УДК 621.391

ДОРОГИЙ Я. Ю.,
ШПИТА Г. В.

МЕТОДОЛОГІЯ ІМПЛЕМЕНТАЦІЇ ІГРОВИХ АДАПТИВНИХ МЕХАНІК

Дана стаття пропонує набір методів імплементації адаптивних механізмів у ігровому середовищі, які дозволяють ігровому штучному інтелекту адекватно реагувати на неочікувані дії гравця та підлаштовуватися під його стиль гри з метою підвищення якості ігрового досвіду. Наведені методи розглянуті з точки зору реалізації інтелектуальних засобів та правил геймдизайну. Стаття проводить ретельний аналіз існуючих рішень у даній області, відповідає на питання доцільності застосування адаптивних механізмів та пропонує вирішення пов'язаних з використанням даних технологій проблем, які часто стають перешкодами для розробників штучного інтелекту у відеоіграх.

This article offers a set of methods for implementing adaptive mechanisms in the non-linear gaming environment, which allow gaming artificial intelligence to respond properly to unexpected player's actions and adapt to his playing style to improve the quality of the game experience. These methods are considered from the perspective of intellectual assets and rules of game design. Article conducts a thorough analysis of existing solutions in the field, answers the question of the usefulness of adaptive mechanisms and offers solutions of problems which are associated with the use of presented technologies and often become an obstacle for videogame AI developers.

Ключові слова: ігрова адаптивна механіка, геймдизайн, нейронна мережа, генетичні алгоритми.

Введення в проблему

Лінійність у іграх – це міра того, наскільки гравець є прив'язаним до певного порядку дій та обов'язкових ступенів, які він має здолати для проходження гри. Ігри називають нелінійними тоді, коли гравець має сам встановлювати для себе мету, проходити завдання різними шляхами у довільному порядку. У повністю нелінійній грі не можна сказати коли вона є остаточно пройденою, якщо ж така гра може бути завершена, вона зветься «напівлінійною» (хоча зазвичай цей термін не застосовують, а замість нього кажуть просто «нелінійна гра»).

При розробці нелінійної гри одна із проблем полягає в тому, що така гра передбачає величезну кількість можливих ігрових ситуацій і геймдизайнер має передбачити якомога більше з них. Чим складніше гра, тим важче продумати заздалегідь як ігровий ШІ має реагувати на дії гравця при різних станах ігрового середовища, або як мають розвиватися події коли немає певного сценарію. Це є вірним і для тих ігор, які пропонують велику кількість засобів для проходження та варіантів подій.

Як правило, ця проблема вирішується розглядом максимально можливої кількості ігрових ситуацій, що можуть виникнути у процесі геймплею, написанням окремих скриптів для кожного з сценаріїв та створенням штучного інтелекту, що базується на великій кількості умовних переходів. Усе це є доволі складним процесом, та потребує громіздкого підходу до розробки різноманітних можливих сценаріїв.

До того ж, більшість розробників бажає охопити якнайбільшу аудиторію, що означає необхідність продумати це більше варіацій ігрових елементів.

Мета роботи

Метою даної роботи є запропонування комплексу адаптивних механік, застосування яких дозволить розробити щільну ігрову систему, яка здатна навчатися у процесі гри, самостійно знаходити адекватні рішення у непередбачених розробником ситуаціях та підлаштовуватися під навички та стиль гри окремих споживачів.

Особливості використання адаптивних механік

Якщо розглянути [1-3], то, в цілому, можна зробити висновок, що люди грають у ігри заради почуття азарту та сюрпризів, які пропонує гра. При цьому, інтерес гравця підтримується доті,

доки гра продовжує підносити сюрпризи. Це означає, що створення непередбачуваного штучного інтелекту (далі ШІ), який у аналогічних ситуаціях може діяти по різному, підлаштовуючись під обставини, є одним із способів, якими розробник може підвищити реіграбельність продукту.

Іншою функцією, яку здатен виконувати адаптивний ШІ, є корекція ігрового процесу згідно до навичок гравця для досягнення оптимального рівню складності. Складність гри є надзвичайно важливим фактором, що визначає наскільки задовільним виявиться ігровий процес. Як висока, так і низька складність можуть призвести до роздратування гравця і негативного досвіду, при чому «правильний» її рівень є сугубо особистим питанням [4]. Зазвичай з цією метою більшість ігор на початку пропонують вибір рівню складності, але, насправді, такий підхід доволі часто виявляється безплідним [5].

Адаптивні механіки дозволяють зробити ШІ більш схожим на живого супротивника. Керуючись [1] можна відмітити, що такий підхід призводить до зниження вродженої складності, але підвищує набуту. Авжеж, для різних типів гравців це може виявитися як позитивною, так і негативною стороною, тому перед застосуванням адаптивного ШІ слід проаналізувати тип запланованої гри та цільової аудиторії щоб переконатися у доцільності рішення.

Тим не менш, не дивлячись на переваги застосування адаптивного ШІ, у сучасній ігровій індустрії можна помітити тотальне панування простих ШІ на основі дерев рішень. Така тенденція пов’язана з наступними причинами:

- Звичайний ШІ легше тестувати через його передбачуваність.
- Неочевидність способів демонстрації гравцю природи ШІ з яким він грає, що може призвести до того, що деякі люди не помітять різниці.
- Небезпека потенціального росту собівартості проекту при дослідженні та застосуванні адаптивних механік.
- Небезпека «перенавчання», при якому конкретному гравцю може стати набагато легше перемогти адаптивний ШІ на чужому комп’ютері, аніж на своєму.

Для великих ігрових компаній будь-які експерименти можуть бути доволі небезпечними, тому що вони вкладають багато ресурсів у розробку своїх продуктів. Тим не менш, корисний ефект, який пропонують адаптивні системи, значно перевищує проблеми, що пов’язані з його реалізацією у грі. Наприклад, проблема тестування значно спрощується якщо комбінувати адаптивні механіки з класичними рішеннями у області ігрових ШІ (що також зменшує вартість розробки). Супроводження набуття досвіду ШІ різноманітними реакціями користувацького інтерфейсу та наративними елементами надасть гравцю уявлення про природу ШІ, з яким він має справу. Якщо здатність системи навчатися природно вписується у ігровий дизайн, гравцю буде цікавіше чекати нових сюрпризів, які можуть підносити адаптивні системи. Небезпека «перенавчання» не є такою страшною, якщо згадати що грати на чужій платформі може бути нецікаво і з других причин: доступ до інших ігрових предметів, інший клас та рівень персонажу, інший прогрес у кампанії і.т.д. Все це є результатами прояву особистості гравця у процесі гри, так само як і результати навчання ШІ.

Аналіз існуючих рішень

Для того, щоб визначити як правильно підійти до розробки адаптивного штучного інтелекту для відеогри, розглянемо декілька прикладів у відповідних областях. Варто зазначити, що частина прикладів відноситься до відеоігор, що розроблялися як масовий продукт, а частина – до наукових робіт, метою яких було дослідження властивостей ШІ. Кожна з груп має власні особливості у підході до розробки алгоритму роботи ШІ, тому слід розділити їх та проаналізувати окремо.

Ігрові рішення:

Creatures[7] - це серія комп'ютерних ігор з симуляцією штучного життя. Ігровий процес полягає у вирощуванні інопланетних створінь (Норнів) та їх навчанні. Гравцю доступний спеціальний інтерфейс, який дозволяє навчати Норнів мові та інструктувати їх щодо бажаної поведінки, при цьому комп’ютерні створіння

самі вирішують чи підкорятися наказам чи ні. Замість використання скриптового підходу, звичного для відеоігор, розробники Creatures керувалися розробками у області біологічного і неврологічного моделювання, що призвело до незвичайного результату. Creatures є однією з небагатьох ігор, які вдало змогла застосувати нейромережі і адаптивні алгоритми як основне джерело ігрової механіки.

Forza Motorsport 5[8] – відеогра про автомобільні перегони у жанрі «автосимулятор». Зазвичай, автомобільні боти у іграх керуються спеціальними скриптовими алгоритмами, які надають їм більш-менш правдоподібну поведінку: вони здатні дотримуватися дороги, вписуватися у складні повороти і навіть іноді атакувати гравця для отримання переваги у гонці. Тим не менш, все ці можливості є заздалегідь прописаними у коді гри і перевершити власний рівень такі боти не здатні. Forza Motorsport 5, навідміну, застосовує систему «Drivatar» на основі нейромереж для навчання ботів. Протягом заїзду система збирає інформацію про навички гравця, після чого завантажує дані на сервери, де алгоритми Drivatar невпинно шукають характерні шаблони у масивах даних. Це дозволяє комп'ютеру постійно вивчати нові маневри та поводити себе майже як справжній водій під час гри, демонструючи хитрі прийоми, якім він навчився у живих гравців.

Left4Dead[9] – багатокористувацька гра у жанрах «FPS» та «survival horror». Однією з особливостей гри є додатковий ШІ, який тут називається «Режисер» (AI Director). Режисер здатний впливати на ігровий процес, слідкуючи за статистикою успішності гравців та підлаштовуючись під їх рівень, змінюючи такі параметри гри, як кількість звичайних та спеціальних ворогів, частоту та місце їх появи.

Використання цієї системи робить проходження одних й тих самих рівнів несхожими, що позитивно впливає на реіграбельність.

Dynamic game difficulty balancing[10] або метод динамічного балансування складності – це процес автоматичної зміни параметрів, сценаріїв та поведінки у відеогрі в режимі реального часу, заснований на навичках гравця, з метою уникнення утворення відчуття нудьги у гравця (якщо гра дуже легка) або розчарування у своїх силах (якщо вона занадто складна). Знімаючи деякі обмеження з комп'ютерних противників, гра може дозволити їм шахраювати, наприклад, ШІ-супротивникам у перегонах може бути надана необмежена швидкість, щоб постійно залишатися поруч з гравцем, підтримуючи так рівень напруги. Мета динамічного балансування складності - утримання користувача зацікавленим від початку до кінця, забезпечуючи хороший рівень виклику. Динамічне балансування складності є однією з функцій системи «Режисер», про яку мова йшла вище.

Академічні рішення:

Giraffe[11] – комп'ютерна програма для гри в шахи, розроблена за допомогою застосування нейронних мереж. У [11] автор пише, що після тренування протягом 72 годин програма обирала найкращий можливий хід в 46% випадків, а один з трьох найкращих ходів - в 70% випадків. Це доволі непоганий результат навіть для звичайних шахових програм. Основною особливістю Giraffe є те, що її ніхто не налаштовував, вона сама вчилася грати. До того ж, замість звичайного перебору гілок з обмеженням по глибині, програма використовує «ймовірнісний» підхід. Вона більш глибоко опрацьовує ті гілки, для яких більша ймовірність довгого продовження, що є доволі практичним рішенням, оскільки супротивники зазвичай намагаються зробити найкращий можливий хід.

AlpgaGo[12] – це комп'ютерна програма для гри в ґо, що розроблена компанією Google DeepMind. У жовтні 2015 року вона стала першою комп'ютерною програмою, що в грі на рівних на класичній дошці 19x19 завдала поразки людині – професійному гравцю в го. Скептики стверджували, що програми в го або ніколи не переможуть, або отримають перемогу дуже нескоро. Го складніше шахів в 10^100 разів - саме в стільки разів більше можливих позицій каменів на стандартній дошці 19 × 19, ніж в шахах. Це ускладнює використання традиційних комп'ютерних методів, наприклад методу повного перебору. Згідно до [12], у AlphaGo одна з нейронних мереж (value network) використовується для оцінки позицій в термінальних вузлах дерева пошуку, інша (policy network) призначена для розумного відбору ходів-кандидатів. Також використовується

невелике дерево перебору Монте-Карло. У кожному вузлі розглядається 1-2 ходи, середній коефіцієнт розгалуження - менше двійки. Завдяки цьому стало можливим порахувати осмислене дерево з дуже обмеженим числом варіантів, і за рахунок цього знайти дійсно найсильніший хід в позиції.

Тепер розглянемо представлені відомі рішення з точки зору потенціалу ігрових адаптивних механік.

Creatures	Гра показала, що нейронні мережі цілком можуть становити основу ігрової механіки, при цьому це зовсім не обов'язково має призводити до значних затрат комп'ютерних ресурсів. Дивлячись на ігрову механіку, нескладно помітити, що вона цілком відштовхується від особливостей місцевого штучного інтелекту. Його здатність адаптуватися стоїть у центрі ігрового процесу та визначає характер усіх ігрових елементів, створюючи щільну та цікаву для дослідження гравцем систему. Якби у Creatures не застосовувались нейромережі, гра сприймалася б зовсім інакше.
Forza Motorsport 5	Ця гра є чудовим прикладом застосування ШІ на основі нейромереж, як лише однієї з ігрових особливостей, а не центральної теми гри. Forza Motorsport була б цікавою навіть і без «Drivatar», але ця система допомогла зробити перегони більш напруженими. Основна проблема полягає в тому, що навчання системи відбувається не на локальній платформі, а у хмарі, що потребує постійне з'єднання з мережею.
Left4Dead	Ця гра являється прикладом того, як сильно адаптивні механіки підвищують реіграбельність. Гравці Left4dead проходять одні й ті самі рівні десятки разів і при цьому завжди щось змінюється. Ігровий режисер аналізує навички, статус та позиції гравців та управляє не лише інтелектом супротивників, а й погодними умовами, музикою, візуальними ефектами та навіть діалогами між героями гри. Це наштовхує на думку, що окрім ШІ окремих «юнітів», адаптуватися може навіть саме ігрове середовище, змінюючи свої параметри на основі досвіду взаємодії с гравцем.
Dynamic game difficulty balancing	Метод динамічного балансування є найбільш поширеною механікою адаптивності у відеоіграх. Він пропонує чудове рішення для підлаштовування ігрових параметрів під особисті навички гравця. Зазвичай, він заснований на зборі певної статистики, значення елементів якої пізніше впливає на параметри, що відповідають за складність. Гра Resident Evil 4 автоматично зменшувала кількість та силу супротивників якщо гравець програвав багато разів підряд
Giraffe	Успіх Giraffe є чудовим показником розвитку систем штучного інтелекту, але в нього є не так багато переваг над звичайними шаховими програмами з точки зору реалізації масового продукту. Основна з них – це простота використання застосованої технології для навчання комп'ютерних супротивників, що спрощує процес розробки: Giraffe не були потрібні навчальні дані, вона вчилася грати сама. Тим не менш, лише для професіональних шахістів буде відчутна різниця між грою з Giraffe та будь-якою складною шаховою програмою. Що ще важливіше, навіть їм буде складно відчути чи продовжує програма вчитися протягом гри з ними.
AlphaGo	Для AlphaGo Основна відмінність від ситуації з Giraffe полягає в тому, що через складність Го, протягом довгого часу ігрові програми не могли скласти достатню конкуренцію гравцям високого рівня. Професіоналам з Го просто було нецікаво грати з комп'ютером. Враховуючи ці факти, можна зробити висновок: деякі ігри підходять для використання адаптивних механізмів краще аніж інші і складність системи грає в цьому не останню роль.

Переглянувши існуючі рішення та попередні висновки, просумуємо отриманий досвід у вигляді набору тезисів:

1) Ігрова адаптивність проявляється у зміні алгоритму роботи ШІ (який в свою чергу визначає стан і поведінку об'єктів ігрового середовища) у відповідності до характеру дій гравця.

2) Адаптивні механізми є надзвичайно потужним засобом збільшення реіграбельної якості продукту та інструментом індивідуальної оптимізації.

3) Вхідними параметрами для навчання ШІ можуть бути дії гравця, вибори які він робить, засоби які він використовує, його коефіцієнт успішності, стан ігрової системи на момент певних дій гравця. Збирати цю інформацію можна безпосередньо під час активної фази гри, або у спеціально розроблені окремі періоди.

4) Адаптуватися під індивідуальні особливості гравця може майже все: параметри складності, правила гри, поведінка ШІ, наративні компоненти та вигляд ігрового середовища.

5) Використання адаптивних технік може призвести до мутацій ігрової механіки та зовсім іншого сприйняття гри.

6) Адаптивні механізми не потрібно застосовувати тоді, коли є простіші способи досягти аналогічної мети. Особливо це стосується тих випадків, коли не існує чіткого способу продемонструвати гравцю здатність системи до самонавчання.

7) Складі ігри з великою кількістю параметрів краще підходять для застосування алгоритмів навчання, адже так гравцю буде легше помітити прогрес інтелектуалізації ШІ.

8) Навіть якщо процес навчання ШІ виявиться занадто ресурсозатратним, цю функцію можна перекласти на віддалені сервери, щоб запобігти перенавантаженню платформи гравця.

Методи імплементації адаптивних механізмів

Маючи на увазі попередні висновки, визначимо способи реалізації ігрової адаптивності.

Нейронні мережі

Одна з основних проблем застосування нейромереж у ігровому процесі пов'язана з тим, що для отримання достатньо малої помилки їм може знадобитися задано багато тренувальних даних. Оскільки такі дані формуються у процесі гри, існує небезпека, що гравцю набридне дивитися на помилки ШІ раніше, аніж нейронна мережа почне адекватно працювати. До того ж, важко говорити про час, який може знадобитися для навчання протягом однієї епохи, оскільки це значення може сильно варіюватися в залежності від кількості шарів, обраного алгоритму, розміру виборки та застосованого процесору. Простіше використовувати нейромережі для навчання ШІ на стадії розробки для знаходження оптимальної поведінки, а не у фінальному продукті.

Якщо, все ж таки, деякі елементи ігрового процесу потребують динамічного навчання під час гри, то слід обрати один з двох варіантів: ставити процес навчання нейромереж у центр ігрової механіки для того, щоб забезпечити їх достатньою кількістю даних і часом навчання, або ж використовувати дані отримані від великої кількості гравців для корекції роботи ШІ. Перший варіант, наприклад, може уявляти з себе гру, метою у якій є навчання певних агентів, Гравець не буде спантеличений тим, що у певні моменти ШІ може приймати дивні і нелогічні рішення, оскільки розумітиме, що це є невід'ємною частиною процесу навчання. Другий варіант передбачає, що платформи гравців будуть певним чином обмінюватися із серверами гри даними, необхідними для навчання нейронної мережі. Це дозволить перекласти навантаження по інтелектуалізації на спеціалізовані комп'ютери, залишаючи ігровим платформам лише збір статистики. Як будувати на цих принципах ігрові елементи вже залежить від розробника. Наприклад, гравці можуть навчати своїх ботів для онлайн-змагань між ними, або навчати комп'ютерних партнерів для отримання бажаних союзників у грі,

Генетичні алгоритми

Використання таких алгоритмів є чудовим способом навчання нейронної мережі, але для даної задачі вони можуть бути використані і самі по собі. Генетичні алгоритми використовують для рішення оптимізаційних задач, застосовуючи механізми аналогічні природному відбору, такі як: наслідування, випадкові мутації, відбір і кросинговер. Такий алгоритм добре підходить для пошуку найкращого рішення у відповідності до деякого «ідеального» результату. У кожному поколінні, що складається варіантів рішень (сукупностей параметрів, що характеризують рішення) будуть обиратися два представники, які

мають найменшу помилку відносно «ідеального» результату, між якими виконуватиметься схрещування, після чого нові рішення замінять найгірші рішення попереднього покоління (якщо вони мають меншу помилку). Авжеж, є багато способів виконувати селекцію та обирати пари для схрещування, такі деталі залишаються на розсуд розробника алгоритму. Процес еволюції продовжуватиметься доки не буде знайдено задовільне рішення, або не пройде певна кількість циклів. Якщо «ідеальне» рішення, що шукається, визначається діями гравця, то адаптивна властивість системи проявиться у знаходженні найкращої сукупності параметрів ігрового середовища для даного гравця.

Наприклад, припустимо що метою є створення ігрового ШІ, який слідкує успіхами гравців під час кооперативного проходження рівнів шутеру і відповідно до цього змінює параметри складності гри. Нехай є функція f залежності від середнього часу проходження одного рівню, точності стрільби, кількості «фрагів» та кількості програшів на останньому рівні, частота «виходу з ладу» гравців на останньому рівні (поширена механіка для кооперативних шутерів: якщо один з гравців втратив все здоров'я, його персонаж падає на місці і очікує допомоги напарнику. Якщо всі гравці «впадуть», то вони програють) і.т.д. Нехай значення функції варіюється від деякого f_{min} (якщо гравці досягли планки найгірших результатів, яку встановили розробники) до f_{max} (якщо гравці досягли високого рівню успішності). Функція може бути сумою всіх параметрів, якщо всі вони взаємно відповідні (найменше значення – найгірший результат, найбільше – найкращий результат).

Після кожного пройденого рівню, ШІ вираховує значення f, після чого починає оптимізацію структури наступного рівня за допомогою генетичного алгоритму. Параметрами рішень, що виступають у ролі генотипів є значення змінних, що визначають стан ігрового середовища. Це може бути: кількість «спаунів» зброї, патронів та медпакетів; кількість, сила та уважність ворогів; погодні умови та різні обставини, які можуть ускладнити гру. Для селекції застосовуватиметься деяка цільова функція f2, яка також змінюються від f_{min} до f_{max}, при цьому її мінімальне значення відповідатиме максимально поблажливим умовам, а найбільше – найекстремальнішим. Ідеальним значення для функції f2 буде отримане значення f.

Адаптивна природа такої системи дозволить не лише точно корегувати складність гри відповідно до навичок гравців, але й гарантуватиме високу реіграбельність, адже завдяки випадковості мутацій під час схрещування, навіть при однаковому стилі проходження, значення параметрів фінального рішення можуть відрізнятися. Наприклад, одного разу гравці можуть отримати додаткові медичні пакети, а іншого більше патронів, що змусить обрати іншу тактику і збереже інтерес.

Експертні системи

Такими системами називають програми, які за допомогою бази знань вирішують проблеми які погано піддаються формалізації. Зазвичай, такі системи створюються для вирішення практичних задач у деякій вузькій предметній області. Експертна система має логічний вирішувач, який циклічно оброблює побудовану експертами базу знань і вирішує проблеми що задані у робочій пам'яті. Як правило, база знань складається із правил, які представлені у неформальному вигляді. Часто експертна система у ході роботи може виявляти нові правила та доповнювати свою базу знань. В цьому проявляється її властивість до адаптивності.

Така система добре підійде для організації знань про гравця з метою прийняття оптимальних рішень штучним інтелектом.

Наприклад, метою є створення військової стратегії де гравець виступає командиром батальйону і протистоїть іншим командирам, кожен з яких може мати власну базу знань про гравця, яка формується у ході сутичок. Супротивники мають запам'ятовувати дії гравця при кожному бої для того, щоб не потрапляти двічі у одну й ту саму пастку. Нехай база знань командирів представлена масивами даних, що поповнюється після кожної сутички і складаються з наступних даних: локація бою, квадрант сутички, тривалість, дружні юніти, ворожі юніти, обрана тактика, результат. Спочатку ШІ користуватиметься лише стандартними правилами, що задав розробник, але у ході отримування досвіду перед ключовими рішеннями почне звертатися до бази знань і шукати в ній аналогічні ситуації. Якщо у базі є схожі ситуації (тут ступінь достатньої схожості має обрати сам розробник, наприклад може бути достатньо того що мова йде про ту саму локацію та квадрант бою), то ШІ

перевірить їх результати. Позитивні результати означатимуть що можна знову застосувати тих самих юнітів та тактику, а негативні свідчитимуть про необхідність внести зміни. Для вибору характеру змін можна використати дерево рішень, по якому ШІ пропустить результати боїв. Якщо у середньому втрати ворога були доволі великими, то можна просто відправити на декількох юнітів більше, якщо ж ШІ багато разів зазнав розгрому, то є сенс спробувати змінити тактику або автоматично підняти рівень складності.

Прості евристичні алгоритми

Це алгоритми, що включають практичний метод, які не є гарантовано точними або оптимальними, але достатній для рішення поставленої задачі. Генетичний алгоритм є різновидом евристичних, але тут винесений окремо через особливості підходу.

Евристичні рішення для адаптивних механізмів добре підходять тоді, коли розробник не відчуває необхідності нагромаджувати роботі ШІ зайвими обчисленнями. Припустимо що метою розробника є створення гри з десятьма рівнями складності, але замість того щоб пропонувати гравцю зробити вибір, система сама перемикається між ними в залежності від успішності гравця. В такій ситуації немає сенсу вводити складні алгоритми, які будуть вести багатофакторну статистику успіхів гравця та застосовувати складні аналітичні методи. Більш простим і доволі ефективним рішенням буде замірювати кількість програшів на одиницю часу гри та понижувати складність якщо ця величина зависока, або ж повільно підвищувати, якщо вона дорівнює нулю. Схожі рішення стоять в основі механік динамічного балансування складності. Точність таких засобів можна підвищити, сильніше залучаючи гравця до процесу адаптації. Наприклад, гра у жанрі «horror» може запропонувати гравцю пройти психологічний тест для вибору найстрашніших наративних елементів для даного індивідуума.

Приклад застосування

Для порівняння особливостей ігрового процесу з різними механізмами штучного інтелекту було розроблено дві програми, кожна з яких уявляє з себе гру в «морський бій» проти комп'ютера. Для того щоб прискорити тестування правила були модифіковані: бій йде на полях 7 на 7 клітин, кожна зі сторін має один трьохпалубний корабель, два двопалубних і три однопалубних. Програми не відрізняються візуально, але керуються різними алгоритмами. Перша програма застосовує оптимальний спосіб гри у морський бій[13]. Вона розставляє свої кораблі таким чином, щоб у випадку їх знаходження мінімізувати кількість гарантовано вільних клітин. Програма ставить двопалубні та трьохпалубні кораблі на випадково обрані місця на межах ігрового поля, таким чином зменшуючи інформацію, що отримає супротивник коли корабель буде знайдено. Однопалубні ставляться на випадкові клітини з тих що залишились для того щоб ускладнити їх пошук. Приклад розстановки сил оптимального алгоритму наведено на рисунку 1.

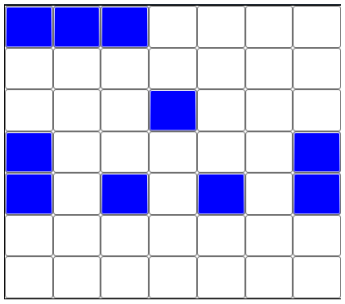

Рис. 1. Розставлення сил програмою з оптимальним алгоритмом

Під час гри оптимальний алгоритм шукає кораблі супротивника, перебираючи варіанти зі зсувом у три клітини. Такий підхід дозволяє знайти найбільший корабель максимум за 16 кроків. Після цього алгоритм перевірить інші клітини з таким самим зсувом щоб забезпечити знаходження двопалубних. Для пошуку однопалубних кораблів оптимального варіанту не існує, тому алгоритм може знайти їх лише випадковим чином. При грі з такою програмою слід очікувати що через декілька раундів гравець помітить закономірність у роботі алгоритму та буде відносно швидко знаходити основні сили супротивника, очікуючи що вони знаходяться на межах поля. Пошук інших кораблів буде зводитися лише до вдачі, що може бути нудно для більшості гравців.

Друга розроблена програма застосовує адаптивні механізми. Для даного експерименту було б нераціонально застосовувати складний алгоритм, наприклад, на основі нейронних мереж, оскільки це поставило б під сумнів доцільність застосування адаптивних механізмів при значному підвищенні

складності програмування такої примітивної гри як морський бій при тому, що її оптимальний алгоритм можна запрограмувати доволі легко. У якості розроблюваного адаптивного алгоритму було обрано відносно простий механізм навчання на основі статистики ігор. Після кожної гри програма зберігає інформацію про результат бою, кількість кораблів що залишилася у переможця, знайдені кораблі гравця, їх положення на полі, порядок та результат вистрілу по кожній клітині. Перед вибором розташування кожного корабля алгоритм звертається до цих даних для того щоб застосовані клітини мали найменшу статистичну частоту перевірки гравцем. Другим критерієм є вибір горизонтальної чи вертикальної орієнтації корабля випадковим чином, але з врахуванням співвідношення середньостатистичного очікування гравця. Наприклад, якщо гравець після знаходження першої клітини корабля, в основному спочатку перевіряє ліву та праву клітини від неї, то кораблі з більшою ймовірністю матимуть вертикальну орієнтацію. Третім критерієм є мінімізація гарантовано вільних клітин на полі. Для стрільби алгоритм другої програми керуються принципом, аналогічним першій, але для кожного з кроків зміщення цілі обстрілу сортує усі можливі варіанти за збільшенням відношення частоти успіху вистрілу по даній клітині до її ваги (вагою в даному випадку є величина, що характеризує як рано в середньому дана клітина виявляється перевіреною). Навіть якщо з точки зору пояснення принципу алгоритм може здатися доволі комплексним, математично він є простим і його можна порівнювати з першим.

Обидві програми були запропоновані двадцяти гравцям, які не повідомлялись про особливості роботи кожної з них. Кожен гравець зіграв 5 раундів проти обох програм, після чого відповів на ряд питань. Результати представлені на таблиці 1.

Таблиця 1 – Результати ігрових сесій

	Неадаптивний ШІ	Адаптивний ШІ
Середня кількість перемог ШІ з 5 раундів	1.2	2.2
Середня тривалість гри	42.28 ходів	47.8 ходів
Загальна кількість неуражених клітин гравців (100 раундів)	384	168
Загальна кількість неуражених клітин ШІ (100 раундів)	84	152

Після усіх раундів кожному гравцю було запропоновано відповісти на декілька питань, результати представлені на таблиці 2.

Таблиця 2 – Результати опитування

Питання	Неадапт. ШІ	Адапт. ШІ	Утрималося
Обрати складніший ШІ	20%	70%	10%
Обрати ШІ з яким цікавіше грати	15%	65%	20%
Обрати ШІ, гра якого більше нагадує людську	5%	70%	25%

Не дивлячись на переваги оптимального алгоритму, другий варіант, завдяки адаптивним механізмам у вигляді збору індивідуальних статистичних даних та їх застосування у роботі ШІ, виявився більш цікавим та складним для більшості гравців, згідно до результатів опитування. На рисунку 2 наведено один з результатів гри проти оптимального алгоритму. Схожий результат мала більшість гравців після 2-3 раундів, коли тактика комп'ютеру ставала очевидною. Гравці починали пошук кораблів з меж поля, щоб підвищити свої шанси на виграш, зосереджуючись на пошуку більших кораблів, а однопалубні знаходили випадково доки ШІ не встиг перевірити усі необхідні поля.

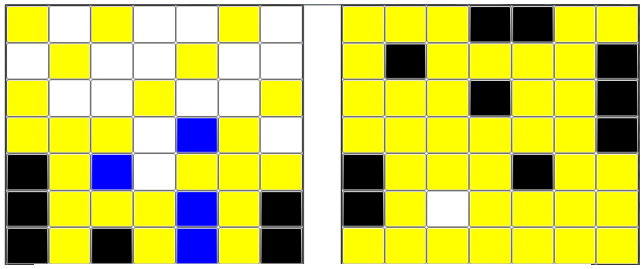

Рис. 2. Приклад результату бою проти оптимального алгоритму (Зліва – поле гравця, справа – ШІ. Сині клітини – неуражені, жовті – промахи/відсіяні, чорні – знищені кораблі).

Справа не лише у складності гри, а також у тому, що після розгадки принципу дії гри, багатьом вже було не так цікаво продовжувати. Гра перестала містити у собі елемент сюрпризу. Коли ж справа перейшла до другої програми, більшості довелося змінювати тактику вже після другого раунду, оскільки ШІ починав обстріл з тих позицій, на які гравці звикли ставити кораблі. Як видно з результатів експерименту,

використання адаптивних механік дозволило ШІ робити відносно неочікувані гравцями рішення по розставленню своїх кораблів, що трохи підвищило середню кількість неуражених клітин ШІ. Цікавий результат бою проти адаптивного ШІ, що тривав 78 кроків, наведено на рисунку 3.

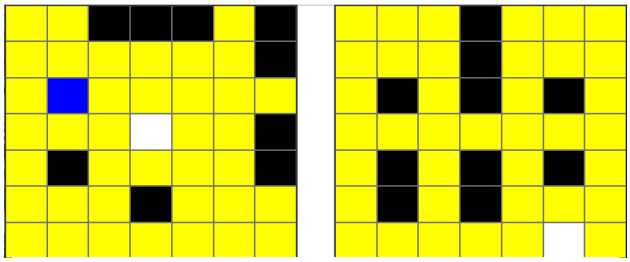

Рисунок 3 – Один з результатів гри проти адаптивного ШІ. Під кінець раунду у обох сторін залишалося по два варіанти для пострілу.

Висновки

У ході проведення дослідження запропоновано набір методів імплементації адаптивних механізмів штучного інтелекту у відеоіграх та рекомендації до їх застосування. Для кожного з методів наведено огляд та приклад застосування з коментуванням з точки зору принципів якісного відеоігрового дизайну.

Для демонстрації переваг методології адаптивних механізмів було розроблено дві ігрових програми на основі адаптивного та неадаптивного ШІ. У результаті сліпого тестування групою гравців було виявлено що застосування адаптивних механізмів збільшує інтерес гравців та реіграбельну цінність. Більшість гравців порахувало адаптивний ШІ складнішим супротивником та відмітили що його поведінка більш схожа на людську, аніж у випадку з неадаптивним ШІ.

Використання наведених методів дозволить розробникам ігрових програм створити гнучкі системи, здатні пристосовуватися до індивідуальних якостей гравців та підтримувати їх інтерес у ході ігрового процесу, а наведені приклади рішень ігрового дизайну допоможуть підібрати оптимальні допоміжні методи для проектів, структура яких вже визначена.

Список посилань

1. Schell J. The Art of Game Design: A Deck of Lenses / J. Schell – Morgan Kaufmann, 2008. – 489 p.
2. Fullerton T. Game Design Workshop / T. Fullerton – Morgan Kaufmann, 2008. – 496 p.
3. Rogers S. Level Up! The Guide to Great Video Game Design / S. Rogers. – John Wiley & Sons, 2010. – 520 p.
4. Csikszentmihalyi M. Flow: The Psychology of Optimal Experience / M. Csikszentmihalyi – Harper Perennial, 2008. – 336 p.
5. Difficulty Levels And Why You Should Never Use Them [Електронний ресурс]. Режим доступу: http://www.roguesnail.com/difficulty-levels/.
6. Designing Artificial Intelligence for Games [Електронний ресурс]. Режим доступу: https://software.intel.com/ru-ru/articles/designing-artificial-intelligence-for-games-part-1.
7. Creatures Wikia [Електронний ресурс]. Режим доступу: http://creatures.wikia.com/wiki/Creatures.
8. Офіційна сторінка Forza Motosport 5 [Електронний ресурс]. Режим доступу: https://www.forzamotorsport.net/en-US/games/fm5.
9. Офіційний сайт Left 4 Dead [Електронний ресурс]. Режим доступу: http://www.l4d.com/game.html.
10. Tijs T.J.W. Dynamic Game Balancing by Recognizing Affect / T.J.W. Tijs, D. Brokken, W.A. IJsselsteijn.// Lecture Notes in Computer Science : book series – 2008. – vol. 5294. – P. 6.
11. M. Lai. Giraffe: Using Deep Reinforcement Learning to Play Chess [Електронний ресурс]. Режим доступу: https://arxiv.org/abs/1509.01549v1.
12. D. Silver і D. Hassabis. AlphaGo: Mastering the ancient game of Go with Machine Learning [Електронний ресурс]. Режим доступу: https://research.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html
13. Оптимальный алгоритм игры в морской бой [Електронний ресурс]. Режим доступу: https://habrahabr.ru/post/180995/