УДК 004.928:004.43

ЗАСТОСУВАННЯ JAVASCRIPT ВИРАЗІВ ДЛЯ КЕРУВАННЯ 2D-АНІМАЦІЄЮ ПЕРСОНАЖІВ У МУЛЬТИМЕДІЙНИХ ПРОЄКТАХ

[1]Єлизавета ЧУЙКО, [2]Анна МОРОЗОВА

[1]кафедра МСТ, Харківський національний університет радіоелектроніки,
[2]доцент, кафедра МСТ, Харківський національний університет радіоелектроніки

Abstract. *This research explores how JavaScript expressions are used to control 2D character animation in multimedia projects. The study shows how simple code and mathematical rules can automatically create realistic movements and physics, replacing the need for manual keyframing. These procedural methods help optimize performance, reduce file sizes, and make animations highly interactive for modern web applications.*

Keywords: *2D animation, JavaScript expressions, procedural animation, inverse kinematics, policy-space diffusion, reinforcement learning, ECMAScript.*

Сучасні мультимедійні проєкти та інтерактивні вебзастосунки вимагають від розробників вміння знаходити баланс. З одного боку, картинка має бути красивою та складною, а з іншого сторінка чи гра повинна працювати швидко і не зависати. Особливо це важливо сьогодні, коли один і той самий продукт має однаково плавно працювати як на потужних комп'ютерах, так і на звичайних смартфонах. Як показують останні наукові статті, сфера комп'ютерної графіки поступово відмовляється від суто ручної праці художників [1]. Усе частіше розробники використовують код та алгоритми, щоб комп'ютер сам виконував частину роботи. У 2D-анімації цей перехід відбувається завдяки процедурній анімації. Вона керується за допомогою виразів JavaScript. Цей підхід дозволяє створювати складні, фізично правильні та природні рухи в реальному часі, і для цього не потрібно вручну промальовувати чи налаштовувати кожен кадр окремо.

Такий підхід, коли код допомагає створювати графіку, чудово показує себе у різних сферах. Наприклад, вчені розробили системи, які можуть автоматично генерувати тіні та кольори для японських мультфільмів (аніме). Це показує, як алгоритми можуть успішно замінювати довгу ручну роботу, автоматизуючи створення красивого візуального стилю [2]. Подібний принцип роботи з кодом зараз застосовується і в 2D-анімації. Там JavaScript-вирази працюють як невидимий помічник: вони керують об'єктами, імітуючи складні процеси. Так само як спеціальні програми замінюють ручне малювання текстур, математичний код у 2D-графіці дозволяє системі самостійно вираховувати, де має знаходитися рука чи нога персонажа в наступну секунду. Завдяки цьому файли з анімацією стають набагато меншими за розміром, а фінальний продукт завантажується і працює значно швидше.

Одним із найскладніших завдань для аніматорів завжди було створення природних рухів, які підкоряються законам фізики (наприклад, гравітації чи інерції). Сьогодні передові розробки у 3D-графіці використовують новітні математичні моделі Штучного Інтелекту, щоб автоматично генерувати фізику рухів для персонажів. Ці програми можуть створювати безліч нових варіантів рухів, які ідеально підлаштовуються під різних персонажів, при цьому зберігаючи

високу якість картинки без ручного створення ключових кадрів [3]. У світі 2D-анімації розробники застосовують дещо простіший, але дуже схожий за логікою підхід. За допомогою JavaScript-коду вони імітують масу тіла та інерцію, використовуючи формули, що описують звичайний маятник або пружину. Хоча в 2D використовуються простіші інструменти, мета залишається тією ж: створити таку систему, яка сама буде реагувати на рухи без потреби малювати кожен крок. Наприклад, завдяки цьому можна автоматично створити рух волосся, що розвівається, або одягу, що продовжує рухатися після того, як персонаж зупинився.

Ще одна велика проблема це навчити персонажа реалістично ходити. Сучасні складні системи вміють генерувати такі рухи завдяки так званому «навчанню з підкріпленням». Тобто комп'ютер сам вчиться, як правильно переставляти ноги, не копіюючи рухи живих акторів (як це роблять за допомогою датчиків руху). Такий підхід дозволяє персонажам на двох ногах підлаштовуватися під будь-яку поверхню, не втрачати рівновагу і точно крокувати, і все це без використання заздалегідь записаних рухів [4]. Коли ж створюють 2D-персонажів, цей же принцип автоматичного контролю ніг працює завдяки «інверсній кінематиці». Якщо говорити просто, то замість того, щоб вручну згинати стегно, потім коліно, а потім стопу, аніматор лише рухає стопу. А код сам вираховує, під яким кутом має зігнутися коліно, щоб нога виглядала природно [4]. Цей автоматичний підхід особливо рятує, коли персонаж знаходиться на сайті і має реагувати на непередбачувані дії користувача, наприклад, коли він має повертатися за курсором миші.

Отже, можна сказати, що використання JavaScript-виразів для керування 2D-анімацією повністю відповідає головним трендам сучасних комп'ютерних технологій. Використання коду, щоб комп'ютер сам вираховував положення суглобів та симулював фізику, дозволяє 2D-аніматорам досягати значних результатів. Ці результати дуже схожі на те, що роблять найсучасніші алгоритми для розфарбовування кадрів, автоматичної генерації фізичних рухів та навчання персонажів ходьбі. Все це перетворює створення анімації з довгої ручної роботи художника на сучасне, гнучке та зручне завдання для розробника.

Список літератури:

1. Bashford-Rogers, T., Meneveaux, D., Ziat, M., Ammi, M., Jänicke, S., Purchase, H., ... & de Sousa, A. A. (Eds.). (2026). *Computer Vision, Imaging and Computer Graphics Theory and Applications: 19th International Joint Conference, VISIGRAPP 2024, Rome, Italy, February 27–29, 2024, Revised Selected Papers*. Springer Nature.

2. Koyama, Y., Tsuruta, T., Saito, H., Takizawa, D., & Moriguchi, H. (2021). A Procedural MatCap System for Cel-Shaded Japanese Animation Production. In *SIGGRAPH Asia 2021 Posters* (pp. 1-2).

3. Rocca, M., Darkner, S., Erleben, K., & Andrews, S. (2025). Policy-space diffusion for physics-based character animation. *ACM Transactions on Graphics*, *44*(3), 1-18.

4. Matthé, T., Ioannidis, N., & van de Panne, M. (2025). Walk This Way: Imitation-free Reinforcement Learning of Flexibly-Constrained Walking Controllers. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, *8*(4), 1-19.