Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

**Бакалаврська кваліфікаційна робота**

на тему: «Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects»

Виконав: студент IV курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Подунай В. В.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ткаченко О. М.
(прізвище та ініціали)

Рецензент:. к.т.н., доц. каф ОТ Кадук О. В.
(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н.
«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
ФОП Михальнюк О. О.
«21» березня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«21» березня 2025 року

**З А В Д А Н Н Я**
**НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ**

Подунаю Владиславу Вячеславовичу

1. Тема роботи – Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects.
Керівник роботи: Ткаченко Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.
2. Строк подання студентом роботи 2 червня 2025.
3. Вихідні дані до роботи: методи автоматизації робочих процесів у Adobe After Effects, методи створення та управління шарами, текстовими шарами, 3D-об'єктами і кейфреймами, методи інтеграції графічного інтерфейсу через CEP-панелі, максимальна кількість шарів і композицій у проєкті не обмежена, дані про розташування шарів, параметри анімації та характеристики 3D-об'єктів у результуючій композиції, кінцевий вигляд автоматизованих робочих процесів та їхньої інтеграції у робоче середовище After Effects.
4. Зміст текстової частини: методи автоматизації робочих процесів у Adobe After Effects; аналіз існуючих інструментів для створення відеоконтенту; огляд методів роботи з шарами, текстом, 3D-об'єктами та кейфреймами; алгоритми автоматизації сортування шарів і створення прекомпозицій; алгоритми обробки текстових шарів і зміни якірної точки; методи створення та текстурування 3D-об'єктів; алгоритми управління кейфреймами, включаючи копіювання, вставлення та реверс; розробка модуля роботи з шарами; розробка модуля для створення 3D-об'єктів; розробка модуля управління текстом; розробка модуля для роботи з

кейфреймами; розробка інструкцій користувача.

5. Перелік ілюстративної частини: основні етапи алгоритму сортува шарів і створення прекомпозицій; алгоритми обробки текстових шарів і зм якірної точки; алгоритми створення та текстурування 3D-об'єктів; алгорит роботи з кейфреймами, включаючи реверс анімації; інтерфейс роботи з шарам композиціями; інтерфейс управління 3D-об'єктами та текстом; інтерфейс С панелі для автоматизації робочих процесів.

6. Консультанти розділів роботи

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1-4 | Ткаченко О. М., к.т.н., доцент кафедри ПЗ | 24.03.2025 | 01.06.2025 |

7. Дата видачі завдання 24 березня 2025 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1. | Аналіз стану технологій автоматизації робочих процесів у створенні відеоконтенту | 25.03.25 – 31.03.25 | |
| 2. | Розробка моделі та методів програмного модуля | 01.04.25 - 18.04.25 | |
| 3. | Розробка програмних модулів для автоматизації робочих процесів застосунку Adobe After Effects | 19.04.25 - 06.05.25 | |
| 4. | Тестування та практичне застосування програмного модуля | 07.05.25 - 24.05.25 | |
| 5. | Оформлення матеріалів до захисту БДР | 25.05.25 - 30.05.25 | |

**Студент** (підпис) **Подунай В. В.** (прізвище та ініціали)

**Керівник бакалаврської кваліфікаційної роботи** (підпис) **Ткаченко О. М.** (прізвище та ініціали)

УДК 004.912.032.26

Подунай В. В. Розро
процесів застосунку Adobe
спеціальності 121 – інжене
інженерія програмного забе

На укр. мові. Бібліогр

У бакалаврській квалі
автоматизації робочих про
створення модулів для сп
текстом та кейфрейма
продуктивності та зручнос

Програмні модулі, р
відеовиробництва, фріланс
After Effects для створення

Програмний модул
JavaScript та HTML у с
(Common Extensibility P
бібліотеки jQuery та CSIn

Ключові слова: А
процеси.

# АНОТАЦІЯ


УДК 004.912.032.26

Подунай В. В. Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects: бакалаврська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 111 с.

На укр. мові. Бібліогр.: 23 назв; рис.: 38; табл. 2.

У бакалаврській кваліфікаційній роботі розроблено програмний модуль для автоматизації робочих процесів застосунку Adobe After Effects. Проєкт включає створення модулів для спрощення роботи з шарами, тривимірними об'єктами, текстом та кейфреймами. Розширення призначене для підвищення продуктивності та зручності роботи відеомонтажерів, дизайнерів та аніматорів.

Програмні модулі, розроблені в проєкті, можуть бути використані студіями відеовиробництва, фрілансерами та іншими фахівцями, які працюють з Adobe After Effects для створення професійного контенту.

Програмний модуль розроблено з використанням мов програмування JavaScript та HTML у середовищі Visual Studio Code із застосуванням CEP (Common Extensibility Platform). Для інтеграції з After Effects використано бібліотеки jQuery та CSInterface.js, а також кастомні стилі CSS.



Ключові слова: Adobe After Effects, відеоконтент, автоматизація, робочі процеси.

# ANNOTATION

Podunai V. V. Development of a Software Module for Automating Workflows in Adobe After Effects. Vinnytsia: VNTU, 2025. 111 p.

In Ukrainian language. Bibliographer: 23 titles; fig.: 38; table. 2.

The bachelor’s thesis focuses on the development of a software module for automating workflows in Adobe After Effects. The project includes the creation of modules to simplify working with layers, three-dimensional objects, text, and keyframes. The extension is designed to enhance productivity and convenience for video editors, designers, and animators.

The software modules developed in this project can be utilized by video production studios, freelancers, and other professionals working with Adobe After Effects to create professional content.

The software module was developed using JavaScript and HTML programming languages in the Visual Studio Code environment with the application of the Common Extensibility Platform (CEP). Integration with After Effects was achieved using the jQuery and CSInterface.js libraries, along with custom CSS styles.

Keywords: Adobe After Effects, video content, automation, workflow.

ЗМІСТ

# ВСТУП

**Обґрунтування вибору теми дослідження**

Сучасні тенденції у сфері створення відеоконтенту вимагають постійного вдосконалення робочих процесів, особливо з урахуванням швидкої цифрової трансформації у сфері дизайну та анімації. Одним із ключових викликів у цій галузі є підвищення ефективності роботи з Adobe After Effects шляхом автоматизації рутинних завдань та впровадження зручних інструментів для створення складних візуальних ефектів [1].

Багато фахівців, які працюють з After Effects, стикаються з проблемами, пов’язаними з ручним виконанням повторюваних операцій, таких як: сортування та створення відповідних шарів, налаштування тексту та кейфреймів, що призводить до значних витрат часу та зниження продуктивності. У цьому контексті створення програмного модуля для автоматизації робочих процесів дозволяє значно спростити завдання, зменшити ризик помилок і прискорити створення професійного контенту [2].

Сучасні інструменти автоматизації для After Effects можуть не лише прискорити виконання базових операцій, але й забезпечити зручний інтерфейс для роботи з різними аспектами проєкту. Більшість доступних рішень на ринку є комерційними або не повністю відповідають потребам користувачів, тому актуальною є розробка власного застосунку [3].

**Зв’язок роботи з науковими програмами, планами, темами.** Дослідження було проведено згідно з планом наукових досліджень на кафедрі програмного забезпечення.

**Мета та задачі дослідження.** Метою бакалаврської кваліфікаційної роботи є підвищення ефективності роботи у застосунку Adobe After Effects завдяки зменшенню часу на виконання базових операцій за рахунок автоматизації та оптимізації процесів роботи з шарами, 3D-об’єктами, текстом та кейфреймами.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- провести аналіз існуючих робочих процесів у Adobe After Effects для

визначення операцій, що потребують автоматизації;

- налаштувати API та CEP-панелі для взаємодії з інтерфейсом Adobe After Effects;
- розробити модулі для автоматизації роботи з текстом, шарами, 3D-об’єктами та ключовими кадрами.
- інтегрувати всі розроблені модулі у єдину систему для ефективної роботи;
- провести експериментальну перевірку й оцінити ефективність практичного застосування розробленого застосунку.

**Об'єктом дослідження** є процес розробки анімаційних ефектів та відеоконтенту з використанням програмних засобів автоматизації в Adobe After Effects.

**Предметом дослідження** є методи та засоби автоматизації робочих процесів у створенні візуального контенту.

**Методи досліджень.** У процесі досліджень використовувались: методи аналітичної геометрії для розробки моделей створення тривимірних об’єктів; методи типографіки та текстового аналізу для розбиття текстових шарів; методи чисельного аналізу для точного управління кейфреймами; комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

**Наукова новизна отриманих результатів.**

1. Отримав подальшого розвитку метод декомпозиції тексту в Adobe After Effects, який, на відміну від існуючих, передбачає унікальний підхід до розбиття тексту на слова та символи з адаптивним збереженням анімаційних ключів і автоматичним створенням фонових солідів, що дозволило спростити створення текстових анімаційних ефектів і підвищити продуктивність роботи з текстовими шарами.

2. Отримав подальшого розвитку метод автоматичного текстурування 3D-кубів в Adobe After Effects, який, на відміну від існуючих, виконує сегментацію кубів із автоматичним нанесенням текстур та їх адаптивним розміщенням, що забезпечує високу ефективність створення ефектів розпаду та каркасного відображення,

спрощуючи процес роботи із тривимірними об’єктами.

**Практичне значення отриманих результатів.** Розроблено інтерактивний програмний модуль для автоматизації робочих процесів у Adobe After Effects, який спрощує створення та управління шарами, текстом, 3D-об’єктами та кейфреймами, а також підвищує ефективність роботи відеомонтажерів і аніматорів за рахунок скорочення часу на базові операції.

**Особистий внесок здобувача.**

Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. Наукові праці опубліковано одноосібно.

**Апробація матеріалів бакалаврської кваліфікаційної роботи.**

Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університет (Вінниця, 2025 р.) і XIII Міжнародній науково-практичній конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 (Київ, 2025 р.).

**Публікації.** Результати роботи опубліковано в матеріалах LIV Всеукраїнської науково-технічної конференції підрозділів Вінницького національного технічного університету і XIII Міжнародної науково-практичної конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 [4, 5].

# 1 АНАЛІЗ РОЗВИТКУ ТЕХНОЛОГІЙ У СФЕРІ СТВОРЕННЯ ВІДЕОКОНТЕНТУ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

## 1.1 Аналіз стану технологій автоматизації робочих процесів у створенні відеоконтенту

У сучасному середовищі створення відеоконтенту та анімації автоматизація робочих процесів є одним із ключових напрямів, що безпосередньо впливає на продуктивність фахівців. Adobe After Effects є одним із популярних виборів серед авторів для вирішення таких задач, як створення складних анімаційних ефектів, монтаж відеоряду, налаштування рухомих камер і обробка звукових доріжок. Однак варто зазначити, що даний застосунок не є єдиним інструментом у цій сфері, адже багато професіоналів також використовують альтернативні програми, такі як Adobe Premiere Pro, DaVinci Resolve, Blender чи Cinema 4D, залежно від специфіки їхніх проєктів.

Одним із перспективних напрямків є розробка розширень для оптимізації робочих процесів у Adobe After Effects [6]. Сучасні інформаційні технології дозволяють автоматизувати численні операції, такі як створення анімаційних ефектів, монтаж відеоряду, налаштування рухомих камер, обробка звукових доріжок, а також спрощення підготовки основних елементів проєкту до фінального рендерингу. Завдяки цьому, користувачі можуть зменшити витрати часу та зусиль, а також знизити ймовірність помилок, що виникають при ручному виконанні складних завдань у процесі створення відеоконтенту.

Особливу увагу слід приділити інтеграції таких розширень із робочим середовищем програм для відеовиробництва, що дозволяє створювати єдину платформу для роботи з різними аспектами проєкту. Такі рішення можуть включати інструменти для швидкого налаштування анімаційних ефектів, управління шарами та автоматизації повторюваних операцій [7].

Використання розширень для автоматизації робочих процесів у створенні відеоконтенту не лише оптимізує внутрішні операції, але й забезпечує більш точний контроль над проєктом, що дозволяє фахівцям прискорювати створення

контенту та підвищувати його якість.

Крім того, впровадження розширень відкриває нові можливості для творчого процесу та оптимізації робочого часу. Автоматизація основних операцій дозволяє не лише оперативно виконувати поточні завдання, а й планувати складні візуальні ефекти, визначати ключові етапи виробництва та створювати ефективні робочі процеси. Це дозволяє користувачам не лише підвищувати продуктивність наявних ресурсів, а й адаптуватися до нових проєктів, сприяючи інноваційному розвитку у сфері відеовиробництва в умовах високої конкуренції [8].

Отже, розробка програмного модуля для автоматизації робочих процесів у сфері створення відеоконтенту є важливою складовою сучасного відеовиробництва, яка дозволяє значно підвищити продуктивність, забезпечити зручність роботи та створити умови для реалізації складних творчих ідей.

## 1.2 Порівняльний аналіз аналогів

Використання технологій автоматизації робочих процесів у створенні відеоконтенту стає дедалі популярнішим, і Adobe After Effects є одним із ключових інструментів для цього. Розробка розширень для спрощення взаємодії з робочими процесами дозволяє прискорити виконання завдань, оптимізувати структуру проєктів і підвищити якість кінцевого продукту. До найбільш відомих рішень належать:

- Motion Bro;
- Flow;
- FX Console;
- Ease and Wizz.

Motion Bro є одним із найпопулярніших розширень для Adobe After Effects і Premiere Pro, розробленим командою Videolancer, яке спеціалізується на спрощенні додавання анімаційних переходів, елементів дизайну та звукових ефектів у відеопроєкти [9]. Цей інструмент призначений для прискорення робочих процесів творців контенту, надаючи користувачам доступ до широкого набору готових пресетів, що дозволяють створювати професійні анімації та візуальні ефекти з

мінімальними зусиллями. Воно інтегрується з обома програмами Adobe Creative Cloud через панель розширень, забезпечуючи зручний графічний інтерфейс для роботи. Приклад роботи розширення Motion Bro показано на рисунку 1.1.

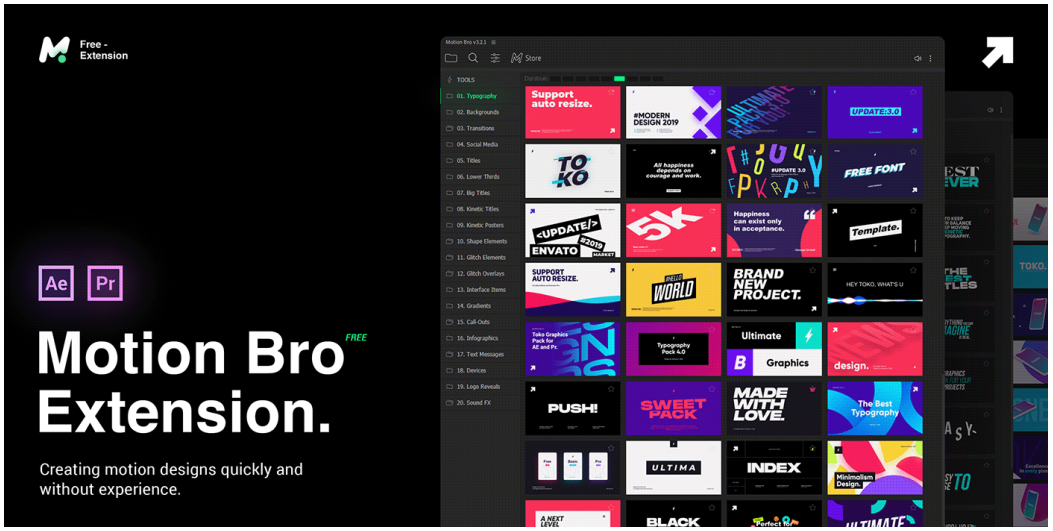



Рисунок 1.1 – Приклад роботи розширення Motion Bro

Інший приклад – розширення Flow, це розширення для Adobe After Effects, розроблене командою Zack Lovatt і renderTom, яке значно спрощує процес налаштування анімаційних кривих через інтуїтивний графічний редактор, замінюючи складний вбудований Graph Editor [10]. Воно призначене для прискорення робочих процесів аніматорів, надаючи інструменти для точного контролю над швидкістю, прискоренням і динамікою анімацій. Приклад роботи розширення Flow показано на рисунку 1.2.


Рисунок 1.2 – Приклад роботи розширення Flow

FX Console – це розширення, яке значно прискорює робочий процес завдяки швидкому доступу до ефектів і панелей [11]. Дозволяє здійснювати миттєвий пошук ефектів, керувати пресетами та знімати скріншоти робочої області. Його головна перевага – зручність використання та економія часу, але воно не містить додаткових анімаційних функцій та можливості роботи з кейфремами текстом та іншими елементами. На рисунку 1.3 показано приклад роботи розширення FX Console.

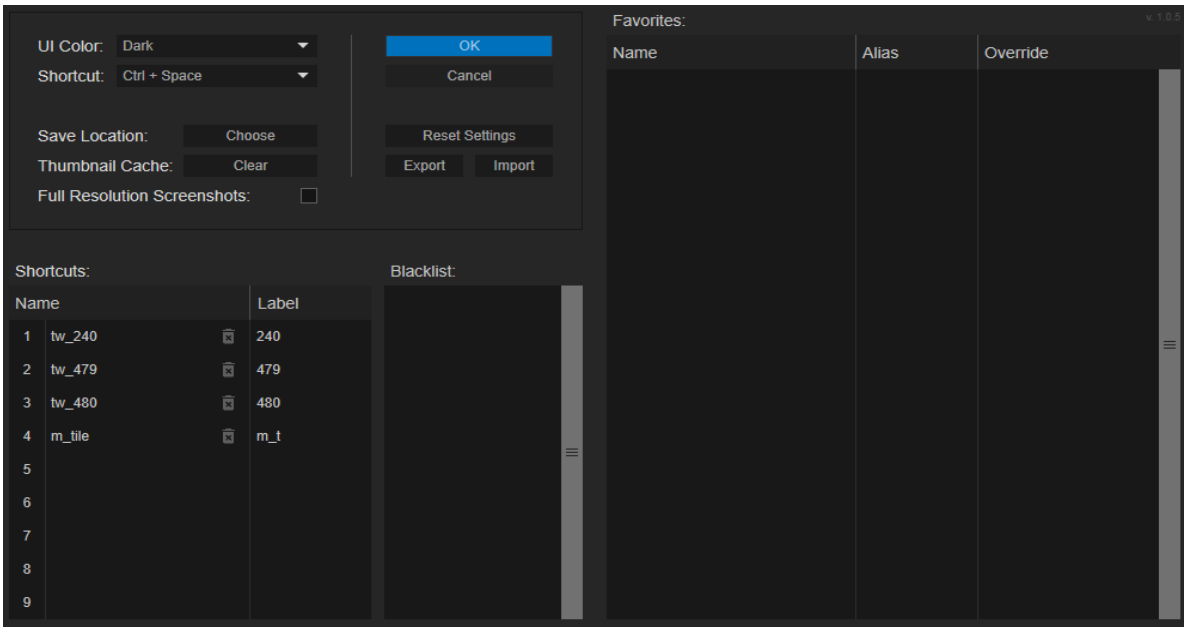



Рисунок 1.3 – Приклад роботи розширення FX Console

Ease and Wizz – це розширення для автоматизації роботи з кейфреймами, яке додає різні варіанти плавного прискорення та сповільнення руху об'єктів в анімації. Воно дозволяє швидко застосовувати складні інтерполяційні криві без ручного налаштування графіка швидкості, що значно спрощує анімаційний процес [12]. Розширення пропонує кілька типів кривих інтерполяції, таких як: Expo, Back, Bounce, Elastic та інші. Користувачу достатньо виділити ключові кадри, обрати потрібний тип кривої, і розширення автоматично застосує відповідний вираз для створення плавної анімації. Однак його функціонал обмежений тільки роботою з ключовими кадрами та не включає інші інструменти для керування шарами чи ефектами. Приклад роботи розширення Ease and Wizz показано на рисунку 1.4.

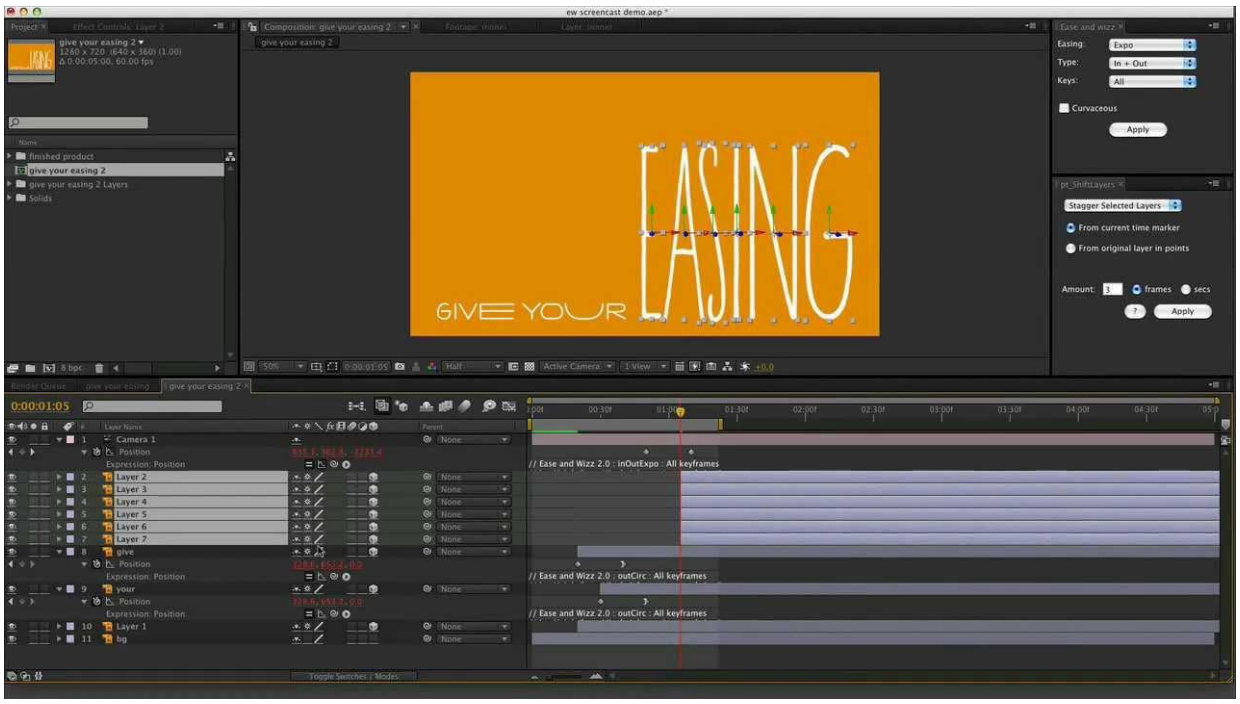



Рисунок 1.4 – Приклад роботи розширення Ease and Wizz

Розробка розширень для автоматизації робочих процесів у Adobe After Effects вимагає досвіду роботи з анімаційними інструментами, розробки програмного забезпечення та дизайну інтуїтивного інтерфейсу користувача. Це передбачає створення внутрішньої системи, яка буде здатна автоматизувати обробку шарів, кейфреймів і тривимірних об’єктів, а також оптимізувати процеси рендерингу та видалення невикористовуваних елементів. Складова інтерфейсу користувача передбачає розробку графічних панелей, які дозволять користувачу зручно взаємодіяти з розробленими алгоритмами та функціями розширення.

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

| Критерії | Motion Bro | Flow | FX Console | Ease and Wizz | Власна розробка |
|---|---|---|---|---|---|
| Робота з 3D-об’єктами | - | - | - | - | + |
| Оптимізація композицій | - | - | - | + | + |

Продовження таблиці 1.1

| Критерії | Motion Bro | Flow | FX Console | Ease and Wizz | Власна розробка |
|---|---|---|---|---|---|
| Інтуїтивність інтерфейсу | - | + | + | + | + |
| Взаємодія з текстовими шарами | + | - | - | - | + |
| Взаємодія з ключовими кадрами | + | + | - | + | + |
| Унікальні функції | + | - | - | - | + |
| Загальна оцінка | 50% | 33% | 17% | 50% | 100% |

Відповідно до таблиці порівняльних характеристик, розробка власного розширення є доцільною. Воно перевершує аналоги завдяки автоматизації роботи з шарами, текстом, 3D-об’єктами, оптимізації композицій, зручного інтерфейсу та унікальних функцій.

Отже, розробка розширення для автоматизації робочих процесів у Adobe After Effects відкриває нові можливості для відеовиробництва, дозволяючи прискорити виконання завдань і підвищити якість кінцевого продукту. Завдяки інноваційним функціям, розширення стає конкурентоспроможним рішенням для творців контенту.

## 1.3 Аналіз методів розв’язання задачі

Розробка розширень для автоматизації робочих процесів у Adobe After Effects вимагає комплексного підходу до вирішення завдань, пов’язаних із спрощенням

роботи з проєктами та підвищенням ефективності створення відеоконтенту. Існують різні методи, кожен з яких має свої переваги та недоліки. У цьому розділі проаналізовано найбільш ефективні методи розробки таких розширень.

Один з основних аспектів розробки розширень для After Effects – це автоматизація обробки основних елементів проєкту, таких як шари, кейфрейми та анімаційні ефекти. Для цього можуть бути використані алгоритми для спрощення налаштування анімацій і управління робочим простором. Проте цей підхід має певні обмеження, оскільки він може вимагати значних обчислювальних ресурсів і може бути менш ефективним, якщо алгоритми не оптимізовані для специфічних задач. Тому для такого методу потрібно розробляти додаткові механізми адаптації до різних типів проєктів.

Інший підхід полягає у використанні скриптів і плагінів для аналізу робочих процесів у After Effects. Ці інструменти дозволяють виявляти повторювані операції та автоматизувати їх, наприклад, сортування шарів або підготовку до рендерингу. Такі методи можуть включати автоматизоване управління композиціями чи очищення невикористовуваних елементів, але їхня ефективність залежить від якості налаштувань і може бути обмеженою у складних проєктах через відсутність гнучкості.

Найефективнішим підходом є створення інтегрованих розширень, які об’єднують різні модулі для роботи з шарами, 3D-об’єктами, текстом та кейфреймами в єдину систему. Такі розширення забезпечують зручний доступ до інструментів, автоматизують рутинні операції та надають інтуїтивний інтерфейс для користувачів [13].

Розглянувши переваги та недоліки кожного методу, було прийнято рішення комбінувати скрипти для автоматизації з інтегрованими розширеннями. Це дозволить отримати більш ефективні результати в спрощенні робочих процесів у After Effects, зокрема завдяки автоматизації рутинних завдань і адаптації до специфічних потреб користувачів. Такий підхід гарантує високу продуктивність і гнучкість у створенні відеоконтенту.

## 1.4 Постановка задач бакалаврської кваліфікаційної роботи

Проаналізувавши переваги та недоліки існуючих розширень для автоматизації робочих процесів застосунку Adobe After Effects, було виявлено потребу у створенні нового інструменту з модульною структурою та інтуїтивним інтерфейсом. Такий підхід дозволить оптимізувати управління шарами, текстом, 3D-об’єктами та кейфреймами, відповідаючи потребам як початківців, так і професіоналів. На основі цього аналізу було визначено наступні завдання, які необхідно виконати для успішної розробки розширення:

- провести аналіз існуючих робочих процесів у Adobe After Effects для визначення операцій, що потребують автоматизації;
- налаштувати API та CEP-панелі для взаємодії з інтерфейсом Adobe After Effects;
- розробити модулі для автоматизації роботи з текстом, шарами, 3D-об’єктами та ключовими кадрами.
- інтегрувати всі розроблені модулі у єдину систему для ефективної роботи;
- провести експериментальну перевірку й оцінити ефективність практичного застосування розробленого застосунку.

## 1.5 Висновки

У першому розділі було проведено аналіз сучасних програмних рішень для автоматизації робочих процесів у середовищі Adobe After Effects. Розглянуто існуючі розширення та інструменти, їхні функціональні можливості, переваги й недоліки. Було виявлено, що наявні засоби не повністю задовольняють потреби користувачів у зручному та гнучкому управлінні шарами, текстом, 3D-об’єктами й ключовими кадрами, особливо при створенні складних відеокомпозицій.

У результаті було сформульовано основні завдання для реалізації розширення: аналіз існуючих робочих процесів, налаштування API та CEP-панелей, розробка окремих модулів для роботи з різними елементами композиції, інтеграція цих модулів у єдину систему та проведення експериментального дослідження ефективності запропонованого рішення.

# 2 РОЗРОБКА МОДЕЛЕЙ ТА МЕТОДІВ ПРОГРАМНОГО МОДУЛЯ

## 2.1 Аналіз вхідних даних програмного модуля

Для реалізації розширення для автоматизації робочих процесів у Adobe After Effects буде використано мову програмування JavaScript у поєднанні з технологією ExtendScript і CEP-панелями (Creative Cloud Extension Panels), а також вбудовані API After Effects для роботи з шарами, композиціями, кейфреймами, 3D-об’єктами, текстом та звуком. JavaScript є основною мовою для створення скриптів у After Effects, що дозволяє розробляти гнучкі та потужні інструменти для автоматизації. ExtendScript забезпечує доступ до внутрішньої структури After Effects, дозволяючи взаємодіяти з об’єктами проєкту, такими як шари, ефекти, композиції та аудіодоріжки. CEP-панелі, побудовані на HTML, CSS і JavaScript, дозволяють створювати сучасні графічні інтерфейси користувача, які інтегруються в робочий простір After Effects [14].

Розробка розширення для автоматизації робочих процесів у Adobe After Effects передбачає кілька ключових етапів. На першому етапі буде створено модуль для обробки шарів, який відповідатиме за сортування шарів, створення коригувальних шарів, нульових об’єктів, прекомпозицій, а також додавання композицій до черги рендерингу. Цей модуль використовуватиме API After Effects (app.project.activeItem.layers) для аналізу та впорядкування шарів за типом (наприклад, текст, 3D, відео), а також об’єкт app.project.renderQueue для управління чергою рендерингу [15]. Для забезпечення точності сортування шарів модуль також враховуватиме їхні параметри, такі як розмір, позиція та видимість, що дозволить уникнути помилок при обробці складних композицій.

Далі буде розроблено модуль для роботи з 3D-об’єктами, який відповідатиме за створення тривимірних кубів із можливістю текстурування та розбиття на частини. Цей модуль використовуватиме методи Layer.transform і Layer.threeDLayer для створення та налаштування 3D-об’єктів, дозволяючи швидко додавати тривимірні елементи до проєкту. Для текстурування кубів передбачено використання матеріалів із підтримкою різних форматів зображень, що забезпечить

гнучкість у дизайні. Також модуль дозволить автоматично генерувати анімацію розбиття кубів, що спростить створення динамічних візуальних ефектів у проєкті.

Наступним етапом стане створення модуля для роботи з текстом, який дозволить декомпозицію текстових шарів на окремі символи або слова, а також застосування попередньо визначених текстових анімацій. Цей модуль використовуватиме API After Effects (TextDocument) для роботи з текстовими шарами та їхньою анімацією, спрощуючи створення складних текстових ефектів. Користувачі зможуть налаштовувати параметри анімації, такі як швидкість і напрямок руху символів, що додасть більше творчої свободи. Крім того, модуль передбачає можливість збереження користувацьких шаблонів анімацій, що значно прискорить повторне використання ефектів у різних проєктах.

Ще одним важливим модулем є управління кейфреймами, який надасть користувачу можливість вирівнювати ключові кадри, копіювати та вставляти графіки анімації, а також перевертати порядок кейфреймів для створення зворотного ефекту. Для реалізації цього модуля буде використано методи setTemporalEaseAtKey і keyTime, що забезпечують доступ до параметрів кейфреймів та їхньої часової поведінки. Модуль також включатиме функцію попереднього перегляду змін у реальному часі, що дозволить користувачам швидко оцінити результат редагування.

Розробка цих програмних модулів потребує глибокого розуміння API After Effects, роботи з JavaScript і ExtendScript, а також створення графічних інтерфейсів через CEP-панелі. Для реалізації будуть використані інструменти та бібліотеки, зокрема Adobe ExtendScript Toolkit і Visual Studio Code як основні середовища розробки, а також SDK After Effects для доступу до API. Програмні модулі будуть ретельно протестовані для забезпечення їхньої стабільності, швидкості роботи та точності обробки даних [16-19].

Отже, розробка розширення для автоматизації робочих процесів у Adobe After Effects на основі JavaScript, ExtendScript і CEP-панелей є комплексним завданням, яке включає кілька етапів. Модулі мають бути ретельно спроектовані, реалізовані та протестовані, щоб забезпечити користувачам зручний і ефективний

інструмент для управління робочими процесами. Завдяки використанню сучасних технологій програмний продукт зможе задовольнити потреби користувачів, допомагаючи оптимізувати робочий процес у After Effects.

## 2.2 Розробка моделі програмного модуля

Для кращого розуміння роботи модулів розширення для автоматизації робочих процесів у Adobe After Effects було вирішено розробити модель роботи системи на прикладі UML діаграми діяльності [20]. Згідно з вказаною діаграмою користувач для початку взаємодії з розширенням повинен відкрити проєкт After Effects із активною композицією, у разі відсутності активної композиції програма видаватиме повідомлення про помилку. Наступним кроком буде вибір одного з доступних розділів для роботи, таких як обробка шарів, створення 3D-об’єктів, робота з текстом та кейфреймами. Якщо в проєкті відсутні необхідні елементи для виконання операції, розширення повідомить про це у робочому середовищі і виведе перелік функцій, які неможливо виконати. У разі наявності всіх необхідних елементів розширення надасть доступ до відповідного функціоналу.

Після вибору розділу розширення автоматично проводить аналіз активної композиції, визначаючи доступні шари, кейфрейми або інші елементи, залежно від обраного завдання. Далі користувач може виконати автоматизовані дії, такі як сортування шарів, створення 3D-кубів, декомпозиція тексту, вирівнювання кейфреймів або видалення невикористовуваних елементів, унаслідок чого розширення автоматично обробить дані та застосує зміни до проєкту. Після завершення операцій користувач може переглянути результати у робочому середовищі After Effects, взаємодіяти з оновленою композицією (змінювати порядок шарів, переглядати анімації) та додати композицію до черги рендерингу для експорту результатів у вибраний формат. Такий підхід забезпечує чітку структуру взаємодії з розширенням, мінімізуючи помилки користувача. На рисунку 2.1 показано UML-діаграму діяльності, яка ілюструє послідовність кроків для полегшення розуміння функціоналу.

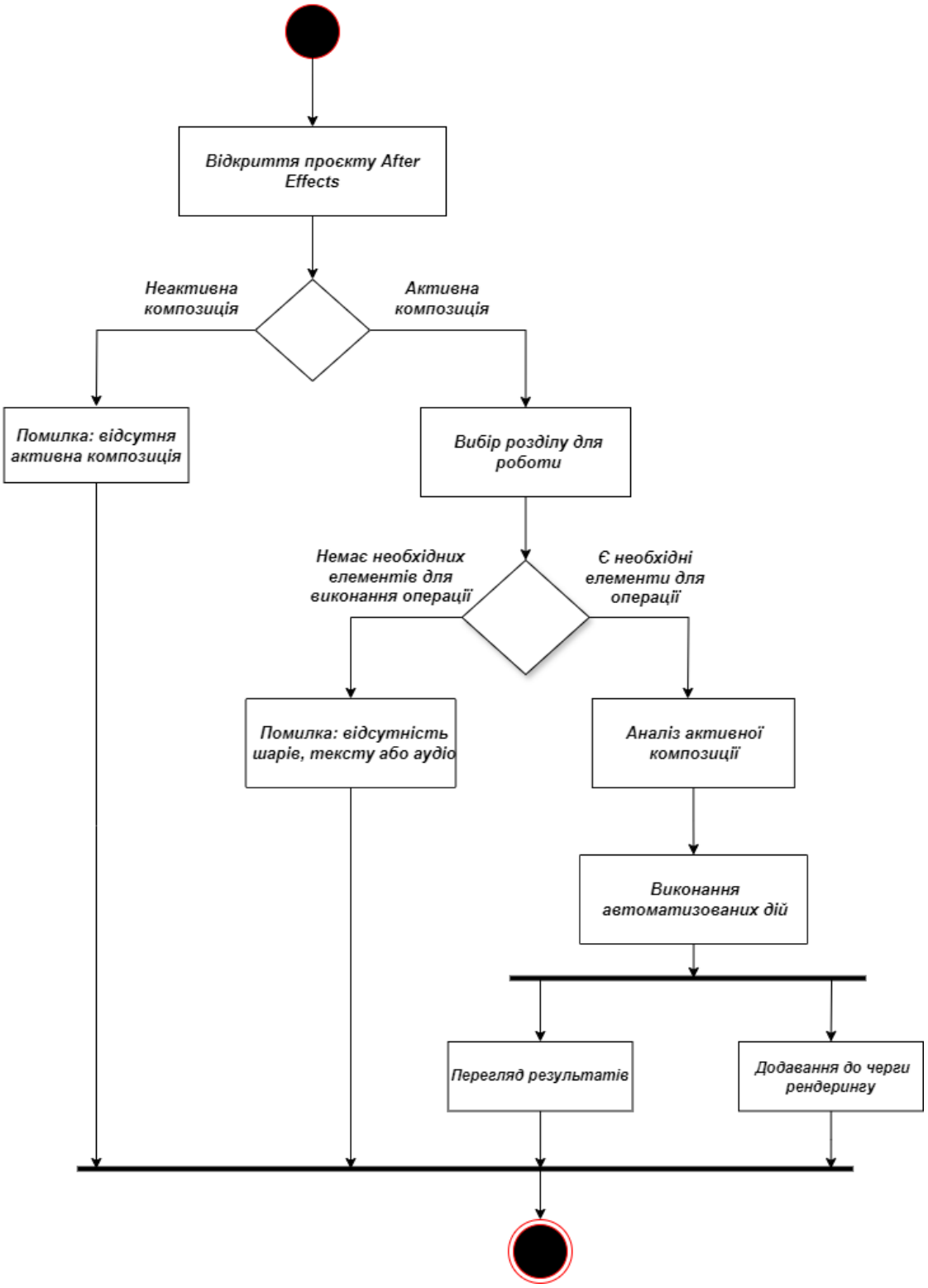



Рисунок 2.1 – Діаграма діяльності

Таким чином, розробка моделі системи передбачала комплексний підхід до аналізу функціональних вимог, проектування структури модулів та створення інтуїтивно зрозумілого робочого процесу для автоматизації завдань у Adobe After Effects.

## 2.3 Розробка методу декомпозиції тексту на слова та символи

Після реалізації базових методів декомпозиції тексту в Adobe After Effects, необхідно було розробити удосконалений підхід для роботи з текстовими шарами, який би розширив можливості стандартних операцій розбиття тексту на символи та слова, а також додав інтеграцію з автоматичним створенням фонових солідів. Основним методом стала модифікація традиційного розбиття тексту з урахуванням його анімаційних властивостей та оптимізації робочого процесу.

Стандартний метод декомпозиції тексту зазвичай обмежується простим розбиттям на окремі елементи без збереження контексту анімації чи додаткових шарів. Удосконалений метод, запропонований у цій роботі, включає розширені операції, такі як динамічне налаштування якірних точок і додавання солідів під текст, що дозволяє створювати складніші анімаційні ефекти. Як можливі підходи використовуються розбиття на символи та слова, доповнене функцією створення фону.

Основні кроки методу декомпозиції тексту:

- перевіряється наявність активної композиції типу CompItem і вибору рівно одного текстового шару, який не є параграфним;
- виконується розбиття тексту на символи або слова з дублікуванням вихідного шару для кожного елемента;
- додається аніматор із виразом, який приховує всі елементи, крім поточного, використовуючи селектор із режимом «Characters Excluding Spaces» для символів або «Words» для слів;
- вихідний шар вимикається, а процес завершується з можливістю скасування.

Метод може бути реалізований як ітеративно (з циклом по кожному символу чи слову), так і з використанням рекурсивних перевірок для складних текстів. Ітеративний підхід дозволяє ефективно обробляти великі обсяги тексту, зберігаючи стабільність у Adobe After Effects.

Стандартний метод декомпозиції має обмеження, зокрема відсутність інтеграції з анімаційними атрибутами та фоновим оформленням, що ускладнює створення ефектів.

Удосконалений метод вирішує ці проблеми:

- відсутність збереження анімаційних елементів, стандартний розбиток не враховує якірні точки чи анімаційні властивості тексту, що ускладнює процес декомпозиції;
- відсутність автоматичного створення фонового соліду, через що користувач змушений вручну додавати солід для кожного символу чи слову.

Для покращення цих недоліків запропоновано вдосконалений метод, який використовує масив тимчасових шарів для вимірювання розмірів тексту та HashMap для зберігання анімаційних атрибутів (наприклад, прозорість і позиція). Це дозволяє уникнути дублювання коду та підвищити ефективність обробки.

Основні кроки вдосконаленого методу:

- перевіряється активна композиція та текстовий шар, як у базовому методі;
- виконується розбиття на символи чи слова з автоматичним збереженням анімаційних властивостей у HashMap;
- створюється солід під кожним елементом із точним розрахунком позиції та розміру, використовуючи тимчасові шари;
- аніматор застосовується з виразом, що враховує збережені атрибути з HashMap;

Запропонований метод забезпечує гнучкість у створенні текстових анімацій, дозволяючи користувачам швидко адаптувати ефекти до різних потреб, що є значним кроком вперед порівняно зі стандартним підходом.

## 2.4 Розробка методу автоматичного створення та текстурування 3D-кубів

У процесі створення 3D-об’єктів у Adobe After Effects виникла потреба в удосконаленні методів роботи з кубами, зокрема для автоматизації текстурування та сегментації. Запропонований метод модифікує традиційний підхід до створення 3D-кубів, додаючи автоматичне текстурування з розбиттям на сегменти та

адаптивне позиціонування текстур, що дозволяє створювати ефекти розпаду та каркасного відображення з більшою ефективністю. Основним методом стала модифікація стандартного процесу створення кубів.

Стандартний метод роботи з 3D-об’єктами в After Effects передбачає ручне створення композицій для граней куба, ручне текстурування та налаштування позицій, що ускладнює створення складних ефектів, таких як розпад чи каркасне відображення. Удосконалений метод, запропонований у цій роботі, автоматизує ці процеси, дозволяючи користувачу швидко створювати сегментовані куби з текстурами, адаптованими до різних частин об’єкта. Як основні операції використовуються автоматичне створення граней, сегментація та адаптивне текстурування.

Основні кроки методу роботи з 3D-об’єктами:

- перевіряється наявність активної композиції типу CompItem і відповідних параметрів (розмір куба, кількість сегментів);
- формуються шість композицій (для базового куба) або більше (для сегментованого куба) із заданими розмірами (наприклад, 1080x1080 для граней або 1080xpartHeight для сегментів);
- якщо обрано текстуру (selectedImage), вона додається до кожної грані з адаптивним позиціонуванням залежно від сегмента; інакше додається суцільний шар із кольором, що залежить від номера сегмента;
- грані позиціонуються в основній композиції з урахуванням 3D-поворотів для створення кубічної структури;

Метод може бути реалізований як ітеративно (з циклом по гранях і сегментах), так і з урахуванням рекурсивних перевірок для складних структур. Ітеративний підхід забезпечує стабільність під час роботи з великою кількістю сегментів у After Effects.

Стандартний метод має кілька обмежень, які ускладнюють роботу з 3D-об’єктами:

- користувачу доводиться вручну додавати текстури до кожної грані, що займає багато часу;

- стандартний підхід не передбачає автоматичного розбиття куба на сегменти, що обмежує створення ефектів та взаємодію з об’єктом та його частинами.

Для подолання цих недоліків запропоновано вдосконалений метод, який використовує масив для зберігання граней і їх параметрів, а також алгоритм адаптивного позиціонування текстур залежно від кількості сегментів . Це дозволяє автоматизувати текстурування та сегментацію, зменшуючи кількість ручних операцій.

Основні кроки вдосконаленого методу:

- перевіряється активна композиція та параметри (розмір, сегменти), як у базовому методі;
- куб розбивається на сегменти з урахуванням кількості частин (parts), створюючи відповідну кількість граней;
- текстура позиціонується з урахуванням розміру сегмента і типу грані (верхня, нижня чи бокова), використовуючи тимчасові обчислення;
- грані об’єднуються в 3D-структуру з автоматичним налаштуванням.

Запропонований метод забезпечує ефективність створення 3D-об’єктів, дозволяючи користувачам швидко створювати ефекти розпаду та каркасного відображення, що є значним удосконаленням порівняно зі стандартним підходом. Завдяки автоматичному текстуруванню та сегментації кубів користувачі можуть досягати професійних результатів із меншою кількістю ручних операцій. Це значно спрощує робочий процес і підвищує продуктивність при створенні візуальних ефектів в Adobe After Effects.

## 2.5 Розробка алгоритмів роботи програмного модуля

Для розробки розширення для автоматизації робочих процесів у Adobe After Effects було створено загальний алгоритм, що визначає порядок дій програмного модуля, блок-схему якого показано на рисунку 2.2 [21].

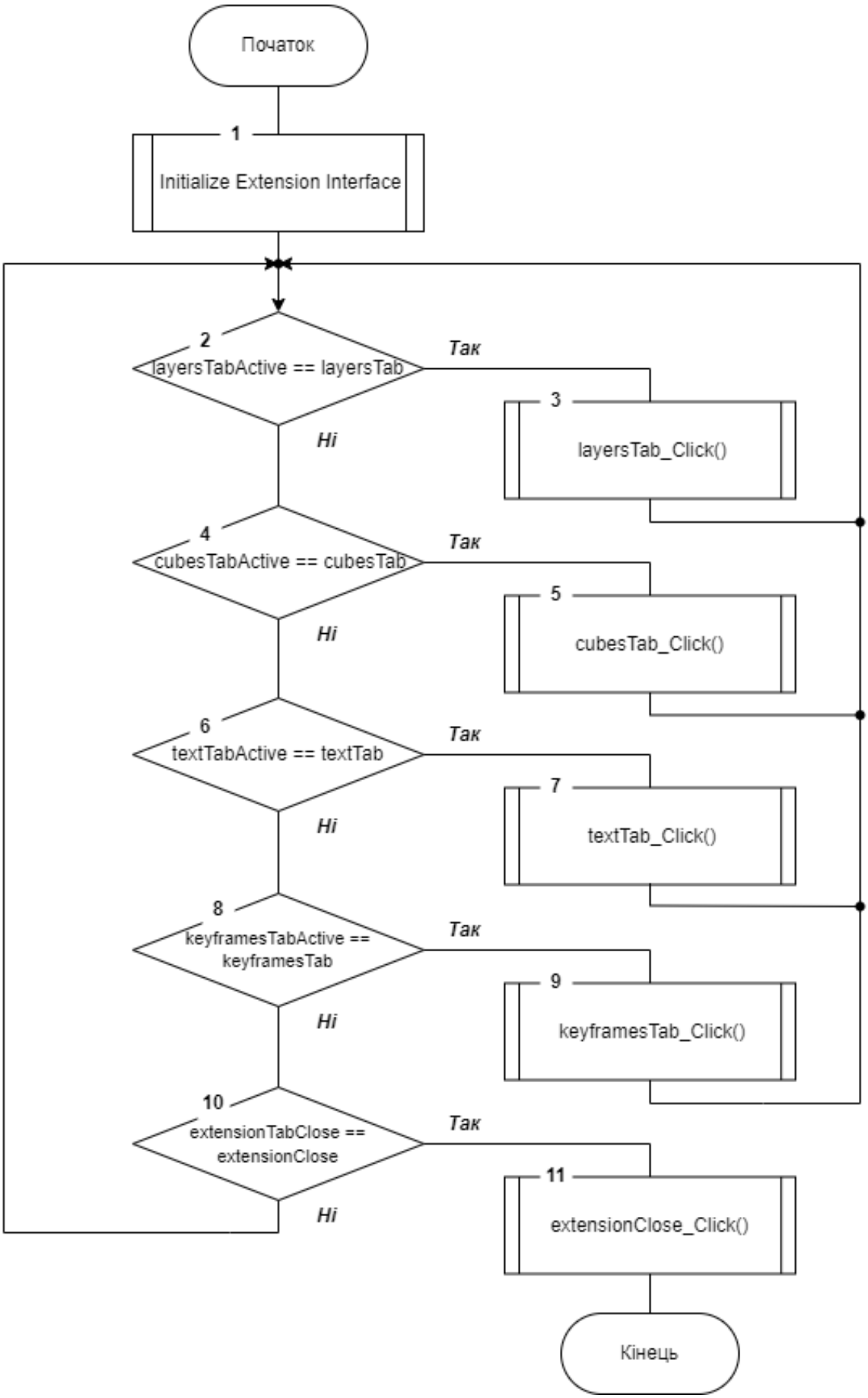



Рисунок 2.2 – Блок-схема загального алгоритму роботи розширення

## 2.6 Висновки

У другому розділі розглянуто розробку програмних модулів для автоматизації робочих процесів в Adobe After Effects, включаючи обробку шарів, створення 3D-об’єктів, декомпозицію тексту та управління кейфреймами. Описано вдосконалені методи текстурування 3D-кубів і декомпозиції тексту з інтеграцією анімаційних атрибутів та фонових солідів. Створено UML-діаграму діяльності та блок-схему загального алгоритму роботи розширення, що забезпечує чітку структуру взаємодії.

# 3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ДЛЯ АВТОМАТИЗАЦІЇ РОБОЧИХ ПРОЦЕСІВ ЗАСТОСУНКУ ADOBE AFTER EFFECTS

## 3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного модуля

При розробці розширення для автоматизації робочих процесів у Adobe After Effects важливо обрати оптимальні технології для створення функціональних модулів і зручного інтерфейсу. JavaScript є основною мовою для розробки розширень After Effects, оскільки він підтримується Adobe через ExtendScript API, що дозволяє взаємодіяти з ядром програми, автоматизувати операції з шарами, кейфреймами, аудіо та іншими елементами композиції. Використання JavaScript забезпечує ефективну обробку даних і виконання складних сценаріїв автоматизації.

Для створення графічного інтерфейсу було обрано технологію CEP (Creative Cloud Extension Panels), яка базується на HTML, CSS і JavaScript. CEP-панелі дозволяють розробляти сучасні інтерфейси, інтегровані в робочий простір After Effects, із підтримкою вкладок, кнопок, полів введення та інших елементів. HTML і CSS забезпечують гнучке оформлення інтерфейсу, дозволяючи створювати темну тему з чіткими іконками, що відповідає дизайну After Effects, а JavaScript обробляє події користувача, такі як натискання кнопок для запуску функцій.

Node.js використовується для інтеграції додаткових бібліотек і модулів у розширення. Завдяки Node.js можна підключати зовнішні інструменти, наприклад, для обробки файлів або створення складних обчислень, таких як аналіз аудіодоріжок для позначення бітів. Це робить розширення більш універсальним і дозволяє розширити його функціонал у майбутньому.

Visual Studio Code було обрано як основне середовище розробки завдяки його підтримці JavaScript, HTML і CSS, а також інтеграції з ExtendScript Toolkit для тестування коду в After Effects. Visual Studio Code надає зручні інструменти для відлагодження, автодоповнення коду та роботи з Git, що прискорює процес розробки. Adobe ExtendScript Toolkit використовується для прямого виконання

скриптів у After Effects, що дозволяє швидко перевіряти функціонал розширення, наприклад, створення 3D-кубів або декомпозицію тексту.

Для обробки 3D-об’єктів використовується вбудований функціонал After Effects, зокрема підтримка 3D-шарів і камер. Це дозволяє створювати прості 3D-об’єкти, такі як куби, без залучення додаткових бібліотек, що спрощує розробку і зменшує залежність від зовнішніх інструментів.

Поєднання цих технологій забезпечує ефективну розробку розширення для автоматизації робочих процесів у Adobe After Effects. JavaScript і ExtendScript API дозволяють створювати потужну кодову базу для автоматизації операцій із шарами, текстом, кейфреймами та аудіо, тоді як CEP-панелі на основі HTML, CSS і JavaScript забезпечують зручний і сучасний інтерфейс. Node.js додає гнучкість для інтеграції додаткових модулів, а Visual Studio Code і ExtendScript Toolkit оптимізують процес розробки та тестування.

Отже, вибір технологій для розробки розширення, таких як JavaScript, CEP-панелі (HTML, CSS, JavaScript), Visual Studio Code і ExtendScript Toolkit, є обґрунтованим для створення ефективного інструменту автоматизації в After Effects. Поєднання цих засобів забезпечує комплексний підхід до розробки, дозволяючи створити продукт із високим рівнем функціональності, зручності та сумісності з Adobe After Effects.

## 3.2 Розробка інтерфейсу програмного модуля

При розробці інтерфейсу розширення особливу увагу було приділено його зрозумілості та зручності для користувача. Основним тематичним кольором інтерфейсу обрано темний сірий із різними відтінками, що відповідає стандартному дизайну Adobe After Effects.

Інтерфейс розширення виконаний у вигляді єдиної панелі, яка інтегрується в робочий простір After Effects через CEP-панелі. При відкритті розширення користувач бачить головну панель із вкладками у верхній частині, які дозволяють перемикатися між різними функціональними розділами: «Layers», «3D objects», «Text», «Keyframes». Головна панель із розділом «Layers» показана на рисунку 3.1.

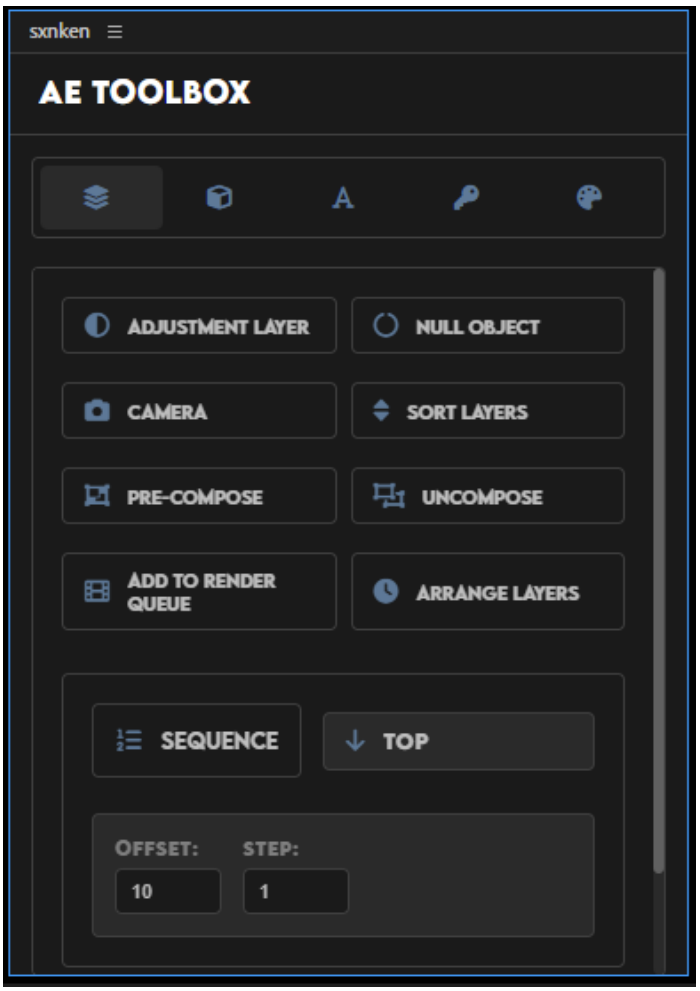



Рисунок 3.1 – Головна панель із розділом «Layers»

Розділ «Layers» призначений для автоматизації роботи зі шарами в After Effects: додає коригувальні шари та нульові об’єкти, сортує шари за типом, створює та розпаковує прекомпозиції, додає композиції до черги рендерингу, вирівнює шари відносно полоси відтворення чи початку композиції, а також виконує зміщення шарів з заданим кроком та проміжком зміщення. Кнопки розташовані вертикально, із чіткими іконками, що полегшують навігацію.

На рисунку 3.2 показано панель із розділом «3D objects», призначений для створення 3D-кубів у проєкті, який дозволяє вибрати текстуру, створює прості або каркасні 3D-куби, розбиває їх на частини для ефектів розпаду та додає до композиції. Розділ також підтримує автоматичне позиціонування текстур залежно від сегментів куба, що спрощує створення складних анімацій. Користувач може налаштувати параметри куба, такі як розмір і кількість сегментів, через інтуїтивний інтерфейс панелі.

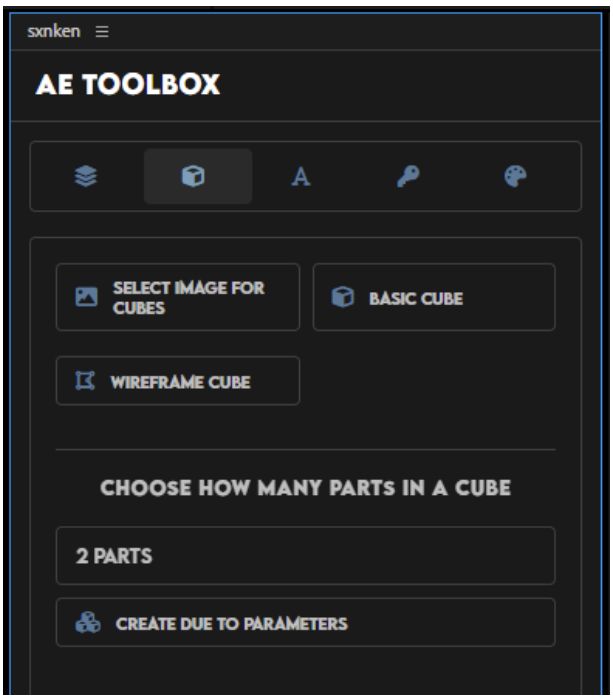



Рисунок 3.2 – Розділ «3D objects»

На рисунку 3.3 показано панель із розділом «Text», що призначений для автоматизації роботи з текстовими шарами, виконує декомпозицію тексту на символи або слова з можливістю створення фонового соліду, створення субтитрів за вписаним текстом, а також дозволяє змінити положення якірної точки для обраного текстового слою.

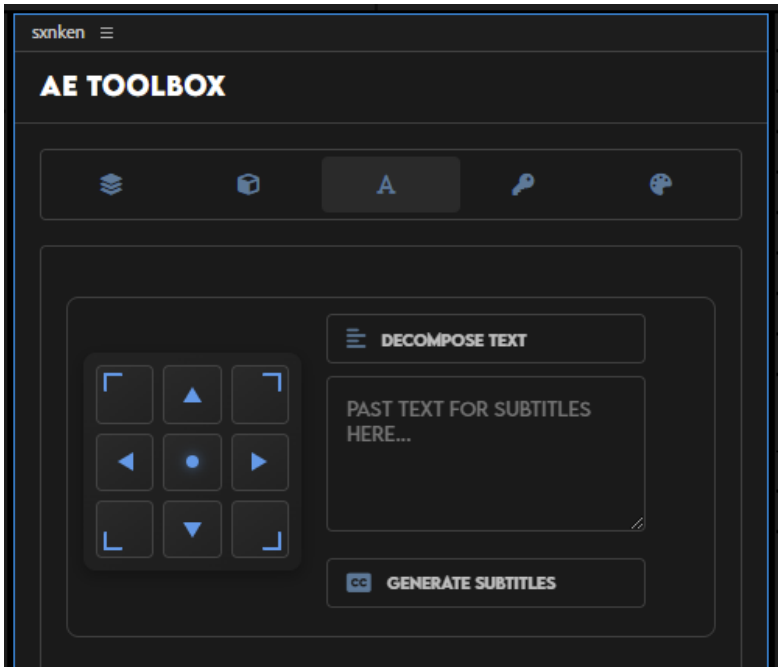



Рисунок 3.3 – Розділ «Text»

На рисунку 3.4 показано вікно розбиття обраного текстового шару на слова та символи, а також створення суцільного фонового шару.

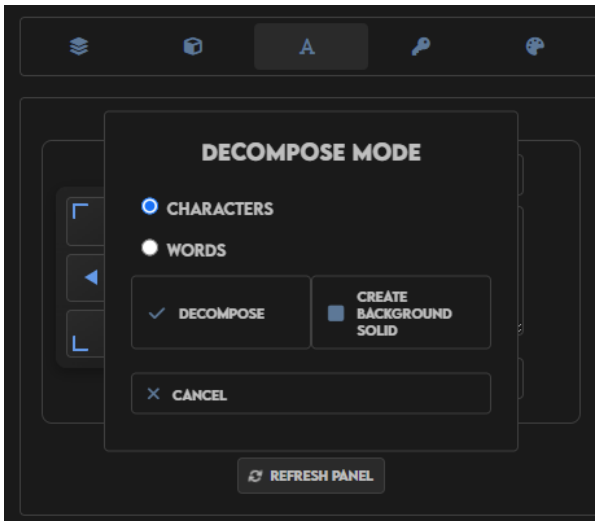



Рисунок 3.4 – Вікно розбиття текстового шару

На рисунку 3.5 показано панель із розділом «Keyframes» призначений для автоматизації роботи з ключовими кадрами, вирівнюванням кейфреймів, копіювання та застосування скопійованих графіків анімації, реверс кейфреймів, можливість клонування обраних ключових кадрів, додання виразу для ефекту пружності для кейфреймів позиції та масштабу.

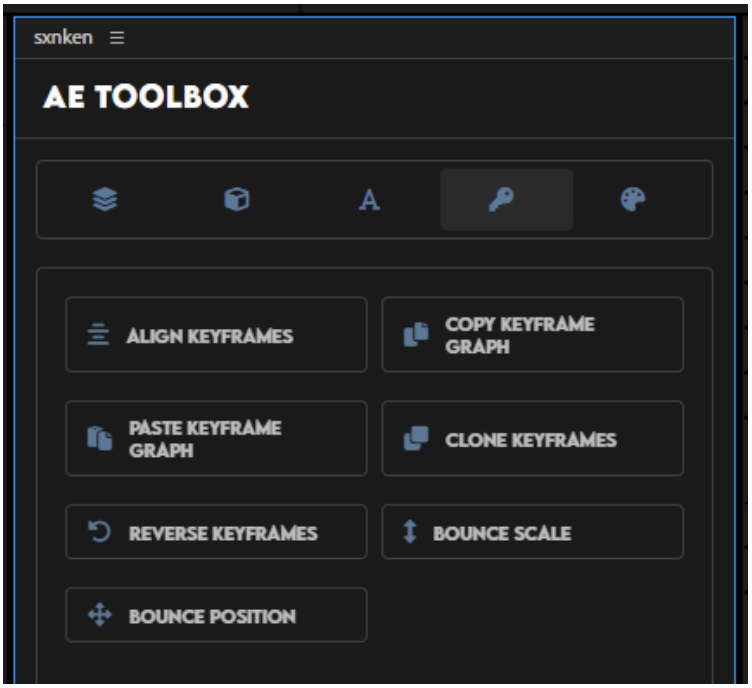



Рисунок 3.5 – Розділ «Keyframes»

У нижній частині панелі розташована кнопка «Refresh Panel», яка дозволяє оновити інтерфейс після внесення змін у проєкті. Дизайн виконаний у темній темі з чіткими іконками та текстом для зручного сприйняття. Кнопки в кожному розділі мають інтуїтивно зрозумілі назви та іконки, що полегшують навігацію.

Розробка графічного інтерфейсу розширення для автоматизації робочих процесів у Adobe After Effects була проведена з використанням технології CEP-панелей, що базуються на HTML, CSS і JavaScript. HTML і CSS застосовувалися для створення структури та стилів інтерфейсу, а JavaScript використовувався для обробки подій і взаємодії з API After Effects. Для розробки застосовувалися середовища Visual Studio Code та Adobe ExtendScript Toolkit, що забезпечують зручне налаштування та тестування. Інтерфейс протестований на сумісність із різними версіями After Effects (починаючи з CC 2022), щоб гарантувати стабільність і зручність використання.

### 3.3 Розробка модуля для взаємодії з шарами

Модуль обробки шарів автоматизує операції з шарами в Adobe After Effects, спрощуючи рутинні завдання. Він включає наступні функції: додавання коригувального шару, створення нульового об’єкта, сортування шарів за типом, створення прекомпозиції з вибраних шарів, додання виділених елементів до прекомпозиції, розпакування прекомпозиції, додавання композиції до черги рендерингу, вирівнювання шарів відносно полоси відтворення чи початку композиції, а також зміщення шарів з заданим кроком та проміжком зміщення.

На рисунку 3.6 показано реалізацію функції sortLayersByType яка Функція сортування шарів за типом упорядковує вибрані шари в активній композиції типу CompItem, якщо їх щонайменше два. Вона визначає тип кожного і групує їх. Шари сортуються за обраним порядком: варіант «1» – Adjustment, Solids, Text тощо; варіант «2» – Text, Solids, Adjustment тощо; варіант «3» – Solids, Text, Adjustment тощо. При некоректному варіанті видається помилка. Шари переміщуються на початок стеку, зберігаючи групову цілісність для однакових типів. Після виконання користувач отримує повідомлення про успіх або помилку.

```
sortLayersByType: function(sortOption) {
    app.beginUndoGroup("Sort Selected Layers by Type");

    try {
        var comp = app.project.activeItem;
        if (!(comp instanceof CompItem)) {
            alert("Виберіть композицію!");
            app.endUndoGroup();
            return false;
        }

        var selectedLayers = comp.selectedLayers;
        if (selectedLayers.length < 2) {
            alert("Виберіть мінімум два шари для сортування!");
            app.endUndoGroup();
            return false;
        }

        var typeGroups = {};
        for (var i = 0; i < selectedLayers.length; i++) {
            var layer = selectedLayers[i];
            var layerType = $._layers.getLayerType(layer);
            var typeName = (function(type) {
                switch (type) {
                    case 1: return "Text";
                    case 2: return "Adjustment";
                    case 3: return "Solids";
                    case 4: return "Shapes";
                    case 5: return "Comps";
                    case 6: return "Video";
                    case 7: return "Audio";
                    default: return "Other";
                }
            })(layerType);
            if (!typeGroups[typeName]) typeGroups[typeName] = [];
            typeGroups[typeName].push({ layer: layer, originalIndex: i });
        }

        var sortedTypes = [];
        if (sortOption === "1") {
            sortedTypes = ["Adjustment", "Solids", "Text", "Shapes", "Comps", "Video", "Audio", "Other"];
        } else if (sortOption === "2") {
            sortedTypes = ["Text", "Solids", "Adjustment", "Shapes", "Comps", "Video", "Audio", "Other"];
        } else if (sortOption === "3") {
            sortedTypes = ["Solids", "Text", "Adjustment", "Shapes", "Comps", "Video", "Audio", "Other"];
        } else {
            alert("Невірний варіант сортування!");
            app.endUndoGroup();
            return false;
        }

        var sortableLayers = [];
        for (var i = 0; i < sortedTypes.length; i++) {
            if (typeGroups[sortedTypes[i]]) {
                sortableLayers = sortableLayers.concat(typeGroups[sortedTypes[i]]);
            }
        }

        if (sortableLayers.length === 0) {
            alert("Вибрані шари не підтримуються для сортування!");
            app.endUndoGroup();
            return false;
        }
```

Рисунок 3.6 – Код функції sortLayersByType

На рисунку 3.7 показано реалізацію функції addNullObject автоматизує створення нульового об’єкта в активній композиції типу CompItem із трьома режимами роботи залежно від модифікаторів і вибраних шарів. При Shift + Click додає «Null Controller» на всю тривалість композиції. При Alt + Click із кількома шарами створює один нульовий об’єкт із назвою «Null for []», що охоплює їхній часовий діапазон, і прив’язує шари до нього. За звичайного кліку кожен вибраний шар отримує свій нульовий об’єкт «Null for []», синхронізований за часом і розміщений перед ним як батьківський.

```
addNullObject: function(isAltPressed, isShiftPressed) {
    app.beginUndoGroup("Add Null Object");

    var comp = app.project.activeItem;
    if (!(comp instanceof CompItem)) {
        alert("Select a composition in the project window or timeline!");
        app.endUndoGroup();
        return false;
    }

    var selectedLayers = comp.selectedLayers;

    if (isShiftPressed) {
        // Shift + Click:
        var nullLayer = comp.layers.addNull();
        nullLayer.name = "Null Controller";
        nullLayer.startTime = 0;
        nullLayer.outPoint = comp.duration;
        app.endUndoGroup();
        return true;
    }

    if (selectedLayers.length === 0) {
        alert("Select at least one layer!");
        app.endUndoGroup();
        return false;
    }

    if (isAltPressed && selectedLayers.length > 1) {
        // Alt + Click:
        var maxDuration = 0;
        var earliestStart = comp.duration;
        var layerNames = [];
        var sortedLayers = selectedLayers.slice().sort(function(a, b) {
            return b.index - a.index;
        });

        for (var i = 0; i < sortedLayers.length; i++) {
            var layer = sortedLayers[i];
            var duration = layer.outPoint - layer.inPoint;
            maxDuration = Math.max(maxDuration, duration);
            earliestStart = Math.min(earliestStart, layer.inPoint);
            layerNames.push(layer.name);
        }

        var nullLayer = comp.layers.addNull();
        nullLayer.name = "Null for " + layerNames.join(", ");
        nullLayer.startTime = earliestStart;
        nullLayer.outPoint = earliestStart + maxDuration;
        nullLayer.moveToBeginning();

        for (var i = 0; i < selectedLayers.length; i++) {
            selectedLayers[i].parent = nullLayer;
        }
```

Рисунок 3.7 – Код функції addNullObject

Функція preCompose створює прекомпозицію з вибраних шарів, спочатку перевіряючи наявність активної композиції типу CompItem і хоча б одного вибраного шару. Вона сортує шари за індексами, створює масив їхніх індексів і генерує унікальну назву для нової композиції на основі дати й часу. Тривалість прекомпозиції обчислюється як різниця між найранішим inPoint і найпізнішим outPoint вибраних шарів, а часові параметри шарів у новій композиції коригуються для коректного розміщення на часовій шкалі.

Розпаковка прекомпозицій (uncomposed) повертає вміст у вихідну композицію. Перевіряється наявність активної композиції та вибраних прекомпозицій. Кожен шар із прекомпозиції – чи то текст, шейп, коригувальний шар, камера, світло, нульовий об’єкт чи медіа – копіюється з усіма властивостями, часом і батьківськими зв’язками. Нові шари розміщуються над прекомпозицією, яка залишається прихованою, а порожні прекомпозиції пропускаються з повідомленням. Це дозволяє легко повернути шари до основної композиції без втрати налаштувань.

Для впорядкування шарів у часі передбачено функцію послідовного зміщення. Вона діє на вибрані шари в активній композиції, якщо їх два або більше. Шари сортуються за порядком у стеку, а за потреби інвертуються. Кожен шар зміщується на заданий інтервал у кадрах, який накопичується після певної кількості шарів, визначеної кроком. Це забезпечує плавне розподілення шарів у таймлайні, зберігаючи їхню тривалість.

Додавання до черги рендерингу автоматизує підготовку композиції до експорту. Функція перевіряє наявність проекту та активної композиції, додаючи її до черги з таймкодом робочої області або повної тривалості. Залежно від модифікатора, обирається формат виводу (h.264 або mov) і шлях збереження з унікальним індексом для уникнення конфліктів імен. Якщо шаблон рендерингу недоступний, застосовується стандартний. Після виконання фокус повертається до активного вікна, забезпечуючи безперервність роботи.

### 3.4 Розробка модуля для створення 3D-об’єктів

Модуль створення 3D-об’єктів генерує 3D-куби в Adobe After Effects, надаючи користувачу інструменти для створення об’єктів із різними характеристиками. Основні функції модуля включають вибір текстури для куба, створення простого куба, створення каркасного куба і створення розбитого куба з кількістю частин, заданою користувачем, які дозволяють користувачу генерувати 3D-об’єкти з текстурою, у вигляді каркаса або з розділеними сегментами.

На рисунку 3.8 показано реалізацію функції createBasicCube.

```
createBasicCube: function() {
app.beginUndoGroup("Create Basic 3D Cube");

var comp = app.project.activeItem;
if (!comp || !(comp instanceof CompItem)) {
    alert("Выберите композицию в окне проекта или таймлайне!");
    app.endUndoGroup();
    return;
}

var cubeSize = 1080;
var mainComp = comp;

var cubeFaces = [];
var cubeLayers = [];

for (var i = 0; i < 6; i++) {
    var faceComp = app.project.items.addComp("Basic Cube Face " + (i+1), cubeSize, cubeSize, 1.0, mainComp.duration, mainComp.frameRate);
    cubeFaces.push(faceComp);

    if (selectedImage) {
        var imageLayer = faceComp.layers.add(selectedImage);
        imageLayer.position.setValue([cubeSize / 2, cubeSize / 2]); /
    } else {
        faceComp.layers.addSolid([0.5, 0.5, 0.5], "Solid", cubeSize, cubeSize, 1);
    }
}

for (var i = 0; i < 6; i++) {
    var layer = mainComp.layers.add(cubeFaces[i]);
    layer.threeDLayer = true;
    cubeLayers.push(layer);
}

cubeLayers[0].position.setValue([mainComp.width/2, mainComp.height/2, -cubeSize/2]);
cubeLayers[1].position.setValue([mainComp.width/2, mainComp.height/2, cubeSize/2]);
cubeLayers[1].rotationY.setValue(180);
cubeLayers[2].position.setValue([mainComp.width/2 - cubeSize/2, mainComp.height/2, 0]);
cubeLayers[2].rotationY.setValue(90);
cubeLayers[3].position.setValue([mainComp.width/2 + cubeSize/2, mainComp.height/2, 0]);
cubeLayers[3].rotationY.setValue(-90);
cubeLayers[4].position.setValue([mainComp.width/2, mainComp.height/2 - cubeSize/2, 0]);
cubeLayers[4].rotationX.setValue(-90);
cubeLayers[5].position.setValue([mainComp.width/2, mainComp.height/2 + cubeSize/2, 0]);
cubeLayers[5].rotationX.setValue(90);
```

Рисунок 3.8 – Код функції createBasicCube

Дана функція перевіряє наявність активної композиції типу CompItem, після чого створює шість композицій для граней куба розміром 1080x1080 пікселів. Якщо користувач обрав текстуру через функцію selectImageForCubes, вона додається до кожної грані та центрується; інакше використовується суцільний шар сірого кольору. Грані розміщуються у 3D-просторі композиції з відповідними поворотами (rotationX, rotationY) для формування куба: передня, задня, ліва, права, верхня і нижня грані. Для управління кубом додається нульовий об’єкт «Basic Cube Ctrl», до якого прив’язуються всі грані через властивість parent.

### 3.5 Розробка модуля для роботи з текстом

Модуль обробки тексту в Adobe After Effects автоматизує роботу з текстовими шарами, спрощуючи створення складних анімацій і субтитрів. Він

пропонує набір інструментів для маніпуляції текстовими та шейповими шарами, зокрема встановлення опорної точки, декомпозицію тексту на символи або слова, створення фонових суцільних шарів і генерацію субтитрів із заданого тексту.

Для встановлення опорної точки текстових або шейпових шарів використовується механізм, який дозволяє розміщувати її за сіткою з дев’яти позицій. Цей процес враховує масштаб і розміри шару, автоматично коригуючи позицію після зміни опорної точки для забезпечення точного розташування.

Декомпозиція тексту на символи чи слова дає змогу створювати індивідуальні шари для кожного елемента. Під час розбиття тексту на символи модуль перевіряє наявність активної композиції типу CompItem і одного вибраного текстового шару (не параграфного). Текст розбивається на окремі символи, пропускаючи пробіли, і для кожного створюється новий шар із відповідною назвою. До кожного шару додається аніматор із виразом, який приховує всі символи, крім поточного, за допомогою селектора в режимі «Characters Excluding Spaces». Вихідний шар після цього вимикається. Аналогічно працює декомпозиція на слова: текстовий шар розбивається за пробілами за допомогою регулярного виразу, створюючи для кожного слова окремий шар із унікальною назвою у форматі «word [слово] – [індекс]». Кожен новий шар отримує аніматор із селектором виразу, який забезпечує видимість лише поточного слова, що полегшує створення анімацій для окремих слів.

Для створення фонового шару під текстом модуль перевіряє наявність активної композиції типу CompItem і одного вибраного текстового шару (не параграфного). У разі відповідності умов створюється суцільний шар, який розміщується під текстом і налаштовується як фон, враховуючи позицію та розміри тексту. Якщо умови не виконуються, користувач отримує повідомлення про помилку.

Генерація субтитрів дозволяє створювати субтитри в новій композиції на основі введеного тексту. Модуль перевіряє наявність активної композиції типу CompItem і валідність тексту. Текст розбивається на рядки, і для кожного непорожнього рядка створюються два текстові шари (один із аніматором для зміни

кольору) та прямокутний фоновий шар. Час відображення кожного рядка залежить від частоти кадрів (30/частота). Вирази використовуються для динамічного налаштування розміру та позиції фону. Додатково застосовується шар корекції з пресетом «text_heatwave.ffx» (якщо він доступний), а субкомпозиція вставляється в основну композицію. У разі помилок або успішного виконання користувач отримує відповідне повідомлення.

### 3.6 Розробка модуля для роботи з ключовими кадрами

Модуль роботи з ключовими кадрами автоматизує взаємодію з анімацією, забезпечуючи вирівнювання, копіювання, клонування, вставлення графіків швидкості, реверсування ключових кадрів, а також застосування ефектів відскоку для масштабу та позиції. Кожен інструмент плавно інтегрується в робочий процес, спрощуючи створення складних анімацій.

Функція вирівнювання ключових кадрів перерозподіляє вибрані ключові кадри властивості в активній композиції типу CompItem у рівномірну послідовність. Вона перевіряє наявність проекту, активної композиції та щонайменше двох вибраних ключових кадрів. Кадри видаляються, а потім відтворюються з однаковим інтервалом, починаючи з часу першого кадру,

Копіювання графіку швидкості зберігає характеристики анімації вибраної властивості з двома або більше ключовими кадрами. Перевіряється наявність проекту, активної композиції та відповідної властивості. Дані про типи інтерполяції, параметри плавності та чи є властивість Time Remap зберігаються в глобальній змінній для подальшого використання. Після успішного копіювання видається повідомлення, що спрощує перехід до наступного етапу.

Клонування ключових кадрів відносно позиції програвача копіює вибрані ключові кадри однієї властивості на той самий таймлайн, починаючи з поточного часу програвача. Перевіряється наявність проекту, композиції, однієї властивості та хоча б одного ключового кадру. Кадри зберігаються з їхніми значеннями, часом відносно першого кадру, типами інтерполяції та параметрами плавності. При Shift + Клік кадри вставляються в зворотному порядку, з коригуванням плавності для

Time Remap, щоб забезпечити монотонність. При Alt + Клік нові кадри виділяються. У разі помилок, наприклад, порушення монотонності для Time Remap, видається повідомлення.

Вставлення графіку швидкості застосовує збережені характеристики анімації до іншої властивості з двома або більше вибраними ключовими кадрами. Перевіряється наявність проекту, композиції, властивості та попередньо скопійованих даних. Типи інтерполяції та параметри плавності переносяться, зберігаючи оригінальні значення кадрів. Для Time Remap забезпечується монотонність, а для інших властивостей швидкість коригується, щоб уникнути від'ємних значень. Після виконання видається повідомлення про успіх.

Реверсування ключових кадрів обмінює значення та характеристики двох вибраних ключових кадрів однієї властивості. Перевіряється наявність проекту, композиції, однієї властивості та рівно двох ключових кадрів. Для Time Remap значення інвертуються відносно тривалості шару, а для інших властивостей просто міняються місцями. Типи інтерполяції та параметри плавності також обмінюються з відповідним коригуванням швидкості для Bezier-інтерполяції. Якщо шар відключений, він тимчасово вмикається для виконання операції, а потім повертається до початкового стану.

Застосування ефекту відскоку для масштабу додає вираз анімації до вибраних властивостей, створюючи ефект пружного руху. Перевіряється наявність проекту, композиції та хоча б однієї властивості, що підтримує вирази. До шару додаються слайдери для амплітуди, частоти та затухання, які керують виразом, що додає коливання до значення властивості після найближчого ключового кадру. У разі помилки видається повідомлення для кожної властивості.

Застосування ефекту відскоку для позиції аналогічно додає вираз пружного руху до вибраних властивостей позиції. Слайдери для амплітуди, частоти та затухання створюються або використовуються наявні, а вираз додає коливання до руху шару після ключового кадру. Це створює природний ефект відскоку, а повідомлення підтверджує успішне виконання або вказує на помилки.

## 3.7 Висновки

У третьому розділі обґрунтовано вибір технологій для розробки розширення для Adobe After Effects, зокрема JavaScript, ExtendScript API, CEP-панелі (HTML, CSS, JavaScript), Node.js, а також середовищ Visual Studio Code і ExtendScript Toolkit. Описано створення чотирьох програмних модулів для автоматизації роботи з шарами, 3D-об’єктами, текстом і ключовими кадрами, кожен із яких оптимізує рутинні завдання. Описано інтерфейс із вкладками для зручної навігації та ефективної взаємодії з функціоналом. Детально описано функції модулів, включаючи сортування шарів, створення текстурованих 3D-кубів, декомпозицію тексту та вирівнювання кейфреймів. Реалізовані модулі протестовано для забезпечення стабільності та сумісності з Adobe After Effects, що сприяє підвищенню продуктивності робочих процесів.

# 4 ТЕСТУВАННЯ ТА ПРАКТИЧНЕ ЗАСТОСУВАННЯ ПРОГРАМНОГО МОДУЛЯ

## 4.1 Тестування програмного модуля

Тестування програмного забезпечення є критично важливим етапом у процесі розробки будь-якого програмного продукту. Це процес перевірки програмного забезпечення на наявність помилок і дефектів, а також оцінка його функціональності відповідно до встановлених вимог. Тестування може включати різні методи, які залежать від етапу розробки і специфіки продукту. Один з основних видів тестування – функціональне тестування, яке спрямоване на перевірку коректності виконання основних функцій програми. Це включає в себе перевірку всіх функціональних можливостей, які повинні бути реалізовані, щоб програма працювала відповідно до вимог замовника [22].

Іншим важливим видом є тестування продуктивності, яке дозволяє оцінити швидкість роботи програми, її час відгуку та здатність працювати під високим навантаженням або з великими обсягами даних. Це тестування необхідне для виявлення проблем з оптимізацією та стабільністю роботи, що може вплинути на загальний досвід користувача. Крім того, важливе тестування безпеки, яке включає перевірку програми на вразливості, можливість витоків даних чи зловмисного втручання. Тестування безпеки необхідне для захисту користувачів від потенційних загроз і забезпечення цілісності даних.

Також існує тестування сумісності, яке перевіряє, як програма працює на різних операційних системах і апаратних платформах, а також чи виникають конфлікти або проблеми при взаємодії з іншими програмами чи версіями програмного забезпечення.

Для розробленого розширення для автоматизації робочих процесів у Adobe After Effects тестування є надзвичайно важливим для забезпечення його стабільності та ефективності. Оскільки розширення взаємодіє з різними компонентами After Effects, зокрема шарами, 3D-об'єктами, текстом, кейфреймами та аудіо, необхідно провести комплексне функціональне тестування для перевірки

правильності роботи кожної з цих функцій. Це дозволить переконатися в тому, що розширення працює без помилок та виконує всі визначені завдання.

Особливу увагу слід приділити тестуванню сумісності, оскільки розширення повинно працювати на різних операційних системах та версіях After Effects. Це забезпечить коректну роботу розширення на різних конфігураціях комп'ютерів і в різних робочих середовищах. Таким чином, тестування є важливим етапом для забезпечення надійної роботи розширення та задоволення вимог користувачів.

Під час першого етапу тестування програмного модуля перевіряється реакція програми на спробу сортування шарів без їх попереднього виділення. Якщо користувач намагається виконати операцію сортування без виділення хоча б одного шару, програма має вивести відповідне повідомлення, інформуючи користувача про необхідність вибору шару для виконання операції. Інтерфейс програми з повідомленням, що вказує на необхідність виділення шарів перед сортуванням показано на рисунку 4.1.

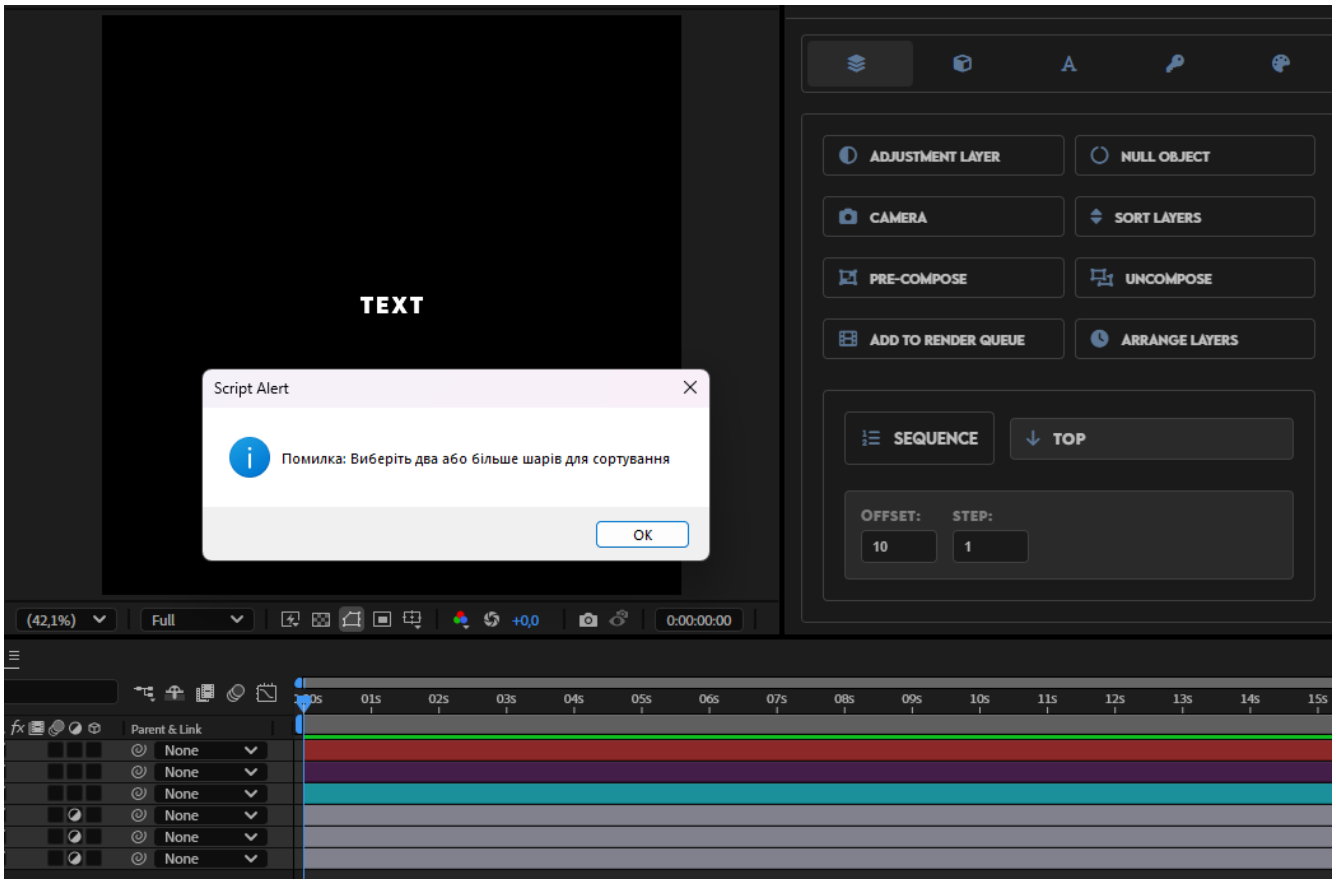



Рисунок 4.1 – Повідомлення про необхідність виділення шарів перед сортуванням

Під час тестування програми було перевірено, як вона реагує на ситуацію, коли користувач намагається додати композицію до черги рендеру, не вибравши її. У такому випадку програма виводить повідомлення про помилку, що інформує користувача про необхідність вибору композиції перед додаванням її до черги рендеру. Це допомагає уникнути помилок і забезпечує коректну роботу програми.

На рисунку 4.2 показано вікно з повідомленням про помилку, яке з'являється в інтерфейсі програми у разі спроби виконати дію без вибору композиції.

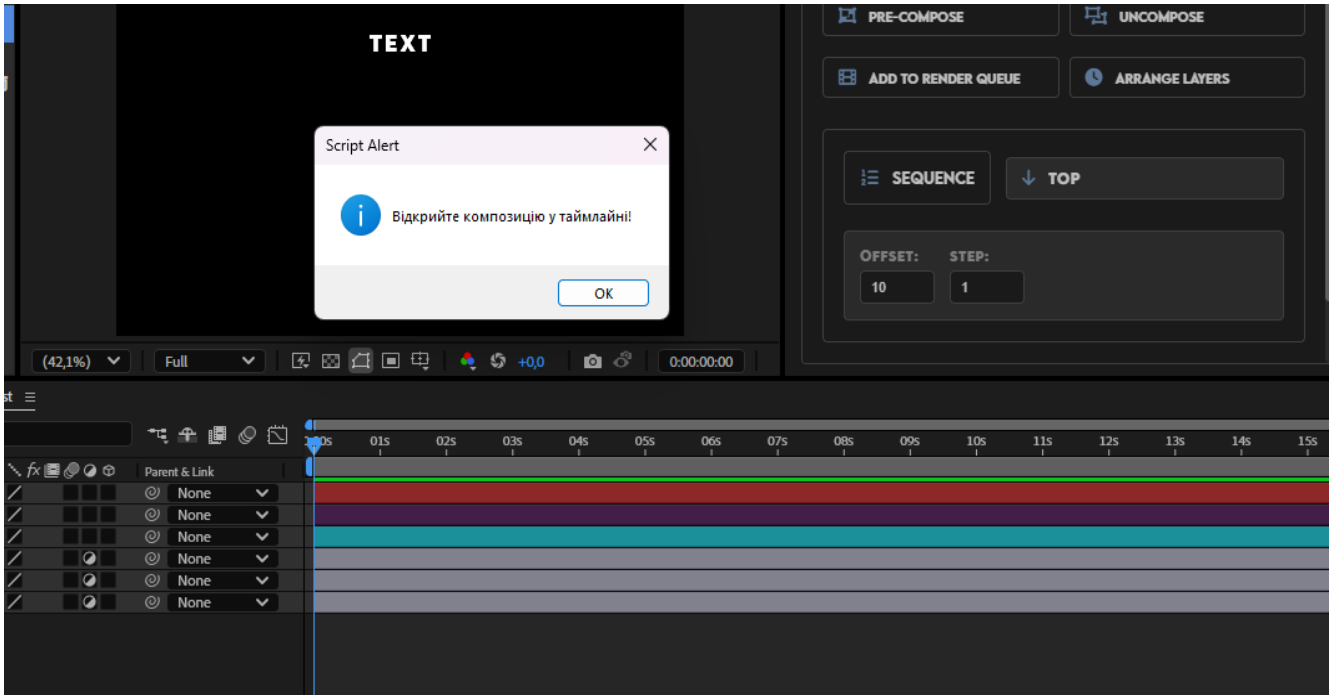



Рисунок 4.2 – Повідомлення про помилку при спробі додати композицію в чергу рендеру без її вибору

У процесі тестування функції створення 3D-об'єктів, зокрема при створенні куба, перевіряється взаємодія користувача з опцією вибору зображення. Якщо користувач не вибирає фото для текстурування куба, система автоматично створює стандартний куб без текстури. Після цього програма виводить повідомлення про те, що фото не було обрано, і нагадує користувачеві про необхідність вибору зображення для текстурування об'єкта.

На рисунку 4.3 представлено інтерфейс програми, де користувач створює 3D-куб без текстури.

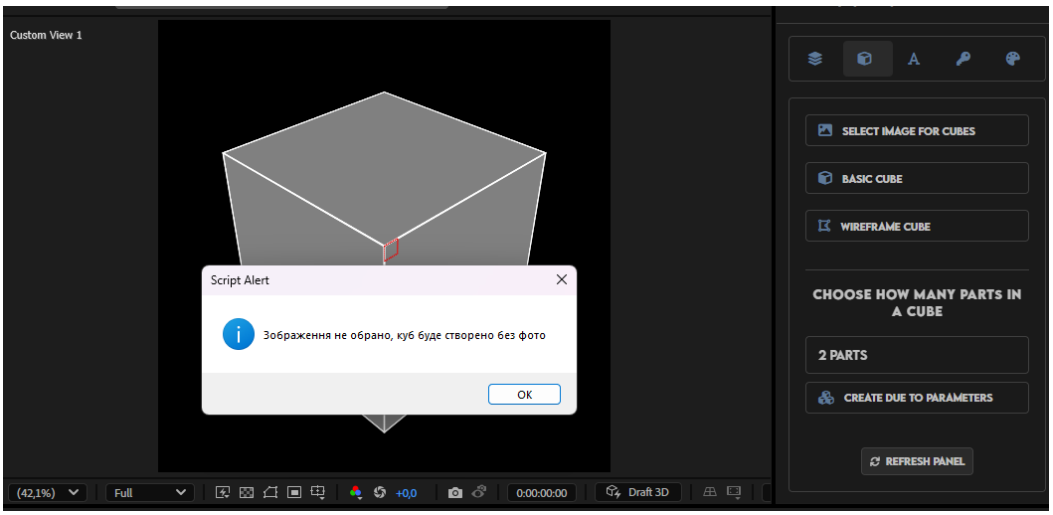



Рисунок 4.3 – Повідомлення про відсутність фото для текстурування 3D-куба

На рисунку 4.4 показано успішно створений куб із зображенням, обраним через провідник.

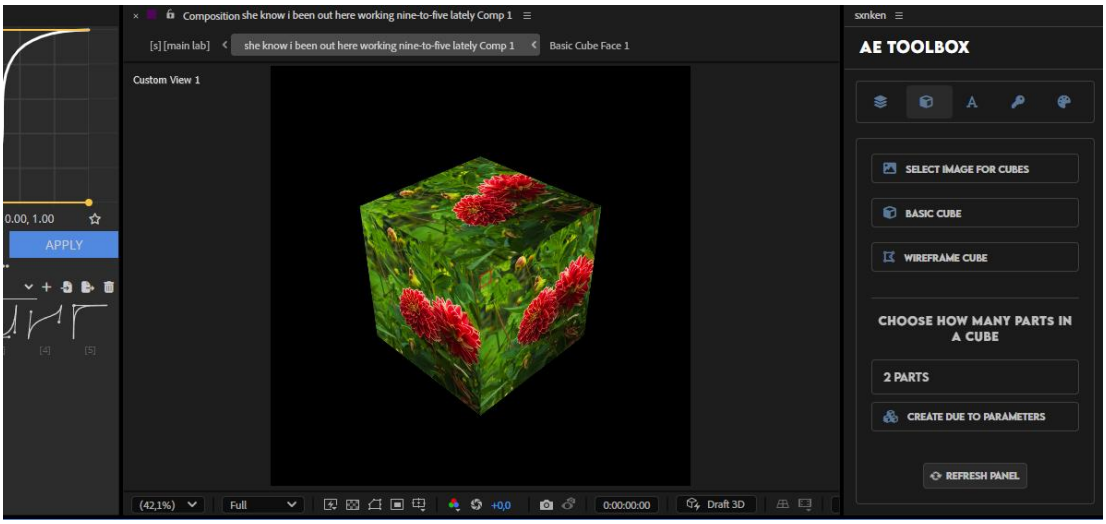



Рисунок 4.4 – Успішно створений куб з обраним фото

У процесі тестування продуктивності було перевірено швидкість роботи розширення, час відгуку та його здатність стабільно функціонувати при високому навантаженні. Для цього в проєкті було використано 20–30 шарів, що є типовим сценарієм для складних композицій. Було протестовано сортування, оновлення та фільтрацію шарів – усі функції працювали коректно, без значних затримок або збоїв. Отримані результати свідчать про належну оптимізацію та стабільність роботи розширення при взаємодії з великими обсягами даних, що показано на рисунку 4.5.

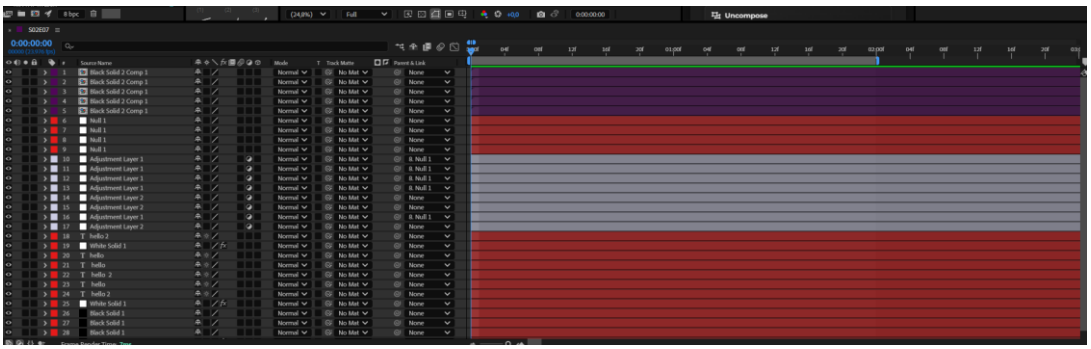


Рисунок 4.5 – Робота розширення з великою кількістю шарів у проєкті

У процесі тестування сумісності основна розробка та перевірка функціональності розширення здійснювалася у середовищі Adobe After Effects 2025. Водночас було проведено додаткове тестування на версії CC 2022 для перевірки зворотної сумісності. Це дозволило виявити відмінності в API та поведінці інтерфейсів між версіями.

На рисунку 4.6 відображено результати тестування програмного модуля для Adobe After Effects, які підтверджують його стабільну роботу без помилок у різних версіях застосунку.

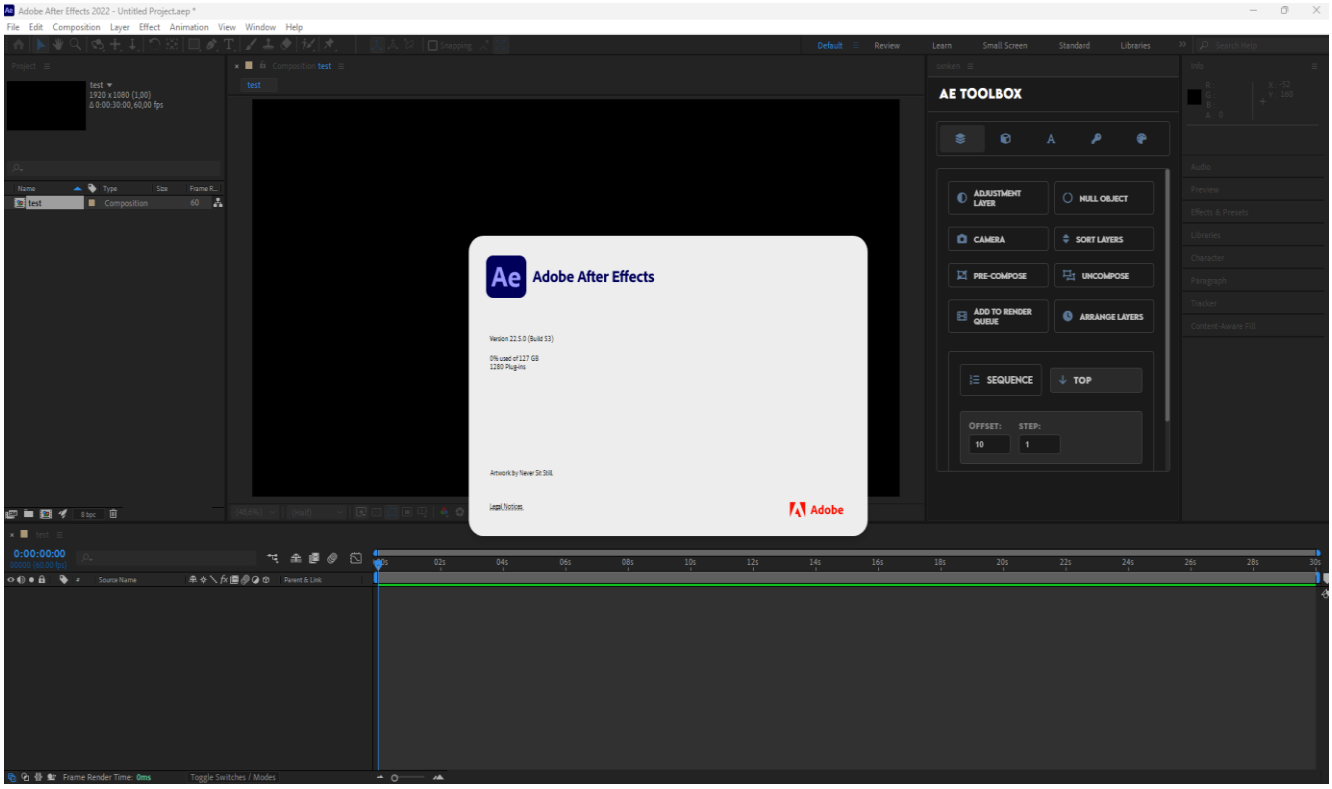



Рисунок 4.6 – Тестування сумісності програмного модуля на різних версіях застосунку

## 4.2 Розробка інструкції користувача

Інструкція користувача є невід’ємною частиною документації програмного забезпечення. Вона описує порядок встановлення, запуску, а також правила взаємодії з основними модулями розширення. Для стабільної та безпечної роботи системи необхідно дотримуватись мінімальних або рекомендованих технічних вимог, які наведено в таблиці 4.1.

Таблиця 4.1 – Вимоги до апаратного забезпечення

| Вимоги до конфігурування | Рекомендовано | Мінімально |
|---|---|---|
| Процесор | Intel Core i9 або Apple M2 | Intel Core i3 |
| Оперативна пам'ять | 32 ГБ RAM | 8 ГБ RAM |
| Вільний простір на диску | 10 ГБ | 2 ГБ |
| Відеокарта | NVIDIA RTX 3070 або вища | NVIDIA GTX 1050 або аналог |
| Операційна система | Windows 11 / macOS Ventura | Windows 10 / macOS Mojave |

Інструкція користувача для розширення, створеного з метою автоматизації робочих процесів у Adobe After Effects, допомагає користувачам швидко ознайомитися з функціоналом та ефективно використовувати його в щоденній роботі. Вона охоплює всі основні етапи взаємодії – від відкриття розширення до виконання дій у модулях редагування шарів, ключових кадрів, 3D-об’єктів та роботи з текстом, забезпечуючи зрозумілий алгоритм дій для аніматорів, дизайнерів і монтажерів.

Після встановлення розширення та запуску Adobe After Effects, користувач відкриває розширення через меню Window → Extensions → AE TOOLBOX. У результаті з’являється головна панель, що містить доступ до функціональних модулів.

Після запуску розширення користувач переходить до модуля роботи з шарами, який пропонує зручні інструменти для створення та керування шарами.

Майже кожна кнопка супроводжується підказкою, при наведені на неї, що пояснює її дію, і виконує різні функції залежно від типу натиску – звичайний Click, Shift + Click або Alt + Click. Наприклад, кнопка «Adjustment Layer» пропонує створення окремих шарів для кожного вибраного елемента при звичайному натисканні та єдиний шар на всю композицію при Alt + Click. На рисунку 4.9 показано інтерфейс модуля при наведенні на кнопку Adjustment Layer.


Рисунок 4.9 –Інтерфейс модуля роботи з шарами з інтерактивними підказками та багатофункціональними кнопками

На рисунку 4.10 показано результат створення Adjustment Layer для декількох композицій.

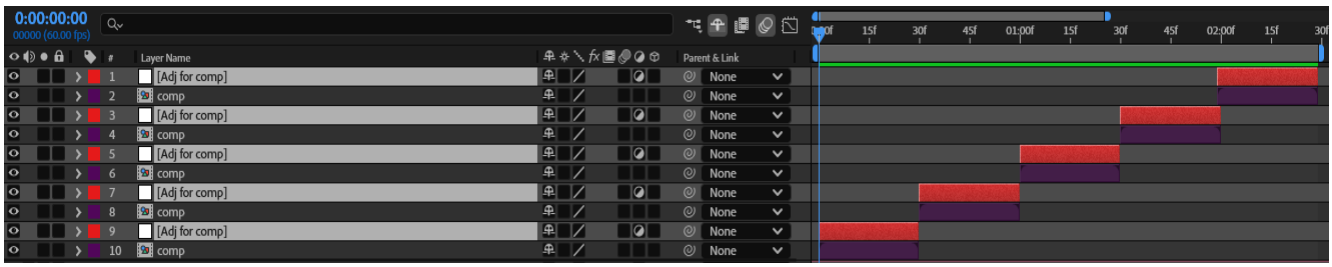



Рисунок 4.10 – Результат створення шару типу Adjustment Layer для декількох композицій

На рисунку 4.11 показано результат зміщення декількох шарів за допомогою функції Sequence з кроком зміщенням для кожного шару у 10 кадрів.

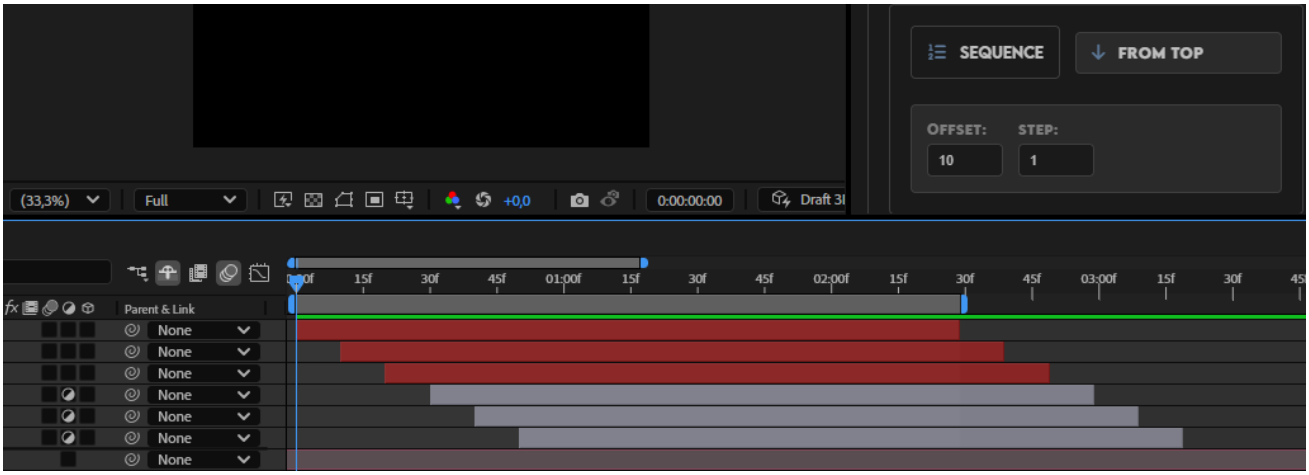



Рисунок 4.11 – Результат зміщення декількох шарів за допомогою функції Sequence з заданим кроком зміщенням

Після роботи з шарами користувач може перейти до модуля створення 3D-об'єктів. У цьому розділі розширення користувач може створювати різні типи 3D-кубів. Є можливість вибрати стандартний куб, налаштувати для нього текстуру, створити куб у стилі wireframe або поділити куб на кілька частин для більш складних анімацій. Користувач може редагувати кожну частину окремо, змінюючи її розміри, позицію та інші властивості. На рисунку 4.12 показано приклад створення куба, поділеного на дві частини, з можливістю контролю кожної з частин за допомогою Null Object. Це дозволяє незалежно анімувавати кожну частину куба, зберігаючи загальну структуру, що зручно для створення складніших анімацій та трансформацій.

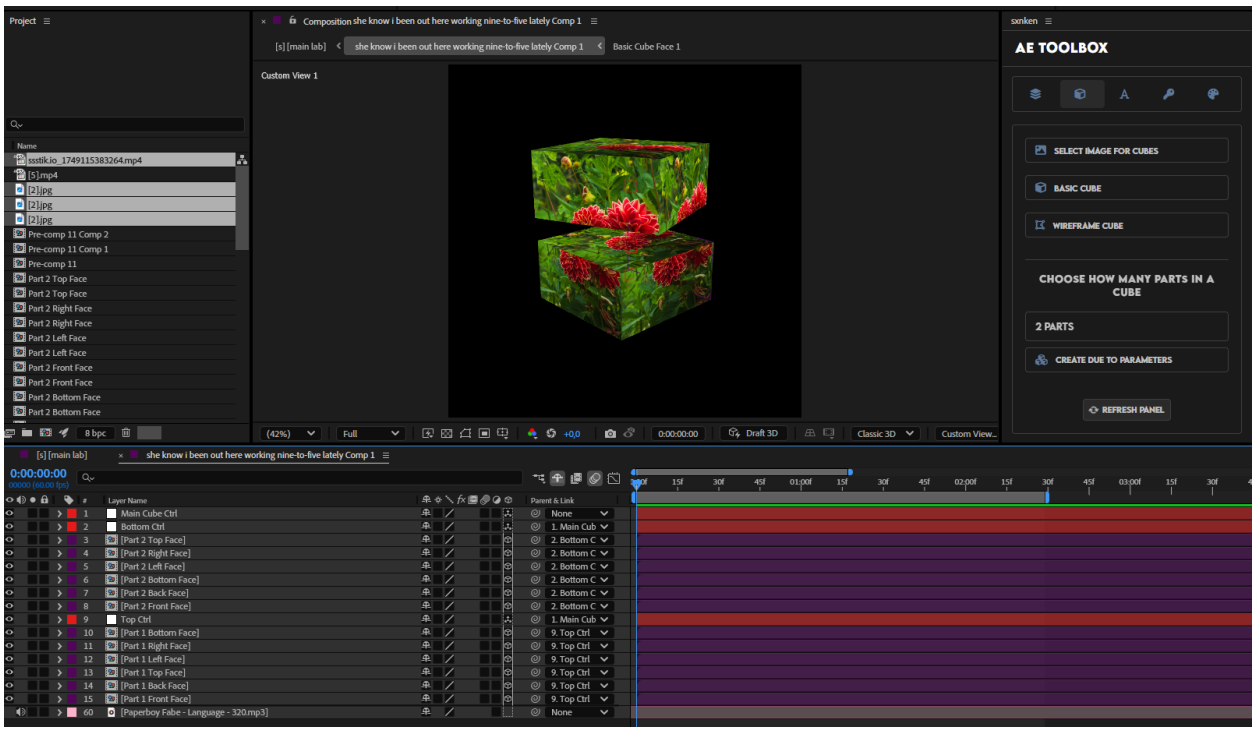



Рисунок 4.12 – Результат створення кубу поділеного на дві частини

Після роботи з 3D-об'єктами користувач може перейти до модуля взаємодії з текстом. У розділі роботи з текстом користувач може змінювати якірну точку тексту, що дозволяє точніше налаштовувати його розташування та анімацію. Крім того, є можливість декомпозувати текст на окремі слова та букви, а також створювати фоновий шар для виділеного текстового шару, що дає більшу гнучкість у створенні анімацій для кожного елементу тексту. Декомпозиція тексту може виконуватися двома основними способами: за допомогою виразів (expressions) та за допомогою аналітичного розбиття (фізичного розбиття). Обидва методи дозволяють розкласти текст на окремі елементи (символи чи слова) для подальшої анімації, але мають різні підходи та застосування.

Декомпозиція за допомогою виразів дозволяє створювати анімації без необхідності дублювання шарів, що оптимізує робочий процес і полегшує редагування тексту. Натомість аналітичне розбиття забезпечує створення окремих шарів для кожного символу чи слова, що ідеально підходить для складних анімацій. На рисунку 4.13 показано реалізацію розбиття текстового шару на слова, де кожне слово стає окремим шаром із власними параметрами трансформації, що дозволяє незалежно анімувати їх у композиції.

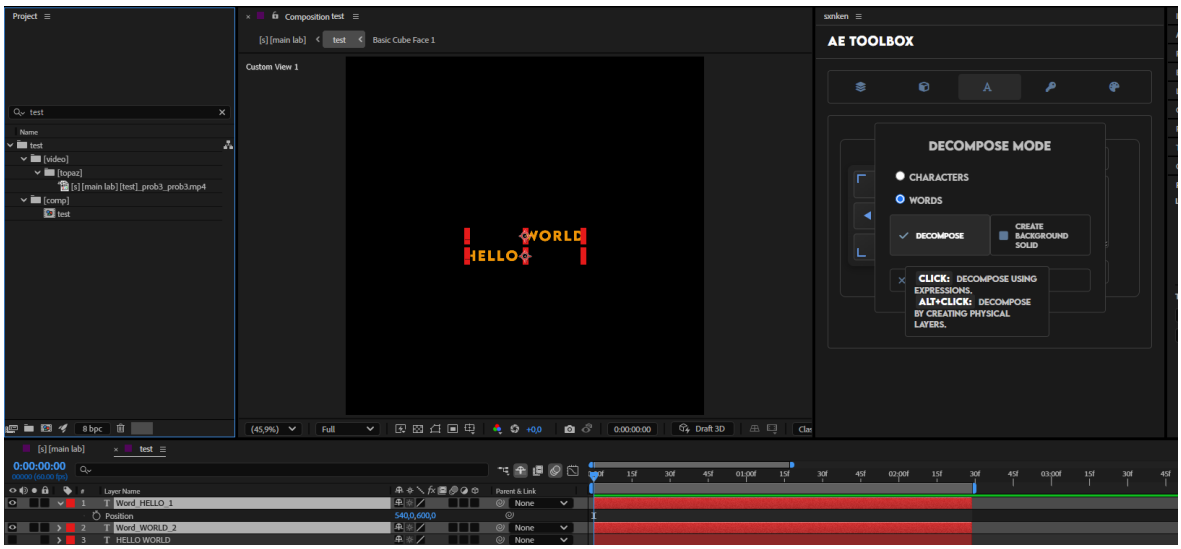



Рисунок 4.13 – Результат розбиття тексту на слова

Розбиття тексту на слова перетворює один текстовий шар на кілька шарів для кожного слова, зберігаючи їх на тій самій позиції. Основний шар стає невидимим, і тепер можна редагувати кожен окремий шар. Це спрощує створення анімацій та роботу з текстом, даючи більше контролю над кожним словом або буквою при застосуванні ефектів чи анімацій.

На рисунку 4.14 показано реалізацію зміни якірної точки текстового шару на одну з 9 точок (Middle Right), що дозволяє більш точно налаштовувати розташування тексту та спрощує взаємодію з ним.

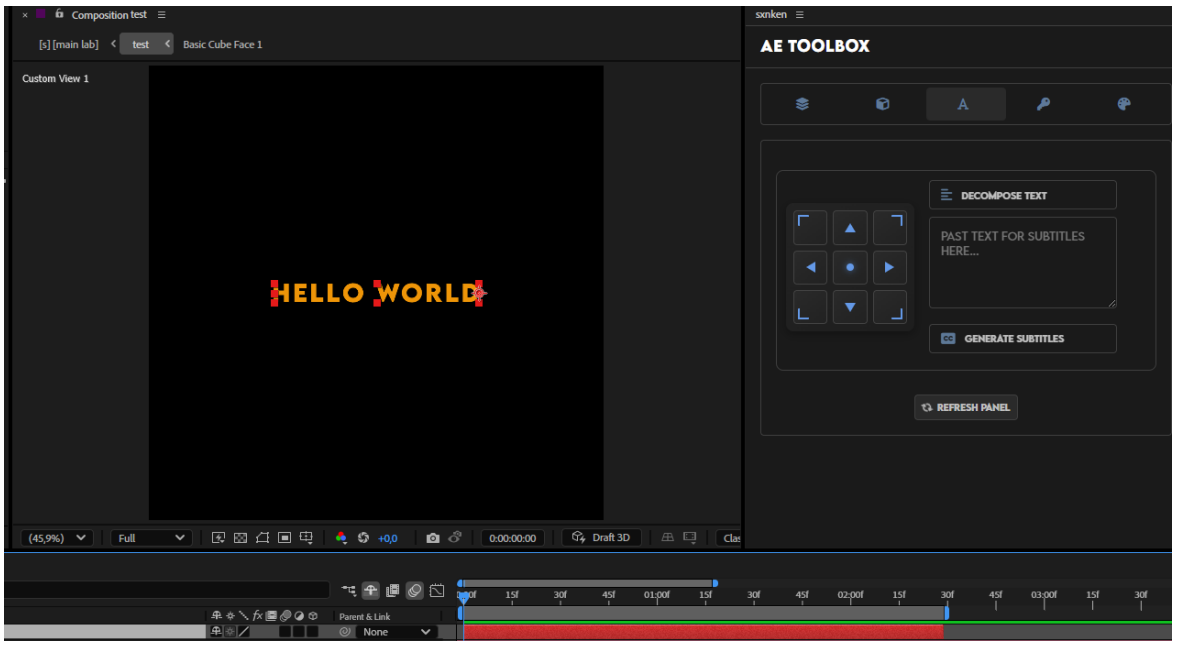



Рисунок 4.14 – Реалізація зміни якірної точки текстового шару

Для створення субтитрів користувач вводить текст у текстове меню, де кожен рядок вважається окремим шаром у композиції. Шари розділені інтервалом у 30 кадрів для зручного маніпулювання. Для кожного текстового шару створюються два шари: основний з анімацією кольору через Animator (колір #FF9600) та додатковий із синхронізованим вмістом. Також додається фоновий шар, який динамічно адаптується до розміру тексту за допомогою виразів, із чорним заливанням та прозорістю 60%. Нова композиція створюється з тривалістю, що залежить від кількості непорожніх рядків, і додається до основної композиції. Adjustment Layer із пресетом text_heatwave.ffx застосовується для додаткових ефектів. Результат створення субтитрів показано на рисунку 4.15.

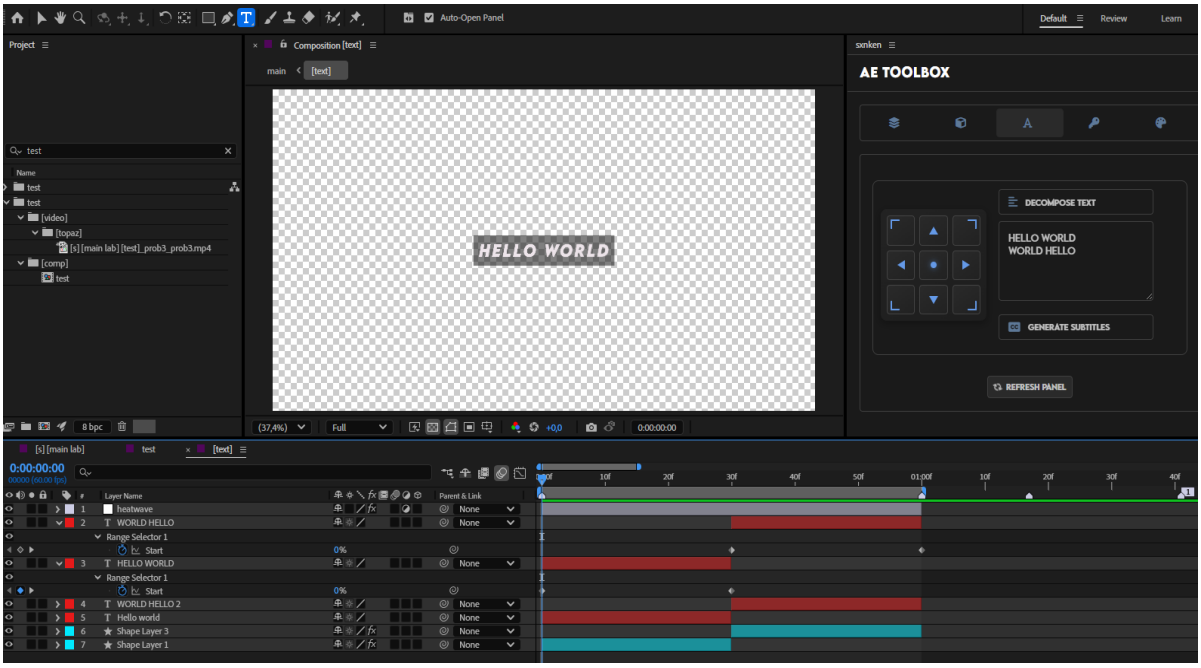



Рисунок 4.15 – Результат створення субтитрів за вказаним текстом

Після роботи з текстом користувач може перейти до модуля взаємодії з кейфреймами. У розділі роботи з ключовими кадрами можна виконувати низку корисних дій для спрощення та оптимізації анімаційного процесу. Серед функцій є вирівнювання ключових кадрів у часі, копіювання та вставлення графіків швидкості, реверсування порядку ключових кадрів, клонування їх відносно позиції програвача, а також застосування виразів для створення ефектів, таких як «bounce» для масштабування чи позиції.

На рисунку 4.16 показано положення кейфреймів до реалізації функції Align Keyframes.

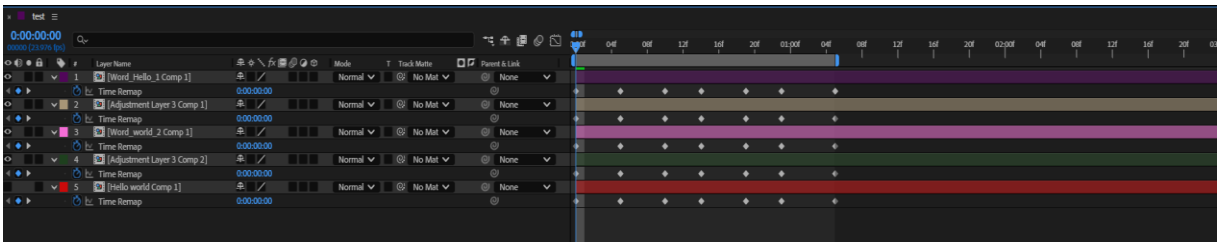


Рисунок 4.16 – Кейфрейми до реалізації функції Align Keyframes

На рисунку 4.17 показано положення кейфреймів після реалізації функції Align Keyframes.

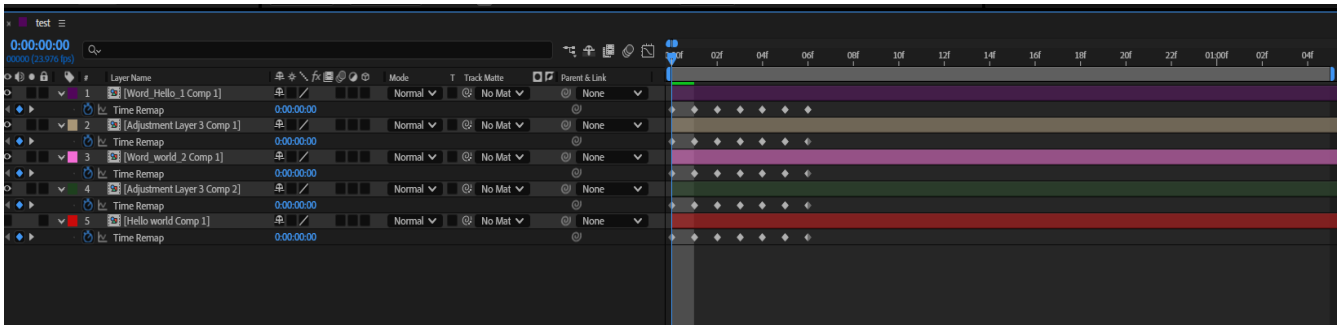


Рисунок 4.17 – Кейфрейми після реалізації функції Align Keyframes

Функція вирівнювання ключових кадрів дозволяє розташувати їх рівномірно по часу між першим і останнім кадром. Це зручно, коли потрібно зробити плавну анімацію з однаковими інтервалами між подіями, не виставляючи кожен кадр вручну.

Також користувач може копіювати та вставляти графік, який був застосований між певними ключовими кадрами. Наприклад, якщо в одному шарі створено анімацію зміщення з плавним прискоренням і уповільненням, користувач може скопіювати графік цієї позиції та застосувати його до ключових кадрів іншого шару – таким чином обидві анімації матимуть однакову динаміку без необхідності вручну повторювати форму кривої.

На рисунку 4.18 показано графік анімації текстового шару по позиції, який буде скопійовано для іншого шару.

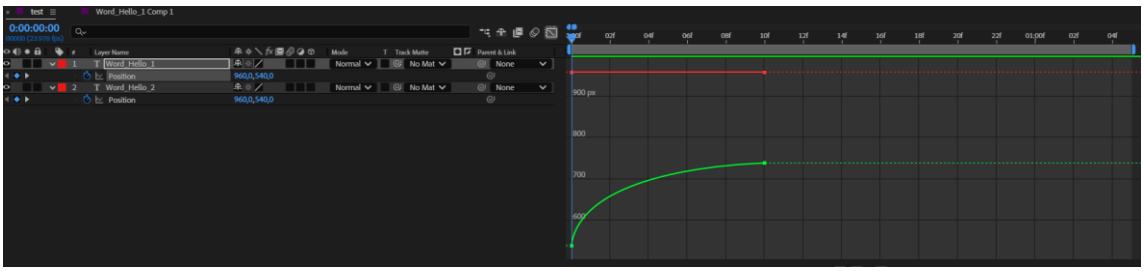


Рисунок 4.18 – Графік анімації позиції текстового шару

На рисунку 4.19 та 4.20 показано реалізацію копіювання графіку анімації для іншого текстового шару за допомогою функції Copy Keyframes Graph та Paste Keyframes Graph.

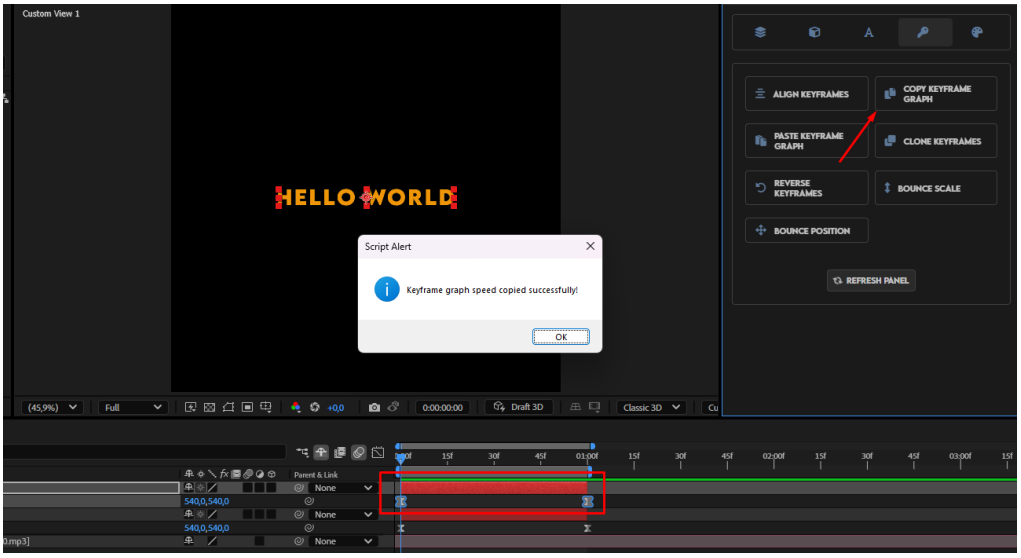



Рисунок 4.19 – Реалізація копіювання та вставлення графіку анімації для іншого шару

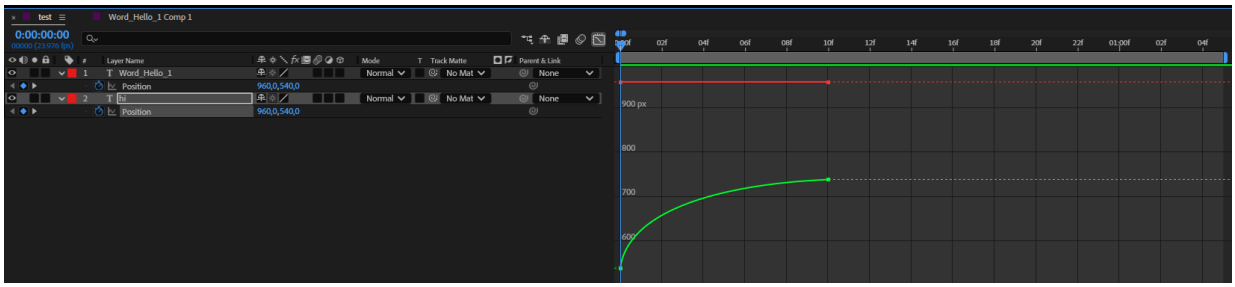


Рисунок 4.20 – Скопійований графік анімації за позицією для іншого шару

Для використання функції реверсу ключових кадрів Reverse Keyframes користувачу необхідно відкрити активну композицію в Adobe After Effects,

виділити рівно одну анімовану властивість (наприклад, позицію, масштаб або Time Remap) на шарі та вибрати два ключові кадри. Після цього виконання функції змінить порядок обраних ключових кадрів, зберігаючи їхні значення, типи інтерполяції та параметри легкості (ease), що дозволяє швидко створювати зворотні анімації.

Для того щоб скористатися функцією клонування ключових кадрів потрібно виділити бажані кейфрейми та настиснути на кнопку Clone Keyframes, дана функція забезпечує клонування ключових кадрів відносно поточної позиції програвача. На рисунку 4.20 показано реалізацію клонування ключових кадрів позиції текстового шару.

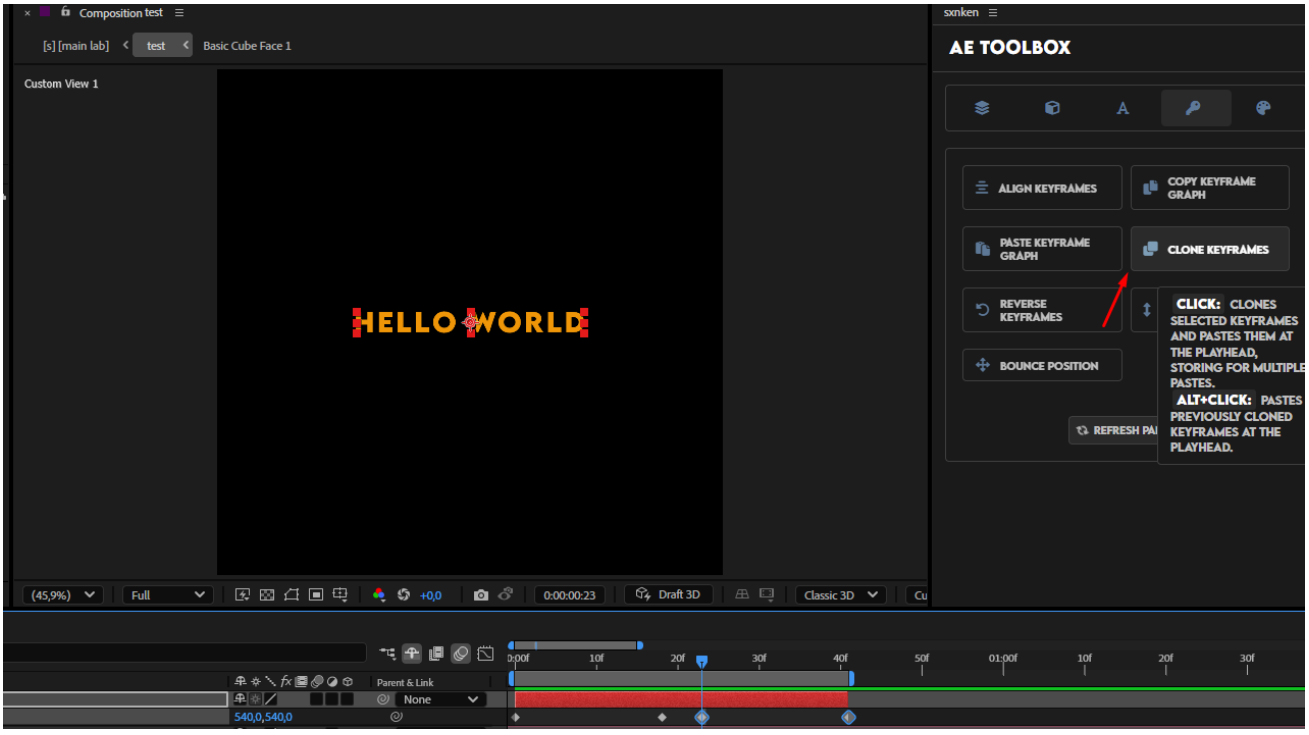



Рисунок 4.20 – Результат клонування ключових кадрів

Для застосування ефектів BounceScale або BouncePosition створіть два ключові кадри (початок і кінець руху), виділіть їх і натисніть відповідну кнопку (Bounce Scale або Bounce Position). Вирази використовують слайдер-контроли (Amplitude, Frequency, Decay) для налаштування амплітуди, частоти та загасання, забезпечуючи живу й природну анімацію.

## 4.3 Практичне застосування програмного модуля

Програмний модуль значно підвищує ефективність роботи в Adobe After Effects, автоматизуючи рутинні операції з шарами, 3D-об’єктами, текстом і кейфреймами. Модуль призначений для відеомонтажерів, аніматорів і дизайнерів, які прагнуть оптимізувати робочий процес, зменшити час на базові завдання та зосередитися на креативних аспектах проєкту, що особливо важливо в умовах обмежених дедлайнів.

Основні функції модуля охоплюють кілька ключових напрямів. У роботі з текстом модуль забезпечує розбиття тексту на символи чи слова, автоматичну зміну якірної точки для спрощення анімації та створення фонових солідів для кожного текстового елемента, а також створення субтитрів для введеного тексту. Для шарів модуль створення Adjustment Layer і Null Object, сортування шарів за типом, створення та розпакування прекомпозицій, додавання композицій до черги рендерингу, позиціонування шарів відносно програвача або початку композиції, а також зміщення шарів з можливістю задання кроку зміщення.

У роботі з 3D-об’єктами реалізовано створення базових і сегментованих кубів із автоматичним текстуруванням. Для кейфреймів модуль пропонує вирівнювання, копіювання графіків анімації та їх реверс, а також клонування та анімацію коливання за допомогою виразів, що значно спрощує налаштування анімаційних ефектів.

Нижче наведено детальний порівняльний аналіз виконання основних операцій без розширення (ручний спосіб) і з використанням модуля, із зазначенням кількості кроків для кожного процесу.

Модуль «Layers» функція створення Adjustment Layer до обраного шару без програмного модуля:

1. Відкрити головне меню «Layer».
2. Обернути курсор до підменю «New».
3. Клікнути на «Adjustment Layer» для створення нового шару.
4. Перейти до панелі «Composition» і вручну встановити часові межі (in-point і out-point) відповідно до бажаного сегмента.

5. Перетягнути шар у часовій шкалі, щоб розмістити його над потрібним шаром у стеку.

6. Перевірити правильність розташування та скоригувати, якщо необхідно.

Кількість кроків: 4, приблизний час: 14 секунд

З програмним модулем:

1. Виділити один шар у композиції.
2. Клікнути кнопку «Add Adjustment Layer» у панелі розширення.

Кількість кроків: 2, приблизний час: 5 секунд

Автоматичне налаштування скорочує процес у 2,8 рази, економлячи до 9 секунд на операцію.

На рисунку 4.21 та 4.22 показано як створюється Adjustment Layer без використання програмного модуля.

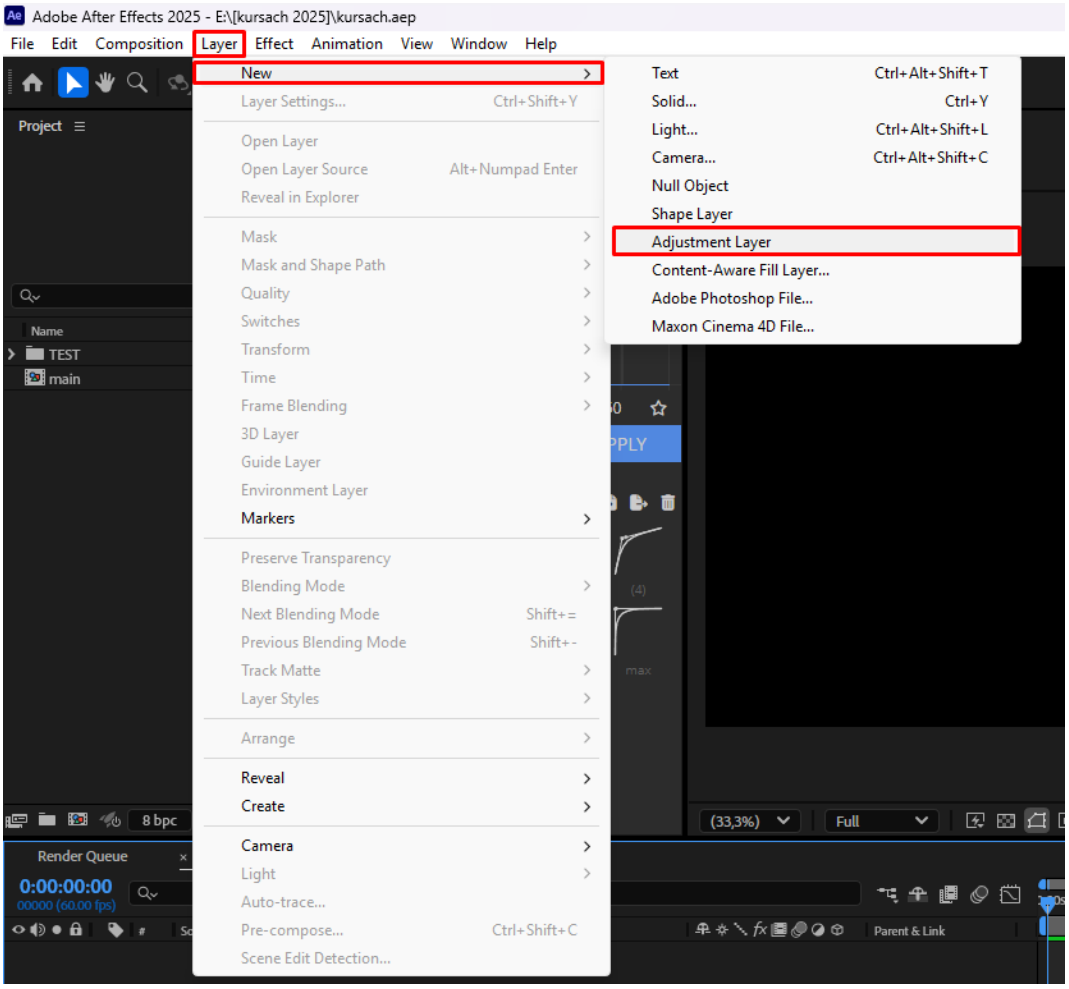



Рисунок 4.21 – Створення коригуючого шару без використання програмного модуля

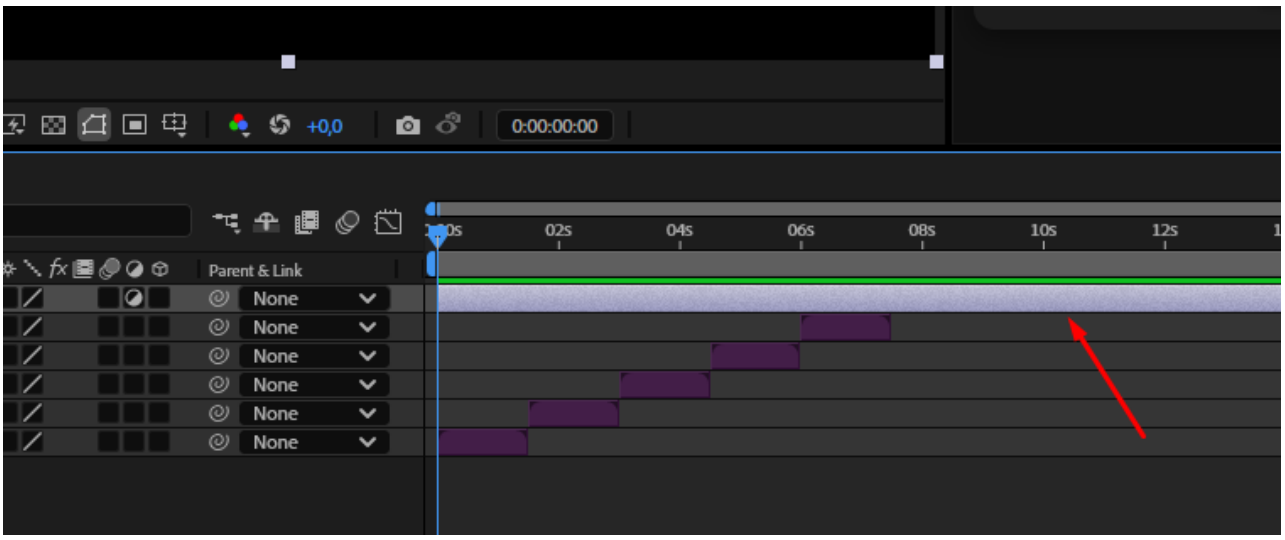



Рисунок 4.22 – Створення коригуючого шару без використання програмного модуля та його положення у композиції

На рисунку 4.23 показано створення коригуючого шару з використанням програмного модуля та його положення у композиції відповідно до обраного об’єкту.

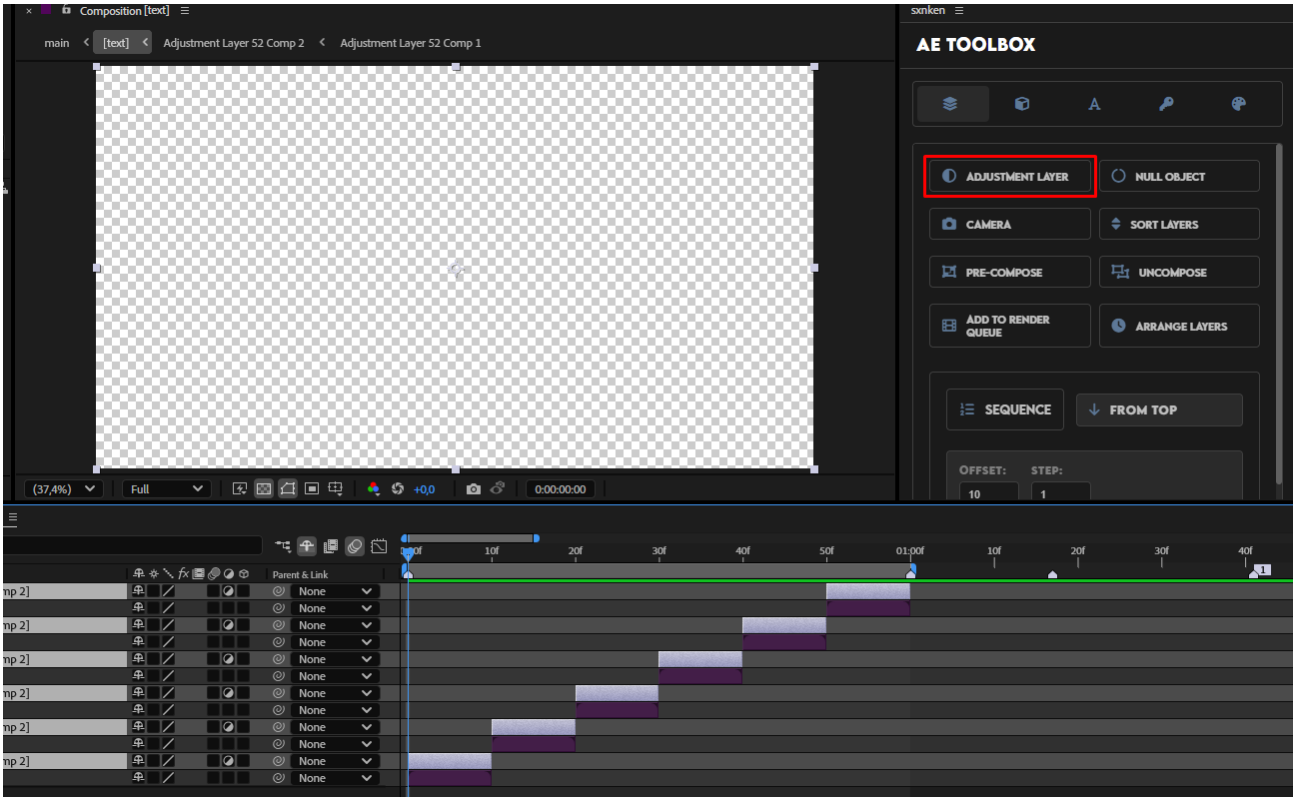



Рисунок 4.23 – Створення коригуючого шару та його положення у композиції з використанням програмного модуля

Модуль «Layers» функція сортування шарів за типом без програмного модуля:

1. Виділити всі шари (наприклад, 10 шарів).
2. Перетягнути кожен шар вручну в часовій шкалі, визначаючи тип (Solid, Adjustment, Null, Comp).
3. Перевірити порядок і переміщати шари, якщо типи змішані.
4. Налаштувати часові межі для кожного шару.

Кількість кроків: 4, приблизний час для 10 шарів: 30–35 секунд.

З програмним модулем:

1. Виділити два або більше шарів (наприклад, 10 шарів).
2. Клікнути кнопку «Sort Layers By Type».

Кількість кроків: 2, приблизний час: 5 секунд.

Автоматичне сортування за типом (Solid → Adjustment → Null → Comp) скорочує процес у 8–10 разів для 10 шарів, економлячи до 30 секунд на операцію.

Модуль «Cubes» функція створення базового 3D-куба без програмного модуля:

1. Створити нову композицію розміром 1080x1080.
2. Додати шість нових композицій для граней через «Composition» → «New Composition».
3. У кожній композиції додати суцільний шар через «Layer» → «New» → «Solid».
4. Вручну налаштувати розмір і колір кожного шару.
5. Повернутися до головної композиції, додати всі грані як шари.
6. Увімкнути 3D для кожного шару і вручну налаштувати повороти для створення куба.
7. Перевірити правильність 3D-структури.

Кількість кроків: 7, приблизний час: 80-100 секунд.

З програмним модулем:

1. Перейти до модулю 3D-objects у розширенні.
2. Клікнути кнопку «Create Basic Cube» у відповідній панелі.

Кількість кроків: 2, приблизний час: 10 секунд. Автоматичне створення куба скорочує процес у 8-10 разів.

На рисунку 4.24 показано створений куб з використанням програмного модуля та кількість створених композицій (граней) а також null-об’єктів що керують позицію, ротацію та масштабування кубу.

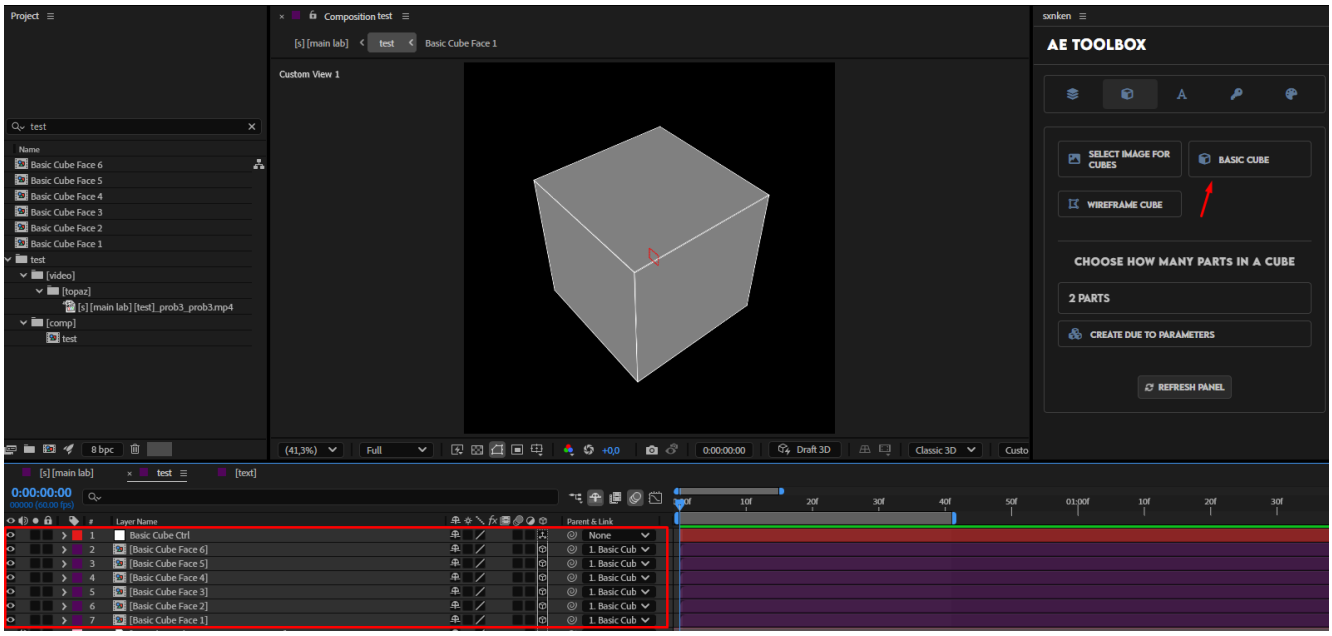



Рисунок 4.24 – Результат створення 3D-кубу з використанням програмного модуля

Модуль «Text» функція розбиття тексту на слова та символи без програмного модуля:

1. Виділити текстовий шар у композиції.
2. Копіювати шар для кожного символу (для символів) або слова (для слів) вручну (Ctrl+C, Ctrl+V).
3. Додати аніматор через «Text» → «Add Animation» → «Property».
4. Налаштувати селектор на «Characters Excluding Spaces» (для символів) або «Words» (для слів) вручну.
5. Встановити вираз для приховання всіх символів, крім одного (для символів), або для відображення одного слова (для слів).
6. Вимкнути вихідний шар вручну.

Кількість кроків: 6, приблизний час: 20–25 секунд залежно від кількості елементів.

З програмним модулем:

1. Виділити текстовий шар.
2. Клікнути кнопку «Decompose Text» (для символів) або «Decompose Words» (для слів) у панелі розширення.

Кількість кроків: 2, приблизний час: 5–7 секунд.

Модуль «Keyframes» функція копіювання та вставлення графіків анімації без програмного модуля:

1. Виділити кейфрейми у властивості шару.
2. Відкрити редактор графіку, запам’ятати положення графіку кривої для обраних кейфреймів.
3. Перейти до іншої властивості.
4. Вручну налаштувати графік кривої між кейфреймами.

Кількість кроків: 4, приблизний час: 30-35 секунд.

З програмним модулем:

1. Виділити кейфрейми у властивості.
2. Клікнути «Copy Keyframe Graph» у панелі розширення.
3. Виділити кейфреми на який потрібно вставити скопійований графік.
4. Клікнути «Paste Keyframe Graph» у панелі розширення

Кількість кроків: 4, приблизний час: 10-15 секунд. Автоматичне перенесення графіка з адаптацією інтерполяції скорочує процес у 3 рази, економлячи до 20 секунд на операцію.

Описані функції демонструють значну перевагу використання програмного розширення порівняно з ручними методами. Таким чином, власна розробка забезпечує ефективну автоматизацію, мінімізуючи рутинні операції та дозволяючи аніматорам зосередитися на творчих аспектах проєкту. Це робить розширення важливим інструментом для оптимізації робочого процесу в Adobe After Effects.

## 4.4 Висновки

У четвертому розділі описано процес тестування розробленого розширення для Adobe After Effects, що включає функціональне тестування, перевірку продуктивності та сумісності. Проведено аналіз реакції програми на некоректні дії користувача, такі як спроби сортування шарів без їх виділення чи додавання композиції до черги рендерингу без вибору. Тестування продуктивності підтвердило стабільну роботу розширення при обробці великих обсягів даних, зокрема 20–30 шарів, без значних затримок. Перевірка сумісності на версіях After Effects CC 2022 та 2025 показала надійність і універсальність розширення. Практичне застосування модуля продемонструвало значне скорочення часу на виконання рутинних операцій, оскільки базові операції зазвичай становлять близько 15-20% робочого процесу в більшості проєктів Adobe After Effects. Автоматизація цих завдань дозволяє економити до 8–12% загального часу роботи, що підвищує ефективність і дає змогу користувачам зосередитися на творчих аспектах проєкту.

# ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи було розроблено програмний модуль для автоматизації робочих процесів у середовищі Adobe After Effects. Для реалізації проєкту використовувались сучасні технології: ExtendScript, JavaScript, HTML/CSS, CEP-панелі та середовище розробки Visual Studio Code. Реалізація роботи відповідає вимогам методичних вказівок [23].

У процесі виконання кваліфікаційної роботи було проведено глибокий аналіз сучасного стану інструментів для автоматизації процесів у відеовиробництві, зокрема розглянуто популярні розширення, виявлено їхні обмеження та обґрунтовано доцільність створення власного рішення. На основі цього було сформульовано перелік задач, реалізація яких дозволила досягти поставленої мети.

У бакалаврській кваліфікаційній роботі було виконано наступне:

- проведено аналіз існуючих робочих процесів у Adobe After Effects для виявлення операцій, що потребують автоматизації;

- налаштовано API та CEP-панелі для взаємодії з інтерфейсом Adobe After Effects;

- розроблено модулі для автоматизації роботи з шарами, текстом, 3D-об’єктами та ключовими кадрами;

- створено графічний інтерфейс користувача з логічною структурою, вкладками та кнопками швидкого доступу до функцій;

- всі модулі були інтегровані в єдину систему;

- проведено тестування розробленого розширення та експериментальну перевірку його практичного застосування.

Тестування довело працездатність програмного модуля та його відповідність поставленим задачам. Практичне застосування розробленого модуля продемонструвало скорочення часу на виконання базових операцій, які становлять близько 15-20% робочого процесу, що дозволяє економити до 8–12% загального часу роботи. Отже, мету бакалаврської кваліфікаційної роботи досягнуто.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сучасні тенденції у створенні відеоконтенту: автоматизація та цифрові технології [Електронний ресурс]. Режим доступу до ресурсу: https://dizz.in.ua/uk/golovni-tendenczii-videomarketingu-yaki-ne-mozhna-ignoruvati/ (дата звернення: 03.03.2025).

2. Adobe After Effects: Features and Capabilities [Електронний ресурс]. Adobe Official Website. 2025. Режим доступу до ресурсу: https://www.adobe.com/products/aftereffects.html (дата звернення: 03.03.2025).

3. Adobe After Effects – Навчання й підтримка [Електронний ресурс]. Adobe Help Center. 2025. Режим доступу до ресурсу: https://helpx.adobe.com/ua/support/after-effects.html (дата звернення: 03.03.2025).

4. Подунай В.В. Програмний модуль для автоматизації робочих процесів застосунку Adobe After Effects. Вінниця: ВНТУ, 2025. 2с. [Електронний ресурс]. Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/24390.

5. Подунай В.В. Розробка програмного модуля для автоматизації базових операцій застосунку Adobe After Effects. XIII Міжнародна науково-практична конференція з інформаційних систем та технологій Infocom Advanced Solutions 2025. Київ: КПІ, 2025. 3с. [Електронний ресурс] – Режим доступу до ресурсу: https://ist.kpi.ua/uk/konferencziya-infocom/.

6. Оптимізація робочого процесу в After Effects: розуміння важливих функцій [Електронний ресурс]. TutKit. 2024. Режим доступу до ресурсу: https://www.tutkit.com/uk/tekstovi-uroki/20895-optimizacia-robocogoproces u-v-after-effects-rozuminna-vazlivih-funkcij (дата звернення: 10.03.2025).

7. How to Automate Your Workflow in Adobe After Effects with Scripts [Електронний ресурс]. Envato Tuts+. 2024. Режим доступу до ресурсу: https://motionarray.com/plugins/ (дата звернення: 10.03.2025).

8. Відеоконтент для сучасного бізнесу: навіщо потрібен та основні переваги [Електронний ресурс]. Режим доступу до ресурсу:

https://edpit.org/uk/blog-uk/chomu-videokontent-potriben-kozhnomu-biznesu-u-2025-roczi/ (дата звернення: 10.03.2025).

9. Motion Bro – Creative Assets for Motion Designers and Video Editors [Електронний ресурс]. Режим доступу до ресурсу: https://motionbro.com/ (дата звернення: 10.03.2025).

10. Flow [Електронний ресурс]. AEScripts. 2022. Режим доступу до ресурсу: https://aescripts.com/flow/ (дата звернення: 10.03.2025).

11. FX Console [Електронний ресурс]. Video Copilot Blog. 2018. Режим доступу до ресурсу: https://www.videocopilot.net/blog/2018/05/fx-console-updated-to-v1-0-3/ (дата звернення: 10.03.2025).

12. Ease and Wizz [Електронний ресурс]. AEScripts. 2016. Режим доступу до ресурсу: https://aescripts.com/ease-and-wizz/ (дата звернення: 10.03.2025).

13. Getting Started guides and samples for CEP extensions [Електронний ресурс]. Adobe-CEP. 2020. Режим доступу до ресурсу: https://github.com/Adobe-CEP/Getting-Started-guides (дата звернення: 20.03.2025).

14. Fridsma L., Gyncild B. Adobe After Effects CC Classroom in a Book (2023 Release). Adobe Press, 2023. 416 p.

15. RenderQueueItem – After Effects Scripting Guide [Електронний ресурс]. Docs for Adobe. 2024. Режим доступу до ресурсу: https://ae-scripting.docsforadobe.dev/renderqueue/renderqueueitem/ (дата звернення: 20.03.2025).

16. The ExtendScript Toolkit [Електронний ресурс]. Adobe Extendscript Scripting Guide. 2025. Режим доступу до ресурсу: https://extendscript.docsforadobe.dev/tools/extendscript-toolkit.html (дата звернення: 22.03.2025).

17. After Effects Scripting Guide [Електронний ресурс]. Docs for Adobe. 2021. Режим доступу до ресурсу: https://ae-scripting.docsforadobe.dev/ (дата звернення: 23.03.2025).

18. After Effects Javascript API [Електронний ресурс]. Adobe After Effects Wiki, Fandom. 2024. Режим доступу до ресурсу:

https://aftereffects.fandom.com/wiki/After_Effects_Javascript_API (дата звернення: 25.03.2025).

19. Visual Studio Code for Adobe ExtendScript [Електронний ресурс]. Code and Motion. 2019. Режим доступу до ресурсу: https://www.codeandmotion.com/visual-studio-code-for-adobe-extendscript (дата звернення: 25.03.2025).

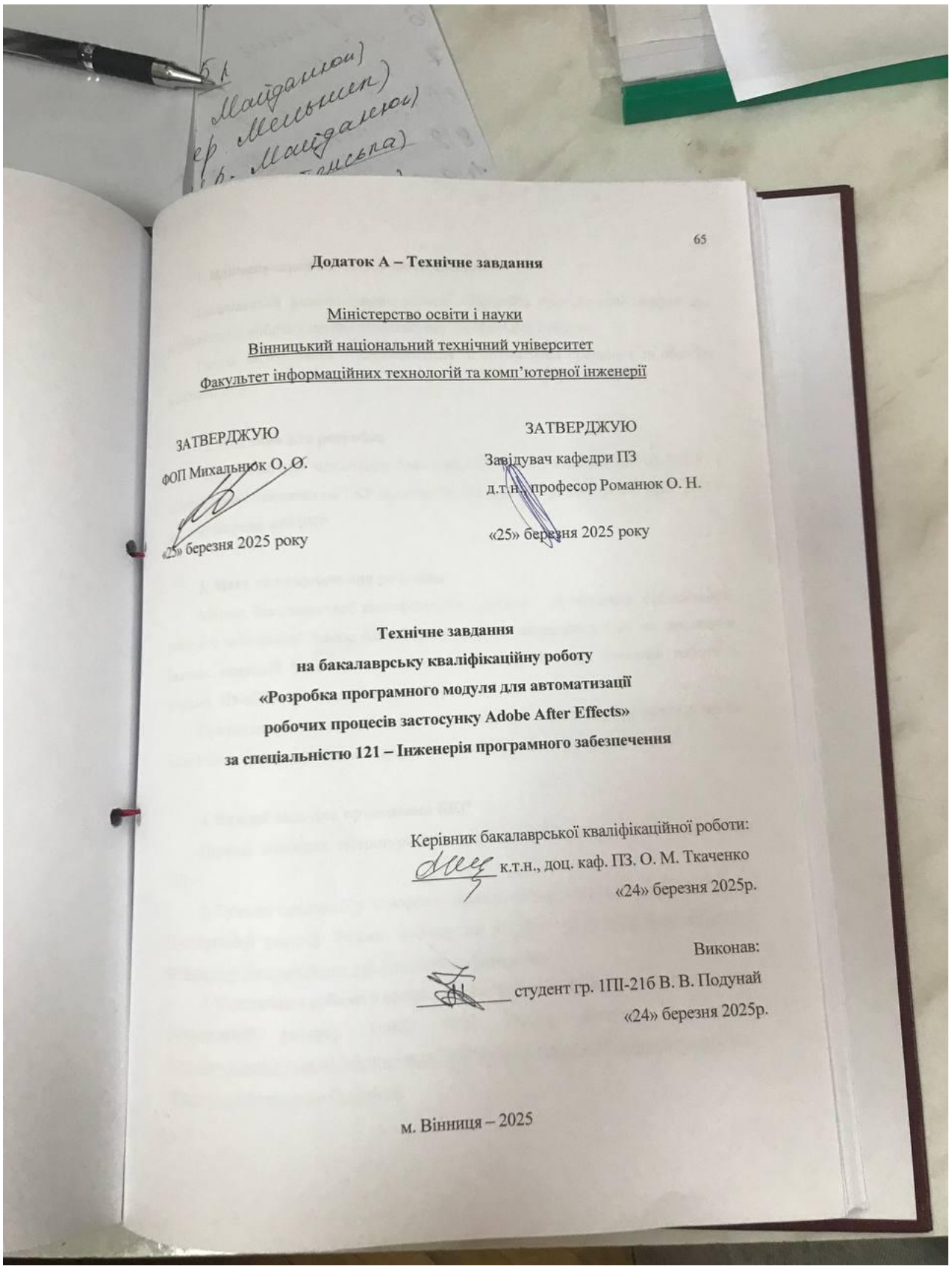
20. UML Activity Diagram Tutorial [Електронний ресурс]. Lucidchart. 2023. Режим доступу до ресурсу: https://www.lucidchart.com/pages/uml-activity-diagram (дата звернення: 28.03.2025).

21. How to Create Flowcharts for Programming [Електронний ресурс]. Zen Flowchart. 2023. Режим доступу до ресурсу: https://www.zenflowchart.com/guides/flowcharts-for-programming (дата звернення: 29.03.2025).

22. What is Functional Testing? Types and Examples [Електронний ресурс]. Guru99. 2024. Режим доступу до ресурсу: https://www.guru99.com/functional-testing.html (дата звернення: 31.03.2025).

23. Романюк О. Н., Чорноволик Г. О., Методичні вказівки до виконання бакалаврських дипломних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення»: методичні – вказівки Вінниця : ВНТУ, 2022. 51 с.

# ДОДАТКИ





**Додаток А – Технічне завдання**

Міністерство освіти і науки

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
ФОП Михальнюк О. О.
«25» березня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«25» березня 2025 року

**Технічне завдання**
**на бакалаврську кваліфікаційну роботу**
**«Розробка програмного модуля для автоматизації**
**робочих процесів застосунку Adobe After Effects»**
**за спеціальністю 121 – Інженерія програмного забезпечення**

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ. О. М. Ткаченко
«24» березня 2025р.

Виконав:
студент гр. 1ПІ-21б В. В. Подунай
«24» березня 2025р.

м. Вінниця – 2025

**1. Найменування та галузь застосування**

Бакалаврська кваліфікаційна робота: «Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects».

Галузь застосування – автоматизація та оптимізація створення та обробки відеоконтенту.

**2. Підстава для розробки**

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ 20 березня 2025р. №97 ректора по ВНТУ про закріплення тем БКР.

**3. Мета та призначення розробки**

Метою бакалаврської кваліфікаційної роботи є підвищення ефективності роботи у застосунку Adobe After Effects завдяки зменшенню часу на виконання базових операцій за рахунок автоматизації та оптимізації процесів роботи з шарами, 3D-об’єктами, текстом та кейфреймами.

Призначення роботи – автоматизація робочих процесів у застосунку Adobe After Effects для полегшення створення анімацій і відеоконтенту.

**4. Вихідні дані для проведення БКР**

Перелік основних літературних джерел, на основі яких буде виконуватись БКР.

1. Сучасні тенденції у створенні відеоконтенту: автоматизація та цифрові [Електронний ресурс]. Режим доступу до ресурсу: https://dizz.in.ua/uk/golovni-tendenczii-videomarketingu-yaki-ne-mozhna-ignoruvati/

2. Оптимізація робочого процесу в After Effects: розуміння важливих функцій [Електронний ресурс]. TutKit. 2024. Режим доступу до ресурсу: https://www.tutkit.com/uk/tekstovi-uroki/20895-optimizacia-robocogoprocesu-vafter-effects-rozuminna-vazlivih-funkcij

3. Adobe After Effects: Features and Capabilities [Електронний ресурс]. Adobe Official Website. 2025. Режим доступу до ресурсу: https://www.adobe.com/products/aftereffects.html

4. After Effects Scripting Guide [Електронний ресурс]. Docs for Adobe. 2021. Режим доступу до ресурсу: https://ae-scripting.docsforadobe.dev/

**5. Технічні вимоги**

Методи для створення коригуючих та null шарів, метод створення 3Д-кубів з можливістю розбиття на частини, метод розбиття тексту на слова та символи, метод для додання обраних елементів у прекомпозицію, метод для реверсу кейфреймів, метод для копіювання графіку анімації, метод для зміни якірної точки обраних шарів.

**6. Конструктивні вимоги**

Графічна та текстова документація повинна відповідати діючим стандартам України.

**7. Перелік технічної документації, що пред'являється по закінченню робіт**

1. Пояснювальна записка до БКР.
2. Технічне завдання.
3. Лістинги програми.

**8. Вимоги до рівня уніфікації та стандартизації**

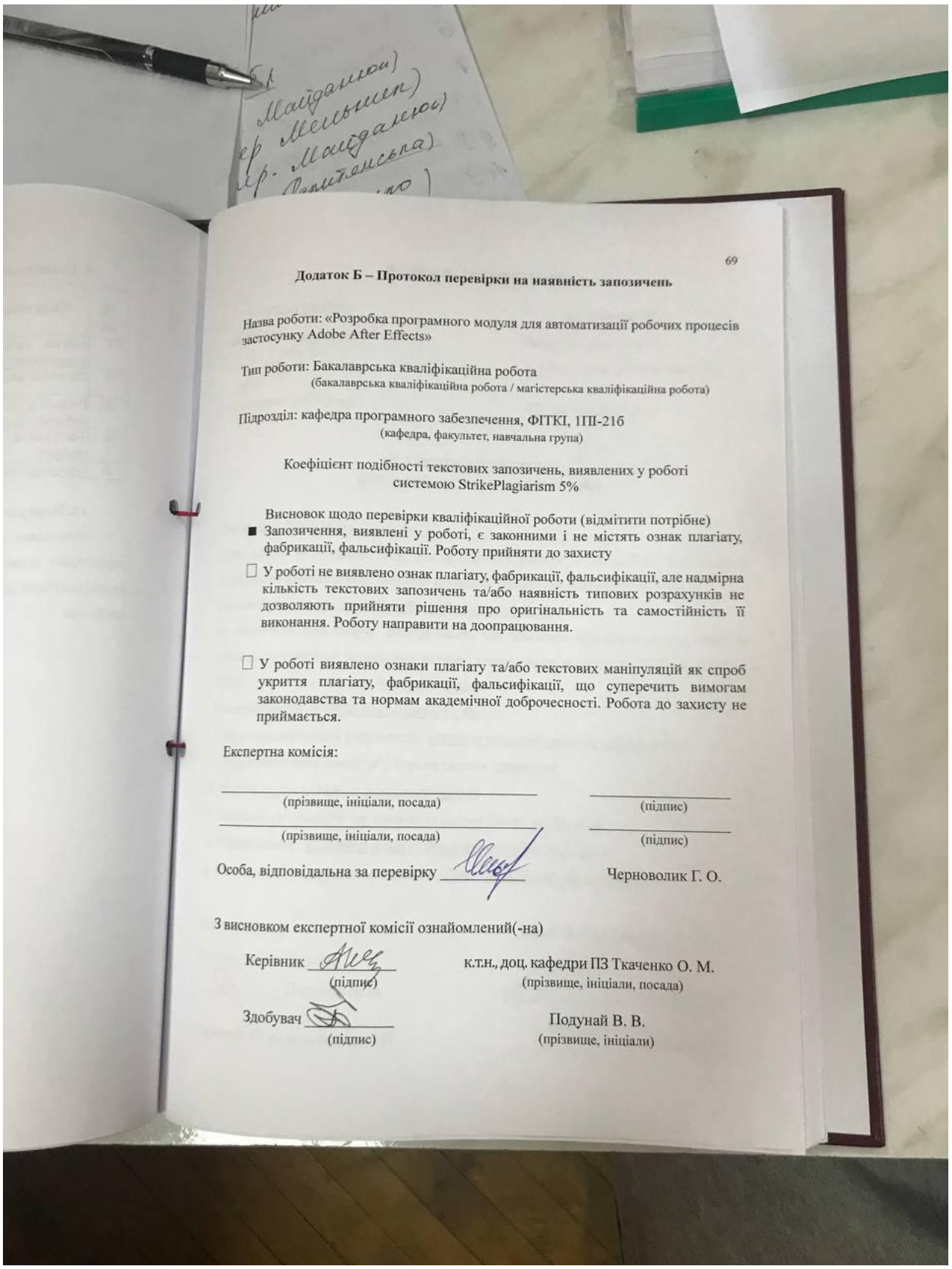
При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ

## 9. Стадії та етапи розробки

| № з/п | Назва етапів бакалаврської кваліфікаційної роботи | Строк виконання етапів роботи |
|---|---|---|
| 1. | Аналіз розвитку технологій у сфері створення відеоконтенту та постановка задач дослідження | 25.03.25 – 31.03.25 |
| 2. | Розробка моделі та методів програмного модуля | 01.04.25 - 18.04.25 |
| 3. | Розробка програмних модулів для автоматизації робочих процесів застосунку Adobe After Effects | 19.04.25 - 06.05.25 |
| 4. | Тестування та практичне застосування програмного модуля | 07.05.25 - 24.05.25 |
| 5. | Оформлення матеріалів до захисту БКР | 25.05.25 - 30.05.25 |

## 10. Порядок контролю та прийняття

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.





**Додаток Б – Протокол перевірки на наявність запозичень**

Назва роботи: «Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects»

Тип роботи: Бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ: кафедра програмного забезпечення, ФІТКІ, 1ПП-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 5%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

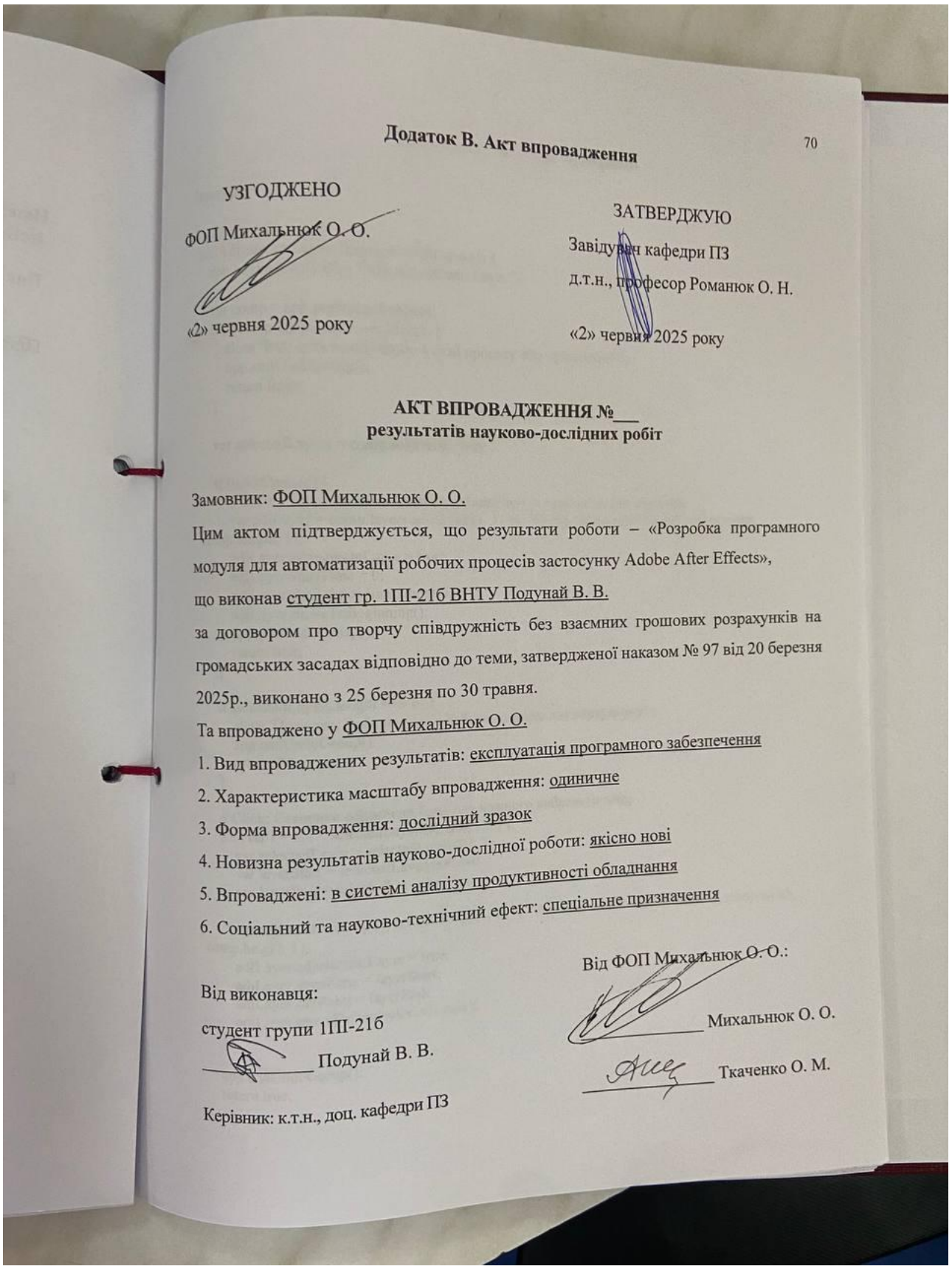
_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ к.т.н., доц. кафедри ПЗ Ткаченко О. М.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Подунай В. В.
(підпис) (прізвище, ініціали)



## Додаток В. Акт впровадження



УЗГОДЖЕНО

ФОП Михальнюк О. О.

«2» червня 2025 року

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

д.т.н., професор Романюк О. Н.

«2» червня 2025 року

**АКТ ВПРОВАДЖЕННЯ №___**
**результатів науково-дослідних робіт**

Замовник: ФОП Михальнюк О. О.

Цим актом підтверджується, що результати роботи – «Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects»,

що виконав студент гр. 1ПІ-21б ВНТУ Подунай В. В.

за договором про творчу співдружність без взаємних грошових розрахунків на громадських засадах відповідно до теми, затвердженої наказом № 97 від 20 березня 2025р., виконано з 25 березня по 30 травня.

Та впроваджено у ФОП Михальнюк О. О.

1. Вид впроваджених результатів: експлуатація програмного забезпечення
2. Характеристика масштабу впровадження: одиничне
3. Форма впровадження: дослідний зразок
4. Новизна результатів науково-дослідної роботи: якісно нові
5. Впроваджені: в системі аналізу продуктивності обладнання
6. Соціальний та науково-технічний ефект: спеціальне призначення

Від виконавця:

студент групи 1ПІ-21б

_____ Подунай В. В.

Керівник: к.т.н., доц. кафедри ПЗ

Від ФОП Михальнюк О. О.:

_____ Михальнюк О. О.

_____ Ткаченко О. М.

# Г. Лістинг програми

***layers.jsx***

```
$._layers = {
 addAdjustmentLayer: function(isAltPressed) {
    app.beginUndoGroup("Add Adjustment Layer");

    var comp = app.project.activeItem;
    if (!(comp instanceof CompItem)) {
        alert("Виберіть композицію в окні проекту або таймлайні!");
        app.endUndoGroup();
        return false;
    }

    var selectedLayers = comp.selectedLayers;

    if (isAltPressed) {
        // Alt + Click: Створити один adjustment layer з тривалістю композиції
        var adjLayer = comp.layers.addSolid([1, 1, 1], "Comp Adjustment Layer", comp.width,
comp.height, 1);
        adjLayer.adjustmentLayer = true;
        adjLayer.startTime = 0;
        adjLayer.outPoint = comp.duration;
        adjLayer.moveToBeginning();
        app.endUndoGroup();
        return true;
    }

    if (selectedLayers.length === 0) {
        alert("Помилка: Виберіть два або більше шарів для сортування");
        app.endUndoGroup();
        return false;
    }

    // Click: Створити adjustment layer для кожного вибраного шару
    for (var i = 0; i < selectedLayers.length; i++) {
        var selectedLayer = selectedLayers[i];
        var layerStart = selectedLayer.inPoint;
        var layerEnd = selectedLayer.outPoint;

        var adjLayer = comp.layers.addSolid([1, 1, 1], "Adj for " + selectedLayer.name, comp.width,
comp.height, 1);
        adjLayer.adjustmentLayer = true;
        adjLayer.startTime = layerStart;
        adjLayer.outPoint = layerEnd;
        adjLayer.moveBefore(selectedLayer);
    }

    app.endUndoGroup();
    return true;
},
```

```
addNullObject: function(isAltPressed, isShiftPressed) {
  app.beginUndoGroup("Add Null Object");

  var comp = app.project.activeItem;
  if (!(comp instanceof CompItem)) {
     alert("Select a composition in the project window or timeline!");
     app.endUndoGroup();
     return false;
  }

  var selectedLayers = comp.selectedLayers;

  if (isShiftPressed) {
     // Shift + Click:
     var nullLayer = comp.layers.addNull();
     nullLayer.name = "Null Controller";
     nullLayer.startTime = 0;
     nullLayer.outPoint = comp.duration;
     app.endUndoGroup();
     return true;
  }

  if (selectedLayers.length === 0) {
     alert("Select at least one layer!");
     app.endUndoGroup();
     return false;
  }

  if (isAltPressed && selectedLayers.length > 1) {
     // Alt + Click:
     var maxDuration = 0;
     var earliestStart = comp.duration;
     var layerNames = [];
     var sortedLayers = selectedLayers.slice().sort(function(a, b) {
        return b.index - a.index;
     });

     for (var i = 0; i < sortedLayers.length; i++) {
        var layer = sortedLayers[i];
        var duration = layer.outPoint - layer.inPoint;
        maxDuration = Math.max(maxDuration, duration);
        earliestStart = Math.min(earliestStart, layer.inPoint);
        layerNames.push(layer.name);
     }

     var nullLayer = comp.layers.addNull();
     nullLayer.name = "Null for " + layerNames.join(", ");
     nullLayer.startTime = earliestStart;
     nullLayer.outPoint = earliestStart + maxDuration;
     nullLayer.moveToBeginning();
```

```
            for (var i = 0; i < selectedLayers.length; i++) {
                selectedLayers[i].parent = nullLayer;
            }

            app.endUndoGroup();
            return true;
        }

        // Click:
        for (var i = 0; i < selectedLayers.length; i++) {
            var selectedLayer = selectedLayers[i];
            var layerStart = selectedLayer.inPoint;
            var layerEnd = selectedLayer.outPoint;

            var nullLayer = comp.layers.addNull();
            nullLayer.name = "Null for " + selectedLayer.name;
            nullLayer.startTime = layerStart;
            nullLayer.outPoint = layerEnd;
            nullLayer.moveBefore(selectedLayer);
            selectedLayer.parent = nullLayer;
        }

        app.endUndoGroup();
        return true;
    },

preCompose: function(compName) {
    try {
        app.beginUndoGroup("Pre-compose Selected Layers");
        var comp = app.project.activeItem;
        if (!comp || !(comp instanceof CompItem)) {
            alert("Виберіть композицію!");
            app.endUndoGroup();
            return false;
        }
        var selectedLayers = comp.selectedLayers;
        if (selectedLayers.length === 0) {
            alert("Виберіть хоча б один шар!");
            app.endUndoGroup();
            return false;
        }
        var earliestStart = comp.duration, latestEnd = 0;
        for (var i = 0; i < selectedLayers.length; i++) {
            earliestStart = Math.min(earliestStart, selectedLayers[i].inPoint);
            latestEnd = Math.max(latestEnd, selectedLayers[i].outPoint);
        }
        var duration = latestEnd - earliestStart;
        if (!compName) {
            var date = new Date();
            compName = "Pre-comp_" + date.getFullYear() + "_" + (date.getMonth() + 1) + "_" +
date.getDate();
        }
```

```
        var precomp = app.project.items.addComp(compName, comp.width, comp.height,
comp.pixelAspect, duration, comp.frameRate);

        for (var i = 0; i < selectedLayers.length; i++) {
            var layer = selectedLayers[i];
            var newLayer = precomp.layers.add(layer.source);
            newLayer.name = layer.name;
            newLayer.startTime = layer.inPoint - earliestStart;
            newLayer.inPoint = layer.inPoint - earliestStart;
            newLayer.outPoint = layer.outPoint - earliestStart;
            $._layers.copyLayerProperties(layer, newLayer);
            layer.remove();
        }
        var precompLayer = comp.layers.add(precomp);
        precompLayer.startTime = earliestStart;
        precompLayer.outPoint = latestEnd;

        app.endUndoGroup();
        return true;
    } catch (e) {
        alert("Помилка: " + e.toString());
        app.endUndoGroup();
        return false;
    }
},

uncompose: function() {
    app.beginUndoGroup("Uncompose Selected Precomps");

    try {
        var comp = app.project.activeItem;
        if (!(comp instanceof CompItem)) {
            alert("Виберіть композицію!");
            app.endUndoGroup();
            return false;
        }

        var selectedLayers = comp.selectedLayers;
        if (selectedLayers.length === 0) {
            alert("Виберіть хоча б одну пре-композицію для розпаковки!");
            app.endUndoGroup();
            return false;
        }

        var precompCount = 0;
        for (var i = 0; i < selectedLayers.length; i++) {
            if (selectedLayers[i].source instanceof CompItem) {
                precompCount++;
            }
        }

        if (precompCount === 0) {
```

```
        alert("Жодної пре-композиції не вибрано!");
        app.endUndoGroup();
        return false;
    }

    // Process each selected precomp in reverse order to avoid index issues
    for (var i = selectedLayers.length - 1; i >= 0; i--) {
        var precompLayer = selectedLayers[i];
        if (precompLayer.source instanceof CompItem) {
            var precomp = precompLayer.source;
            var precompLayers = precomp.layers;
            if (precompLayers.length === 0) {
                alert("Прекомпозиція порожня! Пропускаю: " + precompLayer.name);
                continue;
            }

            var precompStartTime = precompLayer.startTime;
            var newLayers = [];
            var parentingInfo = [];

            // Copy layers top to bottom from precomp
            for (var j = 1; j <= precompLayers.length; j++) {
                var sourceLayer = precompLayers[j];
                var newLayer;

                if (sourceLayer.adjustmentLayer) {
                    newLayer = comp.layers.addSolid([1, 1, 1], sourceLayer.name, comp.width,
comp.height, comp.pixelAspect);
                    newLayer.adjustmentLayer = true;
                } else if (sourceLayer instanceof TextLayer) {
                    var textProp = sourceLayer.text.sourceText;
                    newLayer = comp.layers.addText(textProp.value.toString());
                } else if (sourceLayer instanceof ShapeLayer) {
                    newLayer = comp.layers.addShape();
                } else if (sourceLayer instanceof CameraLayer) {
                    var cameraPos = sourceLayer.transform.position.value;
                    newLayer = comp.layers.addCamera(sourceLayer.name, cameraPos);
                } else if (sourceLayer instanceof LightLayer) {
                    newLayer = comp.layers.addLight(sourceLayer.name,
sourceLayer.transform.position.value);
                    newLayer.lightType = sourceLayer.lightType;
                } else if (sourceLayer.nullLayer) {
                    newLayer = comp.layers.addNull();
                    newLayer.name = sourceLayer.name;
                } else if (sourceLayer.source instanceof FootageItem || sourceLayer.source instanceof
CompItem) {
                    newLayer = comp.layers.add(sourceLayer.source);
                    newLayer.name = sourceLayer.name;
                } else {
                    continue;
                }
```

```
                // Copy all properties
                $._layers.copyLayerProperties(sourceLayer, newLayer);

                // Set timing
                newLayer.startTime = precompStartTime + sourceLayer.startTime;
                newLayer.inPoint = precompStartTime + sourceLayer.inPoint;
                newLayer.outPoint = precompStartTime + sourceLayer.outPoint;

                // Store parenting
                if (sourceLayer.parent) {
                    parentingInfo.push({ layer: newLayer, parentIndex: sourceLayer.parent.index });
                }

                newLayers.push(newLayer);
            }

            // Restore parenting within the new layers
            for (var j = 0; j < parentingInfo.length; j++) {
                var info = parentingInfo[j];
                for (var k = 0; k < newLayers.length; k++) {
                    if (newLayers[k].index === comp.layers.length - precompLayers.length +
info.parentIndex) {
                        info.layer.parent = newLayers[k];
                        break;
                    }
                }
            }

            // Move new layers above the precomp layer
            for (var j = 0; j < newLayers.length; j++) {
                newLayers[j].moveBefore(precompLayer);
            }

            // Instead of removing, hide the precomp layer
            precompLayer.enabled = false;
        }
    }

    alert("Пре-композиції розпаковано!");
    app.endUndoGroup();
    return true;
  } catch (e) {
    alert("Помилка: " + e.toString());
    app.endUndoGroup();
    return false;
  }
},

copyLayerProperties: function(sourceLayer, targetLayer) {
  // Copy transform
  var props = ["position", "scale", "rotation", "opacity", "anchorPoint"];
  for (var i = 0; i < props.length; i++) {
```

```
        var sourceProp = sourceLayer.transform[props[i]];
        var targetProp = targetLayer.transform[props[i]];
        if (sourceProp.numKeys > 0) {
            for (var j = 1; j <= sourceProp.numKeys; j++) {
                targetProp.setValueAtTime(sourceProp.keyTime(j), sourceProp.keyValue(j));
                targetProp.setInterpolationTypeAtKey(j, sourceProp.keyInInterpolationType(j),
sourceProp.keyOutInterpolationType(j));
                if (sourceProp.keyInInterpolationType(j) === KeyframeInterpolationType.BEZIER) {
                    targetProp.setTemporalEaseAtKey(j, sourceProp.keyInTemporalEase(j),
sourceProp.keyOutTemporalEase(j));
                }
            }
        } else {
            targetProp.setValue(sourceProp.value);
        }
    }

    // Copy effects
    if (sourceLayer.effect) {
        for (var i = 1; i <= sourceLayer.effect.numProperties; i++) {
            var sourceEffect = sourceLayer.effect(i);
            var targetEffect = targetLayer.Effects.addProperty(sourceEffect.matchName);
            for (var j = 1; j <= sourceEffect.numProperties; j++) {
                var sourceProp = sourceEffect.property(j);
                var targetProp = targetEffect.property(j);
                if (sourceProp.numKeys > 0) {
                    for (var k = 1; k <= sourceProp.numKeys; k++) {
                        targetProp.setValueAtTime(sourceProp.keyTime(k), sourceProp.keyValue(k));
                        targetProp.setInterpolationTypeAtKey(k, sourceProp.keyInInterpolationType(k),
sourceProp.keyOutInterpolationType(k));
                    }
                } else if (sourceProp.value !== undefined) {
                    targetProp.setValue(sourceProp.value);
                }
            }
        }
    }

    // Copy masks
    if (sourceLayer.mask) {
        for (var i = 1; i <= sourceLayer.mask.numProperties; i++) {
            var sourceMask = sourceLayer.mask(i);
            var targetMask = targetLayer.Masks.addProperty("Mask");
            targetMask.name = sourceMask.name;
            var maskPath = sourceMask.maskPath;
            if (maskPath.numKeys > 0) {
                for (var j = 1; j <= maskPath.numKeys; j++) {
                    targetMask.maskPath.setValueAtTime(maskPath.keyTime(j), maskPath.keyValue(j));
                }
            } else {
                targetMask.maskPath.setValue(maskPath.value);
            }
```

```
        }
    }

    // Copy shape layer contents
    if (sourceLayer instanceof ShapeLayer) {
        var sourceContents = sourceLayer.property("Contents");
        var targetContents = targetLayer.property("Contents");
        for (var i = 1; i <= sourceContents.numProperties; i++) {
            var sourceGroup = sourceContents.property(i);
            var targetGroup = targetContents.addProperty(sourceGroup.matchName);
            targetGroup.name = sourceGroup.name;
            for (var j = 1; j <= sourceGroup.numProperties; j++) {
                var sourceProp = sourceGroup.property(j);
                var targetProp = targetGroup.property(j);
                if (sourceProp.numKeys > 0) {
                    for (var k = 1; k <= sourceProp.numKeys; k++) {
                        targetProp.setValueAtTime(sourceProp.keyTime(k), sourceProp.keyValue(k));
                        targetProp.setInterpolationTypeAtKey(k, sourceProp.keyInInterpolationType(k),
sourceProp.keyOutInterpolationType(k));
                    }
                } else if (sourceProp.value !== undefined) {
                    targetProp.setValue(sourceProp.value);
                }
            }
        }
    }

    // Copy layer styles
    if (sourceLayer.layerStyle) {
        for (var i = 1; i <= sourceLayer.layerStyle.numProperties; i++) {
            var sourceStyle = sourceLayer.layerStyle.property(i);
            var targetStyle = targetLayer.layerStyle.property(sourceStyle.matchName);
            if (sourceStyle.numKeys > 0) {
                for (var j = 1; j <= sourceStyle.numKeys; j++) {
                    targetStyle.setValueAtTime(sourceStyle.keyTime(j), sourceStyle.keyValue(j));
                }
            } else if (sourceStyle.value !== undefined) {
                targetStyle.setValue(sourceStyle.value);
            }
        }
    }
},
    arrangeToPlayhead: function(isAltPressed, isShiftPressed) {
    try {
        app.beginUndoGroup("Arrange to Playhead or Comp Start");

        var comp = app.project.activeItem;
        if (!(comp instanceof CompItem)) {
            alert("Виберіть композицію!");
            app.endUndoGroup();
            return false;
        }
```

```
var selectedLayers = comp.selectedLayers;
if (selectedLayers.length === 0) {
    alert("Виберіть хоча б один шар!");
    app.endUndoGroup();
    return false;
}

var playheadTime = comp.time;
var targetTime;

if (isAltPressed) {
    targetTime = 0; // Alt + Click: до початку композиції
} else if (isShiftPressed) {
    var layers = [];
    for (var i = 0; i < selectedLayers.length; i++) {
        layers.push(selectedLayers[i]);
    }
    layers.sort(function(a, b) {
        return a.index - b.index;
    });
    targetTime = layers[0].inPoint; // Shift + Click: до in-point першого шару
} else {
    targetTime = playheadTime; // Click: до плейхеда
}

var keyframesSelected = false;
for (var i = 0; i < selectedLayers.length && !keyframesSelected; i++) {
    var layer = selectedLayers[i];
    var props = layer.selectedProperties;
    for (var j = 0; j < props.length; j++) {
        if (props[j].selectedKeys.length > 0) {
            keyframesSelected = true;
            break;
        }
    }
}

if (keyframesSelected && !isAltPressed && !isShiftPressed) {
    for (var i = 0; i < selectedLayers.length; i++) {
        var layer = selectedLayers[i];
        var props = layer.selectedProperties;
        for (var j = 0; j < props.length; j++) {
            var prop = props[j];
            var selectedKeys = prop.selectedKeys;
            if (selectedKeys.length > 0) {
                var firstKeyTime = prop.keyTime(selectedKeys[0]);
                var offset = targetTime - firstKeyTime;
                for (var k = 0; k < selectedKeys.length; k++) {
                    var keyIndex = selectedKeys[k];
                    var keyTime = prop.keyTime(keyIndex);
                    prop.setValueAtTime(keyTime + offset, prop.keyValue(keyIndex));
```

```
                    prop.removeKey(keyIndex);
                }
            }
        }
    }
} else {
    for (var i = 0; i < selectedLayers.length; i++) {
        var layer = selectedLayers[i];
        var duration = layer.outPoint - layer.inPoint;
        layer.startTime = targetTime;
        layer.inPoint = targetTime;
        layer.outPoint = targetTime + duration;
    }
}

app.endUndoGroup();
return true;
} catch (e) {
    alert("Помилка: " + e.toString());
    app.endUndoGroup();
    return false;
}
},

sequenceLayers: function(direction, offset, step) {
  try {
    app.beginUndoGroup("Sequence Layers");

    var comp = app.project.activeItem;
    if (!(comp instanceof CompItem)) {
        alert("Выберите композицию!");
        app.endUndoGroup();
        return false;
    }

    var selectedLayers = comp.selectedLayers;
    if (selectedLayers.length < 2) {
        alert("select more than 1 layer!");
        app.endUndoGroup();
        return false;
    }

    // Преобразуем offset в кадры и преобразуем в секунды
    offset = parseInt(offset) || 0;
    step = parseInt(step) || 1;

    // Сортируем слои по их текущему индексу
    var layers = [];
    for (var i = 0; i < selectedLayers.length; i++) {
        layers.push(selectedLayers[i]);
    }
    layers.sort(function(a, b) {
```

```
            return a.index - b.index;
        });

        // Если направление "from bottom", переворачиваем массив
        if (direction === "bottom") {
            layers.reverse();
        }

        // Смещение накапливается относительно текущих позиций
        var offsetCounter = 0;

        for (var i = 0; i < layers.length; i++) {
            var layer = layers[i];

            var currentStartTime = layer.startTime;

            var frameOffset = offsetCounter * offset;
            var newStartTime = currentStartTime + (frameOffset / comp.frameRate); // Добавляем
смещение

            layer.startTime = newStartTime;

            if ((i + 1) % step === 0) {
                offsetCounter++;
            }
        }

        app.endUndoGroup();

        return true;
    } catch (e) {
        alert("Ошибка: " + e.toString());
        app.endUndoGroup();
        return false;
    }
},
addToRenderQueue: function(isAltPressed) {
    app.beginUndoGroup("Add to Render Queue");
    var proj = app.project;
    if (!proj) {
        alert("Відкрийте проект!");
        app.endUndoGroup();
        return false;
    }
    var comp = app.project.activeItem;
    if (!(comp instanceof CompItem)) {
        alert("Відкрийте композицію у таймлайні!");
        app.endUndoGroup();
        return false;
    }
    try {
        var renderQueue = proj.renderQueue;
```

```
        var renderItem = renderQueue.items.add(comp);
        var startTime = comp.workAreaStart;
        var duration = comp.workAreaDuration;
        if (duration <= 0) {
            startTime = 0;
            duration = comp.duration;
        }
        renderItem.timeSpanStart = startTime;
        renderItem.timeSpanDuration = duration;
        var outputModule = renderItem.outputModule(1);
        var template = isAltPressed ? "[h.264] [template]" : "[mov] [template]";
        var outputPathBase = isAltPressed
            ? "D:\\[a] [scenes]\\[cutter] [ae render]\\[h.264]\\"
            : "D:\\[a] [scenes]\\[cutter] [ae render]\\[mov]\\";
        var fileExtension = isAltPressed ? ".mp4" : ".mov";
        var baseFileName = comp.name;
        var fileIndex = 0;
        for (var i = 1; i <= renderQueue.numItems; i++) {
            var item = renderQueue.item(i);
            if (item.comp === comp && item !== renderItem) {
                var itemPath = item.outputModule(1).file.fsName;
                var pattern = baseFileName.replace(/([.*+?^${}()|\[\]\/\\])/g, "\\$1") + "(?: \\[(\\d+)\\])?" +
fileExtension.replace(".", "\\.");
                var match = itemPath.match(new RegExp(pattern));
                if (match) {
                    var index = match[1] ? parseInt(match[1]) : 0;
                    fileIndex = Math.max(fileIndex, index + 1);
                }
            }
        }
        var outputPath = fileIndex === 0
            ? outputPathBase + baseFileName + fileExtension
            : outputPathBase + baseFileName + " [" + fileIndex + "]" + fileExtension;
        try {
            outputModule.applyTemplate(template);
        } catch (e) {
            alert("Шаблон '" + template + "' не знайдено! Застосовано стандартний шаблон.");
            outputModule.applyTemplate("Lossless");
        }
        outputModule.file = new File(outputPath);
        if (app.activeViewer) {
            app.activeViewer.setActive();
        }

        app.endUndoGroup();
        return true;
    } catch (e) {
        alert("Помилка при додаванні в чергу рендера: " + e.message);
        app.endUndoGroup();
        return false;
    }
},
```

```
    returnFocusToPanel: function() {
        // Пустая функция для совместимости
    }
};

if (! $._ext) {
    $._ext = {};
}

$._ext.addAdjustmentLayer = function(isAltPressed) {
    return $._layers.addAdjustmentLayer(isAltPressed) ? "true" : "false";
};

$._ext.addNullObject = function(isAltPressed, isShiftPressed) {
    return $._layers.addNullObject(isAltPressed, isShiftPressed) ? "true" : "false";
};

$._ext.createCameraWithNulls = function(isAltPressed) {
    return $._layers.createCameraWithNulls(isAltPressed) ? "true" : "false";
};

$._ext.uncompose = function() {
    return $._layers.uncompose() ? "true" : "false";
};

$._ext.addToRenderQueue = function(isAltPressed) {
    return $._layers.addToRenderQueue(isAltPressed) ? "true" : "false";
};
$._ext.arrangeToPlayhead = function(isAltPressed, isShiftPressed) {
    return $._layers.arrangeToPlayhead(isAltPressed, isShiftPressed) ? "true" : "false";
};

$._ext.sequenceLayers = function(direction, offset, step) {
    return $._layers.sequenceLayers(direction, offset, step) ? "true" : "false";
};

$._ext.preCompose = function(compName) {
    return $._layers.preCompose(compName) ? "true" : "false";
};

$._ext.returnFocusToPanel = function() {
    $._layers.returnFocusToPanel();
};
```

### *text.jsx*

```
$._text = {
        setAnchorPointGrid: function(position) {
            try {
                app.beginUndoGroup("Set Anchor Point Grid for Multiple Shape/Text Layers");
```

```
if (!app.project) {
    alert("No project open!");
    app.endUndoGroup();
    return false;
}

var comp = app.project.activeItem;
if (!comp || !(comp instanceof CompItem)) {
    alert("Please select a composition in the project window or timeline!");
    app.endUndoGroup();
    return false;
}

var selectedLayers = comp.selectedLayers;
if (selectedLayers.length === 0) {
    alert("Please select at least one Shape or Text layer!");
    app.endUndoGroup();
    return false;
}

var curTime = comp.time;
var validLayersFound = false;

for (var i = 0; i < selectedLayers.length; i++) {
    var layer = selectedLayers[i];
    if (!(layer instanceof ShapeLayer || layer instanceof TextLayer)) {
        continue;
    }
    validLayersFound = true;

    var currentAnchor = layer.anchorPoint.value;
    var currentPosition = layer.position.value;
    var scale = layer.scale.value;
    var scaleX = scale[0] / 100;
    var scaleY = scale[1] / 100;

    var sourceRect = layer.sourceRectAtTime(curTime, false);
    var w = sourceRect.width;
    var h = sourceRect.height;
    var l = sourceRect.left;
    var t = sourceRect.top;

    var newAnchorX, newAnchorY;
    switch (position) {
        case 0: newAnchorX = l; newAnchorY = t; break;
        case 1: newAnchorX = l + (w / 2); newAnchorY = t; break;
        case 2: newAnchorX = l + w; newAnchorY = t; break;
        case 3: newAnchorX = l; newAnchorY = t + (h / 2); break;
        case 4: newAnchorX = l + (w / 2); newAnchorY = t + (h / 2); break;
        case 5: newAnchorX = l + w; newAnchorY = t + (h / 2); break;
        case 6: newAnchorX = l; newAnchorY = t + h; break;
        case 7: newAnchorX = l + (w / 2); newAnchorY = t + h; break;
```

```
                case 8: newAnchorX = l + w; newAnchorY = t + h; break;
                default:
                    continue;
            }

            var layerWidth = layer.width;
            var layerHeight = layer.height;
            newAnchorX = Math.min(newAnchorX, layerWidth);
            newAnchorY = Math.min(newAnchorY, layerHeight);

            var xDelta = newAnchorX - currentAnchor[0];
            var yDelta = newAnchorY - currentAnchor[1];
            var xAdd = xDelta * scaleX;
            var yAdd = yDelta * scaleY;

            layer.anchorPoint.setValue([newAnchorX, newAnchorY]);
            layer.position.setValue([currentPosition[0] + xAdd, currentPosition[1] + yAdd]);
        }

        if (!validLayersFound) {
            alert("No valid Shape or Text layers selected!");
            app.endUndoGroup();
            return false;
        }

        app.endUndoGroup();
        return true;
    } catch (e) {
        alert("Error setting anchor point: " + e.toString());
        app.endUndoGroup();
        return false;
    }
},

decompText: function() {
    app.beginUndoGroup("Decompose Text - Characters");

    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Выберите композицию в окне проекта или таймлайне!");
        app.endUndoGroup();
        return false;
    }

    var selectedLayers = comp.selectedLayers;
    if (selectedLayers.length === 0) {
        alert("Выберите хотя бы один текстовый слой!");
        app.endUndoGroup();
        return false;
    }

    var validLayersFound = false;
```

```
        for (var i = 0; i < selectedLayers.length; i++) {
            var textLayer = selectedLayers[i];
            if (!(textLayer instanceof TextLayer)) {
                continue;
            }

            validLayersFound = true;
            var textString = textLayer.sourceText.value.toString();

            if (!textString || textString.length === 0) {
                alert("Текстовый слой '" + textLayer.name + "' пуст!");
                continue;
            }

            if (textLayer.property("ADBE Text Properties").property("ADBE Text
Document").value.boxText) {
                alert("Paragraph text is not supported for layer '" + textLayer.name + "'.");
                continue;
            }

            var numSkips = 0;
            for (var charId = 0; charId < textString.length; charId++) {
                var character = textString.charAt(charId);

                if (character.match(/\s+/)) {
                    numSkips++;
                    continue;
                }

                var newLayer = textLayer.duplicate();
                newLayer.name = "Letter_" + character + "_" + (charId + 1 - numSkips);

                var animator = newLayer.property("ADBE Text Properties").property("ADBE Text
Animators").addProperty("ADBE Text Animator");
                var opacityProp = animator.property("ADBE Text Animator
Properties").addProperty("ADBE Text Opacity");
                var expressionSelector = animator.property("ADBE Text Selectors").addProperty("ADBE
Text Expressible Selector");
                opacityProp.setValue(0);

                expressionSelector.property("ADBE Text Expressible Amount").expression =
"selectorValue * (textIndex != " + (charId + 1 - numSkips) + ");";
                var basedOnProp = expressionSelector.property("ADBE Text Range Type2");
                basedOnProp.setValue(2);
            }

            textLayer.enabled = false;
        }

        if (!validLayersFound) {
            alert("Не выбрано ни одного текстового слоя!");
```

```
            app.endUndoGroup();
            return false;
        }

        app.endUndoGroup();
        return true;
    },

    decompWords: function() {
        app.beginUndoGroup("Decompose Text - Words");

        var comp = app.project.activeItem;
        if (!comp || !(comp instanceof CompItem)) {
            alert("Выберите композицию в окне проекта или таймлайне!");
            app.endUndoGroup();
            return false;
        }

        var selectedLayers = comp.selectedLayers;
        if (selectedLayers.length === 0) {
            alert("Выберите хотя бы один текстовый слой!");
            app.endUndoGroup();
            return false;
        }

        var validLayersFound = false;

        for (var i = 0; i < selectedLayers.length; i++) {
            var textLayer = selectedLayers[i];
            if (!(textLayer instanceof TextLayer)) {
                continue;
            }

            validLayersFound = true;
            var textString = textLayer.sourceText.value.toString();

            if (!textString || textString.length === 0) {
                alert("Текстовый слой '" + textLayer.name + "' пуст!");
                continue;
            }

            if (textLayer.property("ADBE Text Properties").property("ADBE Text
Document").value.boxText) {
                alert("Paragraph text is not supported for layer '" + textLayer.name + "'.");
                continue;
            }

            var words = textString.split(/\s+/);
            for (var wordId = 0; wordId < words.length; wordId++) {
                var word = words[wordId];

                if (!word) continue;
```

```
                var newLayer = textLayer.duplicate();
                newLayer.name = "Word_" + word + "_" + (wordId + 1);

                var animator = newLayer.property("ADBE Text Properties").property("ADBE Text
Animators").addProperty("ADBE Text Animator");
                var opacityProp = animator.property("ADBE Text Animator
Properties").addProperty("ADBE Text Opacity");
                var expressionSelector = animator.property("ADBE Text Selectors").addProperty("ADBE
Text Expressible Selector");

                opacityProp.setValue(0);

                expressionSelector.property("ADBE Text Expressible Amount").expression =
"selectorValue * (textIndex != " + (wordId + 1) + ");";

                var basedOnProp = expressionSelector.property("ADBE Text Range Type2");
                basedOnProp.setValue(3);
            }

            textLayer.enabled = false;
        }

        if (!validLayersFound) {
            alert("Не выбрано ни одного текстового слоя!");
            app.endUndoGroup();
            return false;
        }

        app.endUndoGroup();
        return true;
    },

    decompTextPhysical: function() {
        try {
            app.beginUndoGroup("Decompose Text - Characters (Physical)");
            var comp = app.project.activeItem;
            if (!comp || !(comp instanceof CompItem)) {
                alert("Виберіть композицію в таймлайні!");
                app.endUndoGroup();
                return false;
            }
            var selectedLayers = comp.selectedLayers;
            if (selectedLayers.length === 0) {
                alert("Виберіть хоча б один текстовий шар!");
                app.endUndoGroup();
                return false;
            }
            var validLayersFound = false;
            for (var i = 0; i < selectedLayers.length; i++) {
                var textLayer = selectedLayers[i];
                if (!(textLayer instanceof TextLayer)) {
```

```
                continue;
            }
            validLayersFound = true;
            var textString = textLayer.sourceText.value.toString();
            if (!textString || textString.length === 0) {
                alert("Текстовый шар '" + textLayer.name + "' порожній!");
                continue;
            }
            if (textLayer.property("ADBE Text Properties").property("ADBE Text
Document").value.boxText) {
                alert("Paragraph text не підтримується для шару '" + textLayer.name + "'.");
                continue;
            }
            var startTime = textLayer.startTime;
            var outPoint = textLayer.outPoint;
            var layerIndex = textLayer.index;
            var textPos = textLayer.transform.position.value;
            var textDoc = textLayer.property("Text").property("Source Text").value;
            var newLayers = [];
            var totalWidth = 0; // Накопичуємо ширину для зсуву

            for (var charId = 0; charId < textString.length; charId++) {
                var character = textString.charAt(charId);
                if (character.match(/\s+/)) continue;
                var newLayer = comp.layers.addText(character);
                newLayer.name = "Letter_" + character + "_" + (charId + 1);
                newLayer.startTime = startTime;
                newLayer.outPoint = outPoint;
                var newTextProp = newLayer.property("Text").property("Source Text");
                newTextProp.font = textDoc.font;
                newTextProp.fontSize = textDoc.fontSize;
                newTextProp.applyFill = textDoc.applyFill;
                newTextProp.fillColor = textDoc.fillColor;
                var charRect = newLayer.sourceRectAtTime(startTime, false);
                var charPosX = textPos[0] + totalWidth + (charRect.width / 2); // Зсув + центр
                var charPosY = textPos[1] + (charRect.height / 2); // Центр по Y
                newLayer.transform.position.setValue([charPosX, charPosY]);
                newLayer.transform.anchorPoint.setValue([charRect.width / 2, charRect.height / 2]); //
Центр букви
                newLayers.push(newLayer);
                totalWidth += charRect.width; // Додаємо ширину поточної букви
            }
            // Розташовуємо шари прямо над textLayer, від низу до верху
            if (newLayers.length > 0) {
                newLayers[newLayers.length - 1].moveBefore(textLayer); // Останній шар над
textLayer
                for (var j = newLayers.length - 2; j >= 0; j--) {
                    newLayers[j].moveBefore(newLayers[j + 1]); // Кожен попередній над наступним
                }
            }
            textLayer.enabled = false;
        }
```

```
            if (!validLayersFound) {
                alert("Не вибрано жодного текстового шару!");
                app.endUndoGroup();
                return false;
            }
            app.endUndoGroup();
            return true;
        } catch (e) {
            alert("Помилка при декомпозиції тексту: " + e.message);
            app.endUndoGroup();
            return false;
        }
    },

    decompWordsPhysical: function() {
        try {
            app.beginUndoGroup("Decompose Text - Words (Physical)");
            var comp = app.project.activeItem;
            if (!comp || !(comp instanceof CompItem)) {
                alert("Виберіть композицію в таймлайні!");
                app.endUndoGroup();
                return false;
            }
            var selectedLayers = comp.selectedLayers;
            if (selectedLayers.length === 0) {
                alert("Виберіть хоча б один текстовий шар!");
                app.endUndoGroup();
                return false;
            }
            var validLayersFound = false;
            for (var i = 0; i < selectedLayers.length; i++) {
                var textLayer = selectedLayers[i];
                if (!(textLayer instanceof TextLayer)) {
                    continue;
                }
                validLayersFound = true;
                var textString = textLayer.sourceText.value.toString();
                if (!textString || textString.length === 0) {
                    alert("Текстовый шар '" + textLayer.name + "' порожній!");
                    continue;
                }
                if (textLayer.property("ADBE Text Properties").property("ADBE Text
Document").value.boxText) {
                    alert("Paragraph text не підтримується для шару '" + textLayer.name + "'.");
                    continue;
                }
                var startTime = textLayer.startTime;
                var outPoint = textLayer.outPoint;
                var layerIndex = textLayer.index;
                var textPos = textLayer.transform.position.value;
                var textDoc = textLayer.property("Text").property("Source Text").value;
                var words = textString.split(/\s+/);
```

```
                var newLayers = [];
                var totalWidth = 0; // Накопичуємо ширину для зсуву

                for (var wordId = 0; wordId < words.length; wordId++) {
                    var word = words[wordId];
                    if (!word) continue;
                    var newLayer = comp.layers.addText(word);
                    newLayer.name = "Word_" + word + "_" + (wordId + 1);
                    newLayer.startTime = startTime;
                    newLayer.outPoint = outPoint;
                    var newTextProp = newLayer.property("Text").property("Source Text");
                    newTextProp.font = textDoc.font;
                    newTextProp.fontSize = textDoc.fontSize;
                    newTextProp.applyFill = textDoc.applyFill;
                    newTextProp.fillColor = textDoc.fillColor;
                    var wordRect = newLayer.sourceRectAtTime(startTime, false);
                    var wordPosX = textPos[0] + totalWidth + (wordRect.width / 2); // Зсув + центр
                    var wordPosY = textPos[1] + (wordRect.height / 2); // Центр по Y
                    newLayer.transform.position.setValue([wordPosX, wordPosY]);
                    newLayer.transform.anchorPoint.setValue([wordRect.width / 2, wordRect.height / 2]);
// Центр слова
                    newLayers.push(newLayer);
                    totalWidth += wordRect.width + 5; // Додаємо ширину слова + простір
                }
                // Розташовуємо шари прямо над textLayer, від низу до верху
                if (newLayers.length > 0) {
                    newLayers[newLayers.length - 1].moveBefore(textLayer); // Останній шар над
textLayer
                    for (var j = newLayers.length - 2; j >= 0; j--) {
                        newLayers[j].moveBefore(newLayers[j + 1]); // Кожен попередній над наступним
                    }
                }
                textLayer.enabled = false;
            }
            if (!validLayersFound) {
                alert("Не вибрано жодного текстового шару!");
                app.endUndoGroup();
                return false;
            }
            app.endUndoGroup();
            return true;
        } catch (e) {
            alert("Помилка при декомпозиції тексту: " + e.message);
            app.endUndoGroup();
            return false;
        }
    },



    decompBackgroundSolid: function(mode) {
        try {
```

```
app.beginUndoGroup("Add Background Shape to Text");
var comp = app.project.activeItem;
if (!comp || !(comp instanceof CompItem)) {
   alert("Виберіть композицію в таймлайні!");
   app.endUndoGroup();
   return false;
}
var selectedLayers = comp.selectedLayers;
if (selectedLayers.length === 0) {
   alert("Виберіть хоча б один текстовий шар!");
   app.endUndoGroup();
   return false;
}
var validLayersFound = false;
for (var i = 0; i < selectedLayers.length; i++) {
   var textLayer = selectedLayers[i];
   if (!(textLayer instanceof TextLayer)) {
      continue;
   }
   validLayersFound = true;
   var textString = textLayer.sourceText.value.toString();
   if (!textString || textString.length === 0) {
      alert("Текстовый шар '" + textLayer.name + "' порожній!");
      continue;
   }
   if (textLayer.property("ADBE Text Properties").property("ADBE Text
Document").value.boxText) {
      alert("Paragraph text не підтримується для шару '" + textLayer.name + "'.");
      continue;
   }
   var sourceRect = textLayer.sourceRectAtTime(comp.time, false);
   var shapeLayer = comp.layers.addShape();
   shapeLayer.name = "BG_Shape_" + textLayer.name;
   shapeLayer.startTime = textLayer.startTime;
   shapeLayer.outPoint = textLayer.outPoint;
   shapeLayer.moveAfter(textLayer);
   var shapeGroup = shapeLayer.property("Contents").addProperty("ADBE Vector Group");
   var rectContent = shapeGroup.property("Contents");
   var rectPath = rectContent.addProperty("ADBE Vector Shape - Rect");
   rectPath.property("Size").expression =
      "var textLayer = thisComp.layer(\"" + textLayer.name + "\");" +
      "var r = textLayer.sourceRectAtTime(time, false);" +
      "[r.width + 30, r.height + 30];";
   shapeLayer.property("Transform").property("Position").expression =
      "var textLayer = thisComp.layer(\"" + textLayer.name + "\");" +
      "var r = textLayer.sourceRectAtTime(time, false);" +
      "var textPos = textLayer.transform.position;" +
      "var anchor = textLayer.transform.anchorPoint;" +
      "[textPos[0] + r.left + r.width/2 - anchor[0], textPos[1] + r.top + r.height/2 -
anchor[1]];";
   var fill = rectContent.addProperty("ADBE Vector Graphic - Fill");
   fill.property("Color").setValue([0, 0, 0]);
```

```
            fill.property("Opacity").setValue(60);
        }
        if (!validLayersFound) {
            alert("Не вибрано жодного текстового шару!");
            app.endUndoGroup();
            return false;
        }
        app.endUndoGroup();
        return true;
    } catch (e) {
        alert("Помилка при створенні шейпа: " + e.message);
        app.endUndoGroup();
        return false;
    }
},

generateSubtitles: function(text) {
try {
    app.beginUndoGroup("Generate Subtitles with Shapes");

    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Виберіть композицію в вікні проекта чи таймлайну!");
        app.endUndoGroup();
        return false;
    }

    if (text == null || text === undefined) {
        alert("Ошибка: Текст не передан или является undefined/null!");
        app.endUndoGroup();
        return false;
    }

    text = text.replace(/\\n/g, "\n").replace(/\\\\/g, "\\").replace(/\\"/g, "\"");
    var lines = text.split("\n");
    var frameDuration30 = 30 / comp.frameRate;
    var startTime = 0;
    var totalDuration = 0;

    // Calculate total duration based on non-empty lines
    for (var i = 0; i < lines.length; i++) {
        if (lines[i] && lines[i] !== "") {
            totalDuration += frameDuration30;
        }
    }

    // Create new composition for subtitles with dynamic duration
    var subComp = app.project.items.addComp(
        "[subs]",
        comp.width,
        comp.height,
        comp.pixelAspect,
```

```
            totalDuration,
            comp.frameRate
        );

        for (var i = 0; i < lines.length; i++) {
            var line = lines[i];
            if (!line || line === "") continue;

            // Create first text layer
            var textLayer1 = subComp.layers.addText(line);
            textLayer1.name = "Text " + (i + 1) + "-1";
            textLayer1.startTime = startTime;
            textLayer1.outPoint = startTime + frameDuration30;

            // Add Animator: Fill Color -> RGB
            var animatorGroup =
textLayer1.property("Text").property("Animators").addProperty("ADBE Text Animator");
            var animatorProps = animatorGroup.property("Properties");
            var fillColorProp = animatorProps.addProperty("ADBE Text Fill Color");

            // Set Fill Color to #FF9600
            fillColorProp.setValue([255/255, 150/255, 0/255]);

            // Add Range Selector
            var rangeSelector = animatorGroup.property("Selectors").addProperty("ADBE Text
Selector");
            var rangeStartProp = rangeSelector.property("Start");
            var rangeEndProp = rangeSelector.property("End");

            // Set keyframes for Start
            rangeStartProp.setValueAtTime(startTime, 0);
            rangeStartProp.setValueAtTime(startTime + frameDuration30, 100);

            // Set fixed value for End
            rangeEndProp.setValue(0);

            var R = textLayer1.sourceRectAtTime(textLayer1.inPoint, false);
            var textAnchorPointX = R.left + R.width / 2;
            var textAnchorPointY = R.top + R.height / 2;
            textLayer1.property("Transform").property("Anchor Point").setValue([textAnchorPointX,
textAnchorPointY]);
            textLayer1.property("Transform").property("Position").setValue([subComp.width / 2,
subComp.height / 2]);

            // Create second text layer
            var textLayer2 = subComp.layers.addText(line);
            textLayer2.name = "Text " + (i + 1) + "-2";
            textLayer2.startTime = startTime;
            textLayer2.outPoint = startTime + frameDuration30;

            // Sync text content with textLayer1
            var sourceTextValue = textLayer1.property("Text").property("Source Text").value;
```

```
        textLayer2.property("Text").property("Source Text").setValue(sourceTextValue);

        var textAnchorPointX2 = R.left + R.width / 2;
        var textAnchorPointY2 = R.top + R.height / 2;
        textLayer2.property("Transform").property("Anchor Point").setValue([textAnchorPointX2,
textAnchorPointY2]);
        textLayer2.property("Transform").property("Position").setValue([subComp.width / 2,
subComp.height / 2]);
        textLayer2.moveAfter(textLayer1);

        // Create shape layer
        var shapeLayer = subComp.layers.addShape();
        shapeLayer.name = "Background " + (i + 1);
        shapeLayer.startTime = startTime;
        shapeLayer.outPoint = startTime + frameDuration30;

        var shapeAnchorPointX = R.width / 2;
        var shapeAnchorPointY = R.height / 2;
        shapeLayer.property("Transform").property("Anchor Point").setValue([shapeAnchorPointX,
shapeAnchorPointY]);
        shapeLayer.property("Transform").property("Position").setValue([subComp.width / 2,
subComp.height / 2]);

        var shapeGroup = shapeLayer.property("Contents").addProperty("ADBE Vector Group");
        var rectContent = shapeGroup.property("Contents");
        var rectPath = rectContent.addProperty("ADBE Vector Shape - Rect");

        var paddingWidth = 20;
        var paddingHeight = 10;
        rectPath.property("Size").setValue([R.width + paddingWidth, R.height + paddingHeight]);

        // Expression for dynamic size adjustment
        rectPath.property("Size").expression =
          "var textLayer = thisComp.layer(\"Text " + (i + 1) + "-1\");" +
          "var r = textLayer.sourceRectAtTime(time, false);" +
          "var paddingWidth = 20;" +
          "var paddingHeight = 10;" +
          "[r.width + paddingWidth, r.height + paddingHeight];";

        // Expression for dynamic position adjustment
        shapeLayer.property("Transform").property("Position").expression =
          "var textLayer = thisComp.layer(\"Text " + (i + 1) + "-1\");" +
          "var r = textLayer.sourceRectAtTime(time, false);" +
          "var textPos = textLayer.transform.position;" +
          "var anchor = textLayer.transform.anchorPoint;" +
          "[textPos[0] + r.left + r.width/2 - anchor[0], textPos[1] + r.top + r.height/2 - anchor[1]];";

        var fill = rectContent.addProperty("ADBE Vector Graphic - Fill");
        fill.property("Color").setValue([0, 0, 0]);
        fill.property("Opacity").setValue(60);

        // Ensure shape layer is behind text layers
```

```
            shapeLayer.moveAfter(textLayer2);

            startTime += frameDuration30;
        }

        // Add Adjustment Layer inside subComp
        var adjLayer = subComp.layers.addSolid([1, 1, 1], "Adjustment Layer", subComp.width,
subComp.height, subComp.pixelAspect);
        adjLayer.adjustmentLayer = true;
        adjLayer.moveToBeginning(); // Move to top of layer stack in subComp

        // Apply preset "text_heatwave"
        var presetPath = "C:\\Users\\Користувач\\Documents\\Adobe\\After Effects 2025\\User
Presets\\[sxt]\\text_heatwave.ffx";
        var presetFile = new File(presetPath);
        if (presetFile.exists) {
            adjLayer.applyPreset(presetFile);
        } else {
            alert("Ошибка: Пресет 'text_heatwave.ffx' не найден по пути: " + presetPath);
        }

        // Add the sub composition to the main composition
        var subCompLayer = comp.layers.add(subComp);
        subCompLayer.name = "[subs]";

        app.endUndoGroup();
        return true;
    } catch (e) {
        alert("Ошибка при генерации субтитров: " + e.message);
        app.endUndoGroup();
        return false;
    }
}

};

if (! $._ext) {
    $._ext = {};
}

$._ext.setAnchorPointGrid = function(position) {
    $._text.setAnchorPointGrid(position);
};

$._ext.decompText = function() {
    $._text.decompText();
};

$._ext.decompWords = function() {
    $._text.decompWords();
};
```

```
    $._ext.decompTextPhysical = function() {
        $._text.decompTextPhysical();
    };

    $._ext.decompWordsPhysical = function() {
        $._text.decompWordsPhysical();
    };

    $._ext.decompBackgroundSolid = function(mode) {
        $._text.decompBackgroundSolid(mode);
    };

    $._ext.generateSubtitles = function(text) {
  $._subtitles.generateSubtitles(text);
};
```

### *3D.jsx*

```
var selectedImage = null;

$._3d = {
    selectImageForCubes: function() {
        var imageFile = File.openDialog("Выберите изображение", "*.jpg;*.jpeg;*.png");
        if (imageFile) {
            selectedImage = app.project.importFile(new ImportOptions(imageFile));
            alert("Изображение выбрано: " + imageFile.name);
        } else {
            alert("Зображення не обрано, куб буде створено без фото");
        }
    },

    createBasicCube: function() {
    app.beginUndoGroup("Create Basic 3D Cube");

    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Выберите композицию в окне проекта или таймлайне!");
        app.endUndoGroup();
        return;
    }

    var cubeSize = 1080;
    var mainComp = comp;

    var cubeFaces = [];
    var cubeLayers = [];


    for (var i = 0; i < 6; i++) {
        var faceComp = app.project.items.addComp("Basic Cube Face " + (i+1), cubeSize, cubeSize, 1.0,
mainComp.duration, mainComp.frameRate);
        cubeFaces.push(faceComp);
```

```
        if (selectedImage) {
            var imageLayer = faceComp.layers.add(selectedImage);
            imageLayer.position.setValue([cubeSize / 2, cubeSize / 2]);
        } else {
            faceComp.layers.addSolid([0.5, 0.5, 0.5], "Solid", cubeSize, cubeSize, 1);
        }
    }

    for (var i = 0; i < 6; i++) {
        var layer = mainComp.layers.add(cubeFaces[i]);
        layer.threeDLayer = true;
        cubeLayers.push(layer);
    }

    cubeLayers[0].position.setValue([mainComp.width/2, mainComp.height/2, -cubeSize/2]);
    cubeLayers[1].position.setValue([mainComp.width/2, mainComp.height/2, cubeSize/2]);
    cubeLayers[1].rotationY.setValue(180);
    cubeLayers[2].position.setValue([mainComp.width/2 - cubeSize/2, mainComp.height/2, 0]);
    cubeLayers[2].rotationY.setValue(90);
    cubeLayers[3].position.setValue([mainComp.width/2 + cubeSize/2, mainComp.height/2, 0]);
    cubeLayers[3].rotationY.setValue(-90);
    cubeLayers[4].position.setValue([mainComp.width/2, mainComp.height/2 - cubeSize/2, 0]);
    cubeLayers[4].rotationX.setValue(-90);
    cubeLayers[5].position.setValue([mainComp.width/2, mainComp.height/2 + cubeSize/2, 0]);
    cubeLayers[5].rotationX.setValue(90);

    var cubeController = mainComp.layers.addNull();
    cubeController.name = "Basic Cube Ctrl";
    cubeController.threeDLayer = true;
    cubeController.position.setValue([mainComp.width/2, mainComp.height/2, 0]);

    // Привязываем грани к контроллеру
    for (var i = 0; i < cubeLayers.length; i++) {
        cubeLayers[i].parent = cubeController;
    }

    app.endUndoGroup();
},

    createSplitCube: function(parts) {
        app.beginUndoGroup("Create Split 3D Cube");

        var comp = app.project.activeItem;
        if (!comp || !(comp instanceof CompItem)) {
            alert("Виберіть композицію в вікні проекту або таймлайні!");
            app.endUndoGroup();
            return;
        }

        var cubeSize = 1080;
        var mainComp = comp;
```

```
    var partHeight = cubeSize / parts;
    var partControllers = [];

    for (var p = 0; p < parts; p++) {
        var faces = [];
        var layers = [];
        var color = (p % 2 === 0) ? [0.5, 0.5, 0.5] : [0.3, 0.3, 0.3];

        if (p === 0) {
            faces.push(app.project.items.addComp("Part " + (p+1) + " Front Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Back Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Top Face", cubeSize, cubeSize,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Left Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Right Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Bottom Face", cubeSize,
cubeSize, 1.0, mainComp.duration, mainComp.frameRate));
        } else if (p === parts - 1) {
            faces.push(app.project.items.addComp("Part " + (p+1) + " Front Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Back Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Bottom Face", cubeSize,
cubeSize, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Left Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Right Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Top Face", cubeSize, cubeSize,
1.0, mainComp.duration, mainComp.frameRate));
        } else {
            faces.push(app.project.items.addComp("Part " + (p+1) + " Front Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Back Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Left Face", cubeSize, partHeight,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Right Face", cubeSize,
partHeight, 1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Top Face", cubeSize, cubeSize,
1.0, mainComp.duration, mainComp.frameRate));
            faces.push(app.project.items.addComp("Part " + (p+1) + " Bottom Face", cubeSize,
cubeSize, 1.0, mainComp.duration, mainComp.frameRate));
        }


        for (var i = 0; i < faces.length; i++) {
            if (selectedImage) {
```

```
                var imageLayer = faces[i].layers.add(selectedImage);
                if (faces[i].height === cubeSize) {
                    imageLayer.position.setValue([cubeSize / 2, cubeSize / 2]);
                } else {
                    var yPos;
                    if (parts === 2) {
                        yPos = (p === 0) ? partHeight : 0;
                    } else if (parts === 3) {
                        yPos = (p === 0) ? partHeight : (p === 1) ? -180 : 0;
                    } else if (parts === 4) {
                        yPos = (p === 0) ? partHeight : (p === 1) ? 180 : (p === 2) ? -90 : 0;
                    }
                    imageLayer.position.setValue([cubeSize / 2, yPos]);
                }
            } else {
                faces[i].layers.addSolid(color, "Solid", faces[i].width, faces[i].height, 1.0);
            }
        }

        for (var i = 0; i < faces.length; i++) {
            var layer = mainComp.layers.add(faces[i]);
            layer.threeDLayer = true;
            layers.push(layer);
        }

        var centerY = mainComp.height/2 - cubeSize/2 + partHeight * p + partHeight/2;
        layers[0].position.setValue([mainComp.width/2, centerY, -cubeSize/2]);
        layers[1].position.setValue([mainComp.width/2, centerY, cubeSize/2]);
        layers[1].rotationY.setValue(180);
        if (p === 0) {
            layers[2].position.setValue([mainComp.width/2, mainComp.height/2 - cubeSize/2, 0]);
            layers[2].rotationX.setValue(-90);
            layers[3].position.setValue([mainComp.width/2 - cubeSize/2, centerY, 0]);
            layers[3].rotationY.setValue(90);
            layers[4].position.setValue([mainComp.width/2 + cubeSize/2, centerY, 0]);
            layers[4].rotationY.setValue(-90);
            layers[5].position.setValue([mainComp.width/2, mainComp.height/2 - cubeSize/2 +
partHeight, 0]);
            layers[5].rotationX.setValue(90);
        } else if (p === parts - 1) {
            layers[2].position.setValue([mainComp.width/2, mainComp.height/2 + cubeSize/2, 0]);
            layers[2].rotationX.setValue(90);
            layers[3].position.setValue([mainComp.width/2 - cubeSize/2, centerY, 0]);
            layers[3].rotationY.setValue(90);
            layers[4].position.setValue([mainComp.width/2 + cubeSize/2, centerY, 0]);
            layers[4].rotationY.setValue(-90);
            layers[5].position.setValue([mainComp.width/2, mainComp.height/2 + cubeSize/2 -
partHeight, 0]);
            layers[5].rotationX.setValue(-90);
        } else {
            layers[2].position.setValue([mainComp.width/2 - cubeSize/2, centerY, 0]);
            layers[2].rotationY.setValue(90);
```

```
            layers[3].position.setValue([mainComp.width/2 + cubeSize/2, centerY, 0]);
            layers[3].rotationY.setValue(-90);
            layers[4].position.setValue([mainComp.width/2, centerY - partHeight/2, 0]);
            layers[4].rotationX.setValue(-90);
            layers[5].position.setValue([mainComp.width/2, centerY + partHeight/2, 0]);
            layers[5].rotationX.setValue(90);
        }

        var partController = mainComp.layers.addNull();
        if (parts === 2) {
            partController.name = (p === 0) ? "Top Ctrl" : "Bottom Ctrl";
        } else if (parts === 3) {
            partController.name = (p === 0) ? "Top Ctrl" : (p === 1) ? "Mid Ctrl" : "Bottom Ctrl";
        } else if (parts === 4) {
            partController.name = (p === 0) ? "Top Ctrl" : (p === 1) ? "Upper Mid Ctrl" : (p === 2) ?
"Lower Mid Ctrl" : "Bottom Ctrl";
        }
        partController.threeDLayer = true;
        partController.position.setValue([mainComp.width/2, centerY, 0]);

        for (var i = 0; i < layers.length; i++) {
            layers[i].parent = partController;
        }

        partControllers.push(partController);
    }

    var mainController = mainComp.layers.addNull();
    mainController.name = "Main Cube Ctrl";
    mainController.threeDLayer = true;
    mainController.position.setValue([mainComp.width/2, mainComp.height/2, 0]);

    for (var i = 0; i < partControllers.length; i++) {
        partControllers[i].parent = mainController;
    }

    app.endUndoGroup();
  },

  createWireframeCube: function() {
    app.beginUndoGroup("Create Wireframe 3D Cube");

    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Виберіть композицію в вікні проекту або таймлайні!");
        app.endUndoGroup();
        return;
    }

    var cubeSize = 1080;
    var mainComp = comp;
```

```
        var cubeLayers = [];
        var strokeWidth = 10;

        for (var i = 0; i < 6; i++) {
            var shapeLayer = mainComp.layers.addShape();
            shapeLayer.name = "Wireframe Face " + (i+1);
            shapeLayer.threeDLayer = true;

            var shapeGroup = shapeLayer.property("Contents").addProperty("ADBE Vector Group");
            var rect = shapeGroup.property("Contents").addProperty("ADBE Vector Shape - Rect");
            rect.property("ADBE Vector Rect Size").setValue([cubeSize, cubeSize]);
            rect.property("ADBE Vector Rect Position").setValue([0, 0]);

            var stroke = shapeGroup.property("Contents").addProperty("ADBE Vector Graphic - Stroke");
            stroke.property("ADBE Vector Stroke Color").setValue([1, 1, 1, 1]);
            stroke.property("ADBE Vector Stroke Width").setValue(strokeWidth);

            var fill = shapeGroup.property("Contents").addProperty("ADBE Vector Graphic - Fill");
            fill.property("ADBE Vector Fill Opacity").setValue(0);

            cubeLayers.push(shapeLayer);
        }

        cubeLayers[0].position.setValue([mainComp.width/2, mainComp.height/2, -cubeSize/2]);
        cubeLayers[1].position.setValue([mainComp.width/2, mainComp.height/2, cubeSize/2]);
        cubeLayers[1].rotationY.setValue(180);
        cubeLayers[2].position.setValue([mainComp.width/2 - cubeSize/2, mainComp.height/2, 0]);
        cubeLayers[2].rotationY.setValue(90);
        cubeLayers[3].position.setValue([mainComp.width/2 + cubeSize/2, mainComp.height/2, 0]);
        cubeLayers[3].rotationY.setValue(-90);
        cubeLayers[4].position.setValue([mainComp.width/2, mainComp.height/2 - cubeSize/2, 0]);
        cubeLayers[4].rotationX.setValue(-90);
        cubeLayers[5].position.setValue([mainComp.width/2, mainComp.height/2 + cubeSize/2, 0]);
        cubeLayers[5].rotationX.setValue(90);

        var cubeController = mainComp.layers.addNull();
        cubeController.name = "Wireframe Cube Ctrl";
        cubeController.threeDLayer = true;
        cubeController.position.setValue([mainComp.width/2, mainComp.height/2, 0]);

        for (var i = 0; i < cubeLayers.length; i++) {
            cubeLayers[i].parent = cubeController;
        }

        app.endUndoGroup();
    }
};
```

# Додаток Д. Ілюстрований матеріал


Рисунок Д.1 – Титульний слайд

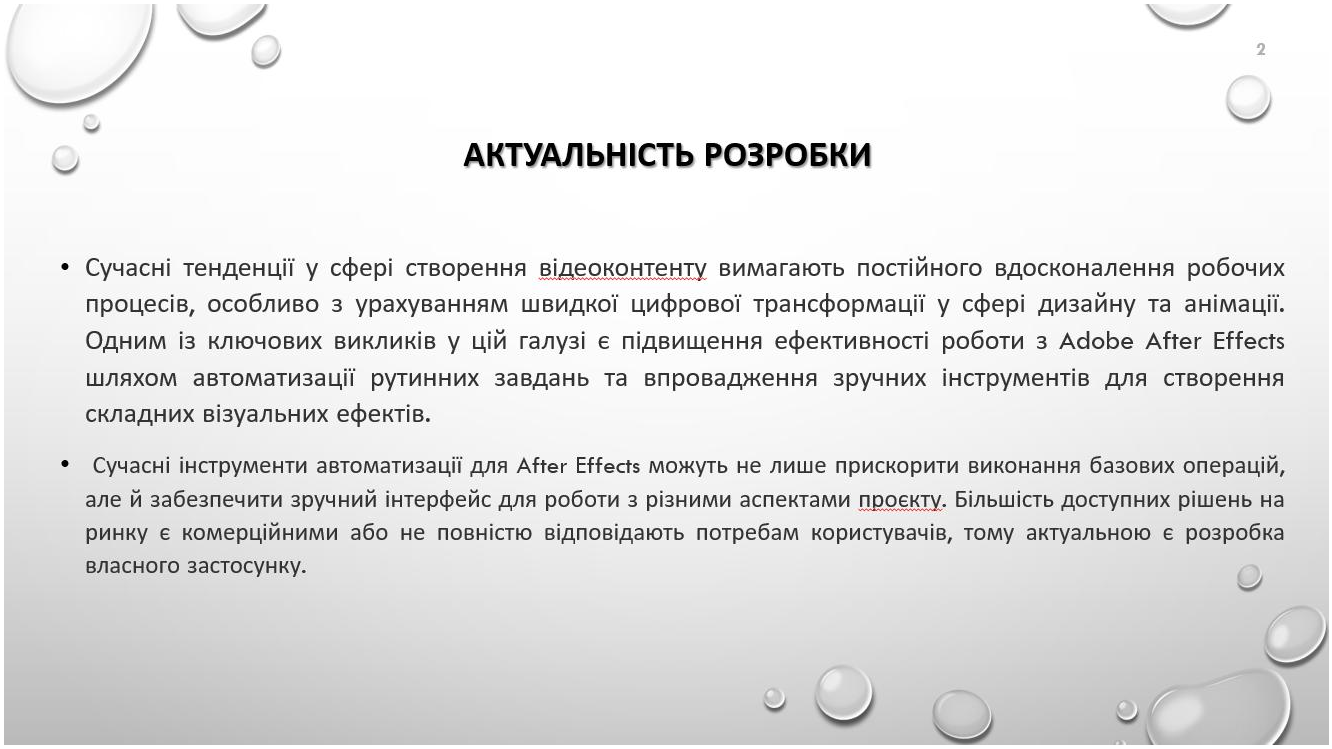



Рисунок Д.2 – Актуальність розробки

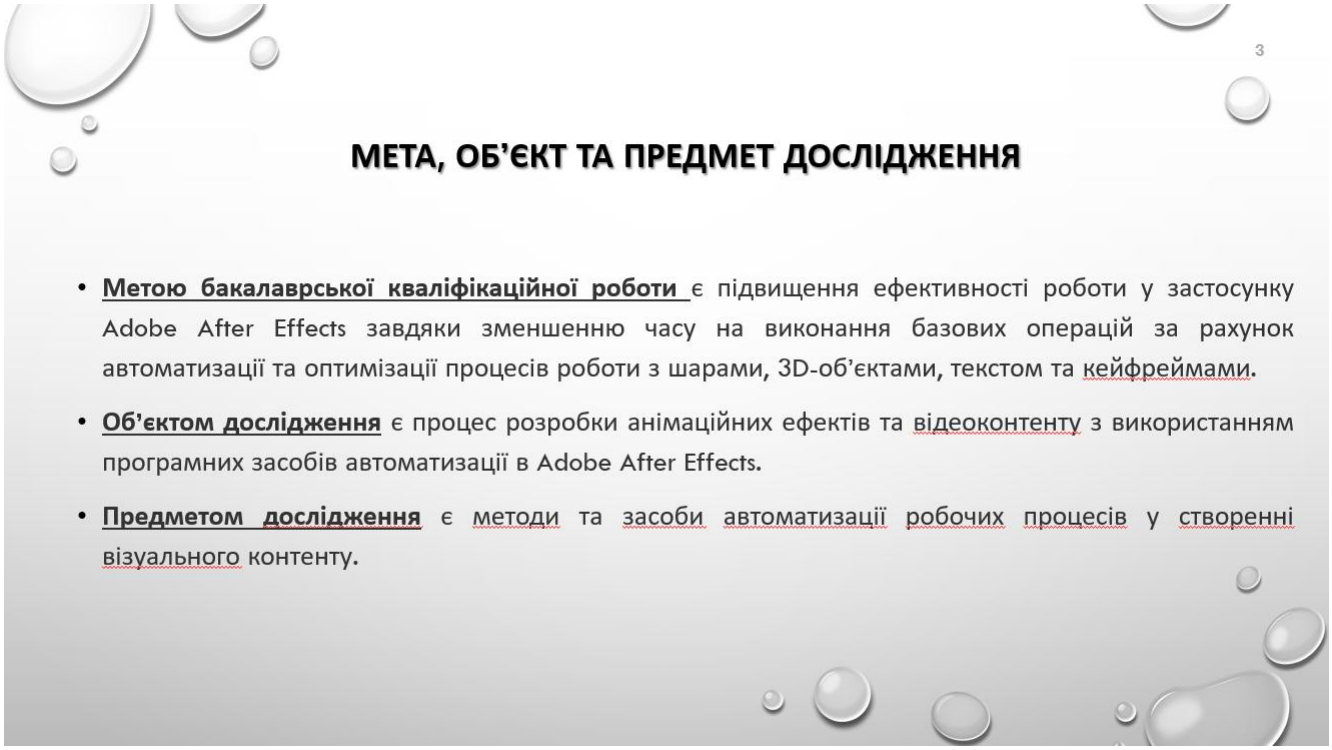



Рисунок Д.3 – Мета, об'єкт та предмет дослідження

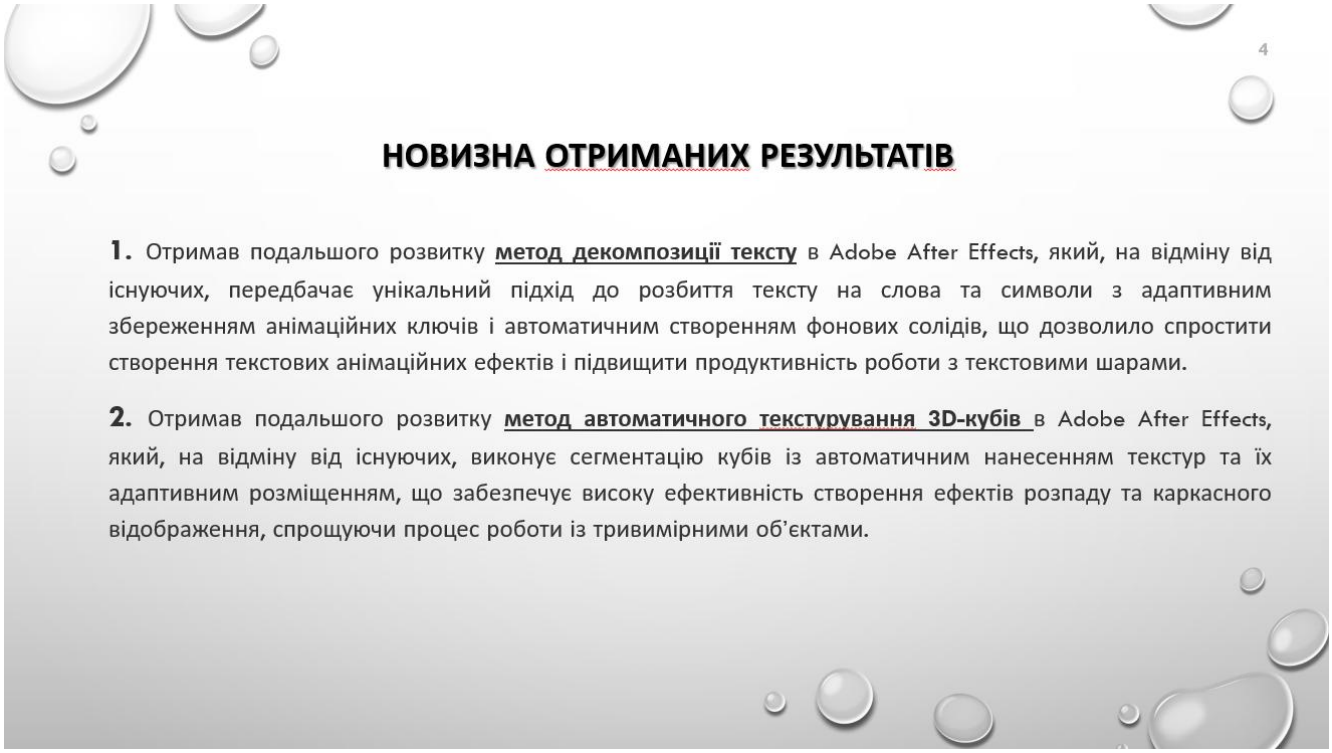



Рисунок Д.4 – Новизна отриманих результатів


Рисунок Д.5 – Практичне значення отриманих результатів

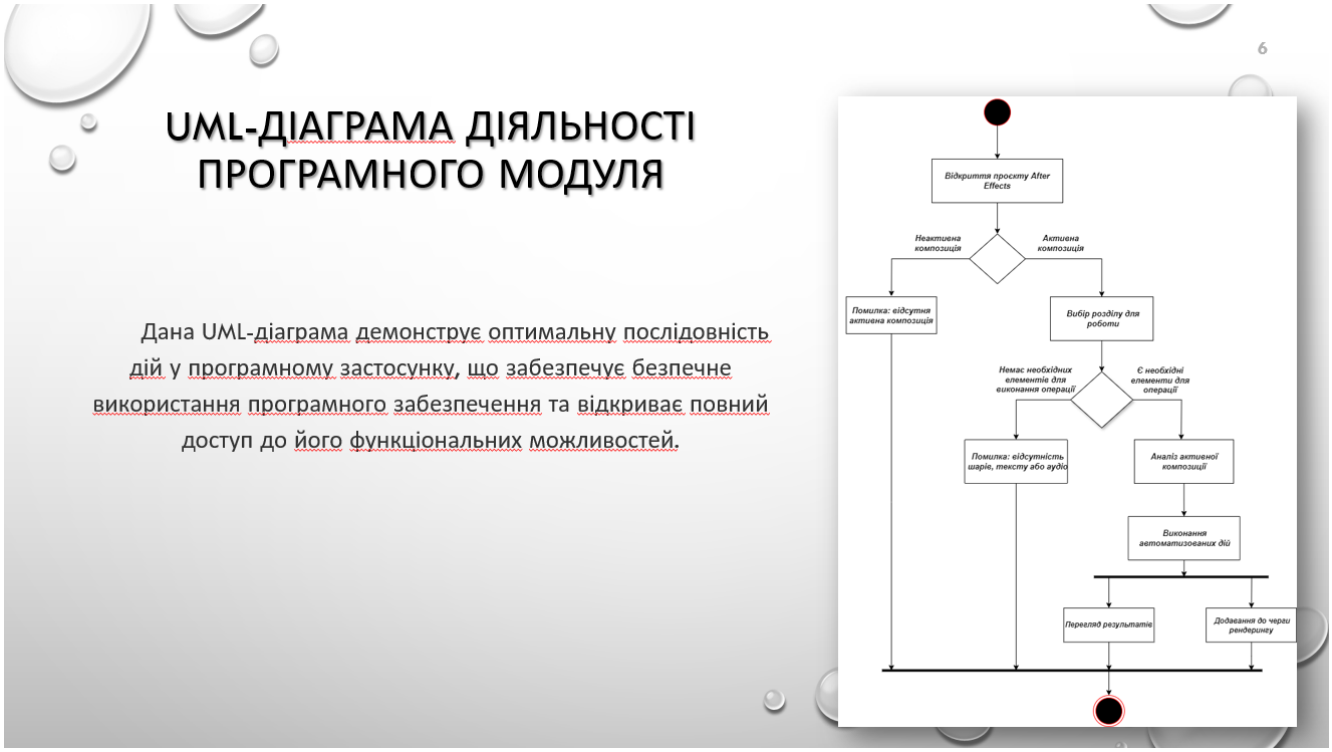



Рисунок Д.6 – UML-діаграма діяльності

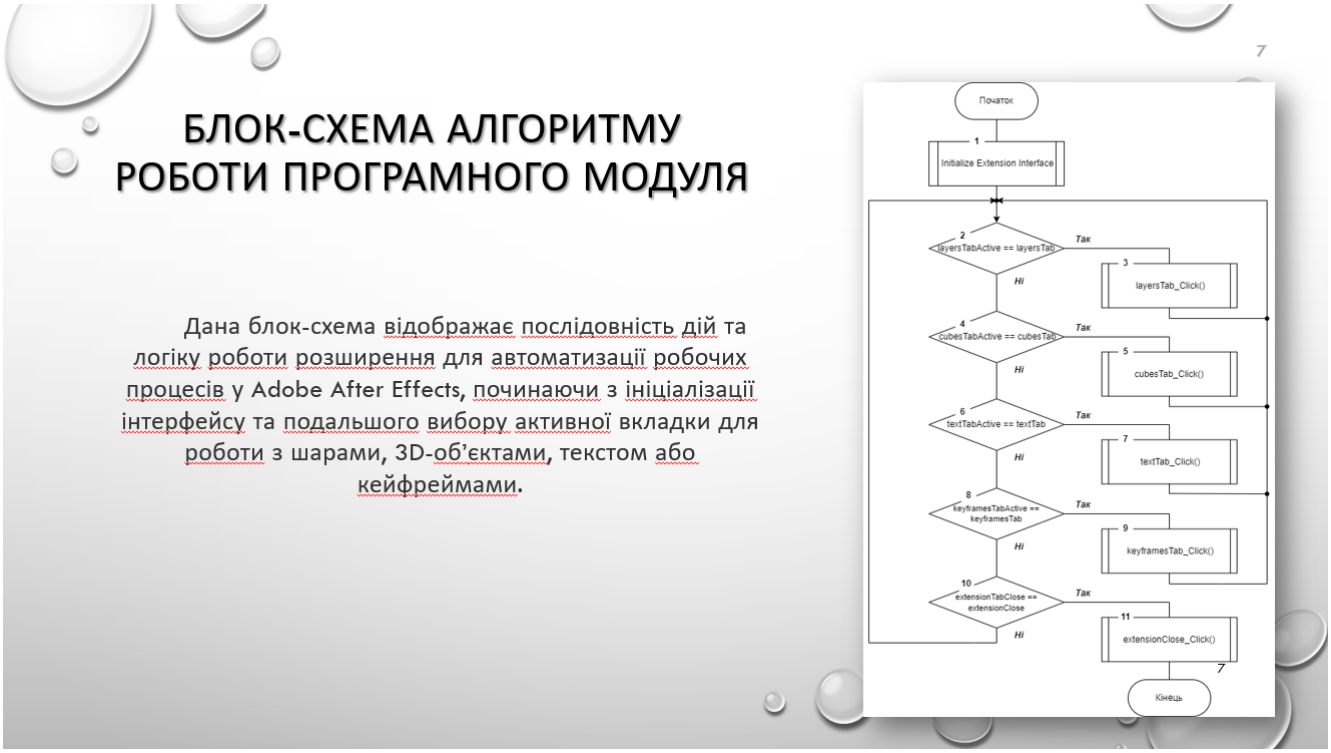



Рисунок Д.7 – Блок-схема алгоритму роботи програмного модуля

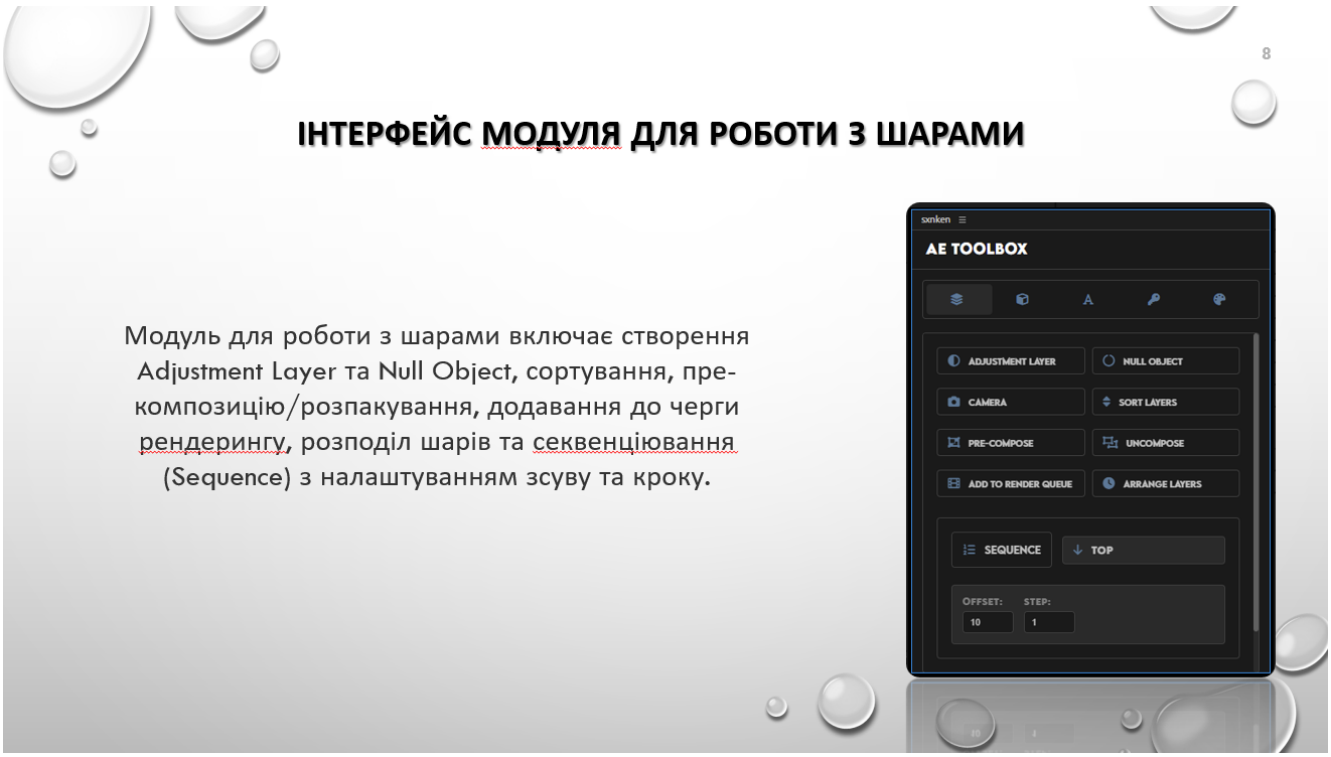



Рисунок Д.8 – Інтерфейс модуля для роботи з шарами

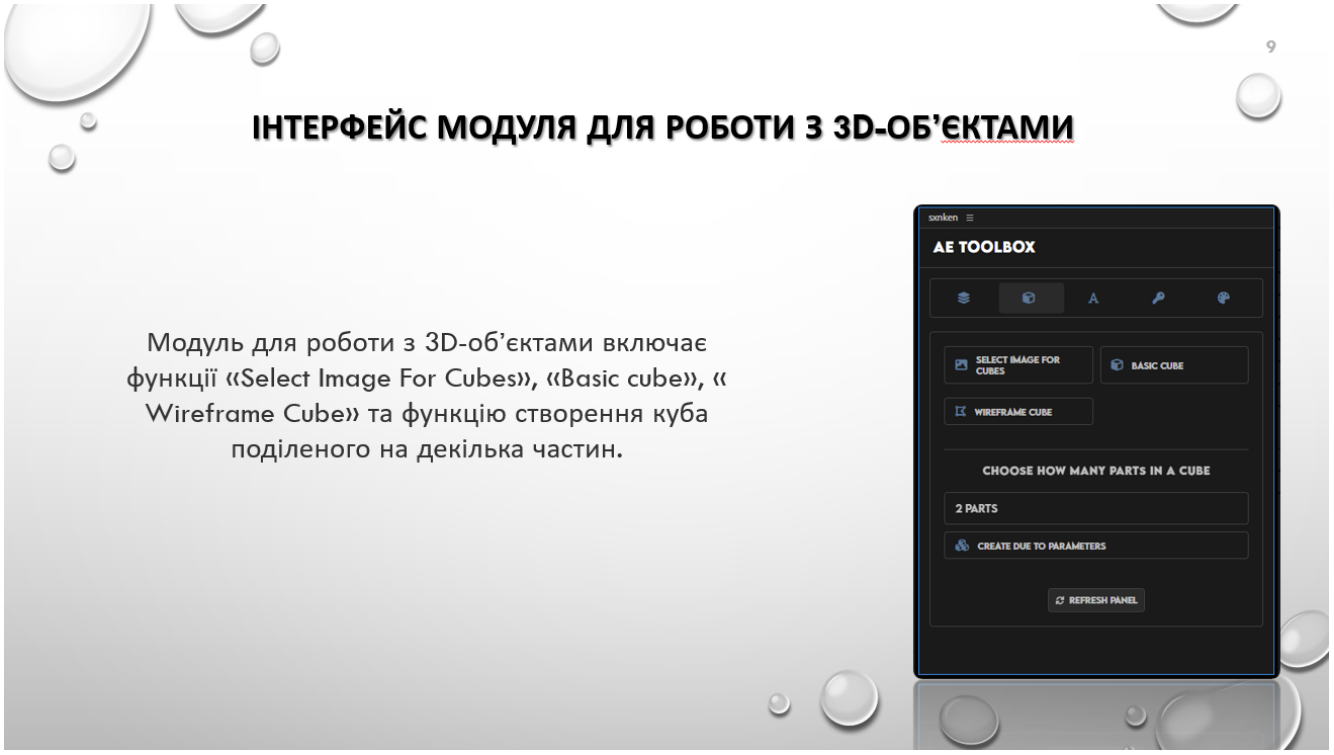



Рисунок Д.9 – Інтерфейс модуля для роботи з 3D-об'єктами

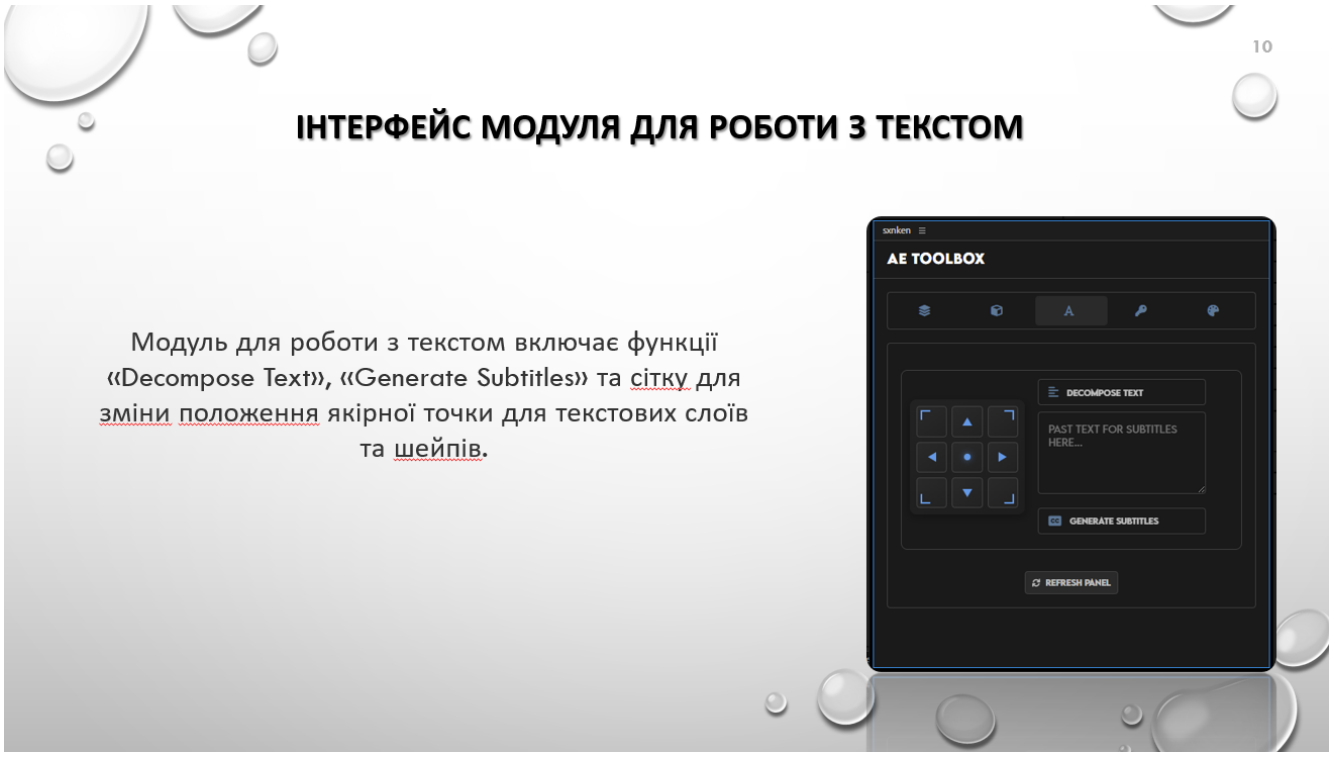



Рисунок Д.10 – Інтерфейс модуля для роботи з текстом

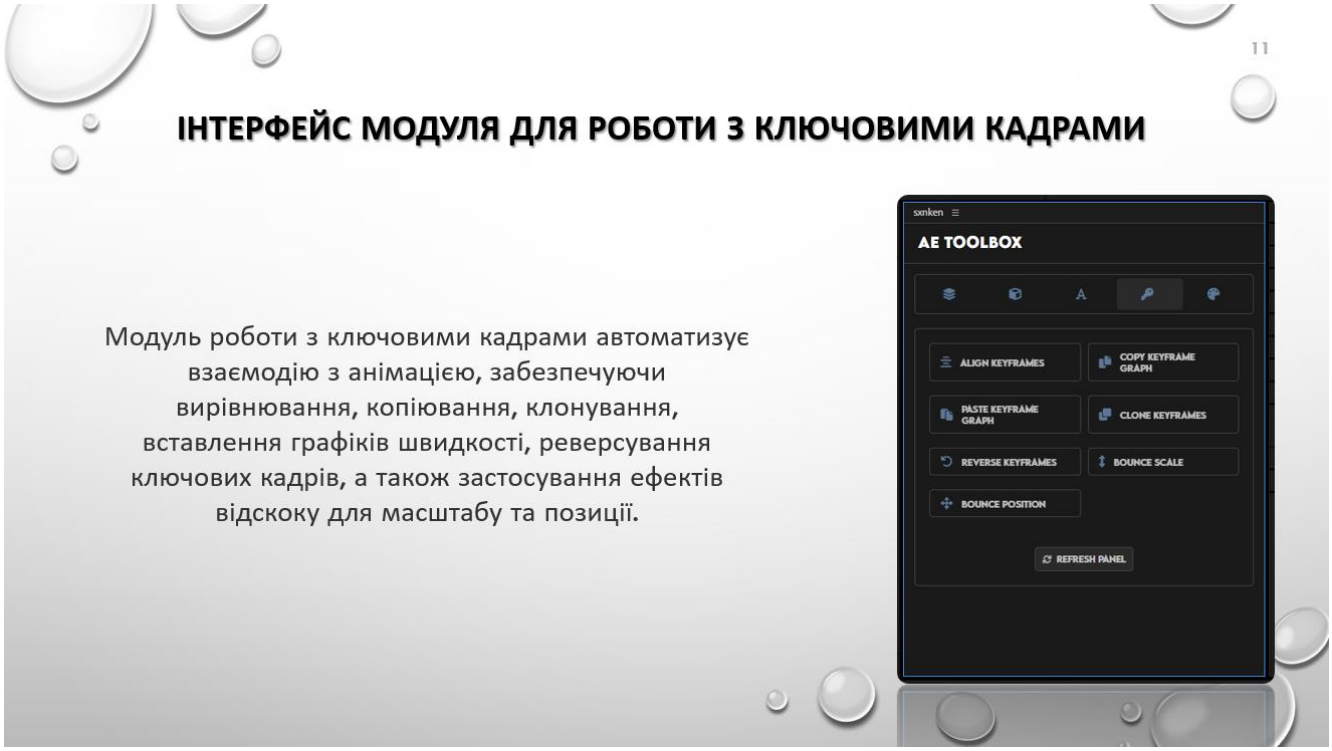



Рисунок Д.11 – Інтерфейс модуля для роботи з ключовими кадрами

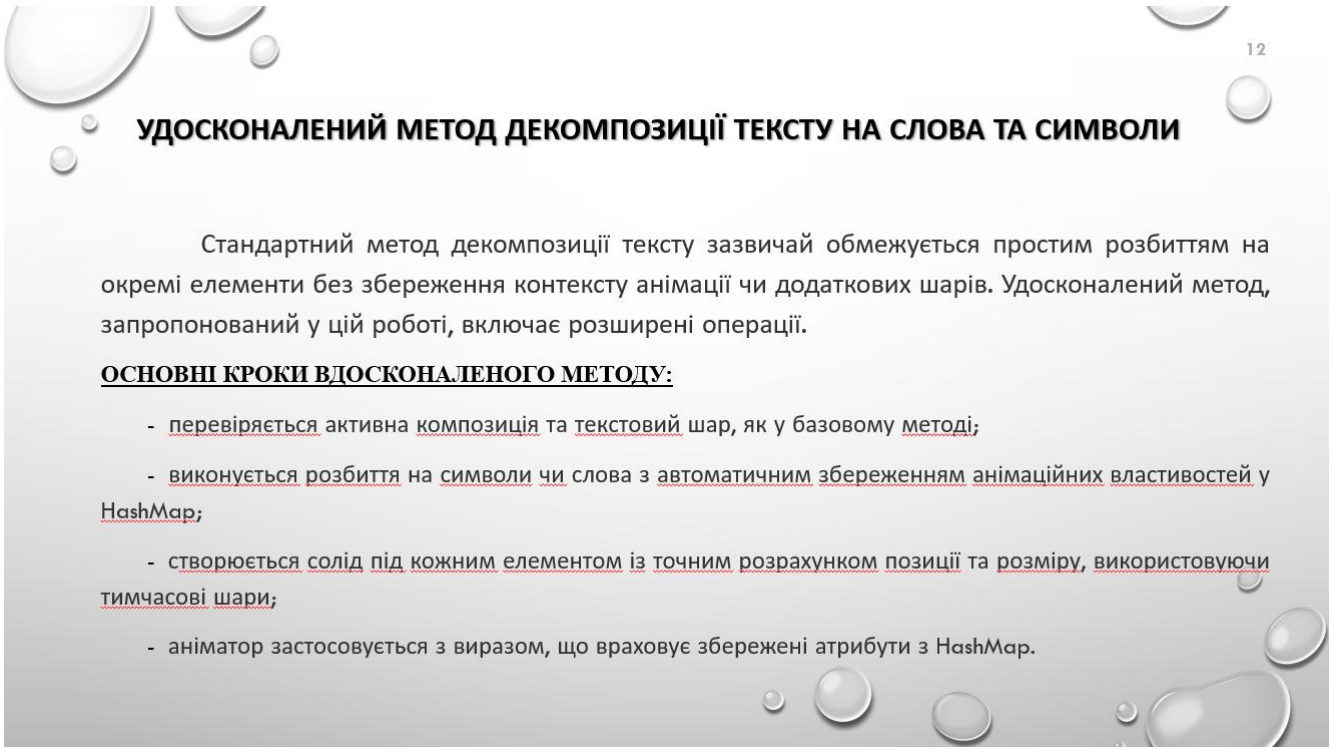



Рисунок Д.12 – Удосконалений метод декомпозиції тексту на слова та символи

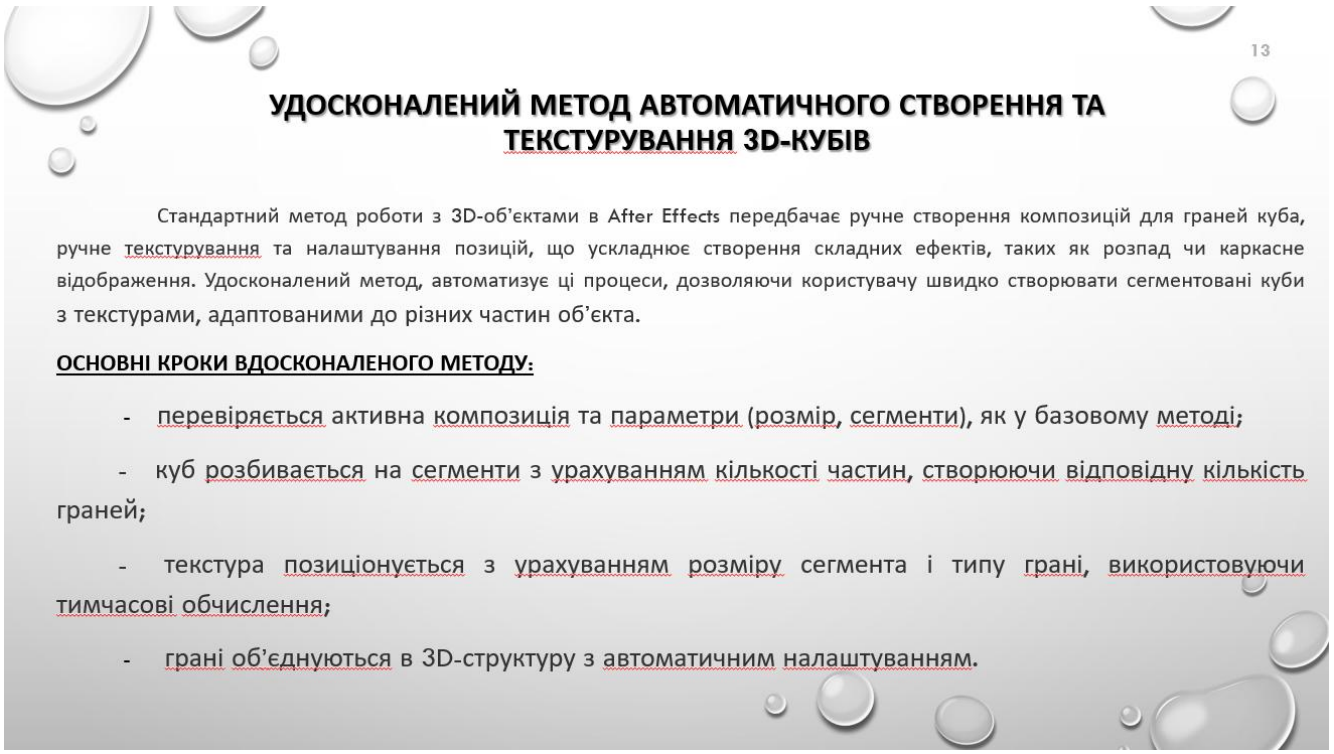



Рисунок Д.13 – Удосконалений метод автоматичного створення та текстурування 3D-кубів

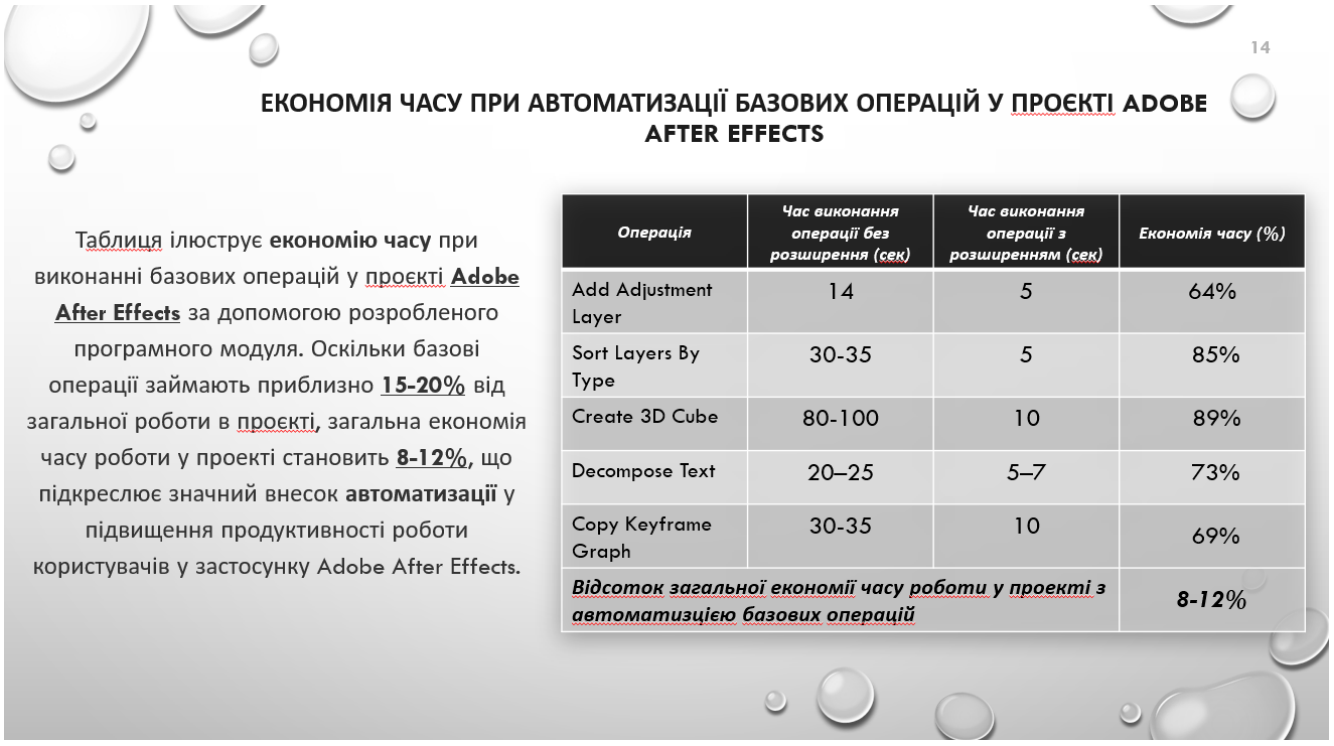



Рисунок Д.14 – Автоматизація базових операції у проєкті Adobe After Effects

15

## АПРОБАЦІЯ ТА ПУБЛІКАЦІЇ РЕЗУЛЬТАТІВ РОБОТИ

### АПРОБАЦІЯ

Результати роботи було представлено на LIV Всеукраїнській науково-технічній конференції підрозділів Вінницького національного технічного університету (Вінниця, 2025 р.) і XIII Міжнародній науково-практичній конференції з інформаційних систем та технологій Infocom Advanced Solutions 2025 (Київ, 2025 р.).

### ПУБЛІКАЦІЇ

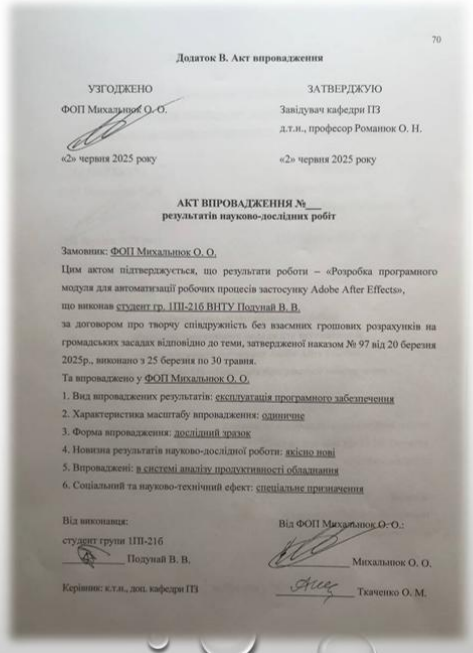
**1.** Подунай В.В. Програмний модуль для автоматизації робочих процесів застосунку Adobe After Effects. LIV Всеукраїнська науково-технічна конференція факультету інформаційних технологій та інженерії Вінниця: ВНТУ, 2025.

**2.** Подунай В.В. Розробка програмного модуля для автоматизації базових операцій застосунку Adobe After Effects. XIII Міжнародна науково-практична конференція з інформаційних систем та технологій Infocom Advanced Solutions 2025. Київ: КПІ, 2025. 3с.

Рисунок Д.15 – Апробація та публікації результатів роботи

16

## АКТ ВПРОВАДЖЕННЯ



70

**Додаток В. Акт впровадження**

УЗГОДЖЕНО
ФОП Михальнюк О. О.
«2» червня 2025 року

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
д.т.н., професор Романюк О. Н.
«2» червня 2025 року

**АКТ ВПРОВАДЖЕННЯ №___**
**результатів науково-дослідних робіт**

Замовник: ФОП Михальнюк О. О.
Цим актом підтверджується, що результати роботи – «Розробка програмного модуля для автоматизації робочих процесів застосунку Adobe After Effects», що виконав студент гр. ІПІ-21б ВНТУ Подунай В. В.
за договором про творчу співдружність без взаємних грошових розрахунків на громадських засадах відповідно до теми, затвердженої наказом № 97 від 20 березня 2025р., виконано з 25 березня по 30 травня.
Та впроваджено у ФОП Михальнюк О. О.
1. Вид впроваджених результатів: експлуатація програмного забезпечення
2. Характеристика масштабу впровадження: одиничне
3. Форма впровадження: дослідний зразок
4. Новизна результатів науково-дослідної роботи: якісно нові
5. Впроваджені: в системі аналізу продуктивності обладнання
6. Соціальний та науково-технічний ефект: спеціальне призначення

Від виконавця:
студент групи ІПІ-21б
_____ Подунай В. В.

Від ФОП Михальнюк О. О.:
_____ Михальнюк О. О.

Керівник: к.т.н., доц. кафедри ПЗ
_____ Ткаченко О. М.

Рисунок Д.16 – Акт впровадження

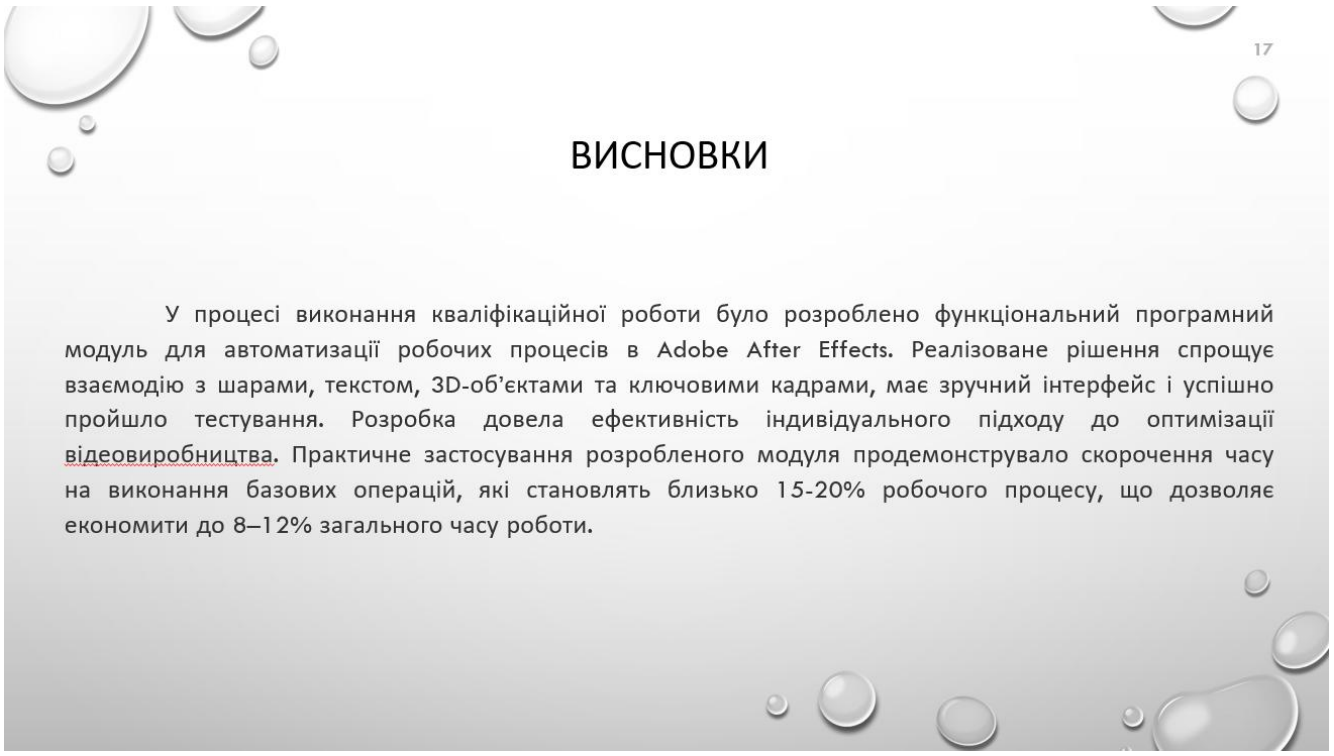



Рисунок Д.17 – Висновки


Рисунок Д.18 – Кінцевий слайд