УДК 004.4:004.92

https://doi.org/10.52058/2786-6025-2025-13(54)-1993-2009

Гуменний Петро Володимирович кандидат технічних наук, доцент, доцент кафедри спеціалізованих комп'ютерних систем, Західноукраїнський національний університет, Тернопіль, https://orcid.org/0000-0003-0982-3305

Юрченко Юрій Юрійович старший викладач кафедри комп'ютерних наук та інформаційних систем, Державний торговельно-економічний університет, м. Київ, https://orcid.org/0000-0002-8047-7647

Федорчук Анна Леонідівна кандидат педагогічних наук, доцент, доцент кафедри комп'ютерних наук та інформаційних технологій, Житомирський державний університет імені Івана Франка, м. Житомир, https://orcid.org/0000-0001-8227-3210

ІНТЕГРАЦІЯ КОМП'ЮТЕРНОЇ ГРАФІКИ З ОПЕРАЦІЙНИМИ СИСТЕМАМИ ДЛЯ ПОКРАЩЕННЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

Анотація. У статті проведено комплексний аналіз підходів до інтеграції засобів комп'ютерної графіки в операційні системи з метою підвищення якості та зручності графічного інтерфейсу користувача (Graphical User Interface, GUI). Розглянуто еволюцію графічних підсистем трьох основних настільних ОС – Windows, Linux та macOS – у контексті використання апаратного прискорення графіки і композитних менеджерів вікон. Відзначено, що перші GUI-системи (починаючи від розробок Xerox PARC 1970-х років) заклали основу WIMP-інтерфейсу (вікна, іконки, меню, вказівники), однак тривалий час обчислювальні обмеження не дозволяли реалізувати складні візуальні ефекти та плавну взаємодію. У статті показано, як із розвитком апаратної частини – появою графічних процесорів (GPU) – відбувся перехід від програмного рендерингу інтерфейсу до апаратно прискореного. Проаналізовано архітектурні зміни: впровадження композиційних менеджерів вікон (Windows DWM, Quartz Compositor в macOS, композитні менеджери в Linux) та нових графічних API (DirectX/Direct2D, OpenGL/Metal, тощо) для відмалювання інтерфейсів. Результати дослідження підтверджують, що глибока інтеграція комп'ютерної графіки в ОС дозволила суттєво підвищити продуктивність і якість UI: забезпечити плавну анімацію на частотах 60 кадрів/с і вище, зменшити навантаження на ЦП за рахунок GPU, реалізувати візуальні ефекти (прозорість,

тіні, 3D-трансформації) без втрати швидкодії. Водночас зазначено існуючі проблеми та невирішені задачі, зокрема затримки (latency), які внесе композитний підхід, і складнощі забезпечення сумісності та доступності. Окреслено перспективи подальшого розвитку, зокрема впровадження інтерфейсів штучної/віртуальної реальності (AR/VR) на прикладі visionOS від Apple, що вказує на початок ери просторових обчислень у користувацьких інтерфейсах.

Ключові слова: графічний інтерфейс користувача; операційна система; комп'ютерна графіка; апаратне прискорення; композитний менеджер; користувацький досвід.

Humennyy Petro PhD, Associate Professor, Associate Professor of the Department of Specialized Computer Systems, West Ukrainian National University, Ternopil, https://orcid.org/0000-0003-0982-3305

Yurchenko Yurii, Senior Lecturer, Department of Computer Science and Information Systems, State University of Trade and Economics, Kyiv, https://orcid.org/0000-0002-8047-7647

Fedorchuk Anna Candidate of Pedagogical Sciences, Docent, Associate Professor of the Department of Computer Science and Information Technology, Zhytomyr Ivan Franko State University, Zhytomyr, https://orcid.org/0000-0001-8227-3210

INTEGRATION OF COMPUTER GRAPHICS WITH OPERATING SYSTEMS TO IMPROVE THE USER INTERFACE

Abstract. The article provides a comprehensive analysis of approaches to integrating computer graphics tools into operating systems in order to improve the quality and convenience of the graphical user interface (GUI). The evolution of the graphics subsystems of the three main desktop OSes – Windows, Linux, and macOS – is considered in the context of using hardware graphics acceleration and composite window managers. It is noted that the first GUI systems (starting with the developments of Xerox PARC in the 1970s) laid the foundation for the WIMP interface (windows, icons, menus, pointers), but for a long time computational limitations did not allow the implementation of complex visual effects and smooth interaction. The article shows how with the development of the hardware part – the advent of graphics processors (GPUs) – the transition from software rendering of the interface to hardware-accelerated rendering took place. Architectural changes are analyzed: the introduction of composite window managers (Windows DWM, Quartz

Compositor in macOS, composite managers in Linux) and new graphics APIs (DirectX/Direct2D, OpenGL/Metal, etc.) for drawing interfaces. The results of the study confirm that the deep integration of computer graphics into the OS has significantly improved the performance and quality of the UI: ensuring smooth animation at frequencies of 60 frames/s and higher, reducing the load on the CPU due to the GPU, implementing visual effects (transparency, shadows, 3D transformations) without losing performance. At the same time, existing problems and unresolved issues are noted, in particular, the latency that the composite approach will introduce, and the difficulties of ensuring compatibility and accessibility. Prospects for further development are outlined, in particular, the introduction of artificial/virtual reality (AR/VR) interfaces using the example of Apple's visionOS, which indicates the beginning of the era of spatial computing in user interfaces.

Keywords: graphical user interface; operating system; computer graphics; hardware acceleration; composite manager; user experience.

Постановка проблеми. Графічний інтерфейс користувача є критичною складовою всіх сучасних операційних систем, оскільки саме він визначає зручність взаємодії людини з комп'ютером. Історично, GUI розроблялися для підвищення юзабельності порівняно з текстовими інтерфейсами, дозволяючи працювати через візуальні метафори – вікна, кнопки, значки тощо [1]. Проте ранні GUI-системи були обмежені продуктивністю центрального процесора (ЦП) та відсутністю спеціалізованих графічних пристроїв. Кожне застосування малювало своє вікно в основну відеопам'ять, що у разі багатозадачності призводило до візуальних артефактів: наприклад, при перетягуванні вікна "шлейф" від нього залишався через те, що фонове вікно не встигало перемалюватися [2]. Відсутність апаратного згладжування та обмежена кольорова палітра призводили до "зубчастих" шрифтів і графіки невисокої якості. З розширенням функціональності GUI постала проблема продуктивності: користувацький інтерфейс мав залишатися чуйним (відгук менше ~0,1 с для непомітності затримки) навіть зі зростанням роздільної здатності екранів, складності елементів та графічних ефектів. Досягти цього суто програмними методами малювання на ЦП ставало дедалі важче.

Таким чином, актуальною проблемою стала інтеграція засобів комп'ютерної графіки на рівні операційної системи – зокрема, використання можливостей графічного процесора та сучасних графічних API – для підвищення продуктивності та якості GUI. Це включає запровадження нового програмного забезпечення в ОС (дисплейних серверів, менеджерів вікон з композицією, графічних бібліотек) та апаратних вимог (наявність GPU з необхідними можливостями) задля розв'язання важливих практичних завдань: плавної анімації інтерфейсу, відсутності мерехтіння і артефактів, підтримки

високих DPI, складних візуальних ефектів і тривимірних елементів. Вирішення цієї проблеми має вагоме значення, адже якість UI безпосередньо впливає на ефективність роботи користувачів, їх задоволеність програмним продуктом та доступність комп'ютерних систем для широкого кола людей.

Аналіз останніх досліджень і публікацій. Питання побудови ефективних графічних інтерфейсів користувача досліджується ще з 1970-х років. Класичними є роботи лабораторії Xerox PARC, де було створено перший у світі прототип GUI (система Alto, 1973 р.), що започаткував еру комп'ютерної графіки в інтерфейсах [3]. Комерційний розвиток GUI розпочався в 1980-х: Apple Lisa (1983) та Macintosh (1984) стали піонерами впровадження графічного інтерфейсу в персональних комп'ютерах. В цих ранніх системах використовувалися програмні графічні рушії (наприклад, QuickDraw у класичному Mac OS, GDI в Windows), які працювали виключно на ЦП і мали суттєві обмеження. Зокрема, GDI в Windows відповідав за базові операції малювання примітивів і тексту та виведення їх на екран або принтер [4]. Аналогічні компоненти були й в інших ОС (QuickDraw в Mac OS, ядро X11 у UNIX [4]. Вони забезпечували крос-пристроєву графічну абстракцію ("що бачиш те й отримаєш" – WYSIWYG-друк) [4], але не були розраховані на складну анімацію чи 3D-графіку. Дослідження показали, що без апаратної підтримки такі рушії не можуть забезпечити високу частоту оновлення та плавність сучасних інтерфейсів [4].

З появою доступних графічних процесорів у 1990-х – 2000-х роках науковці та інженери почали шукати шляхи використання апаратного прискорення у GUI. Важливим кроком стала поява композиційних менеджерів вікон (compositing window managers). У 2001 році Apple випустила Mac OS X з новим графічним ядром Quartz: вперше в масовій ОС усі вікна рендерились у позакадрові буфери й складалися на екран спеціальним процесом – Quartz Compositor [5]. Це дозволило реалізувати раніше недоступні ефекти (прозорість вікон, згладжені тіні тощо) [5]. Однак початково вся композиція виконувалась на ЦП, через що продуктивність була низькою: за словами оглядачів, перша версія Aqua-інтерфейсу була "нестерпно повільною та ресурсомісткою" [5]. Дослідники відзначали цей недолік і вказували на необхідність залучення GPU. Вже в Mac OS X 10.2 Jaguar (2002) Apple активувала технологію Quartz Extreme – перенесення композиції вікон на графічний процесор – що дозволило зрівняти швидкодію з докомпозитним рівнем [5]. Цей крок підтвердив твердження, що апаратне прискорення є вирішальним для продуктивності GUI [6]. Паралельно в дослідженнях згадувалось, що схожі ідеї циркулювали і раніше: наприклад, Amiga ще в 1985 р. мала апаратно підтримувані графічні накладення, а Windows 2000 ввів механізм "layered windows" для часткової композиції окремих прозорих вікон [5].

У середині 2000-х відбувся прорив у графічних можливостях настільних ОС. Microsoft у 2006 році випустила Windows Vista, в якій було фундаментально змінено архітектуру графічної підсистеми шляхом впровадження Desktop Window Manager (DWM) – композитного менеджера вікон на базі DirectX [2]. Це рішення спиралося на досвід Apple і дослідження в галузі: кожне вікно тепер малювалося у власний буфер, а DWM компонував з них кінцеве зображення на GPU. Відомо, що для роботи нового інтерфейсу Windows Aero Vista вимагала відеокарту з підтримкою DirectX 9 і шейдерів 2.0, інакше розширені ефекти (прозорість “скляних” рамок, 3D-перемикання Windows Flip 3D) вимикалися. Таким чином, спочатку апаратні вимоги GPU були прямо пов’язані з можливістю використання повного спектру UI функцій ОС. DWM у Vista/Windows 7 продемонстрував практичну цінність композиції: зменшилась кількість перемальовувань прихованих областей, зникли артефакти “шлейфу” при пересуванні вікон, з’явилися візуальні ефекти (напівпрозорі вікна, згладжування, анімації) без помітного падіння швидкодії [2]. Науковці відзначали, що завдяки DWM Windows почала повноцінно використовувати можливості GPU для рендерингу інтерфейсу, звільнивши CPU від графічних задач [7].

У спільноті розробників Linux і дослідників відкритих систем також тривали пошуки оптимальних рішень. Традиційний X11-дисплейний сервер, спроєктований ще у 1980-х, працював за схемою клієнт–сервер: програми надсилали запити малювання на X-сервер, який виконував їх і оновлював спільний екран. Це забезпечило мережеву прозорість, але призвело до накопичення застарілого коду і ускладнювало впровадження новітніх графічних технологій [8]. У 2004–2006 рр. з’явилися перші композитні менеджери для X11, як-от Compiz, які використовували OpenGL для прискорення інтерфейсу. Дослідники відзначали, що Compiz об’єднав класичний менеджер вікон і композитний сервер, використовуючи 3D-акселерацію, і надав Linux-десктопам низку новаторських ефектів – 3D-обертання робочих просторів “кубом”, прозорі та “вологі” вікна, згладжені масштабовані прев’ю тощо [9]. Це не лише зробило роботу привабливішою візуально, але й спростило взаємодію (наприклад, ефект Exposé у Compiz для перегляду всіх вікон, аналогічний Mac OS X) [9]. Однак X11 залишався обмеженим архітектурно, тому з 2008 р. розпочато проект Wayland, про який йдеться в низці публікацій. Wayland пропонує простішу модель: програма сама малює інтерфейс засобами GPU (через OpenGL ES/EGL), а композитор Wayland безпосередньо відображає готові буфери на екран [8]. Відмова від посередництва центрального X-сервера усуває затримки та надлишкові копіювання даних, підвищуючи швидкодію GUI (менші затримки при відкритті вікон, плавніше перетягування та масштабування) [8]. Останні дослідження

підтверджують, що перехід до Wayland дає виграш у продуктивності та безпеці: нова система усуває низку legacy-проблем X11 і дозволяє гнучкіше використовувати ресурси ядра Linux (DRM, KMS) для планування графічних операцій [10].

Невирішені раніше частини загальної проблеми пов'язані з оптимізацією балансу між багатством графічних можливостей та ефективністю. Сучасні дослідження звертають увагу на затримки відображення при композитному підході. Хоча GPU-композиція прискорила анімацію, вона вводить додатковий етап обробки кадрів. Згідно з аналізом Р.Левієна, сама природа композитора додає приблизно "кадр затримки" (~16 мс при 60 Гц) до часу від реагування додатка на дію користувача до показу результату [5]. Це призводить до накопичення латентності ~30–60 мс навіть у кращих випадках, що помітно вимогливим користувачам (особливо в інтенсивних сценаріях, як-от ігри чи VR).

У спільноті розробників вже обговорюються шляхи вирішення – від спеціалізованих апаратних оверлеїв (апаратне накладання курсора, відео тощо) [5] до радикальної переробки архітектури композитора в майбутньому [5]. Також окремо стоїть проблема доступності (accessibility) та сумісності: інтерфейси, що рендеряться "напряму" на GPU нестандартними засобами, можуть не враховувати масштаб шрифтів ОС або не бути видимими для екранних читалок [6]. У новітніх дослідженнях (наприклад, у розробках GPU-орієнтованих UI-фреймворків) наголошується, що ці питання доведеться вирішувати додатковими зусиллями, оскільки стандартні засоби доступності прив'язані до традиційних UI-інструментаріїв [6].

Інтеграція комп'ютерної графіки в ОС пройшла шлях від перших концепцій GUI до повсюдного використання апаратного прискорення сьогодні. Невирішеними залишаються питання оптимізації затримок і забезпечення повної підтримки всіх можливостей (безпека, доступність) у нових графічних стеках. Дана стаття присвячена узагальненню цього досвіду та висвітленню сучасного стану і перспектив вирішення означених проблем.

Мета статті. Метою цієї статті є дослідження еволюції та сучасні підходи до інтеграції комп'ютерної графіки з операційними системами для поліпшення користувацького інтерфейсу, а також проаналізувати переваги та виклики, пов'язані з апаратно прискореним графічним інтерфейсом. Задля досягнення поставленої мети в роботі вирішуються такі завдання:

- надання характеристики процесу реалізації та загальної архітектури графічного інтерфейсу в популярних ОС (Windows, Linux, macOS) з акцентом на використанні GPU;
- порівняння їх підходів до композиції вікон, згладжування графіки, роботи з 2D/3D-примітивами та високими роздільностями;

- узагальнити вплив інтеграції комп'ютерної графіки на продуктивність та UX (користувацький досвід) в кожній із систем;
- виявити типові проблеми та обмеження нинішніх рішень і окреслити перспективні напрямки подальшого розвитку графічних підсистем ОС.

Виклад основного матеріалу дослідження.

Windows. У операційній системі Microsoft Windows підтримка графічного інтерфейсу еволюціонувала від простих програмних засобів малювання до повноцінного використання GPU-акселерації. Ранні версії Windows (3.x, 95/98, до Windows XP) спиралися на графічний рушій GDI (Graphics Device Interface), який надавав базові функції виведення 2D-графіки – малювання ліній, прямокутників, тексту, роботи з палітрою кольорів тощо [4]. GDI оперував абстрактними "контекстами пристрою" і дозволяв однаково виводити зображення на екран та на друк, забезпечуючи принцип WYSIWYG для офісних програм [4]. Однак GDI мав суттєві обмеження: він працював у режимі негайного малювання (immediate mode), де кожне застосування безпосередньо змінювало буфер екрану. У багатовіконному середовищі це вимагало численних повідомлень WM_PAINT прихованим вікнам і могло спричиняти мерехтіння та артефакти при перерисовці (як згадано, "шлейфи" від вікон) [2]. Крім того, GDI майже не використовував апаратне прискорення: лише окремі операції могли бути розігнані через відеодрайвер (напр. BitBLT для копіювання прямокутників) [4]. Продвинута графіка – згладжування (anti-aliasing), прозорість, 3D – не підтримувались або реалізовувались на боці ЦП із низькою швидкістю [4]. Наприклад, GDI не міг апаратно синхронізувати малювання з кадрами дисплея чи виконувати складні трансформації; для 3D-графіки пропонувалось використовувати окремі API (Direct3D, OpenGL) в обхід GDI [4].

Починаючи з Windows 2001 XP, Microsoft зробила крок до покращення 2D-графіки, запровадивши GDI+ – розширення, що додало підтримку згладжених шрифтів, прозорості та сучасних форматів зображень. Проте GDI+ все одно рендерив усе CPU-методами і не міг повністю розв'язати проблем продуктивності [11]. Кардинальні зміни відбулися у Windows Vista (2006), де було впроваджено Desktop Window Manager (DWM) – композиційний менеджер вікон. З увімкненим DWM програми більше не малюють напряму в загальний буфер кадру; замість цього кожне вікно рендериться у свій позаекранний буфер (поверхню), а DWM на кожному кадрі компонує всі буфери на екран за допомогою GPU [2]. Така архітектура одразу дала кілька переваг [2]: якщо вікно прикрите іншим, його вміст не перемальовується щоразу, що знижує навантаження, кадри переходів між вікнами стають цільними, з'явилась можливість глобальних візуальних ефектів – прозорості та розмиття на рівні композиції (ефект Aero Glass), 3D-трансформації всього вікна (ефект Flip 3D) тощо, оскільки композитор має доступ до пікселів всіх вікон і

може комбінувати їх з шейдерами [7]. У Vista/Windows 7 DWM працював поверх графічного API DirectX 9; фактично роботу композитора було організовано через 3D-сцену: робочий стіл став 3D-поверхнею, кожне вікно – текстурою на прямокутній сітці-трикутнику, яку можна плавно трансформувати (зум, обертання) [7]. Вікна з традиційним GDI-рендерингом DWM перехоплює та поміщає у текстури, щоб забезпечити їх участь у композиції [7]. Завдяки переходу на DirectX, більшість роботи з отрисовки пікселів бере на себе GPU, розвантажуючи CPU [7].

Для користувачів та розробників це означало якісно новий рівень інтерфейсу в Windows. Візуальні ефекти та анімації стали плавними і багатими: з'явилась прозорість вікон з динамічним розмиттям фону, анімація згортання/розгортання, живі ескізи вікон на панелі завдань. Продуктивність і відгук інтерфейсу покращились – GPU виконує обчислення, оптимізовані для графіки, тож більшість типових операцій малювання (переміщення, прокрутка, оновлення інтерфейсу) відбуваються швидко і без мерехтінь [11]. Як зазначено в документації, Direct2D (сучасний 2D-API Windows, представлений у Windows 7) перевершує GDI/GDI+ у більшості сценаріїв завдяки повному використанню апаратного прискорення, виконуючи рендеринг примітивів на графічному процесорі [11]. Він підтримує апаратне альфа-змішування (прозорість) та згладжування геометрії, що було недоступне або повільне в GDI [11]. На практиці це дає значно плавніші шрифти і криві лінії, відсутність "зубців" на контрастних границях об'єктів, покращуючи читабельність і вигляд інтерфейсу [11]. DWM також забезпечив масштабування інтерфейсу для дисплеїв з високою щільністю пікселів (High DPI) – у Vista/7 він вміє автоматично масштабувати вміст старих додатків, якщо ті не готові до High DPI, аби хоча б приблизно узгодити їх розміри [2].

У наступних версіях Windows графічна підсистема продовжила удосконалюватися. Windows 7 ввів покращений драйверний модель WDDM 1.1, що дозволила зберіга=ти GDI-буфери виключно у відеопам'яті та уникнути зайвих копій в системній пам'яті. Windows 8 зробила DWM обов'язковим (його не можна вимкнути навіть якщо Aero-ефекти неактивні) – це підкреслило остаточний перехід до постійної GPU-композиції на всіх машинах [7]. Такі технології як DirectComposition (введена у Windows 8) дозволили стороннім додаткам і фреймворкам інтегрувати власні інтерфейсні елементи з апаратним композиціюванням, що особливо використовувалось у новому середовищі Windows Store/Metro [5]. В Windows 10/11 Microsoft продовжила курс на збагачення UI-графіки: наприклад, концепція Fluent Design включає ефекти акрилового розмиття, тіней, підсвічування тощо, які опираються на можливості сучасного GPU (шейдерні ефекти) і прозоро вбудовані у систему. Таким чином, Windows наразі повністю інтегрує комп'ютерну графіку у відображення

інтерфейсу – від рендерингу кожної кнопки шрифтів з субпіксельним згладжуванням до складення робочого столу як 3D-сцени. Це значно поліпшило суб'єктивну плавність та інтерактивність роботи: навіть на великому 4K-моніторі сучасний Windows UI оперує переважно 60 fps+ і виглядає чітко завдяки векторним шрифтам і формам, що масштабуються без втрати якості [11]. Вимоги до апаратного забезпечення зросли відповідно: GPU з підтримкою DirectX тепер фактично обов'язкова частина системи (інтегровані графічні ядра або дискретні відеокарти), без яких неможливо уявити повноцінний інтерфейс Windows.

MacOS. Операційна система Apple macOS (раніше Mac OS X) від самого початку була орієнтована на потужну графічну підсистему і в багатьох аспектах виступила новатором інтеграції комп'ютерної графіки у UI. Попередник macOS – класичний Mac OS (система Macintosh 1984–2001 рр.) – мав однин з перших успішних GUI, в якому використовувався графічний рушій QuickDraw. QuickDraw, подібно до Windows GDI, був 2D-графічним API на боці ЦП, що забезпечував відмальовування вікон, меню, контролів у доволі простій растровій графіці тих років. Суттєвих засобів апаратного прискорення у той час не існувало, але Apple ще з 1980-х прагнула високої графічної якості: наприклад, було впроваджено шрифти TrueType для гарного відображення тексту (Apple була однією з перших, хто забезпечив антиаліасінг тексту в 90-х). Проте справжня революція сталася з виходом Mac OS X 10.0 (2001). Ця UNIX-спадкоємиця класичного Mac OS отримала новий графічний архітектурний шар під назвою Quartz (Core Graphics), який складався з двох частин: Quartz 2D – бібліотека двовимірного графічного рендерингу (векторні шрифти, графіка, антиаліасінг), і Quartz Compositor – композиційний менеджер вікон [12]. Фактично Apple першими серед ОС масового користування впровадили повну композицію інтерфейсу: замість прямого малювання у framebuffer всі програми малювали у оффскрін-буфери, а спеціальний процес (Compositor) об'єднував їх на екрані. Це дало змогу реалізувати "візуальні примхи" нової теми Aqua: прозорі елементи, напівпрозорі тіні під вікнами, анімацію відкриття вікон, ефекти "джинна" при згортанні тощо – речі, небачені доти в широкому вжитку [5]. Усі ці ефекти вимагали альфа-композиції з глибинними буферами, що й робив Quartz Compositor. Однак, як згадано, перша реалізація працювала цілком на ЦП, оскільки GPU того часу (2001 р.) були доволі слабкими для таких задач [5]. В результаті продуктивність Mac OS X 10.0/10.1 була неконкурентною: інтерфейс відчувався млявим, анімації "посипались" кадрами. Це зафіксовано навіть у відгуках: "Aqua була нестерпно повільною та вимогливою до ресурсів" на старті [5]. Apple оперативно відреагувала: вже в Mac OS X 10.2 Jaguar (2002) було представлено технологію Quartz Extreme, яка перенесла рендеринг композиційного шару на

графічний процесор[14]. За вимогами, Mac з Jaguar повинен був мати 16 МБ VRAM і відеокарту з підтримкою текстур необхідного розміру (наприклад, ATI Radeon чи NVIDIA GeForce2 MX) для активації Quartz Extreme [13]. Якщо GPU відповідав критеріям, усі вікна відтепер складались GPU-засобами, що зняло навантаження з CPU і різко прискорило інтерфейс – він став плавним, порівнянним за швидкодією зі старими (докомпозитними) дизайнами [5].

Таким чином, Mac OS X вперше повністю задіяла можливості 3D-прискорювача для 2D-інтерфейсу, заклавши тренд на роки вперед.

Архітектура графічної підсистеми macOS надалі розвивалась у напрямку все більшої інтеграції різних графічних технологій. Quartz Compositor залишається центральним компонентом і до сьогодні (хоча тепер називається Core Graphics/Core Animation в маркетингових термінах). На рівні API Apple розширила можливості розробників: у 2005 р. з виходом Mac OS X 10.4 Tiger було представлено Quartz 2D Extreme (пізніше перейменований на QuartzGL) – механізм апаратного прискорення не тільки композиції, а й самого Quartz 2D-рендерингу вектора на GPU [12]. Хоча за замовчанням його тоді не ввімкнули через деякі графічні артефакти, сам факт досліджень у цьому напрямку показував прагнення максимально залучити GPU навіть для 2D-графіки. У Mac OS X 10.5 Leopard (2007) Apple зробила черговий крок, додавши Core Animation – вищий рівень API, що дозволяв розробникам легко створювати анімовані інтерфейси, віддаючи більшість важкої роботи (анімацію перетворень, зміни прозорості тощо) на GPU. Філософія була така: застосунок генерує шари інтерфейсу (можливо, рендерені традиційно CPU або GPU), а система анімує їхні позиції/розміри/прозорості на графічному процесорі зі швидкістю 60 fps [5]. Core Animation фактично виносила багато логіки UI в композитор, що працював апаратно. Цей підхід виявився настільки успішним (спочатку обкатаний в iOS на iPhone для плавності інтерфейсу), що Microsoft у Windows 8 додала аналогічний компонент DirectComposition для Metro-інтерфейсу [5].

З точки зору користувача macOS вже давно асоціюється з дуже плавним, “вивіреним” інтерфейсом. Після Quartz Extreme підвищення частоти кадрів і відгуку зробило взаємодію приємною навіть при складних ефектах. Наприклад, функція Exposé (вперше з’явилася в Mac OS X 10.3 Panther, 2003) – відображення всіх вікон на екрані з їх масштабуванням – працює швидко завдяки тому, що GPU може одночасно відрендерити десятки текстур вікон, змінюючи їхні розміри і позиції у реальному часі. Для комфортної роботи Exposé Apple прямо рекомендувала наявність Quartz Extreme-сумісного GPU [13]. Тобто вже у 2003 р. було чітко показано залежність можливостей UI від рівня графічного апаратного забезпечення. До середини 2010-х Apple повністю переходить на рендеринг інтерфейсу через GPU: інтерфейс Retina (високі DPI,

вперше MacBook Pro Retina у 2012) підтримується без втрати чіткості завдяки тому, що всі елементи UI – векторні або растрові з автоскейлом – малюються Core Graphics з антиаліасингом і легко масштабуються GPU-засобами. В останніх версіях macOS (з 2018 р.) Apple навіть відмовилася від OpenGL, замінивши його власним графічним API Metal: це стосується і внутрішнього рендерингу UI. Metal дає ще кращий доступ до можливостей GPU, зменшуючи накладні витрати. Хоч деталей реалізації Apple публічно не розкриває, можна зазначити, що compositor macOS (тепер відомий як WindowServer/SkyLight) працює поверх Metal, забезпечуючи оптимальне використання сучасних графічних процесорів на Mac.

Загалом, інтеграція комп'ютерної графіки в macOS є майже тотальною: кожен піксель інтерфейсу проходить через графічний конвеєр, оптимізований апаратно. Це дозволило Apple реалізувати багаті візуальні ефекти (наприклад, Mission Control – тривимірне розташування робочих просторів, Launchpad – плавна анімація іконок, згладжені трансформації при переходах між дисплеями тощо) без значних компромісів у швидкодії. macOS історично задала високу планку плавності GUI, що підтверджується навіть неформальними спостереженнями: користувачі відзначали, що візуальна плавність Mac часто вища, ніж у інших ОС на аналогічному залізі, завдяки "добре продуманому графічному стеку, де кожен елемент UI максимально використовує GPU". Цей "запас міцності" по графіці також сприяє довговічності дизайну – наприклад, інтерфейс залишається відгукливим навіть під навантаженням, оскільки графічні обчислення виконуються паралельно з основними на окремому процесорі. Відтак, досвід Apple демонструє, наскільки виграшною є стратегія глибокої інтеграції CG в ОС: вона забезпечує і гарну естетику, і кращу продуктивність UI.

Linux. В екосистемі Linux та Unix-подібних ОС еволюція графічних інтерфейсів відбувалася дещо іншим шляхом, але кінцева мета – використання комп'ютерної графіки для покращення UI – така ж сама. Класичний X Window System (X11), який понад 30 років був основою графічного середовища в UNIX/Linux, спроєктований за архітектурою клієнт–сервер. Дисплейний сервер X11 (X-сервер) відповідає за взаємодію з графічним обладнанням і керування вікнами, тоді як програми (клієнти) надсилають серверу команди малювання у стандартизованому протоколі [8]. Така модель, створена у 1980-х, мала на меті незалежність додатків від апаратури і навіть мережеву прозорість (можливість відображати додаток на віддаленому дисплеї), але вона принесла й складності. По-перше, X11 не мав ізольованих буферів для кожного вікна – усі клієнти малювали у спільний екран (за винятком невеликого буфера "заднього кадру" при використанні розширення Xcomposite). По-друге, тривалий час віконні менеджери в X були стековими (stacking WMs): вони оперували розташуванням

вікон, але не складали їх через GPU. Це означало, що візуальні ефекти були обмежені – наприклад, прозорість потребувала хаків, згладжування виконувалось на боці кожного додатка, а перемальовування при перекритті було потенційно повільним.

З появою 3D-прискорювачів розробники X створили розширення GLX та AIGLX для використання OpenGL всередині X-сервера, що відкрило двері для композитних менеджерів. Compiz (2006), як згадано, став першим популярним композитним менеджером вікон на Linux, що об'єднував менеджер вікон і композитора та рендерив все через OpenGL [9]. Compiz вимагав наявності 3D-драйвера (через Xgl або AIGLX) та фактично запровадив GPU-ефекти на робочому столі Linux: тривимірні переходи, анімації згортання вікон, ефект "wobbling" (гумові вікна), куб робочого столу тощо [9]. Це був доказ концепції, що Linux-десктоп може бути не гіршим за пропрietарні ОС у візуальній привабливості і що графічні карти можуть використовуватися не лише для ігор, але й для звичайної роботи. Багато ідей Compiz було пізніше вбудовано у середовища GNOME, KDE: з кінця 2000-х їхні власні менеджери вікон (Mutter у GNOME 3, KWin у KDE 4) отримали режими композиції OpenGL. Цікаво, що поява Clutter (OpenGL-інструментарію для створення GPU-інтерфейсів) у 2006 р. стала важливою віхою: Clutter, розроблений компанією OpenedHand (пізніше Intel), приніс апаратно прискорений рендеринг інтерфейсу (через сценограф, GObject API) у багато Linux-застосунків і сам лежить в основі Mutter (GNOME Shell). Це дозволило робочому столу GNOME Shell (з 2011 р.) мати такі ефекти, як плавне перемикання між віртуальними робочими столами, анімовані сповіщення, масштабування значків, тощо – усе рендерилось через OpenGL. Недоліком була вища вимогливість до GPU: слабкі прискорювачі або віртуальні машини могли погано працювати з GNOME 3/Unity саме через відсутність повноцінного OpenGL-прискорення. Утім, тенденція була однозначною: всі великі Linux-середовища перейшли на композитний режим з GPU до середини 2010-х.

Найсучасніший етап розвитку – це перехід від X11 до Wayland. Wayland не просто менеджер, а цілий протокол і стек, який радикально спрощує графічну підсистему. У Wayland архітектура клієнт–сервер зберігається, але дисплейний сервер = композитор: окрема програма (наприклад, Weston, Mutter, KWin в режимі Wayland) виступає одночасно і сервером, і композитним менеджером [10]. Клієнти (додатки) створюють свої вікна через бібліотеки (наприклад, EGL + OpenGL ES, Vulkan тощо) безпосередньо в буфер, після чого надсилають повідомлення композитору про готовність нових кадрів. Wayland-композитор вже відображає їх на екран, виконуючи композицію кадрів. Ключова відмінність – відсутність проміжного шару X-сервера з його протоколом, через що зменшується затримка і спрощується шлях пікселів до

екрану [8]. У цитаті з FAQ проекту зазначено, що багато функцій, які раніше мусили виконуватись у центральному X-сервері, тепер або перенесено в ядро Linux (KMS, DRM для управління пам'яттю і режимами екрану), або в бібліотеки (напр. Cairo, яка малює в пам'ять) – тому "центральному серверу майже нічого не залишилось робити, окрім власне композиції" [10]. Це і було ідеєю Wayland – "виштовхнути X з гарячого шляху між клієнтом і обладнанням".

На практиці перехід до Wayland ще триває (станом на 2025 р., багато дистрибутивів вже використовують Wayland за замовчанням у GNOME, KDE, але доступна і сумісність через XWayland). Однак переваги вже відчутні: зменшення затримки та відсутність розривів зображення. Наприклад, переміщення вікон чи анімації у Wayland відчуваються плавнішими, бо немає промальовування "на два кроки" (спочатку вікно малює X, потім композитору треба його перехопити). Крім того, Wayland з самого початку проектувався з урахуванням безпеки: програми не можуть "підглядати" ввід з клавіатури/миші одна одної чи малювати поверх інших довільно (те, що в X11 було можливе і потребувало громіздких виправлень [8]. З точки зору графічних можливостей, Wayland дозволив гнучкіше підтримувати сучасні фішки – динамічну частоту оновлення, масштабування інтерфейсу для HiDPI, комбінування апаратних і софтверних оверлеїв. Наприклад, в Linux з'явилась можливість легко впровадити фракційне масштабування (не 2×, а 1.5× і т.д. для HiDPI), яке в X11 вимагало значних зусиль, а у Wayland це штатно підтримується у протоколі. Бенчмарки також показують покращення: у деяких сценаріях (ігри, відтворення відео) на сучасному залізі Wayland дає дещо вищий FPS і відсутність розривів кадру, оскільки композитор краще синхронізує кадри з виведенням на дисплей (vsync) та може напряму керувати буферами без X11-прокладки. Окремо слід згадати, що розробники Linux-оточень активно працюють над оптимізацією своїх композиторів: впроваджується pipewire для ефективного запису екрану без копіювання, DMABUF для передачі буферів між процесами, зниження латентності вводу (наприклад, у Weston налаштовано спеціальний "режим repaint scheduling" для мінімізації затримки між вводом і кадром [5]). Хоч ці технічні деталі виходять за межі огляду, загальна картина така: Linux поступово досяг паритету з Windows/macOS у плані графічної основи UI, повністю перейшовши на GPU-прискорення і композицію.

Варто також відзначити, що відкритість платформи Linux призвела до появи різних інноваційних і експериментальних рішень у царині GUI. Наприклад, існують проекти, де весь інтерфейс програми рендериться вручну засобами GPU (минути X/Wayland). В співтоваристві Rust-розробників створюються GPU-орієнтовані UI-бібліотеки (як *egui*, *vulkan GUI*), що можуть працювати крос-платформено [6]. Хоча такі підходи поки що нішеві,

вони демонструють тенденцію: традиційні бібліотеки GUI (GTK, Qt) також поступово використовують GPU всюди, де це можливо. Наприклад, Qt 5/6 має сценограф на OpenGL для відмалювання віджетів, GTK4 перейшла на рендеринг через Vulkan/OpenGL. Таким чином, і на рівні додатків Linux-середовище прийняло ідею, що для досягнення високої плавності та відгуку UI потрібне апаратне прискорення. Як влучно зазначив один з інженерів: “нині GPU-прискорення фактично потрібне для хорошої продуктивності GUI” [6].

Windows, macOS та сучасні Linux усі використовують GPU та композицію для рендерингу інтерфейсу. Реалізації різняться тільки у деталях – свої графічні API, різні протоколи – але фундаментальний принцип спільний: інтерфейс малюється не безпосередньо, а через шар композиції/абстракції, який ефективно керує екранними ресурсами і забезпечує розширені графічні можливості. Кожна ОС пройшла свій шлях: Windows – від GDI до DWM і Direct2D, macOS – від софтверного Quartz до Quartz Extreme і далі, Linux – від X11 до Compiz і Wayland. У результаті сьогодні користувачі отримують багатий досвід: багатовіконні інтерфейси без мерехтіння, згладжений рендеринг шрифтів на будь-якому DPI, анімації на 60 fps, 3D-ефекти (в межах UX-дизайну) з мінімальним навантаженням. З точки зору практичних наукових та інженерних завдань, інтеграція комп’ютерної графіки в ОС вирішила проблему розриву між можливостями сучасного графічного обладнання та потребами людського сприйняття: було створено UI, які повністю задіюють апаратні ресурси для досягнення плавності, що задовольняє око і забезпечує інтуїтивність.

Висновки. У даному дослідженні проаналізовано еволюцію та сучасний стан інтеграції комп’ютерної графіки з операційними системами для покращення користувацького інтерфейсу. Історичний огляд показав, що перехід від програмного рендерингу GUI (на кшталт GDI, QuickDraw) до апаратно прискореного став визначальним етапом розвитку ОС: саме впровадження композиційних менеджерів вікон і використання GPU для відмалювання інтерфейсу дозволило подолати обмеження продуктивності і якості, властиві раннім системам.

Сучасні операційні системи (Windows, macOS, Linux) повною мірою інтегрують засоби комп’ютерної графіки у свій UI. Усі вони використовують композицію вікон: кожне вікно малюється у буфер і об’єднується системним композитором, що працює на графічному процесорі. Це забезпечує багаті графічні ефекти (прозорість, тіні, анімації) та високу швидкодію без артефактів перемальовування. Користувацький інтерфейс став більш привабливим і інтуїтивним, оскільки візуальні підказки та реакції системи тепер плавні і точні.

Інтеграція CG в усіх ОС призвела до уніфікації користувацького досвіду: тепер очікуваним є, що переміщення вікна або прокрутка сторінки відбувається

плавно і без затримок, незалежно від платформи. Апаратне прискорення GUI стало стандартом де-факто. Кожна система має внутрішні оптимізації, що дозволяють більш ефективно використовувати GPU. Наприклад, всі три ОС реалізували техніку подвійної буферизації кадрів для запобігання мерехтінню при оновленні вікон. Також повсюдною є підтримка згладжування (шрифтів і графіки) – користувацькі інтерфейси нині візуально гладенькі, криві лінії та кола малюються без сходинок, що стало можливим лише з достатньою GPU-міццю.

Незважаючи на досягнення, інтеграція графіки ставить і нові задачі. Одна з них – мінімізація латентності. Додатковий крок композитора додає затримку між дією користувача і відображенням її результату. Це критично у сферах VR/AR, кіберспорту, де боротьба йде за кожен мілісекунд. Вирішення бачать у подальшому вдосконаленні графічних стеків: використанні багатопланових оверлеїв (multiplane overlays), прямих сканованих буферів (direct scanout) для повноекранних вікон, оптимізації планування кадрів у реальному часі. Інший виклик – забезпечення сумісності: всі три платформи підтримують старі програми, які не знають про нові методи рендерингу. Тут застосовуються різні підходи: Windows має емуляцію GDI через DWM, Linux – XWayland, macOS – “synchronization updates” для додатків без Metal. В цілому ці механізми успішні, але потребують уваги, щоб старі додатки також працювали плавно на новому графічному ядрі.

Перспективи подальших досліджень у напрямі інтеграції комп’ютерної графіки з ОС пов’язані з новими технологічними тенденціями. Одним з найважливіших трендів є просторові інтерфейси і змішана реальність (AR/VR). Тут інтеграція графіки набуває нового виміру: операційна система повинна малювати не плоскі вікна, а тривимірні об’єкти в просторі навколо користувача. Яскравий приклад – анонсована Apple система visionOS для гарнітури Vision Pro, яку компанія називає початком “ери просторових обчислень”. У visionOS традиційні поняття вікон поєднуються з 3D-об’ємами, а композиторами стають цілі 3D-сценографи з рендерингом на двох дисплеях перед очима користувача. Інтеграція комп’ютерної графіки тут максимальна: GPU використовується на повну для відтворення реалістичних об’ємних інтерфейсів, врахування фізичного оточення (через камери і LiDAR) та підтримки високої частоти оновлення (90–120 FPS для VR). Подібні системи ставлять нові виклики – наприклад, забезпечення мінімальної затримки, аби уникнути дискомфорту (motion sickness), і синхронізація рендерингу з трекінгом рухів голови. Це буде полем активних досліджень на стику графіки та ОС у найближчі роки.

Ще один напрям – підвищення енергоефективності графічних інтерфейсів. Використання GPU, особливо в мобільних пристроях, впливає на витрату енергії. Тому йдуть роботи над динамічними частотами оновлення

(Promotion, VRR), “розумним” зниженням FPS в моменти бездіяльності, передачею частини композитних задач в спеціалізовані чипи (наприклад, в iPad є чип “Display Engine”).

На завершення, можна впевнено стверджувати, що інтеграція комп’ютерної графіки з операційними системами суттєво підняла планку можливостей для користувацького інтерфейсу. Завдяки тісній співпраці розробників системного ПЗ та графічного обладнання, сучасні GUI стали швидкими, красивими та функціональними. Подальший прогрес у цій галузі відкриватиме перед користувачами нові враження – від голографічних робочих столів до повного занурення у віртуальні інтерфейси – зберігаючи при цьому основний принцип: комп’ютерна графіка служить заради зручності і ефективності взаємодії людини з комп’ютером.

Література:

1. Wikipedia. Graphical user interface. https://en.wikipedia.org/wiki/Graphical_user_interface

2. Microsoft. The Desktop Window Manager. Microsoft Learn (Win32 apps documentation), updated Apr 27, 2021. https://learn.microsoft.com/en-us/windows/win32/learnwin32/the-desktop-window-manager

3. WDD Staff. Operating System Interface Design Between 1981–2009. Webdesigner Depot, 2009. https://webdesignerdepot.com/operating-system-interface-design-between-1981-2009/

4. Graphics Device Interface. Microsoft Windows API – Wikipedia. https://en.wikipedia.org/wiki/Graphics_Device_Interface

5. Levien, R. The compositor is evil. Personal blog, Sept 13, 2020. https://raphlinus.github.io/ui/graphics/2020/09/13/compositor-is-evil.html

6. Browsertech Digest (Paul). GPU-backed User Interfaces. Browsertech Digest Newsletter, Issue #12, Feb 7, 2023. https://digest.browsertech.com/archive/gpu-backed-user-interfaces/

7. Wikipedia. Desktop Window Manager - Architecture. https://en.wikipedia.org/wiki/Desktop_Window_Manager

8. Messina, G. Why Use Wayland versus X11? CBT Nuggets Networking/Technology Blog, Nov 2, 2021. https://www.cbtnuggets.com/blog/technology/networking/why-use-wayland-versus-x11

9. Emms, S. Compiz - OpenGL compositing manager. LinuxLinks, Sept 12, 2023. https://www.linuxlinks.com/compiz/

10. Wikipedia. Wayland (protocol). https://en.wikipedia.org/wiki/Wayland_(protocol)

11. Microsoft. Overview of the Windows Graphics Architecture. Microsoft Learn (Win32 apps documentation), updated Aug 23, 2022. https://learn.microsoft.com/en-us/windows/win32/learnwin32/overview-of-the-windows-graphics-architecture

12. Wikipedia. Quartz (graphics layer). https://en.wikipedia.org/wiki/Quartz_(graphics_layer)

13. InvGate ITDB. Mac OS X Panther - Basic Info and Technical Requirements. InvGate Asset Management Database, 2003. https://invgate.com/itdb/mac-os-x-panther

14. Proven, L. GNOME Project retires OpenGL rendering library Clutter. The Register, Feb 18, 2022. https://www.theregister.com/2022/02/18/clutter_gnome_retired/

References:

1. Wikipedia. Graphical user interface. Retrieved from: https://en.wikipedia.org/wiki/Graphical_user_interface
2. Microsoft. The Desktop Window Manager. Microsoft Learn (Win32 apps documentation), updated Apr 27, 2021. Retrieved from: https://learn.microsoft.com/en-us/windows/win32/learnwin32/the-desktop-window-manager
3. WDD Staff. Operating System Interface Design Between 1981–2009. Webdesigner Depot, 2009. Retrieved from: https://webdesignerdepot.com/operating-system-interface-design-between-1981-2009/
4. Graphics Device Interface. Microsoft Windows API – Wikipedia. Retrieved from: https://en.wikipedia.org/wiki/Graphics_Device_Interface
5. Levien, R. The compositor is evil. Personal blog, Sept 13, 2020. Retrieved from: https://raphlinus.github.io/ui/graphics/2020/09/13/compositor-is-evil.html
6. Browsertech Digest (Paul). GPU-backed User Interfaces. Browsertech Digest Newsletter, Issue #12, Feb 7, 2023. Retrieved from: https://digest.browsertech.com/archive/gpu-backed-user-interfaces/
7. Wikipedia. Desktop Window Manager - Architecture. Retrieved from: https://en.wikipedia.org/wiki/Desktop_Window_Manager
8. Messina, G. Why Use Wayland versus X11? CBT Nuggets Networking/Technology Blog, Nov 2, 2021. Retrieved from: https://www.cbtnuggets.com/blog/technology/networking/why-use-wayland-versus-x11
9. Emms, S. Compiz - OpenGL compositing manager. LinuxLinks, Sept 12, 2023. Retrieved from: https://www.linuxlinks.com/compiz/
10. Wikipedia. Wayland (protocol). Retrieved from: https://en.wikipedia.org/wiki/Wayland_(protocol)
11. Microsoft. Overview of the Windows Graphics Architecture. Microsoft Learn (Win32 apps documentation), updated Aug 23, 2022. Retrieved from: https://learn.microsoft.com/en-us/windows/win32/learnwin32/overview-of-the-windows-graphics-architecture
12. Wikipedia. Quartz (graphics layer). Retrieved from: https://en.wikipedia.org/wiki/Quartz_(graphics_layer)
13. InvGate ITDB. Mac OS X Panther - Basic Info and Technical Requirements. InvGate Asset Management Database, 2003. Retrieved from: https://invgate.com/itdb/mac-os-x-panther
14. Proven, L. GNOME Project retires OpenGL rendering library Clutter. The Register, Feb 18, 2022. Retrieved from: https://www.theregister.com/2022/02/18/clutter_gnome_retired/