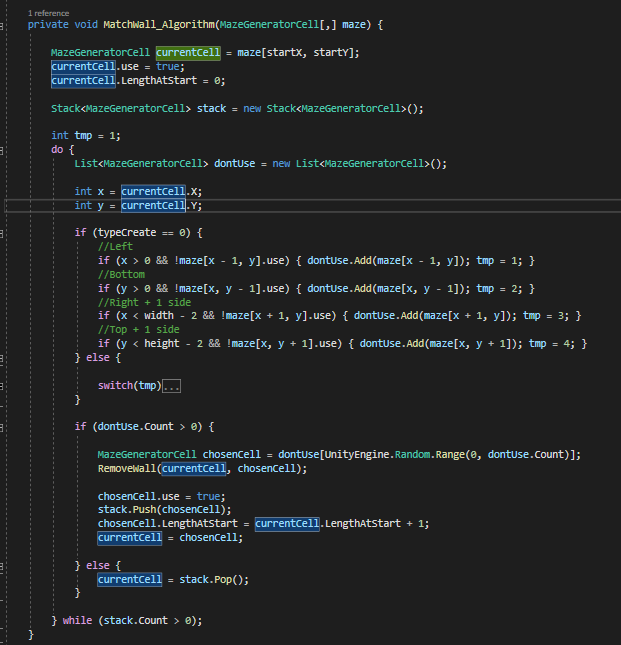
Оскільки перша модель лабіринту закінчена, слід перейти до реалізації головного алгоритму, який і буде створювати структуру нашого лабіринту:

Рисунок 4.6 – Алгоритм для структури лабіринт

Оскільки ми уже маємо повноцінний лабіринт, потрібно добавити спостерігача. Через якого(задопомогою) будемо відслідковувати генерацію лабіринта та будемо вести наші слідкування:

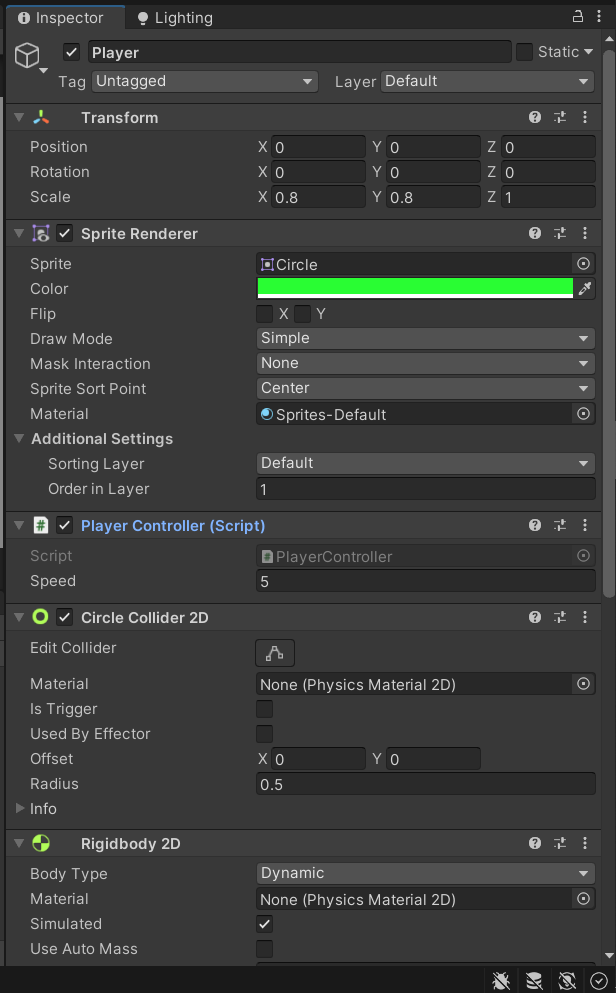


Рисунок 4.7 – Гравець в Unity

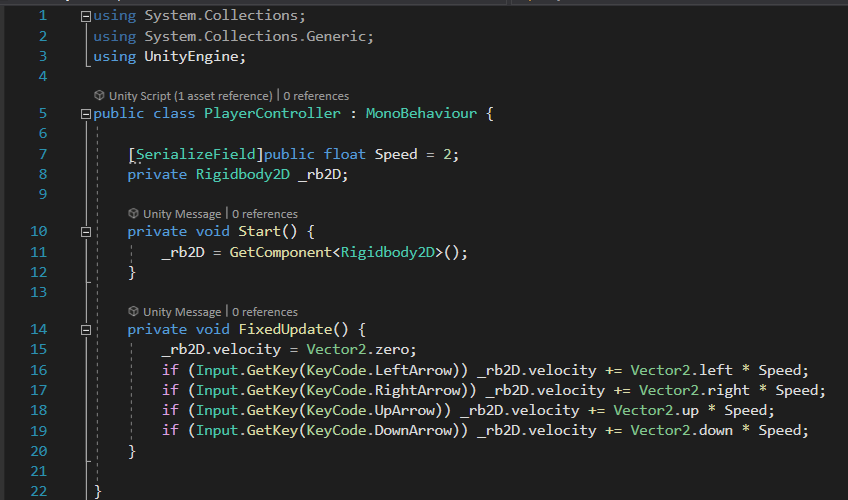
Добави код, який буде відповідати за рух гравця див Рис. 4.8

Рисунок 4.8 – Код для руху гравця

Далі слід створити код для пошуку вихода з лабіринта і його візуалізацію:

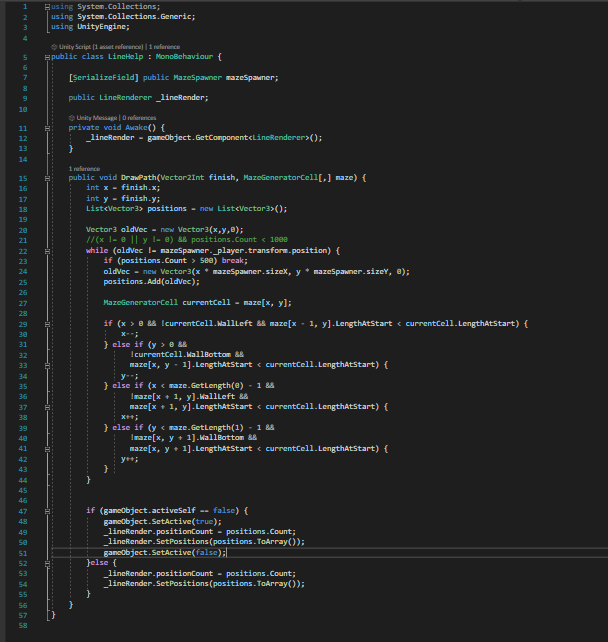


Рисунок 4.8 – Алгоритму пошуку вихода з лабіринта.

Також слід написати допоміжний код, який дасть можливість змінювати розмір лабіринта, стартове місце гравця і його оглядову можливість. Для цього створтимо окремий UISetting.cs файл який буде відповыдати за це.

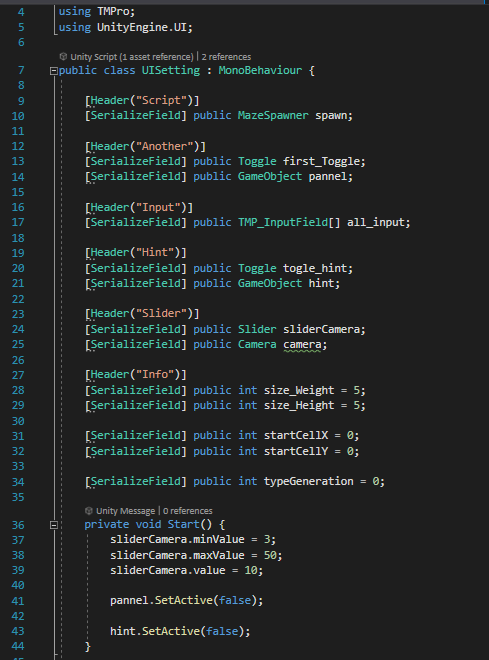


Рисунок 4.9 – Персональні налаштування користуча.

Далі зробимо візуальну частину для кода, що б користувач міх зразу в додатку змінювати ці параметри:

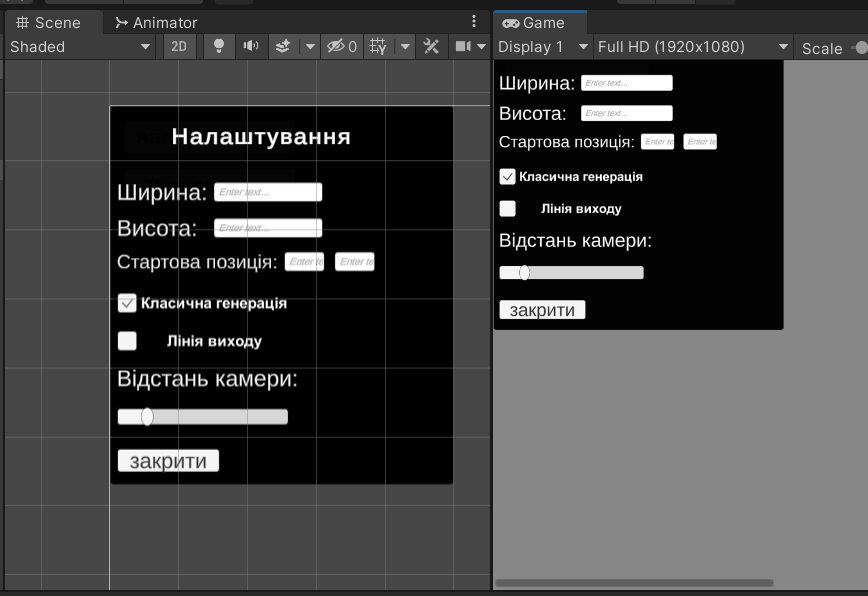


Рисунок 4.10 – Персональні налаштування користуча в Unity.

На останньому етапі створемо просте меню, з якого користувач зможе генерувати свій лабіринт і створимо кнопку вихід

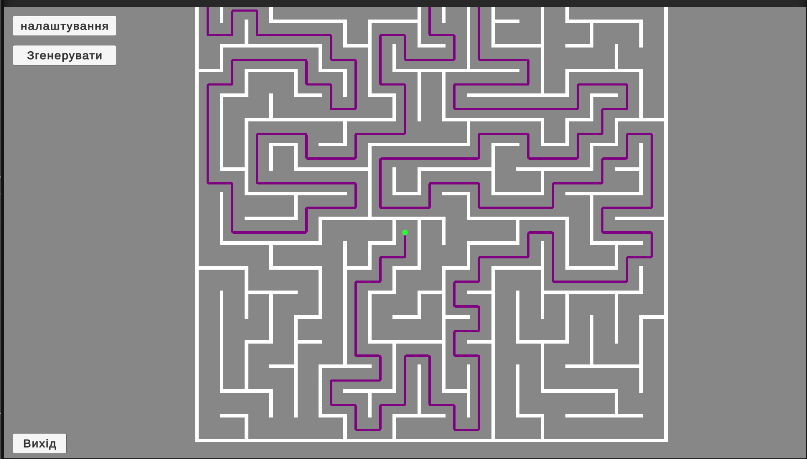
****

Рисунок 4.11 – Додаток в редакторі Unity.

## **Короткий обзор додатку**

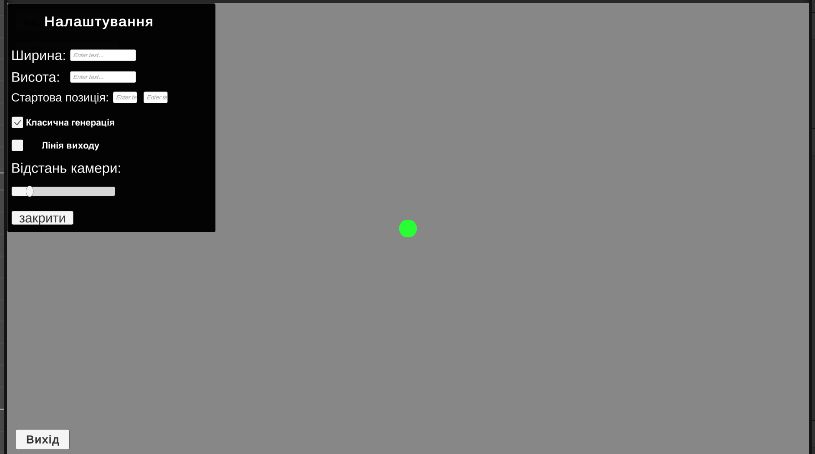
Рисунок 4.12 – Початковий екран.

Рисунок 4.13 – Налаштування.

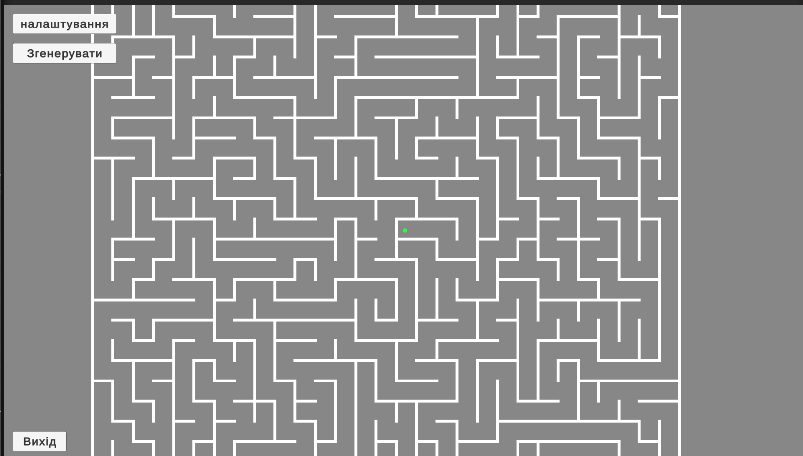


Рисунок 4.14 – Створений лабіринт.

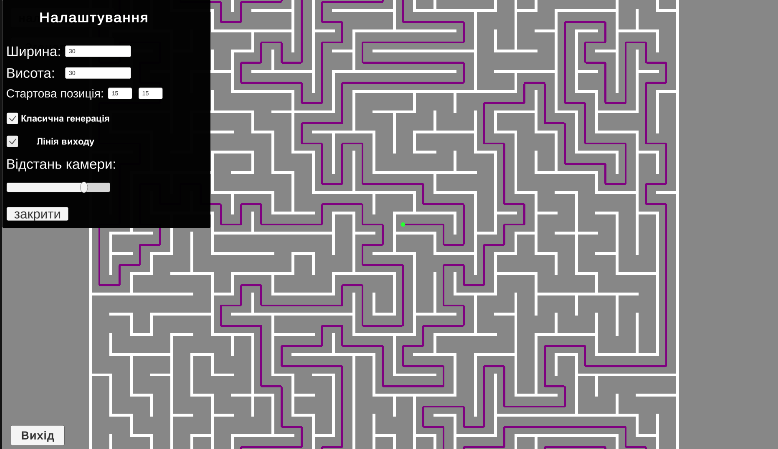


Рисунок 4.15 – Візуальне відобреження виходу з лабіринту.

## **Перспективи проекту**

Проєкт має потенціал для різноманітних перспектив і може бути корисним у різних областях. Ось деякі можливі перспективи проекту:

Ігрова індустрія: Генерація лабіринтів є важливою складовою для створення захоплюючих ігор. Ігрові розробники можуть використовувати ваш генератор лабіринтів для створення випадкових та унікальних рівнів, що дозволяє збільшити повторюваність гри та забезпечити гравцям нові виклики.

Навчання та тренування: Генерація лабіринтів може бути використана для навчання алгоритмів штучного інтелекту (AI). Ваш проект може допомогти в розвитку та тестуванні різних алгоритмів пошуку шляху, змагальних агентів та інших AI-рішень.

Віртуальна та доповнена реальність: Генератор лабіринтів може бути використаний для створення захоплюючих віртуальних або доповнених реальності досвідів. Ви можете розробити ігри, навчальні симуляції або навіть архітектурні додатки, де користувачі можуть досліджувати та навчатися в лабіринтових середовищах.

Навчання географії та орієнтування: Генерація лабіринтів може бути використана для навчання та вивчення географії, орієнтування та картографії. Ваш проект може допомогти створити віртуальні лабіринти, де користувачі будуть навчатися визначати своє місцезнаходження, розуміти картографічні показники та виконувати завдання на основі орієнтування.

Дослідницькі проекти: Генерація лабіринтів є цікавою темою для дослідницьких проектів у галузі комп'ютерної графіки, алгоритмів, штучного інтелекту та інших суміжних областей. Ваш проект може послужити основою для подальших досліджень та розвитку нових методів генерації та розв'язання проблем лабіринтів.

Загалом, генератор лабіринтів має широкі можливості застосування та потенціал для розвитку у різних сферах. Важливо продовжувати досліджувати, вдосконалювати та впроваджувати нові ідеї для забезпечення успіху проекту.

# **ВИСНОВКИ**

У данній курсовій роботі була розглянута тема генерування лабіринту та пошуку виходу з нього з використанням мови програмування C# та платформи Unity. Було досліджено історію створення та використання лабіринтів, а також оглянуто перші алгоритми їх створення.

Unity була обрана для реалізації проекту через свою популярність, широкі можливості розробки ігор, зручний інтерфейс та підтримку мови програмування C#. Порівняння з конкурентами показало, що Unity має свої переваги у доступності та розширюваності.

Проект генератора лабіринтів виявився перспективним у різних галузях, включаючи ігрову індустрію, навчання та тренування, віртуальну та доповнену реальність, навчання географії та орієнтування, а також дослідницькі проекти. Він може бути використаний для створення захоплюючих ігрових рівнів, навчальних симуляцій, дослідження алгоритмів та багато іншого.

В результаті цієї роботи було розроблено генератор лабіринту, який дозволяє створювати випадкові та унікальні лабіринти, а також реалізовано алгоритми пошуку виходу з них. Проект має потенціал для подальшого розширення, вдосконалення та використання у різних сферах.

Загалом, данна курсова робота дозволила детально вивчити тему генерування лабіринту та пошуку виходу з нього, а також успішно реалізувати проект з використанням C# і Unity. Вона стала важливим етапом у розвитку навичок програмування та розробки ігор, а також відкрила нові можливості для подальшого дослідження та використання генератора лабіринтів.

# **ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**

1. QR-Code [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/QR-%D0%BA%D0%BE%D0%B4>
2. C# [Електронний ресурс]. – 1998. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/C_Sharp>
3. .NET [Електронний ресурс]. – 2000. – Режим доступу до ресурсу:
4. .NET Framework [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/.NET_Framework>
5. .NET Core [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://ru.wikipedia.org/wiki/.NET_Core>
6. Microsoft [Електронний ресурс]. – 1995. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Microsoft>
7. MIT [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81](https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B0%D1%87%D1%83%D1%81%D0%B5%D1%82%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D1%82%D0%B5%D1%85%D0%BD%D0%BE%D0%BB%D0%BE%D0%B3%D1%96%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82)
8. UNIX systems [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/UNIX>
9. IBM [Електронний ресурс]. – 1990. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/IBM>
10. C++ [Електронний ресурс]. – 1999. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/C%2B%2B>
11. Інтернет [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Інтернет](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82)
12. ASP.NET [Електронний ресурс]. – 1989. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/ASP.NET>
13. Java [Електронний ресурс]. – 1995. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Java>
14. SmallTalk [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Smalltalk>
15. Delphi [Електронний ресурс]. – 2002. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Object_Pascal>
16. Visual Basic [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Visual_Basic>
17. Assembler [Електронний ресурс]. – 1999. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Асемблер](https://uk.wikipedia.org/wiki/%D0%90%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80)
18. IL code [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Common_Intermediate_Language>
19. JVM [Електронний ресурс]. – 2000. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Віртуальна\_Машина\_Java](https://uk.wikipedia.org/wiki/%D0%92%D1%96%D1%80%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_Java)
20. COOL [Електронний ресурс]. – 1997. – Режим доступу до ресурсу: [https://www.donnfelker.com/Firstname\_of\_C#](https://www.donnfelker.com/how-did-c-get-its-name/#:~:text=The%20C%23%20language%20was%20originally,C%20like%20Object%20Oriented%20Language%27.)
21. LINQ [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/LINQ>
22. TPL [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: [https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-parallel-library-tpl)
23. PLINQ [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: [https://docs.microsoft.com/en-us/dotnet/standard/introduction-to-plinq](https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq)
24. .NET 5 [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: <https://dotnet.microsoft.com/download/dotnet/5.0>
25. Windows [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Microsoft_Windows>
26. Visual Studio [Електронний ресурс]. – 2013. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio>
27. Microsoft Visio [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Microsoft_Visio>
28. WCF [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Windows_Communication_Foundation>
29. WPF [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Windows_Presentation_Foundation>
30. Ajax/JSON [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: <https://ru.reactjs.org/docs/faq-ajax.html>
31. RESTful [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/REST>
32. Unity [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Unity](https://uk.wikipedia.org/wiki/Unity_(%D1%80%D1%83%D1%88%D1%96%D0%B9_%D0%B3%D1%80%D0%B8))
33. UWP [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/UWP](https://ru.wikipedia.org/wiki/%D0%A3%D0%BD%D0%B8%D0%B2%D0%B5%D1%80%D1%81%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D0%B0_Windows)
34. JetBrains Rider [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://www.jetbrains.com/rider/>
35. NetBeans [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/NetBeans>
36. PHP [Електронний ресурс]. – 2007. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/PHP>
37. Python [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Python>
38. Entity Framework Core [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: <https://github.com/dotnet/efcore>
39. MediatR [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: <https://github.com/jbogard/MediatR>
40. Swagger [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://github.com/domaindrivendev/Swashbuckle.WebApi>
41. AutoMapper [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://github.com/AutoMapper/AutoMapper>
42. Serilog [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: <https://github.com/serilog/serilog>
43. GitHub [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/GitHub>
44. СУБД [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/СУБД](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)
45. Багато-до-багатьох [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: <https://ukr.kagutech.com/4138595-many-to-many-relationship-an-example-in-access-in-sql-how-to-make-a-many-to-many-relationship>
46. Repository [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: <https://streletzcoder.ru/realizatsiya-patterna-repozitoriy-repository-v-c/>
47. DI Container [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: [https://docs.microsoft.com/en-us/aspnet/core/fundamentals/ dependency-injection?view=aspnetcore-5.0](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-5.0)
48. API [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/API](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81)
49. Middleware [Електронний ресурс]. – 2009. – Режим доступу до ресурсу: [https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/? view=aspnetcore-5.0](https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-5.0)
50. OAuth 2.0 [Електронний ресурс]. – 2012. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/OAuth>
51. Request Headers [Електронний ресурс]. – 2005. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Заголовки\_HTTP](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BA%D0%B8_HTTP)
52. Закон України Про захист персональних даних [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: [https://zakon.rada.gov.ua/laws/show/ 2297-17](https://zakon.rada.gov.ua/laws/show/2297-17#Text)
53. SPA [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: [https://uk.wikipedia.org/wiki/Односторінковий\_застосунок](https://uk.wikipedia.org/wiki/%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D1%82%D0%BE%D1%80%D1%96%D0%BD%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B7%D0%B0%D1%81%D1%82%D0%BE%D1%81%D1%83%D0%BD%D0%BE%D0%BA)

# **ДОДАТОК А**

Весь вихідний код можна знайти на сторінці GitHub за посилання:

https://github.com/Pro100Vlad1047/Course\_project