УДК 004.94
DOI https://doi.org/10.36994/2788-5518-2025-02-10-11

РОЗРОБЛЕННЯ ТА АНАЛІЗ МАТЕМАТИЧНИХ МОДЕЛЕЙ І ОБЧИСЛЮВАЛЬНИХ МЕТОДІВ ДЛЯ СИМУЛЯЦІЇ ПРОЦЕСІВ У ВІРТУАЛЬНІЙ РЕАЛЬНОСТІ

Кулик Ю. Р., *аспірант, Національний університет «Львівська політехніка», Львів, Україна. https://orcid.org/0009-0006-1121-3302, yurii-marko.r.kulyk@lpnu.ua*

Батюк А. Є., *к.т.н., доц., Національний університет «Львівська політехніка», Львів, Україна. https://orcid.org/0000-0001-7650-7383, anatolii.y.batiuk@lpnu.ua*

Анотація. *Дана робота представляє комплексний аналіз математичних моделей, обчислювальних методів та алгоритмів візуалізації, які формують фундаментальну основу для розробки інтерактивних середовищ віртуальної реальності (ВР) з високим ступенем фізичної достовірності та візуальної якості. Метою дослідження є не лише систематизація теоретичних основ тривимірного геометричного моделювання, фізичної симуляції твердих тіл та алгоритмів рендерингу у реальному часі, але й глибокий критичний аналіз їхньої практичної реалізації, архітектурних обмежень, що поєднує рушій "Source" у його реалізації через платформу "Garry's Mod" з сучасним програмним інтерфейсом віртуальної реальності "OpenVR", з використанням стаціонарної робочої машини (персонального комп'ютера) на базі операційної системи "Windows 10".*

Центральним науковим та інженерним викликом у даному контексті виступає застосування не просто добре відомих математичних методів до задач віртуальної реальності, а архітектурних рішень сучасних ВР-систем, для підтримки стереоскопічного рендерингу, просторового відстеження, взаємодію з об'єктами та інструментами в віртуальному середовищі, у принципово іншу архітектуру рушія "Source".

Описане у даній роботі програмне рішення, що перебуває у стані активної розробки та вдосконалення, виступає як складний багаторівневий «шар-транслятор» або програмний міст, який забезпечує з'єднання між низькорівневим апаратним забезпеченням віртуальної реальності. Включає роботу з шоломами з вбудованими дисплеями високої роздільності та системами відстеження, контролерів, що комунікують через стандартизований програмний інтерфейс OpenVR з використанням математично чистих представлень просторових трансформацій через одиничні кватерніони та однорідні матриці, та високорівневим середовищем "Garry's Mod" "Lua", яке оперує традиційними для рушія "Source" концепціями сутностей, подієво-орієнтованої архітектури.

Ключові слова: *віртуальна реальність, математичне модулювання, геометричні моделі, OpenVR, 3D-візуалізація, конвеєр рендерингу, інтерактивність.*

DEVELOPMENT AND ANALYSIS OF MATHEMATICAL MODELS AND COMPUTATIONAL METHODS FOR SIMULATING PROCESSES IN VIRTUAL REALITY

Yurii-Marko Kulyk*, Postgraduate student, Lviv Polytechnic National University, Lviv, Ukraine. https://orcid.org/0009-0006-1121-3302, yurii-marko.r.kulyk@lpnu.ua*

Anatolii Batiuk*, PhD in Engineering, Associate Professor, Lviv Polytechnic National University, Lviv, Ukraine. https://orcid.org/0000-0001-7650-7383, anatolii.y.batiuk@lpnu.ua*

Abstract. *This paper presents a comprehensive analysis of mathematical models, computational methods, and visualization algorithms that form the fundamental basis for developing interactive virtual reality (VR) environments with a high degree of physical realism and visual fidelity. The purpose of the study is not only to systematize the theoretical foundations of three-dimensional geometric modeling, rigid-body physical simulation, and real-time rendering algorithms, but also to provide an in-depth critical analysis of their practical implementation and architectural constraints, combining the "Source" engine as realized through the "Garry's Mod" platform with the modern virtual reality application programming interface "OpenVR", running on a stationary workstation (personal computer) based on the "Windows 10" operating system.*

The central scientific and engineering challenge in this context lies not merely in applying well-known mathematical methods to VR tasks, but in adapting architectural solutions of modern VR systems to support stereoscopic rendering, spatial tracking, and interaction with objects and tools within the virtual environment, within the fundamentally different architecture of the "Source" engine.

The software solution described in this paper, currently under active development and refinement, functions as a complex multi-layered translation layer or software bridge that connects low-level VR hardware, including head-mounted displays with high-resolution integrated screens and tracking systems, as well as controllers communicating via the standardized "OpenVR" interface using mathematically pure representations of spatial transformations through unit quaternions and homogeneous matrices, with the high-level "Garry's Mod" "Lua" environment, which operates on the traditional "Source" engine concepts of entities and event-driven architecture.

Key words: *virtual reality, mathematical modeling, geometric models, OpenVR, 3D visualization, rendering pipeline, interactivity.*

Вступ

Робота структурована у логічній послідовності, яка послідовно веде від статичних, теоретичних математичних концепцій через проміжні етапи алгоритмічної обробки до практичних реалізацій інтерактивних механік віртуальної реальності.

Мета статті полягає в дослідженні математичних основ та алгоритмічних підходів до моделювання інтерактивних механік у віртуальній реальності, їх адаптація та застосування, а також розроблення методології інтеграції таких моделей у програмну архітектуру рушія.

Спочатку аналізуються математичні основи геометричного представлення тривимірних об'єктів та спеціалізовані структури даних, а також досліджуються різні системи представлення обертань, математичних властивостей кутів Ейлера та кватерніонів. Автори проводять поглиблений аналіз фізичної симуляції через підсистему рушія з детальним математичним описом рівнянь кінематики та динаміки твердого тіла. Особливу увагу приділено математичним алгоритмам детектування та розв'язання колізій (зіткнень) для запобігання проблеми тунелювання швидко рухомих об'єктів через тонкі геометричні бар'єри та ітераційного розв'язувача обмежень для обчислення коректних імпульсів реакції при зіткненнях з урахуванням тертя та коефіцієнта реституції.

Демонструється, і як всі теоретичні та алгоритмічні елементи взаємодіють через дворівневу архітектуру з бінарним модулем для низькорівневої взаємодії з "OpenVR" та високорівневими рішеннями мови програмування "Lua", при цьому детально розглядаються математичні моделі двох підходів до маніпулювання об'єктами у віртуальній реальності через кінематичне жорстке прикріплення з прямим заданням траєкторії або динамічне фізичне утримування, що моделюються диференціальним регулятором з коефіцієнтами.

У даній науковій роботі наукова новизна полягає в розробці математичної моделі та рішення для гібридної інтеграції віртуальної реальності у рушії з дворівневою архітектурою представлення через автоматичну трансляцію у бінарному модулі. А також математичної моделі та рішення для динамічного маніпулювання об'єктами, детектування колізій швидкорухомих об'єктів віртуальної реальності у системі з геометричним представленням у віртуальній реальності.

Практична цінність даного дослідження полягає не лише у систематизації математичного апарату віртуальної реальності, але й у демонстрації методології адаптації сучасних BP-технологій до існуючих програмних платформ через створення програмних мостів та шарів абстракції, що дає змогу розширити життєвий цикл класичних ігрових рушіїв та надати їм нові можливості без повного переписування базової архітектури. Описані підходи та математичні моделі можуть бути адаптовані для інтеграції віртуальної реальності у інші рушії аналогічного покоління або для створення сумісних шарів між різнорідними програмними системами, де необхідна трансляція між різними представленнями геометричних трансформацій, систем координат або парадигм фізичної симуляції.

Виконання досліджень

Геометричне представлення об'єктів у віртуальному просторі. Фундаментальною одиницею представлення геометрії у тривимірній комп'ютерній графіці є полігональна сітка, яка математично являє собою складну топологічну структуру, що складається з трьох взаємопов'язаних множин елементів. Перша множина містить вершини, які формально можна визначити як впорядковану послідовність векторів у тривимірному евклідовому просторі, де кожна вершина описується координатами:

$$v_i = \left(x_i, y_i, z_i \right) \in R^3,$$

при цьому індекс і приймає значення від 1 до *N*, де *N* позначає загальну кількість вершин у сітці. Друга множина представлена ребрами, які математично визначаються як невпорядковані пари вершин:

$$e_j = v_k, v_l,$$

де кожне ребро встановлює топологічний зв'язок між двома просторово розміщеними точками, формуючи граф зв'язності геометричного об'єкта. Третя множина складається з граней, які зазвичай представлені у формі трикутників або чотирикутників та математично описуються як впорядковані послідовності вершин:

$$f_m = \left(v_p, v_q, v_r \right),$$

що визначають орієнтовану поверхню у просторі через задання послідовності обходу вершин, яка впливає на обчислення нормалі до поверхні за допомогою векторного добутку [1–3].

Для коректного опису позиції, орієнтації та кінематичні характеристики руху об'єктів у тривимірному віртуальному просторі необхідне використання системи фундаментальних математичних інструментів, які формують алгебраїчний базис для всіх подальших обчислень у симуляції. Вектори представляють собою найбільш елементарні конструкти цього математичного апарату та використовуються для визначення як просторового розташування об'єктів відносно заданої системи координат, так і для представлення похідних кінематичних величин, таких як швидкість, прискорення та напрямок руху чи дії сил. У контексті програмного інтерфейсу на мові "Lua" вектори реалізовані як спеціалізований тип даних "Vector", який інкапсулює три дійсні числа та надає набір операцій для векторної алгебри, включаючи додавання векторів за правилом:

$$v_1 + v_2 = \left(x_1 + x_2, y_1 + y_2, z_1 + z_2\right),$$

скалярний добуток, що обчислюється як:

$$v_1 * v_2 = x_1x_2 + y_1y_2 + z_1z_2,$$

та задовольняє властивості комутативності, а також векторний добуток:

$$v_1xv_2 = \left(y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2\right),$$

який визначає вектор, перпендикулярний до площини, що натягується на вихідні вектори, з модулем, що дорівнює площі паралелограма, побудованого на цих векторах.

Кути Ейлера представляють собою історично сформований та інтуїтивно зрозумілий спосіб математичного опису тривимірного обертання через композицію трьох послідовних елементарних обертань навколо фіксованих або рухомих координатних осей [4], при цьому у найбільш поширеній авіаційній конвенції ці кути отримали назви тангаж (pitch), рискання (yaw) та крен (roll), що відповідають обертанням навколо поперечної, вертикальної та поздовжньої осей відповідно.

Математично, обертання на кут α навколо осі x описується матрицею $R_x(\alpha)$, яка у тривимірному просторі має вигляд матриці розміру три на три з елементами, де перший рядок містить значення $(1,0,0)$, другий рядок представлений як $(0,\cos\alpha,-\sin\alpha)$, а третій рядок має форму, при цьому аналогічні матриці обертання $R_y(\beta)$ та $R_z(\gamma)$ навколо осей у та z можуть бути записані з відповідною перестановкою позицій одиниць та тригонометричних функцій.

Результуюча матриця обертання для системи кутів Ейлера отримується множенням цих трьох елементарних матриць у певному порядку, наприклад:

$$R = R_z(\gamma)R_y(\beta)R_x(\alpha),$$

при цьому важливо зауважити, що матричне множення не є комутативною операцією, тобто порядок застосування обертань принципово впливає на кінцевий результат, що математично виражається нерівністю:

$$R_z(\gamma)R_y(\beta) \neq R_y(\beta)R_z(\gamma)$$

для більшості значень кутів.

Незважаючи на їхню інтуїтивність та широке використання у прикладних галузях, таких як навігація та авіоніка, кути Ейлера страждають від фундаментальної математичної проблеми, відомої як блокування обертання (gimbal lock) [5], яка виникає у специфічних конфігураціях орієнтації, коли два з трьох осей обертання виявляються вирівняними у паралельних або антипаралельних напрямках. Коли кут тангажу досягає значення $\beta = \pm 90$ градусів, що призводить до того, що матриця обертання $R_y(\pm\pi/2)$ трансформує вісь x у напрямок, паралельний або антипаралельний осі z, внаслідок чого подальші обертання навколо цих осей стають математично еквівалентними та призводять до втрати одного ступеня свободи обертання. Дана математична патологія робить кути Ейлера принципово непридатними для представлення довільних плавних обертань з шістьма ступенями свободи, які є абсолютно необхідними для коректної роботи систем віртуальної реальності, де контролери та шолом можуть приймати будь-які просторові орієнтації без обмежень.

Елегантним та математично обґрунтованим рішенням проблеми є використання кватерніонів [6; 7], які представляють собою розширення системи комплексних чисел до чотиривимірного алгебраїчного простору. Кватерніон складається з чотирьох дійсних компонент, які традиційно позначаються як:

$$q = w + xi + yj + zk,$$

де w є скалярною або дійсною частиною, а коефіцієнти x, y, z при уявних одиницях i, j, k формують векторну або уявну частину кватерніона. Уявні одиниці підпорядковуються фундаментальним алгебраїчним співвідношенням:

$$i^2 = j^2 = k^2 = ijk = -1,$$

з яких випливають властивості:

$$ij = k, jk = i, ki = j,$$

$$ji = -k, kj = -i, ik = -j,$$

що надає алгебрі кватерніонів некомутативну структуру.

Скалярна компонента одиничного кватерніона обертання визначається як:

$$w = \cos\left(\theta / 2\right),$$

тоді як векторна частина обчислюється через добуток вектора осі обертання на синус половинного кута у вигляді:

$$x = n_x \cdot \sin\left(\theta / 2\right),$$

$$y = n_y \cdot \sin\left(\theta / 2\right),$$

$$z = n_z \cdot \sin\left(\theta / 2\right).$$

Критичною перевагою кватерніонів є їхня здатність забезпечувати плавну та математично коректну інтерполяцію між двома довільними просторовими орієнтаціями, що реалізується через алгоритм сферичної лінійної інтерполяції, відомий як SLERP (Spherical Linear Interpolation). Геометрично, множина всіх одиничних кватерніонів утворює тривимірну сферу S^3 у чотиривимірному просторі, яка визначається умовою:

$$w^2 + x^2 + y^2 + z^2 = 1,$$

і будь-яке обертання представлене точкою на цій гіперсфері. Найкоротший шлях між двома точками на сфері є дугою великого кола, і SLERP забезпечує рівномірне переміщення вздовж цієї дуги з постійною кутовою швидкістю. Математична формула для SLERP між двома одиничними кватерніонами q_1 та q_2 з параметром інтерполяції $t \in [0,1]$ має вигляд

$$q(t) = \left(\sin\left((1-t)\Omega\right) / \sin\Omega\right) q + \left(\sin\left(t\Omega\right) / \sin\Omega\right) q_2,$$

де кут Ω між кватерніонами визначається через їхній скалярний добуток згідно з формулою:

$$\cos\Omega = q_1 * q_2 = w_1 w_2 + x_1 x_2 + y_1 y_2 + z_1 z_2.$$

Вона гарантує, що при $t = 0$ отримується початкова орієнтація q_1, при $t = 1$ досягається кінцева орієнтація q_2 а для проміжних значень параметра забезпечується плавний перехід з постійною кутовою швидкістю обертання, що є критично важливим для створення природних анімацій та відстеження руху у віртуальній реальності.

Фізична симуляція та математичні моделі динаміки об'єктів. Математичний опис руху об'єктів у тривимірному віртуальному просторі може бути здійснений двома фундаментально різними підходами, які відрізняються рівнем фізичної точності симуляції та обчислювальною складністю необхідних математичних операцій. Перший підхід, відомий як кінематика, представляє собою чисто геометричний опис траєкторії руху без урахування фізичних причин, які викликають цей рух, тобто без явного моделювання сил, мас, моментів інерції та інших динамічних характеристик фізичних тіл. Математично, такий підхід базується на інтегруванні рівнянь руху найпростішої форми, де нова позиція об'єкта обчислюється через пряме додавання вектора переміщення до поточної позиції за формулою:

$$p\left(t + \Delta t\right) = p\left(t\right) + v * \Delta t,$$

де $p\left(t\right)$ позначає вектор позиції об'єкта у момент часу t, v є вектором швидкості, який у простому кінематичному підході вважається постійним або заданим зовнішньою функцією без зв'язку з діючими силами, а Δt представляє часовий крок симуляції. Для обертального руху аналогічне кінематичне рівняння має вигляд:

$$\theta\left(t + \Delta t\right) = \theta\left(t\right) + \omega * \Delta t,$$

де θ представляє кутову орієнтацію, а ω є вектором кутової швидкості, при цьому для тривимірного випадку це рівняння має бути модифіковане для роботи з кватерніонами через формулу:

$$q\left(t + \Delta t\right) = q\left(t\right) + \left(1/2\right) * \Delta t * \omega_{quat} * q\left(t\right),$$

де $\omega_{quat} = \left(0, \omega_x, \omega_y, \omega_z\right)$ є чистим уявним кватерніоном, побудованим з вектора кутової швидкості, а множення є кватерніонним добутком.

Якщо кінематично керований об'єкт рухається зі швидкістю v у напрямку твердої стіни, то наївне застосування кінематичного рівняння руху призведе до того, що нова позиція виявиться за межами стіни у фізично неможливій конфігурації, що математично означає порушення умов непроникності твердих тіл. Альтернативно, якщо система детектування колізій виявить зіткнення та примусово зупинить об'єкт, встановивши його швидкість миттєво у нуль незалежно від значення $v(t)$.

Другий підхід до опису руху, відомий як динаміка твердого тіла, представляє собою повноцінну математичну симуляцію фізичних процесів з урахуванням фундаментальних законів класичної механіки, сформульованих Ісааком Ньютоном у сімнадцятому столітті та розвинутих наступними поколіннями вчених. У даному підході об'єкти характеризуються набором фізичних властивостей, таких як маса m, яка визначає інерцію поступального руху, тензор інерції I, який є симетричною матрицею розміру три на три та визначає опір обертальному руху відносно різних осей, коефіцієнти тертя μ_s та μ_k для статичного та кінетичного тертя відповідно, а також коефіцієнт пружності або реституції $e \in [0,1]$, який визначає збереження кінетичної енергії при зіткненнях. Рух таких об'єктів визначається не довільним заданням швидкостей, а інтегруванням рівнянь руху, які випливають з другого закону Ньютона у формі:

$$F = ma,$$

для поступального руху, та рівняння Ейлера для обертального руху:

$$\tau = I\alpha + \omega \times (I\omega),$$

де F позначає сумарну силу, що діє на тіло, т є сумарним моментом сил або крутним моментом, $\alpha = d\omega / dt$ представляє кутове прискорення, а доданок $\omega \times (I\omega)$ враховує гіроскопічний ефект, що виникає через обертання самого тензора інерції разом з тілом [9].

Критичною компонентою динамічної симуляції є обробка зіткнень між твердими тілами, яка включає дві фундаментально різні підзадачі: детектування колізій та розв'язання колізій. Детектування колізій є суто геометричною задачею визначення, чи перетинаються два або більше тіл у просторі, і для складних геометрій може бути обчислювально дуже затратним процесом з наївною складністю $O(n^2)$ для n об'єктів, оскільки потенційно кожна пара об'єктів повинна бути перевірена на зіткнення [10]. Оптимізація досягається через ієрархічні просторові структури даних, такі як дерева обмежувальних об'ємів (BVH), октодерева або просторовий хешинг, які дають можливість швидко виключити з розгляду пари об'єктів, які гарантовано не можуть перетинатися через їхнє розташування у віддалених областях простору, зменшуючи середню складність до $O(nlogn)$ або навіть $O(n)$ у оптимальних випадках з рівномірним розподілом об'єктів. Після виявлення зіткнення необхідно обчислити точку контакту $p_{contact}$, нормаль зіткнення $n_{contact}$, яка спрямована від одного об'єкта до іншого, та глибину проникнення $d_{penetration}$, яка вимірює, наскільки два об'єкти перекриваються у просторі внаслідок дискретного часового кроку симуляції.

Для інтерактивного середовища віртуальної реальності кінематичний підхід використовується виключно для об'єктів, які повністю контролюються користувачем та не повинні підкорятися звичайним фізичним законам, таких як віртуальні представлення рук гравця або об'єкти, які він утримує у «мертвому» захоплені з фіксованим зв'язком між контролером та об'єктом. У таких випадках позиція та орієнтація об'єктів безпосередньо встановлюються відповідно до відстежуваних поз апаратних пристроїв через кінематичні рівняння без фізичних обчислень, що забезпечує мінімальну затримку та точну відповідність між реальними рухами користувача та віртуальним представленням.

Інтеграція віртуальної реальності у рушій. "OpenVR", також відомий під комерційною назвою SteamVR у контексті дистрибуційної платформи компанії Valve Corporation, представляє собою агностичний відносно виробника обладнання програмний інтерфейс додатків, який стандартизує та уніфікує процес комунікації між різноманітним апаратним забезпеченням віртуальної реальності, що включає шоломи з вбудованими дисплеями та системами відстеження, ручні контролери з кнопками, тригерами, тачпадами та системами тактильного зворотного зв'язку через вібромотори, а також базові станції або камери зовнішнього відстеження, та програмними додатками, які бажають використовувати можливості віртуальної реальності [11].

Фундаментальною функцією "OpenVR" є надання точних даних про просторове відстеження пристроїв через повернення матриць афінних перетворень або еквівалентних структур поз, які містять інформацію про позицію та орієнтацію кожного відстежуваного пристрою у глобальній системі координат віртуального простору.

Розроблене програмне рішення представляє собою програмний міст між високорівневим скриптовим середовищем "Garry's Mod" та низькорівневим програмним інтерфейсом "OpenVR", при цьому архітектура цього рішення базується на дворівневій структурі з чітким розділенням відповідальності між компонентами різних рівнів абстракції.

Перший компонент архітектури складається з набору скриптів, написаних мовою "Lua", вбудованій у "Garry's Mod", та забезпечують високорівневе середовище виконання для ігрової логіки з автоматичним управлінням пам'яттю через збирач сміття та динамічною типізацією. Компоненти

рішення виконують множину критичних функцій, серед яких надання структурованого програмного інтерфейсу для інших розробників модифікацій через експортовані функції та хуки, які дають змогу стороннім скриптам отримувати доступ до даних віртуальної реальності та реагувати на події ВР-взаємодій [12].

Другий компонент є бінарним модулем у формі динамічної бібліотеки з розширенням dll для операційних систем Windows, який представляє собою фактичне серце функціональності рішення та забезпечує низькорівневу інтеграцію з рушієм “Source” [13]. Він завантажується процесом “Garry’s Mod” під час ініціалізації через стандартний механізм завантаження нативних модулів “Lua”, який дає змогу розширити функціональність через функції, написані мовами системного програмування C++ з прямим доступом до системних викликів операційної системи та можливістю виконання обчислень без накладних витрат інтерпретації.

Критичною функцією бінарного модуля є «ін’єкція» можливостей віртуальної реальності безпосередньо у конвеєр рендерингу рушія через перехоплення викликів графічного програмного інтерфейсу та модифікацію поведінки камери. Математично, стандартний рендеринг використовує одну камеру з єдиною матрицею проекції P_{mono} та матрицею виду V_{mono}, які визначають перетворення з світових координат у координати відсікання для монокулярного відображення на плоскому екрані.

Для стереоскопічного рендерингу віртуальної реальності необхідно генерувати два окремі зображення з дещо різних точок спостереження, що відповідають позиціям лівого та правого ока користувача, при цьому ці позиції зміщені відносно центральної точки шолома на відстань, що дорівнює половині міжзіничної відстані IPD (Inter-Pupillary Distance), яка є індивідуальним антропометричним параметром з типовими значеннями у діапазоні від п’ятдесяти п’яти до семидесяти п’яти міліметрів для дорослої людини [14].

Математична модель стереоскопічного рендерингу передбачає обчислення двох матриць виду V_l та V_r , які отримуються модифікацією базової матриці виду HMD через застосування трансляцій вздовж міжокулярної осі згідно з формулами:

$$V_l = T\left(-IPD/2,0,0\right) * V_{HMD},$$

$$V_r = T\left(+IPD/2,0,0\right) * V_{HMD},$$

де $T\left(\Delta x, \Delta y, \Delta z\right)$ позначає матрицю трансляції, а матриця V_{HMD} визначається інверсією пози шолома, через співвідношення $V_{HMD} = (T_{HMD})^{(-1)}$ [15; 16].

Реалізація інтерактивності: маніпулювання об’єктами. Математичні моделі руху та фізичної симуляції, трансформуються у інтерактивний досвід віртуальної реальності через реалізацію механік маніпулювання об’єктами, які дають можливість користувачеві природно взаємодіяти з віртуальним середовищем через захоплення, переміщення, обертання та кидання фізичних об’єктів за допомогою віртуальних представлень його рук.

Воно базується на повному ігноруванні фізичних властивостей об’єкта під час його утримування у руці та використанні чисто геометричних перетворень для синхронізації позиції та орієнтації об’єкта з руками контролера. На кожному кадрі оновлення, що відбувається з частотою рендерингу клієнта, типово дев’яносто кадрів за секунду або вище, спеціальний обробник у хуку Think виконує примусове оновлення трансформації утримуваного об’єкта через пряме присвоєння його позиції та орієнтації відповідним значенням від сутності руки згідно з операціями “prop:SetPos(hand:GetPos() + hand:GetForward() * holdOffset)” та “prop:SetAngles(hand:GetAngles() + angleOffset)», де параметри “holdOffset” та “angleOffset” визначають відносне зміщення об’єкта відносно контролера для забезпечення природного розташування у віртуальній руці, що математично описується локальним перетворенням:

$$T_{props} = T_{hand} * T_{local},$$

з матрицею локального зміщення:

$$T_{local} = \left[\left[R_{offset}, t_{offset}\right],\left[0,1\right]\right],$$

яка залишається постійною протягом всього процесу утримування.

Механіка кидання об’єкта у кінематичному підході реалізується через детектування події відпускання кнопки захоплення, після чого виконується повторне увімкнення фізичної симуляції, що повертає об’єкт у множину динамічно симульованих тіл, та застосування одномоментного імпульсу, який визначається поточною швидкістю руху контролера згідно з “phys:SetVelocity(hand:GetVelocity() * throwMultiplier)”, де швидкість контролера може бути отримана через прямий запит до “OpenVR” для отримання даних про швидкість з вбудованих датчиків інерціальної навігації контролера, які використовують інтегрування даних акселерометра та гіроскопа для оцінки лінійної та кутової швидкостей з високою точністю.

Для реалізації більш просунутих механік маніпулювання, таких як обертання утримуваного об'єкта навколо довільних осей або точне позиціонування з контролем орієнтації, математична модель повинна бути розширена для включення обертальної динаміки через застосування крутних моментів. Використовується метод "PhysObj:ApplyForceOffset(J, p_application)", який застосовує імпульс J у точці з світовими координатами $p_{application}$ на поверхні або всередині об'єкта. Математично, цей метод одночасно модифікує як лінійний імпульс:

$$p_{linear} \leftarrow p_{linear} + J,$$

так і кутовий імпульс:

$$L_{angular} \leftarrow L_{angular} + r * J,$$

де вектор $r = p_{application} - p_{COM}$ є вектором від центру мас до точки застосування сили, а операція векторного добутку $r * J$ обчислює крутний момент або момент сили відносно центру мас згідно з фундаментальним визначенням $\tau = r * F$.Ця можливість дає змогу реалізувати складну механіку обертання утримуваного об'єкта через аналіз положення пальців користувача на тачпаді контролера або кута нахилу контролера для визначення бажаної осі обертання «axis_rotation» та кутової швидкості $\omega_{desired}$, після чого обчислюються точки застосування сил на протилежних сторонах об'єкта відносно осі обертання:

$$p_1 = p_{COM} + r_{perp},$$

$$p_2 = p_{COM} - r_{perp},$$

де r_{perp} є вектором, перпендикулярним до $axis_{rotation}$ з величиною, що відповідає характерному розміру об'єкта, і застосовуються протилежно спрямовані імпульси:

$$J_1 = k_{rot} * \omega_{desired} * r_{perp},$$

$$J_2 = -k_{rot} * \omega_{desired} * r_{perp}$$

у цих точках, що створює пару сил, який прагне розкрутити об'єкт до бажаної кутової швидкості з можливістю застосування демпфування через доданок пропорційний різниці $\omega_{desired} - \omega_{current}$ для запобігання надмірного розкручування.

Математична основа даних механік інтерактивності демонструє елегантну інтеграцію високорівневих абстракцій скриптової мови з низькорівневою фізичною симуляцією, для оперування інтуїтивними концепціями векторів позицій, кутів обертань та сил.

На прикладі нижче (рис. 1), наведений знімок екрану, а саме демонстрація роботи програмного рішення, в реальному часі, взаємодія віртуальних рук з 3D-об'єктом (що контролюються ВР контролерами, синхронізуються з руками користувача в реальному світі), а саме зміну його положення (підняття) та легке обертання відносно точки дотику.

Рис. 1. Приклад роботи рішення з віртуальним об'єктом 3D-сцени

Висновок

Проведене дослідження і реалізація математичних моделей та обчислювальних методів для симуляції процесів у віртуальній реальності на прикладі інтеграції BP-функціональності демонструє складну багаторівневу архітектуру, де фундаментальні математичні концепції класичної механіки, лінійної алгебри та обчислювальної геометрії трансформуються у інтерактивний досвід через послідовність абстракцій та оптимізацій на різних рівнях програмного забезпечення.

Математична формалізація полігональних сіток через множини вершин, ребер та граней у тривимірному евклідовому просторі створює основу для всіх подальших обчислень трансформацій та відображення. Дослідження математичних представлень обертань виявило фундаментальну проблему блокування у системі кутів Ейлера, яка робить це інтуїтивне представлення непридатним для довільних шестиступеневих обертань контролерів віртуальної реальності, та продемонструвало елегантне рішення через використання кватерніонів, які забезпечують вільне від сингулярностей представлення обертань у чотиривимірному просторі з компонентами, що обчислюються через половинний кут обертання та одиничний вектор осі.

Фізична симуляція у системі представлена двома принципово різними математичними підходами до опису руху об'єктів, де кінематичний підхід базується на чисто геометричних рівняннях руху без урахування сил та мас через пряме інтегрування швидкостей для отримання позицій, тоді як повноцінна динамічна симуляція через підсистему "VPhysics" реалізує чисельне інтегрування рівнянь Ньютона-Ейлера з урахуванням всіх діючих сил, моментів інерції та реакцій зіткнень.

Інтеграція віртуальної реальності у рушій "Source" через архітектуру рішення представляє собою елегантне рішення проблеми зв'язування низькорівневого програмного інтерфейсу "OpenVR" з високорівневим середовищем "Lua" через дворівневу структуру з бінарним модулем для критичних за продуктивністю операцій матричних перетворень та трансляції між системами координат та скриптовими компонентами для гнучкої реалізації ігрової логіки. Математичний аналіз перетворень між системою координат "OpenVR" та "Source" через матриці обертання та трансляції з урахуванням різної орієнтації вертикальних осей в реальному часі демонструє важливість узгодженості між фізичними рухами користувача (контролерів) та їхнім віртуальним представленням (віртуальних моделей рук), при цьому підхід до представлення контролерів як звичайних ігрових сутностей значно спрощує розробку через використання стандартних методів отримання позицій та орієнтацій без необхідності роботи з специфічними структурами даних віртуальної реальності.

Реалізація інтерактивності об'єктів через механіки маніпулювання об'єктами демонструє практичне застосування математичних моделей кінематики та динаміки для створення природних та інтуїтивних взаємодій у віртуальній реальності.

Можна стверджувати, що успішна реалізація віртуальної реальності у рушії через розроблене (но не завершене, оскільки потребує покращень) базується на глибокій інтеграції математичних моделей геометрії, кінематики, динаміки та графічного рендерингу через багаторівневу архітектуру з чітким розділенням відповідальності між компонентами різних рівнів абстракції. Математичний фундамент системи включає векторну та матричну алгебру для представлення позицій та трансформацій, кватерніонну алгебру для коректного опису обертань без сингулярностей, диференціальні рівняння класичної механіки для фізичної симуляції руху під дією сил та геометричні алгоритми для ефективного детектування та розв'язання зіткнень між складними тривимірними об'єктами.

Перспективи подальших досліджень включають розширення аналізу на додаткові аспекти симуляції віртуальної реальності, такі як рішення тактильного зворотного зв'язку через вібромотори з оптимізацією параметрів частоти та амплітуди для передачі різних тактильних відчуттів, алгоритми прогнозування руху для компенсації затримок між рухами користувача та оновленням відображення через екстраполяцію траєкторій за допомогою фільтрів Калмана або нейронних мереж, методи оптимізації продуктивності рендерингу специфічно для віртуальної реальності з адаптивною роздільністю залежно від напрямку погляду для підтримання частоти кадрів при падінні продуктивності.

Література

1. Botsch M., Pauly M., Kobbelt L., Alliez P., Levy B., Bischoff S., Roossl C. Geometric modeling based on polygonal meshes. *Association for Computing Machinery*. 2007. DOI: https://doi.org/10.1145/1281500.1281640.

2. Heckbert P. S., Garland M. Survey of polygonal surface simplification algorithms. *Carnegie Mellon University*. 1997.

3. Botsch M., Kobbelt L., Pauly M., Alliez P., Levy B. Polygon mesh processing. Boca Raton: *CRC Press.* 2010. 261 c.

4. Diebel J. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*. 2006. Vol. 58. No. 15–16. P. 1–35.

5. Hemingway E. G., O'Reilly O. M. Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody System Dynamics*. 2018. Vol. 44. No. 1. P. 31–56. DOI: https://doi.org/10.1007/s11044-018-9620-0.

6. Voight J. Quaternion algebras. *Springer Nature*, 2021. 885 p.

7. Mukundan R. Quaternions. In: Advanced methods in computer graphics: with examples in OpenGL. *Springer London*. 2012. P. 77–112. DOI: https://doi.org/10.1007/978-1-4471-2340-8_5.

8. Facepunch Studios. Entity. *Garry's Mod Developer Wiki*. 2025. URL: https://wiki.facepunch.com/gmod/ENTITY

9. Goldstein H., Poole C. P., Safko J. L. *Classical mechanics*. 2002. 3rd ed (ch. 1, 2, 7).

10. Hubbard P. M. Collision detection for interactive graphics applications. *Transactions on Visualization and Computer Graphics*. 2002. IEEE. Vol. 1. No. 3. P. 218–230. DOI: https://doi.org/10.1109/2945.466717.

11. Nassi M. Introduction to OpenVR 101 Series: What is OpenVR and how to get started with its APIs. *The Ghost Howls*. 2018. URL: https://skarredghost.com/2018/03/15/introduction-to-openvr-101-series-what-is-openvr-and-how-to-get-started-with-its-apis

12. Ierusalimschy R., De Figueiredo L. H., Filho W. C. Lua – an extensible extension language. S*oftware: Practice and Experience*. 1996. Vol. 26. No. 6. P. 635–652. DOI: https://doi.org/10.1002/(SICI)1097-024X(199606)26:6<635::AID-SPE26>3.0.CO;2-P.

13. Valve Software. OpenVR. *GitHub*. 2025. URL: https://github.com/ValveSoftware/openvr/tree/master/bin/win64

14. Moradi A., Haghparast A., Aghajani F., Shojaei A., Khoee S. H., Noorizadeh F., Vafaei F. Comparison of three methods of measuring the inter-pupillary distance. *Ophthalmic and Optometric Sciences*. 2022. Vol. 6. No. 3. P. 7–14. DOI: https://doi.org/10.22037/joos.v6i3.43349.

15. Hast A. 3D stereoscopic rendering: An overview of implementation issues. *Game Engine Gems*. Vol. 1. 2010. P. 123–138.

16. Blonde L., Doyen D., Borel T. 3D stereo rendering challenges and techniques. *44th Annual Conference on Information Sciences and Systems (CISS)*, 2010, March. IEEE, 2010. P. 1–6. DOI: https://doi.org/10.1109/CISS.2010.5464936.

References

1. Botsch M., Pauly M., Kobbelt L., Alliez P., Levy B., Bischoff S., Roossl C. (2007). Geometric modeling based on polygonal meshes. *Association for Computing Machinery*. https://doi.org/10.1145/1281500.1281640

2. Heckbert, P. S., & Garland, M. (1997). Survey of polygonal surface simplification algorithms. *Carnegie Mellon University*.

3. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., & Levy, B. (2010). Polygon mesh processing. *CRC press*.

4. Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*. 58 (15–16). 1–35.

5. Hemingway, E. G., & O'Reilly, O. M. (2018). Perspectives on Euler angle singularities, gimbal lock, and the orthogonality of applied forces and applied moments. *Multibody system dynamics*. 44 (1). 31–56. DOI: https://doi.org/10.1007/s11044-018-9620-0.

6. Voight, J. (2021). Quaternion algebras (p. 885). *Springer Nature*.

7. Mukundan, R. (2012). Quaternions. In Advanced methods in computer graphics: with examples in OpenGL (pp. 77–112). *Springer London*. DOI: https://doi.org/10.1007/978-1-4471-2340-8_5.

8. Facepunch Studios. (2025). Entity. *Garry's Mod Developer Wiki*. URL: https://wiki.facepunch.com/gmod/ENTITY

9. Goldstein, H., Poole, C. P., & Safko, J. L. (2002). *Classical mechanics,* third edition, (ch. 1,2,7)

10. Hubbard, P. M. (2002). Collision detection for interactive graphics applications. IEEE *Transactions on Visualization and Computer Graphics*. 1 (3). 218–230. DOI: https://doi.org/10.1109/2945.466717

11. Nassi M. (2018). Introduction to OpenVR 101 Series: What is OpenVR and how to get started with its APIs. *The Ghost Hownls*. URL: https://skarredghost.com/2018/03/15/introduction-to-openvr-101-series-what-is-openvr-and-how-to-get-started-with-its-apis

12. Ierusalimschy R., De Figueiredo, L. H., & Filho, W. C. (1996). Lua-an extensible extension language. *Software: Practice and Experience*, 26(6), 635-652. DOI: https://doi.org/10.1002/(SICI)1097-024X(199606)26:6%3C635::AID-SPE26%3E3.0.CO;2-P.

13. Valve Sofrware. (2025). OpenVR. *GitHub*. https://github.com/ValveSoftware/openvr/tree/master/bin/win64

14. Moradi, A., Haghparast, A., Aghajani, F., Shojaei, A., Khoee, S. H., Noorizadeh, F., & Vafaei, F. (2022). Comparison of Three Methods of Measuring the Inter-Pupillary Distance. *Ophthalmic and Optometric Sciences*. 6 (3). 7–14. DOI: https://doi.org/10.22037/joos.v6i3.43349.

15. Hast, A. (2010). 3D Stereoscopic Rendering: An Overview of Implementation Issues. *Game Engine Gems*. 1. 123–138.

16. Blonde, L., Doyen, D., & Borel, T. (2010, March). 3D stereo rendering challenges and techniques. In *44th Annual Conference on Information Sciences and Systems (CISS).* Pp. 1–6. IEEE. DOI: https://doi.org/10.1109/CISS.2010.5464936.

Дата першого надходження статті до видання: 21.10.2025
Дата прийняття статті до друку після рецензування: 21.11.2025
Дата публікації (оприлюднення) статті: 26.12.2025