УДК 004.8+65.05+681.5

DOI: 10.31673/2412-9070.2022.011725

Ю. І. КАТКОВ, доктор техн. наук, доцент;
О. Ю. ІЛЬЇН, доктор техн. наук, професор;
І. В. РЕЗНІЧЕНКО, студент;
В. О. ВИШНІВСЬКИЙ, студент,
Державний університет телекомунікацій, Київ

РОЗРОБЛЕННЯ КОМП'ЮТЕРНИХ ІГОР ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ ІГРОВОГО ШТУЧНОГО ІНТЕЛЕКТУ

Статтю присвячено актуальному питанню пошуку нових ефективних методів і вдосконаленню теперішніх доволі поширених способів розроблення комп'ютерних ігор із використанням технологій штучного інтелекту. Головну увагу приділено пошуку та підвищенню ефективності застосування інструментів для побудови комп'ютерних ігор завдяки впровадженню сучасних технологій. Наведено таку постановку завдання: застосування наявних наукових досліджень у галузі штучного інтелекту — один із вірних шляхів розв'язання складних завдань в ігровій індустрії. Відомо, що вимоги сьогоднішніх гравців до комп'ютерної гри є досить вагомими, тому розробникам потрібно зосереджувати свою увагу на таких нововведеннях, які можуть забезпечити впровадження нових технологій ігрового штучного інтелекту. Для цього потрібно проаналізувати способи побудови гри, розробити алгоритми оцінювання поточного рівня вмінь гравця, щоб запропоновувати йому під час гри ситуації, які надають більше цікавості ігровому процесу. Вирішення завдання дає змогу зменшити обчислювальну складність та підвищити якість рішень, що приймаються, у реальному масштабі часу. Для досягнення цієї мети в статті виконано: опис методів використання технологій ігрового штучного інтелекту, розглянуто логіку реалізації цієї гри різними методиками, розкрито питання чи є ігровий штучний інтелект підгалуззю великої галузі штучного інтелекту, описано способи машинного навчання та надано пояснення чому машинне навчання не завжди доцільно використовувати в процесі розроблення ігор, описано основні напрямки розвитку ігрового штучного інтелекту в подальшому розробленні комп'ютерних ігор, а також розглянуто принцип побудови ігор із використанням алгоритму оцінювання вмінь гравця та ігрового рушія Unity.

Ключові слова: GameDev; Game artificial intelligence.

ВСТУП

Інтелектуальні технології безперечно є флагманом розвитку цілої низки галузей: автоматизації промислових процесів, хмарних рішень, медицини та багатьох інших, серед яких є індустрія онлайн-ігор (GameDev). Відомо, що основою інтелектуальних технологій є різноманітні технології штучного інтелекту (*Artificial Intelligence Technologies* — **AI**). Вони надають великі можливості розробникам програмного забезпечення, дають змогу використовувати штучний інтелект для вирішення складних задач, які раніше здавалося неможливо розв'язати. Загалом AI можна пояснити так: програмному комплексу надаються деякі початкові дані досліджень певної галузі; на основі цих даних комп'ютер за допомогою спеціального програмного забезпечення знаходить певні закономірності, а отже, починає відрізняти одні речі від інших.

Сьогодні AI використовується в багатьох напрямках індустрії онлайн-ігор GameDev. Наприклад, знаходить застосовування для створення віртуальної реальності та ігор із доповненою реальністю (*Virtual Reality and Augmented Reality Gaming*), що уможливлює істотне підвищення якості графіки та природності динаміки різних об'єктів: людей, транспорту, тварин, погодних проявів. Так, компанія Oculus, яка розробляє окуляри віртуальної реальності, використовує технологію розпізнавання об'єктів на картинці. А отже, в окулярах віртуальної реальності Oculus Quest і Oculus Quest 2 було реалізовано фічу під назвою «Hands Tracking», що дає змогу розпізнавати руки гравця за допомогою камер, влаштованих в окуляри. Це дає можливість отримувати координати їх положення в просторі і саме положення пальців та передавати цю інформацію системі, якою можна потім скористатися під час розроблення ігор. Також AI застосовується в технології машинного навчання (machine learning), що дає змогу виявляти найбільш релевантні інтегральні показники, котрі відповідають за моделювання деяких процесів, зумовлюючи появу в нових іграх надзвичайно реалістичної графіки. У технології блокчейн (*Blockchain Technology*) AI дає змогу записувати інформацію так, щоб ускладнити або унеможливити зміну, злом чи обман системи. У процесі впровадження технології AI для створення персональних мобільних пристроїв (*Wearables Technology*), які з'єднують реальний та цифровий світи за допомогою технології Infineon, забезпечується раціональне споживання енергії, мобільність та безпека цих пристроїв. Також AI застосовується в мобільних ігрових програмах, які перебувають в IoT (Internet of Things) пристроях.

В індустрії онлайн-ігор застосовується ігровий штучний інтелект (*Game artificial intelligence* — **GAI**) у відеоіграх (*Artificial intelligence in video games* — **AIVG**). Сьогодні відеоігри є однією з найбільш динамічних та технологічних галузей світової економіки, що перебуває на стику цілої низки сфер: програмування, психології, маркетингу, математики, дизайну тощо. Ігровий штучний інтелект надає гравцю відчуття ілюзії того, що всі інші ігрові персонажі здатні думати. Важко уявити хороший шутер, в якому не було «розумних» керованих комп'ютером ботів або неігрових персонажів. Наприклад: в шутерах від першої особи ігрові персонажі вміють добре стріляти і реагувати на певні ситуації. Хоча варто зазначити, що ця навичка стрільби зараз використовує певні відхилення і затримку під час стрільби, без яких ігровий персонаж, керований комп'ютером, зможе миттєво прицілюватися і не давати жодного шансу людині-гравцю на виграш, що робить гру нецікавою. Також у шутерах ботам властива чудова реакція на наближення людини-гравця. Вони вміють створювати засідки, кидати в потрібний момент гранати, знаходити укриття під час обстрілу, навіть діяти командою. Потрібно зауважити, що сучасною особливістю застосування AI в індустрії онлайн-ігор є те, що GAI використовують на благо гравців на етапі їх навчання (коли ті перший раз грають у гру) або під час тренувань щодо вдосконалювання гравцями своїх навичок. Чудова думка про головне завдання GAI була в Тимура Бухараєва: «Головним завданням ігрового штучного інтелекту є не виграти у гравця, а красиво йому «піддатися» [20]. Для забезпечення «піддатися» нові ігрові технології використовують кращий досвід, надають більш цікавий ігровий процес, можливість гравцю навчатися. Те, що раніше здавалося неможливим, сьогодні є доступним у віртуальній реальності. Наступним кроком упровадження GAI — це створення можливості для розробників комп'ютерних ігор уводити мультиплатформеність для своїх ігор, що дає змогу гравцям грати в одну й ту саму гру не тільки на стаціонарних комп'ютерах, а й на планшетах, телефонах та інших мобільних пристроях. Мультиплатформеність для ігор створюється завдяки вебзастосункам, даючи змогу користувачу грати в ігри прямо у веббраузері, наприклад застосовуючи для розробки прикладний програмний інтерфейс WebGL.

Отже, за останні кілька років індустрія онлайн-ігор досягла великих успіхів завдяки впровадженню технологій ігрового штучного інтелекту. Це швидко скорочує розрив між відеоіграми та реальністю, надаючи геймерам незабутні враження. Такий стрімкий розвиток онлайн-ігор ґрунтується на інтелектуальних технологіях на основі штучного інтелекту, які створюють віртуальні умови для онлайн-ігор, упроваджують нові ігрові функції або варіанти мобільних ігор. Тому ***метою статті*** є аналіз особливостей застосування ігрового штучного інтелекту, здійснення аналітичного огляду методів і алгоритмів ігрового штучного інтелекту та оцінювання подальшого розвитку цієї галузі [1].

Постановка завдання

Застосування сучасних наукових досліджень в галузі AI — один з ефективних шляхів розв'язання складних завдань у GameDev. Відомо, що вимоги сучасних гравців до комп'ютерної гри постійно зростають. Тому розробникам потрібно зосереджувати свою увагу на таких нововведеннях, які можуть забезпечити впровадження нових технологій GAI. Отже, потрібно здійснити аналіз способів побудови гри, розроблення алгоритмів оцінювання поточного рівня вмінь гравця, щоб запропоновувати йому під час гри ситуації, які підвищуватимуть зацікавленість до ігрового процесу. Вирішення цього завдання дає можливість зменшити обчислювальну складність та підвищити якість рішень, що приймаються, у реальному масштабі часу.

Аналіз останніх наукових досліджень

Сьогодні існує чимало різних алгоритмів, якими можна описати поведінку ігрового персонажа. Виокремимо деякі з них: алгоритми пошуку шляху Дейкстри [2]; поведінкові алгоритми (дерева розв'язків, скінченні автомати) [3; 4]; різні методи машинного навчання (навчання з учителем, без учителя, з підкріпленням) [5; 6]. Але треба зважати на те, що існує багато суперечностей у питанні — чи потрібно розглядати ігровий штучний інтелект як підгалузь загального штучного інтелекту?

Ці суперечності спричинені тим, що зазвичай ігровий штучний інтелект є набором певних алгоритмів, які заздалегідь визначають поведінку персонажа гри і не використовують машинне навчання для опису поведінки та оцінювання рівня ігрового персонажа (гравця) для застосування перспективних засобів побудови ігор, що надають більше цікавості ігровому процесу, та інших більш важких завдань. Це можна пояснити тим, що даних, яких можна здобути з ігрових об'єктів, достатньо для того, аби побудувати цікавий ігровий процес. Так, наприклад, не потрібно вводити в гру цілу нейронну мережу для того, щоб навчити «ворога» відрізняти гравця від інших об'єктів, йому просто можна передати позицію цього гравця в тривимірній системі координат. Крім того, машинне навчання в цьому разі займало б чимало обчислювальних ресурсів комп'ютера, що призвело б до інформаційної надмірності — затримки дій. Більш того, GAI знає точні дані стосовно

гравця, тому він мало залишає шансів гравцеві на перемогу, а отже, постає проблема щодо створення цікавості гри для гравця (він програє завжди, тому гра стає не цікавою). Цю проблему можна розв'язати через зменшення розрахункової точності, затримку дій комп'ютера-гравця, керованого штучним інтелектом, але чи це доцільно? Водночас в іграх, де є важлива творчість, наприклад у стратегіях, перегонах, персонажам завдяки штучному інтелекту стали надавати певні переваги, аби компенсувати нестачу мислення. Так, наприклад, у грі жанру «стратегія», коли ресурси справжнього гравця значно перевищують ресурси віртуального гравця, останній може раптово отримати певну кількість ігрових ресурсів, щоб наздогнати справжнього гравця [7-19].

Отже, реалізація GAI істотно впливає на геймплей, системні вимоги та бюджет гри, і розробники балансують між цими вимогами, намагаючись зробити цікавий та невибагливий до ресурсів ігрового штучного інтелекту досвід малою ціною. Тому підхід до застосування методів GAI серйозно відрізняється від підходу до традиційного штучного інтелекту, де широко використовуються різні спрощення, підлаштування певних подій. Звідси треба шукати нові вирішення. Одним із підходів до розв'язання цього завдання є потреба розробити алгоритм оцінювання поточного рівня вмінь гравця, щоб запропоновувати йому під час гри ситуації, які надають більше цікавості ігровому процесу. Для цього потрібно проаналізувати можливі принципи побудови GAI.

ОСНОВНА ЧАСТИНА

Ігровий штучний інтелект **(***Game artificial intelligence*, **GAI)** — це набір певних програмних алгоритмів, математичних функцій, які під час застосування надають ігровому персонажу поведінку, схожу до поведінки реальної людини-гравця. Така технологія об'єднує в собі як просту класичну побудову ігор за певним заданим алгоритмом, так і новітні досягнення сучасних інформаційних технологій, зокрема нейронні мережі, робототехніка.

GAI пройшов досить тривалий відрізок удосконалення, перш ніж стати таким, яким він використовується в сучасних іграх. Спочатку він являв собою певні шаблони, яких ігрові персонажі повинні були дотримуватися і які не завжди були коректними. Проте завдяки роботі над помилками з кожною новою версією гри він ставав все кращим. Новим кроком вважається залучення нейронних мереж, які дають змогу вивчати поведінку гравця і виконувати певні відповідні дії. З кожною новою версією гри з'явилася потреба в залученні ігрового штучного інтелекту в такий спосіб, щоб дедалі більше зацікавити гравця. Передусім постало питання застосування штучного інтелекту в нових напрямках ігрової індустрії, а саме: використання GAI для навчання гравців, коли ті перший раз грають у гру або просто хочуть удосконалити свої навички. Тому основним завданням GAI стало не виграти у гравця, а ефектно йому піддатися.

Персонажі з комп'ютерних ігор, керованих GAI, поділяють на неігрові персонажі (*Non-player character*, **NPC)** — це персонажі, які за своєю поведінкою є дружніми або нейтральними до гравця; боти **(Bot)** — вони зазвичай є ворожими до гравця, але й є випадки, коли боти допомагають гравцю в певний ігровий момент; а також на моби **(Mob)** — це вороги гравця, які є менш складними для програмування, зазвичай гравець «вбиває» їх для отримання очок, досвіду, відкриття нових досягнень.

Основним принципом, що лежить в основі роботи GAI, є прийняття оптимального вирішення. Для вибору таких вирішень система має давати вказівки об'єктам (NPC, Bot, Mob) з допомогою GAI. Щоб GAI міг приймати осмислені вирішення, йому потрібно будь-як сприймати середовище. У найпростіших системах таке сприйняття може обмежуватися простою перевіркою положення об'єктів у віртуальному середовищі. У більш складних системах потрібно визначати основні показники якості ігрового світу, зокрема можливі маршрути для пересування, наявність природних сховищ біля ділянки конфліктів. При цьому розробникам необхідно вигадувати спосіб виявлення та визначення основних властивостей ігрового світу, важливих для GAI-системи. Наприклад, сховища на місцевості можуть бути визначені дизайнерами рівнів чи заздалегідь розраховані під час завантаження або компіляції карти рівня. Деякі елементи потрібно розраховувати в режимі реального часу, наприклад карти конфліктів і найближчі загрози.

У сучасних іграх існує кілька способів побудови систем GAI для вибору вирішень: алгоритмічний підхід (традиційний підхід, що передбачає систему простих або умовних правил, алгоритми пошуку шляху, дерева розв'язків, скінченний автомат тощо) та адаптивний підхід (передбачення, сприйняття та пошук шляхів, що вимагає наявності зору, слуху, тимчасових сховищ тощо, тобто все те, що залучається штучними нейронними мережами).

Варто зауважити, що основним завданням у цьому разі є зацікавлення гравця, тобто потрібно зробити так, аби ігровий процес став більш різноманітним і захопливим. А отже, основна мета GAI у грі — це не перемогти гравця, а влучно йому «піддатися». Для ілюстрації цього підходу розглянемо один з яскравих прикладів — ближній бій у різних жанрах ігор. У багатьох іграх для створення

ситуації «влучно піддатися» «вороги» не відразу всі нападають на гравця, а використовують певну тактику, в такий спосіб надаючи гравцю певну перевагу. Залежно від рівня навичок у грі гравцю ігровий штучний інтелект надає відповідні умови гри (переваги або поразки). Дійсно, не зовсім було б цікаво грати в гру, де гравець виграв або програв кілька разів поспіль. Тому після кожної перемоги/поразки GAI приймає рішення про рівень складності гри: він або зростає до рівня потенційної невдачі, або зменшується до рівня потенційної перемоги.

Постає питання, який спосіб побудови систем GAI для вибору вирішень доцільно застосувати: алгоритмічний підхід або адаптивний підхід.

Відомо, що взагалі ***алгоритмічний підхід*** — це виконання певної послідовності дій над певними вихідними даними для досягнення певного результату після завершення. Алгоритмічні методи є методами використання певних наперед заданих дій, які комп'ютеру потрібно виконати. ***Адаптивний підхід*** більш точно відбиває штучна нейронна мережа (*artificial neural networks*, **ANN**) або конекціоністські системи (*connectionist systems*, **CS**) — це спеціальні обчислювальні системи, які мають можливість навчатися завдяки нагромадженню даних про об'єкти у вигляді суджень, що порівнюються. У результаті вони здатні створювати доречний шлях розв'язку завдання і відшукувати певні відповідності між заданими даними. Природно, що таку функцію більш доцільно доручити нейронній мережі, яка навчається на діях гравця і за допомогою певних суджень вирішує, яку тактику (сценарій) застосувати проти нього. Часто алгоритмічні та адаптивні методи використовуються разом. Тому так можна створити певне дерево розв'язків для ігрового персонажа і за певних його умов застосувати нейронну мережу для вдосконалення всього ігрового процесу. Звідси можна дійти висновку про те, що залежно від способу вирішення завдання у грі треба застосовувати відповідний метод. Тому розглянемо особливості використання цих методів окремо.

1. ***Метод пошуку шляхів.*** Пошук шляху — це алгоритмічний метод, його алгоритм використовується для побудови маршруту на карті між заданими точками A і B, беручи до уваги при цьому всі завади, які можуть трапитися на шляху. Зазвичай використовується в іграх жанру стратегії, де користувач за допомогою миші курсора задає певну точку руху. Одним із типових завдань, яке вирішує даний алгоритм, є пошук виходу з лабіринту. Для розв'язання задачі пошуку шляху використовують різні алгоритми, зокрема алгоритми теорії графів, а саме: breadth-first search, depth-first search, алгоритм пошуку A^*, алгоритм Дейкстри, алгоритм пошуку D^* тощо.

1.1. *Алгоритм breadth-first search, depth-first search.* У процесі використання одного з алгоритмів теорії графів (breadth-first search, depth-first search) кожна клітинка являтиме собою вершину графа, а ребра — це шляхи, які сполучають вершини. Кожному ребру надано певне число, що дорівнює довжині переміщення між вершинами. Потім на цьому графі застосовують один із алгоритмів. Під час використання breath-first і depth-first відбувається перебирання всіх можливих шляхів від точки A до B, з яких вибирається мінімальний шлях. Вагома різниця в цих алгоритмах полягає в тому, що в breath-first прохід по вершинах графа здійснюється поступово. Тобто, якщо як початкова точка вибирається вершина $v1$, то вона має ребра з вершинами $l1$, $l2$, де вершина $l1$ має ребра з вершинами $k1$, $k2$, а вершина $l2$ має ребра з вершинами $k3$, $k4$ [8]. Граф зображено на рис. 1.

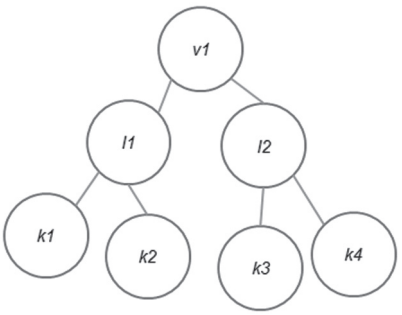

Рис. 1. Зображення графа breadth-first search, depth-first search

На зображенні графа breadth-first search, depth-first search наведено прохід для алгоритму, який має такий вигляд: $v1 \dashrightarrow l1 \dashrightarrow l2 \dashrightarrow k1 \dashrightarrow k2 \dashrightarrow k3 \dashrightarrow k4$. Для depth-first спочатку потрібно пройтися по всіх сусідах по черзі в глибину. У цьому разі алгоритм пройде вершини так: $v1 \dashrightarrow l1 \dashrightarrow k1 \dashrightarrow k2 \rightarrow l2 \dashrightarrow k3 \dashrightarrow k4$.

1.2. *Алгоритм пошуку A^** (англ. *A* star) — алгоритм пошуку, який знаходить у зваженому графі маршрут найменшої вартості від початкової вершини до вибраної кінцевої [8]. Пошук A^* — з погляду математики — це алгоритм пошуку на графі, який знаходить маршрут із найменшою вартістю (наприклад, відстанню) від однієї вершини до другої (рис. 2).

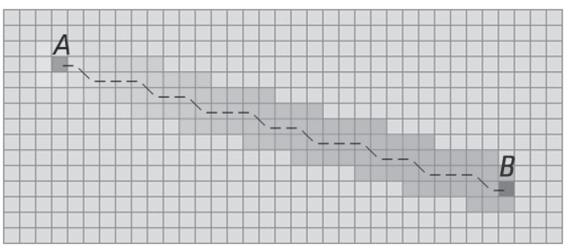

Рис. 2. Зображення графа алгоритму пошуку A^*

Порядок проходу вершин визначається евристичною функцією «відстань + вартість». Для прикладу, якщо розглянути алгоритм пошуку A^*,

то для його використання спочатку ігрову карту розбивають на сітку певного розміру, де кожна клітинка може містити тип місцевості, певні завади, які унеможливлюють рух. Усі ці показники враховуються під час роботи алгоритму. Даний етап є основою для більшості алгоритмів пошуку шляху. Кожній клітинці також присвоюється значення, що дорівнює довжині шляху, котрий потрібно пройти від сусідньої клітинки. Рухаючись від точки *A*, алгоритм для руху вибирає ту клітинку, до якої шлях є коротшим і від якої відстань до точки *B* є мінімальною. Отже, під час вибору наступної клітинки для руху порівнюється значення, що дорівнює сумі загальної довжини шляху до клітинки, тобто сумі шляху, який гравець вже пройшов, і сумі шляху, який потрібно пройти до вибраної клітинки, та відстані до кінцевої точки. При цьому клітинки, в яких гравець уже побував, не будуть враховуватися в подальших обчисленнях. Відстань до кінцевої точки може бути подана як в одиницях довжини (метри, кілометри), так і в кількості клітинок.

1.3. *Алгоритм Дейкстри.* У процесі використання алгоритму Дейкстри застосовують тактику, схожу до алгоритму *A** з певними відмінностями: спочатку відстань від початкової вершини до кожної іншої вершини позначають нескінченно великим числом, а вершину, з якої стартують — нулем. Кожне ребро містить число, що дорівнює довжині переміщення між вершинами (рис. 3).

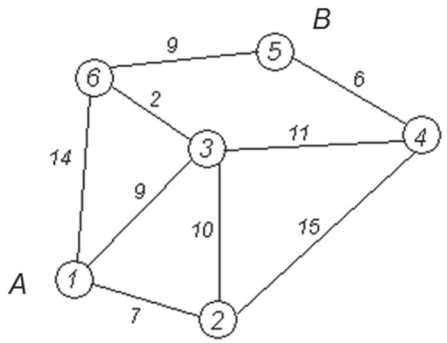

Рис. 3. Зображення графа алгоритму Дейкстри

Під час кожної ітерації для кожної вершини обчислюють число, що дорівнює сумі шляху від початкової вершини до вершини, в якій наразі перебуває гравець, і сумі довжини ребер до вибраної вершини. Якщо таке число буде меншим, ніж число, вже присвоєне вершині, то число замінюється на обчислене. Потім для переміщення вибирають ту вершину, яка містить найменше число, обчислене вже розглянутим способом. Ці числа зазначають, яку мінімальну відстань потрібно пройти від початкової вершини, аби досягти дану вершину. У результаті після всіх ітерацій алгоритму дістанемо мінімальний шлях від початкової точки *A* до *B*.

1.4. *Алгоритм пошуку D**. Алгоритм пошуку *D** (англ. *D* star) — це будь-який із пов'язаних алгоритмів інкрементного пошуку (рис. 4): оригінальний алгоритм пошуку *D**, розроблений Ентоні Стенцем, — це поінформований алгоритм інкрементного пошуку; сфокусований алгоритм пошуку *D** — це інкрементно-евристичний пошук, який поєднує ідеї *A** і оригінального *D**; полегшений *D** — це алгоритм інкрементально-евристичного пошуку, створений Свеном Кенігом та Максимом Лихачовим, який засновано на алгоритмі *LPA** (Label propagation is a semi-supervised machine learning algorithm), алгоритмі інкрементального евристичного пошуку, що поєднує ідеї алгоритму пошуку *A** та динамічного SWSF-FP.

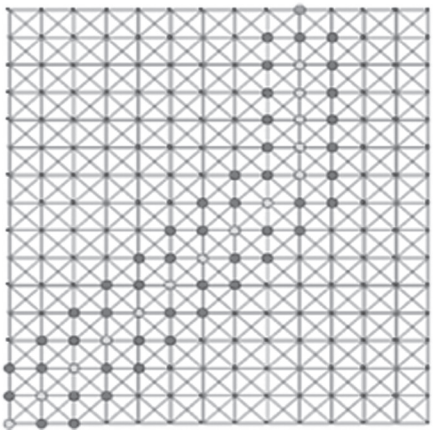

Рис. 4. Зображення графа алгоритму пошуку *D**

Алгоритм пошуку *D** та його варіанти широко використовувалися для мобільного робота та автономного транспортного засобу навігації. Сучасні системи зазвичай засновано на полегшеному, а не оригінальному або сфокусованому *D**.

Розглянуті алгоритми не є вичерпним списком усіх алгоритмів пошуку шляху, наприклад: алгоритми планування шляху з довільними кутами та інші. Усі алгоритми пошуку вирішують одні й ті самі проблеми і засновані на припущеннях планування шляху, зокрема й планування з припущенням про вільний простір, коли робот має переміщуватися до заданих координат цілі на невідомій місцевості. Він робить припущення про невідому частину місцевості, наприклад, що в ній немає завад, і знаходить найкоротший шлях від її поточних координат до координат мети за цих припущень. Коли він спостерігає нову інформацію на карті, наприклад раніше невідомі завади, він додає цю інформацію на свою карту і, за потреби, переплановує найкоротший шлях від поточних координат до заданих координат мети. Він повторює процес доти, доки досягне координат мети чи визначить, що координати мети неможливо досягнути. У разі перетину невідомої місцевості можуть часто виявлятися нові завади, тому це перепланування має бути швидким. Інкрементальні (евристичні) алгоритми пошуку прискорюють пошук

послідовностей схожих пошукових задач, використовуючи досвід вирішення попередніх проблем для прискорення пошуку поточної.

2. ***Метод поведінкових алгоритмів.*** Поведінкові алгоритми здебільшого використовуються для того, щоб надати ігровим персонажам, керованим комп'ютером, певну послідовність дій, які потрібно застосувати до певної ситуації. Наприклад, ігровий персонаж може змінювати ставлення до головного героя після певних ігрових подій. До прикладу, дружній ігровий персонаж може стати «ворогом», якщо гравець почне його атакувати. У багатьох шутерах під час проходження гри користувач може найняти собі підмогу, яка буде йти за ним. У разі боротьби з ворогами гравець може кинути гранату і спостерігати відповідну реакцію ворога, який намагатиметься знайти собі укриття. З усіх цих ситуацій випливає, що стан неігрових персонажів змінюється з одного на другий. При цьому виконуються певні умови, за яких ці переходи між станами/діями гравця відбуваються. Для опису всіх цих дій в іграх використовують теорію скінчених автоматів і поведінкові дерева. Розглянемо кожен із них.

2.1. *Поведінкові дерева.* Поведінковим деревом можна вважати структуру даних, схожу на однонапрямлений граф. Це дерево, яке містить у собі вузли, що описують певну дію, умову, і ребра, які показують напрямок подальших дій. Поведінкове дерево містить такі типи вузлів: *кореневий вузол* — вузол з якого починається поведінкове дерево і від якого починає надходити сигнал до інших вузлів. Містить тільки один дочірній елемент, яким зазвичай є composite вузол. *Composite вузлами* є вузли selector і sequence; *Task* (задача) — вузол, який містить реалізацію певної, переважно простої дії, наприклад, перейти від точки *A* до *B*, перезарядити зброю та ін. Кожен етап виконання завдання можна описати трьома станами: *running* (у виконанні), *success* (задачу виконано успішно), *failure* (задачу провалено). Виконання всіх дій у поведінковому дереві закінчується на цих вузлах; *Decorator* — містить певну умову, при якій вузол задачі має виконуватися, наприклад, завдання вести вогонь, якщо є набої; *Selector* («відбирач») — вузол, який містить нащадки, кількість яких більша чи така, що дорівнює одиниці, і виконує певну задачу, яку можна успішно виконати, наприклад, можна описати три стани гравця: сон, патрулювання, обід. Так, якщо гравець не хоче спати, він патрулює, якщо час для обідньої перерви, то він обідає. Тобто в один момент часу виконується тільки одна дія; *Sequence* («послідовність») — також містить нащадки, кількість яких більша чи дорівнює одиниці, і на відміну від «відбирача» виконує всі дії, описані в дочірніх вузлах, допоки вони всі виконуються успішно, в іншому разі всі наступні дії після невдалої виконуватися не будуть. Для унаочнення можна навести класичний приклад із приготуванням яєшні. Так, якщо етапу розбиття яйця в сковорідку не буде, то всі подальші операції буде відмінено.

Такий підхід має великі переваги стосовно інших підходів для прийняття рішення. Адже дерево розв'язків є досить таки зрозумілим, і при першому погляді на його структуру можна дізнатися, що саме становить даний ігровий об'єкт. Одним із прикладів застосування поведінкового дерева є гра Halo 2, де поведінку «ворогів» було описано цим способом.

2.2. *Скінченний автомат* є ще одним підходом реалізації поведінки ігрових персонажів. Спосіб відомий досить давно і був застосований у популярній грі pacman. Завдання гравця — «з'їдати» бонуси і не траплятися ворогам на шляху. У даній грі «вороги» були реалізовані за допомогою скінченного автомата. Так, наприклад, персонаж «привид» може переходити зі стану атакуючого у стан вразливого. Цей перехід зумовлюється підбиранням гравцем бонуса. Після того, як «привид» перейшов у стан вразливості, користувач може завдати йому шкоди, у такий спосіб повернувши його до попереднього стану, або через певний час, який відведено на перебування в цьому стані, «привид» знову повертається в попередній стан. Тобто машину станів у цьому разі можна зобразити графом, який містить дві вершини і два ребра, які їх з'єднують, а також містять умови, за яких ці переходи між станами відбуваються. Отже, загалом можна сказати, що машина станів є певним графом станів і переходів із певними умовами.

Використання машини станів у сучасних іграх не є таким популярним, як застосування дерева розв'язків. Машина станів чудово підійде для персонажів, котрі не мають надто складної логіки з багатьма станами. В іншому разі машина станів може перетворитися на дуже складну структуру з багатьма залежностями, для розуміння якої потрібен певний час. У таких ситуаціях краще застосовувати дерево розв'язків, що дасть більш зрозумілу картину.

Ще одним прикладом застосування машини станів є реалізація «охоронця», якого можна описати трьома станами: спостереження, переслідування, атака. Коли гравець поза зоною видимості «охоронця», він перебуває в стані спостереження. За умови, коли гравець опиняється в полі зору «охоронця», він переходить зі стану спостереження в стан переслідування. Коли «охоронець» достатньо близько підійшов до гравця, він змінює стан на атакуючий. За умови, що гравець покинув зону видимості, «охоронець» знову переходить у режим спостереження.

3. ***Метод прийняття рішень Мінімах.*** У багатьох іграх часто потрібно оцінити певні дії гравця, які він може зробити в майбутньому, і згідно з цим оцінюванням прийняти певне рішення. Наприклад, у процесі гри в шахи комп'ютеру потрібно оцінити можливі дії гравця і зіграти йому на випередження. Для цього використовують алгоритм minimax. Алгоритм по суті є перебиранням усіх можливих варіантів і вибором найбільш сприятливого. Наприклад, розглянемо гру в хрестики нулики, де AI-гравець грає хрестиками, а людина грає нуликами. Для того, щоб реалізувати даний алгоритм, потрібна інформація про клітинки, які ще не використані, і про гравця, який зараз робить свій хід. Для реалізації використовуватиметься рекурсивна функція, яка прийматиме на вхід два аргументи: гравця, який зараз ходить, і стан грального поля під час ходу даного гравця. Функція буде повертати три числа: (–1) — якщо нулики перемагають; (0) — якщо нічия; (+1) — якщо перемагають хрестики. На кожній ітерації ми перебираємо пусті клітинки і підставляємо в них значення хрестика або нулика, змінюємо гравця та передаємо інформацію про гравця і про гральне поле наступному рекурсивному спуску. Після перебирання варіантів вибираємо той, який є виграшним для гравця, котрий грає за даної ітерації, тобто мінімальне значення для нуликів і максимальне значення для хрестиків, водночас запам'ятовуючи цей хід. На цьому етапі шукаємо найбільше мінімальне значення для нуликів, яке заразом буде виграшним значенням для хрестиків. Так, наприклад, якщо на якісь ітерації з'ясується, що мінімальне значення для нуликів буде (+1), то для хрестиків це буде виграшна комбінація і один із найкращих ходів. Тобто сама назва алгоритму говорить, що максимізуємо значення з мінімумів.

Для оптимізації алгоритму minimax використовують альфа-бета відсікання. Припустимо, на якісь ітерації алгоритму minimax виявляється, що один із можливих ходів призводить до несприятливого результату для гравця, також існують інші ходи суперника, які можуть принести гравцю перемогу. Ці всі інші ходи суперника можна відкинути, адже він просто не вибере їх.

У процесі використання цього алгоритму бувають ситуації, коли кількість варіантів майбутніх ігрових подій може сягати великого числа. Наприклад, якщо використовуватимемо даний алгоритм на етапі написання гри в шашки, то під час застосування цього алгоритму при першому ході гравця комп'ютеру, щоб порахувати всі можливі варіанти ходів і вибрати сприятливий, потрібно багато часу. Тому в цьому разі використовують певну глибину проходу варіантів, яка становить певну кількість ходів, і серед цих варіантів вибирається найбільш сприятливий.

4. ***Метод використання машинного навчання в розробленні ігрового штучного інтелекту.*** Методи машинного навчання зазвичай використовуються в тих випадках, коли складно сформулювати поведінку певного об'єкта, застосовуючи лише певний алгоритм. Наприклад, підлаштування дій ігрового персонажа під дії гравця, які заздалегідь передбачити неможливо, оцінювання дій гравця, а отже, подальші його дії. Сам принцип машинного навчання побудовано на нейронних мережах, яким на вхід подаються певні значення, а на виході дістаємо певний результат. Саме правильна побудова даної нейронної мережі, яка буде розв'язувати певну задачу, і є основним принципом. Для навчання застосовується деякий ігровий агент, який збирає певні дані з середовища, аналізує і відповідно до аналізу ухвалює певне рішення. Зазвичай для навчання агентів використовується підхід машинного навчання з підкріпленням (*Reinforcement Learning*).

Загалом можна виокремити три типи підходів до машинного навчання.

4.1. *Навчання з учителем* — це тип навчання, коли існують певні вхідні дані для оброблення і певні вихідні дані, які сподіваються дістати в результаті. Під час кожної ітерації здобуті і бажані результати порівнюються і залежно від порівняння нейронну мережу переналаштовують, використовуючи градієнтний або інші математичні аналізи. Наприклад, можна навчити нейронну мережу розпізнавати рукописні цифри, подаючи на вхід числове представлення кольору пікселів, і отримати на виході числа від 0 до 9. Навчивши нейрону мережу розпізнавати одне число, вона може за встановленими закономірностями виявляти інші.

4.2. *Навчання без учителя* — тип навчання, за якого задається певна множина вхідних даних, і потрібно знайти певні взаємозв'язки між ними. Часто використовується в задачах кластеризації.

4.3. *Навчання з підкріпленням* — являє собою розв'язання задачі максимізації винагороди, яку отримує агент у певному середовищі, тобто для того, щоб агент зрозумів чи в правильному напрямку виконуються дії. За кожне правильно виконане завдання йому присвоюється певна числова винагорода, в іншому разі — певне покарання. Наприклад, можна розглянути побудову нейронної мережі, яка навчає агента переміщуватися до вказаної цілі, водночас оминаючи завади. Для розв'язання цієї задачі агенту потрібно знати, де він наразі перебуває і де знаходиться зазначена ціль. Дані будуть унаочнено в тривимірній системі координат і подано на вхід. На виході отримуватиметься вектор переміщення. Відповідно, коли навчання буде розпочато, агент у разі влучання в заваду буде отримувати певне числове покарання (число з від'ємним знаком), а в разі досягнення

цілі діставатиме винагороду. Після кількох таких інтеграції отримаємо робочу нейронну мережу ігрового персонажа, яку потім можна буде використовувати в подальшому розробленні гри. У сучасних ігрових рушіях система ігрового агента вже є реалізованою і готовою для навчання. Наприклад, ігровий рушій Unity використовує свій пакет Unity Machine Learning Agents (ML-Agents), який реалізує даний функціонал. В ігровому рушії Unreal Engine застосовується плагін MindMaker Machine Learning AI.

Що ж до адаптивних підходів, то це окремий напрямок машинного навчання персонажів, який потребує особливого розгляду. Він застосовує машинне навчання для створення автоматично згенерованих карт гри, які підлаштовуватимуться під інтереси гравця. Сьогодні вже застосовуються схожі принципи в процесі створення небачених раніше людських облич, автогенерації панорам ландшафтів, вулиць, краєвидів. Іншим прикладом адаптивних підходів є технологія розпізнавання людського голосу, яку також побудовано на технології машинного навчання. Її можна застосовувати для голосових команд керування ігровим персонажем. Технологію розпізнавання обличчя користувача можна використовувати як для автентифікації під час входу в онлайн-гру, так і для розпізнавання емоцій користувача і, відповідно,подальших ігрових подій.

ВИСНОВКИ

Проведений у статті аналіз способів побудови гри та розроблення алгоритмів оцінювання поточного рівня вмінь гравця дав змогу дійти таких висновків:

1. Ігровий штучний інтелект є досить поширеним поняттям в ігровій індустрії. В узагальненому вигляді його призначено для створення такої поведінки ігрового персонажа, яка дає гравцю враження ніби він має певний інтелект.

2. Ігровий штучний інтелект використовує як алгоритмічний підхід, так і адаптивний. Інколи ці підходи застосовують разом, тобто можна сконструювати дерево розв'язків, яке використовуватиме нейронну мережу. Наприклад, умова, за якої ігровий персонаж буде «нападати» на гравця, може бути прописаною в дереві розв'язків, а сама послідовність дій вирішуватиметься за допомогою нейронної мережі.

3. Проте використання машинного навчання має певний недолік порівняно з наперед заданим алгоритмом дій. Для того, щоб тренувати ігрового персонажа виконувати певну дію, потрібно набагато більше часу, ніж просто знайти якусь закономірність і прописати її в алгоритмі. Наприклад, щоб тренувати ігрову модель знаходити шлях від однієї точки до іншої, потрібно набагато більше часу, ніж застосувати один із відомих алгоритмів. Також використання машинного навчання безпосередньо в ігровому процесі є досить затратним ресурсом, тому часто спочатку ігрову модель навчають, а потім використовують в ігровому процесі.

4. Машинне навчання буде доцільним тоді, коли важко або й неможливо описати певну закономірність алгоритмом. Зазвичай в іграх для навчання ігрових персонажів використовується навчання з підкріпленням, схоже на дослідження світу, яке людина робить у перші роки свого життя.

5. Машинне навчання має досить великий потенціал у майбутньому розробленні ігор. Цю систему можна використати для автоматичного генерування карт, рівнів, оцінювання емоційного стану гравця, голосового керування тощо. Усі ці методи формують поняття ігрового штучного інтелекту, який є певною підгалуззю великої галузі штучного інтелекту.

Список використаної літератури

1. ***Надмірність*** *інформації [Електронний ресурс]. URL:*

https://uk.wikipedia.org/wiki-Text.

2. ***Харт П., Нільсен Н., Рафаель Б.*** *A* algoritam / Stanford CS Theory - Intorduction to A* / [Електронний ресурс]. URL:*

http://www.alas.matf.bg.ac.rs/~mi15170/seminarski.pdf.

3. ***Zhou Y., Pei Sh.*** *A hybrid co-evolutionary particle swarm optimization algorithm for solving constrained engineering design problems // China Journal of computers. 2010. Vol. 5, No. 6. P. 965–972. [Електронний ресурс]. URL:*

https://www.researchgate.net/publication/283661315_Co-hybridization_of_PSO.

4. ***Alomari O., Othman Z. A.*** *Bees algorithm for feature selection in network anomaly detection // Journal of Applied Sciences Research. 2012. № 3. P. 1748–1756.*

5. ***A* algoritam.*** *Seminarski rad u okviru kursa Tehnicko i naucno pisanje Matematicki fakultet http://www.alas.matf.bg.ac.rs/~mi15170/seminarski.pdf [Електронний ресурс]. URL:*

http://www.alas.matf.bg.ac.rs/~mi15170/seminarski.pdf.

6. ***Rathi B., Jadhav D. V.*** *Network Intrusion Detection Using PSO Based on Adaptive Mutation and Genetic Algorithm // International Journal of Scientific & Engineering Research. 2014. № 8. P. 142–144.*

7. ***Copeland J.*** *A Brief History of Computing (англ.). AlanTuring.net (июнь 2000).*

8. ***Алгоритм*** *пошуку D* [Електронний ресурс]. URL:*

https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%

BC_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0_D.*

*9. **What** is Artificial Intelligence? [Електронний ресурс]. URL:*

https://builtin.com/artificial-intelligence.

*10. **Turing A. M.** Computing Machinery and Intelligence [Електронний ресурс]. URL:*

https://www.csee.umbc.edu/courses/471/papers/turing.pdf.

*11. **Pathfinding** Demystified (Part I): Generic Search Algorithm [Електронний ресурс]. URL:*

https://www.gabrielgambetta.com/generic-search.html.

*12. **The University** of Western Australia, Practical Path Finding [Електронний ресурс]. URL:*

https://teaching.csse.uwa.edu.au/units/CITS4242/17-paths.pdf.

*13. **Stout B.** Smart Move: Intelligent Path-Finding [Електронний ресурс]. URL:*

https://www.gamasutra.com/view/feature/3317/smart_move_intelligent?print=.

*14. **Buckland M.** Programming Game AI by Example [Електронний ресурс]. URL:*

https://app.box.com/s/y4gvcrknxfmkefxbhlxt9ox5pxotks68.

*15. **Kylotan.** The Total Beginner's Guide to Game AI [Електронний ресурс]. URL:*

https://www.gamedev.net/articles/programming/artificialintelligence/thetotalbeginners-guide-to-game-ai-r4942.

*16. **Lague S.** A* Pathfinding (E01: algorithm explanation) [Електронний ресурс]. URL:*

https://www.youtube.com/watch?v=-L-WgKMFuhE.

*17. **Simpson C.** Behaviour trees for AI: How they work [Електронний ресурс]. URL:*

https://www.gamedeveloper.com/programming/behavior-trees-for-ai-how-they-work.

*18. **Introduction** to Minimax Algorithm [Електронний ресурс]. URL:*

https://www.baeldung.com/java-minimax-algorithm.

*19. **Akshay L Aradhya.** Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning) [Електронний ресурс]. URL:*

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alphabetapruning.

*20. **Ігровий** штучний інтелект [Електронний ресурс]. URL:*

https://uk.wikipedia.org/wiki/%D0%86%D0%B3%D1%80%D0%BE%D0%B2%D0%B8%D0%B9_%D1%88%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82.

Yu. I. Katkov, O. Yu. Ilyin, I. V. Reznichenko, V. O. Vyshnivskyi

DEVELOPMENT OF COMPUTER GAMES USING ARTIFICIAL GAME INTELLIGENCE TECHNOLOGIES

The article is devoted to the topical issue of finding new effective methods and improving existing common ways of developing computer games using artificial intelligence technologies. The main focus is on finding and improving the efficiency of tools for building computer games through the introduction of modern technologies. The article presents the following statement of the problem: the use of modern research in the field of artificial intelligence — one of the surest ways to solve complex problems in the gaming industry. It is known that the requirements of modern gamers for computer games are quite small, so developers need to focus on such innovations that can ensure the introduction of new gaming artificial intelligence technologies. To do this, it is necessary to analyze ways to build the game, to develop algorithms for assessing the current level of skills of the player to offer him during the game situations that give more interest to the gameplay. Solving the problem allows you to reduce computational complexity and improve the quality of decisions made in real time. To solve this problem, the article describes: methods of using gaming artificial intelligence technologies, considers the logic of this game by different methods, reveals whether gaming artificial intelligence is a subsector of a large field of artificial intelligence, describes machine learning methods and explains why machine learning is not always appropriate. Used in game development, describes the main directions of development of artificial intelligence in the further development of computer games, as well as the principle of building games using the algorithm for assessing the skills of the player and the game engine Unity.

Keywords: GameDev; Game artificial intelligence.