УДК 004.4:004.92 https://doi.org/10.31713/vt220267

Гуменний П.В. *[1, ORCID: 0000-0003-0982-3305]*,
к.т.н., доцент,

[1]Західноукраїнський національний університет, м. Тернопіль

МЕТОДИ ПРОСТОРОВОГО МОДЕЛЮВАННЯ ТА 3D-ВІЗУАЛІЗАЦІЇ У СУЧАСНИХ ОПЕРАЦІЙНИХ СИСТЕМАХ

У статті досліджено методи тривимірної (3D) візуалізації даних систем автоматичного керування засобами графічних підсистем сучасних операційних систем. Розглянуто математичний апарат геометричних перетворень, перспективного проєктування та моделей освітлення (зокрема модель Фонга), які лежать в основі графічного конвеєра під час відображення керуючої інформації в інтерфейсах SCADA/HMI. Здійснено комп'ютерне моделювання динаміки типового лінійного об'єкта керування третього порядку, заданого рівняннями простору станів, та виконано відображення перехідного процесу у вигляді фазової траєкторії в тривимірному просторі станів; визначено усталене значення та оцінено тривалість перехідного процесу. Проведено обчислювальний експеримент щодо продуктивності програмної (CPU) та апаратної (GPU-орієнтованої) реалізації 3D-рендерингу: встановлено залежність частоти кадрів від кількості полігонів сцени та показано, що для сцен середньої та високої складності апаратна реалізація забезпечує приблизно на порядок вищу продуктивність порівняно з програмною растеризацією (зокрема, при мільйоні полігонів — близько 714 кадрів за секунду проти 71). Досліджено ефективну пропускну здатність відеопам'яті типів GDDR6 та HBM2 залежно від обсягу переданих графічних даних, а також розподіл часу обробки кадру за етапами графічного конвеєра для різних рівнів деталізації сцени; показано, що етап шейдингу та освітлення стає домінуючою складовою при високій деталізації. Отримані результати можуть бути використані під час проєктування підсистем людино-машинного інтерфейсу SCADA/HMI з підтримкою 3D-візуалізації в операційних системах загального та реального часу, а також під час вибору апаратної платформи для систем

 ISSN 2306-5478

людино-машинної взаємодії з підвищеними вимогами до наочності представлення інформації.

Ключові слова: 3D-візуалізація; графічний конвеєр; операційна система; GPU; моделювання систем керування; фазова траєкторія; продуктивність рендерингу; SCADA; HMI.

Актуальність дослідження. Сучасний розвиток інформаційних технологій, високопродуктивних обчислень та інженерного програмного забезпечення супроводжується зростанням складності систем автоматичного керування (САК), що вимагає застосування ефективних методів математичного моделювання, аналізу та візуального представлення динамічних процесів. У сучасних умовах проектування складних технічних систем дедалі більшого значення набуває інтеграція чисельних методів із засобами комп'ютерної графіки, що дозволяє не лише підвищити точність досліджень, але й забезпечити наочну інтерпретацію результатів моделювання.

Одним із перспективних напрямів розвитку інженерного аналізу є використання засобів тривимірної візуалізації для представлення фазових траєкторій, просторових моделей станів та результатів числових експериментів. Використання 3D-візуалізації дозволяє підвищити інформативність аналізу динамічних систем, спростити виявлення нелінійних ефектів, оцінювання стійкості та дослідження поведінки систем у різних режимах функціонування.

Водночас ефективність реалізації процесів моделювання та візуалізації суттєво залежить від архітектури сучасних операційних систем, особливостей взаємодії із графічними API та механізмів використання обчислювальних ресурсів центрального й графічного процесорів. Розвиток таких технологій, як Vulkan, DirectX, Metal, а також інструментальних середовищ MATLAB/Simulink і Python/SciPy, створює нові можливості для побудови високопродуктивних програмних комплексів аналізу та верифікації систем керування.

У зв'язку з цим актуальним є дослідження методів моделювання, аналізу та 3D-візуалізації систем автоматичного керування у сучасних операційних системах, а також оцінювання впливу програмно-апаратної платформи на продуктивність обчислень і якість графічного представлення результатів моделювання.

Метою роботи є аналіз методів 3D-візуалізації та моделювання систем керування в сучасних операційних системах, формалізація відповідного математичного апарату (геометричні перетворення,

проєктування, моделі освітлення, показники продуктивності) та проведення комп'ютерного експерименту, що демонструє застосування цих методів для відображення динаміки об'єкта керування й оцінювання продуктивності графічного конвеєра.

Огляд сучасних досліджень. Проблематика просторового моделювання та 3D-візуалізації останніми роками набула особливої актуальності у зв'язку зі стрімким розвитком графічних процесорів, сучасних графічних API, засобів математичного моделювання та операційних систем, що забезпечують підтримку високопродуктивного рендерингу. Аналіз сучасних наукових праць показує, що основні дослідження спрямовані на вдосконалення методів побудови тривимірних сцен, оптимізацію графічного конвеєра, прискорення обчислень за допомогою GPU та інтеграцію засобів візуалізації з програмними середовищами моделювання.

У фундаментальній праці Hughes та інших [1] систематизовано математичні основи сучасної комп'ютерної графіки, розглянуто методи геометричного моделювання, афінні та проєктивні перетворення, побудову графічного конвеєра, алгоритми освітлення та растеризації. Особливу увагу автори приділяють математичному апарату формування тривимірних моделей і взаємодії між центральним та графічним процесорами. Викладені підходи є теоретичною основою для побудови сучасних систем 3D-візуалізації, що використовуються під час моделювання систем автоматичного керування. У представленій статті ці положення розвинено шляхом застосування математичного апарату до просторового відображення результатів моделювання динамічних систем та аналізу продуктивності їх реалізації в сучасних операційних системах.

Подальший розвиток технологій реального часу висвітлено у роботі Akenine-Möller, Haines та Hoffman [2], яка присвячена сучасним алгоритмам GPU-рендерингу. Автори детально досліджують архітектуру графічного конвеєра, програмовані шейдери, методи оптимізації продуктивності, технології OpenGL, Vulkan і DirectX, а також особливості побудови інтерактивних графічних застосунків. Особливе місце відведено питанням зменшення часу формування кадру та ефективного використання ресурсів графічного процесора. На відміну від цієї роботи, у запропонованому дослідженні розглядається використання зазначених технологій саме для відображення результатів математичного моделювання систем керування, а також

ISSN 2306-5478

аналізується вплив операційної системи на швидкодію графічного конвеєра.

У монографії Marschner та Shirley [3] узагальнено сучасні методи математичного моделювання тривимірних об'єктів, алгоритми побудови полігональних моделей, методи освітлення, текстурування та формування реалістичних сцен. Значну увагу приділено просторовим перетворенням, математичним моделям камер та сучасним методам побудови віртуального середовища. Отримані результати є базою для реалізації високоякісної тривимірної візуалізації, тоді як у даній роботі ці методи використано для побудови інтерактивного представлення фазових траєкторій та інших результатів моделювання систем автоматичного керування.

Серед сучасних досліджень особливий інтерес становить робота Allison та співавт. [4], у якій представлено графічний рушій FIRE-3DV, побудований на основі API Vulkan. Автори запропонували платформонезалежну архітектуру високопродуктивного рендерингу, що забезпечує ефективне використання багатоядерних процесорів і сучасних GPU при моделюванні складних тривимірних сцен. Робота демонструє можливість створення універсального програмного забезпечення для інтерактивного моделювання робототехнічних систем та цифрових двійників. На відміну від неї, у даному дослідженні увага акцентується не лише на архітектурі графічного рушія, а й на інтеграції математичного моделювання динамічних систем із засобами 3D-візуалізації сучасних операційних систем.

У статті Boutsi, Ioannidis та співавт. [5] досліджено використання багатопотокового рендерингу Vulkan для побудови великомасштабних тривимірних моделей геопросторових об'єктів. Запропоновані методи дозволяють ефективно розподіляти навантаження між потоками центрального процесора та графічним процесором, забезпечуючи суттєве підвищення продуктивності систем візуалізації. У даній роботі аналогічний підхід розглянуто в контексті побудови 3D-візуалізації результатів моделювання систем керування та оцінювання впливу архітектури операційної системи на продуктивність графічного конвеєра.

Значним досягненням сучасної комп'ютерної графіки є робота Kerbl та співавт. [6], у якій запропоновано метод 3D Gaussian Splatting для високоякісного рендерингу сцен у реальному часі. Автори показали, що використання гаусових примітивів дозволяє досягти значно вищої швидкодії порівняно з традиційними підходами NeRF при збереженні високої якості візуалізації.

Незважаючи на те, що ця технологія орієнтована переважно на фотореалістичний рендеринг, її результати підтверджують сучасні тенденції розвитку методів просторової візуалізації, які можуть бути використані і в інженерному моделюванні.

Робота Kato та співавт. [7] є одним із найповніших оглядів сучасних методів диференційованого рендерингу. Авторами проаналізовано математичні моделі побудови зображення, алгоритми оптимізації, використання нейронних мереж та методів машинного навчання для відновлення тривимірної геометрії. На відміну від досліджень, орієнтованих на комп'ютерний зір, у представленій статті розглядаються класичні методи математичного моделювання та просторової візуалізації технічних систем без використання складних алгоритмів штучного інтелекту.

Офіційне керівництво CUDA C++ Programming Guide [8] містить сучасні методи організації паралельних обчислень на GPU, архітектуру пам'яті графічного процесора та способи оптимізації продуктивності програм. Представлені рекомендації широко використовуються при створенні високопродуктивних систем моделювання та візуалізації. У даній роботі ці принципи враховано під час аналізу ефективності апаратного рендерингу та оцінювання переваг використання графічного процесора порівняно із програмною реалізацією.

Офіційна специфікація Vulkan API [9] визначає сучасну низькорівневу модель взаємодії операційної системи з графічним процесором. Документ описує механізми керування пам'яттю, командними буферами, синхронізацією та багатопотоковим виконанням графічних команд. Саме ці особливості забезпечують можливість створення високопродуктивних систем тривимірної візуалізації, що використано при аналізі сучасних графічних підсистем операційних систем у представленій роботі.

Специфікація OpenGL 4.6 [10] залишається одним із базових міжнародних стандартів програмування графічних застосунків. Документ регламентує побудову графічного конвеєра, роботу із шейдерами, буферами, текстурами та механізмами взаємодії програмного забезпечення з графічним адаптером. У статті ці принципи застосовано для пояснення процесу формування тривимірних сцен під час моделювання динамічних систем.

Документація MATLAB Graphics [11] демонструє сучасні засоби побудови дво- та тривимірних графіків, анімації, інтерактивних

 ISSN 2306-5478

поверхонь і просторових моделей, що широко використовуються у задачах аналізу систем автоматичного керування. Аналогічні підходи використано й у представленому дослідженні для візуалізації фазових траєкторій та результатів математичного моделювання.

У документації SciPy [12] наведено сучасні алгоритми чисельного інтегрування звичайних диференціальних рівнянь, оптимізації, математичного аналізу та моделювання складних технічних систем. Саме ці бібліотеки забезпечують основу реалізації комп'ютерного експерименту, проведеного у даній статті.

Документація Matplotlib mplot3d [13] описує інструментальні засоби побудови тривимірних графіків, поверхонь, фазових портретів та просторових траєкторій засобами Python. Використання цього інструментарію є безпосередньо співзвучним із запропонованим у статті підходом до побудови просторової візуалізації результатів моделювання.

У сучасній роботі Carter, Hitschfeld та Navarro [14] представлено бібліотеку Mìmir, призначену для інтерактивної візуалізації програм CUDA у режимі реального часу. Автори дослідили засоби візуального аналізу виконання паралельних алгоритмів, що дозволяють оцінювати ефективність використання GPU під час складних обчислень. Отримані результати підтверджують зростаючу роль графічних процесорів не лише як засобів формування зображення, а й як потужної обчислювальної платформи для задач математичного моделювання.

Проведений аналіз літератури свідчить, що сучасні дослідження переважно зосереджені на окремих аспектах комп'ютерної графіки: побудові графічних конвеєрів, розробленні нових алгоритмів рендерингу, оптимізації використання GPU або створенні спеціалізованих засобів тривимірної візуалізації. Водночас у наявних роботах недостатньо висвітлено комплексне поєднання математичного моделювання систем автоматичного керування, просторової 3D-візуалізації результатів моделювання та аналізу впливу сучасних операційних систем на продуктивність графічного конвеєра.

Наукова цінність представленої публікації полягає у комплексному дослідженні взаємозв'язку між методами математичного моделювання динамічних систем, сучасними технологіями тривимірної візуалізації та архітектурою графічних підсистем операційних систем. На відміну від існуючих робіт, стаття

поєднує теоретичний апарат геометричних перетворень і моделей освітлення з практичним комп'ютерним експериментом, у якому досліджено моделювання динаміки об'єкта керування, просторове представлення фазових траєкторій, а також проведено аналіз продуктивності CPU- та GPU-орієнтованого рендерингу, ефективності використання відеопам'яті та впливу етапів графічного конвеєра на швидкодію сучасних систем візуалізації. Такий комплексний підхід дозволяє обґрунтувати вибір програмно-апаратної платформи для створення високопродуктивних SCADA/HMI-систем і засобів інженерного моделювання нового покоління.

Виклад основного матеріалу. Базовим елементом будь-якої системи 3D-візуалізації є представлення геометрії об'єктів у вигляді сукупності вершин у тривимірному просторі та послідовність афінних перетворень, що переводять координати з локальної (модельної) системи у систему координат екрана. Графічна підсистема операційної системи виконує це перетворення у вигляді композиції матриць моделі, виду (камери) та проєктування, які застосовуються до однорідних координат вершин.

Орієнтація об'єкта в просторі (наприклад, положення робочого органу маніпулятора чи стрілки давача) задається композицією поворотів навколо координатних осей[3]:

$$\mathrm{R}_{\mathrm{xyz}} = R_z(\gamma)R_y(\beta)R_x(\alpha) \qquad (1)$$

де $R_x(\alpha)$, $R_y(\beta)$, $R_z(\gamma)$ — матриці елементарних поворотів навколо осей *X, Y, Z* на кути *α, в, γ* відповідно (кути Ейлера). Така композиція дозволяє коректно відтворювати орієнтацію рухомих елементів об'єкта керування — клапанів, насосів, поворотних платформ тощо — у тривимірній сцені.

Після переходу у простір камери (*eye space*) координати точок підлягають перспективному проєктуванню, що формує враження глибини сцени. Спрощена залежність координати x у просторі відсікання (clip space) від координати у просторі камери має вигляд:

$$\mathrm{x}_{\mathrm{clip}} = \frac{2n}{r-l}x_{eye} + \frac{r+l}{r-l}z_{eye}, \quad w_{clip} = -z_{eye} \qquad (2)$$

де *n, f* — відстані до близької та далекої площин відсікання, *l*, *r* — межі видимого об'єму по горизонталі. Аналогічні співвідношення формуються для координат *y* та *z*. Після ділення на компоненту w_{clip} отримуються нормалізовані координати пристрою (NDC), які графічна

 ISSN 2306-5478

підсистема ОС перетворює у координати пікселів буфера кадру відповідно до роздільної здатності вікна або дисплея.

Важливо зазначити, що сучасні графічні API виконують ці перетворення безпосередньо на *GPU* у вершинному шейдері, тоді як ОС відповідає за створення графічного контексту, синхронізацію буферів обміну (*swap chain*) та диспетчеризацію команд між CPU і GPU.

Для відображення стану обладнання (температури, рівня заповнення, навантаження) у 3D-сценах систем керування широко застосовуються локальні моделі освітлення, які обчислюють інтенсивність кольору пікселя залежно від властивостей поверхні та джерел світла. Найпоширенішою серед них є модель Фонга, що описує сумарну освітленість точки поверхні як суму фонової, дифузної та дзеркальної складових[4]:

$$I = k_n I_n + k_d I_1 (N \cdot L) + k_s I_l (R \cdot V)^n \quad (3)$$

де k_a, k_d, k_s — коефіцієнти фонового, дифузного та дзеркального відбиття матеріалу; I_a, I_l — інтенсивності фонового та точкового джерел світла; *N, L, R, V* — одиничні вектори нормалі до поверхні, напрямку до джерела світла, відбитого променя та напрямку до спостерігача відповідно; n — показник блиску матеріалу.

У контексті систем керування модель освітлення може бути додатково модульована поточним значенням технологічного параметра — наприклад, колір дифузної складової k_d резервуара або трубопроводу динамічно змінюється відповідно до рівня заповнення чи температури середовища, що перетворює 3D-сцену на повноцінний інформаційний шар АСУТП, інтегрований у графічну підсистему ОС.

Якість 3D-візуалізації систем керування визначається не лише геометричною та фотометричною коректністю зображення, а й продуктивністю графічного конвеєра, оскільки затримка оновлення кадру безпосередньо впливає на сприйняття оператором динаміки процесу. Основним показником продуктивності є частота кадрів (FPS), що обернено пропорційна часу формування одного кадру[5]:

$$FPS = \frac{1}{T_{frame}} = \frac{1}{T_{CPU} + T_{GPU} + T_{sync}} \quad (4)$$

де T_{CPU} — час підготовки команд та логіки сцени центральним процесором, T_{GPU} — час виконання графічного конвеєра на GPU, T_{sync} — час очікування синхронізації (наприклад, вертикальної синхронізації дисплея), що адмініструється графічною підсистемою ОС.

Теоретична обчислювальна продуктивність GPU, яка визначає верхню межу T_{GPU}, може бути оцінена за формулою[6]:

$$P_{GPU} = N_{cores} \cdot f_{clk} \cdot N_{FLOP/cyc} \cdot 10^{-9} [GFLOPS] \quad (5)$$

де N_{cores} — кількість обчислювальних ядер (потокових процесорів) GPU, f_{clk} — тактова частота, $N_{FLOP/cyc}$ — кількість операцій з рухомою комою за такт на одне ядро.

Окрім обчислювальної продуктивності, критичним фактором є пропускна здатність відеопам'яті, яка обмежує швидкість передавання геометричних та текстурних даних між системною та відеопам'яттю:

$$B_{eff} = \frac{V_{data}}{T_{Transfer}} = \frac{N_{vertices} \cdot S_{vertex}}{T_{Transfer}} [GB/S] \quad (6)$$

де V_{data} — обсяг переданих даних (вершинних буферів, текстур), $T_{transfer}$ — час передавання, $N_{vertices}$ — кількість вершин, S_{vertex} — розмір даних на одну вершину. Операційна система через драйвер графічного процесора керує розміщенням буферів у відеопам'яті та політикою їх кешування, що безпосередньо впливає на ефективне значення B_{eff}.

Для забезпечення змістовної 3D-візуалізації стану об'єкта керування необхідна математична модель його динаміки. У загальному вигляді лінійний об'єкт керування описується системою диференціальних рівнянь у формі простору станів:

$$x(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t) \quad (7)$$

де x(t) — вектор стану об'єкта (наприклад, рівень, швидкість його зміни та похідна другого порядку), *u(t)* — керуючий вплив, *y(t)* — вихідна (вимірювана) величина, *A, B, C, D* — матриці параметрів об'єкта.

Для 3D-візуалізації перехідного процесу вектор стану безпосередньо відображається у тривимірний геометричний простір сцени:

$$v_{3D}(t) = (x_1(t), x_2(t), x_3(t)) \in R^3 \quad (8)$$

тобто кожна з трьох компонент вектора стану інтерпретується як координата точки траєкторії в 3D-сцені, а сама траєкторія відображається полілінією або анімованою позначкою (маркером), положення якої оновлюється графічною підсистемою ОС із кожним новим кроком моделювання.

 ISSN 2306-5478

Для обчислювального експерименту обрано типовий об'єкт третього порядку з матрицею стану $A = [[0, 1, 0], [0, 0, 1], [-6, -11, -6]]$, $B = [0, 0, 1]^T$, що відповідає стійкій системі з характеристичним поліномом $\lambda^3 + 6\lambda^2 + 11\lambda + 6 = (\lambda+1)(\lambda+2)(\lambda+3)$, тобто з власними числами $-1, -2, -3$. Моделювання реакції об'єкта на одиничний ступінчастий вплив виконано чисельним інтегруванням рівняння стану методом, реалізованим у бібліотеці SciPy (odeint), з кроком дискретизації 8/1200 с на інтервалі 8 секунд. Для практичного підтвердження результатів, отриманих у процесі математичного моделювання та аналізу методів тривимірної візуалізації, розроблено інтерактивний програмний додаток «3D-візуалізація та моделювання систем керування в операційних системах» (рис.1), який виконує функції цифрового лабораторного стенда верифікації. Програмний комплекс забезпечує відтворення всіх основних обчислювальних експериментів, наведених у статті, та дозволяє здійснювати інтерактивний аналіз впливу параметрів математичних моделей на результати моделювання у режимі реального часу. Запропонований стенд забезпечує безпосередню взаємодію дослідника з математичними моделями, користувач має можливість змінювати параметри моделей, виконувати повторні обчислення та миттєво оцінювати вплив кожного параметра на характеристики досліджуваних процесів. Такий підхід забезпечує повну відтворюваність експериментів та значно підвищує достовірність отриманих результатів.

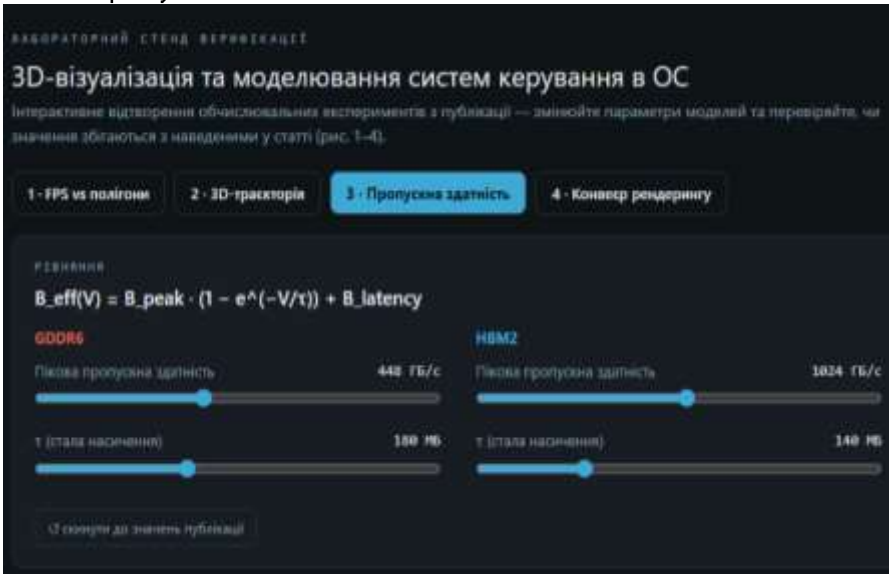

Рис.1. Лабораторний стенд «3D-візуалізація та моделювання систем керування в операційних системах»

Архітектура програмного забезпечення побудована за модульним принципом і складається з чотирьох взаємопов'язаних функціональних модулів, кожний з яких відповідає окремому напряму досліджень, наведених у статті.

Перший модуль реалізує дослідження залежності продуктивності графічної підсистеми від геометричної складності тривимірної сцени. За допомогою математичних моделей здійснюється порівняння програмної растеризації, що виконується центральним процесором, із апаратно-прискореним рендерингом засобами графічного процесора. Користувач може змінювати параметри моделі часу формування кадру та аналізувати залежність частоти кадрів від кількості полігонів сцени, перевіряючи експериментально отримані результати щодо переваги GPU-рендерингу, наведені у статті.

Другий модуль призначений для просторового моделювання динамічних систем автоматичного керування. У ньому реалізовано чисельне інтегрування системи диференціальних рівнянь методом Рунге–Кутти четвертого порядку з подальшим відображенням фазової траєкторії у тривимірному просторі станів. Інтерактивна зміна коефіцієнтів характеристичного полінома дозволяє досліджувати вплив параметрів системи на характер перехідного процесу, стійкість моделі, координати усталеного режиму та тривалість перехідного процесу, що безпосередньо підтверджує результати математичного моделювання, наведені у роботі.

Третій модуль забезпечує дослідження ефективної пропускної здатності сучасних типів відеопам'яті залежно від обсягу графічних даних. У програмі реалізовано асимптотичну модель насичення пропускної здатності для пам'яті GDDR6 та HBM2, що дозволяє аналізувати особливості передачі великих обсягів геометричних і текстурних даних. Отримані результати демонструють характер наближення до пікових значень пропускної здатності та підтверджують закономірності, встановлені під час проведення комп'ютерного експерименту.

Четвертий модуль призначений для аналізу структури часу виконання графічного конвеєра. Він моделює розподіл часу обробки кадру між основними етапами рендерингу — геометричними перетвореннями, растеризацією, шейдингом та постобробкою. Інтерактивне редагування параметрів дозволяє досліджувати вплив рівня деталізації сцени на завантаження окремих етапів графічного

ISSN 2306-5478

конвеєра та підтверджує домінування обчислень шейдингу при складних тривимірних сценах.

Розроблений програмний додаток є важливим елементом наукового дослідження та виконує функцію цифрового верифікаційного середовища, яке забезпечує практичне підтвердження достовірності математичних моделей і результатів комп'ютерного експерименту, представлених у статті. На відміну від статичних графіків і числових таблиць, інтерактивний програмний комплекс надає можливість безпосередньо відтворювати всі експерименти, змінювати початкові параметри моделей та аналізувати їхній вплив на кінцеві результати.

Використання такого підходу істотно підвищує рівень відтворюваності наукових досліджень, що є одним із ключових критеріїв сучасної наукової методології. Крім підтвердження результатів, наведених у статті, програмний стенд дозволяє проводити додаткові параметричні дослідження, аналізувати граничні режими функціонування моделей та оцінювати чутливість математичних залежностей до зміни вхідних параметрів.

Таким чином, програмний додаток є не лише допоміжним програмним засобом, а повноцінним інструментом експериментальної верифікації, який суттєво розширює практичну цінність дослідження. Його використання забезпечує об'єктивне підтвердження отриманих результатів, підвищує наукову обґрунтованість запропонованих моделей просторового моделювання та 3D-візуалізації, а також створює основу для подальшого розвитку інтерактивних програмних засобів аналізу систем автоматичного керування у сучасних операційних системах.

Результат моделювання представлено на рис. 2 у вигляді фазової траєкторії в координатах (x_1, x_2, x_3), що відповідають вихідній координаті об'єкта, швидкості та прискоренню її зміни. Траєкторія демонструє типовий аперіодичний перехідний процес: система рухається з початку координат (точка старту, позначена зеленим маркером) до сталого стану $x_{ss} \approx (0{,}1665;\ 0;\ 0)$ (червоний маркер), що відповідає встановленому значенню вихідної величини при одиничному вхідному впливі ($1/6 = 0{,}1667$, що узгоджується з коефіцієнтом передачі об'єкта за постійним сигналом). Оціночний час перехідного процесу (до входження у 5%-у зону навколо сталого значення) становить приблизно 4,1 с. Така візуалізація дозволяє оператору не лише спостерігати поточне значення параметра, а й якісно оцінювати характер перехідного процесу — наявність

коливань, перерегулювання чи монотонного підходу до уставки — безпосередньо у тривимірній сцені.

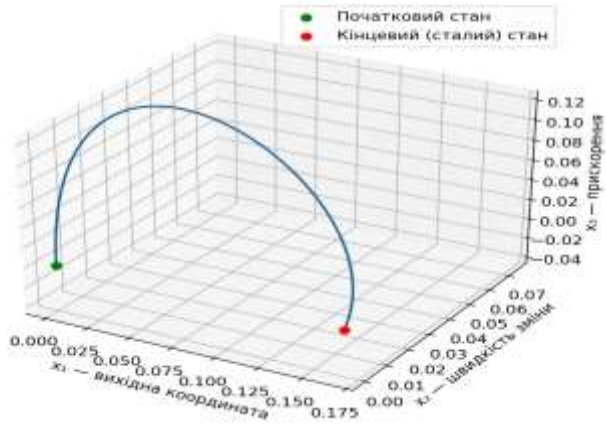

Рис.2. 3D-візуалізація фазової траєкторії об'єкта керування (перехідний процес при ступінчастому впливі

Для оцінювання впливу методів реалізації 3D-візуалізації на продуктивність графічної підсистеми ОС проведено обчислювальний експеримент, що моделює залежність частоти кадрів від кількості полігонів сцени для двох варіантів реалізації — програмної растеризації засобами CPU та апаратного рендерингу засобами *GPU*.

На рис. 3 наведено результати моделювання за формулою (4) для діапазону кількості полігонів від 10^3 до 10^7. За низької геометричної складності сцени (до 10^4 полігонів) обидва підходи забезпечують частоту кадрів, що значно перевищує умовний поріг комфортного сприйняття динамічної інформації (*60 FPS*). Проте зі зростанням кількості полігонів продуктивність програмної растеризації падає значно швидше: при кількості полігонів 10^6 програмна реалізація забезпечує близько *71 FPS*, тоді як апаратна реалізація — близько 714 FPS, тобто приблизно у 10 разів вище. Це підтверджує доцільність використання апаратно-прискореного графічного конвеєра (через API типу *OpenGL/Vulkan/Direct3D*, підтримувані драйверами ОС) для сцен середньої та високої складності, характерних для деталізованих 3D-моделей технологічного обладнання. Іншим важливим аспектом продуктивності є пропускна здатність відеопам'яті, яка визначає швидкість завантаження геометричних і текстурних даних сцени.

 ISSN 2306-5478

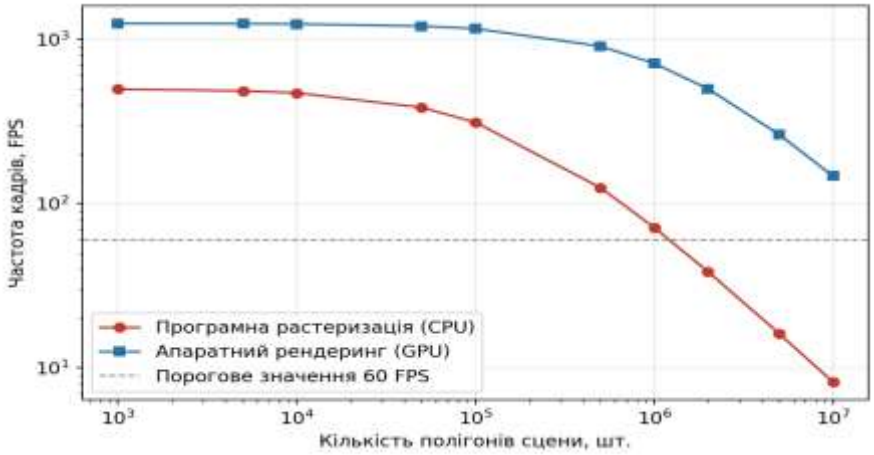

Рис.3. Залежність частоти кадрів від кількості полігонів сцени

На рис. 4 показано залежність ефективної пропускної здатності, обчисленої за формулою (6) з використанням асимптотичної моделі наближення до пікового значення, від обсягу графічних даних для двох типів відеопам'яті — GDDR6 (пікова пропускна здатність 448 ГБ/с) та HBM2 (1024 ГБ/с).

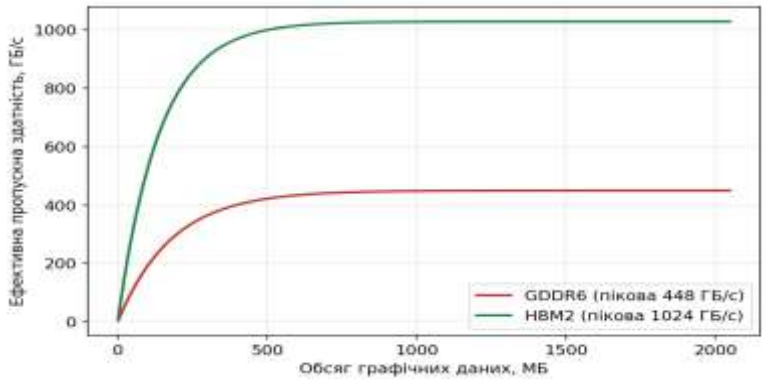

Рис.4. Залежність ефективної пропускної здатності відеопам'яті від обсягу переданих даних

Графік демонструє, що за малих обсягів даних (одиниці–десятки мегабайтів) ефективна пропускна здатність обох типів пам'яті є значно нижчою за пікову внаслідок накладних витрат на ініціалізацію передавання, тоді як за обсягів понад 500 МБ обидві характеристики виходять на насичення, наближаючись до пікових значень. Для 3D-сцен систем керування, що зазвичай містять помірний обсяг геометрії, але значну кількість текстур інтерфейсу (мнемосхеми, індикатори), цей ефект необхідно враховувати при

проєктуванні структури графічних ресурсів та політики їх кешування операційною системою.

Нарешті, на рис. 5 представлено розподіл часу обробки кадру за чотирма основними етапами графічного конвеєра — геометричним етапом, растеризацією, шейдингом/освітленням та постобробкою — для чотирьох рівнів деталізації сцени (низький, середній, високий, ультра).

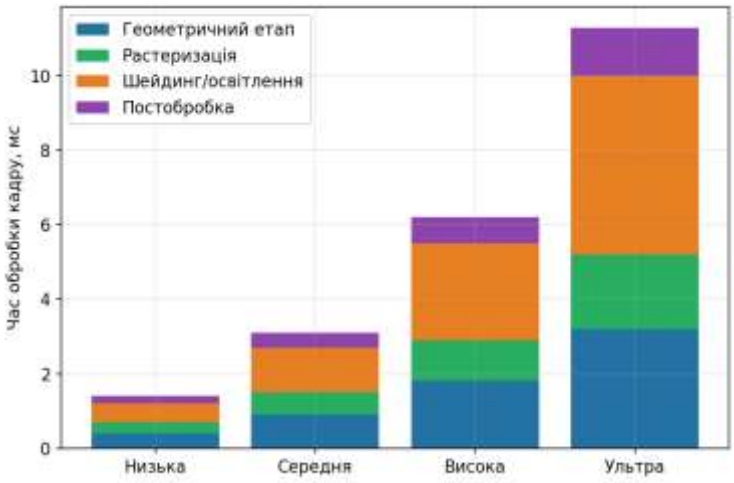

Рис.5. Розподіл часу обробки кадру за етапами конвеєра рендерингу для різних рівнів деталізації сцени

Результати показують, що зі зростанням деталізації сцени домінуючою складовою часу кадру стає етап шейдингу та освітлення (модель Фонга та подібні), частка якого зростає з приблизно 36% при низькій деталізації до понад 42% при ультра-деталізації. Це пояснюється тим, що складність обчислень освітлення зростає пропорційно кількості пікселів та джерел світла в сцені, тоді як геометричний етап масштабується переважно з кількістю вершин.

Отже, при оптимізації 3D-інтерфейсів систем керування для платформ з обмеженими апаратними ресурсами першочергову увагу доцільно приділяти спрощенню моделей освітлення (наприклад, переходу від моделі Фонга до моделі Гуро для вторинних елементів сцени) та використанню технік відкладеного рендерингу (deferred shading), що дозволяють уникати надлишкових обчислень освітлення для прихованих фрагментів.

ISSN 2306-5478

Висновки. У роботі розглянуто комплекс методів 3D-візуалізації та моделювання систем керування в сучасних операційних системах, що охоплює математичний апарат геометричних перетворень і перспективного проєктування, моделі локального освітлення для фотореалістичного відображення технологічних параметрів, а також показники продуктивності графічного конвеєра — частоту кадрів, обчислювальну потужність GPU та пропускну здатність відеопам'яті. Виконане комп'ютерне моделювання динаміки об'єкта керування третього порядку та її відображення у вигляді фазової траєкторії в тривимірному просторі станів підтвердило практичну можливість поєднання класичних методів теорії автоматичного керування із сучасними засобами 3D-графіки операційних систем. Проведений обчислювальний експеримент засвідчив, що апаратно-прискорена реалізація графічного конвеєра забезпечує приблизно на порядок вищу продуктивність порівняно з програмною растеризацією за умов сцен середньої та високої складності, а етап шейдингу та освітлення є домінуючою складовою часу обробки кадру при високій деталізації сцени, що визначає пріоритетні напрями оптимізації при розробці 3D-інтерфейсів АСУТП. Отримані результати можуть бути використані під час проєктування підсистем візуалізації SCADA/HMI, що функціонують у складі сучасних операційних систем загального та реального часу, а також під час вибору апаратної платформи для систем людино-машинної взаємодії з підвищеними вимогами до наочності представлення інформації.

1. Hughes van Dam A., McGuire M., Sklar D. F., Hoffman N., Whitted T., Feiner S. Computer Graphics: Principles and Practice. 3rd ed.; Addison-Wesley: Boston, MA, USA, 2013. https://www.informit.com/store/computer-graphics-principles-and-practice-9780321399526. 2. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering. 4th ed.; CRC Press: Boca Raton, FL, USA, 2018. https://www.realtimerendering.com. 3. Marschner S., Shirley P. Fundamentals of Computer Graphics. 5th ed.; CRC Press: Boca Raton, FL, USA, 2021. https://www.routledge.com/Fundamentals-of-Computer-Graphics/Marschner-Shirley/p/book/9780367505035. 4. Allison C. J.; Zhou H.; Munawar A.; Kazanzides P.; Barragan J. A. FIRE-3DV: Framework-Independent Rendering Engine for 3D Graphics Using Vulkan. arXiv 2024, arXiv:2410.05095. https://doi.org/10.48550/arXiv.2410.05095. 5. Boutsi A.; Ioannidis C.; et al. Multithreaded Rendering for Cross-Platform 3D Visualization Based on Vulkan API. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* 2020, XLIV-4/W1, 57–64. https://doi.org/10.5194/isprs-archives-XLIV-4-W1-2020-57-2020. 6. Kerbl B., Kopanas G., Leimkühler T. Drettakis G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 2023, 42(4), Article 139.

https://doi.org/10.1145/3592433. 7. Kato H., Beker D., Morariu M., Ando M.; Matsuoka Y. Differentiable Rendering: A Survey. arXiv 2020, arXiv:2006.12057. https://doi.org/10.48550/arXiv.2006.12057. 8. NVIDIA Corporation. CUDA C++ Programming Guide. Version 12.6; NVIDIA: Santa Clara, CA, USA, 2024. https://docs.nvidia.com/cuda/. 9. Khronos Group. Vulkan API Specification. Version 1.4; Khronos Group, 2025. https://registry.khronos.org/vulkan/. 10. Khronos Group. OpenGL® 4.6 Core Profile Specification. Khronos Group, 2019. https://registry.khronos.org/OpenGL/specs/gl/glspec46.core.pdf. 11. MathWorks. MATLAB Graphics Documentation. MathWorks Inc., 2025. https://www.mathworks.com/help/matlab/graphics.html. 12. SciPy Community. SciPy User Guide. Version 1.16; SciPy Developers, 2025. https://docs.scipy.org/doc/scipy/. 13. Matplotlib Development Team. mplot3d Toolkit Documentation. Matplotlib 3.10 Documentation, 2025. https://matplotlib.org/stable/api/toolkits/mplot3d.html. 14. Carter F.; Hitschfeld N.; Navarro C. A. Mìmir: A Real-Time Interactive Visualization Library for CUDA Programs. arXiv 2025, arXiv:2504.20937. https://doi.org/10.48550/arXiv.2504.20937.

REFERENCE

1. Hughes van Dam A., McGuire M., Sklar D. F., Hoffman N., Whitted T., Feiner S. Computer Graphics: Principles and Practice. 3rd ed.; Addison-Wesley: Boston, MA, USA, 2013. https://www.informit.com/store/computer-graphics-principles-and-practice-9780321399526. 2. Akenine-Möller T., Haines E., Hoffman N. Real-Time Rendering. 4th ed.; CRC Press: Boca Raton, FL, USA, 2018. https://www.realtimerendering.com. 3. Marschner S., Shirley P. Fundamentals of Computer Graphics. 5th ed.; CRC Press: Boca Raton, FL, USA, 2021. https://www.routledge.com/Fundamentals-of-Computer-Graphics/Marschner-Shirley/p/book/9780367505035. 4. Allison C. J.; Zhou H.; Munawar A.; Kazanzides P.; Barragan J. A. FIRE-3DV: Framework-Independent Rendering Engine for 3D Graphics Using Vulkan. arXiv 2024, arXiv:2410.05095. https://doi.org/10.48550/arXiv.2410.05095. 5. Boutsi A.; Ioannidis C.; et al. Multithreaded Rendering for Cross-Platform 3D Visualization Based on Vulkan API. *Int. Arch. Photogramm. Remote Sens. Spatial Inf. Sci.* 2020, XLIV-4/W1, 57–64. https://doi.org/10.5194/isprs-archives-XLIV-4-W1-2020-57-2020. 6. Kerbl B., Kopanas G., Leimkühler T. Drettakis G. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 2023, 42(4), Article 139. https://doi.org/10.1145/3592433. 7. Kato H., Beker D., Morariu M., Ando M.; Matsuoka Y. Differentiable Rendering: A Survey. arXiv 2020, arXiv:2006.12057. https://doi.org/10.48550/arXiv.2006.12057. 8. NVIDIA Corporation. CUDA C++ Programming Guide. Version 12.6; NVIDIA: Santa Clara, CA, USA, 2024. https://docs.nvidia.com/cuda/. 9. Khronos Group. Vulkan API Specification.

 ISSN 2306-5478

Version 1.4; Khronos Group, 2025. https://registry.khronos.org/vulkan/. 10. Khronos Group. OpenGL® 4.6 Core Profile Specification. Khronos Group, 2019. https://registry.khronos.org/OpenGL/specs/gl/glspec46.core.pdf. 11. MathWorks. MATLAB Graphics Documentation. MathWorks Inc., 2025. https://www.mathworks.com/help/matlab/graphics.html. 12. SciPy Community. SciPy User Guide. Version 1.16; SciPy Developers, 2025. https://docs.scipy.org/doc/scipy/. 13. Matplotlib Development Team. mplot3d Toolkit Documentation. Matplotlib 3.10 Documentation, 2025. https://matplotlib.org/stable/api/toolkits/mplot3d.html. 14. Carter F.; Hitschfeld N.; Navarro C. A. Mìmir: A Real-Time Interactive Visualization Library for CUDA Programs. arXiv 2025, arXiv:2504.20937. https://doi.org/10.48550/arXiv.2504.20937.

HumenniyP.V. [1, ORCID ID:0000-0003-0982-3305]
PHd Associate Professor

[1]West Ukrainian National University, Ternopil

METHODS OF SPATIAL MODELLING AND 3D VISUALISATION IN MODERN OPERATING SYSTEMS

This article investigates methods for three-dimensional (3D) visualisation of automatic control system data using the graphics subsystems of modern operating systems. The mathematical framework of geometric transformations, perspective projection, and lighting models (in particular, the Phong reflection model) that underpins the graphics pipeline for rendering control information in SCADA/HMI interfaces is examined. A computer simulation of the dynamics of a typical third-order linear control system described in the state-space form is performed, and its transient response is represented as a phase trajectory in three-dimensional state space. The steady-state value is determined, and the transient response time is estimated. A computational experiment comparing the rendering performance of software-based (CPU) and hardware-accelerated (GPU-based) 3D rendering implementations is conducted. The relationship between frame rate and the number of scene polygons is established, demonstrating that, for scenes of medium and high complexity, the hardware implementation delivers approximately an order of magnitude higher performance than software rasterisation (specifically, at one million polygons, approximately 714 frames per

second compared with 71 frames per second). Furthermore, the effective memory bandwidth of GDDR6 and HBM2 video memory as a function of the volume of transferred graphics data is analysed, together with the distribution of frame processing time across the stages of the graphics pipeline for different levels of scene complexity. The results indicate that the shading and lighting stage becomes the dominant contributor to the total rendering time as the level of scene detail increases. The proposed methods and the obtained results can be applied to the design of SCADA/HMI human–machine interface subsystems with integrated 3D visualisation support in both general-purpose and real-time operating systems, as well as to the selection of hardware platforms for human–machine interaction systems requiring high-performance graphical visualisation and clear information presentation.
Keywords: 3D visualisation; graphics pipeline; operating system; GPU; control system modelling; phase trajectory; rendering performance; SCADA; HMI.

Отримано: 31 беерзня 2026 року
Прорецензовано: 27 квітня 2026 року
Прийнято до друку: 29 травня 2026 року