УДК 53:004.92]:004.946.5
DOI: 10.31866/2617-796X.6.1.2023.283971

Людмила Вовк,
кандидат фізико-математичних наук,
доцент кафедри комп'ютерних наук,
Київський національний університет культури і мистецтв,
Київ, Україна
ludmylavera@gmail.com
https://orcid.org/0000-0001-8067-3640

Дмитро Недавній,
магістрант кафедри комп'ютерних наук,
Київський національний університет культури і мистецтв,
Київ, Україна
dim.ned125@gmail.com

ФІЗИКА В ІГРАХ. СТВОРЕННЯ ІГРОВОГО ДВИГУНА НА ОСНОВІ ФІЗИЧНИХ ПРОЦЕСІВ

Мета дослідження полягає в аналізі наявних ігрових двигунів і вивченні можливостей розробки та впровадження ігрового двигуна на основі фізичних процесів з використанням нового підходу, який розширить можливості моделювання ігрового середовища.

Методи дослідження ґрунтуються на застосуванні загальнонаукових і спеціальних методів, зокрема аналізу, синтезу, порівняння, аналогії та моделювання, системно-структурного аналізу, методів теоретичної систематизації та узагальнення результатів, наукового моделювання, що дало змогу змістовно проаналізувати предмет дослідження.

Наукова новизна. Запропоновано новий підхід до створення ігрового двигуна на основі фізичних процесів, який використовує алгоритми розрахунку колізій та динамічного моделювання твердих тіл і рідин.

Висновки. Проаналізовано ігрові двигуни, що на сьогодні функціонують, та описано характеристики й особливості фізичних двигунів. Наголошено, що фізичний двигун в ігровій розробці виконує дві основні функції: виявлення зіткнень між об'єктами й імітацію сил і рухів унаслідок цих зіткнень. Крім імітації фізичних процесів твердих тіл, фізичні дви-

гуни можуть реалізовувати додаткові можливості: спеціальну підтримку моделювання руху твердих тіл, води й інших рідин, симуляцію тканин і одягу, різноманітних частинок, додаткову підтримку для симуляції персонажів – високорівневі контролери персонажів, вбудовану підтримку rag-dolls, підтримку анімації. Описано новий підхід до створення та використання ігрового двигуна на основі фізичних процесів, який дає змогу створювати більш реалістичні ігрові середовища з використанням різноманітних ефектів, імітуючи фізичні процеси в грі. Ігровий двигун, створений на основі нового підходу до вирішення проблеми, використовує алгоритми розрахунку колізій та динамічного моделювання твердих тіл і рідин, що розширює можливості моделювання активностей, забезпечуючи високу точність і реалістичність візуалізації процесів. Новий підхід дає змогу розширювати можливості ігрового двигуна та підтримувати різноманітні сценарії ігор.

Ключові слова: фізичний движок; динаміка; зіткнення об'єктів; бібліотека OpenGL; PhysX; Havok Physics; Construct 2; Unreal Engine; MoneGame / XNA.

Вступ. У галузі розробки комп'ютерних ігор на сьогодні є велика потреба в ігрових двигунах, які забезпечували б широкі можливості моделювання фізичних процесів. Ігровий двигун – це програмне забезпечення, платформа, створена для розробки комп'ютерних ігор. Ігровий двигун забезпечує розробникам ігор необхідний функціонал для створення графіки, фізики, звуку, штучного інтелекту, мережевого зв'язку й інших аспектів, які необхідні для створення ігрового досвіду користувачем ігрового програмного продукту. Розробники можуть використовувати ігрові двигуни для створення різних жанрів ігор – від простих платформерів до складних стратегій і шутерів від першої особи. Ігровий досвід тлумачимо як сукупність вражень, які отримує гравець під час гри, що містить такі елементи: графіку, звук, сюжет, взаємодію з ігровим світом та інші чинники, які впливають на емоційний стан гравця. Відтворення реалістичного фізичного середовища є важливим елементом геймплею в багатьох популярних жанрах ігор.

Моделювання комп'ютерних ігор – це процес створення віртуального світу або сценарію, який використовується для створення гри, а також охоплює створення моделей персонажів, різноманітних об'єктів, транспорту, ландшафту й інших елементів віртуального ігрового світу, установлення їх поведінки, реакції на дії інших гравців тощо. Моделювання комп'ютерних ігор можна виконати різними способами, у тому числі створенням 3D-моделей з використанням спеціалізованих програмних засобів. Після створення моделей їх можна інтегрувати в ігровий двигун для створення повноцінного ігрового світу, з яким можна взаємодіяти. Моделювання є важливим етапом розробки комп'ютерних ігор, оскільки дає змогу створювати реалістичні ігрові світи, які забезпечують більш імерсивний ігровий досвід для гравців. Це можна досягнути за допомогою використання вражальної графіки, реалістичного звуку, глибокого сюжету, штучного інтелекту й інших елементів, які забезпечують ілюзію реальності.

Фізика в комп'ютерних іграх – це система, яка відтворює закони фізики в ігровому середовищі, забезпечуючи реалістичну поведінку об'єктів і персонажів у грі. Фізичний рух, гравітація, динаміка зіткнень, ефекти світла й тіні, поведінка рідин і газів можуть бути більш чи менш реалістично відтворені під час розробки ігор

з використанням законів фізики. У більшості сучасних ігрових двигунів вбудовано фізичну систему, яка дає змогу іграм відтворювати фізичні взаємодії між об'єктами та персонажами в режимі реального часу (Бреславець, 2018). Саме фізика у комп'ютерних іграх відіграє ключову роль для створення реалістичного й імерсивного ігрового досвіду гравців.

За даними статистики, ринок комп'ютерних ігор неухильно зростає і темпи зростання оцінюють у 8 % на рік, водночас обсяг ринку мобільних ігор демонструє щорічне зростання на 15 % (Brightman, 2015).

Тому вдосконалення ігрового двигуна на основі фізичних процесів є предметом цього дослідження та завданням, яке потребує нових цікавих ідей та їх упроваджень.

Фізика в іграх є актуальною темою дослідження для багатьох науковців та інженерів, які працюють у сфері розробки комп'ютерних ігор. Слід зазначити, що в наукових і прикладних дослідженнях фізичні процеси в іграх здебільшого підлягали моделюванню на основі класичної механіки та кінематики (Baraff, 1977a).

У цій роботі запропоновано створення ігрового двигуна, який розширює можливості моделювання фізичних процесів за допомогою нового підходу, що дає змогу досягти більш високої рівноваги між реалістичністю фізичного середовища та продуктивністю гри.

Результати дослідження. Проаналізуємо ігрові двигуни, що на сьогодні функціонують, та опишемо характеристики й особливості фізичних двигунів.

Ігровий двигун зазвичай складається з таких компонентів, як графічний двигун, фізичний двигун, звуковий двигун та ігровий штучний інтелект.

Графічний двигун (англ. graphics engine) – програмне забезпечення, яке відповідає за рендеринг двомірної або тривимірної комп'ютерної графіки, забезпечуючи різні функції, необхідні для створення та відтворення графічних ефектів, таких як рух, світло, тіні, текстури й інші візуальні ефекти. Графічний двигун може охоплювати різні компоненти, такі як рушій геометрії, рушій фізики, системи частинок, системи освітлення та інші елементи, які відповідають за певні складники візуальної частини гри. Ці компоненти взаємодіють між собою та з іншими системами в ігровому двигуні, такими як система штучного інтелекту та система звуку, створюючи ігровий досвід користувача.

Фізичний двигун (англ. physics engine) – програмне забезпечення, яке відповідає за симуляцію реалістичної фізики в іграх. Фізичний двигун дає змогу обчислювати рух об'єктів у грі, їхню поведінку під час зіткнення, гравітації, тертя та інших фізичних взаємодій. Одним з найпопулярніших фізичних двигунів для 2D-ігор є Box2D, який написали спочатку мовою програмування C++, а далі портували практично на всі популярні мови програмування. Іншим досить відомим двигуном є Chipmunk2D, який використовують як основний у таких фреймворках, як Cocos2D (Helgason, 2013).

Є низка жанрів, у яких масове застосування фізики почалося порівняно недавно. Це передусім шутери й ігри з видом від третьої особи. Спектр використання фізичних ефектів у них дуже широкий – від досить простих rag-dolls і штовхання ящиків до спроб побудувати частину геймплею на принципах фізики (Half-life 2, Psi-ops і т.д.).

Звуковий двигун (англ. sound / audio engine) – програмне забезпечення, яке відповідає за генерацію, обробку та відтворення звуків у комп'ютерних іграх. Звук є важливим складником відчуття імерсії гравців в ігрове середовище. Звуковий двигун дає змогу розробникам створювати різноманітні ефекти звуку, які допомагають передати атмосферу гри та підсилюють інтенсивність геймплею. Звукові двигуни можуть мати різні функції, такі як обробка різних типів звуків (музика, звукові ефекти, діалоги тощо), змішування звуків від різних джерел, створення 3D-звуку для відчуття передачі простору в грі та інші.

Ігровий штучний інтелект (англ. game artificial intelligence) – програмне забезпечення, яке відповідає за створення різноманітних алгоритмів і поведінки неігрових персонажів (NPCs) у комп'ютерних іграх. Використовується для вдосконалення реалістичності поведінки персонажів, ігрового середовища та створення різноманітних спецефектів, що комплексно сприяє підвищенню рівня імерсії гравців.

До появи ігрових двигунів існував індивідуалізований підхід до створення нових ігор, особливістю якого була оптимізація під конкретну платформу, оскільки апаратне забезпечення того часу було обмеженим і мало різні архітектури. Кожна гра створювалася з урахуванням можливостей конкретної платформи, тому розробникам доводилося докладати значних зусиль для досягнення максимальної продуктивності та якості гри (Pattrasitidecha, 2014).

Однак поява ігрових двигунів, стандартизованих для розробки ігор, змінила процес створення нових ігор, адже розробники отримали змогу використовувати готові рішення та інструменти, оптимізуючи час і зусилля. Сучасні ігрові двигуни також дають змогу розробникам створювати ігри для різних платформ з використанням одного коду, що забезпечує швидке й ефективне створення гри.

Більшість ігрових двигунів підтримують різні платформи та пристрої, включаючи персональні комп'ютери, консолі та мобільні пристрої, що забезпечує гнучкість й ефективність в розробці ігор, а також дає змогу розробникам створювати ігри для різних платформ, незважаючи на тип апаратного забезпечення, який використовується. Отже, ігрові двигуни значно спрощують процес створення ігор і забезпечують більш швидке й ефективне розроблення ігор для різних платформ.

Зазначимо, що ігрові двигуни стали не лише інструментом для розробки ігор, але й універсальним інструментом для розробки різноманітних застосунків, що вимагають візуалізації технічних процесів, симуляції, візуалізації даних та створення інтерфейсу користувача. Головним чинником, який призвів до такого розвитку ігрових двигунів, є їхня відкритість і гнучкість. Більшість ігрових двигунів, таких як Unity, Unreal Engine, Godot, мають відкритий код, що дає змогу розробникам налаштовувати їх під свої потреби, використовуючи для створення застосунків. Ігрові двигуни мають багатий набір інструментів і можливостей, що дає змогу створювати візуально привабливі й ефективні застосунки. Це забезпечує, зокрема, можливість використання різних ефектів, анімацій, світла та тіней тощо (Gregory, 2014).

Розглянемо детальніше характеристики та властивості сучасних фізичних двигунів. Фізичні двигуни можуть симулювати такі фізичні явища й стани: динаміку абсолютно твердого тіла; динаміку деформованого тіла; динаміку рідин; динаміку газів; поведінку тканин; поведінку мотузок (троси, канати і т. п.). Фізична си-

муляція є важливим складником багатьох ігор, де гравцям надається можливість взаємодіяти з оточенням і різними об'єктами в процесі гри. Фізична симуляція важлива для забезпечення реалістичності поведінки об'єктів.

Слід зазначити, що не всі ігри потребують фізичної симуляції. Це ігри, які використовують простіші механіки руху та колізій і можуть бути оброблені без апаратної підтримки. Використання фізичних двигунів залежить від потреб гри та можливостей апаратного забезпечення.

Раніше фізична симуляція в іграх забезпечувалася лише за допомогою ресурсів центрального процесора, що призвело до значних обмежень щодо кількості об'єктів, які можуть бути симульовані одночасно та рівня деталізації самої симуляції. Однак з появою апаратної підтримки фізичних двигунів, таких як Nvidia PhysX, можливості фізичної симуляції значно збільшилися. Фізичний двигун може використовувати ресурси відеокарт Nvidia для обробки фізичних ефектів у реальному часі. Це дає змогу зменшити навантаження на центральний процесор і збільшити кількість об'єктів, які можуть бути симульовані одночасно, а також рівень деталізації симуляції. Така апаратна підтримка фізичних двигунів дає змогу розробникам створювати більш реалістичні ігри з більшим числом фізичних ефектів. Наприклад, фізичну симуляцію можна використати для створення реалістичних рухів персонажів, взаємодії з оточенням, обробки колізій і вибухів.

Загалом фізичний двигун в ігровій розробці виконує дві основні функції: виявлення зіткнень між об'єктами й імітацію сил і рухів унаслідок цих зіткнень. Для виявлення зіткнень використовують підсистему зіткнень з двома основними параметрами – швидкістю роботи й точністю визначення зіткнень. Недостатня точність може призвести до перекриття об'єктів і пропуску зіткнень у разі значно різних розмірів і швидкостей об'єктів. Для прискорення роботи підсистеми зіткнень використовують різні методи розбиття простору на підпростори, такі як quadtree та octree. Системи зіткнень працюють за принципом дискретно-точкових взаємодій, тому зіткнення швидко рухомих об'єктів можуть не фіксуватися. Для уникнення таких артефактів деякі системи зіткнень використовують Continuous Collision Detection (CCD) – метод, який полягає в перевірці зіткнень між витягнутими обсягами, що представляють рух об'єктів за весь часовий крок.

Більшість фізичних двигунів може симулювати фізику твердого тіла, яке не змінює свою форму, такого як цегла, стіл, стіна і т. п. Фізику твердого тіла використовують у більшості ігор через її прийнятну продуктивність. Для представлення обсягу твердих тіл використовують примітивні тіла, такі як прямокутники, сфери, циліндри, конуси тощо, або більш складні опуклі чи неопуклі багатогранники. Більш складні тіла можуть описуватися за допомогою апроксимації примітивами. Матеріал твердих тіл описується параметрами, такими як коефіцієнт тертя (два типи: коефіцієнт тертя спокою та коефіцієнт тертя руху), пружність тощо. Рух твердих тіл описується за допомогою лінійної, кутової швидкості та прискорення, а їх параметри можна встановлювати за допомогою програми або фізичних сил, які впливають на прискорення тіл або імпульсів, які впливають на швидкості. Тверде тіло також має масу.

Фізична поведінка персонажів у відеоіграх, зокрема ефект тканинної ляльки (rag-doll), реалізується на базі фізики твердого тіла. Однак для цього потріб-

но використовувати не тільки тверді тіла, а й зчленування (joint) або обмеження (constraint). Джоінт – це точка з'єднання двох твердих тіл, яка накладає обмеження на їх положення в просторі або на їх швидкості щодо одне одного. Регдоли, наприклад, реалізуються за допомогою ball-joint і hinge-joint. Ball-joint обмежує переміщення тіл і їх повороти по трьох осях, а hinge-joint – повороти тіл щодо одне одного навколо однієї осі. Застосування джоінтів дуже широке та не обмежується тільки реалізацією регдолів (Baraff, 1977b).

Крім фізики твердого тіла, різні фізичні двигуни можуть реалізовувати додаткові можливості: спеціальну підтримку симуляції руху автомобілів, симуляцію води й інших рідин, симуляцію тканин і одягу, симуляцію частинок, додаткову підтримку для симуляції персонажів – високорівневі контролери персонажів, вбудовану підтримку rag-dolls, підтримку анімації.

Основними етапами створення ігрового двигуна на основі фізичних процесів є:

– визначення фізичних законів, які будуть використовуватися в грі (наприклад, закони Ньютона, закони термодинаміки, закони електродинаміки);

– розробка алгоритмів, які будуть застосовуватися для моделювання фізичних процесів у грі;

– реалізація алгоритмів у програмному забезпеченні, яке використовується для розробки гри;

– тестування та оптимізація ігрового двигуна;

– розробка гри на базі створеного ігрового двигуна (етап розробки графіки, звукового супроводу й інших елементів гри).

Створення ігрового двигуна на основі фізичних процесів вимагає врахування різноманітних фізичних законів, які пов'язані з рухом твердих тіл, взаємодією зі статичними та динамічними об'єктами, а також зі зміною форми та об'єму твердих тіл. Саме для досягнення цих цілей розроблено систему фізичного моделювання, яка базується на принципах фізики твердого тіла.

Наприклад, розглянемо кілька реальних двигунів, наведемо характеристики та створені на їхній основі ігри, адже комерційно успішні проєкти є підставою для подальшого використання саме цих двигунів на практиці (Астахов та Болтач, 2022).

Так, Unity (або Unity3D) є одним з найбільш популярних ігрових двигунів, який використовують для створення ігор на різних платформах, таких як Windows, macOS, Linux, Android, iOS, Xbox, PlayStation тощо. Архітектура Unity3D базується на компонентному підході, що дає змогу розробникам легко створювати та модифікувати різні елементи гри. Двигун підтримує різні мови програмування, такі як C#, JavaScript, Boo, що робить його доступним для широкого кола розробників. Unity3D налічує велику спільноту, яка надає користувачам безкоштовну і платну допомогу, навчальні курси та відеоуроки. Наявність безплатної версії дає змогу навчатися та експериментувати з двигуном без великих витрат на програмне забезпечення. Тож не дивно, що двигун, створений у 2005 році, у 2013 році нараховував кількість зареєстрованих користувачів Unity3D у розмірі 2 мільйонів (Unity, Source, 2015), що засвідчило популярність двигуна серед розробників ігор. За допомогою Unity3D можна створювати як 2D-, так і 3D-ігри різних жанрів (від

платформерів до шутерів і масштабних RPG). Двигун також підтримує створення різних інтерактивних застосунків і віртуальної реальності (Hocking, 2018).

Прикладами впровадження стали такі відомі ігри як Endless Legend, Endless Space, Temple Run та багато інших.

Відомим продуктом Microsoft є ігровий двигун MoneGame/XNA, який створили на основі .Net Framework. До появи Windows 8 двигун давав можливість розробки ігор для всіх платформ Windows. Після відмови Microsoft у подальшій підтримці проєкту створено версію MonoGame, яка є популярною та розвивається і нині завдяки популярності та простоті ігрового двигуна XNA.

До особливостей MonoGame зараховують відносно низький поріг входження, оскільки простий інтерфейс дає змогу швидко розпочати процес розробки ігор; відкритий код – MonoGame є вільним і відкритим програмним забезпеченням та підтримує співпрацю з іншими відкритими проєктами; інтегрується з Visual Studio, що дає змогу зручно розробляти ігри для платформ Windows, Xbox, PlayStation; окрім зазначених, підтримує ще такі платформи, як macOS, Linux, Android, iOS, Nintendo Switch; має широкий інструмент для розробки 2D- і 3D-графіки; підтримує різні функції: аудіо, фізику, мережеву гру, штучний інтелект, що надає допомогу в розробці складних ігор з багатьма функціями (Бреславець, 2018).

Приклади впровадження – ігрові продукти, такі як Transistor, Bastio, Fez Magicka й інші.

Компанія Epic Games є розробником добре відомого та популярного в професійному середовищі розробників ігрового двигуна Unreal Engine. Переваги: багатофункціональність, оскільки Unreal Engine підтримує створення ігор різних жанрів і рівнів складності; графічна якість – завдяки сучасним графічним технологіям DirectX, OpenGL; кросплатформність – підтримка створення ігор для платформ Windows, Mac, Linux, Xbox, PlayStation, Nintendo, iOS, Android, AR і VR; Unreal Engine пропонує потужні візуальні ефекти, включаючи реалістичне освітлення, воду, вогонь, дим, туман, тіні й інше; забезпечує швидкодію; має вбудовану систему візуального програмування Blueprints, яка дає змогу розробникам створювати складні функції та логіку гри, не займаючись програмуванням на C++, та інші переваги й особливості (Батіна уклад., 2021).

Відомі впроваджені ігрові продукти: серії ігор Тома Clancy's Splinter Cell Series та BioShock, Might & Magic Heroes VII та інші (Батіна уклад., 2021).

Відомий у середовищі як початківців, так і професійних розробників ігровий двигун Construct 2, особливостями якого є простота використання; інтеграція з іншими програмами; велика кількість готових рішень; підтримка фізики; підтримка мережевої гри; є інструменти для розвитку. Це дає змогу розробникам завдяки вбудованим інструментам тестувати гру, швидко виявляти та виправляти помилки (Батіна уклад., 2021). На сьогодні ще немає ігор, розроблених з використанням цього двигуна.

Найпоширенішими фізичними двигунами є:

– PhysX – основний конкурент Havok, єдиний у світі фізичний двигун, що має апаратну підтримку. На сьогодні PhysX займає перше місце за популярністю серед фізичних двигунів;

– Bullet Physics Library – другий за популярністю серед вільних фізичних двигунів;

– Open Dynamics Engine – третій за популярністю серед вільних фізичних двигунів;

– Tokamak – фізичний двигун з відкритим вихідним кодом;

– Newton Game Dynamics – спочатку пропрієтарний, а з лютого 2011 року – вільний фізичний двигун;

– Box2D – мультиплатформенний двигун для симуляції фізики твердих тіл;

– Havok Physics – фізичний двигун, який використовують у випадках, коли необхідно реалізувати в грі складну симуляцію різних фізичних впливів у режимі реального часу. Якщо планують реалістичну та видовищну гру, без застосування Havok Physics не обійтися. Цей двигун є мультиплатформенним і використовується для створення ігрових застосунків не тільки для комп'ютерів, а й для всіх актуальних ігрових консолей, включаючи портативні та гібридні приставки.

Для описаних вище двигунів наведемо приклад вибору найбільш оптимального. Для оцінки опційних і апріорних вимог використаємо метод рейтингової оцінки. Це означає, якщо двигун відповідає такій вимозі, він отримує додатковий бал рейтингу. Для кожної з цих характеристик використаємо трибальну шкалу оцінювання:

1. Опційна вимога:

0 балів – якщо двигун не відповідає вимозі;

2 бали – якщо відповідає.

2. Документація:

1 бал – майже немає документації у відкритому доступі;

2 бали – хороша документація англійською мовою;

3 бали – документація доступна рідною мовою.

3. Історія застосування:

0 балів – немає відомих ігор;

1 бал – 1–5 відомих ігор;

2 бали – 5–9 відомих ігор;

3 бали – 10 і більше відомих ігор.

На сьогодні за допомогою фізичного двигуна Havok Physics створено понад півтори сотні ігор. Є навіть інформація, що саме Havok Physics був частково інтегрований у відомий двигун Source, у ту частину, що відповідає безпосередньо за фізику моделей. Також його використовують і в інших видах професійного програмного забезпечення, таких як 3ds Max і Maya. Використовують його і в різних реалістичних симуляторах і тренажерах, щоб навчитися водити наземний транспорт і пілотувати авіасудна. Також є тривимірний графічний двигун Havok Vision Engine і програма для реалізації ігрового штучного інтелекту Havok AI.

Одним з ключових елементів розробки ігрового двигуна є алгоритм розрахунку колізій між твердими тілами (Curtis, Tamstorf and Manocha, 2008). У роботі запропоновано використання алгоритму дискретної моделі, який забезпечує високу точність обчислень колізій, а також оптимізацію часу, який витрачається на

обчислення. Крім того, розроблено алгоритми для моделювання руху твердих тіл з урахуванням впливу сили тертя, повітряного опору й інших фізичних чинників.

Для вибору засобів розробки ігрового двигуна використано порівняльно-аналітичний метод, відображений у таблиці 1.

Таблиця 1

Порівняльно-аналітичний метод вибору засобів розробки ігрового двигуна

Характеристика	Вимоги	Unity	Unreal	MonoGame	Construct 2
Режим рендерингу	3D	✓	✓	✓	-
Фізичний двигун		✓	✓	✓	✓
Цільовий жанр	RPG	✓	✓	✓	✓
Цільові платформи	Windows, OSX	✓	✓	✓	✓
Мова розробки	C# або JavaScript	✓	✓	✓	✓
Інтеграція з IDE	Visual Studio	✓	✓	✓	-
ОС для розробки	Windows	✓	✓	✓	✓
Цільова аудиторія	Для професіоналів	✓	✓	✓	-
Ціна	Безкоштовно	✓	✓	✓	-
Якість документації		3	3	3	3
Рівень сумісності		3	2	1	1
Інтерфейс користувача		3	3	2	3
Історія застосування		3	3	2	0
Остаточний рейтинг		12	12	8	Не підходить

Джерело: результати власних досліджень.

Ще одним важливим складником ігрового двигуна є система динамічної рідини. Розроблено алгоритми, які дають змогу моделювати поведінку рідини з урахуванням впливу гравітації, сили тиску й інших фізичних чинників. Це дає змогу досягти більш високої реалістичності в графічному відображенні води й інших рідинних середовищ. Запропоновані нові підходи до моделювання ігрового двигуна дають змогу підтримувати різноманітні геометричні форми, включаючи прямокутники, кола та багатокутники. Під час досліджень розглянуто та вирішено завдання розробки системи динамічної зміни форми твердих тіл, яка б дала змогу моделювати деформацію твердих тіл під час зіткнення з іншими об'єктами. Це забезпечує створення більш реалістичних сцен, пов'язаних з руйнуванням під час зіткнення об'єктів.

Для реалізації ігрового двигуна використано мову програмування C++, яка є однією з найбільш популярних мов програмування в галузі розробки комп'ютерних ігор. Для рендерингу графіки та фізичного моделювання використано відкриті бібліотеки OpenGL. Завдяки цьому перевагою отриманого ігрового двигуна стала підтримка різноманітних графічних ефектів, таких як, наприклад, водяні

хвилі, сніг, дим й інші. Створено низку демонстраційних сценаріїв, які ілюструють різні можливості нового ігрового двигуна.

Наприклад, під час виявлення явищ зіткнень об'єктів у створюваному фізичному двигуні можна абстрагуватися від концепції різних форм і враховувати тільки нормаль між точками і те, наскільки глибоко об'єкти перебувають один в одному. У кожної форми має бути свій тип колайдера, що містить її властивості, і база для збереження. Будь-який тип колайдера повинен мати можливість перевірити наявність зіткнення з будь-яким іншим типом, тому додаються функції до бази кожного з них. Ці функції прийматимуть Transform, тому колайдери зможуть використовувати відносні координати. Програмний код буде мати вигляд:

```
struct Collider {
virtual CollisionPoints TestCollision(
const Transform* transform,
const Collider* collider,
const Transform* colliderTransform) const = 0;
virtual CollisionPoints TestCollision(
const Transform* transform,
const SphereCollider* sphere,
const Transform* sphereTransform) const = 0;
virtual CollisionPoints TestCollision(
const Transform* transform,
const PlaneCollider* plane,
const Transform* planeTransform) const = 0;
};
```

Створимо два типи колайдерів і подивимося, як вони взаємодіють. Сфера визначається як точка і радіус, а площина – як вектор і відстань. Перевизначимо функції з Collider. Для кожного колайдера можна вибрати, які інші колайдери він виявить, заповнивши або залишивши ці функції порожніми. У цьому разі нам не потрібні зіткнення між площинами, тому повертаємо порожню точку зіткнення.

```
struct SphereCollider
: Collider
{
vector3 Center;
float Radius;
CollisionPoints TestCollision(
const Transform* transform,
const Collider* collider,
const Transform* colliderTransform) const override
{
return collider->TestCollision(colliderTransform, this, transform);
}
CollisionPoints TestCollision(
const Transform* transform,
const SphereCollider* sphere,
```

```
const Transform* sphereTransform) const override
{
return algo::FindSphereSphereCollisionPoints(
this, transform, sphere, sphereTransform);
}
CollisionPoints TestCollision(
const Transform* transform,
const PlaneCollider* plane,
const Transform* planeTransform) const override
{
return algo::FindSpherePlaneCollisionPoints(
this, transform, plane, planeTransform);
}
};
```

Висновки. Розглянуто наявні на сьогодні ігрові двигуни. Окремо проаналізовано види фізичних двигунів та описано їхні особливості. Зроблено спроби створення ігрового двигуна на основі фізичних процесів, що дає змогу створювати більш реалістичні ігрові середовища з використанням різноманітних ефектів, симулюючи фізичні процеси в грі. Ігровий двигун, створений на основі нових ідей та інноваційних підходів до вирішення проблеми, використовує алгоритми розрахунку колізій і динамічного моделювання твердих тіл та рідин, що забезпечує високу точність і реалістичність моделювання.

Новий підхід дає змогу розширювати можливості ігрового двигуна та підтримувати різноманітні сценарії ігор. Слід зазначити, що тривають подальші дослідження щодо розширення спектра можливостей комп'ютерної графіки на основі ігрового двигуна, вивчається можливість забезпечення нових ефектів на основі таких фізичних процесів, як підтримка м'яких тіл, більш точна симуляція рідини та повітряних потоків тощо.

СПИСОК ПОСИЛАНЬ

Астахов, В.І. та Болтач, С.В., 2022. Порівняльний аналіз використання доповненої та віртуальної реальності в сфері розробки ігор. В: *Комп'ютерні ігри та мультимедіа як інноваційний підхід до комунікації,* II Всеукраїнська науково-технічна конференція молодих вчених, аспірантів та студентів. Одеса, Україна, 29-30 вересня 2022 р. Одеса, с.101-104.

Батіна, О.А., уклад., 2021. *Технології створення освітніх комп'ютерних ігор та проектування доповненої реальності*. [online] Київ: КПІ ім. Ігоря Сікорського. Доступно: <https://ela.kpi.ua/bitstream/123456789/43547/1/Konspekt_lektsii.pdf> [Дата звернення 20 квітня 2023].

Бреславець, В.С., 2018. *Технології розробки комп'ютерних ігор*. Харків: Друкарня Мадрид.

Baraff, D., 1997a. *An Introduction to Physically Based Modeling: Rigid Body Simulation I – Unconstrained Rigid Body Dynamics.* [online] Available at: <http://www.cs.cmu.edu/~baraff/sigcourse/notesd1.pdf> [Accessed 20 April 2023].

Baraff, D., 1997b. *An Introduction to Physically Based Modeling: Rigid Body Simulation II – Nonpenetration Constraints.* [online] Available at: <http: // www. cs. cmu. edu / ~baraff / sigcourse / notesd2. pdf> [Accessed 20 April 2023].
Brightman, J., 2015. Games software revenues to reach $110 billion by 2018 - Digi-Capital. *GamesIndustry.biz*, [online] 4 May. Available at: <https: // www. gamesindustry. biz / games-software-revenues-to-reach-usd110-billion-by-2018-digi-capital> [Accessed 20 April 2023].
Curtis, S., Tamstorf, R. and Manocha, D., 2008. Fast collision detection for deformable models using representative-triangles. In: *Proceedings of the 2008 Symposium on Interactive 3D Graphics, SI3D 2008,* 15-17 February 2008. Redwood City, CA, USA. [e-journal] New York, pp.61-69. https://doi.org/10.1145/1342250.1342260
Gregory, J., 2014. *Game Engine Architecture*. 2nd ed. New York: CRC Press.
Helgason, D., 2013. Another million unity developers in the house. *Unity,* [online] 9 July. Available at: <https://blog.unity.com/community/another-million-unity-developers-in-the-house> [Accessed 20 April 2023].
Hocking, J., 2018. *Unity in Action: Multiplatform game development in C#*. 2nd ed. Manning.
Pattrasitidecha, A., 2014. Comparison and evaluation of 3D mobile game engines. *Chalmers.* [online] Available at: <http:// publications. lib. chalmers. se / records / fulltext / 193979 / 193979. pdf> [Accessed 20 April 2023].
Unity, Source 2, Unreal Engine 4, or CryENGINE – Which Game Engine Should I Choose?, 2015. *Pluralsight*, [online] 5 March. Available at: <https://www.pluralsight.com/blog/film-games/unity-udk-cryengine-game-engine-choose> [Accessed 20 April 2023].

REFERENCES

Astakhov, V.I. and Boltach, S.V., 2022. Porivnialnyi analiz vykorystannia dopovnenoi ta virtualnoi realnosti v sferi rozrobky ihor [Comparative analysis of the use of augmented and virtual reality in the field of game development]. In: *Kompiuterni ihry ta multymedia yak innovatsiinyi pidkhid do komunikatsii* [Computer games and multimedia as an innovative approach to communication], II All-Ukrainian scientific and technical conference of young scientists, graduate students and students. Odesa, Ukraine, 29-30 September 2022. Odesa, pp.101-104.
Baraff, D., 1997a. *An Introduction to Physically Based Modeling: Rigid Body Simulation I – Unconstrained Rigid Body Dynamics*. [online] Available at: <http://www.cs.cmu.edu/~baraff/sigcourse/notesd1.pdf> [Accessed 20 April 2023].
Baraff, D., 1997b. *An Introduction to Physically Based Modeling: Rigid Body Simulation II – Nonpenetration Constraints*. [online] Available at: <http: // www. cs. cmu. edu / ~baraff / sigcourse / notesd2. pdf> [Accessed 20 April 2023].
Batina, O.A., comp., 2021. *Tekhnolohiï stvorennia osvitnikh kompiuternykh ihor ta proektuvannia dopovnenoï realnosti* [Technologies for creating educational computer games and designing augmented reality]. [online] Kyiv: KPI im. Ihoria Sikorskoho. Available at: <https://ela.kpi.ua/bitstream/123456789/43547/1/Konspekt_lektsii.pdf> [Accessed 20 April 2023].
Breslavets, V.S., 2018. *Tekhnolohii rozrobky kompiuternykh ihor* [Computer game development technologies]. Xarkiv: Drukarnia Madryd.
Brightman, J., 2015. Games software revenues to reach $110 billion by 2018 – Digi-Capital. *GamesIndustry.biz*, [online] 4 May. Available at: <https://www. gamesindustry.biz/games-software-revenues-to-reach-usd110-billion-by-2018-digi-capital> [Accessed 20 April 2023].

Curtis, S., Tamstorf, R. and Manocha, D., 2008. Fast collision detection for deformable models using representative-triangles. In: *Proceedings of the 2008 Symposium on Interactive 3D Graphics, SI3D 2008, 15-17 February 2008*. Redwood City, CA, USA. [e-journal] New York, pp.61-69. https://doi.org/10.1145/1342250.1342260
Gregory, J., 2014. *Game Engine Architecture*. 2nd ed. New York: CRC Press.
Helgason, D., 2013. Another million unity developers in the house. *Unity*, [online] 9 July. Available at: <https://blog.unity.com/community/another-million-unity-developers-in-the-house> [Accessed 20 April 2023].
Hocking, J., 2018. *Unity in Action: Multiplatform game development in C#*. 2nd ed. Manning.
Pattrasitidecha, A., 2014. Comparison and evaluation of 3D mobile game engines. *Chalmers*. [online] Available at: <http:// publications. lib. chalmers. se / records / fulltext / 193979 / 193979. pdf> [Accessed 20 April 2023].
Unity, Source 2, Unreal Engine 4, or CryENGINE – Which Game Engine Should I Choose?, 2015. *Pluralsight*, [online] 5 March. Available at: <https://www.pluralsight.com/blog/film-games/unity-udk-cryengine-game-engine-choose> [Accessed 20 April 2023].

UDC 53:004.92]:004.946.5

Liudmyla Vovk,
PhD in Physics and Mathematics,
Associate Professor at the Department of Computer Science,
Kyiv National University of Culture and Arts,
Kyiv, Ukraine
ludmylavera@gmail.com
https://orcid.org/0000-0001-8067-3640

Dmytro Nedavnii,
Master's Student at the Department of Computer Science,
Kyiv National University of Culture and Arts,
Kyiv, Ukraine
dim.ned125@gmail.com

PHYSICS IN GAMES. CREATION A GAME ENGINE BASED ON PHYSICAL PROCESSES

The purpose of the research is to analyze existing game engines and study the possibilities of developing and implementing a game engine based on physical processes using a new approach that will expand the capabilities of game environment modelling.

The research methods are based on the implementation of general scientific and specific methods, including analysis, synthesis, comparison, analogy and modelling, system-structural analysis, methods of theoretical systematization and generalization of results, and scientific modelling, which allowed for a meaningful analysis of the subject matter.

Scientific novelty. A new approach to the creation of a game engine based on physical processes, which uses algorithms for calculating collisions and dynamic modelling of solids and liquids, is proposed.

Conclusions. The currently functioning game engines are analyzed and the characteristics and features of physical engines are described. It is emphasized that the physics engine in game development performs two main functions: detecting collisions between objects and simulating forces and movements resulting from these collisions. In addition to simulating the physical processes of solids, physics engines can implement additional features: special support for modelling the movement of solids, water and other liquids, simulation of fabrics and clothing, various particles, additional support for character simulation – high-level character controllers, built-in support for rag-dolls, and support for animation. A new approach to creating and using a game engine based on physical processes is described, which allows creating more realistic game environments using a variety of effects, simulating the physical processes in the game. The game engine, created on the basis of a new approach to solving the problem, uses algorithms for calculating collisions and dynamic modelling of solids and liquids, which expands the capabilities of activity modelling, ensuring high accuracy and realistic visualization of processes. The new approach allows expanding the capabilities of the game engine and supporting a variety of game scenarios.

Keywords: physics engine; dynamics; object collision; OpenGL library; PhysX; Havok Physics; Construct 2; Unreal Engine; MoneGame / XNA.

24.04.2023