

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка розширення до програмного
забезпечення для автоматизації процесів анімації та відео
монтажу"»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

*Кваліфікаційна робота містить результати власних досліджень.
Використання ідей, результатів і текстів інших авторів мають посилання
на відповідне джерело*

_____ Максим ПОПОВИЧ
(підпис)

Виконав: здобувач вищої освіти групи ПД-44

_____ Максим ПОПОВИЧ

Керівник: _____ Юрій АБРОСКІН
асист.

Рецензент: _____

Київ 2026

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____ Ірина ЗАМРІЙ

« ____ » _____ 2026 р.

**ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ**

Поповичу Максиму Сергійовичу

1. Тема кваліфікаційної роботи: «Розробка розширення до програмного забезпечення для автоматизації процесів анімації та відео монтажу»
керівник кваліфікаційної роботи асистент кафедри ІІЗ Юрій АБРОСКИН,

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: матеріали переддипломної практики, програмний проєкт FAST PANEL, документація Adobe CEP та After Effects Scripting, результати тестування панелі в типових сценаріях motion-дизайну

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі автоматизації роботи з шарами та субтитрами в Adobe After Effects.

2. Огляд засобів реалізації CEP-розширення та вимог до програмного забезпечення.

3. Проєктування архітектури, сценаріїв використання і алгоритмів FAST PANEL.

4. Реалізація, тестування та оцінювання практичної корисності розробленої панелі.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.

2. Вимоги до програмного забезпечення.

3. Програмні засоби реалізації.

4. Діаграма прецедентів за скриптом Fast Panel.

5. Послідовність створення шару.

6. Архітектура CEP-розширення

7. Екранні форми.

8. Труднощі під час розробки.

6. Дата видачі завдання «23» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	25.02 - 24.03.2026	
2	Аналіз предметної галузі та типових операцій в After Effects	05.03-13.03.2026	
3	Формування функціональних і нефункціональних вимог	14.03-20.03.2026	
4	Проєктування архітектури CEP панелі FAST PANEL	21.03-05.04.2026	
5	Реалізація модулів створення шарів в Anchor Point, Hotkeys, SRT	06.03-24.04.2026	
6	Тестування роботи панелі у типових сценаріях	25.04-05.05.2026	
7	Підготовка пояснювальної записки та демонстраційних матеріалів	05.04-05.05.2026	
8	Розробка демонстраційних матеріалів	06.05-20.05.2026	
9	Попередній захист роботи	21.05-02.06.2026	

Здобувач вищої освіти

_____ (підпис)

Максим ПОПОВИЧ

Керівник
кваліфікаційної роботи

_____ (підпис)

Юрій АБРОСКІН

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 81 стор., 11 табл., 8 рис., 25 джерел.

Мета роботи – підвищення ефективності виконання типових операцій motion-дизайнера в Adobe After Effects за рахунок автоматизації рутинних дій.

Об'єкт дослідження – процес автоматизації операцій з шарами, композиціями та субтитрами в Adobe After Effects.

Предмет дослідження – програмне забезпечення FAST PANEL для автоматизованого створення шарів, зміни точки прив'язки, керування таймінгом та імпорту субтитрів у Adobe After Effects.

Короткий зміст роботи: В роботі проаналізовано задачі motion-дизайнера та типові проблеми ручної роботи з шарами і субтитрами в Adobe After Effects. Проаналізовано існуючі засоби автоматизації: стандартні засоби After Effects, окремі JSX-скрипти та сторонні панелі Animation Composer, Motion Tools і AEUX. Розроблено архітектуру застосунку та програмно реалізовано ключові функціональні можливості, зокрема: створення службових шарів, прив'язку нового шару до виділеного (Trim to Layer), зміну точки прив'язки (Anchor Point), підгонку шарів до композиції (Fit), ущільнення ключових кадрів (Pack), призначення гарячих клавіш (Hotkeys), керування кольорами міток (Label Colors) та імпорт субтитрів (SRT Import). Проведено функціональне та модульне тестування додатку. В роботі використано технологію CEP-розширень Adobe, HTML, CSS і JavaScript для інтерфейсу, CSInterface для зв'язку середовищ та ExtendScript для виконання команд у After Effects.

Сферою використання застосунку є автоматизація роботи motion-дизайнера та відеомонтажера у процесі підготовки композицій в Adobe After Effects.

КЛЮЧОВІ СЛОВА: ADOBE AFTER EFFECTS, CEP, EXTENDSCRIPT, CSINTERFACE, FAST PANEL, АВТОМАТИЗАЦІЯ, ШАРИ, SRT, СУБТИТРИ, MOTION DESIGN.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	9
ВСТУП	10
1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ	12
1.1 Характеристика задач motion-дизайнера в Adobe After Effects.....	12
1.2 Типові проблеми ручної роботи з шарами та субтитрами.....	14
1.3 Аналіз аналогів та обґрунтування власного рішення.....	15
1.4 Вимоги до програмного забезпечення	17
1.5 Огляд засобів автоматизації Adobe та вибір платформи реалізації	19
1.6 Висновки до розділу 1	20
2 ЗАСОБИ РЕАЛІЗАЦІЇ СЕР-ПАНЕЛІ FAST PANEL	21
2.1 Технологічна модель СЕР-розширення	21
2.2 HTML, CSS і JavaScript як основа інтерфейсу	22
2.3 ExtendScript та об'єктна модель After Effects	23
2.4 Збереження налаштувань і робота з локальними файлами	25
2.5 Структура проєкту, маніфест та засоби відлагодження.....	26
2.6 Модель подієвої взаємодії та зв'язок середовищ	27
2.7 Проєктування інтерфейсу панелі засобами Figma.....	28
2.8 Висновки до розділу 2	30
3 ПРОЄКТУВАННЯ СИСТЕМИ.....	32
3.1 Загальна архітектура FAST PANEL	32
3.2 Варіанти використання та сценарії роботи.....	33
3.3 Проєктування алгоритмів обробки шарів.....	35
3.4 Проєктування імпорту SRT-субтитрів	36
3.5 Проєктування структури даних та конфігурації команд.....	37
3.6 Проєктування перевірок вхідних даних та обробки помилок	38
3.7 Проєктування взаємодії інтерфейсу з ExtendScript	40
3.8 Проєктування операцій з композицією та шарами.....	41
3.9 Проєктування імпорту субтитрів як окремого сценарію	43

3.10	Проектування станів інтерфейсу та зручності використання	45
3.11	Проектування можливостей масштабування та супроводу	46
3.12	Висновки до розділу 3	47
4	РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ	51
4.1	Структура програмного проєкту	51
4.2	Реалізація інтерфейсу та взаємодії з користувачем	52
4.3	Тестування FAST PANEL	56
	ВИСНОВКИ	58
	ПЕРЕЛІК ПОСИЛАНЬ	60
	ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ	62
	ДОДАТОК Б. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ	69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Позначення	Пояснення
AE	Adobe After Effects - програмне середовище для motion-графіки та композитингу.
CEP	Common Extensibility Platform - платформа Adobe для HTML/JavaScript-розширень.
CSXS	Службова назва середовища виконання CEP, що використовується у manifest.xml.
JSX	Файл ExtendScript, який виконується у середовищі host-додатку Adobe.
DOM	Document Object Model; у роботі використовується як HTML DOM, так і об'єктна модель After Effects.
SRT	Текстовий формат субтитрів з часовими мітками початку і завершення репліки.
UI	User Interface - користувацький інтерфейс програмного забезпечення.
Undo group	Групування дій After Effects, яке дозволяє скасувати комплексну операцію одним викликом Undo.

ВСТУП

У процесі створення відеоконтенту Adobe After Effects часто використовується не тільки як середовище для творчого монтажу, а й як робочий інструмент з великою кількістю повторюваних технічних дій. Motion-дизайнер створює службові шари, вирівнює об'єкти, змінює часові межі, переносить точку прив'язки, готує текстові елементи і синхронізує їх з таймлайном. Кожна окрема операція може займати небагато часу, але в реальному проєкті їх кількість швидко зростає.

Актуальність теми пов'язана з потребою скоротити рутинну частину роботи у After Effects без втрати контролю над результатом. Стандартні меню програми мають широкий функціонал, проте вони не завжди зручні для коротких повторюваних сценаріїв. Для користувача важливо мати компактну панель, де найбільш потрібні команди зібрано в одному місці, а налаштування зберігаються між сеансами роботи.

FAST PANEL розроблено як CEP-розширення для Adobe After Effects. Такий підхід дозволяє поєднати HTML/CSS-інтерфейс, клієнтську логіку JavaScript і виконання команд через ExtendScript. Панель працює всередині After Effects, а тому користувач не переходить до зовнішньої програми і не порушує звичний workflow.

Об'єктом дослідження є процес автоматизації операцій з шарами, композиціями та субтитрами у Adobe After Effects. Предметом дослідження є програмне забезпечення FAST PANEL, яке реалізує створення типових шарів, зміну anchor point, керування таймінгом, hotkeys, label colors і імпорт SRT-файлів.

Метою роботи є підвищення ефективності роботи користувача Adobe After Effects за рахунок розробки CEP-панелі FAST PANEL, що об'єднує типові команди motion-дизайнера в одному інтерфейсі та автоматизує дії, які зазвичай виконуються вручну.

Для досягнення мети необхідно проаналізувати предметну галузь, визначити вимоги до програмного забезпечення, дослідити засоби створення CEP-розширень,

спроектувати архітектуру панелі, реалізувати її основні модулі, перевірити роботу в практичних сценаріях і підготувати демонстраційні матеріали.

Практична цінність роботи полягає у зменшенні кількості ручних дій під час підготовки композицій. Наприклад, створення шару з потрібним типом, кольором, тривалістю, режимом 3D і parent-зв'язком може виконуватися одним натисканням кнопки. Аналогічно, імпорт SRT-файлу перетворює набір субтитрів на окремі текстові шари з готовими часовими межами.

Під час підготовки роботи використано матеріали переддипломної практики, програмний код FAST PANEL, екранні форми розробленої панелі, а також документацію Adobe CEP, After Effects Scripting і довідкові матеріали MDN щодо Web API.

Додатково під час роботи було враховано особливості реального робочого процесу motion-дизайнера, де важливими є не лише швидкість виконання окремої команди, а й стабільність результату, повторюваність дій та можливість швидкого повернення до попереднього стану проєкту. Саме тому в FAST PANEL передбачено логіку перевірки активної композиції, групування операцій для скасування через Undo, а також збереження користувацьких налаштувань між сеансами роботи.

Розроблена панель орієнтована на практичне використання у щоденній роботі з After Effects. Вона не замінює стандартні інструменти програми, а доповнює їх, виносячи найбільш уживані дії в окремий компактний інтерфейс. Такий підхід дозволяє зменшити кількість переходів між меню, скоротити час підготовки композиції та знизити ймовірність помилок під час виконання однотипних операцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Характеристика задач motion-дизайнера в Adobe After Effects

After Effects використовується для створення анімаційної графіки, композитингу, титрів, рекламних вставок, відеопрезентацій та шаблонів для регулярного контенту. У таких задачах важливе не лише художнє рішення, а й організація технічної роботи: структура композиції, порядок шарів, часові межі, прив'язки, батьківські зв'язки та повторювані налаштування.

Сфера застосування After Effects охоплює рекламні ролики, анімовані заставки, оформлення відео для соціальних мереж, інфографіку та субтитрування. У більшості цих напрямів дизайнер працює з типовими структурами: фоновий шар, група текстових шарів, керувальні Null-об'єкти та коригувальні шари з ефектами. Підготовка такої структури повторюється від проєкту до проєкту і займає помітну частину робочого часу.

Робочий процес motion-дизайнера складається з великої кількості малих дій. Частина з них є творчою і залежить від задуму проєкту, але значна частина має технічний характер: створити Null, додати Adjustment Layer, підготувати Solid, швидко поставити текстовий шар, змінити anchor point або підлаштувати шар під тривалість композиції. Саме ця частина добре піддається автоматизації.

Особливість After Effects полягає у шаровій моделі композиції. Кожен шар має часові межі, трансформації, властивості, ефекти та може бути пов'язаний з іншими шарами. Якщо користувач часто повторює однакові кроки для різних шарів, то навіть невелике скорочення кількості кліків дає відчутний результат.

Зі збільшенням кількості шарів зростає навантаження на дизайнера: потрібно пам'ятати порядок шарів, їхні часові межі та зв'язки. Перемикання між мишею і клавіатурою, пошук потрібної команди в меню та повторне налаштування однакових параметрів уповільнюють роботу. Тому інструменти, що скорочують

кількість дій і виносять часті команди на окрему панель, дають прямий виграш у продуктивності.

Якщо оцінити робочий день дизайнера, то на технічну підготовку — створення службових шарів, вирівнювання, підгонку меж і повторне введення однакових налаштувань — припадає істотна частка операцій. Автоматизація навіть кількох таких дій за одне натискання накопичує помітну економію часу впродовж проєкту.

Під час переддипломної практики було визначено, що найбільший практичний інтерес мають операції створення технічних шарів, зміни точки прив'язки без зсуву зображення, підгонки таймінгу, пакування keyframes та перетворення SRT-субтитрів у текстові шари. Ці операції стали основою функціонального складу FAST PANEL.

Таблиця 1.1

Типові задачі користувача After Effects

Група задач	Приклади дій	Причина автоматизації
Створення шарів	Solid, Text, Null, Adjustment, Shape, Camera, Light	Команди розташовані у різних меню, а налаштування часто повторюються.
Робота з таймінгом	Trim to Layer, Fit to Comp, Pack keyframes	Ручне редагування inPoint/outPoint займає час і створює ризик помилки.
Точка прив'язки	9 позицій anchor point	Звичайна зміна anchor point може змістити шар у кадрі.
Субтитри	Імпорт SRT, No gaps, очищення пунктуації	Ручне створення десятків текстових шарів є повільним і монотонним.
Персоналізація	Hotkeys, label colors, стан секцій	Користувачу потрібне збереження власного способу роботи.

1.2 Типові проблеми ручної роботи з шарами та субтитрами

Основна проблема ручної роботи полягає у фрагментованості інтерфейсу. Команди створення різних типів шарів, зміни тривалості, вирівнювання і налаштування властивостей не зібрані в одній панелі. Користувач постійно переходить між меню, контекстними командами, timeline і вікном композиції.

Друга проблема пов'язана з часовими межами шарів. У типових задачах дизайнеру потрібно швидко створити новий шар точно в межах вибраного шару або розтягнути групу шарів до тривалості композиції. Якщо робити це вручну, доводиться контролювати початок, кінець і порядок шарів, а це збільшує кількість помилок.

Постійне перемикання контексту між timeline, вікном композиції та меню також знижує концентрацію. Кожен перехід вимагає від користувача знайти потрібний елемент інтерфейсу, що відволікає від творчої частини роботи. Зведення часто вживаних команд в одну панель зменшує цей розсіяний фокус.

Накопичення дрібних затримок особливо помітне у проектах із жорсткими дедлайнами, де той самий набір підготовчих дій повторюється для багатьох композицій. У таких умовах навіть незначне пришвидшення рутинних операцій впливає на загальний темп роботи.

Третя проблема виникає під час зміни anchor point. Для текстових і shape-шарів точка прив'язки часто має стояти у центрі, у куті або на середині сторони. Якщо змінити anchor point без компенсації координат position, візуальне положення шару зміниться. Тому алгоритм повинен одночасно обчислювати нову точку прив'язки і компенсувати позицію.

Окрема група проблем стосується субтитрів. Формат SRT зручний для зберігання тексту і таймкодів, але After Effects не перетворює такий файл у готовий набір текстових шарів стандартним коротким сценарієм. Якщо субтитрів багато, ручне копіювання тексту і часу стає непродуктивним.

Ручне перенесення субтитрів складається з кількох однотипних кроків: відкрити файл, скопіювати текст блоку, створити текстовий шар, виставити його початок і кінець за таймкодом, повторити для кожного блоку. За десятки реплік така робота стає джерелом помилок у таймінгу і потребує значного часу.

FAST PANEL орієнтовано саме на ці проблеми. Панель не намагається замінити After Effects, а створює швидкий шар між користувачем і найчастішими командами. Такий підхід дає змогу залишити творчі рішення дизайнеру, а технічні кроки виконувати автоматизовано.

1.3 Аналіз аналогів та обґрунтування власного рішення

Під час аналізу були розглянуті стандартні засоби After Effects, окремі JSX-скрипти і сторонні панелі. Стандартні засоби є стабільними та не потребують встановлення, однак вони не забезпечують компактного сценарію для всіх потрібних дій. Окремі скрипти можуть швидко вирішувати одну задачу, але не формують цілісний інтерфейс для щоденної роботи.

Стандартні пресети та функції After Effects, як-от Create Nulls From Paths чи команди вирівнювання, покривають окремі задачі, але вони розосереджені інтерфейсом і не налаштовуються під конкретний робочий процес. Користувач все одно змушений запам'ятовувати, де саме розташована кожна команда.

Сторонні панелі на кшталт Animation Composer, Motion Tools або AEUX частково перекривають задачі автоматизації. Проте їхня логіка орієнтована на власний набір сценаріїв і не завжди враховує локальні вимоги: створення конкретних типів шарів, власні label colors, власні hotkeys, імпорт SRT і просте розширення JSX-коду.

Animation Composer пропонує бібліотеку готових пресетів і перехідних анімацій; Motion Tools та Motion Bro зосереджені на швидкому застосуванні ефектів і шаблонів, а AEUX забезпечує обмін макетами між Figma і After Effects. Спільним обмеженням цих рішень є закритий код і орієнтація на власні сценарії,

через що їх важко адаптувати під специфічний набір дрібних технічних операцій конкретного дизайнера.

Для порівняння підходів обрано критерії, що відображають щоденні потреби: наявність єдиного інтерфейсу, набір операцій над шарами, можливість локального налаштування, підтримку імпорту субтитрів та відкритість для розширення. За цими критеріями власне рішення зіставлено з наявними засобами (таблиця 1.2).

Власне рішення FAST PANEL обґрунтоване потребою створити невелику, контрольовану і зрозумілу панель під конкретний workflow. Код розміщується у каталозі CEP-розширення, має чіткий поділ на інтерфейс, клієнтську логіку та ExtendScript-команди, а функції можуть доповнюватися без зміни всієї архітектури.

Таблиця 1.2

Порівняння підходів до автоматизації

Критерій	Нативні засоби АЕ	Animation Composer	Motion Tools	Motion / Bro	FAST PANEL
Створення шарів	ручне створення через меню	готові пресети, не фокус на шарах	допоміжні дії з шарами	пресети та шаблони	Solid, Text, Null, Adj, Camera, Light
Trim / Fit / Pack	можливо вручну на timeline	підтримується частково через пресети	є інструмент и таймінгу	переважно через шаблони	Trim to Layer, Fit to Comp, Pack Keyframes
Hotkeys	стандартні комбінації АЕ	обмежено діями плагіна	є швидкі кнопки	не є основною функцією	власні hotkeys у панелі

Порівняння підходів до автоматизації

SRT-субтитри	немає прямого імпорту SRT	не підтримує	не підтримує	не підтримує	парсинг .srt і створення Text Layers
Власний workflow	низька: багато ручних дій	середня: готові пресети	середня: утиліти workflow	середня: бібліотеки шаблонів	висока: функції під задачу

1.4 Вимоги до програмного забезпечення

Функціональні вимоги до FAST PANEL сформовано на основі практичних сценаріїв роботи. Панель повинна створювати основні типи шарів, розміщувати новий шар відносно виділеного шару, підтримувати Trim to Layer, змінювати anchor point у дев'яти позиціях, керувати label colors, призначати hotkeys та імпортувати SRT-субтитри.

До нефункціональних вимог належать робота всередині Adobe After Effects як CEP-панель, зрозумілий темний інтерфейс, збереження налаштувань у localStorage, коректна обробка відсутньої активної композиції, використання undo groups і можливість розширення коду без перебудови всієї системи.

Важливою вимогою є передбачуваність результату. Якщо користувач натискає кнопку створення шару, система повинна перевірити активну композицію, створити шар, застосувати опції, розмістити його у правильному порядку і залишити можливість скасувати дію одним Undo. Така поведінка робить панель безпечною для реального проєкту.

Окремо враховано обмеження середовища. Панель працює лише за наявності встановленого After Effects із підтримкою CEP, а виконання непідписаного

розширення потребує режиму PlayerDebugMode. Ці умови прийнятні для інструмента, що розробляється під власний робочий процес.

Зручність використання також віднесено до вимог: керування має бути зосереджене в одній панелі, а підписи й елементи — зрозумілими без додаткового навчання. Це знижує поріг входження і робить інструмент придатним для щоденного застосування.

Таблиця 1.3

Функціональні та нефункціональні вимоги

Тип вимоги	Вимога	Очікуваний результат
Функціональна	Створення Solid, Text, Null, Adjustment, Shape, Camera, Light	Користувач отримує готовий шар без переходу до меню After Effects.
Функціональна	Trim to Layer та Fit to Comp	Часові межі шару задаються автоматично.
Функціональна	Anchor Point 3x3	Точка прив'язки змінюється без візуального зміщення шару.
Функціональна	SRT IMPORT	Субтитри перетворюються на текстові шари з таймінгом.
Нефункціональна	localStorage	Налаштування користувача відновлюються після повторного запуску панелі.
Нефункціональна	Undo groups	Комплексна дія скасовується одним кроком.

1.5 Огляд засобів автоматизації Adobe та вибір платформи реалізації

Adobe пропонує кілька технологій для розширення After Effects. Найдавнішою є скриптинг мовою ExtendScript: сценарії запускаються через меню File > Scripts і автоматизують дії над проєктом, але не мають постійного інтерфейсу. ScriptUI дає змогу будувати прості панелі засобами самого ExtendScript, проте можливості оформлення таких панелей обмежені [11].

Сучаснішим підходом є CEP (Common Extensibility Platform) — платформа, що дозволяє створювати панелі на веб-технологіях HTML, CSS і JavaScript із доступом до об'єктної моделі програми через ExtendScript. CEP-панель вбудовується в інтерфейс After Effects як повноцінне docking-вікно і підтримує звичні засоби фронтенд-розробки [1].

Adobe також розвиває новішу платформу UXP, яка поступово замінює CEP у частині застосунків. Проте на момент розробки FAST PANEL підтримка UXP в After Effects обмежена, тоді як CEP лишається стабільним і документованим засобом із широкою базою прикладів [5].

Для реалізації обрано CEP, оскільки він поєднує гнучкість веб-інтерфейсу з повним доступом до можливостей After Effects, не потребує платних ліцензій і дозволяє нарощувати функціональність окремими ExtendScript-функціями. Такий вибір узгоджується з метою створити компактний, відкритий для розширення інструмент під конкретний workflow.

Важливою перевагою CEP є відкритість коду розширення: усі файли — розмітка, стилі, клієнтська логіка та ExtendScript — зберігаються локально у каталозі розширення і доступні для редагування. Це дозволяє додавати нові команди або змінювати наявні без перекомпіляції та без залежності від стороннього постачальника.

Платформа CEP має кілька версій (CSXS), що відповідають різним випускам Creative Cloud і визначають доступну версію вбудованого рушія Chromium. FAST PANEL орієнтовано на діапазон версій, який охоплює актуальні випуски After

Effects; цей діапазон задається у файлі manifest.xml і забезпечує сумісність із поширеними конфігураціями.

Таблиця 1.4

Порівняння технологій розширення Adobe

Платформа	Технологія	Перевага	Обмеження
ExtendScript-скрипт	ExtendScript	Прямий доступ до АЕ, без встановлення	Немає постійного інтерфейсу
ScriptUI-панель	ExtendScript	Власне вікно в АЕ	Обмежені засоби оформлення
СЕР	HTML/CSS/JS + ExtendScript	Веб-інтерфейс і повний доступ до АЕ	Потрібен PlayerDebugMode для непідписаних
UXP	HTML/JS + UXP API	Сучасна платформа Adobe	Обмежена підтримка в After Effects

1.6 Висновки до розділу 1

У розділі проаналізовано предметну галузь — задачі motion-дизайнера в After Effects і проблеми ручного виконання повторюваних технічних операцій. Визначено, що найбільший практичний ефект дає автоматизація створення технічних шарів, керування таймінгом, пакування ключових кадрів та імпорту субтитрів.

Порівняння наявних засобів показало, що стандартні функції та сторонні панелі не покривають потрібний набір дрібних операцій у єдиному, налаштовуваному інтерфейсі. На підставі цього сформульовано вимоги до власного рішення та обґрунтовано вибір платформи СЕР. Це визначає завдання наступних розділів — проєктування та реалізацію панелі FAST PANEL.

2 ЗАСОБИ РЕАЛІЗАЦІЇ CEP-ПАНЕЛІ FAST PANEL

2.1 Технологічна модель CEP-розширення

Розділ присвячено засобам, застосованим під час розробки FAST PANEL. Розглянуто технології інтерфейсу, мову доступу до After Effects, засоби збереження стану, структуру проєкту та інструмент проєктування дизайну. Усі описані засоби безпосередньо використано у реалізації панелі.

CEP-розширення Adobe складається з HTML-інтерфейсу, JavaScript-коду, файлу manifest.xml і, за потреби, ExtendScript-файлів. У FAST PANEL manifest.xml описує ідентифікатор розширення, host-додаток AEFT, мінімальні та максимальні розміри панелі, головний HTML-файл і шлях до JSX-скрипта.

Документація Adobe CEP зазначає, що HTML/JavaScript-розширення мають окремий JavaScript-двигун, тоді як доступ до об'єктної моделі host-додатку виконується через ExtendScript. Для зв'язку між цими середовищами використовується CSInterface.evalScript [1]. Саме ця модель застосована у FAST PANEL.

Альтернативою CEP є виконання окремих скриптів через меню File > Scripts, проте такий підхід не дає постійного графічного інтерфейсу. CEP обрано саме тому, що він поєднує звичні веб-технології для побудови панелі з повним доступом до можливостей After Effects через ExtendScript, лишаючись доступним і безкоштовним засобом розробки [5].

У практичній реалізації користувач взаємодіє з HTML-кнопками, main.js читає стан інтерфейсу, формує рядок JSX-виклику і передає його до After Effects через CSInterface. Функції hostscript.jsx виконують зміни в активній композиції, використовуючи об'єкти CompItem, Layer, властивості transform і методи LayerCollection.

Таблиця 2.1

Призначення основних технологій

Технологія	Роль у FAST PANEL	Приклад використання
HTML5	Структура інтерфейсу панелі	select layer-type, buttons ADD/FIT/PACK, input для SRT
CSS3	Оформлення темної панелі	сітки кнопок, toggle, collapsible-секції
JavaScript	Події інтерфейсу і стан користувача	addEventListener, localStorage, FileReader
CSInterface.js	Міст між HTML-панеллю і host-додатком	evalScript для виклику JSX-функцій
ExtendScript	Керування After Effects	створення шарів, зміна anchorPoint, importSRT
manifest.xml	Опис CEP-розширення	ExtensionBundleId, Host AEFT, Geometry

2.2 HTML, CSS і JavaScript як основа інтерфейсу

HTML у FAST PANEL виконує роль мови розмітки інтерфейсу. У файлі index.html описано структуру панелі: елементи select і button для команд, поля input для гарячих клавіш, а також контейнери секцій. HTML надає й елемент input з типом file, який є стандартним засобом вибору локального SRT-файлу безпосередньо в інтерфейсі панелі.

CSS застосовується як засіб оформлення інтерфейсу. Файл style.css задає темну палітру з бірюзовими акцентами, розміри елементів і компонування секцій. Вибір темного оформлення зумовлений тим, що After Effects також використовує темний інтерфейс, тому панель візуально узгоджується з робочим середовищем.

JavaScript у файлі `main.js` є засобом клієнтської логіки панелі. Він звертається до елементів інтерфейсу через DOM API (`document.getElementById`, `querySelector`), підписується на події через `addEventListener` [8] і формує рядки виклику `ExtendScript`-функцій. JavaScript також читає локальний файл субтитрів і перетворює його у структуру даних перед передаванням у `host`-середовище.

Для збереження стану інтерфейсу використовується Web Storage API, а саме об'єкт `localStorage`. Він зберігає призначені гарячі клавіші, стан перемикача `Trim to Layer`, режим 3D, `parent`-зв'язок і стан згортання секцій. Це стандартний браузерний засіб, який не потребує зовнішнього сховища і доступний у середовищі CEP [6].

Поділ за призначенням є базовим принципом цього стека: HTML описує структуру, CSS відповідає за вигляд, а JavaScript — за поведінку. Такий поділ полегшує супровід коду, оскільки зміна оформлення не зачіпає логіку, а зміна логіки не порушує розмітку.

JavaScript у CEP виконується у вбудованому браузерному рушії Chromium, тому панелі доступні сучасні можливості мови та DOM. Це дозволяє застосовувати стандартні підходи фронтенд-розробки — обробку подій, роботу з колекціями елементів і асинхронне читання файлів — без додаткових бібліотек [10].

2.3 ExtendScript та об'єктна модель After Effects

`ExtendScript` є основним засобом доступу до функціональності After Effects. Це діалект ECMAScript [12], розширений об'єктною моделлю програми. У FAST PANEL `ExtendScript`-код розміщено у файлі `hostscript.jsx`, а його функції викликаються з `main.js` через `CSInterface.evalScript`. Кожна операція, що змінює композицію, обгортається у `app.beginUndoGroup` та `app.endUndoGroup`, завдяки чому дію можна скасувати одним `Undo`.

Під час створення шарів використовується активна композиція `app.project.activeItem`. Якщо активного `CompItem` немає, користувач отримує повідомлення про помилку. Така перевірка запобігає виконанню команд у неправильному контексті та робить поведінку панелі зрозумілою.

Робота з композицією спирається на об'єктну модель After Effects. Активна композиція отримується через `app.project.activeItem` як об'єкт `CompItem`, а шари створюються методами колекції `LayerCollection`: `addSolid`, `addNull`, `addText`, `addCamera` і `addLight`. Для керування часовими межами використовуються властивості `startTime`, `inPoint` та `outPoint`, а для визначення геометрії вмісту шару — метод `sourceRectAtTime`.

На основі цих засобів реалізовано службові функції панелі. Функція `trimLayerToSelected` визначає референсний шар і вирівнює часові межі нового шару за ним. Функції `alignLayers` і `distributeLayers` обчислюють межі шарів через `valueAtTime` і `sourceRectAtTime` та змінюють властивість `position`. Функція `fitToComp` розтягує шар на всю тривалість композиції.

Окремим засобом є робота з ключовими кадрами. Функція `packLayers` рекурсивно обходить властивості шару і використовує методи `keyframe-інтерфейсу`: `keyTime`, `keyValue`, `addKey`, `removeKey` та `setValueAtKey`, а також `setInterpolationTypeAtKey` і `setTemporalEaseAtKey` для збереження типу інтерполяції та згладжування. Це дозволяє масштабувати анімацію у нові часові межі без втрати її форми.

На відміну від клієнтського JavaScript, `ExtendScript` базується на ранній версії ECMAScript і не має доступу до DOM чи браузерних API. Натомість він працює з об'єктною моделлю After Effects, де кореневим є об'єкт `app`, а композиції та шари подаються об'єктами `Item` і `Layer`. Це визначає стиль коду у `hostscript.jsx`: послідовні виклики методів об'єктної моделі без асинхронних конструкцій.

Для діагностики `ExtendScript`-коду застосовуються прості засоби — повернення рядків-результатів у JavaScript і повідомлення про помилки через діалог `alert`.

Кожна функція повертає текстовий статус, який панель показує користувачу, що спрощує виявлення некоректного контексту виконання, наприклад відсутньої активної композиції.

Основні засоби об'єктної моделі After Effects

Засіб об'єктної моделі	Призначення
app.project.activeItem	Отримання активної композиції (CompItem)
LayerCollection.addSolid / addNull / addText	Створення шарів відповідного типу
LayerCollection.addCamera / addLight	Створення камери та джерела світла
Layer.sourceRectAtTime	Визначення меж вмісту шару
Property.valueAtTime	Отримання значення властивості в заданий момент часу
Property.addKey / removeKey / setValueAtKey	Керування ключовими кадрами анімації
app.beginUndoGroup / endUndoGroup	Групування операцій для скасування однією дією Undo

2.4 Збереження налаштувань і робота з локальними файлами

У FAST PANEL локальний стан зберігається у localStorage у вигляді ключів fastPanelHotkeys, fastPanelSnap, fastPanel3D, fastPanelParent та службових ключів згортання секцій fastPanelHotkeysCollapsed і fastPanelSRTCollapsed. Цього засобу достатньо для панелі, оскільки дані мають невеликий обсяг і не потребують серверної синхронізації.

Для імпорту субтитрів використовується HTML input type=file з обмеженням accept=.srt. Після вибору файлу FileReader читає текст, parseSRT розбиває його на блоки, перетворює timescode у секунди і формує масив об'єктів. Далі дані серіалізуються у JSON і передаються в ExtendScript.

Формат JSON відіграє роль спільної мови між середовищами. JavaScript серіалізує масив субтитрів у рядок JSON, передає його одним викликом evalScript, а ExtendScript розбирає рядок назад у структуру даних. Такий підхід уникає

передавання складних об'єктів через межу середовищ і робить обмін даними передбачуваним.

Такий поділ відповідальності є практичним: JavaScript зручний для роботи з DOM, файлами користувача та JSON, а ExtendScript потрібен для створення шарів у After Effects. Межа між середовищами проходить через CSInterface.evalScript, тому структура даних має бути простою і безпечною для передавання.

Додатково такий механізм збереження налаштувань підвищує зручність повторного використання панелі. Після перезапуску After Effects користувач не повинен заново налаштовувати основні параметри роботи, оскільки вони автоматично відновлюються з локального сховища. Це стосується як гарячих клавіш, так і стану перемикачів або згорнутих секцій інтерфейсу.

Робота з локальними файлами також робить панель незалежною від зовнішніх сервісів. Для імпорту SRT-файлу не потрібне підключення до інтернету або додаткове програмне забезпечення. Користувач вибирає файл на власному комп'ютері, після чого FAST PANEL обробляє його вміст і передає підготовлені дані до After Effects для створення текстових шарів.

Отже, використання localStorage, FileReader і JSON забезпечує просту та надійну модель роботи з налаштуваннями і локальними даними. Ці засоби відповідають задачам FAST PANEL, оскільки дозволяють зберігати персональні параметри користувача, читати файли субтитрів і передавати дані між JavaScript та ExtendScript без ускладнення структури проєкту.

2.5 Структура проєкту, маніфест та засоби відлагодження

Проєкт FAST PANEL організовано за стандартом CEP-розширення. Каталог CSXS містить файл manifest.xml, каталог css — стилі, каталог js — клієнтську логіку та бібліотеку CSInterface.js, каталог jsx — ExtendScript-код, а в корені розміщено головний файл index.html. Такий поділ є вимогою середовища CEP і дозволяє After Effects коректно виявити та завантажити розширення.

Бібліотека `CSInterface.js`, що постачається разом із CEP, є обов'язковим засобом проєкту. Вона надає клас `CSInterface` з методами `evalScript`, `getHostEnvironment` та керування подіями розширення, інкапсулюючи деталі зв'язку з host-додатком. Підключення цієї бібліотеки у `index.html` робить доступним єдиний канал виклику `ExtendScript`-функцій.

Файл `manifest.xml` є засобом опису розширення для host-додатку. Він задає ідентифікатор `ExtensionBundleId` `com.fastpanel.extension`, host `AEFT` з діапазоном підтримуваних версій, тип `Panel`, назву пункту меню та геометрію панелі — мінімальні, максимальні й типові розміри. У ньому також вказано головний HTML-файл і шлях до JSX-скрипта [2].

Для запуску непідписаного розширення під час розробки використовується режим `PlayerDebugMode`. Файл `enable_debug.reg` вносить до реєстру Windows значення `PlayerDebugMode=1` для гілок `CSXS.11–CSXS.17`, що відповідають версіям середовища CEP. Без цього режиму `After Effects` відмовляється завантажувати локально встановлене розширення.

Для відлагодження клієнтської частини CEP підтримує віддалену діагностику через інструменти розробника `Chromium`. Завдяки цьому розмітку, стилі та виконання `JavaScript` можна перевіряти у звичному середовищі `DevTools`, а повідомлення `console.log` використовувати для трасування логіки під час розробки [1].

Встановлення автоматизовано засобами командного рядка. Сценарій `install.bat` створює каталог розширення у `Common Files\Adobe\CEP\extensions`, копіює файли проєкту командою `xcopy` і застосовує `enable_debug.reg`. Сценарій `sync_update.bat` використовується для швидкого оновлення вже встановленого розширення під час розробки, копіюючи лише змінені каталоги.

2.6 Модель подієвої взаємодії та зв'язок середовищ

Взаємодія в `FAST PANEL` побудована на подієвій моделі браузерного середовища. Елементи керування реагують на події `click`, `change` і `keydown`, які

обробляються через `addEventListener`. Обробник події читає поточний стан інтерфейсу — вибраний тип шару, колір, опції `Trim`, `3D` і `parent` — і формує рядок виклику відповідної `ExtendScript`-функції.

Окремою підсистемою подієвої моделі є гарячі клавіші. Панель зберігає відповідність між сполученнями клавіш і командами, перехоплює подію `keydown`, формує нормалізований код сполучення і викликає прив'язану дію. Це дозволяє виконувати часті операції без звертання до миші.

Зв'язок між JavaScript-середовищем панелі та середовищем `After Effects` забезпечує об'єкт `CSInterface`. Метод `evalScript` передає сформований рядок коду у `host`-додаток і за потреби отримує результат через зворотний виклик. Саме через цю межу проходить уся взаємодія, тому дані передаються у простому вигляді — рядками та `JSON`.

Для функцій, що повертають значення (наприклад, `importSRT`), використовується `callback` методу `evalScript`. `ExtendScript` повертає рядок результату, а JavaScript аналізує його і оновлює статус в інтерфейсі. Такий механізм робить зв'язок між середовищами асинхронним і безпечним для передавання структур даних.

Для зворотного зв'язку з користувачем застосовується коротке візуальне підсвічування елемента (`flash`) після виконання команди. Такий механізм підтверджує, що дію оброблено, і робить роботу з панеллю передбачуванішою.

2.7 Проєктування інтерфейсу панелі засобами Figma

Перед написанням HTML- і CSS-коду інтерфейс `FAST PANEL` спроєктовано у `Figma` — хмарному векторному редакторі для дизайну інтерфейсів. `Figma` використано як засіб попереднього моделювання зовнішнього вигляду панелі, що дало змогу визначити структуру, розміри та оформлення елементів ще до етапу програмування [25].

У Figma побудовано макет панелі з поділом на функціональні секції: блок створення шарів, перемикач Trim to Layer, інструменти вирівнювання, секцію гарячих клавіш та секцію імпорту субтитрів. Для кожної секції задано відступи, типорозміри кнопок і полів та сітку компоновання, що відповідає вузькій вертикальній панелі After Effects.

Колірну схему також підібрано у Figma. Темну палітру з бірюзовими акцентами обрано так, щоб панель візуально узгоджувалася з інтерфейсом After Effects. Значення кольорів і розмірів, визначені в макеті, згодом перенесено безпосередньо у файл style.css, що зменшило кількість правок під час верстки.

Figma підтримує компонентний підхід та auto-layout, тому повторювані елементи — кнопки, випадні списки та поля вводу — оформлено як компоненти з єдиним стилем. Це забезпечило візуальну однорідність інтерфейсу й полегшило відтворення макета засобами HTML і CSS.

Інтерактивний режим Figma дав змогу змоделювати поведінку секцій, що згортаються, і стани наведення курсора ще до реалізації. Готовий макет слугував специфікацією інтерфейсу: під час верстки HTML-структура та властивості CSS відтворювали затверджений дизайн, завдяки чому скорочено доопрацювання на пізніх етапах.

У макеті застосовано систему відступів і типорозмірів (design tokens), що задає узгоджені значення полів, проміжків і кеглів. Єдина шкала розмірів спростила перенесення параметрів у CSS і забезпечила рівномірне компоновання секцій незалежно від їхнього вмісту.

Режим інспектора Figma дав змогу зчитувати точні значення кольорів, розмірів і відступів безпосередньо з елементів макета. Ці значення використано як орієнтир під час написання style.css, що зменшило розбіжності між задуманим дизайном і кінцевою версткою.

Іконки та графічні елементи панелі підготовлено у векторному форматі, що забезпечує чіткість за різних масштабів інтерфейсу. Оскільки панель має фіксовану вузьку ширину, у макеті заздалегідь перевірено читабельність підписів і доступність елементів керування у компактному компонованні.

Використання Figma як окремого етапу проєктування відокремило роботу над дизайном від написання коду. Це дало змогу узгодити вигляд панелі до реалізації та зменшити кількість змін інтерфейсу на етапі програмування, коли правки коштують дорожче.

Таблиця 2.3

Засоби, застосовані під час розробки FAST PANEL

Засіб	Роль у розробці
HTML / CSS	Структура та оформлення інтерфейсу панелі
JavaScript (main.js)	Клієнтська логіка, обробка подій, читання SRT
CSInterface.js	Зв'язок панелі з After Effects через evalScript
ExtendScript (hostscript.jsx)	Команди над композицією та шарами
manifest.xml	Опис розширення для середовища CEP
Figma	Проектування макета та оформлення інтерфейсу
Chromium DevTools	Відлагодження клієнтської частини панелі

2.8 Висновки до розділу 2

У розділі розглянуто засоби реалізації CEP-панелі FAST PANEL. Інтерфейс спроектовано у Figma та реалізовано засобами HTML, CSS і JavaScript; доступ до функціональності After Effects забезпечено через ExtendScript і об'єктну модель програми, а зв'язок між середовищами — через CSInterface.evalScript. Збереження налаштувань реалізовано засобами Web Storage API, структуру проєкту організовано за стандартом CEP, а відлагодження виконано в режимі PlayerDebugMode.

Поєднання цих засобів утворює цілісний інструментальний стек — від проєктування інтерфейсу до виконання команд у композиції. Обрані технології є стандартними для розробки розширень Adobe, що робить рішення відтворюваним і придатним для подальшого розвитку.

Загалом обрані засоби покривають усі етапи створення панелі — від макета інтерфейсу до виконання команд у композиції — і не потребують додаткових платних компонентів. Це підтверджує практичність вибраного стека для задач автоматизації роботи motion-дизайнера та можливість його повторного використання в подібних проєктах.

Крім того, чіткий поділ проєкту на рівні інтерфейсу, клієнтської логіки та ExtendScript спрощує супровід коду: кожен рівень можна змінювати окремо, не зачіпаючи інші. Це створює основу для наступного етапу роботи — детального проєктування архітектури та алгоритмів панелі, що розглядається у третьому розділі.

3 ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Загальна архітектура FAST PANEL

Архітектуру FAST PANEL спроектовано як тришарову взаємодію: інтерфейс користувача, клієнтська логіка панелі та виконання команд у After Effects. Інтерфейс не звертається безпосередньо до композиції, а лише формує події. JavaScript-код інтерпретує ці події, готує параметри і передає команду у JSX-рівень.

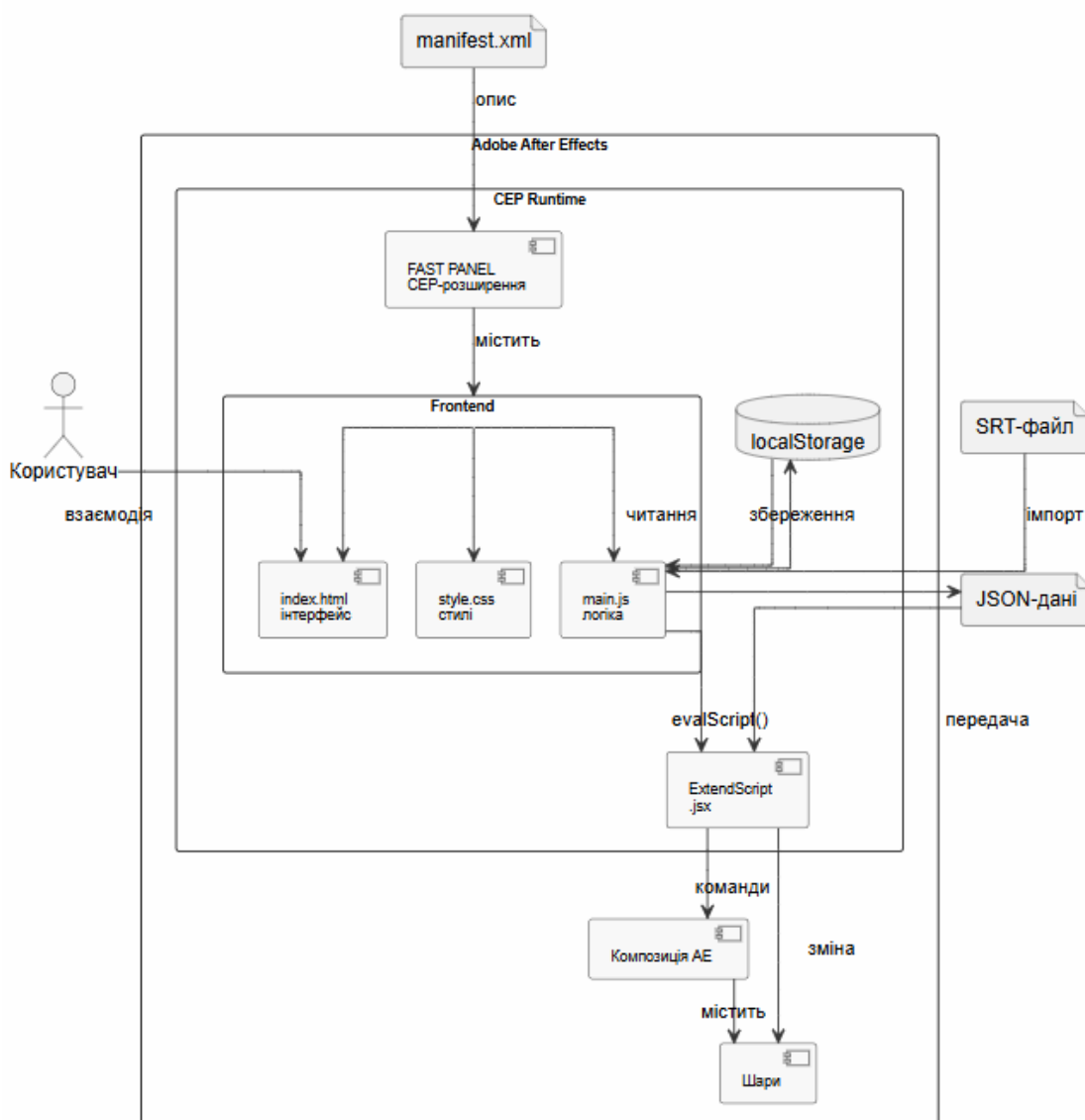


Рис. 3.1 Архітектура CEP-розширення FAST PANEL

Такий поділ зменшує зв'язаність компонентів. HTML і CSS можна змінювати без переписування алгоритмів роботи з шарами, а ExtendScript-функції можна доповнювати, не порушуючи структуру інтерфейсу. Це важливо для розширення панелі новими командами, наприклад align/distribute або додатковими режимами імпорту.

3.2 Варіанти використання та сценарії роботи

Основним актором системи є motion-дизайнер або відеомонтажер, який працює з активною композицією. Користувач може створювати службові шари, змінювати anchor point, налаштовувати hotkeys, імпортувати субтитри, підганяти часові межі шарів і очищувати ефекти.

Сценарій створення шару починається з вибору типу layer-type або натискання швидкої кнопки. Панель читає опції Trim to Layer, 3D, Set parent і label color. Після цього main.js формує виклик відповідної функції hostscript.jsx. Якщо активна композиція є валідною, новий шар створюється і налаштовується.

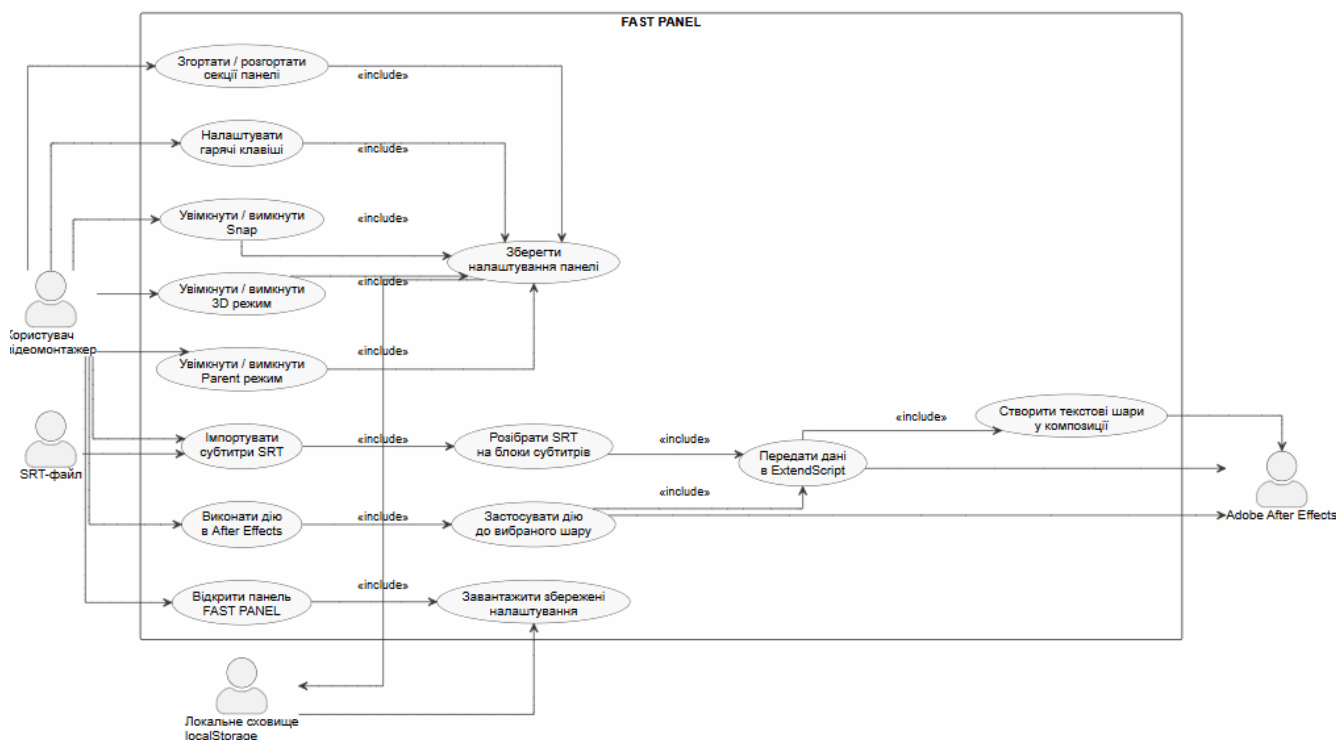


Рис. 3.2 Діаграма варіантів використання FAST PANEL

Сценарій імпорту SRT починається з вибору файлу. Панель показує ім'я файлу, кількість розпізнаних субтитрів і дозволяє ввімкнути No gaps або Strip punctuation. Після натискання Create Subtitles кожен субтитр перетворюється на текстовий шар із власними inPoint та outPoint.

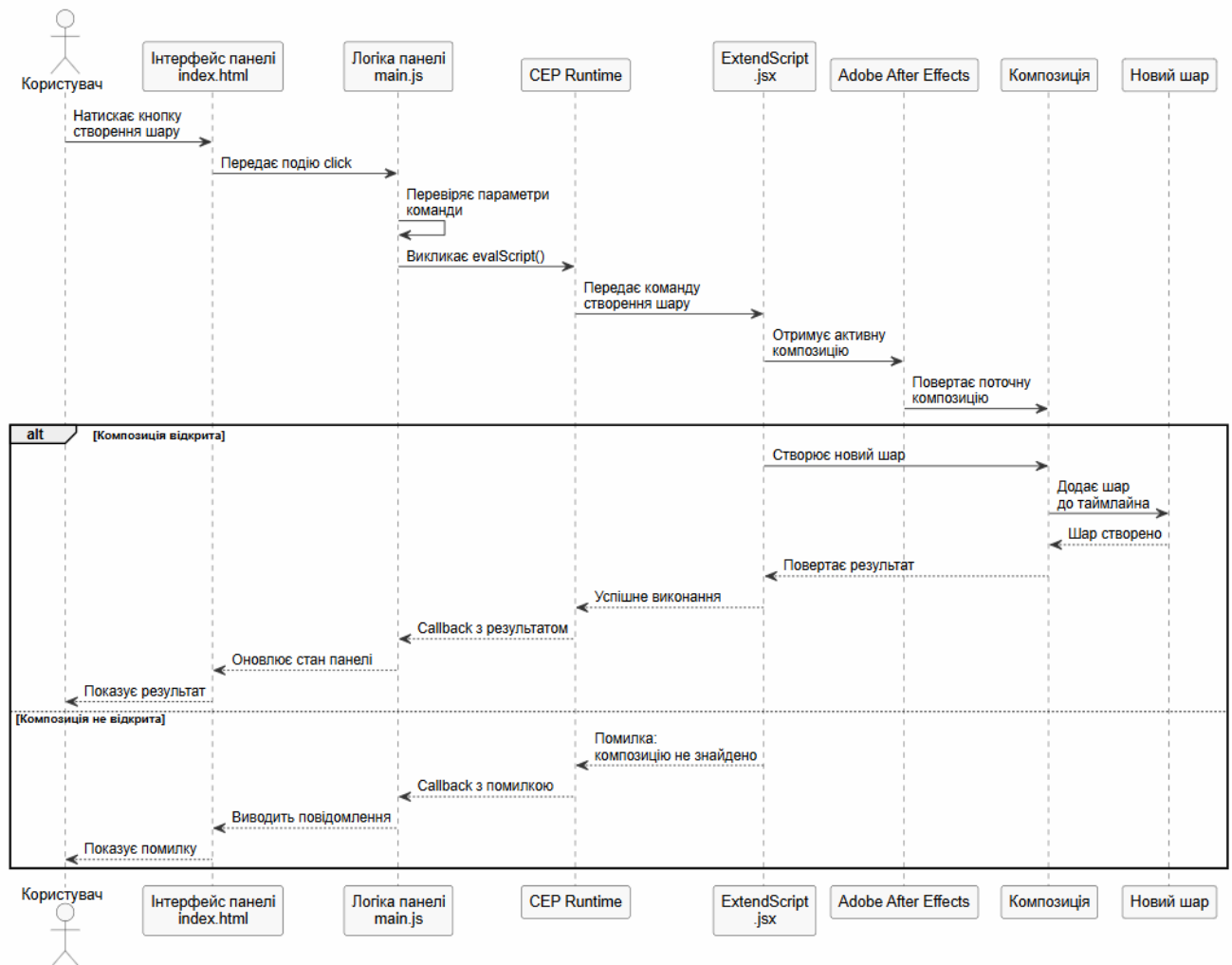


Рис. 3.3 Послідовність виконання команди створення шару

3.3 Проєктування алгоритмів обробки шарів

Алгоритми обробки шарів повинні враховувати контекст активної композиції. Перед будь-якою операцією перевіряється, чи є `app.project.activeItem` об'єктом `CompItem`. Якщо композиція відсутня, користувач отримує повідомлення, а команда не виконується.

Під час створення нового шару важливо визначити референсний шар. У FAST PANEL використовується вибраний шар, якщо він існує. Новий шар розміщується перед референсним, а за ввімкненого `Trim to Layer` отримує такі самі `startTime`, `inPoint` та `outPoint`. Якщо референсний шар відсутній, шар залишається у межах композиції.

Таблиця 3.1

Основні алгоритми FAST PANEL

Алгоритм	Вхідні дані	Результат
Create Layer	тип шару, Trim, 3D, parent, label color	створений і налаштований шар у активній композиції
Trim to Layer	новий шар, reference layer, час композиції	часові межі нового шару відповідають референсному шару
Set Anchor Point	вибрані шари, режим anchorMode	anchorPoint змінено без зміщення шару у кадрі
Fit to Comp	вибрані шари, duration композиції	inPoint=0, outPoint=duration
Pack Layers	вибрані шари з keyframes	ключові кадри розтягнуто у межах шару
Clear Effects	вибрані шари	ефекти видалено з ADBE Effect Parade

Алгоритм зміни `anchor point` побудовано на визначенні прямокутника вмісту через `sourceRectAtTime`. Залежно від вибраної позиції обчислюється `target point`. Після зміни `anchorPoint` виконується компенсація `position` з урахуванням `scale`, щоб шар візуально залишився на місці.

Алгоритм `packLayers` проходить по вибраних шарах і масштабує часові позиції `keyframes` до діапазону між `inPoint` та `outPoint`. Для цього функція рекурсивно обходить властивості шару, зберігає значення, `interpolation type` і `temporal ease`, видаляє старі ключі та додає їх у нових позиціях.

3.4 Проєктування імпорту SRT-субтитрів

Модуль `SRT IMPORT` спроектовано як окремий сценарій, оскільки він працює не тільки з інтерфейсом `After Effects`, а й з локальним текстовим файлом. На рівні `HTML` користувач вибирає файл, а `JavaScript` читає його через `FileReader`. Далі текст нормалізується за переносами рядків і розбивається на блоки субтитрів.

Кожен блок перевіряється на наявність рядка `timecode` з розділювачем `-->`. Час початку і завершення перетворюється у секунди, а текстові рядки очищуються від `HTML`-тегів. Якщо ввімкнено `Strip punctuation`, з тексту видаляються основні розділові знаки. Якщо ввімкнено `No gaps`, короткі проміжки між сусідніми субтитрами прибираються шляхом зміни `outTime` попереднього субтитру.

Після парсингу масив об'єктів передається у `hostscript.jsx`. `ExtendScript` перевіряє композицію, пропускає субтитри поза тривалістю композиції і для кожного валідного елемента створює текстовий шар. Такий алгоритм суттєво скорочує час підготовки титрів і зменшує ризик помилки у таймінгу.

Передавання даних виконується одним викликом, що спрощує обробку помилок: якщо рядок порожній або має некоректний формат, сценарій повідомляє про це користувача і не створює зайвих шарів. Таке проєктування робить імпорт стійким до неповних або пошкоджених файлів субтитрів.

Окремо у модулі передбачено формування назв шарів за порядковим номером репліки, що полегшує подальше редагування композиції. Завдяки цьому

навіть великий файл субтитрів перетворюється на впорядкований і зрозумілий набір текстових шарів, готовий до анімації та оформлення.

3.5 Проєктування структури даних та конфігурації команд

Для стабільної роботи FAST PANEL важливо не тільки реалізувати окремі кнопки, а й заздалегідь визначити, які дані передаються між інтерфейсом, клієнтською логікою та ExtendScript-рівнем.

На етапі проєктування було прийнято рішення розглядати кожну дію панелі як окрему команду з власним набором параметрів. Такий підхід спрощує супровід програмного коду, тому що зміни в одному сценарії не потребують перероблення всієї панелі.

Основними групами параметрів є параметри створення шару, параметри вирівнювання anchor point, налаштування кольорових міток, параметри гарячих клавіш та дані імпорту субтитрів. Для кожної групи визначається мінімальний набір полів, без яких команда не може бути виконана коректно. Наприклад, для створення шару потрібен тип шару, ознака Trim to Layer та контекст активної композиції, а для імпорту SRT потрібен вміст файлу, часові мітки та текстові репліки.

Під час проєктування враховано, що інтерфейс CEP-панелі працює у середовищі браузерного типу, а фактичні зміни у композиції виконує ExtendScript. Тому дані не повинні передаватися у вигляді неструктурованого тексту.

Доцільно формувати об'єкт параметрів на стороні JavaScript, перевіряти його перед викликом JSX-функції та передавати лише ті значення, які потрібні для конкретної операції. Це зменшує кількість прихованих залежностей між частинами системи.

Окремо було визначено принцип збереження налаштувань користувача. Значення, які не змінюють структуру проєкту After Effects, можуть зберігатися локально: останній вибраний тип шару, активність Trim to Layer, обраний колір

мітки або стан деяких перемикачів. Такі дані не потребують серверної частини і не мають бути спільними між різними користувачами, тому використання `localStorage` є достатнім для поставленої задачі.

Перевагою такої структури даних є передбачуваність. Коли користувач натискає кнопку, панель не збирає випадковий стан інтерфейсу, а формує зрозумілу команду. Це особливо важливо для операцій, що змінюють багато шарів одразу. Якщо команда описана у вигляді набору параметрів, її можна перевірити до виконання, вивести повідомлення про помилку або повторно використати в іншому сценарії.

У проєкті FAST PANEL структура даних також виконує роль внутрішньої документації. Назви параметрів мають відображати їх призначення, наприклад `layerType`, `trimToLayer`, `anchorPreset`, `labelIndex`, `subtitleItems`. Завдяки цьому код легше читати і перевіряти. У майбутньому така модель дозволить додати профілі користувача або імпорт налаштувань без повної зміни архітектури панелі.

3.6 Проєктування перевірок вхідних даних та обробки помилок

Будь-яке розширення для After Effects повинно враховувати ситуації, коли користувач виконує команду у невідповідному контексті. Наприклад, активна композиція може бути відсутня, шар може не бути вибраним, SRT-файл може мати неправильний формат, а часові мітки можуть виходити за межі тривалості композиції. Якщо ці випадки не передбачити на етапі проєктування, користувач отримає незрозумілу помилку або частково змінений проєкт.

Для FAST PANEL доцільно застосувати багаторівневу перевірку. Перший рівень виконується в інтерфейсі: кнопки, поля та перемикачі мають формувати тільки допустимі значення. Другий рівень виконується у JavaScript перед викликом `ExtendScript`: перевіряється повнота параметрів, типи значень та очевидні обмеження. Третій рівень виконується у JSX-коді, оскільки тільки він має доступ до активної композиції, виділених шарів і властивостей After Effects.

Для операцій зі шарами критичною є перевірка активного елемента проєкту. Команда повинна виконуватися тільки тоді, коли користувач працює з композицією. Якщо активним є інший тип елемента або проєкт не містить відкритої композиції, панель має повернути повідомлення з коротким поясненням. Такий підхід не тільки запобігає збоям, а й навчає користувача правильному порядку роботи.

Під час проєктування SRT IMPORT було виділено окрему групу помилок, пов'язаних з текстовим файлом. Формат SRT має блоки з номером, часовим інтервалом і текстом репліки. Якщо хоча б одна частина відсутня, імпорт може створити некоректні текстові шари. Тому алгоритм повинен ігнорувати порожні блоки, перевіряти наявність часової стрілки, коректно обробляти перенесення рядків і не створювати шар для репліки без часу.

Повідомлення про помилки мають бути короткими, але змістовними. Недостатньо показати текст `error` або `failed`, тому що користувач не зрозуміє, що саме виправити. Для типових ситуацій варто передбачити повідомлення на зразок: відкрийте композицію, виберіть шар, перевірте формат SRT-файлу, файл не містить реплік для імпорту. Такі повідомлення не перевантажують інтерфейс і водночас зменшують кількість повторних помилок.

Окремим принципом є атомарність виконання команд. Якщо операція складається з кількох кроків, вона повинна або завершитися повністю, або зупинитися до внесення змін. У After Effects для цього доцільно використовувати `undo group`.

Коли всі зміни об'єднані в одну групу, користувач може скасувати результат одним натисканням `Undo`. Це важливо для довіри до панелі, оскільки користувач не боїться перевіряти нові команди у реальному проєкті.

Таким чином, проєктування обробки помилок у FAST PANEL спрямоване не на приховування проблем, а на керовану взаємодію з ними. Панель повинна розпізнавати неправильний контекст, не виконувати небезпечні дії, пояснювати причину відмови і залишати композицію у передбачуваному стані. Саме це відрізняє робочий інструмент від набору окремих скриптів.

3.7 Проєктування взаємодії інтерфейсу з ExtendScript

СЕР-панель складається з двох середовищ, які мають різні можливості. Інтерфейс на HTML, CSS і JavaScript відповідає за відображення елементів керування, обробку подій користувача та підготовку параметрів. ExtendScript відповідає за роботу з композицією After Effects. Тому під час проєктування необхідно чітко визначити межу між цими частинами і не переносити логіку без потреби.

Користувацький інтерфейс не повинен напряму знати внутрішню структуру об'єктної моделі After Effects. Його задача полягає у тому, щоб отримати вибір користувача і передати його у вигляді команди. Наприклад, кнопка Set Anchor Point передає назву позиції або індекс пресету, а вже ExtendScript визначає, як змінити anchor point для конкретного шару. Це зменшує залежність верстки від особливостей After Effects.

Для зв'язку між JavaScript і ExtendScript використовується CSInterface.js. У проєктній моделі кожен виклик має бути ізольованим: JavaScript формує рядок виклику функції, передає параметри, очікує результат і після цього оновлює стан інтерфейсу. Якщо команда завершилась успішно, користувач отримує підтвердження або бачить результат у композиції. Якщо команда повернула помилку, інтерфейс показує повідомлення без повторного виклику функції.

Важливо врахувати, що передавання складних об'єктів між середовищами може бути незручним. Тому для FAST PANEL доцільно застосовувати прості параметри: рядки, числа, логічні значення та серіалізовані масиви. Для SRT-

імпорту список реплік може бути підготовлений у JavaScript, після чого ExtendScript отримує вже структуровані дані: початок, кінець, текст. Такий розподіл зменшує навантаження на JSX-код і спрощує тестування парсера.

Окремим завданням є синхронізація стану інтерфейсу після виконання команди. Наприклад, якщо користувач вмикає Trim to Layer, значення перемикача має зберігатися і використовуватися в наступних операціях. Якщо команда імпорту субтитрів завершена, панель може очистити поле вибору файлу або залишити його стан без змін. Такі рішення визначаються не випадково, а виходячи з очікуваного сценарію роботи.

Під час проєктування взаємодії також враховано затримки виконання. Деякі операції можуть тривати довше, особливо якщо композиція містить багато шарів або SRT-файл має велику кількість реплік. Тому інтерфейс має блокувати повторне натискання на час виконання команди або принаймні ігнорувати дублювати подій. Це запобігає створенню повторних шарів і неконтрольованим змінам у композиції.

Запропонована модель взаємодії робить FAST PANEL зрозумілою для подальшої розробки. Нову функцію можна додати за сталою схемою: створити елемент інтерфейсу, описати обробник події, сформулювати параметри, викликати JSX-функцію і повернути результат. Така послідовність підвищує якість коду і дозволяє уникнути хаотичного зростання проєкту.

3.8 Проєктування операцій з композицією та шарами

Найважливіші функції FAST PANEL пов'язані зі створенням і зміною шарів у композиції. Саме ці операції найчастіше повторюються у роботі motion-дизайнера і тому мають найбільший практичний ефект. Під час проєктування було визначено, що кожна операція повинна враховувати поточний час композиції, виділені шари, тривалість композиції та властивості створюваного елемента.

Команда створення шару має працювати з кількома типами об'єктів. У загальному випадку користувач очікує, що панель створить потрібний шар у

поточній композиції і розмістить його у зрозумілій позиції на таймлайні. Якщо активовано Trim to Layer, тривалість нового шару повинна узгоджуватися з тривалістю вибраного шару або з поточним робочим діапазоном. Це скорочує ручне підрізання шарів після створення.

Для anchor point важливо зберігати видиме положення шару. Звичайна зміна anchor point може призвести до зсуву об'єкта у кадрі, що створює додаткову роботу для дизайнера. Тому алгоритм повинен розраховувати компенсацію позиції. Логіка полягає у тому, що після зміни точки прив'язки координати шару коригуються так, щоб візуальний результат у композиції залишався тим самим.

Операції Fit to Comp і Pack Layers також потребують проєктного опису. Fit to Comp має враховувати співвідношення сторін шару та композиції, щоб не створювати небажаних спотворень. Pack Layers має працювати з порядком шарів і часовими межами, не руйнуючи вже встановлену структуру композиції. Для таких дій особливо важливо передбачити, що користувач може виділити один шар, кілька шарів або не виділити нічого.

Під час проєктування було визначено, що алгоритми не повинні змінювати приховані або заблоковані елементи без потреби. Якщо шар заблокований, його зміна може суперечити очікуванню користувача. Тому для масових операцій доцільно або пропускати такі шари, або повертати повідомлення про неможливість виконання. Це правило підвищує безпеку роботи з великими композиціями.

Окремо враховано роботу з label colors. Кольорові мітки в After Effects використовуються для швидкого групування шарів за призначенням. FAST PANEL не повинна нав'язувати жорстку систему назв, але має дати змогу швидко застосовувати обраний колір до активних елементів. На рівні алгоритму це проста операція, але на рівні робочого процесу вона зменшує хаос у композиції і робить таймлайн зручнішим для навігації.

Під час виконання кожної операції з композицією система повинна мінімізувати побічні ефекти. Якщо команда створює шар, вона не повинна випадково змінювати параметри інших шарів. Якщо команда працює з виділенням, вона повинна використовувати тільки актуальний список об'єктів. Якщо команда

змінює часові межі, вона повинна враховувати одиниці часу After Effects і не округлювати значення грубо.

Отже, проєктування операцій з композицією у FAST PANEL ґрунтується на принципах керованості, передбачуваності та швидкого скасування. Саме ці принципи роблять автоматизацію корисною у творчому середовищі, де користувач часто експериментує і потребує швидкого повернення до попереднього стану.

3.9 Проєктування імпорту субтитрів як окремого сценарію

Імпорт SRT-субтитрів відрізняється від інших функцій FAST PANEL тим, що працює не тільки з поточним станом композиції, а й із зовнішнім текстовим файлом. Тому цей сценарій потребує окремого проєктування. Його результатом є набір текстових шарів, розміщених у часі відповідно до міток у файлі. Користувач повинен отримати готову структуру, яку можна одразу редагувати засобами After Effects.

Першим кроком сценарію є вибір файлу. На цьому етапі інтерфейс має приймати тільки текстові файли або принаймні повідомляти користувача, якщо обраний файл не може бути прочитаний. Далі FileReader зчитує вміст і передає його у парсер. Парсер не повинен залежати від конкретної кількості рядків у репліці, оскільки SRT може містити як короткі, так і багаторядкові субтитри.

Другим кроком є перетворення часових міток. SRT використовує формат годин, хвилин, секунд і мілісекунд, а After Effects працює з часом у секундах. Тому алгоритм має коректно перераховувати значення і зберігати точність, достатню для монтажу. Якщо час завершення менший за час початку або дорівнює йому, такий блок потрібно вважати помилковим і не створювати для нього текстовий шар.

Третім кроком є створення текстових шарів. Для кожної коректної репліки створюється окремий шар, який отримує текст субтитру, початковий час і тривалість.

Важливо, щоб шари не створювалися всі в одній точці таймлайну без часових меж, інакше користувач втратить основну перевагу автоматизації. Саме розміщення у часі є головною цінністю функції SRT IMPORT.

Під час проєктування було враховано, що субтитри можуть потребувати подальшого стилістичного оформлення. FAST PANEL не повинна замінювати повноцінний модуль дизайну титрів, але створені шари мають бути придатними для подальшого форматування. Тому доцільно задавати базові параметри тексту, які не заважають користувачу змінити шрифт, розмір, позицію або анімацію після імпорту.

Для великих SRT-файлів важливо уникати надмірного навантаження на інтерфейс. Якщо файл містить сотні реплік, операція створення шарів може зайняти помітний час. Тому сценарій має виконуватися як одна керована операція з undo group, а інтерфейс має запобігати повторному запуску до завершення поточного імпорту. Це зменшує ризик дублювання субтитрів.

Окремо було визначено вимогу до повідомлення про результат. Після імпорту користувач повинен знати, скільки реплік було створено і чи були пропущені помилкові блоки. Таке повідомлення допомагає швидко перевірити відповідність між файлом і результатом у композиції. Якщо створено нуль шарів, панель має вказати, що файл не містить коректних SRT-блоків.

Таким чином, імпорт субтитрів розглядається не як допоміжна дрібна функція, а як повноцінний сценарій автоматизації. Він охоплює зчитування файлу, парсинг, перевірку даних, створення шарів і повідомлення користувача про результат. Такий підхід робить функцію надійною і придатною для практичного використання у відеомонтажі.

3.10 Проєктування станів інтерфейсу та зручності використання

Для СЕР-панелі важливо не тільки мати правильні алгоритми, а й забезпечити зручну щоденну взаємодію. Motion-дизайнер працює у швидкому темпі, часто перемикається між шарами, переглядає композицію і виконує багато дрібних дій. Тому інтерфейс FAST PANEL має бути компактним, передбачуваним і не вимагати зайвих пояснень під час роботи.

Під час проєктування інтерфейсу було визначено кілька основних станів: готовність до виконання команди, активне виконання, успішне завершення і помилка. У стані готовності елементи керування доступні користувачу. У стані виконання частина елементів може тимчасово блокуватися. Після успішного завершення панель повертається до нормального стану, а у разі помилки показує коротке повідомлення.

Структура панелі поділена на функціональні блоки. CREATE відповідає за створення шарів, ANCHOR POINT за зміну точки прив'язки, LABEL COLORS за організацію шарів, HOTKEYS за швидкий доступ до команд, SRT IMPORT за імпорт субтитрів. Такий поділ відповідає природному робочому процесу і дозволяє користувачу швидко знайти потрібну дію без перегляду довгого меню.

Для перемикачів і кнопок важливо зберігати послідовність поведінки. Якщо користувач увімкнув певне налаштування, панель повинна пам'ятати його у межах наступних запусків. Якщо налаштування впливає тільки на одну команду, воно має бути розміщене поруч з цією командою. Це зменшує кількість помилок, коли користувач не розуміє, чому виконання відбулося не так, як очікувалося.

Зручність використання також залежить від розміру панелі. СЕР-розширення часто працюють у вузьких бокових областях інтерфейсу After Effects, тому елементи не повинні вимагати великої ширини. Текст кнопок має бути коротким, а групи елементів повинні мати зрозумілий порядок. Надмірна кількість декоративних елементів у такому інструменті недоцільна, тому пріоритет надано функціональності.

Під час проєктування станів інтерфейсу враховано, що користувач може виконувати команди багато разів поспіль. Тому після завершення дії панель не повинна скидати всі налаштування без причини. Наприклад, якщо дизайнер кілька разів створює технічні шари з однаковими параметрами, збереження останнього стану пришвидшує роботу. Це невелика деталь, але вона суттєво впливає на відчуття зручності.

Інтерфейс FAST PANEL також має бути нейтральним щодо стилю конкретного проєкту. Він не створює готовий дизайн сцени, а лише надає інструменти для прискорення технічних операцій. Саме тому панель не повинна нав'язувати користувачу зайві шаблони оформлення. Усі автоматизовані дії залишають можливість подальшого ручного редагування результату.

3.11 Проєктування можливостей масштабування та супроводу

Після створення першої версії FAST PANEL важливо мати можливість розвивати її без повного переписування коду. На етапі проєктування було враховано, що у майбутньому можуть з'явитися нові команди для вирівнювання, пакетної зміни властивостей, роботи з текстовими пресетами або додаткової обробки субтитрів. Тому архітектура повинна підтримувати поступове розширення.

Основним принципом масштабування є модульність. Кожна нова функція має додаватися як окремий блок інтерфейсу, окремий обробник події та окрема JSX-функція. Якщо нова команда потребує спільних допоміжних функцій, вони можуть бути винесені у внутрішній службовий модуль. Такий підхід не дозволяє одному файлу перетворитися на некерований набір випадкових фрагментів коду.

Для супроводу проєкту важливо також мати узгоджені назви файлів і функцій. Наприклад, функції, які створюють шари, повинні мати спільну логіку іменування, а функції, які змінюють властивості вибраних шарів, повинні бути

згруповані. Це спрощує пошук потрібної частини коду під час виправлення помилки або додавання нової можливості.

Масштабування стосується не тільки коду, а й інтерфейсу. Якщо у панель додати забагато кнопок без структури, вона втратить головну перевагу - швидкість. Тому нові можливості мають проходити відбір за частотою використання і практичною користю. Команди, які потрібні рідко, можуть залишатися у стандартних меню After Effects або бути винесені в окремий додатковий блок.

Ще одним напрямом розвитку є підтримка профілів користувача. Різні дизайнери можуть мати різні набори улюблених параметрів: кольори міток, стандартні типи шарів, налаштування субтитрів.

Поточна модель localStorage дозволяє зберігати окремі значення, а у майбутньому її можна розширити до наборів профілів. Це не змінює базову архітектуру, якщо дані вже проєктуються структуровано.

Під час супроводу важливо зберігати сумісність з різними версіями After Effects, які підтримують CEP. Тому нові функції не повинні без перевірки використовувати можливості, доступні тільки у конкретній версії програми. Якщо така можливість потрібна, панель має перевіряти середовище виконання або передбачати альтернативний сценарій.

Отже, масштабування FAST PANEL передбачає контрольоване додавання можливостей, збереження зрозумілої структури коду і обережне ставлення до інтерфейсу. Розширення має залишатися швидким робочим інструментом, а не перетворюватися на перевантажений набір команд. Саме тому питання супроводу враховано вже на етапі проєктування системи.

3.12 Висновки до розділу 3

У третьому розділі було спроектовано систему FAST PANEL як CEP-розширення, що поєднує інтерфейсну частину, клієнтську логіку та ExtendScript-

рівень взаємодії з After Effects. Визначено загальну архітектуру, ролі основних компонентів і принцип передавання команд між середовищами.

Описано варіанти використання панелі та типові сценарії роботи motion-дизайнера. Проаналізовано, які дії користувач виконує найчастіше і які з них доцільно винести в окремі швидкі команди. Завдяки цьому проєктування орієнтоване не на абстрактний набір функцій, а на реальний робочий процес у композиції After Effects.

Сформовано підхід до проєктування алгоритмів роботи з шарами, anchor point, label colors і SRT-субтитрами. Для кожної групи операцій визначено необхідні вхідні дані, очікуваний результат, можливі помилкові ситуації та вимоги до скасування змін. Це дозволяє реалізувати функції послідовно і без дублювання логіки.

Окрему увагу приділено перевірці вхідних даних, обробці помилок, станам інтерфейсу та можливостям подальшого масштабування. Запропонована проєктна модель забезпечує передбачувану поведінку панелі, підвищує зручність використання і створює основу для розвитку FAST PANEL у наступних версіях.

У межах розділу також визначено, що кожна функція панелі повинна мати зрозумілу межу відповідальності. Інтерфейс збирає параметри та відображає результат, JavaScript координує подію і перевіряє дані, а ExtendScript виконує зміни у композиції. Такий розподіл дозволяє локалізувати помилки і не змішувати логіку відображення з логікою роботи After Effects.

Важливим результатом проєктування є опис реакції системи на неправильні дії користувача. Панель повинна не тільки виконувати команди у правильних умовах, а й коректно відмовлятися від виконання, коли композиція не відкрита, файл субтитрів має неправильний формат або виділення шарів не відповідає сценарію. Це підвищує надійність інструменту у повсякденному використанні.

Під час проєктування враховано, що FAST PANEL працює у творчому середовищі, де користувач часто виконує експерименти. Саме тому всі масові зміни повинні бути передбачуваними і скасовуватися однією дією. Використання undo

group є не другорядною технічною деталлю, а обов'язковою умовою безпечної автоматизації у композиції After Effects.

Для сценарію SRT IMPORT окремо визначено повний шлях даних: вибір файлу, зчитування тексту, розбиття на блоки, перевірка часових міток, перетворення часу у секунди, створення текстових шарів і повідомлення користувача про результат. Такий опис дає змогу реалізувати функцію не як разовий скрипт, а як контрольований інструмент для монтажу субтитрів.

Проектування інтерфейсу показало, що компактність є однією з ключових вимог. Панель повинна займати небагато місця в робочому просторі After Effects, але при цьому надавати доступ до найчастіше використовуваних команд. Тому функції згруповано за призначенням, а налаштування розміщено поруч із командами, на які вони впливають.

Описана структура також полегшує тестування наступного розділу. Якщо для кожної команди визначено очікувані вхідні дані, результат і помилкові ситуації, то тестові сценарії можна будувати без здогадок. Це забезпечує зв'язок між проектуванням, реалізацією та перевіркою працездатності програмного забезпечення.

Додатковою перевагою такого проектування є можливість поетапного впровадження панелі у робочий процес. Користувач може спочатку застосовувати тільки базові команди створення шарів і зміни anchor point, а після перевірки зручності перейти до імпорту субтитрів або пакетної організації шарів. Це зменшує ризик відмови від інструменту через надмірну складність першого використання.

Для супроводу проєкту важливо, що описані рішення не прив'язані до одного конкретного макета композиції. Панель може використовуватися у різних типах відеопроєктів: коротких анімаційних роликах, навчальних матеріалах, титрованих відео або монтажних заготовках. Обмеження визначаються не тематикою ролика, а можливостями After Effects і коректністю вхідних даних.

Завдяки цьому FAST PANEL має практичний потенціал як універсальний допоміжний інструмент. Проектування показує, що автоматизація повинна бути обережною: вона прискорює повторювані дії, але не позбавляє користувача

контролю над творчим результатом. Саме такий баланс є доцільним для програмного розширення, яке працює всередині професійного середовища motion-дизайну.

Окремо слід зазначити, що проєктна модель не суперечить стандартним прийомам роботи в After Effects. Усі створені шари залишаються звичайними елементами композиції, доступними для редагування, перейменування, анімації та зміни властивостей. Це важливо, тому що користувач не опиняється у закритій системі і може після автоматизованої дії продовжити роботу звичними інструментами програми.

Проєктування також враховує обмеження CEP як технології. Панель працює у межах можливостей, які надають HTML, JavaScript, CSInterface.js і ExtendScript, тому складні операції мають бути розбиті на зрозумілі етапи. Такий підхід зменшує залежність від неочевидної поведінки середовища і робить роботу розширення більш прогнозованою під час оновлення програмного забезпечення.

У підсумку розділ 3 виконує роль зв'язувальної частини між аналізом предметної галузі та програмною реалізацією. Він пояснює, чому обрані саме такі функції, як вони мають взаємодіяти між собою і за якими правилами повинні виконуватися у композиції. Без такого проєктного етапу реалізація могла б перетворитися на набір окремих кнопок без єдиної логіки.

Таким чином, третій розділ формує основу для реалізації FAST PANEL. У ньому визначено не лише перелік майбутніх функцій, а й правила їх взаємодії, обмеження, вимоги до користувацького досвіду та напрями подальшого розвитку. Завдяки цьому реалізація у четвертому розділі спирається на завершену проєктну модель.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

4.1 Структура програмного проєкту

Програмний проєкт FAST PANEL розміщено у каталозі Adobe CEP extensions. Структура відповідає типовій організації CEP-панелі: окрема папка CSXS з manifest.xml, папка css для стилів, папка js для клієнтської логіки, папка jsx для ExtendScript, а також головний файл index.html.

Таблиця 4.1

Файли та каталоги FAST PANEL

Елемент	Призначення
CSXS/manifest.xml	опис розширення, host-додатка, розмірів панелі та шляхів до ресурсів
index.html	структура інтерфейсу: CREATE, Trim, ANCHOR POINT, LABEL COLORS, HOTKEYS, SRT IMPORT
css/style.css	темне оформлення, сітки кнопок, стани hover/active, collapsible-секції
js/CSInterface.js	бібліотека Adobe CEP для взаємодії з host-додатком
js/main.js	DOM-події, localStorage, hotkeys, парсер SRT, виклики evalScript
js/license.js	активація ліцензії, локальне збереження стану, періодична перевірка
jsx/hostscript.jsx	операції з шарами After Effects і створення субтитрів

Файл `manifest.xml` містить `ExtensionBundleId` `com.fastpanel.extension`, `host` AEFT з діапазоном версій, тип `Panel`, назву меню `Fast Neon Panel` та геометрію панелі. Це дозволяє After Effects знайти розширення, завантажити `index.html` і підключити `hostscript.jsx`.

4.2 Реалізація інтерфейсу та взаємодії з користувачем

Інтерфейс FAST PANEL реалізовано у вигляді вертикальної панелі з компактними секціями. Верхня частина містить назву продукту, блок CREATE і перемикач Trim to Layer. Нижче розміщено сітку ANCHOR POINT, згортані блоки LABEL COLORS і HOTKEYS, а також SRT IMPORT.

Кнопки мають різні акцентні стилі: основні дії виділено бірюзовою заливкою, допоміжні команди мають контурний або темний стиль. Такий поділ допомагає користувачу швидко розпізнати пріоритетні дії, але не перевантажує панель зайвими декоративними елементами.

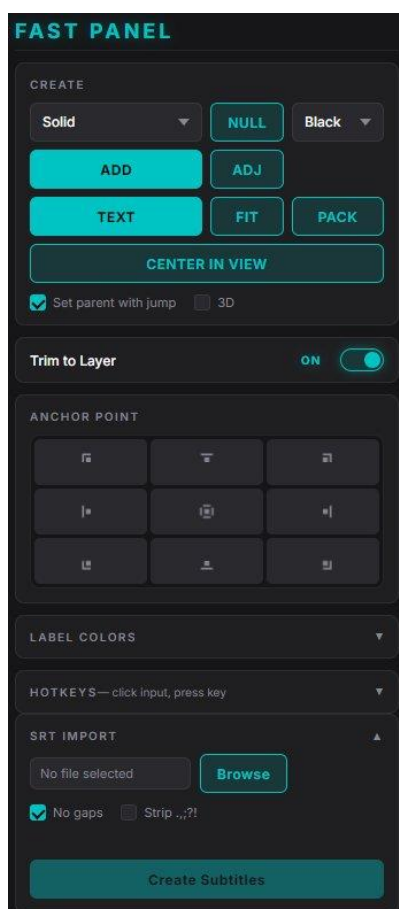


Рис. 4.1 Загальний вигляд інтерфейсу FAST PANEL

Стан інтерфейсу відновлюється під час `init()`. Hotkeys, label colors, Trim to Layer, 3D і parent читаються з `localStorage`. Якщо користувач змінює налаштування, нове значення одразу записується у локальне сховище. Завдяки цьому панель поводить себе стабільно між сеансами.

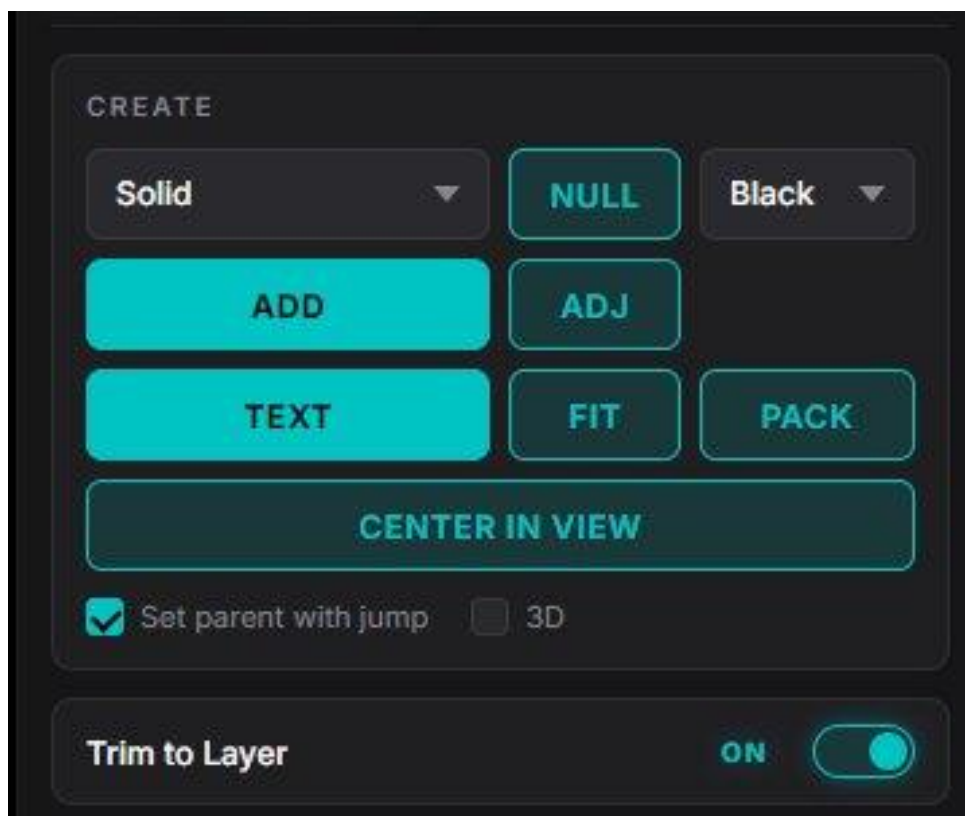


Рис. 4.2 Блок CREATE та перемикач Trim to Layer

Функції `createSolid`, `createNull`, `createAdjLayer`, `createText` і `createShape` мають схожу структуру, тому користувач отримує однакову поведінку для різних типів шарів. Camera і Light мають окремі налаштування, пов'язані з позицією і параметрами камери або джерела світла.

Функція `setAnchorPoint` працює з усіма вибраними шарами. Вона використовує `sourceRectAtTime` для визначення меж вмісту, обирає цільову точку за режимом `top-left`, `center`, `bottom-right` тощо, а потім змінює `anchorPoint` і `position`. Така реалізація вирішує проблему зміщення шару.

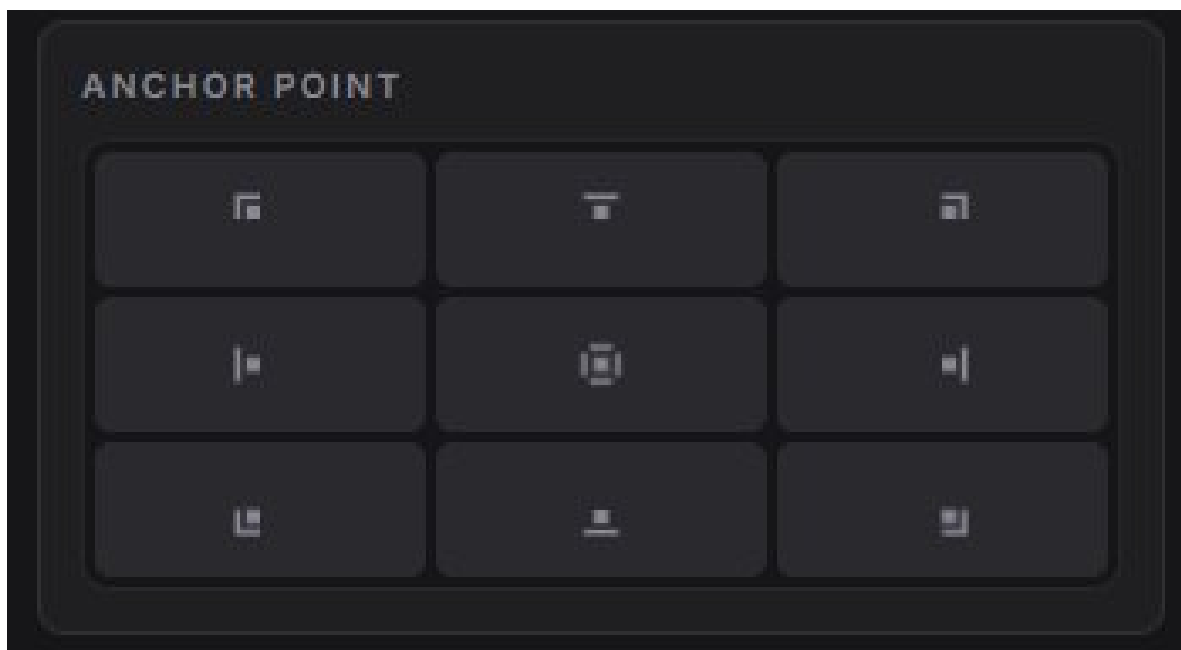


Рис. 4.3 Блок ANCHOR POINT для вибору точки прив'язки

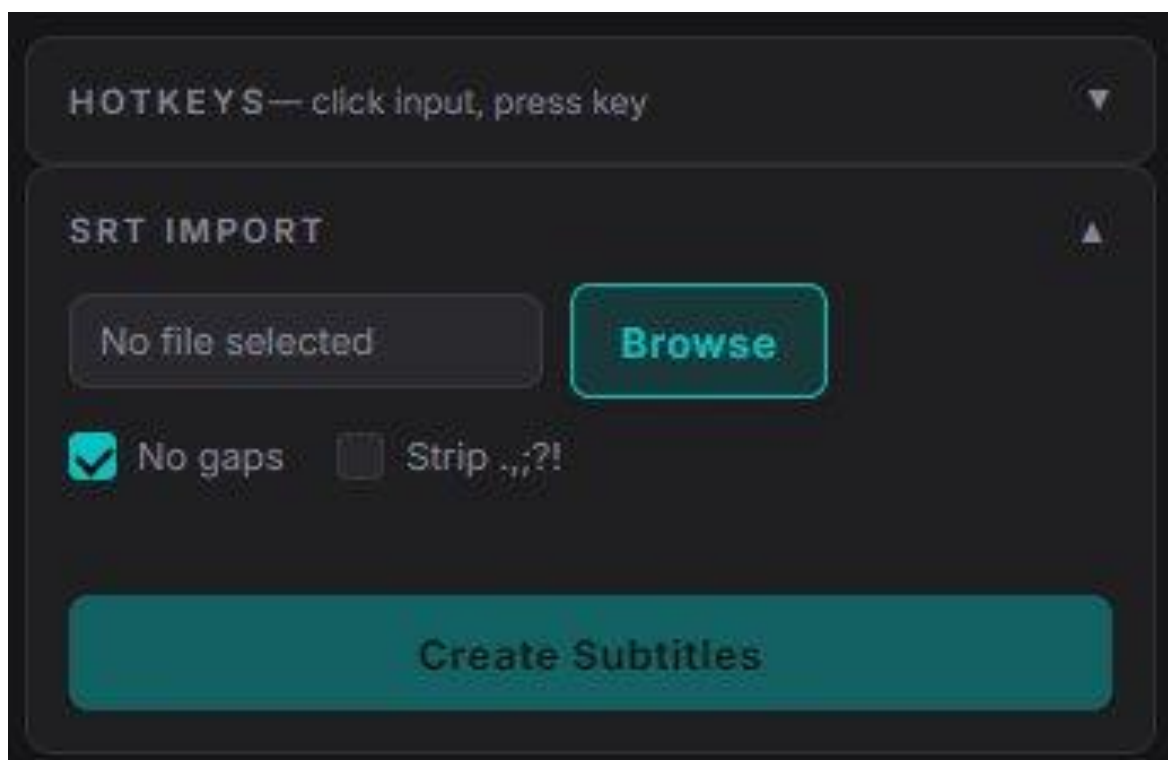


Рис. 4.4 Блок SRT IMPORT для створення субтитрів

Файл `hostscript.jsx` містить функції, які безпосередньо змінюють композицію After Effects. Для створення шарів використовується спільна логіка: отримати активну композицію, знайти `reference layer`, створити шар потрібного типу, розмістити його над референсним, застосувати `Trim`, `3D`, `parent` і `label color`.

Для SRT-субтитрів ExtendScript отримує JSON-рядок, перетворює його на масив об'єктів і створює текстові шари. Кожен шар отримує inPoint та outPoint відповідно до timescode. Якщо субтитр виходить за межі композиції, він обрізається або пропускається.

Таблиця 4.2

Реалізовані функції hostscript.jsx

Функція	Призначення	Особливість реалізації
createSolid/createText/createNull	створення базових шарів	підтримка Trim, 3D, parent і label color
createAdjLayer	створення adjustment layer	solid перетворюється на adjustmentLayer
createCamera/createLight	створення 3D-об'єктів	позиція задається відносно центру композиції
setAnchorPoint	зміна точки прив'язки	компенсація position після зміни anchorPoint
fitToComp	підгонка шарів під композицію	inPoint=0, outPoint=duration
packLayers	розтягування keyframes	рекурсивний обхід властивостей шару
importSRT	створення субтитрів	текстові шари з таймінгом SRT

4.3 Тестування FAST PANEL

Тестування проводилося за сценаріями, які відповідають реальним діям motion-дизайнера. Перевірялися створення різних типів шарів, Trim to Layer, зміна anchor point, Fit to Comp, Pack, імпорт SRT, збереження hotkeys і label colors. Окремо перевірялись ситуації без активної композиції або без вибраних шарів.

Позитивним результатом вважалося коректне виконання операції в активній композиції, правильне розміщення шару, очікувані часові межі, відсутність візуального зсуву після зміни anchor point і можливість скасувати дію одним Undo. Для SRT перевірялась кількість створених текстових шарів і відповідність їхніх меж timescode.

За результатами тестування панель виконала основні сценарії. Найбільш корисними з практичної точки зору виявилися CREATE з Trim to Layer, ANCHOR POINT 3x3 і SRT IMPORT, оскільки саме ці дії у ручному режимі потребують найбільше повторюваних кроків.

Таблиця 4.3

Результати тестування основних сценаріїв

Сценарій	Очікуваний результат	Фактичний результат	Статус
Створення Solid/Text/Null	шар створено у активній композиції	шар створюється, опції застосовуються	пройдено
Trim to Layer	межі шару обрізаються за вибраним шаром	inPoint/outPoint копіюються з вибраного шару	пройдено
Anchor Point 3x3	шар не зміщується у кадрі	position компенсується після зміни anchorPoint	пройдено

Результати тестування основних сценаріїв

Fit to Comp	вибрані шари займають всю тривалість композиції	межі встановлюються від 0 до duration	пройдено
Pack Layers	keyframes розтягуються в межах шару	ключі переносяться рекурсивно	пройдено
SRT IMPORT	створюються текстові шари за timecode	шари створюються, некоректні блоки пропускаються	пройдено
Відсутня композиція	користувач бачить повідомлення	виводиться alert Select a composition first	пройдено

Проведене тестування підтвердило, що FAST PANEL є придатною для використання як допоміжний інструмент у робочому процесі After Effects. Панель не змінює концепцію роботи програми, але прибирає частину повторюваних дій і робить їх доступними з одного компактного інтерфейсу.

ВИСНОВКИ

У кваліфікаційній роботі було розглянуто задачу автоматизації типових операцій motion-дизайнера в Adobe After Effects. Визначено, що найбільш доцільно автоматизувати створення технічних шарів, керування часовими межами, зміну anchor point, налаштування label colors, hotkeys та імпорт SRT-субтитрів.

Проведено аналіз предметної галузі та існуючих підходів до автоматизації. Встановлено, що стандартні засоби After Effects є універсальними, але не завжди зручними для коротких повторюваних сценаріїв. Окремі скрипти вирішують локальні задачі, проте не забезпечують цілісного інтерфейсу. FAST PANEL поєднує потрібні команди в одній CEP-панелі.

Досліджено технологічні засоби реалізації: HTML, CSS, JavaScript, CSInterface.js, ExtendScript і manifest.xml. Обґрунтовано поділ системи на інтерфейс, клієнтську логіку і рівень виконання команд у After Effects. Такий поділ робить код зрозумілим і придатним для подальшого розвитку.

Спроектовано архітектуру FAST PANEL, варіанти використання, послідовність виконання команд, пакетну структуру і алгоритм імпорту SRT. Окремо описано алгоритми Trim to Layer, Set Anchor Point, Fit to Comp, Pack Layers та importSRT.

Реалізовано CEP-розширення FAST PANEL з інтерфейсом CREATE, ANCHOR POINT, LABEL COLORS, HOTKEYS і SRT IMPORT. Реалізація використовує localStorage для збереження налаштувань користувача, FileReader для читання SRT-файлів і ExtendScript для внесення змін до активної композиції.

Тестування підтвердило працездатність основних сценаріїв. Панель створює шари, застосовує параметри, змінює anchor point без зсуву, імпортує субтитри та коректно реагує на помилкові ситуації. Практичний результат полягає у скороченні кількості ручних дій і підвищенні зручності роботи з типовими елементами композиції.

Подальший розвиток FAST PANEL може включати підключення до інтерфейсу функцій align/distribute, розширення налаштувань текстових шарів для

SRT, додавання профілів користувача і покращення ліцензійного модуля. Завдяки обраній архітектурі ці зміни можна вносити поступово без повної переробки проєкту.

Додатково в роботі уточнено проєктні рішення, які визначають надійність FAST PANEL у реальному середовищі After Effects. Зокрема, розглянуто структуру даних команд, перевірку вхідних параметрів, обробку типових помилок, використання undo group та принципи безпечного виконання операцій з композицією.

Практична цінність розробки полягає в тому, що панель не змінює звичний підхід до роботи motion-дизайнера, а скорочує кількість повторюваних технічних дій. Користувач зберігає контроль над композицією, може редагувати створені шари вручну і за потреби скасувати автоматизовану операцію стандартними засобами After Effects.

У результаті проєкт FAST PANEL можна розглядати як основу для подальшого розвитку внутрішніх інструментів студії або індивідуального робочого середовища дизайнера. До перспектив розвитку належать профілі налаштувань, розширені пресети текстових шарів, додаткові команди вирівнювання та покращене повідомлення про результати пакетних операцій.

Тези доповідей:

1. Попович М.С., Аброскін Ю.Ю. Визначення вимог до скрипту Adobe After Effects для прискорення роботи з об'єктами та анкор-точкою. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 231–234.

2. Попович М.С., Аброскін Ю.Ю. Розробка програмного розширення для Adobe After Effects. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, ДУІКТ. Збірник тез. К.: ДУІКТ, 2026. С. 100–103.

ПЕРЕЛІК ПОСИЛАНЬ

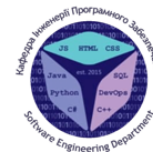
1. Adobe-CEP. CEP 12 HTML Extension Cookbook. URL: https://github.com/Adobe-CEP/CEP-Resources/blob/master/CEP_12.x/Documentation/CEP%2012%20HTML%20Extension%20Cookbook.md (дата звернення: 11.05.2026).
2. Adobe-CEP. CEP Resources: tools and documentation for building Creative Cloud app extensions with CEP. URL: <https://github.com/Adobe-CEP/CEP-Resources> (дата звернення: 11.05.2026).
3. After Effects Scripting Guide. LayerCollection object. URL: <https://ae-scripting.docsforadobe.dev/layer/layercollection/> (дата звернення: 11.05.2026).
4. After Effects Scripting Guide. Layer object. URL: <https://ae-scripting.docsforadobe.dev/layer/layer/> (дата звернення: 11.05.2026).
5. Adobe Developer. UXP for CEP Developers. URL: https://developer.adobe.com/photoshop/uxp/guides/uxp_for_you/uxp_for_cep_devs/ (дата звернення: 11.05.2026).
6. MDN Web Docs. Window: localStorage property. URL: <https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage> (дата звернення: 11.05.2026).
7. MDN Web Docs. FileReader. URL: <https://developer.mozilla.org/en-US/docs/Web/API/FileReader> (дата звернення: 11.05.2026).
8. MDN Web Docs. EventTarget: addEventListener() method. URL: <https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener> (дата звернення: 11.05.2026).
9. MDN Web Docs. HTML input type=file. URL: <https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file> (дата звернення: 11.05.2026).
10. Adobe. After Effects User Guide. URL: <https://helpx.adobe.com/after-effects/user-guide.html> (дата звернення: 11.05.2026).

11. Adobe. Scripting in After Effects. URL: <https://helpx.adobe.com/after-effects/using/scripts.html> (дата звернення: 11.05.2026).
12. Ecma International. ECMAScript Language Specification. URL: <https://tc39.es/ecma262/> (дата звернення: 11.05.2026).
13. W3C. Web Storage Second Edition. URL: <https://www.w3.org/TR/webstorage/> (дата звернення: 11.05.2026).
14. WHATWG. HTML Living Standard. URL: <https://html.spec.whatwg.org/> (дата звернення: 11.05.2026).
15. Adobe. Creative Cloud Developer documentation. URL: <https://developer.adobe.com/> (дата звернення: 11.05.2026).
16. After Effects Scripting Guide. Application object. URL: <https://ae-scripting.docsforadobe.dev/general/application/> (дата звернення: 11.05.2026).
17. After Effects Scripting Guide. CompItem object. URL: <https://ae-scripting.docsforadobe.dev/item/compitem/> (дата звернення: 11.05.2026).
18. After Effects Scripting Guide. TextLayer object. URL: <https://ae-scripting.docsforadobe.dev/layer/textlayer/> (дата звернення: 11.05.2026).
19. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol: O'Reilly Media, 2020. 706 p.
20. Crockford D. JavaScript: The Good Parts. Sebastopol: O'Reilly Media, 2008. 176 p.
21. Duckett J. JavaScript and jQuery: Interactive Front-End Web Development. Indianapolis: John Wiley & Sons, 2014. 640 p.
22. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis: John Wiley & Sons, 2011. 512 p.
23. Meyer C., Meyer T. Creating Motion Graphics with After Effects. 5th ed. Burlington: Focal Press, 2010. 704 p.
24. Christiansen M. Adobe After Effects Classroom in a Book (2021 release). San Jose: Adobe Press, 2021. 384 p.
25. Figma. Help Center: Design and prototype. URL: <https://help.figma.com/> (дата звернення: 11.05.2026).

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ
ТЕХНОЛОГІЙ



Кафедра інженерії програмного забезпечення

Розробка розширення до програмного забезпечення для автоматизації процесів анімації та відеомонтажу

Виконав: студент групи ПД-44 Попович Максим Сергійович

Керівник: асистент кафедри Аброскін Юрій Юрійович

Київ - 2026

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення та прискорення типових операцій motion-дизайнера в Adobe After Effects за рахунок розроблення CEP-панелі FAST PANEL з автоматизованим створенням шарів, керуванням anchor point, гарячими клавішами, імпортом SRT та пакетною обробкою шарів.

Об'єкт дослідження: процес автоматизації роботи з шарами, композиціями та субтитрами в Adobe After Effects.

Предмет дослідження: методи й програмні засоби розроблення CEP-розширень на HTML/CSS/JavaScript та ExtendScript.

ЗАВДАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Проаналізувати типові операції в Adobe After Effects, які займають найбільше часу під час роботи з шарами.
2. Дослідити структуру CEP-розширень та взаємодію HTML/CSS/JavaScript з Extendscript.
3. Сформулювати функціональні та нефункціональні вимоги до програмного забезпечення.
4. Спроекувати інтерфейс FAST PANEL і визначити набір основних функцій панелі.
5. Реалізувати створення шарів, Trim to Layer, Anchor Point, hotkeys, label colors, Pack/Fit/Center та імпорт SRT.
6. Протестувати роботу панелі у типових сценаріях motion-дизайну та підготувати звітну документацію.

3

АНАЛІЗ АНАЛОГІВ

Критерій	Нативні засоби AE	Animation Composer	Motion Tools	Motion Bro	FAST PANEL
Створення шарів	ручне створення через меню	готові пресети, не фокус на шарах	допоміжні дії з шарами	пресети та шаблони	Solid, Text, Null, Adj, Camera, Light
Trim / Fit / Pack	можливо вручну на timeline	підтримується частково через пресети	є інструменти таймінгу	переважно через шаблони	Trim to Layer, Fit to Comp, Pack Keyframes
Hotkeys	стандартні комбінації AE	обмежено діями плагіна	є швидкі кнопки	не є основною функцією	власні hotkeys у панелі
SRT-субтитри	немає прямого імпорту SRT	не підтримує	не підтримує	не підтримує	парсинг .srt і створення Text Layers
Власний workflow	низька: багато ручних дій	середня: готові пресети	середня: утиліти workflow	середня: бібліотеки шаблонів	висока: функції під задачу

4

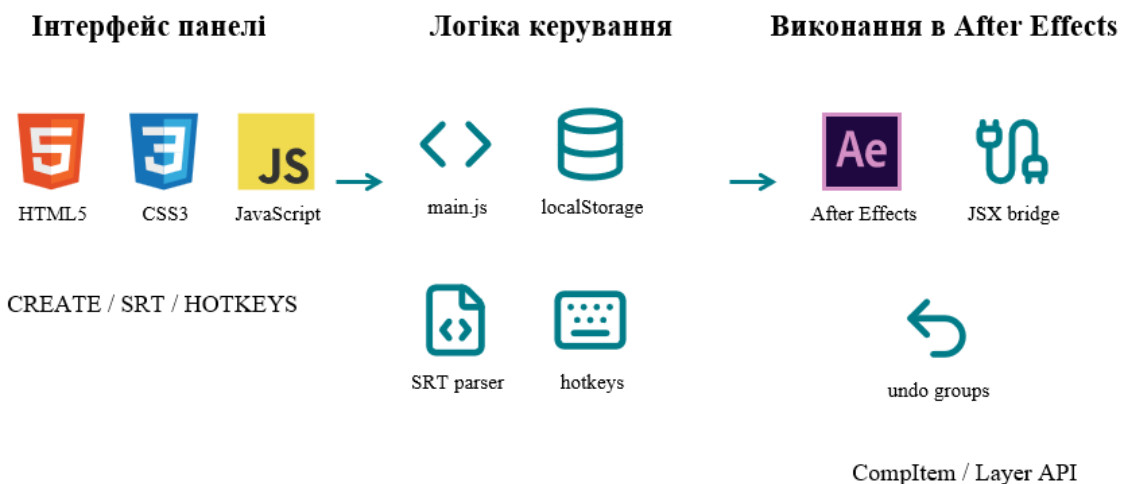
ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

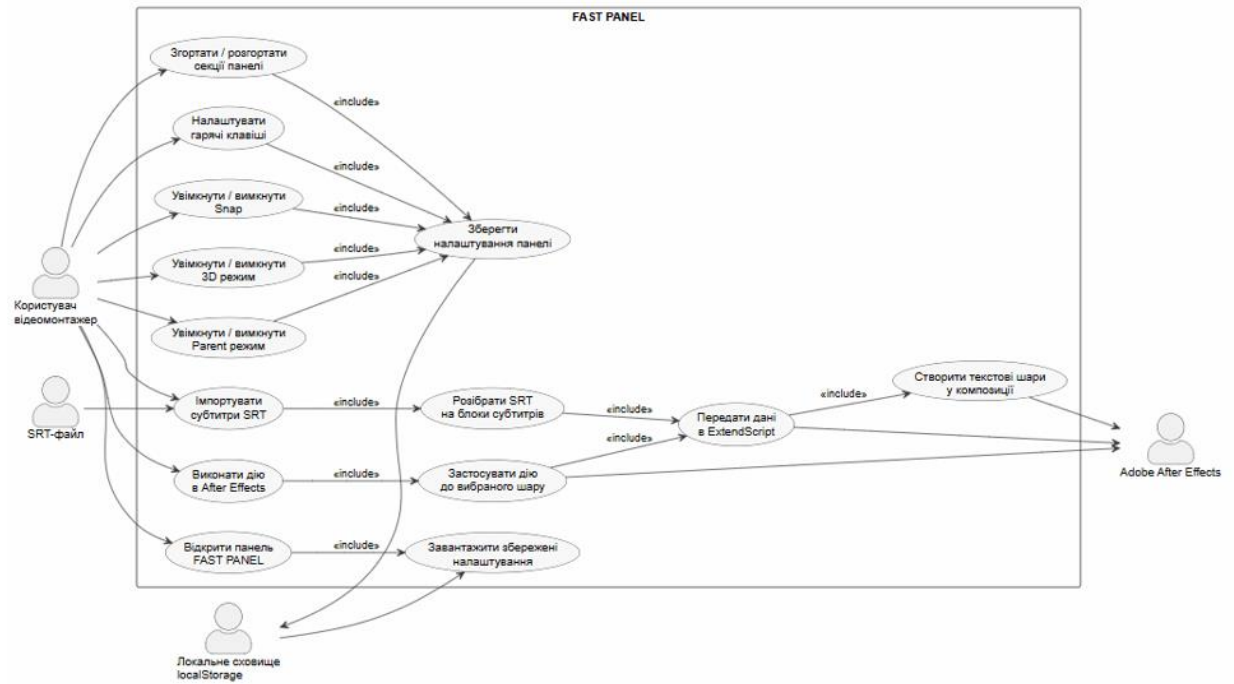
1. створення Solid, Text, Null, Adjustment, Shape, Camera та Light;
2. Trim to Layer, Pack, Fit to Comp і Center in View для швидкої роботи з шарами;
3. зміна Anchor Point у 9 позиціях без візуального зсуву шару;
4. налаштування label colors і користувацьких hotkeys;
5. імпорт SRT-файлів у вигляді текстових шарів з опціями No gaps та Strip punctuation.

5

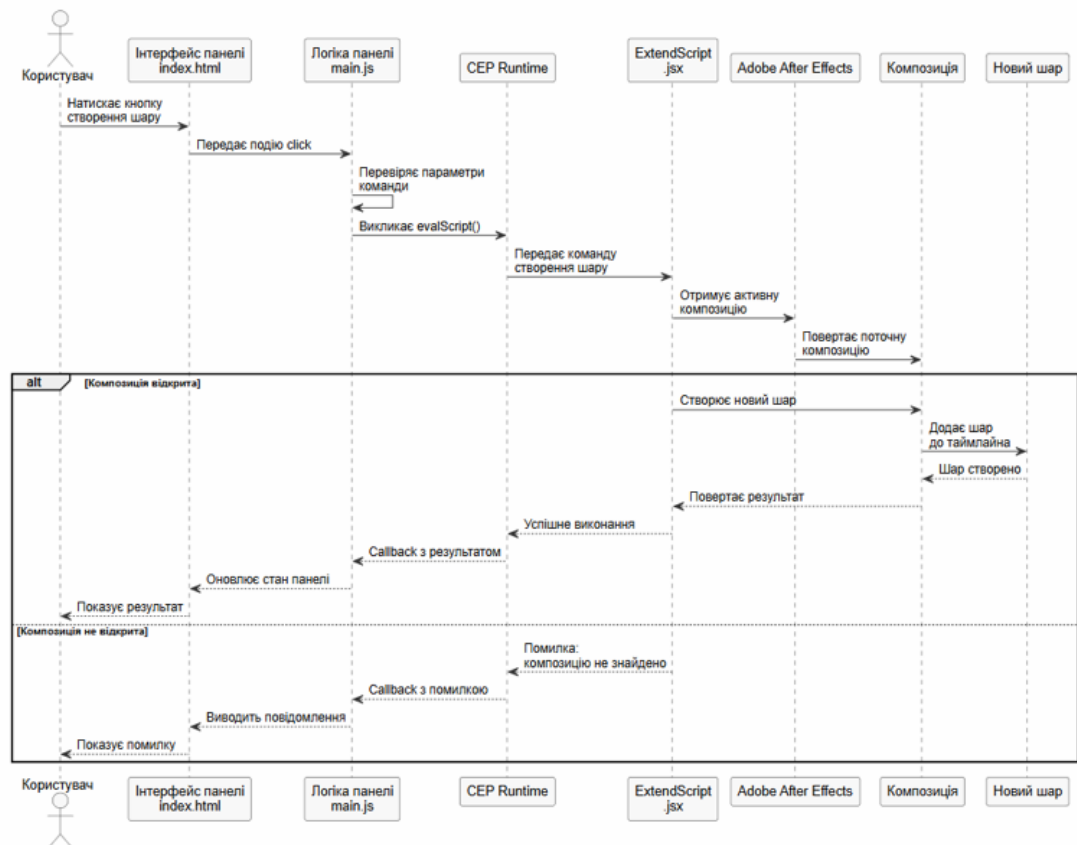
ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



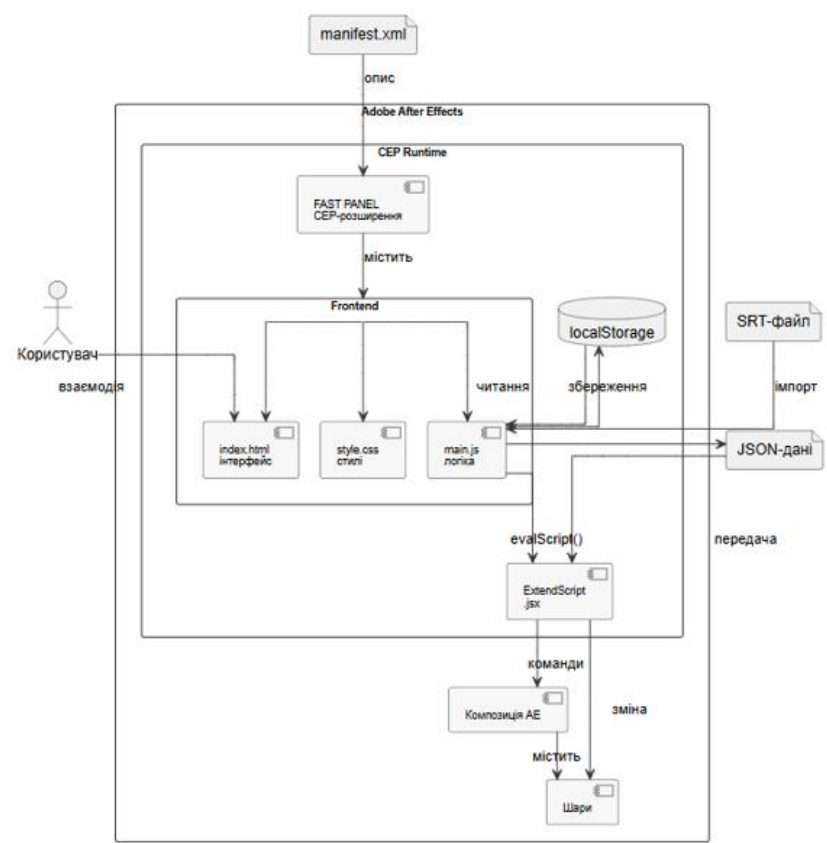
6



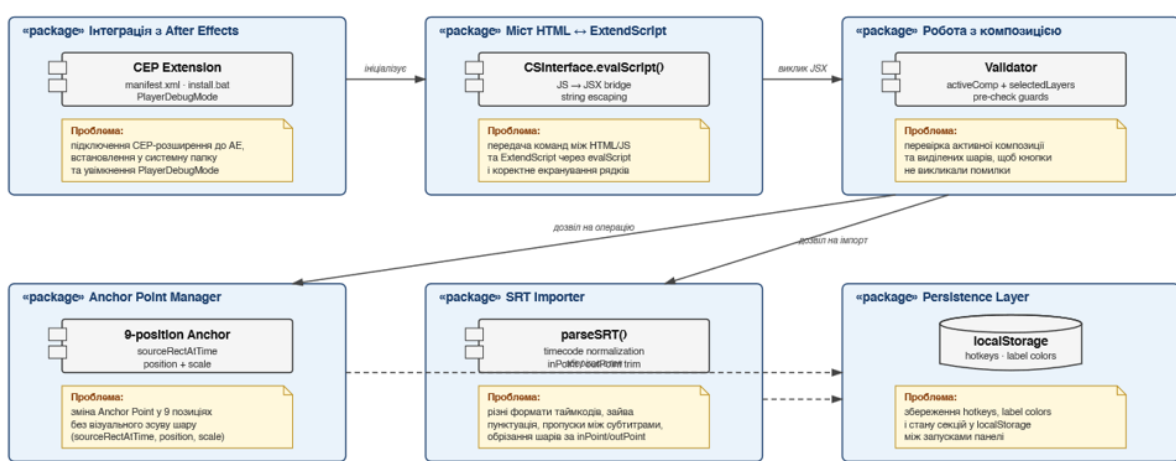
Послідовність створення шару



Архітектура CEP-розширення

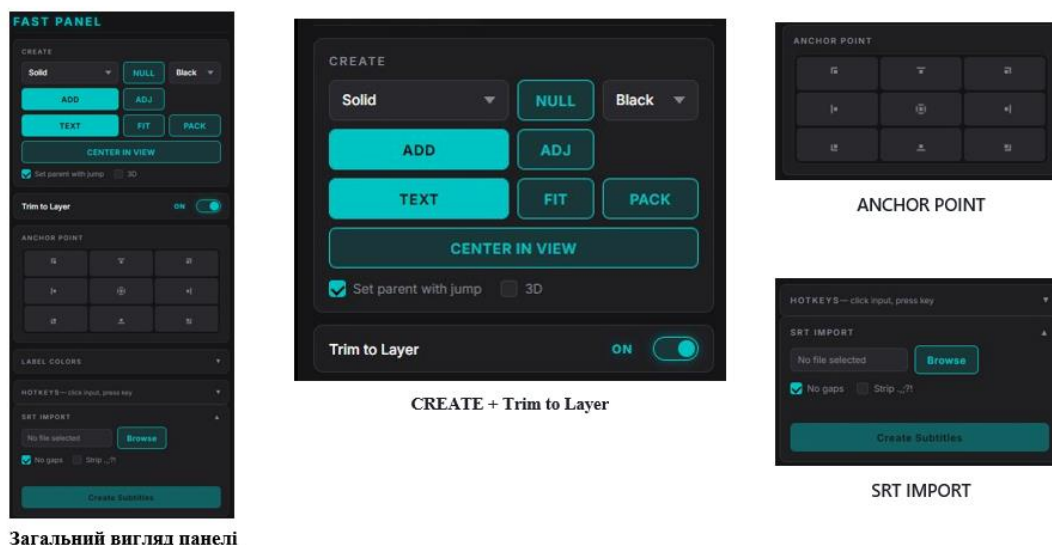


ТРУДНОЩІ ПІД ЧАС РОЗРОБКИ



ІНТЕРФЕЙС FAST PANEL

Скриншоти основних модулів CEP-панелі для автоматизації роботи з шарами та SRT-субтитрами



11

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Попович М.С., Аброскін Ю.Ю. Визначення вимог до скрипту Adobe After Effects для прискорення роботи з об'єктами та анкор-точкою. Матеріали ІІ Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2025. С. 231–234.
2. Попович М.С., Аброскін Ю.Ю. Розробка програмного розширення для Adobe After Effects. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2026. С. 100–103.

13

ВИСНОВКИ

1. Проаналізовано типові операції в Adobe After Effects і визначено, що найбільше часу займають дії зі створення, налаштування та синхронізації шарів.
2. Досліджено структуру CEP-розширень та взаємодію HTML/CSS/JavaScript з ExtendScript через CSInterface і hostscript.jsx.
3. Спроектовано інтерфейс FAST PANEL і визначено основні функції панелі для автоматизації роботи motion-дизайнера.
4. Реалізовано створення шарів, Trim to Layer, Anchor Point, hotkeys, label colors, Pack/Fit/Center та імпорт SRT-субтитрів.
5. Протестовано роботу панелі у типових сценаріях After Effects і перевірено коректність обробки композиції, виділення шарів та undo groups.
6. Підготовлено звітну документацію, презентаційні матеріали та результати апробації для захисту роботи.

ДОДАТОК Б. ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

Б.1 Фрагмент main.js для виклику ExtendScript

```
function execJSX(call) {
    csInterface.evalScript(call, function (r) {
        if (r === 'EvalScript error.')
            console.error('JSX err: ' + call);
    });
}

function flash(btn) {
    btn.style.opacity = '0.6';
    setTimeout(function () { btn.style.opacity = "";
}, 150);
}
```

Б.2 Фрагмент hostscript.jsx для зміни anchor point

```
function getAnchorTargetValue(rect,
anchorMode) {
    var x = rect.left + rect.width / 2;
    var y = rect.top + rect.height / 2;

    if (anchorMode === "top-left" || anchorMode
=== "middle-left" || anchorMode === "bottom-left") {
        x = rect.left;
    } else if (anchorMode === "top-right" ||
anchorMode === "middle-right" || anchorMode ===
"bottom-right") {
        x = rect.left + rect.width;
    }

    if (anchorMode === "top-left" || anchorMode
=== "top-center" || anchorMode === "top-right") {
        y = rect.top;
    } else if (anchorMode === "bottom-left" ||
anchorMode === "bottom-center" || anchorMode ===
"bottom-right") {
        y = rect.top + rect.height;
    }

    return [x, y];
}

function setAnchorForLayer(layer,
anchorMode, compTime) {
```

```
    if (!layer.anchorPoint || !layer.position)
return;

    var rect;
    try {
        rect = layer.sourceRectAtTime(compTime,
false);
    } catch (e) {
        return;
    }
    if (!rect) return;

    var anchor = layer.anchorPoint.value;
    var position = layer.position.value;
    var scale = layer.scale ? layer.scale.value :
[100, 100, 100];
    var target = getAnchorTargetValue(rect,
anchorMode);

    var deltaX = (target[0] - anchor[0]) *
(scale[0] / 100);
    var deltaY = (target[1] - anchor[1]) *
(scale[1] / 100);

    var newAnchor = [target[0], target[1]];
    var newPosition = [position[0] + deltaX,
position[1] + deltaY];

    if (anchor.length > 2) {
        newAnchor.push(anchor[2]);
    }
    if (position.length > 2) {
        newPosition.push(position[2]);
    }

    layer.anchorPoint.setValue(newAnchor);
    layer.position.setValue(newPosition);
}

function setAnchorPoint(anchorMode) {
    app.beginUndoGroup("Set Anchor Point");
    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Select a composition first.");
        app.endUndoGroup();
        return;
    }

    var sel = comp.selectedLayers;
```

```

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

    for (var i = 0; i < sel.length; i++) {
        setAnchorForLayer(sel[i], anchorMode,
comp.time);
    }

    app.endUndoGroup();
}

//
// ALIGN & DISTRIBUTE LAYERS

// Get a layer's bounding box in comp
coordinates (ignores rotation for simplicity)
function getLayerBounds(layer, t) {
    var pos = layer.position.valueAtTime(t,
false);
    var anc = layer.anchorPoint.valueAtTime(t,
false);
    var sc = layer.scale.valueAtTime(t, false);
    var rect = layer.sourceRectAtTime(t, false);

```

Б.3 Фрагмент main.js для парсингу SRT

```

function tcToSec(tc) {
    if (!tc) return 0;
    var parts = tc.replace(',', '.').replace(':',
':').split(':');
    var h = parseFloat(parts[0]) || 0;
    var m = parseFloat(parts[1]) || 0;
    var s = parseFloat(parts[2]) || 0;
    var f = parseFloat(parts[3]) || 0;
    return (h * 3600) + (m * 60) + s + (f / 30);
}

function parseSRT(text, stripPunct) {
    var subs = [];
    text = text.replace(/\r\n/g, '\n').replace(/\r/g,
'\n');
    var lines = text.split('\n');
    var tcRegex =
/^d{1,2}:d{1,2}:d{1,2}[,\.:]d{1,3}\s*--
>\s*d{1,2}:d{1,2}:d{1,2}[,\.:]d{1,3}/;

    var blocks = [];
    var currentBlockLines = [];

```

```

// Group lines into blocks by looking for
Timecode or Index+Timecode
for (var i = 0; i < lines.length; i++) {
    var lineStr = lines[i].trim();
    var isTC = tcRegex.test(lineStr);
    var isIndex = /^d+$/.test(lineStr) && i + 1
< lines.length && tcRegex.test(lines[i + 1]);

    if (isTC || isIndex) {
        if (isIndex) {
            if (currentBlockLines.length > 0)
blocks.push(currentBlockLines);
            currentBlockLines = [lines[i]];
        } else if (isTC) {
            // If it's just a timecode without a
preceding index
            if (currentBlockLines.length === 1
&& /^d+$/.test(currentBlockLines[0].trim())) {
                currentBlockLines.push(lines[i]);
            } else {
                if (currentBlockLines.length > 0)
blocks.push(currentBlockLines);
                currentBlockLines = [lines[i]];
            }
        } else {
            if (currentBlockLines.length > 0) {
                currentBlockLines.push(lines[i]);
            }
        }
    }
    if (currentBlockLines.length > 0) {
        blocks.push(currentBlockLines);
    }
}

for (var i = 0; i < blocks.length; i++) {
    var bLines = blocks[i];
    var tcLineIdx = -1;
    for (var j = 0; j < bLines.length; j++) {
        if (tcRegex.test(bLines[j])) {
            tcLineIdx = j;
            break;
        }
    }
    if (tcLineIdx < 0) continue;

    var tcParts = bLines[tcLineIdx].split('-->');
    if (tcParts.length < 2) continue;

    var inT = tcToSec(tcParts[0].trim());
    var outT =
tcToSec(tcParts[1].replace(/<[>]+>/g, '').trim()).split('
')[0]);

    var textLines = [];

```

```

        for (var k = tcLineIdx + 1; k <
bLines.length; k++) {
            var t = bLines[k].replace(/<[^>]+>/g,
").trim();
            if (t) textLines.push(t);
        }
        var subText = textLines.join(' ');

        if (stripPunct) {
            subText = subText.replace(/[,;?!]/g, "");
        }

        subText = subText.trim();
        if (!subText) continue;

        subs.push({ index: subs.length + 1,
inTime: inT, outTime: outT, text: subText });
    }
    return subs;
}

```

Б.4 Фрагмент hostscript.jsx для створення SRT-шарів

```

function importSRT(subsJson) {
    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Select a composition first.");
        return "error:nocomp";
    }

    // Parse JSON via eval (ExtendScript
compatible)
    var subs;
    try {
        subs = eval("(" + subsJson + ")");
    } catch (e) {
        alert("Failed to parse subtitle data: " +
e.toString());
        return "error:parse";
    }

    if (!subs || subs.length === 0) {
        alert("No subtitles found in data.");
        return "error:empty";
    }

    app.beginUndoGroup("Import SRT
Subtitles");

    var created = 0;
    for (var i = 0; i < subs.length; i++) {
        var sub = subs[i];

```

```

        var inT = sub.inTime;
        var outT = sub.outTime;
        var txt = sub.text;

        // Skip if out of comp range
        if (inT >= comp.duration) continue;
        if (outT > comp.duration) outT =
comp.duration;
        if (outT <= inT) continue;

        try {
            var layer = comp.layers.addText(txt);
            layer.name = txt;

            // Trim layer to subtitle timing
            layer.startTime = 0;
            layer.inPoint = inT;
            layer.outPoint = outT;

            created++;
        } catch (e) {
            // skip problematic layer
        }
    }

    app.endUndoGroup();
    return "ok:" + created;
}

```

Б.5 Фрагмент hostscript.jsx для підготовки нового шару

```

function trimLayerToSelected(newLayer, comp) {
    var target = null;

    // 1. Try to find a selected layer (that is not the new
layer)
    var sel = comp.selectedLayers;
    for (var i = 0; i < sel.length; i++) {
        if (sel[i] !== newLayer) {
            target = sel[i];
            break;
        }
    }

    // 2. Fallback: find the topmost layer whose range
covers the current time
    if (target === null) {
        var t = comp.time;
        for (var i = 1; i <= comp.numLayers; i++) {
            var layer = comp.layer(i);
            if (layer !== newLayer && layer.inPoint <= t
&& layer.outPoint >= t) {
                target = layer;
                break; // layer(1) = topmost in stack
            }
        }
    }
}

```

```

    }

    if (target !== null) {
        // Position new layer just above the target in the
stack
        newLayer.moveBefore(target);
        // Match the target's startTime so in/out trimming
anchors correctly
        newLayer.startTime = target.startTime;
        // Trim to exactly the same visible range
        newLayer.inPoint = target.inPoint;
        newLayer.outPoint = target.outPoint;
    }
    // If still no target found: layer stays at default (full
comp)
}

// — Helper: apply 3D —
function apply3D(layer, is3D) {
    if (is3D === "true") {
        try { layer.threeDLayer = true; } catch (e) {}
    }
}

// — Helper: set parent with jump —
function applyParent(newLayer, comp, doParent) {
    if (doParent !== "true") return;
    var sel = comp.selectedLayers;
    for (var i = 0; i < sel.length; i++) {
        if (sel[i] !== newLayer) {
            newLayer.parent = sel[i];
            break;
        }
    }
}

// — Parse color string "r,g,b" → array —
function parseColor(colorStr) {
    var parts = colorStr.split(",");
    return [parseFloat(parts[0]), parseFloat(parts[1]),
parseFloat(parts[2])];
}

```

Б.6 Фрагмент hostscript.jsx для створення Solid, Null та Adjustment Layer

```

// — CREATE SOLID —
function createSolid(snapMode, is3D, doParent,
colorStr) {
    app.beginUndoGroup("Create Solid");
    var comp = app.project.activeItem;
    if (comp && comp instanceof CompItem) {
        var color = colorStr ? parseColor(colorStr) : [0.2,
0.2, 0.2];
        var solid = comp.layers.addSolid(color, "Solid",
comp.width, comp.height, comp.pixelAspect, comp.duration);
        if (snapMode === "true")
trimLayerToSelected(solid, comp);
        apply3D(solid, is3D);
        applyParent(solid, comp, doParent);
    }
}

```

```

        solid.selected = true;
    } else {
        alert("Select a composition first.");
    }
    app.endUndoGroup();
}

// — CREATE NULL —
function createNull(snapMode, is3D, doParent) {
    app.beginUndoGroup("Create Null");
    var comp = app.project.activeItem;
    if (comp && comp instanceof CompItem) {
        var nulLayer =
comp.layers.addNull(comp.duration);
        if (snapMode === "true")
trimLayerToSelected(nulLayer, comp);
        apply3D(nulLayer, is3D);
        applyParent(nulLayer, comp, doParent);
        nulLayer.selected = true;
    } else {
        alert("Select a composition first.");
    }
    app.endUndoGroup();
}

// — CREATE ADJUSTMENT LAYER —
function createAdjLayer(snapMode, is3D, doParent) {
    app.beginUndoGroup("Create Adjustment Layer");
    var comp = app.project.activeItem;
    if (comp && comp instanceof CompItem) {
        var adj = comp.layers.addSolid([1, 1, 1],
"Adjustment Layer", comp.width, comp.height, comp.pixelAspect,
comp.duration);
        adj.adjustmentLayer = true;
        if (snapMode === "true") trimLayerToSelected(adj,
comp);
        apply3D(adj, is3D);
        applyParent(adj, comp, doParent);
        adj.selected = true;
    } else {
        alert("Select a composition first.");
    }
    app.endUndoGroup();
}

```

Б.7 Фрагмент hostscript.jsx для створення камери, світла та тексту

```

// — CREATE CAMERA —
function createCamera(snapMode, is3D, doParent) {
    app.beginUndoGroup("Create Camera");
    var comp = app.project.activeItem;
    if (comp && comp instanceof CompItem) {
        var cam = comp.layers.addCamera("Camera 1",
[comp.width / 2, comp.height / 2]);
        applyParent(cam, comp, doParent);
        cam.selected = true;
    } else {
        alert("Select a composition first.");
    }
    app.endUndoGroup();
}

```

```
// — CREATE LIGHT —
function createLight(snapMode, is3D, doParent) {
  app.beginUndoGroup("Create Light");
  var comp = app.project.activeItem;
  if (comp && comp instanceof CompItem) {
    var light = comp.layers.addLight("Light 1",
[comp.width / 2, comp.height / 2]);
    applyParent(light, comp, doParent);
    light.selected = true;
  } else {
    alert("Select a composition first.");
  }
  app.endUndoGroup();
}

// — CREATE TEXT —
function createText(snapMode, is3D, doParent) {
  app.beginUndoGroup("Create Text");
  var comp = app.project.activeItem;
  if (comp && comp instanceof CompItem) {
    var txt = comp.layers.addText("New Text");
    if (snapMode === "true") trimLayerToSelected(txt,
comp);

    apply3D(txt, is3D);
    applyParent(txt, comp, doParent);
    txt.selected = true;
  } else {
    alert("Select a composition first.");
  }
  app.endUndoGroup();
}
```

Б.8 Фрагмент hostscript.jsx для підгонки шарів до композиції

```
// — FIT TO COMP: trim selected layers to comp
duration —
function fitToComp() {
  app.beginUndoGroup("Fit to Comp");
  var comp = app.project.activeItem;
  if (comp && comp instanceof CompItem) {
    var sel = comp.selectedLayers;
    if (sel.length === 0) {
      alert("Select at least one layer.");
    } else {
      for (var i = 0; i < sel.length; i++) {
        sel[i].inPoint = 0;
        sel[i].outPoint = comp.duration;
      }
    }
  } else {
    alert("Select a composition first.");
  }
  app.endUndoGroup();
}
```

Б.9 Фрагмент hostscript.jsx для вирівнювання шарів

```
// Get a layer's bounding box in comp coordinates
(ignore rotation for simplicity)
function getLayerBounds(layer, t) {
  var pos = layer.position.valueAtTime(t, false);
  var anc = layer.anchorPoint.valueAtTime(t, false);
  var sc = layer.scale.valueAtTime(t, false);
  var rect = layer.sourceRectAtTime(t, false);

  var scX = sc[0] / 100;
  var scY = sc[1] / 100;

  var left = pos[0] + (rect.left - anc[0]) * scX;
  var top = pos[1] + (rect.top - anc[1]) * scY;
  var right = left + rect.width * scX;
  var bottom = top + rect.height * scY;

  return {
    left: left,
    top: top,
    right: right,
    bottom: bottom,
    cx: (left + right) / 2,
    cy: (top + bottom) / 2,
    width: right - left,
    height: bottom - top
  };
}

// alignType =
"left"|"hcenter"|"right"|"top"|"vcenter"|"bottom"
// alignTo = "selection"|"comp"
function alignLayers(alignType, alignTo) {
  app.beginUndoGroup("Align Layers");
  var comp = app.project.activeItem;
  if (!comp || !(comp instanceof CompItem)) {
    alert("Select a composition first.");
    app.endUndoGroup();
    return;
  }

  var sel = comp.selectedLayers;
  if (sel.length < 1) {
    alert("Select at least one layer.");
    app.endUndoGroup();
    return;
  }

  var t = comp.time;

  // Gather bounds
  var infos = [];
  for (var i = 0; i < sel.length; i++) {
    infos.push({ layer: sel[i], bounds:
getLayerBounds(sel[i], t) });
  }

  // Determine reference bounds
  var refLeft, refRight, refTop, refBottom, refCX,
refCY;

  if (alignTo === "comp") {
    refLeft = 0;
    refTop = 0;
    refRight = comp.width;
    refBottom = comp.height;
  }
```

```

    } else { // selection
        refLeft = infos[0].bounds.left;
        refTop = infos[0].bounds.top;
        refRight = infos[0].bounds.right;
        refBottom = infos[0].bounds.bottom;
        for (var i = 1; i < infos.length; i++) {
            var b = infos[i].bounds;
            if (b.left < refLeft) refLeft = b.left;
            if (b.top < refTop) refTop = b.top;
            if (b.right > refRight) refRight = b.right;
            if (b.bottom > refBottom) refBottom = b.bottom;
        }
        refCX = (refLeft + refRight) / 2;
        refCY = (refTop + refBottom) / 2;

        // Move each layer
        for (var i = 0; i < infos.length; i++) {
            var layer = infos[i].layer;
            var bounds = infos[i].bounds;
            var pos = layer.position.valueAtTime(t, false);

            var dx = 0, dy = 0;

            if (alignType === "left") { dx = refLeft -
bounds.left; }
            else if (alignType === "hcenter") { dx = refCX -
bounds.cx; }
            else if (alignType === "right") { dx = refRight -
bounds.right; }
            else if (alignType === "top") { dy = refTop -
bounds.top; }
            else if (alignType === "vcenter") { dy = refCY -
bounds.cy; }
            else if (alignType === "bottom") { dy = refBottom
- bounds.bottom; }

            var newPos = [pos[0] + dx, pos[1] + dy];
            if (pos.length > 2) newPos.push(pos[2]); // 3D
            layer.position.setValue(newPos);
        }

        app.endUndoGroup();
    }
}

```

Б.10 Фрагмент hostsript.jsx для рівномірного розподілу шарів

```

// distType =
"left"|"hcenter"|"right"|"top"|"vcenter"|"bottom"
function distributeLayers(distType) {
    app.beginUndoGroup("Distribute Layers");
    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Select a composition first.");
        app.endUndoGroup();
        return;
    }

    var sel = comp.selectedLayers;
    if (sel.length < 3) {
        alert("Select at least 3 layers to distribute.");
    }
}

```

```

        app.endUndoGroup();
        return;
    }

    var t = comp.time;

    var infos = [];
    for (var i = 0; i < sel.length; i++) {
        infos.push({ layer: sel[i], bounds:
getLayerBounds(sel[i], t) });
    }

    // Sort by the relevant edge
    var isH = (distType === "left" || distType ===
"hcenter" || distType === "right");
    infos.sort(function (a, b) {
        var va = distType === "left" ? a.bounds.left :
distType === "hcenter" ? a.bounds.cx :
distType === "right" ? a.bounds.right :
distType === "top" ? a.bounds.top :
distType === "vcenter" ? a.bounds.cy :
a.bounds.bottom;
        var vb = distType === "left" ? b.bounds.left :
distType === "hcenter" ? b.bounds.cx :
distType === "right" ? b.bounds.right :
distType === "top" ? b.bounds.top :
distType === "vcenter" ? b.bounds.cy :
b.bounds.bottom;
        return va - vb;
    });

    var first = infos[0];
    var last = infos[infos.length - 1];

    // Get start/end reference values
    var startVal = distType === "left" ? first.bounds.left :
distType === "hcenter" ? first.bounds.cx :
distType === "right" ? first.bounds.right :
distType === "top" ? first.bounds.top :
distType === "vcenter" ? first.bounds.cy :
first.bounds.bottom;
    var endVal = distType === "left" ? last.bounds.left :
distType === "hcenter" ? last.bounds.cx :
distType === "right" ? last.bounds.right :
distType === "top" ? last.bounds.top :
distType === "vcenter" ? last.bounds.cy :
last.bounds.bottom;

    var step = (endVal - startVal) / (infos.length - 1);

    for (var i = 0; i < infos.length; i++) {
        var layer = infos[i].layer;
        var bounds = infos[i].bounds;
        var pos = layer.position.valueAtTime(t, false);

        var target = startVal + i * step;
        var curVal = distType === "left" ? bounds.left :
distType === "hcenter" ? bounds.cx :
distType === "right" ? bounds.right :
distType === "top" ? bounds.top :
distType === "vcenter" ? bounds.cy :
bounds.bottom;

        var delta = target - curVal;
    }
}

```

```

    var newPos;
    if (isH) {
        newPos = [pos[0] + delta, pos[1]];
    } else {
        newPos = [pos[0], pos[1] + delta];
    }
    if (pos.length > 2) newPos.push(pos[2]);
    layer.position.setValue(newPos);
}

app.endUndoGroup();
}

```

Б.11 Фрагмент hostscript.jsx для ущільнення ключових кадрів

```

// — PACK KEYFRAMES: stretch keyframes so first =
layer inPoint, last = layer outPoint —
function packLayers() {
    app.beginUndoGroup("Pack Keyframes to Layer");
    var comp = app.project.activeItem;
    if (!comp || !(comp instanceof CompItem)) {
        alert("Select a composition first.");
        app.endUndoGroup();
        return;
    }

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

    for (var li = 0; li < sel.length; li++) {
        var layer = sel[li];
        var inT = layer.inPoint;
        var outT = layer.outPoint;
        stretchKeysToRange(layer, inT, outT);
    }
    app.endUndoGroup();
}

// Recursively walk all property groups on a layer and
stretch their keyframes
function stretchKeysToRange(propGroup, inT, outT) {
    for (var i = 1; i <= propGroup.numProperties; i++) {
        var prop = propGroup.property(i);

        // Recurse into subgroups (Transform, ADBE
        Effect Parade, etc.)
        if (prop.numProperties !== undefined &&
            prop.numProperties > 0) {
            stretchKeysToRange(prop, inT, outT);
            continue;
        }

        // Only process properties with 2+ keyframes
        if (!prop.numKeys || prop.numKeys < 2) continue;

        var numKeys = prop.numKeys;
        var firstT = prop.keyTime(1);

```

```

        var lastT = prop.keyTime(numKeys);
        var origSpan = lastT - firstT;
        var newSpan = outT - inT;

        if (origSpan <= 0) continue; // all keys at
        same time → skip
        if (Math.abs(newSpan) < 0.0001) continue;

        // Collect all key data (time + value + easing)
        var keyData = [];
        for (var k = 1; k <= numKeys; k++) {
            var kd = {
                norm: (prop.keyTime(k) - firstT) / origSpan, //
                value: prop.keyValue(k),
                easeIn: null,
                easeOut: null,
                interp: [prop.keyInInterpolationType(k),
                    prop.keyOutInterpolationType(k)]
            };
            try {
                kd.easeIn = prop.keyInTemporalEase(k);
                kd.easeOut = prop.keyOutTemporalEase(k);
            } catch (e) {}
            keyData.push(kd);
        }

        // Delete all existing keys (from last to first to keep
        indices valid)
        for (var k = numKeys; k >= 1; k--) {
            prop.removeKey(k);
        }

        // Re-add at new times, stretched to [inT, outT]
        for (var k = 0; k < keyData.length; k++) {
            var kd = keyData[k];
            var newTime = inT + kd.norm * newSpan;

            var newIdx = prop.addKey(newTime);
            try { prop.setValueAtKey(newIdx, kd.value); }
            catch (e) {}

            // Restore interpolation + easing
            try {
                prop.setInterpolationTypeAtKey(newIdx,
                    kd.interp[0], kd.interp[1]);
            } catch (e) {}
            try {
                if (kd.easeIn && kd.easeOut) {
                    prop.setTemporalEaseAtKey(newIdx,
                        kd.easeIn, kd.easeOut);
                }
            } catch (e) {}
        }
    }
}

```

Б.12 Фрагмент main.js для обробки гарячих клавіш

```

// --- Hotkey inputs ---
var activeInput = null;

```



```
<Width>200</Width>  
</MinSize>  
</Geometry>  
</UI>  
</DispatchInfo>  
</Extension>  
</DispatchInfoList>  
</ExtensionManifest>
```