ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

УДК 623.438:623.592:004.94 doi:10.20998/2413-4295.2025.02.08

АЛГОРИТМ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ БЕЗПОСЕРЕДНЬОГО РЕЖИМУ ДЛЯ СИСТЕМ ВІЗУАЛІЗАЦІЇ

А. О. ЗУЄВ*, Д. Г. КАРАМАН, О. М. ЄВСЕЄНКО

Кафедра «Автоматика та управління в технічних системах», НТУ «ХПІ», Харків, УКРАЇНА
**e-mail: andrii.zuiev@khpi.edu.ua*

АНОТАЦІЯ *Проведено глибокий аналіз підходів до побудови графічного інтерфейсу користувача із застосуванням двох основних режимів: збереженого та безпосереднього. Особливу увагу приділено принципам функціонування та реалізації безпосереднього режиму, який характеризується динамічністю та відсутністю необхідності збереження стану функціонально повних візуальних конструкцій (віджетів). Відзначено переваги такого підходу, зокрема спрощення структури коду, зменшення витрат на пам'ять, підвищення швидкодії побудови та відображення в реальному часі (рендерінгу) та можливість динамічної перебудови інтерфейсу залежно від змін у програмі. Запропоновано алгоритм побудови графічного інтерфейсу користувача у безпосередньому режимі, який забезпечує одночасний рендерінг віджетів і обробку подій введення, мінімізуючи ресурсоємність та складність програмної реалізації. Проаналізовано ключові аспекти функціонування такого інтерфейсу, зокрема: відсутність потреби у синхронізації даних між віджетами та основною програмою, адаптивне вирівнювання елементів інтерфейсу на екрані, підтримка динамічних списків і таблиць із прокруткою, а також ефективна обробка вводу від користувача, зокрема з непозиційних пристроїв, таких як геймпади чи джойстики. Особливу увагу приділено практичним аспектам застосування безпосереднього режиму для спеціалізованих програм, що працюють у реальному часі або на обмежених обчислювальних системах, таких як мікроконтролери. Як приклад реалізації розглянуто графічний інтерфейс для програми збору даних із бездротових інерційних сенсорів WitMotion BWT901, яка використовує запропонований алгоритм для динамічного відображення поточних показників сенсорів та вибору пристроїв зі списку активних з'єднань. Окремо розглянуто проблеми відображення підказок при наведенні курсора, адаптивне центрування групи віджетів на екрані, а також механізми сортування та обробки взаємодії з маркерами в системах тривимірної візуалізації, що мають ключове значення для графічних інтерфейсів тренажерних комплексів і HUD-дисплеїв. Запропонований підхід забезпечує гнучкість у розробці інтерфейсів різної складності та ефективне використання апаратних ресурсів, включаючи графічні прискорювачі. Висновки акцентують увагу на тому, що безпосередній режим графічного інтерфейсу користувача має значні переваги над традиційним збереженим режимом у багатьох прикладних сценаріях, де важливими є швидкодія, простота реалізації та мінімальні вимоги до апаратних ресурсів. Реалізація запропонованого алгоритму може знайти застосування в наукових додатках, тренажерних комплексах, системах збору даних із сенсорів, а також у пристроях IoT і вбудованих системах, де ефективність роботи елементів графічного інтерфейсу є критичною.*
Ключові слова: *графічний інтерфейс користувача; віджети; людинно-машинний інтерфейс; тренажерні комплекси; візуалізація даних*

ALGORITHM AND SOFTWARE IMPLEMENTATION OF IMMEDIATE MODE INTERFACE FOR VISUALIZATION SYSTEMS

A. ZUEV*, D. KARAMAN, O. YEVSEIENKO

Department of Automation and Control in Technical Systems, NTU "KhPI", Kharkiv, UKRAINE
**e-mail: andrii.zuiev@khpi.edu.ua*

ABSTRACT *A deep analysis of approaches to building graphical user interfaces has been conducted, focusing on two main modes: retained and immediate. Special attention is given to the principles of operation and implementation of the immediate mode, which is characterized by dynamism and the absence of the need to store the state of functionally complete visual constructs (widgets). The advantages of this approach are noted, including simplification of code structure, reduction of memory consumption, improvement of real-time building and rendering performance, and the ability to dynamically rebuild the interface depending on changes in the program. An algorithm for building a graphical user interface in immediate mode is proposed, which provides simultaneous rendering of widgets and processing of input events, minimizing resource consumption and complexity of software implementation. Key aspects of such an interface's operation are analyzed, including: no need for data synchronization between widgets and the main program, adaptive alignment of interface elements on the screen, support for dynamic scrollable lists and tables, as well as efficient processing of user input, particularly from non-positional devices such as gamepads or joysticks. Special attention is paid to practical aspects of applying immediate mode for specialized programs operating in real-time or on limited computing systems, such as microcontrollers. As an implementation example, the graphical interface for a data collection program from WitMotion BWT901 wireless inertial sensors is examined, which uses the proposed algorithm for dynamic display of current sensor readings and selection of devices from a list of active connections. Separately, the problems of displaying tooltips on cursor hover, adaptive centering of widget groups on the screen, as well as mechanisms for sorting and processing interactions with markers in three-dimensional visualization systems, which are crucial for graphical interfaces of training complexes and HUD displays, are considered. The proposed approach provides flexibility in developing interfaces of varying complexity and efficient use of hardware*

© А. О. ЗУЄВ, Д. Г. КАРАМАН, О. М. ЄВСЕЄНКО, 2025

resources, including graphics accelerators. The conclusions emphasize that the immediate mode of the graphical user interface has significant advantages over the traditional retained mode in many application scenarios where performance, implementation simplicity, and minimal hardware requirements are important. Implementation of the proposed algorithm can find application in scientific applications, training complexes, sensor data collection systems, as well as IoT devices and embedded systems, where the efficiency of graphical interface elements is critical.
***Keywords**: graphical user interface; widgets; human-machine interface; training simulators; data visualization.*

Вступ

Графічний інтерфейс користувача (GUI) спрощує користування комп'ютером, представляючи інформацію у спосіб, який дозволяє швидко її засвоювати та використовувати для ефективної взаємодії з функціональними елементами обчислювальної системи [1]. Використання візуальних конструкцій (віджетів), які імітують фізичні об'єкти: «перемикачі» та «кнопки», може прискорити навчання, шляхом надання інтуїтивно зрозумілого методу введення в комп'ютер [2]. Добре розроблений, інтуїтивно зрозумілий дизайн GUI не вимагає від користувачів спеціально запам'ятовувати кроки для виконання необхідних дій, що особливо важливо у застосуванні наукових програм, де головною метою для користувача є зосередження на взаємодії з об'єктом дослідження, що забезпечується за посередництвом програми, та робота з даними, а не опанування послідовності складних кроків та взаємодій, необхідних для досягнення бажаного результату [3].

Графічні інтерфейси користувача (GUI) відіграють важливу роль у системах візуального навчання та віртуальних імітаційно-тренажерних комплексах, забезпечуючи ефективну взаємодію користувача із програмним середовищем. Їх використання сприяє підвищенню наочності та інтерактивності навчальних процесів, дозволяє створювати реалістичні сценарії для відпрацювання практичних навичок і моделювання складних ситуацій. Елементи GUI для таких комплексів мають забезпечувати максимальний ступінь імерсивності, а також реалізувати інтуїтивно зрозумілий доступ до функцій і завдань системи, підвищуючи ефективність навчання завдяки зручному розташуванню елементів управління та інтерактивним візуальним підказкам.

Особливо актуальною є інтеграція новітніх технологій, таких як віртуальна (VR) і доповнена реальність (AR). Використання VR дозволяє повністю занурити користувача у віртуальне середовище, де можна виконувати навчальні завдання без ризику для здоров'я або пошкодження обладнання. Віртуальні тренажери створюють реалістичні моделі об'єктів і процесів, забезпечуючи користувачам досвід, максимально наближений до реального. GUI у VR-системах зазвичай розташовується в тривимірному просторі та може включати плаваючі панелі управління, віртуальні інструменти й контекстні меню, що полегшує взаємодію з навчальним середовищем. AR, у свою чергу, дозволяє накладати віртуальні елементи на реальні об'єкти, створюючи додаткові шари інформації. Наприклад, AR може відображати інструкції або схеми поверх реального обладнання, що значно спрощує навчання й виконання практичних завдань. Завдяки використанню мобільних додатків або AR-окулярів, інтерфейси забезпечують миттєвий доступ до інформації без необхідності відволікатися від основного процесу.

Мета роботи

Метою цієї статті є практична реалізація графічного інтерфейсу користувача, в безпосередньому режимі для дослідницьких програм, наприклад програм збирання даних з сенсорів, та для систем візуалізації тренажерних комплексів, тобто тих застосунків, які не потребують значної складності віджетів для побудови інтерфейсів, але потребують максимальної ефективності з точки зору використання ресурсів для реалізації функціонування графічного інтерфейсу користувача.

Огляд існуючих рішень та опублікованих робіт

Традиційний підхід до реалізації графічного інтерфейсу користувача (GUI) базується на використанні віджетів, які прив'язуються до об'єктів функціональної частини програмного забезпечення. Цей підхід, відомий як «GUI у збереженому режимі» (retained mode), має низку характерних особливостей (рис. 1).

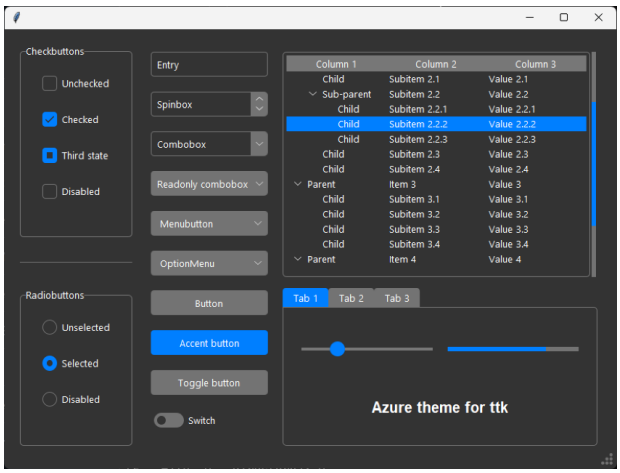

Рис. 1 – Приклад графічного інтерфейсу користувача, який базується на принципі «retained mode GUI» (бібліотека Tkinter)

Зокрема, він передбачає необхідність ручної синхронізації даних програми з компонентами

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

інтерфейсу [4], а також використання процедур зворотного виклику для обробки користувацького введення. Така модель реалізована в багатьох популярних бібліотеках для створення GUI, наприклад, Qt, wxWidgets або Tkinter [5]. Для спрощення опису складних схем інтерфейсів було навіть розроблено спеціалізовані мови програмування [6]. Інструментарій GUI, що базуються на принципі роботи у збереженому режимі, має кілька важливих переваг, які забезпечують його популярність у розробці програмних продуктів: широкий набір готових віджетів та налаштувань з їх оформлення та адаптації до умов використання, модульність, ієрархічна організація та можливість повторного використання. Особливо варто відмітити те, що уся робота по налаштуванню, побудові сцени та рендерингу віджетів, а також усі взаємодії між віджетами виконується засобами бібліотеки, що зменшує кількість низькорівневого коду по роботі з графікою у основній програмі і дозволяє розробнику зосередитися на створенні основного функціоналу.

Попри поширеність та функціональність підходу збереженого режиму, йому притаманні певні недоліки. Серед основних проблем можна виділити складність опису структури інтерфейсу, необхідність постійної синхронізації стану віджетів з логікою програми, а також значне споживання ресурсів під час обробки інтерфейсних подій [7]. Ці фактори стають критичними для спеціалізованих або вузькоспецифічних програмних рішень, зокрема тих, що функціонують на системах із обмеженими апаратними ресурсами, наприклад, на мікроконтролерах [8]. В таких умовах традиційний підхід реалізації GUI у збереженому режимі часто виявляється недостатньо ефективним.

Альтернативним підходом до збереженого режиму є графічний інтерфейс безпосереднього режиму (immediate mode), який відрізняється від збереженого тим, що віджети інтерфейсу не мають стану. На кожному кадрі основна програма задає, що саме відображати на екрані. Системи графічного інтерфейсу безпосереднього режиму широко використовуються в іграх та різноманітних спеціалізованих програмах з графічним інтерфейсом, наприклад, симуляторах та комп'ютерних тренажерних комплексах [9]. Також такий підхід широко використовуються в дослідницьких додатках, наприклад, для розподілених обчислень у реальному часі для CERN, інтерфейсів користувача спеціального призначення [10], для візуалізації архітектури коду [11], середовищ GUI для редагування та аналізу складних булевих функцій [12], а також для реалізації інтерфейсів взаємодії з об'єктами автоматизації у системах оперативного контролю та збору даних в реальному часі [13] (рис. 2). До переваг такого режиму відноситься відсутність стану, і, як наслідок — необхідності синхронізації та копіювання даних, а також обробка вводу, яка може бути відокремлена від самого рендеру інтерфейсу. Програмна реалізація інтерфейсу в такому режимі потребує значно менше коду та має нижчу складність, а також дозволяє динамічну перебудову схеми віджетів.

Рис. 2 – Приклади графічного інтерфейсу користувача, які базуються на принципі «immediate mode GUI»

Але слід також зазначити недоліки, властиві такому підходу. Серед основних проблем можна виділити складність реалізації інтерфейсу для дизайнерів, оскільки безпосередній режим вимагає більш глибокого розуміння програмування графічного середовища. Це значно ускладнює процес розробки для спеціалістів, які не мають достатніх технічних знань. Крім того, на відміну від традиційного підходу зі збереженим режимом, для такого типу інтерфейсів бракує повноцінного інструментарію (тулкіту), що обмежує можливості створення складних багаторівневих графічних інтерфейсів і збільшує час на їх реалізацію.

Найбільш відомою бібліотекою для розробки графічного інтерфейсу в безпосередньому режимі є «Dear ImGui» [4]. Вона широко застосовується у високопродуктивних середовищах, де важливими є швидкодія, мінімізація затримок та оптимізація використання ресурсів. Наприклад, «Dear ImGui» використовується в популярній системі 3D-моделювання Blender для створення внутрішніх інструментів та інтерфейсних елементів, які забезпечують інтерактивну взаємодію користувача з віртуальними об'єктами. Попри свою ефективність і гнучкість, ця бібліотека все ще орієнтована переважно на розробників і не пропонує інструментів, які б значно полегшили роботу дизайнерам або користувачам без досвіду програмування.

Основні принципи функціонування інтерфейсу безпосереднього режиму

Програми з інтерфейсом, що працює у безпосередньому режимі, на кожному кадрі оновлення графічної системи повністю відтворюють

усі елементи GUI у вигляді, який має бути представлений користувачеві. Такий підхід забезпечує високу гнучкість адаптації до змін у програмі та істотно відрізняється від підходу збереженого режиму, де кожен кадр потребує окремого вирішення щодо створення, оновлення або видалення віджетів на основі дескрипторів, якими керує основна програма.

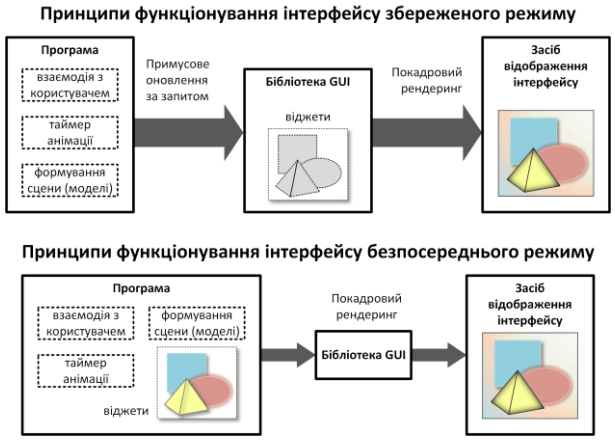

Рис. 3 – Відмінності у роботі GUI в збереженому та безпосередньому режимах

Введені користувачем дані повинні бути оброблені для забезпечення реакції на події, а актуальний набір видимих віджетів передається серверній частині візуалізації. Ця комунікація спирається на геометричні примітиви, що використовуються для відтворення текстових зображень, які складають графічний інтерфейс, враховуючи додаткову стилізацію віджетів. Кожен віджет має унікальний ідентифікатор, який система інтерфейсу застосовує для збереження його стану. Розташування та розміри віджетів визначаються відповідно до схем інтерфейсу, які можуть варіюватися за складністю: від простих, що вимагають одного проходу для обробки, до складних, які потребують кількох циклів опрацювання.

В безпосередньому режимі, така складна процедура обробки віджетів не потрібна, так як схема інтерфейсу будується в процесі візуалізації, і фактично цей режим реалізує схему «що відображено, те й обробляється», яка дуже спрощує всі процедури пов’язані з обробкою елементів інтерфейсу, і не потребує ранжування та впорядкування віджетів, тому що це відбувається природнім шляхом в процесі рендерингу віджетів. Також спрощується процес візуалізації нестандартних віджетів, наприклад, які є частиною системи головного дисплею (HUD) в системі візуалізації тренажерного комплексу [9]. Такі віджети вже не потребують якоїсь спеціалізованої системи обробки або рендерингу, бо вона виконується безпосередньо в процесі формування поточного кадру у системі візуалізації. Теж саме стосується візуалізації різноманітних маркерів, які пов’язані з системою 3D-візуалізації для відображення позакабінної обстановки тренажерного комплексу.

За наявності в програмі засобів візуалізації геометричних примітивів, вона здатна безпосередньо відтворювати класичні віджети, такі як кнопки, перемикачі та інші елементи інтерфейсу. Команди рендерингу можуть легко трансформуватися у список інструкцій, сумісний із графічним процесором (GPU), що включає буфер вершин, індекси, відповідні текстури та команди малювання. Це забезпечує ефективний рендеринг інтерфейсу безпосереднього режиму із залученням ресурсів графічного процесора, що значно підвищує його продуктивність.

Алгоритм побудови схеми інтерфейсу з одночасним рендерингом

Для реалізації рендерингу адаптивної схеми інтерфейсу з використанням методу безпосереднього режиму потрібно визначити контекст інтерфейсу — структуру даних, яка буде зберігати поточний стан інтерфейсу: налаштування відображення (стилю) в активному та пасивному вигляді, позицію керування (наприклад, позицію курсору миші), та ідентифікатор активного контролю, якщо такий знайдено.

Перед початком рендерингу така структура ініціалізується значеннями за замовченням, ідентифікатор події активного контролю виставляється в неактивне значення. Після чого проводиться безпосередньо процедура рендерингу віджетів з перевіркою області віджета на факт потрапляння до неї позиції керування. Якщо такий факт виявлено, запам’ятовується ідентифікатор події віджету, та його малювання здійснюється за допомогою активного стиля, в іншому випадку, малювання здійснюється із застосуванням пасивного стиля. Таким чином після рендерингу всіх віджетів, в структурі контексту збережеться індентифікатор події верхнього віджету до якого було спрямовано взаємодію користувача, якщо така взаємодія мала місце. Схема алгоритму, що забезпечує цей процес, зображена на рис. 4.

На етапі обробки подій введення виконується перевірка на наявність активного віджета та обробка ідентифікатора його події, збереженого в контексті. Такий підхід забезпечує повну консистентність обробки інтерфейсу: доки нова схема інтерфейсу не буде візуалізована, обробка подій продовжується відповідно до попередньої схеми, яку користувач бачить на екрані. Це гарантує синхронність між візуалізацією та функціональністю інтерфейсу, усуваючи потенційні розбіжності між станами системи.

Для виводу маркерів які пов’язані з тривимірною системою візуалізації потрібно попередньо виконати проєкцію їх центрів у простір екрану, запам’ятавши відстань об’єктів з якими асоційований маркер від спостерігача. Після цього маркери сортуються за цією відстанню від найбільш

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

віддаленого до найближчого (за зменшенням відстані). Рендеринг маркерів в такому ж порядку забезпечить коректний вибір найближчого до спостерігача маркеру як звичайного віджету інтерфейсу. Якщо потрібно мінімізувати перетин маркерів в площині екрану для покращення зручності користування інтерфейсом, попередньо потрібно відсортувати маркери по позиції одній з координат на екрані. Після цього маркери, площини яких перетинаються, можуть бути зміщені так, щоб уникнути накладання. У результаті створюються кластери маркерів, розташовані послідовно, і в межах кожного кластера здійснюється корекція координат, що дозволяє зменшити візуальні конфлікти між елементами інтерфейсу та забезпечити кращу оглядовість.

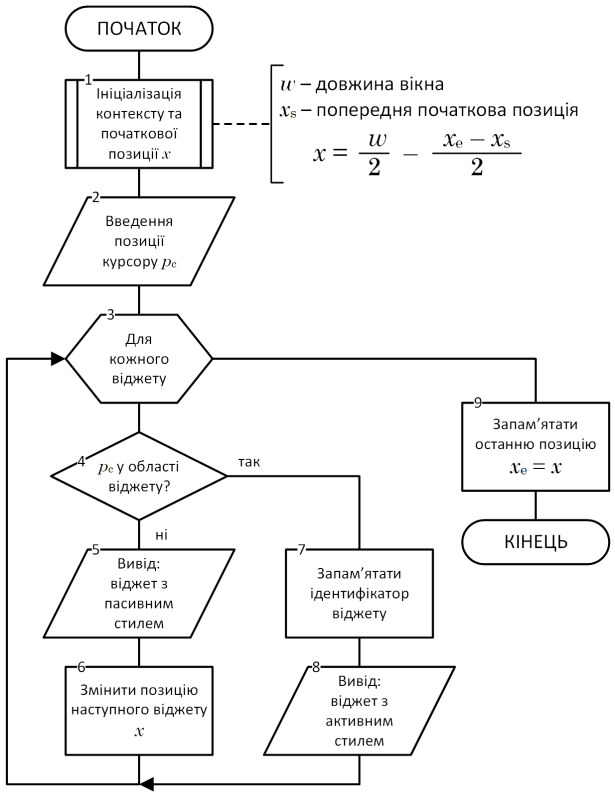

Рис.4 – Схема алгоритму побудови схеми інтерфейсу та рендерингу

Дистанцію до спостерігача також можна використовувати в якості коефіцієнта управління напівпрозорістю віджетів, для того щоб повільно приховувати найбільш віддалені від спостерігача маркери.

Обробка вводу користувача

Важливою частиною будь-якого інтерфейсу є система вводу від користувача, яка забезпечує зворотній зв'язок, і без якої не можливе повноцінне функціонування будь-якого інтерфейсу. Для безпосереднього режиму така система може бути фактично відокремлена від системи візуалізації, що є ще однією значною перевагою використання такого режиму. Для обробки вводу користувача не потрібно використовувати функції зворотного виклику. При цьому сама система вводу залишається незмінною і реалізується через чергу повідомлень від пристроїв вводу-виводу [14], а в контексті інтерфейсу запам'ятовуються лише ті повідомлення, які є релевантними для активного віджету. Оскільки в процесі візуалізації схема інтерфейсу, по суті, повністю перестворюється, адаптуючись під поточний стан програми, то вибір коректного віджета, з яким взаємодіє користувач, реалізується автоматично. Це фактично дозволяє «знайти» віджет, до якого спрямовано подію керування зі складністю O(1), проти складності для збереженого режиму, що оцінюється як O(N) – O(logN).

Фактично ідентифікатор віджета, який використовується для опрацювання вводу користувача, може співпадати з кодом «гарячої» клавіші або комбінації клавіш, що використовується для активації співставленої з віджетом функції. Це дозволяє використовувати для обробки вводу від користувача той самий код, який використовується для опрацювання натискання «гарячих» клавіш. Такий підхід до обробки вводу користувача відрізняється від традиційного підходу, який використовується у реалізаціях безпосереднього режиму інтерфейсу, наприклад Dear ImGui або Nuklear, де обробка активного віджету відбувається безпосередньо після його рендерингу.

Практична реалізація та обговорення

Практична реалізація безпосереднього режиму інтерфейсу була розроблена для програми збору даних з інерційних сенсорів WitMotion BWT901 через бездротовий інтерфейс Bluetooth. Програма призначена для пошуку та підключення активних бездротових сенсорів, та відображення даних які надходять з них в графічному форматі. Інтерфейс програми складається з двох частин: вибір сенсору зі списку та відображення даних які надходять з підключених сенсорів (рис. 5).

Незважаючи на очевидні переваги безпосереднього режиму відтворення інтерфейсу, варто окремо розглянути його специфічні особливості та можливі обмеження.

По-перше, це стосується реалізації підказок (tips), які з'являються при наведенні курсора миші на віджет. Для забезпечення такого функціоналу необхідно зберігати ідентифікатор попереднього активного віджета та час його останньої зміни. Під час оновлення кадру система перевіряє, чи змінився активний віджет. Якщо змін не відбулося, і час перебування курсора на поточному віджеті перевищує заданий поріг, можна відобразити текст підказки. Це слід виконувати після рендерингу всіх інших віджетів, щоб гарантувати, що підказка

накладатиметься поверх решти елементів інтерфейсу. Для відліку часу доцільно застосовувати монотонний годинник (steady clock) зі стандартної бібліотеки мови програмування, що використовується для створення інтерфейсу, або відповідний засіб операційної системи.

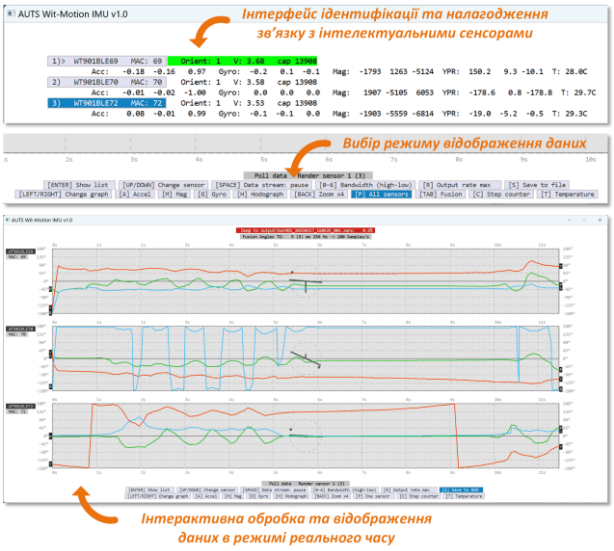

Рис. 5 – Загальний вигляд інтерфейсу розробленої програми

Незважаючи на очевидні переваги безпосереднього режиму відтворення інтерфейсу, варто окремо розглянути його специфічні особливості та можливі обмеження.

По-перше, це стосується реалізації підказок (tips), які з'являються при наведенні курсора миші на віджет. Для забезпечення такого функціоналу необхідно зберігати ідентифікатор попереднього активного віджета та час його останньої зміни. Під час оновлення кадру система перевіряє, чи змінився активний віджет. Якщо змін не відбулося, і час перебування курсора на поточному віджеті перевищує заданий поріг, можна відобразити текст підказки. Це слід виконувати після рендерингу всіх інших віджетів, щоб гарантувати, що підказка накладатиметься поверх решти елементів інтерфейсу. Для відліку часу доцільно застосовувати монотонний годинник (steady clock) зі стандартної бібліотеки мови програмування, що використовується для створення інтерфейсу, або відповідний засіб операційної системи.

По-друге, окремої уваги потребує реалізація таблиць і списків із лінійками прокрутки. Функціонально лінійка прокрутки є частиною стану таблиці, а не окремим віджетом, тому її стан слід зберігати та контролювати у основній програмі. На основі поточного положення лінійки генеруються віджети лише для тих елементів таблиці або списку, які наразі перебувають у межах видимої області екрану. Однак така оптимізація не повністю вирішує проблему прокрутки за допомогою, наприклад, колеса миші.

Для коректної обробки подібних сценаріїв у контекст інтерфейсу доцільно додавати технічні елементи, які не відображаються на екрані, але мають власні ідентифікатори подій і площини, що визначають їхні межі в координатах інтерфейсу. Під час обробки вводу такі елементи перевіряються на факт взаємодії з користувачем. Активний технічний елемент запам'ятовується та обробляється окремо, що дозволяє забезпечити правильну роботу механізмів прокрутки списків і таблиць. Завдяки цьому підходу система може точно визначити, які події стосуються прокрутки, і коректно реагувати на дії користувача навіть у складних випадках.

Ще однією значною складністю при реалізації інтерфейсу в безпосередньому режимі є відображення віджетів із вирівнюванням на екрані. Зазвичай застосовуються два основних методи адаптації їх розташування: використання прихованих проходів або виконання кількох проходів рендерингу інтерфейсу. У першому випадку віджети залишаються неактивними, доки їхня остаточна позиція не буде визначена або скоригована. Другий метод передбачає декілька циклів рендерингу і використовується, наприклад, у системі інтерфейсу рушія Unity [15]. Однак через відсутність збереженого стану в безпосередньому режимі (зокрема, інформації про положення віджетів) виникають труднощі, коли потрібно забезпечити вирівнювання групи елементів, наприклад, по центру екрана. Для вирішення цієї проблеми пропонується алгоритм, що використовує запам'ятовування ключових координат: під час першого проходу слід зберегти координати початку x_s та кінця x_e виведення групи віджетів. Під час наступного проходу можна обчислити різницю між ними, що відповідає загальній ширині групи на екрані. Ця інформація дозволить скоригувати розташування елементів, забезпечивши їхнє вирівнювання по центру або за іншими параметрами.

Такий підхід надає можливість досягти більшої гнучкості у створенні користувацьких інтерфейсів навіть за відсутності збереженого стану, зберігаючи при цьому ефективність обробки та швидкість рендерингу, властиві безпосередньому режиму. Координату початку виводу групи x можна отримати наступним чином

$$x = \frac{w}{2} - \frac{(x_e - x_s)}{2}, \qquad (1)$$

де w – довжина екрану (або вікна виводу).

Початкове значення x можна брати будь яке, наприклад, 0. Такий алгоритм дозволяє динамічно адаптуватися до будь-якої довжини групи. Для того щоб запобігти випадковій активації елементів при зміні, або при початковому розрахунку позиції, доцільно при зміні значення x між кадрами, прибирати відмітку активного віджету на один кадр.

ISSN 2079-5459 (print)
ISSN 2413-4295 (online)

Варто відзначити, що серйозною проблемою стає обробка вводу, якщо користувач використовує не позиційні пристрої на кшталт тачпада чи пари миша-клавіатура, а, наприклад, геймпад або джойстик. В цьому випадку фактично немає чітко визначеної позиції керування, що в свою чергу ускладнює визначення активного віджета до якого відбувається ввід і потребує його запам'ятовування та ітеративної зміни за командами користувача. У традиційному режимі роботи інтерфейсу активний віджет природньо запам'ятовується та його зміна може бути реалізована передбачуваним способом, наприклад, в залежності від того в якому напрямку користувач перемістив стік геймпаду, оскільки реалізація інтерфейсу має уяву про розташування будь-якого віджету на екрані, але в безпосередньому режимі така інформація на етапі обробки вводу вже не є доступною.

Висновки

Графічні інтерфейси можуть створюватися як у безпосередньому, так і в збереженому режимах, однак вони суттєво відрізняються за складністю реалізації. Безпосередній режим є більш гнучким і вимагає менше ресурсів, що критично для мікропроцесорних пристроїв з обмеженою продуктивністю та невеликим обсягом пам'яті. Важливою перевагою цього підходу є майже постійний час рендерингу одного й того ж інтерфейсу від кадру до кадру.

Крім того, реалізація безпосереднього режиму зазвичай потребує менше коду та має нижчу складність завдяки розділенню процесів візуалізації та обробки вводу. Це робить його більш доцільним у багатьох випадках. Такий підхід забезпечує гнучку візуалізацію спеціальних віджетів без необхідності додаткових механізмів з боку програмного інтерфейсу.

Запропонований алгоритм дозволяє створювати графічні інтерфейси різної складності та підтримує різні типи віджетів у реальному часі. Він мінімізує розмір вихідного коду і споживання системних ресурсів завдяки відсутності потреби запам'ятовувати стан віджетів. На відміну від інших підходів, алгоритм розділяє рендеринг і обробку вводу та усуває залежність від зворотних викликів чи абстрактних інтерфейсів, які зазвичай потрібно реалізовувати.

Список літератури

1. Dix A., Finlay J., Abowd G. D., Beale R. *Human-Computer Interaction.* Pearson: Prentice Hall, 2005. 834 p.
2. *User Interface Design. Case study: UX/UI design for complex technical simulation software with professional users*. *Interface Design*. URL: https://interface-design.co.uk/case-studies/ux-ui-design-technical-software-users (дата звернення: 26.03.2025).
3. Paine D., Ghoshal D., Ramakrishnan L. Experiences with a Flexible User Research Process to Build Data Change Tools. *Journal of Open Research Software*. 2020. Vol. 8(1). P. 18. doi: 10.5334/jors.284.
4. Liedtke S., Brendel F. Exploring the immediate mode GUI concept for graphical user interfaces in mixed reality applications. *GI VR/AR Workshop, Gesellschaft für Informatik e.V.* 2022. doi: 10.18420/vrar2022_1678.
5. Essick J. *The Python-Based Laboratory: A Hands-On Guide for Scientists and Engineers. Chapter 2. Graphical User Interface Using Tkinter*. Oxford: Oxford Academic, 2024. pp. 14–32. doi: 10.1093/9780191998478.003.0002.
6. Stokke K. A., Barash M., Järvi J. A domain-specific language for structure manipulation in constraint system-based GUIs. *Journal of Computer Languages.* 2023. Vol. 74(3). P. 101175. doi: 10.1016/j.cola.2022.101175.
7. Mondal S. K., Pei Y., Dai H. N., Dipu Kabir H. M., Sahoo J. P. Boosting UI Rendering in Android Applications. *IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C),* Macau, China. 2020. P. 285-286. doi: 10.1109/QRS-C51114.2020.00055.
8. Zuev A., Karaman D., Olshevskiy A. Wireless Sensor Synchronization Method for Monitoring Short-Term Events. *Advanced Information Systems*. 2023. Vol. 7(4). P. 33–40. doi: 10.20998/2522-9052.2023.4.04.
9. Зуєв А. О., Караман Д. Г., Євсеєнко О. М. *Системи моделювання та візуалізації імітаційно-тренажерних комплексів: монографія.* Харків: Секішова Т. Є., 2023. 223 с.
10. Knoblauch J., Sethuraman A., Hey J. IMGui — a desktop GUI application for isolation with migration analyses. *Molecular Biology and Evolution*. 2017. Vol. 34, no. 2. P. 500–504. doi: 10.1093/molbev/msw252.
11. Olsson T., Ericsson M., Wingkvist A. An exploration and experiment tool suite for code to architecture mapping techniques. *In Proceedings of the 13th European Conference on Software Architecture (ECSA'19), Association for Computing Machinery,* New York, NY, USA, 2019. Vol. 2. P. 26–29. doi: 10.1145/3344948.3344997.
12. Pokorný P., Ševčík D. An application for solving truth functions. *In Intelligent Algorithms in Software Engineering (CSOC 2020).* Springer International Publishing. 2020. Vol. 1224. P. 341–351. doi: 10.1007/978-3-030-51965-0_30.
13. Yevseienko O., Kachanov P. Optimizing Energy Management in Shopping Centers with Smart Grid and Solar Power. *2024 IEEE 5th KhPI Week on Advanced Technology (KhPIWeek),* Kharkiv, Ukraine. 2024. P. 1-6. doi: 10.1109/KhPIWeek61434.2024.10878102.
14. Windows App Development. About Messages and Message Queues. Article, ed. 19.08.2022 (10 contributors). URL: https://learn.microsoft.com/en-us/windows/win32/winmsg/about-messages-and-message-queues. (дата звернення: 26.03.2025).
15. Unity Technologies. MonoBehaviour.OnGUI(). Unity Documentation, Ver. 2020.1. Publ.: 2021-02-24. URL: https://docs.unity3d.com. 2024. (дата звернення: 26.03.2025).

References (transliterated)

1. Dix A., Finlay J., Abowd G. D., Beale R. *Human-Computer Interaction.* Pearson: Prentice Hall, 2005. 834 p.
2. *User Interface Design. Case study: UX/UI design for complex technical simulation software with professional users. Interface Design*. Available at: https://interface-

design.co.uk/case-studies/ux-ui-design-technical-software-users (accessed 26.03.2025).
3. Paine D., Ghoshal D. Ramakrishnan L. Experiences with a Flexible User Research Process to Build Data Change Tools. *Journal of Open Research Software*, 2020, Vol. 8(1), pp. 18, doi: 10.5334/jors.284.
4. Liedtke S., Brendel F. Exploring the immediate mode GUI concept for graphical user interfaces in mixed reality applications. *GI VR/AR Workshop, Gesellschaft für Informatik e.V.* 2022, doi: 10.18420/vrar2022_1678.
5. Essick J. *The Python-Based Laboratory: A Hands-On Guide for Scientists and Engineers. Chapter 2. Graphical User Interface Using Tkinter.* Oxford. Oxford Academic, 2024, pp. 14–32, doi: 10.1093/9780191998478.003.0002.
6. Stokke K. A., Barash M., Järvi J. A domain-specific language for structure manipulation in constraint system-based GUIs. *Journal of Computer Languages*, 2023, Vol. 74(3), pp. 101175, doi: 10.1016/j.cola.2022.101175.
7. Mondal S. K., Pei Y., Dai H. N., Dipu Kabir H. M., Sahoo J. P. Boosting UI Rendering in Android Applications. *IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C),* Macau, China, 2020, pp. 285-286, doi: 10.1109/QRS-C51114.2020.00055.
8. Zuev A., Karaman D., Olshevskiy A. Wireless Sensor Synchronization Method for Monitoring Short-Term Events. *Advanced Information Systems*, 2023, Vol. 7(4), pp. 33–40, doi: 10.20998/2522-9052.2023.4.04.
9. Zuev A. O., Karaman D. G., Yevseyenko O. M. *Systemy modeliuvannia ta vizualizatsii imitatsiino-trenazhernykh kompleksiv: monohrafiia.* Kharkiv. Sekishova T. Ye., 2023. 223 p.
10. Knoblauch J., Sethuraman A., Hey J. IMGui — a desktop GUI application for isolation with migration analyses. *Molecular Biology and Evolution*, 2017, Vol. 34, no. 2, pp. 500–504, doi: 10.1093/molbev/msw252.
11. Olsson T., Ericsson M., Wingkvist A. An exploration and experiment tool suite for code to architecture mapping techniques. *In Proceedings of the 13th European Conference on Software Architecture (ECSA'19), Association for Computing Machinery,* New York, NY, USA, 2019, Vol. 2, pp. 26–29, doi: 10.1145/3344948.3344997.
12. Pokorný P., Ševčík D. An application for solving truth functions. *In Intelligent Algorithms in Software Engineering (CSOC 2020).* Springer International Publishing, 2020, Vol 1224, pp. 341–351, doi: 10.1007/978-3-030-51965-0_30.
13. Yevseienko O., Kachanov P. Optimizing Energy Management in Shopping Centers with Smart Grid and Solar Power. *2024 IEEE 5th KhPI Week on Advanced Technology (KhPIWeek),* Kharkiv, Ukraine, 2024, pp. 1-6, doi: 10.1109/KhPIWeek61434.2024.10878102.
14. Windows App Development. About Messages and Message Queues. Article, ed. 19.08.2022 (10 contributors). Available at: https://learn.microsoft.com/en-us/windows/win32/winmsg/about-messages-and-message-queues. (accessed 26.03.2025).
15. Unity Technologies. MonoBehaviour.OnGUI(). Unity Documentation, Ver. 2020.1. Publ.: 2021-02-24. Available at: https://docs.unity3d.com. 2024. (accessed 26.03.2025).

Відомості про авторів (About authors)

Зуєв Андрій Олександрович – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», завідувач кафедри «Автоматика та управління в технічних системах»; м. Харків, Україна; ORCID: 0000-0001-8206-4304; e-mail: andrii.zuiev@khpi.edu.ua.

Zuev Andrey – Ph. D., Docent (Associate Professor), Head of the Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine; ORCID: 0000-0001-8206-4304; e-mail: andrii.zuiev@khpi.edu.ua.

Євсеєнко Олег Миколайович – кандидат технічних наук, доцент, Національний технічний університет «Харківський політехнічний інститут», доцент кафедри «Автоматика та управління в технічних системах»; м. Харків, Україна; ORCID: 0000-0001-5432-1211; e-mail: oleh.yevseienko@khpi.edu.ua.

Yevseienko Oleh – Ph. D., Docent (Associate Professor), Associate Professor of the Department of Automation and Control in Technical Systems, National Technical University «Kharkiv Polytechnic Institute», Kharkiv, Ukraine; ORCID: 0000-0001-5432-1211; oleh.yevseienko@khpi.edu.ua.

Караман Дмитро Григорович – старший викладач кафедри «Автоматика та управління в технічних системах»; м. Харків, Україна; ORCID: 0000-0002-7252-3172; e-mail: dmytro.karaman@khpi.edu.ua.

Karaman Dmytro – Senior Lecturer of the Department of Automation and Control in Technical Systems, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine; ORCID: 0000-0002-7252-3172; e-mail: dmytro.karaman@khpi.edu.ua.

Будь ласка, посилайтесь на цю статтю наступним чином:

Зуєв А. О., Караман Д. Г., Євсеєнко О. М. Алгоритм та програмна реалізація інтерфейсу безпосереднього режиму для систем візуалізації. *Вісник Національного технічного університету «ХПІ». Серія: Нові рішення в сучасних технологіях*. – Харків: НТУ «ХПІ». 2025. № 2 (24). С. 58-65. doi:10.20998/2413-4295.2025.02.08.

Please cite this article as:

Zuev A., Karaman D., Yevseienko O. Algorithm and Software Implementation of The Immediate Mode Interface for Visualization Systems. *Bulletin of the National Technical University "KhPI". Series: New solutions in modern technology*. – Kharkiv: NTU "KhPI", 2025, no. 2(24), pp. 58-65, doi:10.20998/2413-4295.2025.02.08.

Надійшла (received) 24.04.2025
Прийнята (accepted) 01.06.2025