Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Курсова робота

освітній ступінь – бакалавр

на тему: «**Розробка редактора анімацій з використання можливостей HTML 5**»

Виконав: студент 3-го року навчання,

Спеціальності
122 «Комп’ютерні науки»

Студентки Щербатюк Аліни Олексіївна

Керівник Афонін А.О.

кандидат технічних наук, доцент

«6» червня 2022 р.

Київ – 2022

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Освітній ступінь бакалавр

Спеціальність 122 «Комп‘ютерні науки»

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри мультимедійних систем

Жежерун О. П.

“19” жовтня 2021 року

ЗАВДАННЯ

ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Щербатюк Аліні Олексіївні

1. Тема роботи **«Розробка редактора анімацій з використання можливостей HTML 5»**, керівник роботи Афонін Андрій Олександрович
2. Строк подання студентом роботи 6 червня 2022 р.

3. План роботи

Анотація

Вступ

Розділ 1. Дослідження та аналіз предметної області

1.1 Методи покадрової анімації
1.2 Математична модель допасування додаткових кадрів
1.3 Згладжувальна функція Роберта Пеннера
1.4 Об‘єктна модель документа
1.5 Властивості CSS анімації
1.6 Згладжувальні функції
1.7 Анімація кроками

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	19 жовтня 2021			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	19 жовтеня 2021 – 15 листопада 2021			
3.	Складання плану каліф. роботи та узгодження з науковим керівником	15 листопада 2021			
4.	Написання розділів роботи	15 листопада 2021 – 15 травня 2022			
5.	Проміжний контроль виконання роботи	01 лютого 2022			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	11 січня 2022 – 15 травень 2022			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)	25 січня 2022			
	Розділ 2 (аналітично-дослідницька частина)	01 квітня 2022			
	Розділ 3 (проектно-рекомендаційна частина)	29 квітня 2022			
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	01 червня 2022 – 06 червня 2022			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	17 травня 2022			
9.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	згідно з розкладом роботи ЕК			

Графік узгоджено 19 жовтня 2021 р.

Науковий керівник Афонін Андрій Олександрович

Виконавець курсової роботи Щербатюк Аліна Олексіївна

ЗМІСТ

Анотація

У даній роботі розглядаються особливості використання JavaScript та CSS для створення анімацій та робота з об‘єктною моделлю документа для пришвидшення роботи веб застосунку. Досліджуються також методи покадрової анімації. Описується математична модель допасування додаткових кадрів. Пояснюється чому згладжувальні функції Роберта Пеннера стали такими популярними й використовуються до сьогодні, також показані їх математичні формули. Розглянуто як кривими Безьє можна зобразити згладжувальні функції руху об‘єкта та чим вони відрізняються від лінійних функцій.

Вступ

В сучасному світі анімація набула широкої популярності та почала використовуватись не тільки у мультфільмах, фільмах, рекламах, а також на різноманітних сайтах у якості елементів користувацького інтерфейсу(наприклад анімовані слайди у каруселях), а також як елемент привертання уваги. Існує багато редакторів та створювачів анімації, як для новачків, так і для професіоналів. Прості анімації можна створювати навіть онлайн, коли як складні треба чекати, поди допасуються проміжні кадри і анімація буде виглядати привабливо. Компонуючи декілька об‘єктів можна отримати динамічну картинку, на якій об‘єкти будуть рухатись нелінійно, постійно змінюючи швидкість, як у реальному житті. Існує багато методів, що роблять рухи об‘єктів плавніше та приємніше для людського ока. Рух об‘єкта в анімації описується математично, тобто кожна траєкторія руху – це математичний графік, побудований на певній функції.

Розділ 1. Дослідження та **аналіз предметної області**

1.1 Методи покадрової анімації

Процес анімації – це метод, за допомогою якого імітується рух у послідовності, відображаючи серію картинок чи кадрів. Метод покадрової анімації – це запис значення параметра у певний момент, часовий зсув та повторний запис значень. Комп‘ютер аналізує, які дії виконались між записаними значеннями та заповнює ключові кадри. Зародком цього методу є традиційна ручна анімація. В ній аніматори малювали окремі картинки для найбільш важливих подій, а дії між ними заповнювались іншими картинками. Коли всі зображення зібрані, вони збираються в послідовність для завершення анімації. Відтворена анімація для глядача виглядає як неперервні рухи, адже людське око не розпізнає лише одну картинку, створюючи ілюзію неперервного руху.

1.2 Математична модель допасування додаткових кадрів

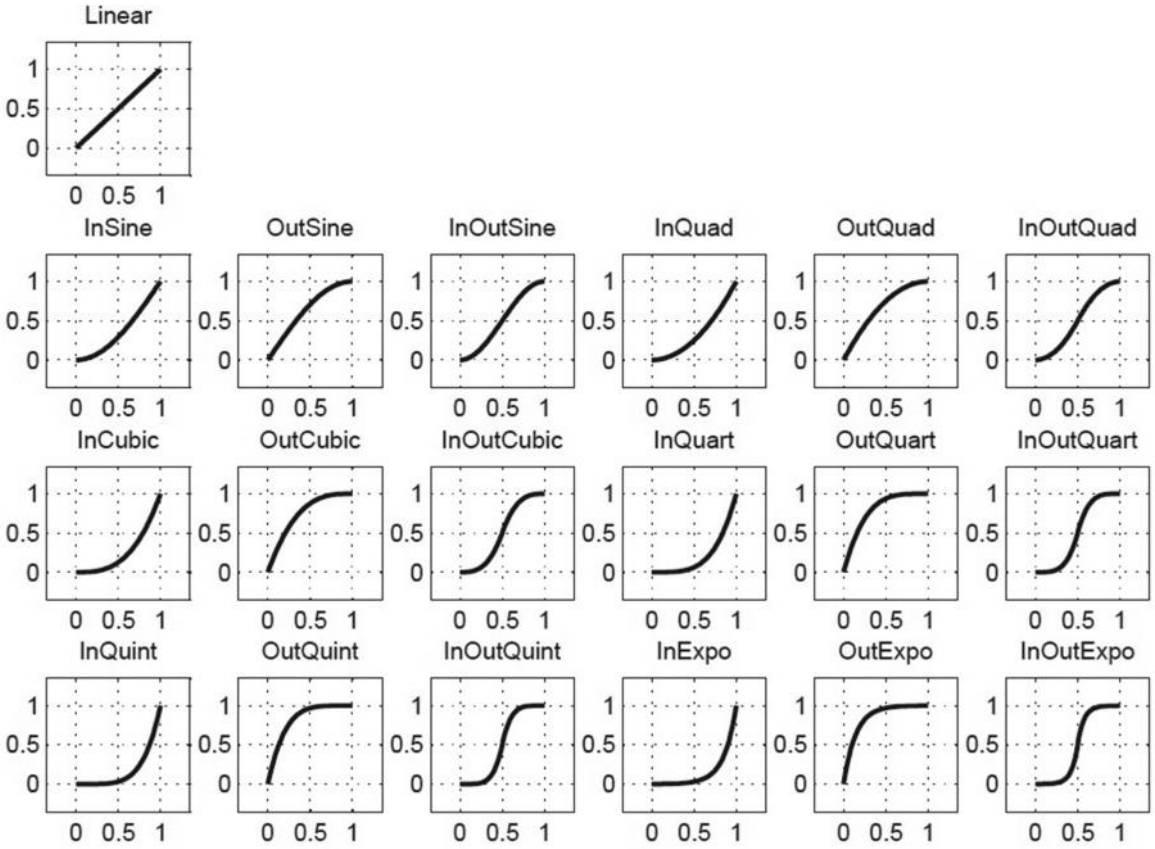
З математичної точки зору кадр – це впорядкована пара (x, y), де x є номером кадру, а y значення анімаційного параметра. Допасування кадрів вираховується комп‘ютерним алгоритмом, який зазвичай є функцією інтерполяції. Інтерполяцією називають спосіб знаходження проміжних значень величин, якщо у нас є лише наявний дискретний набір відомих значень. Нахил функції інтерполяції визначає швидкість зміни параметрів. Тобто, при крутому нахилі зміна значень швидка, а при невеликому нахилі є невеликою.

Для допасування ключових кадрів треба використовувати детерміновану функцію, а також вона має бути представлена у вигляді математичної формули, яка може бути легко інтерпольована центральним процесором.

1.3 Згладжувальна функція Роберта Пеннера

Зміну положення між точкою початку та кінця кадру визначає плавність руху. Найпростіше можна використати лінійну функцію, тоді початок і

кінець будуть з‘єднані прямою лінією, а об‘єкт буде рухатись з лінійною швидкістю. Для людського ока це буде виглядати неприродно. Саме тому краще використовувати динамічне допасування кадрів з нелінійною швидкістю. Автором такого широко використовного методу є Роберт Пеннер. У своїй книзі[1] він виділяє два типи функцій. Функція допасування кадрів, яка описує позицію у певний час, спираючись на початкову позицію, кінцеву позицію і тривалість. Функція згладжування, для контролювання прискорення між станами. Функції згладжування набули більшої популярності, вони використовуються в комп‘ютерній графіці та HTML застосунках, наприклад для переходів у рухах персонажів мультфільмів. Графічний вигляд функцій згладжування можна побачити на Рисунку 1.



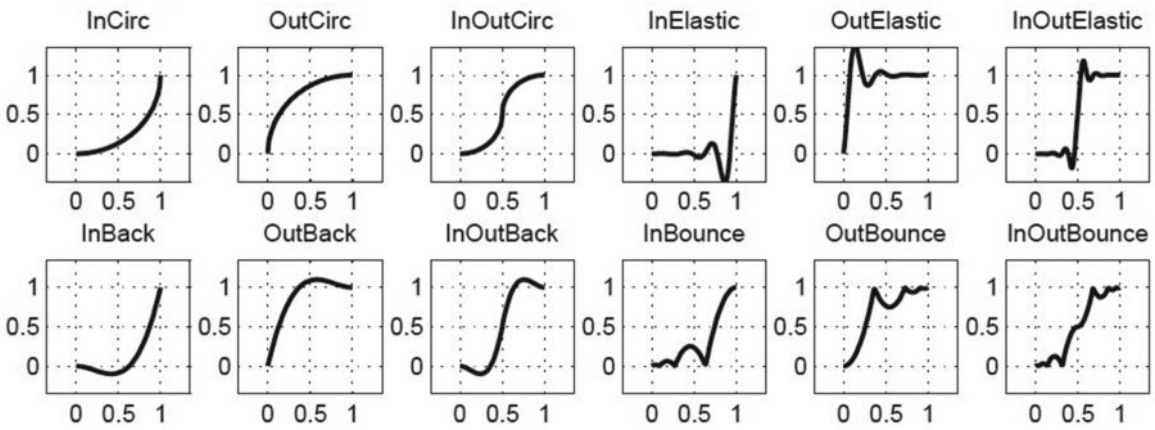


Рисунок 1.

Після досліджень [2], основуючись на реконструкції вихідної програми, можна сказати, що всі функції мають вигляд y=Function(x), де x та y у діапазонах $0 \le x \le 1$ і $0 \le y \le 1$. Математичне представлення цих функцій представлене у Додаток А.

1.4 Об‘єктна модель документа

Об‘єктна модель документа – це інтерфейс прикладного програмування для правильного HTML і добре сформульованих XML документів. Вона визначає логічну структуру документів, спосіб доступу та управління ним. XML представляє дані у вигляді документів, тоді об‘єктна модель документа використовується для керування цими даними. За допомогою об'єктної моделі документа доступним є створення документа, орієнтування в структурі, додавання, змінювання або видаляння елементів та вмісту. Але існують винятки, наприклад, інтерфейси об‘єктної моделі документа для внутрішнього та зовнішнього підмножин XML ще не визначені. [3]Важливо зазначити, що одним з найважливіших завдань об‘єктної моделі документа є забезпечення стандартного інтерфейсу програмування. Це надає можливість використання у різних середовищах і додатках. Також об‘єктну модель документа розроблено для використання з будь-якою мовою програмування. Для забезпечення цього визначено

конкретні специфікації, адже стандартна мова сценаріїв базується на JavaScript та JScript. Об'єктна модель документа може бути реалізована в будь-якому обчислювальному середовищі, і не вимагає прив'язки об'єктів до середовища виконання.

Для прикладу[3], розглянемо таблицю, взяту з HTML документу:

```
<TABLE>
<TBODY>
< TR>
<TD>Shady Grove</TD>
<TD>Aleolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Aleolian</TD>
</TR>
<TBODY>
<TABLE>
```

На Рисунку 2 можна побачити репрезентацію об‘єктної моделі даної таблиці.

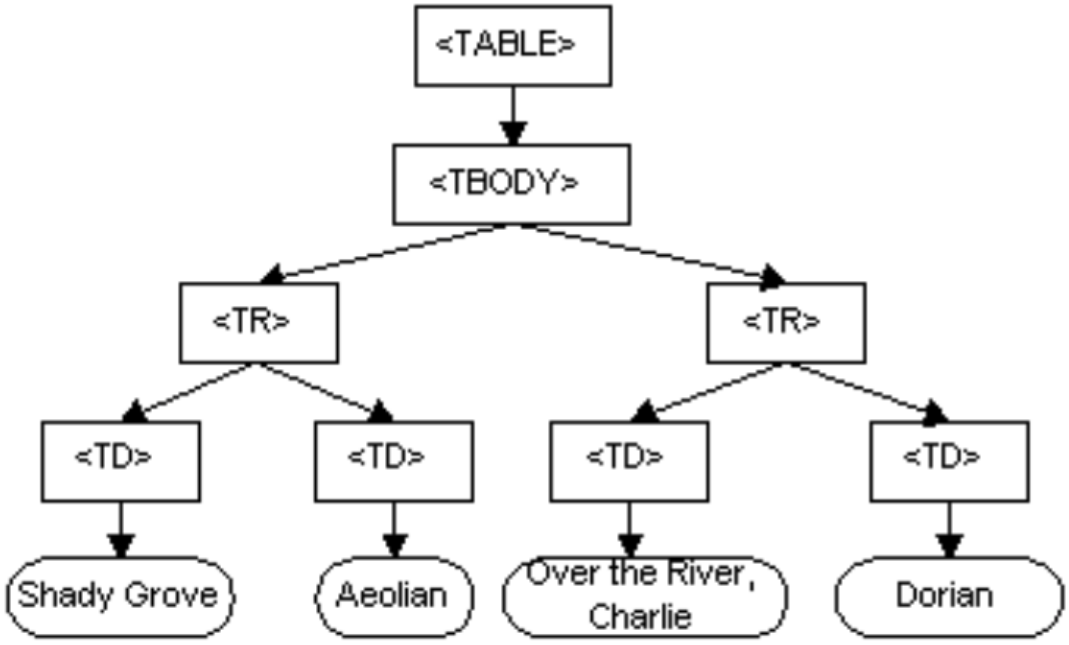


Рисунок 2

Об‘єктна модель документа має логічну структуру і формі дерева. Але таких дерев у документі може бути декілька. Об‘єктна модель документа не визначає як повинні бути визначені відносини між об‘єктами, адже це є логічною моделлю і може бути реалізована будь-яким зручним способом. У специфікацію використовується модель термінологічної структури даних для опису деревоподібного представлення документа.

Одним із важливих властивостей структурних моделей об‘єктної моделі даних є ізоморфізм. Тобто якщо будь-які дві реалізації об‘єктної моделі даних використовуються для створення представлення одного і того ж документа, то вони створять ту саму структурну модель відповідно до XML[4].

Документи модулюються за допомогою об‘єктів, а модель охоплює не лише структуру документа, а й поведінку документів та об‘єктів, з яких він складається. Тобто вузли на Рисунку 2 є об‘єктами, а не структурами даних. Такі об‘єкти мають свої функції та ідентичність.

Об‘єктна модель документа ідентифікує:

- Інтерфейси та об‘єкти, що використовуються для представлення та маніпуляцій з документом;
- Семантику цих інтерфейсів та об‘єктів, включаючи поведінку та атрибути;
- Відносини та взаємодію між цими інтерфейсами та об‘єктами.

1.4 Властивості CSS анімації

CSS, перекладається як каскадні таблиці стилів – це спеціальна мова стилізації сторінок, створена для описання їх зовнішнього вигляду. CSS містить в собі такі властивості як кольори об‘єкта, його границі та розмір. З розвитком мови з‘явились модулі CSS3 Transforms, Transitions і Animation. Це розширення синтаксису CSS, дозволяють контенту на сторінці не бути

статичним, а змінюватись з плином часу. Ці модулі є універсальними та працюють у будь-яких браузерах, що дозволяє полегшити розробку.

CSS Transforms – це функції перетворення. Існує 4 головні функції перетворення:

- rotate(обертання);
- scale(масштабування);
- translate(трансляція);
- skew(нахил).

Обертання перетворює зображення, повертаючи його. Аргументом для обертання можуть бути градуси, гради, оберти та радіани. Також можна використовувати як додатні значення, так і від‘ємні.

Масштабування може бути може бути застосовано до будь-якого HTML-елементу. Наприклад, до тексту, тоді зміна ширина абзацу призведе до перетікання текстового вмісту, але зміниться масштаб і текст фізично буде виглядати більше або менше. В масштаб ми передаємо параметр множник, тобто якщо ми напишемо scale(2), то елемент збільшиться вдвічі, стане вдвічі ширше і вдвічі вище. Якщо вказати scale(.5), це зменшить висоту і ширину об‘єкта вдвічі, тобто площа об‘єкта зменшиться в чотири рази. Також, масштабування можна виконувати по координатах x, y, z. Масштабування можна використовувати, щоб дзеркально зображати елемент. Тобто, якщо оригінальний масштаб починається з 1, як показано на Рисунку 3[5] ліворуч, то елемент буде зменшуватись поки аргумент не стане 0, коли зображення зникне. Якщо аргументом передати значення зі знаком мінус, то зображення почне знову рости, але буде виглядати перевернутим по діагоналі, як показано на правій частині Рисунка 3.

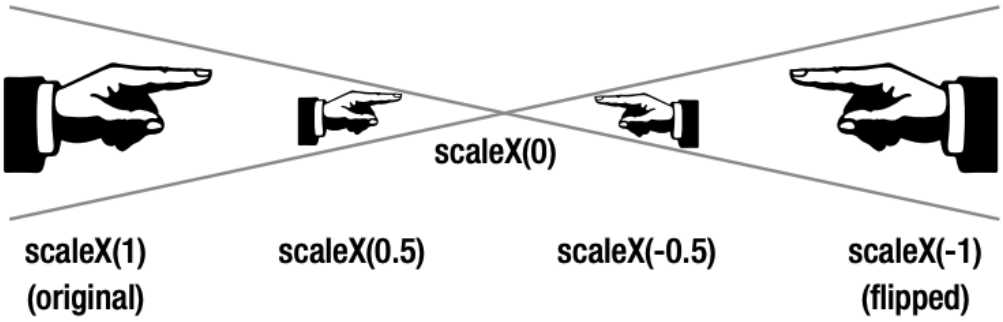


Рисунок 3

Такий спосіб набагато легший і швидший, аніж додаткове використання графічних редакторів, завантаження нового файлу, що потребує часової затрати.

Транслювання, як і масштабування працює з тими ж координатами. За його допомогою можна переміщати елемент по горизонталі та вертикалі, при цьому, використовуючи додатні та від‘ємні значення. Тобто, translateX() – переміщає по горизонталі, а translateY() – по вертикалі.

Нахил елементу здвигає його по горизонталі й вертикалі. Таким ефектом можна показувати швидкість або рух об‘єкта. Аргументом цієї властивості є значення кута у градусах. На прикладі Рисунка 4 можна побачити, що протилежні сторони прямокутника зміщаються, але залишаються паралельними, тобто наш прямокутник стає паралелепіпедом.

Рисунок 4

Нахил правої частини Рисунка 4 виконаний за допомогою skewX(30deg). Це означає, що лівий і правий край зображення будуть встановлені під кутом 30 градусів. Те ж саме зображення можна нахилити так само і в іншу сторону, для цього треба використовувати від‘ємне значення кута, тобто skewX(–30deg). Результат застосування даного ефекту можна побачити на Рисунку 5.

Рисунок 5

Комбінування нахилу по вертикалі та горизонталі допомагає нам зробити ефект того, що ми дивимось на об‘єкт з іншого ракурсу. На прикладі Рисунка 6, видно як буде виглядати елемент при здвигу по горизонталі та вертикалі на 30 градусів. У коді це скорочено буде записано так: skew(-30deg, -30deg);

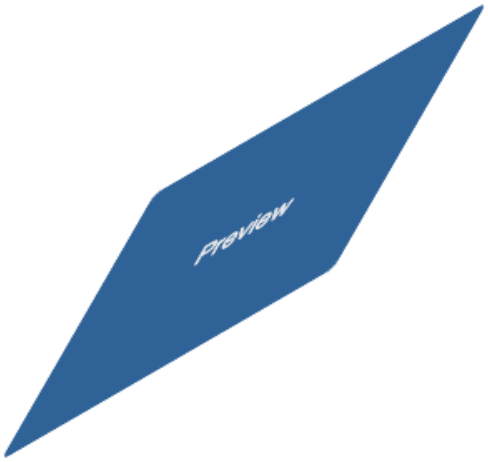


Рисунок 6

CSS Transitions – це найпростіші переходи та форма анімації, тобто рух елементу між двома станами. Перехід від одного візуального стану до іншого, частіше за все відбувається при ініціативі користувача, наприклад при наведенні курсора на елемент. Простіше кажучи, перехід є точковим, тобто є ключові кадри.

1.5 Згладжувальні функції

Якщо при русі об‘єкт рухається плавніше або зі змінною швидкістю – це результат використання згладжувальних функцій. Таким способом рухаються об‘єкти у повсякденному житті. Наприклад, автомобіль збільшує швидкість при старті руху і зменшує перед зупинкою. В каскадній таблиці стилів такий рух використовується за замовчуванням. Але якщо рух треба зробити більш механічним, то використовується лінійний перехід, де швидкість є сталою. Тобто у коді це буде записано так transition: 2s transform linear; Це означає, що об‘єкт дві секунди буде двигатись лінійно, тобто зі сталою швидкість.

Лінійна та згладжувальна функції зображають нам як об‘єкт дістанеться з точки А в точку Б по прямій лінії. Такі часові функції можуть бути зображені на графіку в математичному виразі, тобто як криві Безьє. Наприклад на Рисунку 7 зображено траєкторію руху елементу під час його переходу від 0 до 15 градусів при лінійних властивостях.

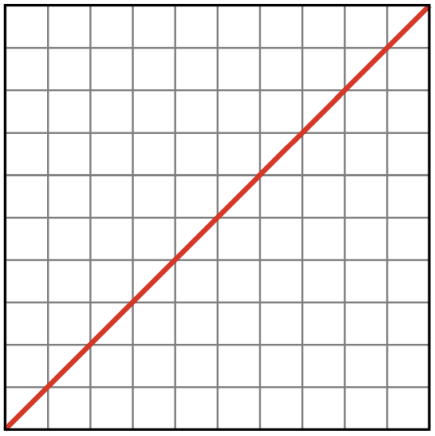


Рисунок 7

Протягом усього заданого часу кут об‘єкта змінюється щосекундно, створюючи постійну швидкість руху.

Прибираючи лінійний параметр руху, ми повертаємо згладження руху, і при побудові на осях графік буде виглядати як Рисунок 8.

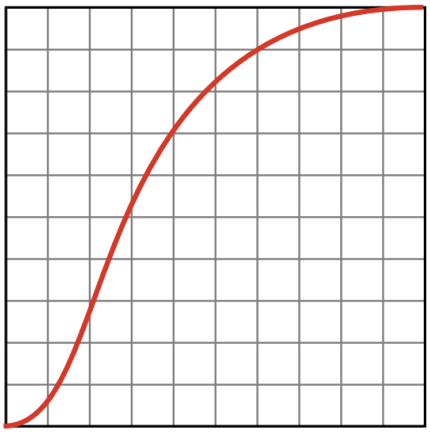


Рисунок 8

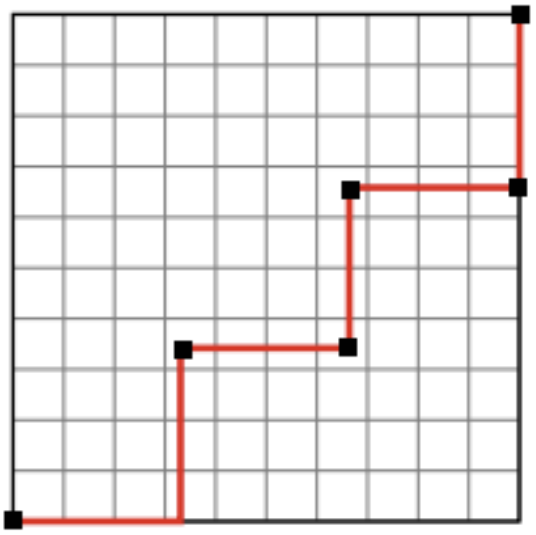
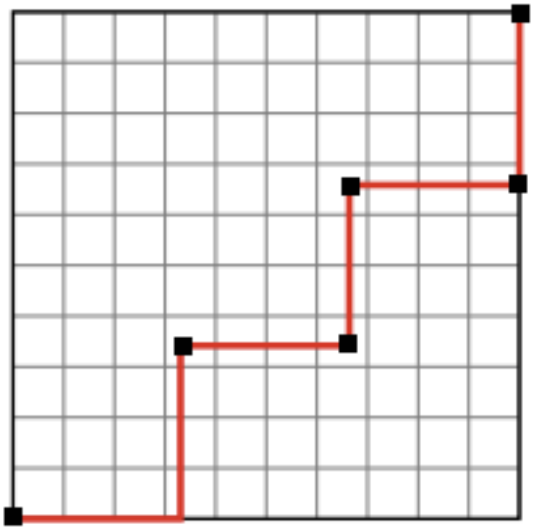
По графіку можна побачити, що кут повороту в перші секунди змінюється повільно, потім ближче до середини швидкість починає зростати, досягає найбільшої швидкості і сповільнюється, поки не вичерпається час руху.

В Додатку Б у вигляді таблиці показані інші криві згладження, які еквівалентні математичним виразам кубічного Безьє:

- пара чисел, в якій кожен набір чисел з плаваючою крапкою описує точку координат простору, створюючи лінію;
- пара чисел, в кожній з яких є набір чисел з плаваючою крапкою описує точку координат простору, створюючи криву переходу.

1.6 Анімація кроками

Окрім плавного руху елементу, можна анімувати його покроково. Прикладом такої анімації є секундна стрілка годинника. Графіки такої анімації можна побачити на Рисунку 9.

Function	Graph
`steps(3)`	
`steps(3), end`	

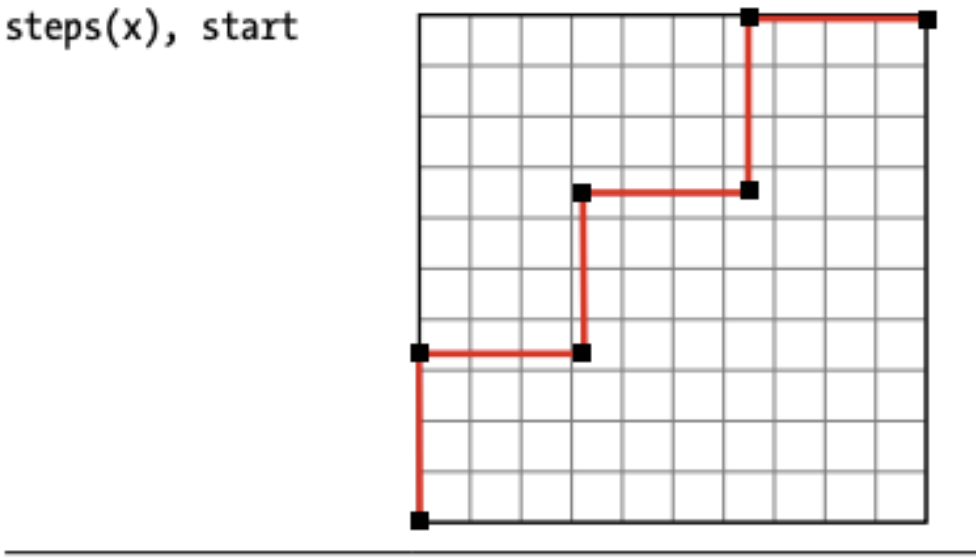


Рисунок 9

Анімація, що складається з 3 кроків показана на першому графіку, де зображено паузу на старті. Аналогічно виглядає і другий графік, де пауза на старті та в кінці. На третьому графіку видно, що без затримки відбувається миттєвий запуск елементу і зупинкою у кінці.

1.8 Редактори анімації

Для створення анімації вручну треба багато годин створення та редагування, щоб отримати певний продукт. Редактори інформації допомагають пришвидшити цей процес для створювача та зробити якісніший продукт. Редактори або створювачі анімації – це програмне забезпечення, що допомагає створювати, налаштовувати та редагувати анімацію. [6]Редактори працюють покадрово, а кадри створюються програмою. На цей час, на ринку є багато програм для створення анімації, кожен з яких має свій набір функцій, що полегшують процес створення анімації. Найбільшою перевагою створювачів анімації є доступність, адже багато з них є безкоштовними або коштують недорого. Це важливо, коли бюджет компанії є обмеженим, а продукт повинен бути виконаний швидко та якісно. Не менш важливою перевагою є те, що зараз відео контент є

популярним способом реклами, а послуги аніматорів можуть коштувати багато, тому іноді легше зробити відеоролик самостійно, не витрачаючи на це фінансові ресурси.

Для використання такого редактора анімації вам треба намалювати або мати створений початковий елемент. На кожному відрізку часу можна змінювати координати певного об‘єкта елементу симулюючи рухи, тим способом, яким ви хочете. Таким чином редактори анімації можуть містити в собі декілька елементів налаштованих на ключових кадрах, як на Рисунку 10. Для людського ока цей рух буде виглядати неперервно і гармонійно, створюючи ефект руху.

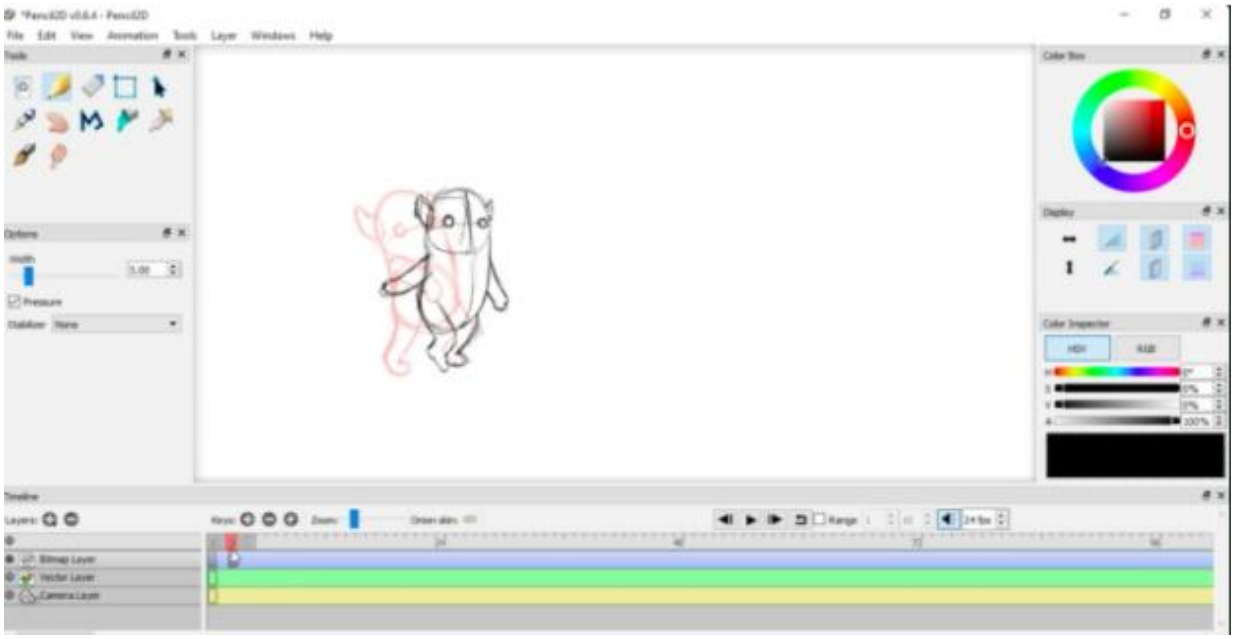


Рисунок 10

Розділ 2. Проєктування та розробка системи

2.1. Опис складових системи та використаних технологій для розробки

Було вирішено зробити редактор анімації, як веб застосунок. Причин цьому декілька. По перше, ринок анімації зараз швидко розвивається, тобто анімацію можна знайти майже на кожному сайті. Тоді контент виглядає динамічніше і приємніше людському оку. По друге, іноді редактори не дають змоги завантажити анімацію, що уже існує для редагування її. Саме тому моя розробка основується анімації, що була до цього створена, яку можна змінювати та редагувати динамічно, одразу спостерігаючи результат.

Клієнтська частина була розроблена за допомогою JavaScript-фреймворку React. Це один з найпопулярніших та зручних фреймворків для створення клієнтської частини вебзастосунку. Для створення об‘єктів у React використовується мова програмування TypeScript[7]. TypeScript є строго типізованою мовою програмування, яка побудована на основі JavaScript.

2.2 Створення проєкту з використанням React

React[8] – це бібліотека для створення користувацьких інтерфейсів. Бібліотека дозволяє підвищити швидкість завантаження даних, тому користувачу не доводиться довго чекати. Також вона знизила проблему неконтрольованих мутацій, завдяки використанню архітектури FLEX, яка зображена на Рисунку 11. Адже замість того, щоб до кожного об‘єкту приєднувати обробник подій, що викликають оновлення об‘єктної моделі даних, React[9] дає можливість управляти станом компонента єдиним способом. Це диспетчеризація дій, тобто коли змінюється стан сховища, система пропонує компоненту виконати перемальовку.

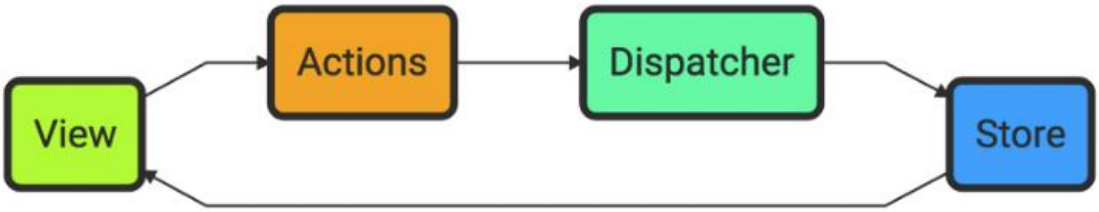


Рисунок 11

Для створення проєкту з використанням React я встановила на комп‘ютер Node.js[9] за допомогою диспетчера пакетів npm. Використавши create-react-app у вказаній мною директорію було створено базовий React-проєкт, зовнішній вигляд можна побачити на Рисунку 12. За допомогою npm install в директорії проєкту виконалось завантаження усіх додаткових залежностей.

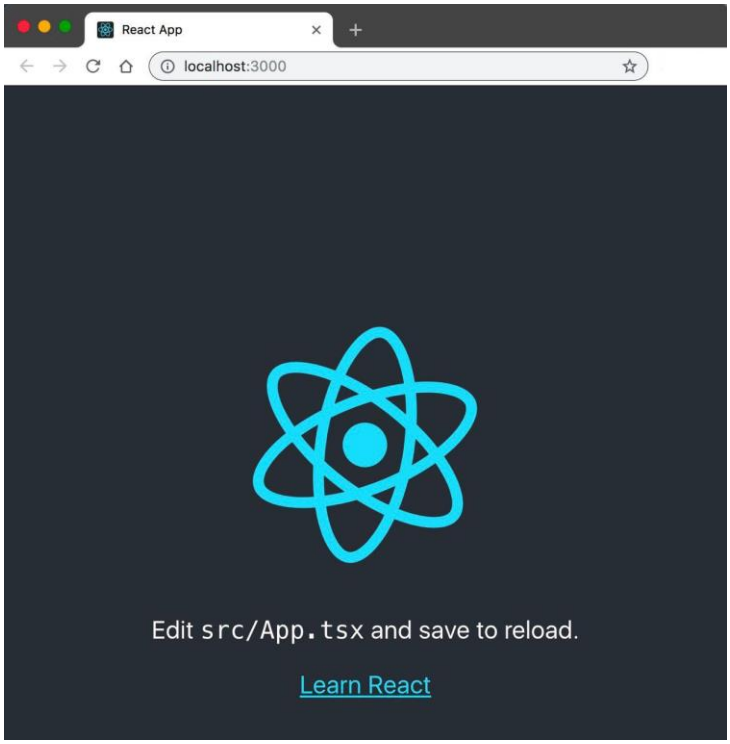


Рисунок 12

В Додатку В я створюю HTML структуру проєкту, де показано елементи, які знаходяться на сторінці вебзастосунку. Всі елементи знаходяться у контейнерах, щоб можна було встановлювати їх стиль та положення.

Бібліотека Scene.js створена для того, щоб можна було полегшити реалізацію анімації. Бібліотека підтримує два види анімації, такі як JavaScript та CSS Animation. Таким чином можливості анімації збільшуються і ви можете створити будь-яким зручним способом для вас.

Основою для створення анімації є клас Scene. Ми створюємо сцену(Рисунок 13), простими словами – полотно, на якому розміщуємо елементи для редагування та саму анімацію.

```
private scene: Scene = new Scene();
```

Рисунок 13

Бібліотека містить в собі клас Timeline, який дозволяє контролювати всі елементи анімації у будь-який проміжок часу. В Додатку Г можна побачити використану бібліотечну шкалу часу з модифікаціями для можливості редагування властивостей елементу у будь-який час. Спочатку я визначила референс, далі вказала сцену, на якій знаходиться шкала часу, визначила її стиль. Ромбики(Рисунок 14) на шкалі часу визначають ключові кадри елементу.

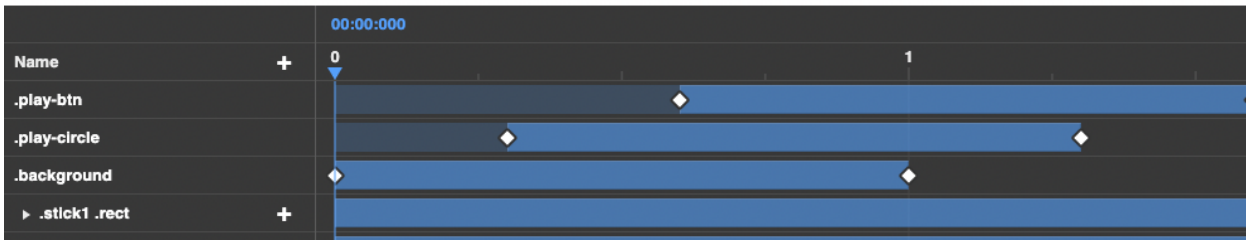


Рисунок 14

Ключові кадри можна зміщати, зманюючи час показу цього кадру. Для цього треба затиснути ромбик і перетягнути його на потрібне місце. Видалити його можна натиснувши на нього і натиснути клавішу Delete.

Запустити анімацію можна або клавішею Пробіл, або натиснувши кнопку плей у вигляді трикутника по центру часової шкали.

Методом onSelect я визначаю, що при натисканні на ключовий кадр об‘єкта з‘являється панель для редагування його властивостей, зображену на Рисунку 15.

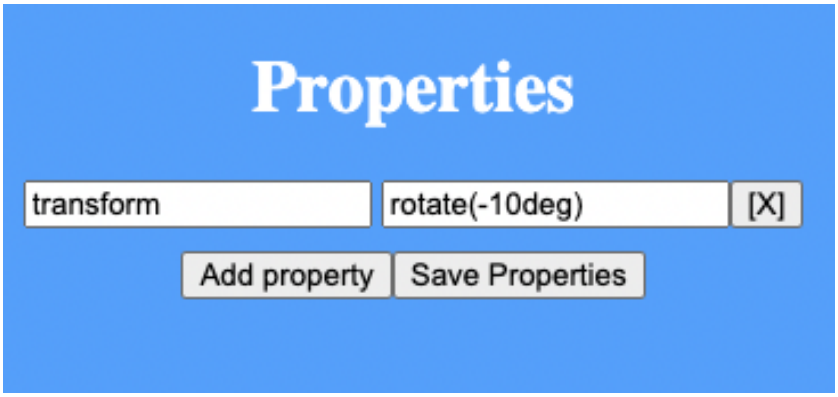


Рисунок 15

На панелі можна побачити уже наявні властивості елементу. Можна додавати нові властивості, видаляти властивості та редагувати властивості. Щоб одразу побачити зміни, треба натиснути кнопку “Save Properties” і вони одразу будуть показані на анімації.

Щоб видалити властивість, треба натиснути на хрестик біля неї. Метод видалення реалізований на основі remove();

Додавання нової властивості реалізоване за допомогою методу показаного в Додатку Ґ. Отримуючи введені дані про назву властивості і її значення, вони додаються до каскадної таблиці стилів елементу на даному ключовому кадрі.

Життєвий цикл React-компонентів створений для захисту стану компоненту. Тобто компонент опиняється у якомусь стані та виводиться на екран. Потім завдяки компонентам життєвого циклу можна додавати до елементу ефекти, впливати на стан, працювати з подіями.

Перший вивід на екран називається монтування(componentDidMount();)[10]. Коли кожен вузол об‘єктної моделі елемента, створений компонентом, видаляється – це називається розмонтування(componentWillUnmount();)[10]. У Додатку Д показано які дані передані для першого виводу на екран. Там передані всі дані для часової шкали для створення ключових кадрів, види анімації, елементи, які беруть участь в анімації.

Висновки

У даній роботі було досліджено та проаналізовано методи покадрової анімації, зокрема математична модель, необхідна для допасування проміжних кадрів між ключовими кадрами. Було описано різноманітні варіації згладжувальної функції руху. Розглянуто об‘єктну модель документа та їх CSS властивості, зокрема ті, які дають можливість створювати анімації, з використанням різноманітних згладжувальних функцій. Було розроблено редактор анімації, котрий дозволяє визначити на часовому проміжку ключові кадри, а згодом, визначивши відповідні значення таблиці стилів, та вказавши необхідні анімації та згладжування, допасувати проміжних кадрів, і реалізувати анімацію. Редактор було реалізовано користуючись сучасним JavaScript фреймворком – React, що додало можливість зручного доопрацювання у майбутньому.

У майбутніх доробках можливо реалізувати також серверну частину, яка б давала можливість додавати нові прошарки та елементи, а також реалізувати зручну функцію експорту на сторонні сайти.

Список використаних джерел

1. Паннер Р.: Motion, tweening, and easing. In: Programming Macromedia Flash MX, розділ 7, 191-240 с. McGraw-Hill/OsborneMedia (2002). Режим доступу: http://robertpenner.com/easing/penner_chapter7_tweening.pdf
2. Іздебський Л., Савікі Д. (2016). Easing Functions in the New Form Based on Bézier Curves. Lecture Notes in Computer Science, 37–48 с. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-319-46418-3_4#citeas
3. Document Object Model (DOM) Level 1 Specification (Second Edition)(2000), 11–12с. Режим доступу: https://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/DOM.pdf
4. Extensible Markup Language(XML)(1998), 5-28с. Режим доступу: http://www.renderx.com/~renderx/Demos/fo2html/xml.pdf
5. Сторней Д.: Pro CSS3 Animation. (2012), розділ 1–2, 1–29с. Режим доступу: https://link.springer.com/content/pdf/bfm%253A978-1-4302-4723-4%252F1.pdf
6. Робертс С.: Characters Animation Fundamentals: Developing Skills for 2D and 3D Character Anination (2011) Розділ 1–2, 10–78с. Режим доступу: https://www.taylorfrancis.com/books/mono/10.4324/9780240522289/character-animation-fundamentals-steve-roberts
7. Бірман Г., Абаді М., Торгерсен М.: Understanding TypeScript(2014), 257-281с. Режим доступу: https://link.springer.com/chapter/10.1007/978-3-662-44202-9_11
8. Гакенхаймер К.: Introduction to React(2015), 2–20с. Режим доступу: https://link.springer.com/content/pdf/bfm%253A978-1-4842-1245-5%252F1.pdf

9. Баш А.: React.js Essentials(2015), 3-43с. Режим доступу: https://books.google.com.ua/books?hl=uk&lr=&id=Rhl1CgAAQBAJ&oi=fnd&pg=PP1&dq=React&ots=JkwtszxWOG&sig=f7eJOOFGEE_xJiICCbjOBdw__uw&redir_esc=y#v=onepage&q=React&f=false
10. Субраманіан В. Pro MERN Stack[React State](2019), 61-84с. Режим доступу: https://link.springer.com/chapter/10.1007/978-1-4842-4391-6_4

Додатки

Додаток А

Математичне представлення функцій згладжування Р. Пеннера

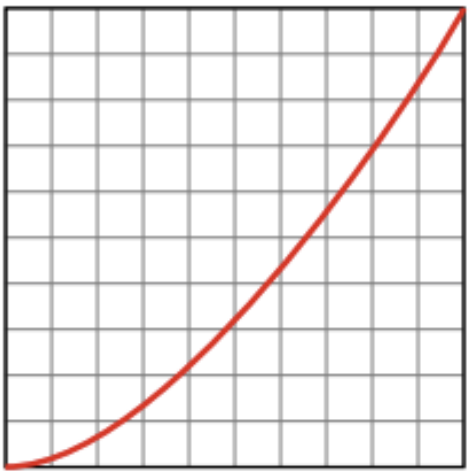
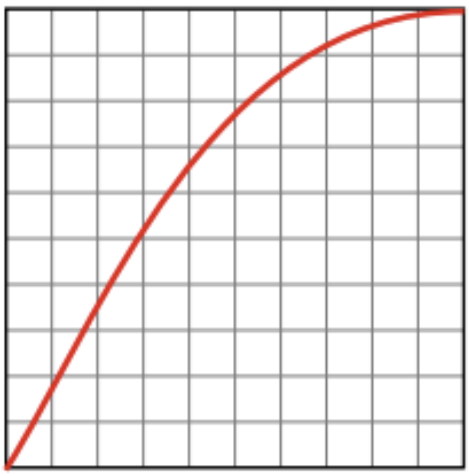
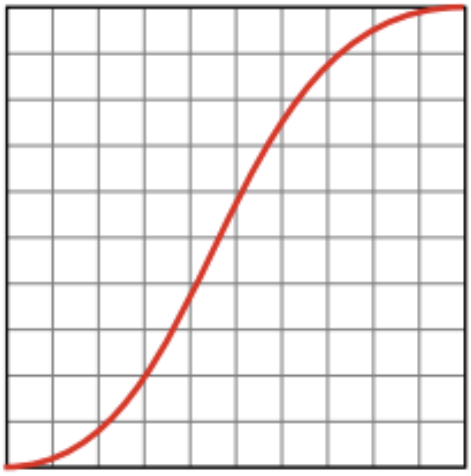
Linear	$y = x \quad x \in [0,1]$
InQuad	$y = x^2 \quad x \in [0,1]$
OutQuad	$y = 1-(x-1)^2 \quad x \in [0,1]$
InOutQuad	$y = \begin{cases} 2x^2 & x \in [0, \frac{1}{2}) \\ 1-2(x-1)^2 & x \in [\frac{1}{2}, 1] \end{cases}$
InCubic	$y = x^3 \quad x \in [0,1]$
OutCubic	$y = 1+(x-1)^3 \quad x \in [0,1]$
InOutCubic	$y = \begin{cases} 4x^3 & x \in [0, \frac{1}{2}) \\ 1+4(x-1)^3 & x \in [\frac{1}{2}, 1] \end{cases}$
InQuart	$y = x^4 \quad x \in [0,1]$
OutQuart	$y = 1-(x-1)^4 \quad x \in [0,1]$
InOutQuart	$y = \begin{cases} 8x^4 & x \in [0, \frac{1}{2}) \\ 1-8(x-1)^4 & x \in [\frac{1}{2}, 1] \end{cases}$
InQuint	$y = x^5 \quad x \in [0,1]$
OutQuint	$y = 1+(x-1)^5 \quad x \in [0,1]$
InOutQuint	$y = \begin{cases} 16x^5 & x \in [0, \frac{1}{2}) \\ 1+16(x-1)^5 & x \in [\frac{1}{2}, 1] \end{cases}$
InSine	$y = 1-\cos(x\frac{\pi}{2}) \quad x \in [0,1]$
OutSine	$y = \sin(x\frac{\pi}{2}) \quad x \in [0,1]$
InOutSine	$y = \frac{1}{2}-\frac{1}{2}\cos(x\pi)$
InExpo	$y = \begin{cases} 0 & x = 0 \\ 2^{10x-10} & (0,1] \end{cases}$
OutExpo	$y = \begin{cases} -2^{-10x}+1 & [0,1) \\ 1 & x = 1 \end{cases}$
InOutExpo	$y = \begin{cases} 0 & x = 0 \\ 2^{20x-11} & (0, \frac{1}{2}] \\ 1-2^{-20x+9} & (\frac{1}{2}, 1) \\ 1 & x = 1 \end{cases}$

InCirc	$y = 1 - \sqrt{1 - x^2} \quad x \in [0, 1]$
OutCirc	$y = \sqrt{1 - (x - 1)^2} \quad x \in [0, 1]$
InOutCirc	$y = \begin{cases} \frac{1}{2} - \frac{1}{2}\sqrt{1 - 4x^2} & x \in [0, \frac{1}{2}) \\ \frac{1}{2} + \frac{1}{2}\sqrt{1 - 4(x - 1)^2} & x \in [\frac{1}{2}, 1] \end{cases}$
InBack	$y = x^2(2.70158x - 1.70158) \quad x \in [0, 1]$
OutBack	$y = (x - 1)^2(2.70158x - 1) + 1 \quad x \in [0, 1]$
InOutBack	$y = \begin{cases} 2x^2(7.189819x - 2.5949095) & x \in [0, \frac{1}{2}) \\ 2(x - 1)^2(7.189819x - 4.5949095) + 1 & x \in [\frac{1}{2}, 1] \end{cases}$
InElastic	$y = \begin{cases} 0 & x = 0 \\ -2^{10x-10}\sin(\frac{20}{3}\pi(x - \frac{43}{40})) & x \in (0, 1) \\ 1 & x = 1 \end{cases}$
OutElastic	$y = \begin{cases} 0 & x = 0 \\ 2^{-10x}\sin(\frac{20}{3}\pi(x - \frac{3}{40})) + 1 & x \in (0, 1) \\ 1 & x = 1 \end{cases}$
InOutElastic	$y = \begin{cases} 0 & x = 0 \\ -2^{20x-11}\sin(\frac{40}{9}\pi(2x - \frac{89}{80})) & x \in (0, \frac{1}{2}] \\ 2^{9-20x}\sin(\frac{40}{9}\pi(2x - \frac{89}{80})) + 1 & x \in (\frac{1}{2}, 1) \\ 1 & x = 1 \end{cases}$
InBounce	$y = \begin{cases} \frac{1}{64} - \frac{121}{16}(\frac{1}{22} - x)^2 & x \in (0, \frac{1}{11}] \\ \frac{1}{16} - \frac{121}{16}(\frac{2}{11} - x)^2 & x \in (\frac{1}{11}, \frac{3}{11}] \\ \frac{1}{4} - \frac{121}{16}(\frac{5}{11} - x)^2 & x \in (\frac{3}{11}, \frac{7}{11}] \\ 1 - \frac{121}{16}(1 - x)^2 & x \in (\frac{7}{11}, 1] \end{cases}$
OutBounce	$y = \begin{cases} \frac{121}{16}x^2 & x \in [0, \frac{4}{11}) \\ \frac{121}{16}(x - \frac{6}{11})^2 + \frac{3}{4} & x \in [\frac{4}{11}, \frac{8}{11}) \\ \frac{121}{16}(x - \frac{9}{11})^2 + \frac{15}{16} & x \in [\frac{8}{11}, \frac{10}{11}) \\ \frac{121}{16}(x - \frac{21}{22})^2 + \frac{63}{64} & x \in [\frac{10}{11}, 1] \end{cases}$

InOutBounce	$y = \begin{cases} \frac{1}{128} - \frac{121}{32}(\frac{1}{22} - 2x)^2 & x \in [0, \frac{1}{22}] \\ \frac{1}{32} - \frac{121}{32}(\frac{2}{11} - 2x)^2 & x \in (\frac{1}{22}, \frac{3}{22}] \\ \frac{1}{8} - \frac{121}{32}(\frac{5}{11} - 2x)^2 & x \in (\frac{3}{22}, \frac{7}{22}] \\ \frac{1}{2} - \frac{121}{32}(1 - 2x)^2 & x \in (\frac{7}{22}, \frac{1}{2}] \\ \frac{1}{2} + \frac{121}{32}(2x - 1)^2 & x \in (\frac{1}{2}, \frac{15}{22}] \\ \frac{7}{8} + \frac{121}{32}(2x - \frac{17}{11})^2 & x \in (\frac{15}{22}, \frac{19}{22}] \\ \frac{31}{32} + \frac{121}{32}(2x - \frac{20}{11})^2 & x \in (\frac{19}{22}, \frac{21}{22}] \\ \frac{127}{128} + \frac{121}{32}(2x - \frac{43}{22})^2 & x \in (\frac{21}{22}, 1] \end{cases}$

Додаток Б

Криві згладження, які еквівалентні математичним виразам кубічного Безьє

Keyword	Graph	Cubic-Bezier
ease-in		0.42, 0, 1, 1
ease-out		0, 0, 0.58, 1
ease-in-out		0.42, 0, 0.58, 1

Додаток В

HTML структура вебінтерфейс

```
<div id="main" className="page page1">
    <div className="container">
        <div className="logo">
            <div className="dash-line"/>
            <div className="dash-line"/>
            <div className="dash-line"/>
            <div className="dash-line"/>
            <div className="clapper">
                <div className="background">
                    <div className="stick stick1">
                        <div className="rect rect1"/>
                        <div className="rect rect2"/>
                        <div className="rect rect3"/>
                        <div className="rect rect4"/>
                        <div className="rect rect5"/>
                        <div className="rect rect6"/>
                    </div>
                    <div className="stick stick1 shadow"/>
                    <div className="stick stick2">
                        <div className="rect rect1"/>
                        <div className="rect rect2"/>
                        <div className="rect rect3"/>
                        <div className="rect rect4"/>
                        <div className="rect rect5"/>
                        <div className="rect rect6"/>
                    </div>
                    <div className="stick stick2 shadow"/>
                    <div className="bottom"/>
                    <div className="bottom shadow"/>
                </div>
                <div className="play-circle"/>
                <div className="play-btn"/>
            </div>
        </div>
    </div>
</div>
```

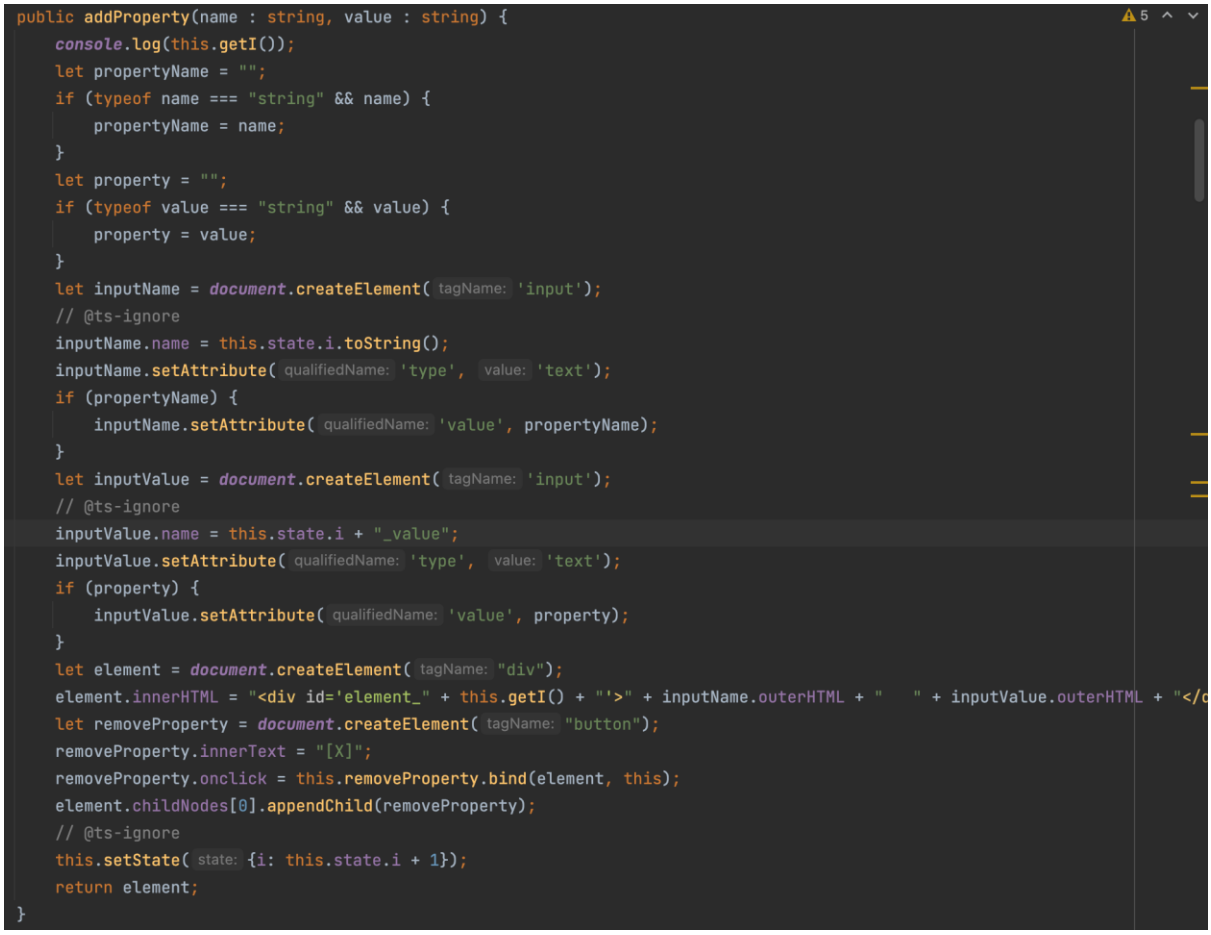
Додаток Г

Використання шкали часу

```
<Timeline
    ref={ref(this, name: "timeline")}
    scene={this.scene}
    style={{ maxHeight: "350px", position: "fixed", bottom: 0, right: 0, left: 0 }}
    onSelect={(e: SelectEvent) => {
        // @ts-ignore
        if (!this.state.isVisible) {
            document.getElementById( elementId: "propertiesParent").style.display = "block";
            // @ts-ignore
            this.state.isVisible = true;
        }
        let time = e.selectedItem.getTime();
        let keyFrame = e.selectedItem.items[time];
        this.setState( state: {currentKeyFrame: e.selectedItem.items[e.selectedItem.getTime()]});
        document.getElementById( elementId: "selectedName").setAttribute( qualifiedName: "value", e.selectedName);

        const myNode = document.getElementById( elementId: "properties");
        while (myNode.lastElementChild) {
            myNode.removeChild(myNode.lastElementChild);
        }
        if (time && keyFrame !== undefined) {

            let frame = new Frame(keyFrame);
            console.log("Frame: "+keyFrame);
            console.log(typeof keyFrame);
            let properties = frame.toCSSObject();

            // @ts-ignore
            this.state.i = 1;
            for(let propertyName in properties ) {
                let property = properties[propertyName];
                myNode.append(this.addProperty(propertyName, property));
                // @ts-ignore
```

Додаток Ґ

Метод додавання властивостей



```
public addProperty(name : string, value : string) {
    console.log(this.getI());
    let propertyName = "";
    if (typeof name === "string" && name) {
        propertyName = name;
    }
    let property = "";
    if (typeof value === "string" && value) {
        property = value;
    }
    let inputName = document.createElement( tagName: 'input');
    // @ts-ignore
    inputName.name = this.state.i.toString();
    inputName.setAttribute( qualifiedName: 'type',  value: 'text');
    if (propertyName) {
        inputName.setAttribute( qualifiedName: 'value', propertyName);
    }
    let inputValue = document.createElement( tagName: 'input');
    // @ts-ignore
    inputValue.name = this.state.i + "_value";
    inputValue.setAttribute( qualifiedName: 'type',  value: 'text');
    if (property) {
        inputValue.setAttribute( qualifiedName: 'value', property);
    }
    let element = document.createElement( tagName: "div");
    element.innerHTML = "<div id='element_" + this.getI() + "'>" + inputName.outerHTML + "    " + inputValue.outerHTML + "</d
    let removeProperty = document.createElement( tagName: "button");
    removeProperty.innerText = "[X]";
    removeProperty.onclick = this.removeProperty.bind(element, this);
    element.childNodes[0].appendChild(removeProperty);
    // @ts-ignore
    this.setState( state: {i: this.state.i + 1});
    return element;
}
```

Додаток Д

Перший вивід елементів на екран

```
public componentDidMount() {
    (window as any).app = this;

    const playBtn = poly( options: {
        width: 60,
        strokeWidth: 8,
        strokeLinejoin: "round",
        rotate: 90,
        origin: "50% 50%",
        left: 10,
        top: 15,
        fill: "#333", stroke: "#333",
    });

    // shadow
    poly( options: {
        width: 60,
        strokeWidth: 8,
        strokeLinejoin: "round",
        rotate: 90,
        origin: "50% 50%",
        left: 20,
        top: 30,
        opacity: 0.2,
        fill: "#333", stroke: "#333",
    }, playBtn);

    document.querySelector( selectors: ".play-btn")!.appendChild(playBtn);

    this.scene.load( properties: {
        ".play-btn": {
            0: {
```

```
this.scene.load( properties: {
    ".play-btn": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
        options: {
            delay: 0.6,
        },
    },
    ".play-circle": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
        options: {
            delay: 0.3,
        },
    },
    ".background": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
    },
    ".stick1 .rect": i => ({
        0: {
            transform: {
```

```
this.scene.load( properties: {
    ".play-btn": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
        options: {
            delay: 0.6,
        },
    },
    ".play-circle": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
        options: {
            delay: 0.3,
        },
    },
    ".background": {
        0: {
            transform: "translate(-50%, -50%) scale(0)",
        },
        1: {
            transform: "scale(1)",
        },
    },
    ".stick1 .rect": i => ({
        0: {
            transform: {
```