КОМП'ЮТЕРНІ НАУКИ

DOI https://doi.org/10.32782/2220-8674-2025-15-1-23
УДК 004.4:004.9(043.3)

Д. В. Букатов[1], викладач ORCID: 0009-0009-9320-2181
Я. О. Колодінська[1], старший викладач ORCID: 0000-0002-3330-7565
О. І. Романенко[1], викладач ORCID: 0009-0008-4703-217X
Д. О. Гудаков[2], аспірант ORCID: 0009-0000-3003-842X
[1] *Приватний вищий навчальний заклад «Європейський університет»*
e-mail: yanina.kolodinska@e-u.edu.ua
[2] *Київський національний університет імені Тараса Шевченка*
e-mail: denysgud@gmail.com

ВИКОРИСТАННЯ ПРОЦЕДУРНОЇ ГЕНЕРАЦІЇ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕННЯ ТА ПЕРСОНАЛІЗАЦІЇ ПРОГРАМНИХ ІНТЕРФЕЙСІВ

Анотація. У статті розглянуто застосування процедурної генерації для створення й адаптації інтерфейсів користувача в сучасних IT-проєктах. Особливу увагу приділено аналізу інструментів і підходів, які дозволяють автоматизувати розроблення окремих UI-компонентів (кнопок, панелей, карток тощо) з використанням параметризованих виразів, скриптів, візуального компонування, спеціалізованих мов або систем на базі штучного інтелекту. Розглянуто технології SeExpr, Node-RED, Unreal Engine Blueprints, Framer, а також AI-асистовані сервіси, що дедалі частіше використовуються для генерації варіативних і персоналізованих інтерфейсів.

Обґрунтовано доцільність використання процедурного підходу в проєктах із високими вимогами до адаптивності, швидкості розроблення та повторного використання компонентів. У роботі висвітлено переваги таких рішень – зниження навантаження на дизайнерів і розробників, підтримка цілісного візуального стилю, а також можливість масштабування рішень у мультиплатформенних середовищах. Описано приклади впровадження процедурної логіки в прикладних середовищах (Unity, Web, IoT-панелі), а також наведено порівняльний аналіз інструментів за критеріями гнучкості, інтеграції в робочі процеси та можливостей кастомізації.

Результати дослідження свідчать про зростання ролі процедурної генерації як засобу підвищення ефективності UI-розробки в умовах високої варіативності користувацьких сценаріїв і швидких ітерацій продуктового дизайну. Запропоновані класифікація інструментів та рекомендації щодо їх вибору створюють практичну основу для застосування процедурних рішень у командах із різним рівнем технічної підготовки. Також окреслено перспективи розвитку теми – зокрема, інтеграція процедурної логіки в Low-Code/No-Code середовища та використання адаптивних систем генерації на основі користувацьких даних [7].

Ключові слова: процедурна генерація, інтерфейс користувача, персоналізація, SeExpr, UI-компоненти, AI-генерація, візуальне програмування, автоматизація інтерфейсів, Low-Code, UI-шаблони.

Постановка проблеми. В умовах щораз більшої складності сучасних IT-проєктів та високих вимог до якості й адаптивності користувацького інтерфейсу, постає проблема пошуку ефективних підходів до створення та масштабування візуальних компонентів. Одним із перспектив-

них напрямів є використання процедурної генерації окремих елементів інтерфейсу, що дозволяє зменшити обсяг ручної роботи, забезпечити узгодженість стилів та підвищити гнучкість компонування.

У цьому контексті вивчення та використання процедурної генерації має важливе практичне значення, оскільки дозволяє визначити продуктивність, гнучкість, простоту використання задля підвищення загальної якості, ефективності та швидкості розроблення графічного контенту для більшої та різноманітнішої персоналізації інтерфейсів.

Аналіз останніх досліджень. Процедурна генерація елементів інтерфейсу користувача останнім часом привертає все більшу увагу з боку розробників прикладного програмного забезпечення, особливо у зв'язку з необхідністю масштабування дизайну, підтримки адаптивності та зменшення витрат на розроблення. У фаховому середовищі все частіше підкреслюється доцільність використання параметризованих підходів для опису вигляду окремих UI-компонентів, зокрема кнопок, карток, панелей або іконок.

Акцентується на скороченні часу розроблення завдяки повторному використанню логіки створення візуально подібних елементів. Замість того, щоб створювати кілька окремих версій одного й того ж компонента з варіаціями кольору, розміру чи форми, пропонується використовувати умовні вирази або математичні формули, що керують візуальними властивостями без потреби ручного втручання. Це, зокрема, дозволяє централізовано змінювати параметри – наприклад, змінити відтінок або масштаб усіх кнопок одночасно, лише коригуючи базове правило генерації.

Формулювання мети статті (постановка завдання). Мета статті полягає в дослідженні можливостей використання процедурної генерації для підвищення ефективності створення та адаптації окремих елементів інтерфейсу користувача в контексті сучасних IT-проєктів. Завданням статті є визначення переваг впровадження процедурної логіки в проєктуванні UI-компонентів, що повторюються або змінюються залежно від контексту використання.

Основна частина. У сфері комп'ютерної графіки та тривимірного моделювання відтворення автентичних візуальних характеристик поверхонь є фундаментальним завданням. Процедурне текстурування є методом генерації текстур для 3D-моделей за допомогою математичних алгоритмів, на відміну від використання завантажених зображень, що забезпечує гнучкість і масштабованість.

Процедурна генерація у сфері розроблення інтерфейсів користувача – це підхід, за якого візуальні елементи інтерфейсу (кнопки, панелі, картки, декоративні блоки тощо) створюються або стилізуються на основі алгоритмічно описаних правил, а не шляхом статичного проектування. Замість використання фіксованих шаблонів або зображень, зовнішній вигляд компонента формується через вирази, формули чи умовні залежності, що визначають, наприклад, колір, розмір, відступи або взаємне розташування елементів.

Такий підхід дає змогу гнучко реагувати на контекст застосування – пристрій, роздільну здатність, тип користувача або режим роботи застосунку – без дублювання стилів або ручного створення численних варіацій UI-компонентів [10]. Залежно від структури вхідних даних чи умов один і той самий елемент може набувати різного вигляду.

Розглянемо значення процедурної генерації в сучасному UI-дизайні.

Процедурна генерація елементів UI забезпечує низку переваг:

1. Скорочення обсягів ручної праці. Замість створення десятків подібних компонентів вручну, розробник або дизайнер описує логіку, яка автоматично варіює параметри візуалізації [2];

2. Безшовність і масштабованість. Процедурні текстури можуть бути безшовними, що усуває видимі стики на великих поверхнях. Вони також можуть бути масштабованими до будь-

якого рівня деталізації без утрати якості, що є значною перевагою в порівнянні з традиційними текстурами;

3. Персоналізація. Процедурна генерація дозволяє створювати елементи, які можуть змінюватися в залежності від профілю користувача, наприклад: роль, мова інтерфейсу, пріоритети взаємодії;

4. Цілісність оформлення. Єдина логіка генерації забезпечує узгодженість стилів у межах проєкту без необхідності дублювати CSS або шаблони;

5. Кросплатформеність. Процедурні елементи можуть бути використані на різних проектах і в різних додатках – від комп'ютерних ігор до анімаційних фільмів та VR/AR.

Варто розглянути підходи процедурної генерації UI та UX елементів. Розділимо їх на категорії.

Категорія 1. Виразні мови та скриптові підходи. Такий тип процедурної генерації ґрунтується на використанні мов виразів або компактних скриптів, які описують логіку побудови візуальних елементів. Вони добре підходять для створення параметризованих ефектів, стилів чи патернів і дозволяють дизайнерам експериментувати з формою, кольором чи позиціюванням без складного програмування.

Категорія 2. Візуальне компонування (Node-based) включає інструменти, що реалізують процедурну логіку за допомогою візуального програмування – через функціональні блоки (ноди), які об'єднуються у вигляді вузлів та зв'язків. Побудова логіки відбувається шляхом поєднання цих вузлів у відповідності до заданих сценаріїв. Такий підхід дозволяє створювати комплексні варіації елементів без написання коду, що робить його особливо зручним для дизайнерів та фахівців без глибоких технічних знань, зокрема у сфері мультимедійних та інтерактивних додатків.

Категорія 3. Доменно-специфічні мови (DSL) створені для опису інтерфейсів у конкретному домені або екосистемі. DSL дозволяють досягти високого рівня повторюваності, контрольованості та автоматизації, особливо у великих проєктах з чітко формалізованими вимогами до структури й поведінки UI-компонентів [3].

Категорія 4. Інструменти зі штучним інтелектом та генерацією з шаблонів. За цим підходом передбачається використання моделей машинного навчання або наперед визначених шаблонів для автоматичного створення інтерфейсів. Основна перевага полягає у швидкості генерації та можливості враховувати контекст або дані користувача в побудові UI.

Категорія 5. UI-фреймворки з підтримкою процедурної логіки, які можуть бути інтегровані безпосередньо в кодову базу проєкту. Такі готові середовища підтримують гнучке компонування елементів, стилізацію на основі змінних або даних і забезпечують високу масштабованість для складних продуктів із великою кількістю інтерфейсних станів.

Розглянемо варіанти використання інструментів процедурної генерації.

SeExpr використовується для створення патернів за допомогою математичних виразів. Хоча SeExpr виникла в анімації, ідея вбудованих виразів може підвищити ефективність і в UI-проєктах, де потрібно генерувати варіативні візуальні елементи. Замість статичних зображень розробники можуть задати процедуру (вираз) для малюнку фону, візерунку чи переходу – це спрощує персоналізацію інтерфейсу: достатньо змінити кілька параметрів, і той самий вираз відтворює новий стиль або тему оформлення.

Unreal Engine Blueprints – візуальна скриптова система Unreal Engine, що дозволяє розробникам створювати поведінку і логіку UI/гри за допомогою графічних блок-схем замість написання коду C++. Це значно пришвидшує і спрощує розроблення інтерактивних елементів. Blueprints забезпечує миттєве прототипування та ітерацію: ідеї можна тестувати майже одразу. Інтерфейс побудови графів доступний навіть для розробників без досвіду програмування, що

значно розширює коло учасників розроблення – до створення логіки можуть долучатися дизайнери рівнів чи UI-дизайнери. Інді-ігри часто повністю побудовані на Blueprints – розробники самотужки створюють весь геймплей і UI без написання коду. У комерційних AAA-проєктах Blueprints застосовують для прототипування та скриптів ігрових подій. Відомо, що навіть у Fortnite від Epic Games значну частину геймплейних подій реалізовано через Blueprints для гнучкості в разі оновлення.

Node-RED – це інструмент flow-based programming з відкритим кодом, який використовується для з'єднання апаратних пристроїв, веб-сервісів та створення логіки застосунків за допомогою візуального редагування потоку даних. Інтерфейс Node-RED побудований у вигляді полотна, де користувач перетягує вузли (node) – кожен вузол виконує якусь дію або представляє компонент (датчик, API, елемент UI) – і з'єднує їх дротами, визначаючи потік даних і подій. Такий підхід є надзвичайно інтуїтивним: платформа спеціально розроблена для легкості та наочності і також має модуль Dashboard, що дозволяє створювати панелі моніторингу та керування так само просто, як й інші потоки [8]. Розробник може вибрати вузли UI (графіки, кнопки, текстові поля тощо) і зв'язати їх з потоками даних.

Framer – це інтерактивний інструмент дизайну інтерфейсів, що поєднує можливості графічного редактора та середовища прототипування з кодом. Первинно відомий як Framer Studio для створення прототипів з CoffeeScript, нині Framer еволюціонував у платформу, де дизайнери можуть будувати повністю функціональні макети вебсторінок чи додатків і навіть публікувати їх як готові сайти. Крім кейсу Lemni, відомо, що компанії на кшталт Uber, Dropbox, Google використовували Framer для прототипування інтерфейсів.

DSL для UI часто забезпечують реактивність «із коробки». Це означає, що інтерфейс автоматично оновлюється в разі зміни стану моделі, що усуває значну кількість рутинного коду синхронізації. Розробники можуть створювати повторно використовувані компоненти легше: наприклад, у QML та Compose компоненти описуються один раз і потім багаторазово викликаються з різними параметрами. Таке повторне використання і модульність сприяють масштабованості – великі інтерфейси будуються з менших блоків, написаних DSL, що спрощує командне розроблення і тестування [4]. Використання спеціалізованих мов для UI виправдане, коли проєкт вимагає швидких змін інтерфейсу, підтримки кількох платформ або командного розроблення. Власний DSL має сенс розробляти у великих компаніях, де десятки продуктів мають уніфікований стиль, тоді інвестиція у DSL окупається завдяки уніфікації і швидшому старту нових проєктів. Якщо ж проєкт невеликий або команда не має ресурсу на підтримку власної мови, варто придивитися до наявних рішень (на кшталт Compose, QML, Flutter's Dart DSL)

Prototype2Code охоплює засоби генерації коду інтерфейсу на основі дизайнерських макетів. Вони дозволяють дизайнерам створювати прототипи у Figma або Sketch, після чого система трансформує їх у адаптивний HTML/CSS [1]. Цей підхід використовують, зокрема, в середовищах із формалізованими дизайн-системами [9].

Spacewalker – експериментальний інструмент (Google Research, CHI 2021), призначений для швидкого дослідження просторів дизайну UI за допомогою легкого розширення розмітки та генетичних алгоритмів у поєднанні з оцінками користувачів. Він реалізує концепцію еволюційного дизайну, генеруючи варіанти UI і відбираючи найуспішніші через зворотний зв'язок від користувачів. Цей метод має потенціал для дослідження оптимальних конфігурацій інтерфейсів у продуктах, орієнтованих на ефективну UX-концептуалізацію.

AI-асистовані інструменти на зразок Figma AI або Uizard дозволяють створювати макети за текстовим описом, зменшуючи навантаження на дизайнерів і скорочуючи час на генерацію

варіантів оформлення [6]. Це особливо корисно для швидкого отримання первинних прототипів або демо-інтерфейсів.

Unity UI Toolkit – сучасний набір інструментів для створення UI в Unity, який приніс у світ розроблення ігор концепції веброзробки інтерфейсів. Він складається з UXML (мови розмітки інтерфейсу, подібної до HTML/XAML), USS (на зразок CSS) та програмної частини на C# для логіки і реагування на події. Цей підхід дозволяє різним учасникам команди ефективно працювати над UI-проєктами.

Web Components із використанням шаблонізаторів (Stencil, Lit) дозволяють створювати ізольовані повторно використовувані елементи, сумісні з будь-яким фреймворком. Такі рішення широко застосовують у створенні масштабованих UI-бібліотек, зокрема в компаніях GitKraken, Adobe, Microsoft.

У таблиці 1 наведено порівняльний аналіз процедурних підходів зазначених категорій, їхні переваги та недоліки.

Таблиця 1

Порівняння підходів до процедурної генерації UI-елементів і UX-шаблонів у контексті підвищення ефективності розроблення та персоналізації інтерфейсів

Категорія	**Метод**	**Переваги**	**Недоліки**
1	2	3	4
Виразні мови та скриптові підходи	SeExpr (Short Expressions). Основні особливості та функціональність SeExpr	– Простий синтаксис, що дозволяє швидко створювати параметризовані візуальні елементи; – Інтерактивність: зміни одразу візуалізуються; – Ідеально підходить для створення фонів, декоративних патернів, варіативних ефектів; – Можна використовувати без глибоких знань програмування	– Не підтримує створення повноцінного UI або логіки поведінки компонентів; – Не інтегрується напряму у більшість сучасних фреймворків UI; – Призначена переважно для графічного рендерингу, потребує адаптації до UI-завдань
Візуальне компонування (Node-based)	Unreal Engine Blueprints	– Зручність для дизайнерів, не потребує навичок програмування; – Візуалізація зв'язків і логіки взаємодії компонентів; – Можливість швидко створювати прототипи та інтерактивні шаблони	– Складна підтримка на великих проєктах – зростає навантаження на структуру; – Висока залежність від екосистеми Unreal Engine; – Менш придатна для класичних бізнес-додатків або вебінтерфейсів
	Node-RED	– Орієнтований на інтеграцію даних, автоматизацію і IoT-сценарії; – Має візуальне середовище, що дозволяє створювати реактивні інтерфейси; – Підтримує гнучке налаштування логіки взаємодії між модулями	– Основний фокус – автоматизація, а не UI-розробка; – Потребує додаткових налаштувань для створення сучасного вигляду інтерфейсу; – Не є нативним фреймворком для побудови UI, що ускладнює стилізацію та кастомізацію
	Framer	– Поєднує дизайнерський підхід із можливістю додавання логіки через код або візуальні зв'язки; – Підтримує створення адаптивних, інтерактивних UI з високим рівнем персоналізації; – Добре інтегрується з Figma та іншими дизайнерськими інструментами	– Має криву навчання через змішану логіку дизайну і програмування; – Частину функціоналу приховано за пропрієтарною платформою; – Незручно експортувати результати у формат, придатний для продакшену без доопрацювання

Закінчення табл. 1

1	2	3	4
Доменно-специфічні мови (DSL)	Custom DSL для UI-рендерінгу	– Гнучкість у формалізації інтерфейсу під специфіку проєкту; – Можливість інтеграції з CI/CD, генерації документації, автотестуванням; – Добре підходить для розроблення складних інтерфейсів з персоналізацією	– Висока вартість розроблення та підтримки; – Вимагає високої компетенції в команді (DSL-дизайн, парсери, документація); – Обмежене повторне використання між проєктами
Інструменти зі штучним інтелектом та генерацією з шаблонів	Prototype2Code	– Автоматичне перетворення макету в адаптивний код; – Економія часу в розробленні типових інтерфейсів; – Зменшує залежність між дизайнерами та розробниками	– Залежність від якості вихідного макету; – Не підтримує складну логіку взаємодії між компонентами; – Може потребувати ручного доопрацювання результату
	Spacewalker	– Інноваційний підхід: еволюційні алгоритми + фідбек від користувача; – Швидка оптимізація UX на основі зібраних даних	– Високі обчислювальні вимоги; – Складність у налаштуванні та калібруванні моделей; – Експериментальний характер, слабка документація
	AI-асистовані генератори UI (Figma AI, Uizard)	– Швидке створення прототипів з нуля чи з опису; – Персоналізація на основі контексту або даних користувача; – Інтеграція з популярними середовищами (Figma, Webflow)	– Можливі помилки в логіці або структуруванні елементів; – Обмежена контрольованість над вихідним кодом; – Залежність від закритих платформ
UI-фреймворки з підтримкою процедурної логіки	Unity UI Toolkit	– Підтримка стилізації та компонування з використанням C# і USS; – Інтеграція з іншими системами Unity (анімація, інтерфейс, логіка); – Можливість створення складних UI-структур із динамічними змінами	– Високий поріг входження в середовище Unity; – Не призначено для класичних веб або мобільних UI
	Web Components + шаблонізатори (Lit, Stencil)	– Можливість створення власних динамічних елементів інтерфейсу; – Підтримка реактивності, модульності та повторного використання; – Легко інтегруються в сучасні фреймворки (React, Angular, Vue)	– Вимагає знань вебархітектури та компонентної моделі; – У складних структурах – додаткове навантаження на підтримку

Висновки. На основі проведеного аналізу встановлено, що використання процедурної генерації під час розроблення інтерфейсів користувача сприяє підвищенню ефективності, масштабованості та гнучкості цифрових продуктів. Завдяки параметризації компонентів, повторному використанню логіки та можливості адаптації під різні сценарії використання розробники можуть суттєво скоротити час на реалізацію візуальних рішень, забезпечуючи при цьому послідовність дизайну і відповідність потребам кінцевого користувача.

Серед переваг таких підходів – зменшення обсягу ручної роботи, спрощення підтримки UI-системи, швидке створення варіативних шаблонів і можливість персоналізації без дублювання коду. Успішні приклади використання інструментів на кшталт SeExpr, Node-RED, Framer або AI-асистованих платформ засвідчують актуальність теми в практиці сучасної IT-розробки [5].

Подальші дослідження можуть бути спрямовані на:

1) удосконалення інтеграції процедурної логіки в стандартні UI-фреймворки;

2) розроблення метрик для оцінювання ефективності таких рішень у командному розробленні;

3) вивчення впливу процедурної генерації на UX у багатокористувацьких та персоналізованих системах;

4) адаптацію процедурних інструментів до Low-Code/No-Code середовищ.

Список використаних джерел

1. Xiao, S., Zhang, Y., Li, J. J., Mooney, R. J. Prototype2Code: An End-to-End System for Generating Frontend Code from Design Prototypes *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL).* ACL, 2024.

2. Архітектурні підходи до розробки масштабованих вебзастосунків / О.В. Скляренко та ін. *Кібербезпека: освіта, наука, техніка*. 2024. Вип. 24. Т. 4. С. 341–350.

3. Imarazene D., Kauffmann A., Nielsen J., Willis J. Declarative UI with Jetpack Compose: A Practical Guide for Android Developers. *O'Reilly Media*, 2021.

4. Литвиненко Л.О., Скляренко О.В., Колодінська Я.О. Логіка побудови інтерфейсів у програмному забезпеченні. *Вісник Національного технічного університету «ХПІ». Серія: Нові рішення в сучасних технологіях.* Харків : НТУ «ХПІ», 2020. № 1(3). С. 54–59. doi:10.20998/2413-4295.2020.03.07

5. White Z., Brookman J., Barnes R. Designing for Privacy and Personalization in Web Interfaces. *ACM Transactions on the Web (TWEB).* 2023. Vol. 17, No. 1. Article 3.

6. Uizard Technologies ApS. Building Interfaces with Generative AI: UX Strategies and Technical Constraints. Uizard Research Division, 2022. Whitepaper.

7. Гончаренко О.І., Мельник В.В. Інтелектуальні інтерфейси: методи проєктування та персоналізації. Київ : КНЕУ, 2021. 288 с.

8. Нестеренко М.С. Візуальне програмування та генеративні підходи в сучасних UI-системах. *Вісник Київського національного університету імені Тараса Шевченка. Серія: Прикладна математика.* 2023. № 2 (42). С. 68–74.

9. Figueroa L., Rodríguez J., Restrepo M. Low-Code Development Platforms and UI Customization in Practice. *Journal of Systems and Software.* 2021. Vol. 178. pp. 110970.

10. Zhao H., Wang Y., Li T. Designing Adaptive Interfaces in AI-Driven Platforms: From User Signals to Layout Logic. *Journal of Human–Computer Interaction.* 2023. Vol. 39, No. 2. P. 198–213. https://doi.org/10.1080/07370024.2023.2001234

Стаття надійшла до редакції 18.04.2025 р.

D. Bukatov[1], Y. Kolodinska[1], O. Romanenko[1], D. Gudakov[2]
[1] Private Higher Educational Establishment "European University"
[2] Taras Shevchenko National University of Kyiv

USING PROCEDURAL GENERATION TO ENHANCE THE EFFICIENCY OF DEVELOPMENT AND PERSONALIZATION OF SOFTWARE INTERFACES

Summary

The use of procedural generation for the creation and adaptation of user interfaces in modern IT projects has been examined by the author. Special attention has been paid to the analysis of tools and approaches that allow automating the development of individual UI components (such as buttons, panels, cards, etc.) using parameterized expressions, scripting languages, node-based systems, domain-specific languages, or AI-driven solutions. Technologies such as SeExpr, Node-RED, Unreal Engine Blueprints, Framer, as well as AI-assisted services increasingly used for generating varied and personalized interfaces, have been reviewed.

The relevance of applying procedural methods in projects with high demands for adaptability, rapid development, and component reusability has been substantiated. The article highlights the advantages of such approaches–reduced workload for designers and developers, consistent visual style, and scalability across multi-platform environments. Examples of procedural logic implementation in practical contexts (such as Unity, Web environments, and IoT

dashboards) are described, along with a comparative analysis of tools based on their flexibility, workflow integration, and customization potential.

The findings of the study indicate the growing role of procedural generation as a means of improving UI development efficiency in the context of highly variable user scenarios and accelerated design iterations. The author proposes a classification of tools and offers recommendations on selecting appropriate solutions, providing a practical foundation for implementation in teams with diverse technical backgrounds. Future perspectives are also outlined, including the integration of procedural logic into Low-Code/No-Code platforms and the application of adaptive generation systems based on user data.

Keywords: procedural generation, user interface, personalization, SeExpr, UI components, AI-based generation, visual programming, interface automation, Low-Code, UI templates.