Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного інструмента оптимізації процесу створення анімацій у мобільних додатках

Виконав: студент IV курсу, групи СПс-41

спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

_____ Ясінський М.Р.
(підпис) (прізвище та ініціали)

Керівник _____ Гладьо Ю.Б.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент _____ Варавін А.В.
(підпис) (прізвище та ініціали)

Тернопіль - 2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
«__» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

Студенту Ясінському Миколі Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного інструмента оптимізації процесу створення анімацій у мобільних додатках

Керівник роботи Гладьо Юрій Богданович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «18» 04 2025 року № 4/7-336

2. Термін подання студентом завершеної роботи 18.06.2025р.

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1. Огляд предметної галузі.
2. Проектування та розробка інструменту.
3. Результати роботи
4. Безпека життєдіяльності, основи охорони праці.
Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульна сторінка. 2 Актуальність дослідження. 3. Мета, задачі дослідження.
4. Порівняння аналогів. 5. Діаграма класів компонентів бібліотеки.
6. Діаграма архітектури бібліотеки. 7. ПЗ та технології, які використані в роботі.
8. Діаграма сценарію користувача. 9. Діаграма послідовності використання бібліотеки..
10. Діаграма програмного модуля роботи із зображеннями.
11. Діаграми роботи модулів відтворення статичного та динамічного зображень.
12. Результати порівняння показників продуктивності протестованих інструментів.
13. Висновки. Основні результати проведеного дослідження.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Лазарюк В.В., доц.каф. МТ		

7. Дата видачі завдання 18.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	18.04 – 19.04.25	*Виконано*
2.	Підбір джерел про оптимізацію процесу створення		
	анімацій у мобільних додатках	20.04 – 22.04.25	*Виконано*
3.	Опрацювання джерел про оптимізацію процесу		
	створення анімацій у мобільних додатках	23.04 – 29.04.25	*Виконано*
4.	Проведення дослідження щодо оптимізації процесу		
	створення анімацій у мобільних додатках	30.04 – 04.05.25	*Виконано*
5	Розроблення програмного коду	05.05 – 12.05.25	*Виконано*
6.	Оформлення розділу «Огляд предметної галузі»	13.05 – 16.05.25	*Виконано*
7.	Оформлення розділу «Проектування та розробка		*Виконано*
	інструменту»	17.05 – 22.05.25	
8.	Оформлення розділу «Результати роботи»	23.05 – 26.05.25	*Виконано*
9.	Виконання завдання до підрозділу «Безпека		
	життєдіяльності, основи хорони праці»	27.05 – 31.05.25	*Виконано*
10.	Оформлення кваліфікаційної роботи		*Виконано*
11.	Нормоконтроль	01.06 – 05.06.25	*Виконано*
12.	Перевірка на плагіат	06.06 – 09.06.25	*Виконано*
13.	Попередній захист кваліфікаційної роботи	10.06 – 14.06.25	*Виконано*
14.	Захист кваліфікаційної роботи	25.06.25	

Студент _____ Ясінський М.Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Гладьо Ю.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2025, Сторінок 50, рисунків 24, джерел 29.

Ключові слова: криві безьє, парсер, svg/vector зображення, Kotlin, Ondraw, motionevent.

У кваліфікаційній роботі бакалавра розроблено на мові Kotlin бібліотеку під Android для оптимізації процесу застосування зображень, їх анімації та промальовки із використанням формул кривих Безьє.

Виконано огляд предметної області дослідження, проаналізовано способи відтворення інтерфейсів користувача, описано структуру кривих Безьє різного роду. Здійснено порівняння альтернативних програмних рішень за наперед визначеними критеріями.

Наведено особливості архітектури бібліотеки, описані основні сценарії її використання. Наведено специфіку реалізації роботи з SVG та Vector зображеннями. Описано Path-атрибут та реалізовано "парсер" для нього. Представлено особливості статичної та динамічної анімації зображення при допомозі структур кривих Безьє.

Здійснено тестування розробленого рішення. Для забезпечення якісної документації бібліотеки успішно використано декілька підходів (коментування коду, застосування документуючих рядків, створення README -файлу).

ABSTRACT

Bachelor's thesis. Ivan Puluj Ternopil National Technical University, Department of Software Engineering, specialty 121 "Software Engineering". TNTU, 2025, Pages 50, figures 24, sources 29.

Key words: bezier curves, parser, svg/vector images, Kotlin, Ondraw, motionevent.

In the bachelor's thesis, a library for Android was developed in the Kotlin language to optimize the process of using images, their animation and drawing using Bezier curve formulas.

The subject area of the study was reviewed, methods of reproducing user interfaces were analyzed, the structure of Bezier curves of various kinds was described. Alternative software solutions were compared according to predefined criteria.

The features of the library architecture are presented, the main scenarios of its use are described. The specifics of implementing work with SVG and Vector images are presented. The Path attribute is described and a "parser" is implemented for it. The features of static and dynamic image animation using Bezier curve structures are presented.

The developed solution was tested. Several approaches were successfully used to ensure high-quality documentation of the library (code commenting, use of documenting lines, creation of a README file).

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Pprogramming Interface) – інтерфейс програмування застосунків (прикладний програмний інтерфейс).

ImageView, SurfaceView – віджети (widget), які використовуються для розробки програм для платформи Android для відображення зображень.

ObjectAnimator – клас в Android SDK, який дозволяє створювати анімації для різних властивостей об'єкта, таких як переміщення, масштабування, поворот та зміна кольору.

Path – атрибут, який містить команди для переміщення курсору, малювання ліній та кривих, визначає форму контуру, що складається із серії точок, з'єднаних відрізками прямих чи кривих ліній

Parsing (парсинг) – процес аналізу тексту чи даних із метою отримання певної інформації чи структури з вихідних даних.

Systrace – інструмент для діагностики продуктивності в Android. Він записує та відображає події системи.

SVG (англ. Scalable Vector Graphics) – масштабована векторна графіка.

View – базовий клас, який представляє елемент інтерфейсу користувача.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. На даний момент платформа Android включає кілька стандартних функцій для промальовки, наприклад такі як ImageView, SurfaceView та інші. Однак стандартний набір відображення анімованих зображень не може повністю покривати весь функціонал, який потрібно реалізувати розробнику мобільних додатків на платформі Android.

Таким чином, було вирішено створити бібліотеку для оптимізації процесу створення анімацій у мобільних додатках мовою Kotlin. Прийнято рішення створення інструменту у складі бібліотеки, який буде здатний малювати та анімувати інтерактивні та статичні зображення, в якому будуть використовуватися формули кривих Безьє, за допомогою яких зображення перетворюватимуться на масив точок та відображатимуться на екрані користувача.

Метою роботи є розробка програмного інструменту (бібліотеки) для оптимізації процесу використання SVG/Vector зображень, їх анімація та промальовки за допомогою формул кривих Безьє.

Для досягнення мети потрібно вирішити наступні завдання:

– вивчення структур формул кривих Безьє та інструментарію, необхідного для створення та анімування зображень на платформі Android;

– реалізація "парсерів" SVG/Vector зображень;

– реалізація промальовування даних статичних зображень;

– реалізація їх різних анімацій за формулами кривих Безьє;

– документування та тестування бібліотеки для оптимізації процесу використання SVG/Vector зображень, їх анімація та промальовки за допомогою формул кривих Безьє.

1 ОГЛЯД ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Вступна інформація

Іноді для промальовки та анімації об'єкту на основі доступних патернів анімацій бібліотек, окрім ImageView [1], SurfaceView [2], потрібні більш полегшені інструменти, які зможуть відмалювати та анімувати складні, нестандартні інтерактивні або статичні уявлення, що відображені на основі SVG [4] або Vector [5] розширень зображень, і які відповідатимуть усім вимогам та технічним завданням. Так, наприклад, іноді використання одного основного інструменту анімації ObjectAnimator[6] буде проблематичним, оскільки іноді досить складно використовувати певні анімації для взаємодії з користувачем.

Було проаналізовано статистику кількості репозиторіїв, пов'язаних з анімаціями на платформі Android упродовж років, яку можна побачити на рис. 1.1. На підставі статистики [7, 8], кількість репозиторіїв, пов'язаних з анімацією, збільшилася більш ніж на 400% порівняно з 2019 роком і на 170,2% у порівнянні із 2022 роком.

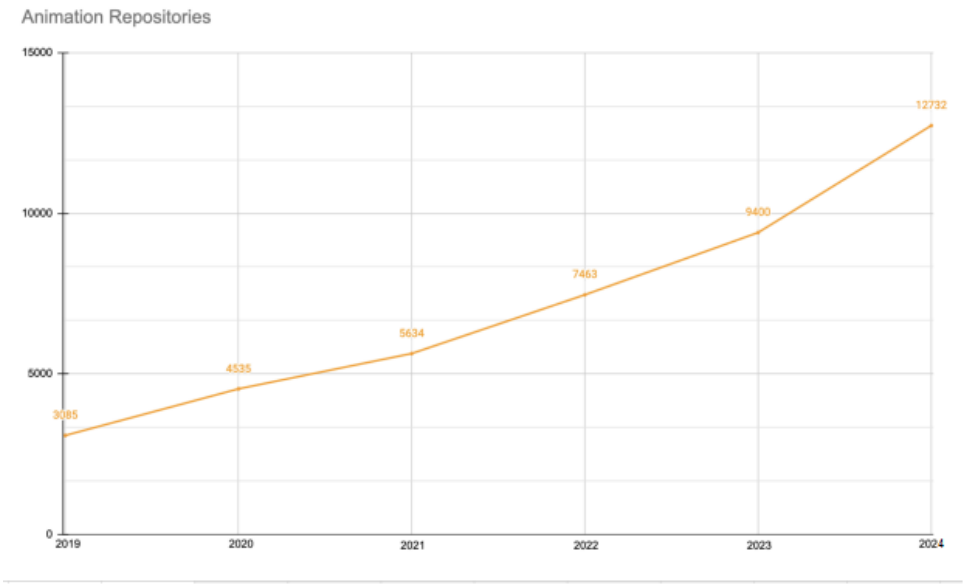

Рисунок 1.1 – Кількість репозиторіїв в часі

Для використання формул кривих Безьє та зручності використання бібліотеки необхідно реалізувати інструмент для “парсингу” SVG зображень у Vector, адже цей процес досить проблематичний і часто доводиться використовувати сторонні ресурси, тому необхідно певне рішення, яке допоможе прискорити як процес “парсингу” SVG зображень у Vector, так і процес відображення та більш плавного анімування View [1].

1.2 Способи відтворення інтерфейсів користувача

У Kotlin існує кілька інструментів для відображення та анімації графіки.

Одним з них є нативний клас View для створення інтерфейсів користувача. За допомогою нього можна створити власний екземпляр класу View і перевизначити безліч методів, наприклад, найбільш базовими є onDraw() і onMeasure(). Цей клас дозволяє розробникам відображати на екрані свої специфічні інтерфейси.

Також розробка мовою Kotlin надає клас Canvas, який містить у собі набір методів для малювання графіки на екрані. За допомогою цього інструменту можна малювати різноманітні лінії, криві, фігури, текст та зображення.

Так, наприклад, можна використовувати клас Canvas для відтворення зображення за певними координатами. Для цього необхідно створити об'єкт даного класу, в ньому викликати drawBitmap() та передати йому: Bitmap (зображення), координати x та y, а також об'єкт Paint (необов'язково). На рис. 1.2 наведено лістинг прикладу програмного коду.

Інструмент OpenGL ES [12] є стандартним API для рендерингу графіки (рис. 1.3). Він може надати розробнику набір низькорівневих функцій для малювання 2D та 3D графіки.

Одним з найпоширеніших методів для створення анімацій є використання класу Animation [13], який містить деякий набір методів для анімації переміщення,

масштабування, обертання та зміни прозорості.

```
// завантажуємо зображення з ресурсів
val bitmap = BitmapFactory.decodeResource
(resources, R.drawable.my image)
// створюємо об'єкт Canvas val canvas = Canvas
(bitmap)
// створюємо об'єкт val paint = Paint()
// малюємо зображення за координатами
Paint canvas.drawBitmap(bitmap, 100f, 100f, paint)
```

Рисунок 1.2 – Приклад коду

Рисунок 1.3 – Логотип OpenGL ES

Property Animation [13] у свою чергу - гнучкий інструмент, який дозволяє анімувати будь-які властивості об'єкта, включаючи колір, форму та розмір. Для його використання необхідно перевизначити метод ValueAnimator та додати об'єкт зображення.

Підклас класу Constraint Layout - Motion Layout [13] є розширеним нативним класом для створення складних анімацій і переходів між екранами. MotionLayout також підтримує ключові кадри, що дозволяє повністю настроювати переходи об'єктів.

Перетворення зображень SVG на VectorDrawable є одним із рутинних занять для анімації за допомогою багатьох інструментів при розробці під систему Android. У свою чергу VectorDrawable дозволяє використовувати векторні зображення в Android -додатках як звичайні зображення і є стандартним класом, який надає можливість відображати векторні зображення у форматі XML.

Для перетворення SVG на VectorDrawable можна використовувати онлайн-конвертери, такі як svg2vector [15] або векторні редактори, наприклад Adobe Illustrator [16].

Для отримання атрибутів зображення VectorDrawable (колір, розмір, форма тощо) можна використовувати методи класу VectorDrawableCompat. Клас VectorDrawableCompat надає зворотну сумісність із VectorDrawable для більш ранніх версій Android, що дозволяє продовжувати розробку досить старих проектів.

Приклад коду для отримання кольору зображення за допомогою інструменту VectorDrawableCompat показано на рис. 1.4:

```
val drawable = ContextCompat.getDrawable(context,
R.drawable.my vector) as
VectorDrawableCompat
val color = drawable.getTint()
```

Рисунок 1.4 – Приклад програмного коду

MotionEvent є класом, який може обробляти події на екрані. Він містить інформацію про координати та тип торкання, час та інші атрибути.

Точки торкання екрана користувачі називаються покажчиками (pointers). Кожна з них має унікальний ідентифікатор, який можна використовувати для відстеження руху торкання екрану девайса.

Щоразу коли користувач торкається до екрану, система створює об'єкт MotionEvent і передає у нього відповідний обробник подій, наприклад метод onTouchEvent() у класі View.

Також клас MotionEvent можна використовувати для анімації та відображення певних зображень. Щоб реалізувати це необхідно використовувати інструменти Canvas та MotionEvent одночасно. Ця комбінація двох інструментів дозволяє розробникам створювати інтерактивні графічні інтерфейси, які будуть реагувати на рух пальців користувача екраном.

1.3 Структури кривих Безьє

У сфері комп'ютерної графіки до створення ліній кривих застосовуються математичні об'єкти - криві Безьє. Даний тип кривих розроблений у Франції у 1960-х роках інженером П'єром Безьє.

Структура кривої Безьє складається з набору контрольних точок, що визначають її форму. Перші та останні контрольні точки називаються вузловими точками і є початком та кінцем лінії кривої, а проміжні контрольні точки називаються ручками. Ручки використовуються для управління напрямом та кривизни кривої.

Усього існує кілька типів кривих Безьє, включаючи лінійні, квадратичні та кубічні криві. Крім того існують деякі складніші форми кривих Безьє, вони називаються кривими Безьє вищого порядку і B-сплайнами та мають більш складні формули, ніж інші типи кривих. Лінійні криві Безьє мають лише дві контрольні точки і є звичайні прямі лінії, тоді як квадратичні мають три контрольні точки, а кубічні чотири, квадратичні і кубічні криві Безьє утворюють дугоподібні лінії, а деяких випадках і специфічні фігури. Кубічні криві Безьє, на відміну від квадратичних, є найбільш поширеними і використовуються в багатьох додатках для створення кривих і плавніших переходів у зображеннях комп'ютерної графіки.

Далі наведені формули для лінійних, квадратичних та кубічних кривих Безьє.

Лінійна крива Безьє (1.1):

$$P(t) = (1 - t) * P0 + t * P1, \quad (1.1)$$

де t – параметр у діапазоні від 0 до 1, P0 та P1 – контрольні точки.

Квадратична крива Безьє (1.2):

$$Рф = (1\text{-}t)2 * P0 + 2t(1\text{-}t) * P1 + t2 * P2. \quad (1.2)$$

Кубічна крива Безьє (1.3):

$$P(t) = (1-t)^3 * P0 + 3t(1-t)^2 * P1 + 3t^2 (1-t) * P2 + t^3 * P3, \quad (1.3)$$

де P0, P1, P2 та P3 – контрольні точки.

Також було побудовано графіки з урахуванням формул кубічної кривої [17] Безьє (рис. 1.5).

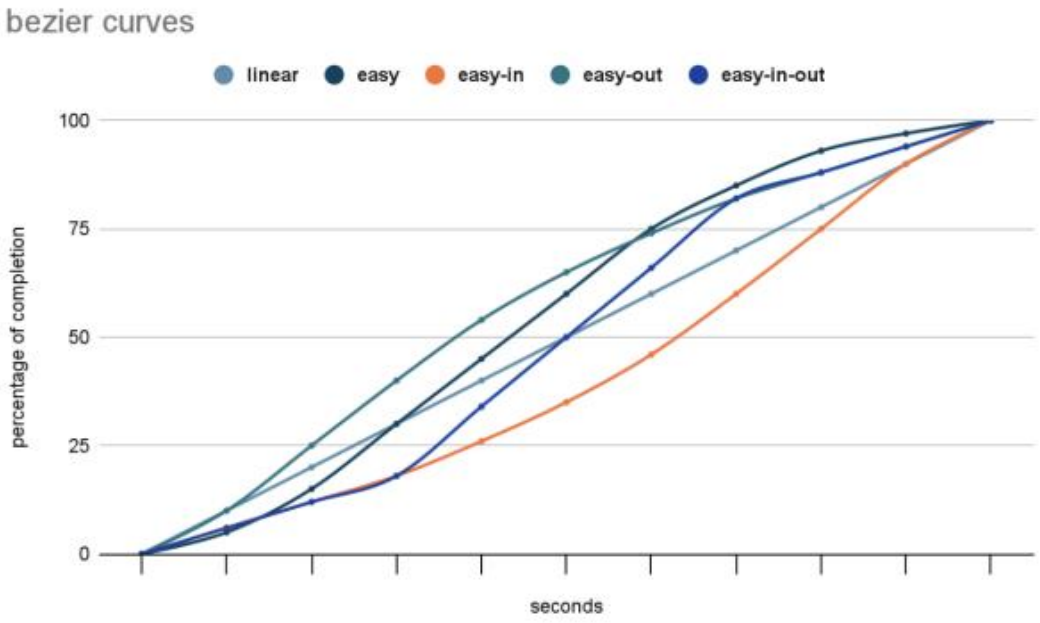

Рисунок 1.5 – Кубічні криві Безьє

Дані формули використовуються для обчислення координат точки P(t) на кривій Безьє в залежності від значення параметра t. Контрольні точки у формулах визначають форму кривої, а параметр t визначає положення точки на кривій.

Криві Безьє також використовуються у векторній графіці, а саме для відображення та масштабування SVG зображень.

Приклад використання кривих Безьє у SVG зображеннях (1.4):

$$d= \text{" M1.04,234.98C1.04,234.88 0.67,226.38 2.17,212.48 ",} \quad (1.4)$$

де C - відповідає за тип кривої Безьє, а всі атрибути, що йдуть за ним, є абсолютними координатами кубічної кривої Безьє (в даному прикладі).

Таким чином, криві Безьє допомагають розробникам створювати більш плавні та природні криві. Що робить їх гарним інструментом для створення графічних зображень, наприклад, логотипів, іконок та шрифтів.

1.4 Альтернативні рішення

Для визначення альтернативного рішення було виділено 6 критеріїв для бібліотек (табл. 1.1).

Таблиця 1.1 - Результати порівняльного аналізу аналогів бібліотек

	1	2	3	4	5	6
Macaw	-	+	+	+	-	+
Elephant	-	+	+	+	-	+
AnimatedSvg View	-	-	+	+	+	-
Leonids	-	+	+	-	+	-
ExplosionField	-	-	+	+	-	-
NineOldAndroids	-	+	+	+	+	-
Android View Animations	+	+	+	-	-	-
Curved Bottom Sheet	+	-	+	+	-	-

1. Використання формул Безьє є одним із ключових критеріїв для реалізації гнучкості в анімації. Створення плавних та природних рухів, які виглядають більш реалістичними для користувачів, високий рівень контролю над анімацією, а саме точніші зміни форми та швидкості руху анімації, легкість масштабування об'єкта, що дозволяють адаптувати анімацію під різні розміри екрана та пристроїв є перевагами використання цих формул.

2. Кількість базових параметрів, варіантів зовнішнього вигляду анімацій, які розробник може налаштувати або вибрати самостійно також є одним із не менш важливих факторів при розробці. Даний критерій дозволяє створювати розробнику такі анімації, які краще відповідають потребам і перевагам користувачів або замовників.

3. Високе навантаження на девайс користувача може позначитися на продуктивності роботи програми в цілому, також це може призвести до уповільнення або зупинки роботи анімації. Оптимізація бібліотеки дозволяє зменшити обсяг використовуваних ресурсів і тим самим знизити дане навантаження на девайс користувача, підвищивши швидкість обробки даних.

4. Нативність бібліотеки під Android забезпечує більш високу продуктивність і швидкість роботи програми і є не менш важливим критерієм у розробці.

5. Документація може підвищити розуміння інструментарію бібліотеки для розробника, що позначається на швидкості розробки програм та збільшенні якості кінцевого продукту.

6. При можливості використовувати SVG і vector зображення в бібліотеці - виключається необхідність застосування зовнішніх сервісів та можливо внутрішніх інструментів, які можуть бути недостатньо гнучкими у використанні.

Також були встановлені критерії для статей (табл. 1.2).

Таблиця 1.2 - Результати порівняльного аналізу аналогів методів/статей

	1	2	3
Lottie	+	-	-
Property Ansmator [18]	-	-	+
View Animation [18]	-	-	+
AnimatedVector Drawable [18]	+	-	-

1. Якщо певний метод досить вимогливий до ресурсів девайсу, то навантаження на девайс користувача зросте, а це може призвести до загального уповільнення роботи, підвищеного енергоспоживання, перегріву та негативного впливу на досвід користувача.

2. Інтерактивність анімацій у свою чергу може покращити даний досвід, підвищують залученість, покращуючи дизайн і навігацію, а також підвищуючи конверсію користувачів програми, що розробляється.

3. Використання зовнішніх ресурсів у розробці може створити проблеми з безпекою та конфіденційністю даних, а також із залежністю та продуктивністю самої програми.

Далі буде розібрано альтернативні рішення.

AnimatedSvgView [19] — це бібліотека для Android, яка дозволяє відображати та анімувати будь-які зображення SVG (рис. 1.6). Бібліотека надає свою власну View, яку можна використовувати у макеті для відображення певного зображення. Також бібліотека підтримує анімацію за допомогою різних вбудованих параметрів. Ця бібліотека написана мовою програмування Kotlin та характеризується простотою використання.

Її переваги:

– відображення анімації відбувається строго за точками, що при використанні складних зображень SVG забезпечує високу точність та плавність

анімації;

– для використання бібліотеки достатньо додати лише одну View до файлу макета, скорочуючи час на налаштування модуля бібліотеки у проекті.

Її недоліки:

– анімація підтримується тільки за допомогою єдиного інструменту, що впливає на функціональність при розробці;

– бібліотека працює тільки з файлами формату SVG, обмежуючи можливість використання інших форматів зображень;

– відсутній парсер координат, що унеможливлює роботу з координатами всередині SVG зображень.

Рисунок 1.6 – Лого AnimatedSvgView

Leonids [20] - бібліотека, написана мовою програмування Java, для розробки Android додатків (рис. 1.7). Надає досить простий API для створення та налаштування систем частинок, включає безліч вбудованих емітерів і модифікаторів.

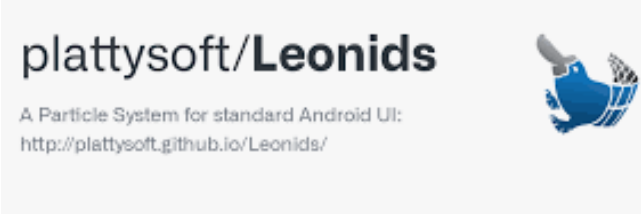

Рисунок 1.7 – Лого Leonids

Переваги бібліотеки:

– багато атрибутів для анімації дозволяє розробникам створювати різноманітні анімації у своїх проектах;

– наявність докладної документації кожного методу може полегшити вивчення та використання бібліотеки розробником у цілому.

Недоліки:

– бібліотека написана мовою програмування Java;

– доступна лише одна модель анімації, що досить обмежує використання бібліотеки у великих проектах.

ExplosionField [21] — бібліотека для розробки під Android (рис. 1.8) для створення анімацій розпаду на точки будь-яких View , яка надає просту API для створення та налаштування ефектів, а також включає безліч вбудованих шаблонів для реалізації. Написана на Java.

Рисунок 1.8 – Логотип ExplosionField

Переваги:

– користувачеві достатньо використовувати лише один інструмент бібліотеки, що полегшує її використання та скорочує час на налаштування;

– наявність докладної документації кожного методу полегшує вивчення та використання бібліотеки для розробників.

Недоліки:

– існування лише єдиної моделі анімації;

– використання застарілих інструментів може негативно позначитися на продуктивності та їх застосуванні у сучасних проектах.

NineOldAndroids [22] - це бібліотека (рис. 1.9), що забезпечує зворотну сумісність для API- інтерфейсів анімації, представлених в Android 3.0

(Honeycomb). Бібліотека дозволяє розробникам використовувати API у старіших версіях Android для створення анімацій, які однаково працюють на різних версіях платформи, включаючи старіші. Бібліотека написана на Java.

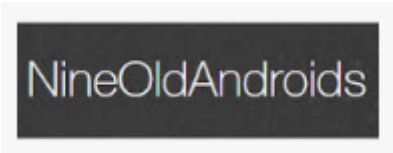

Рисунок 1.9 – Логотип NineOldAndroids

Переваги:

– використання на будь-яких версіях Android може забезпечити створення анімацій, які однаково працюють на різних версіях платформи.

Недоліки:

– написана мовою програмування Java, що обмежує її використання розробниками проектів, які використовують інші мови програмування;

– є застарілою.

AndroidViewAnimations [23] — це бібліотека Android, яка надає різноманітні анімації для View (рис. 1.10). Вона включає широкий спектр даних анімацій, таких як підстрибування, спалах, струшування і багато інших. Ця бібліотека була написана на Java.

Рисунок 1.10 – Логотип AndroidViewAnimations

Переваги:

– наявність величезної кількості атрибутів для анімації дозволяє розробникам створювати більш насичені та яскраві анімації;

– використання «Easing functions» [24] добре позначається на

оптимізації та відображенні більш реалістичного зображення.

Недоліком є те, що відсутність документації ускладнює вивчення та використання цієї бібліотеки.

CurvedBottomSheet [25] - це бібліотека Android, яка надає BottomSheet , що налаштовується, вигнутої форми. Бібліотека дозволяє швидко створювати анімацію BottomSheet з використанням кривих Безьє, а також включає безліч можливостей для налаштування. Бібліотека була написана мовою Kotlin.

Переваги:

– використання формул Безьє для оптимізації та відображення більш реального зображення може підвищити якість анімації та оптимізацію ресурсів девайсу;

– наявність документації полегшує використання бібліотеки.

Недоліком є відсутність варіативності анімацій та їх одноманітність, що може обмежувати функціональність цієї бібліотеки.

2 ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНСТРУМЕНТУ

2.1 Організація архітектури інструменту

В першу чергу перед розробкою програмного рішення було необхідно визначити всі компоненти інтерфейсу бібліотеки, які могли б бути корисні для розробника при реалізації його завдань. На цьому етапі було визначено, які інструменти та можливості будуть доступні для користувача під час роботи з бібліотекою.

Таким чином, були виділені три основні компоненти, які були б найбільш значущими та корисними при використанні програмного рішення. Цими компонентами стали:

– PathParser;

– PathBezier;

– DrawPath.

Компонент PathParser є парсером даних та відповідає за перетворення вхідних даних у різні види координат. Можна сказати, що цей компонент забезпечуватиме правильне відображення даних на екрані користувача.

Наступний компонент PathBezier дозволяє розробнику створювати плавні криві, засновані на координатах PathParser. Він перетворює різні типи координат, використовуючи формули кривих Безьє.

Останній компонент DrawPath буде використовуватися для створення статичних зображень, а також інтерактивних і динамічних анімацій. Таким чином, DrawPath відповідає за відображення та анімацію зображень на екрані користувача.

На рис. 2.1 зображено діаграму класів компонентів бібліотеки.

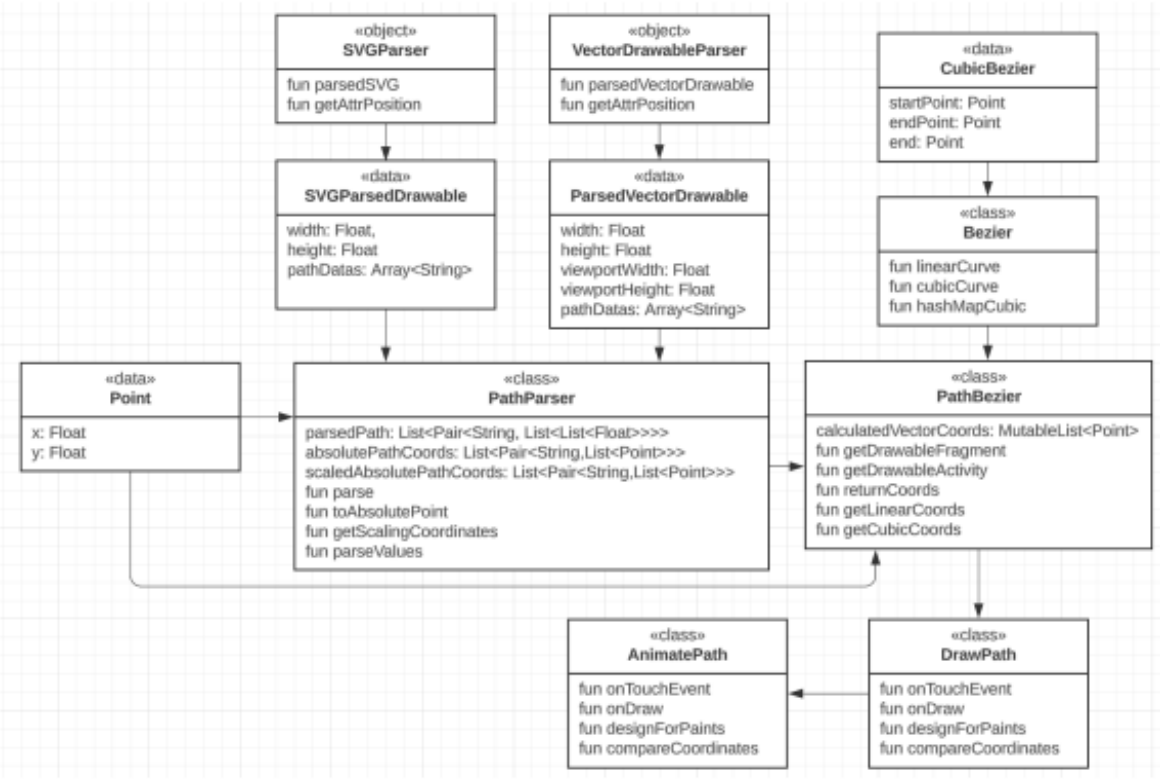

Рисунок 2.1 – Діаграма класів

Також було складено діаграму архітектури бібліотеки (рис. 2.2).

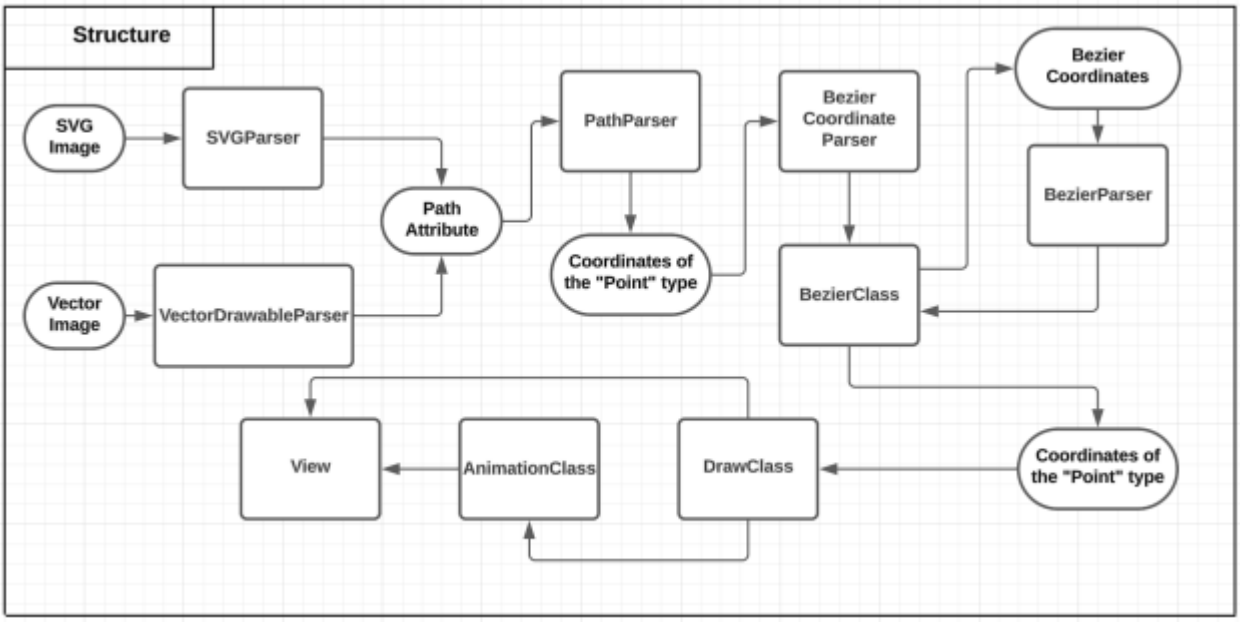

Рисунок 2.2 – Діаграма архітектури бібліотеки

2.2 Сценарії використання

Далі було узагальнено сценарій взаємодії розробника з програмним продуктом.

Атрибут Path передається до основного конструктора PathParser на зображенні (див. рис.2.1), який відповідає за парсинг даних. У PathParser передбачено три варіанти парсингу даних:

– звичайні;

– абсолютні;

– масштабовані координати.

Вибір варіанта парсингу залежить від типу та формату вхідних даних.

Далі отримані координати передаються до класу з формулами кривих Безьє. У цьому класі кожна координата, що відповідає за криві, розраховується за допомогою певних формул і після масив координат передаються в клас для статичного промальовування - DrawPath. У цьому класі використовуються методи Canvas для відображення зображення на екрані. Для подальшого анімації зображення в залежності від типу анімації необхідно заповнювати певні значення. Ці значення використовуються для розрахунку параметрів анімації, таких як тривалість, швидкість та напрямок тощо.

Наочніший приклад використання бібліотеки зображений на рис. 2.3.

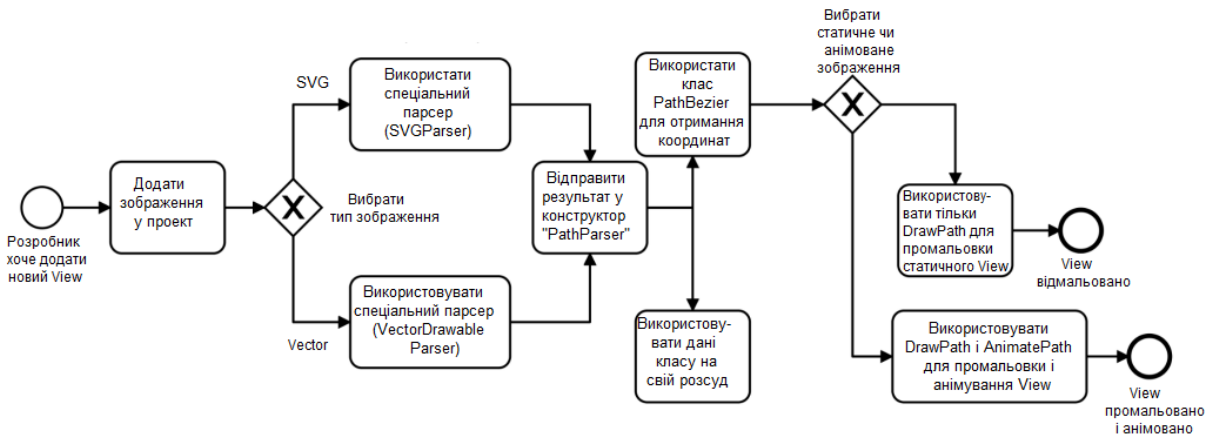

Рисунок 2.3 – Діаграма сценарію користувача

Також було складено узагальнену діаграму послідовності роботи з бібліотекою (рис. 2.4).

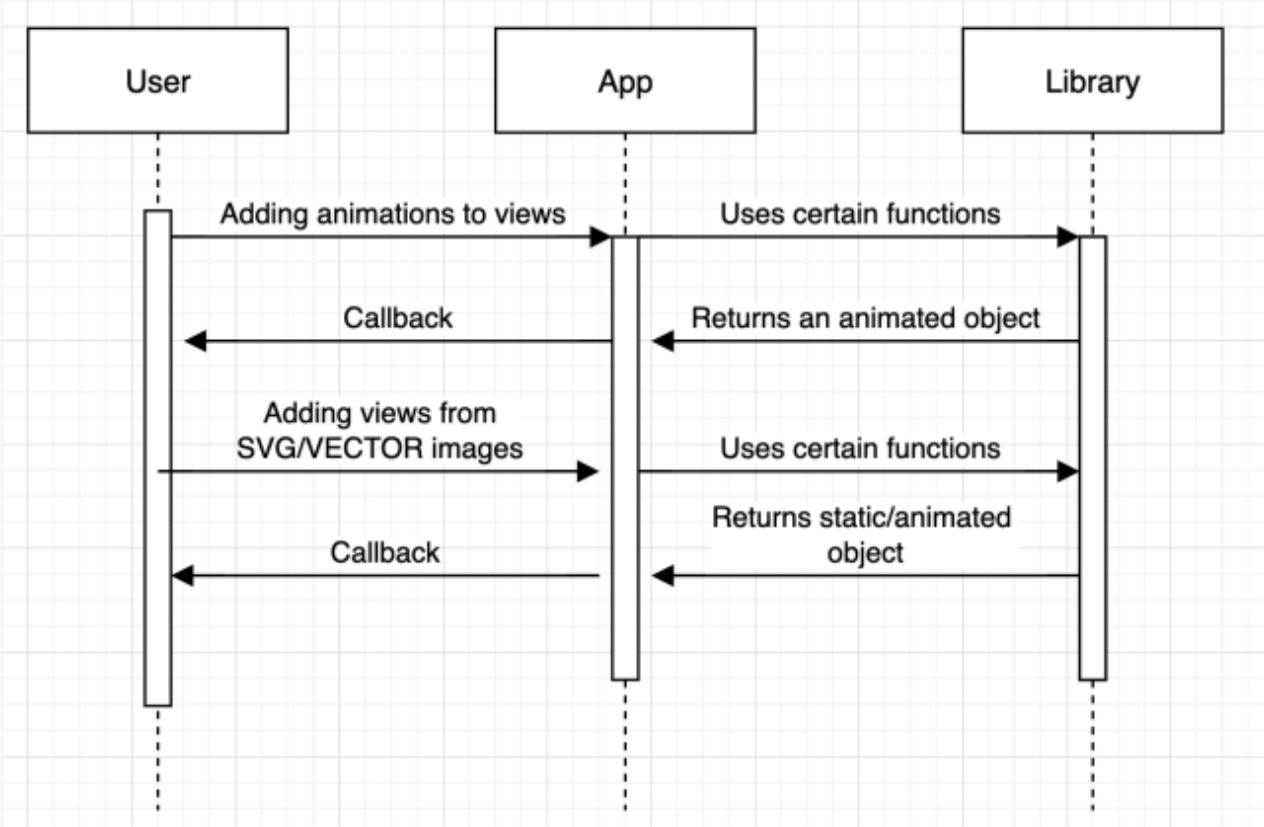

Рисунок 2.4 – Діаграма послідовності використання бібліотеки

2.3 Реалізація роботи з SVG та Vector зображеннями

Наступним завданням було створити парсер для SVG/Vector зображень, який дозволяв би отримувати такі дані:

– атрибут "Path";
– колір;
– ширина зображення;
– висота зображення.

Для парсингу SVG/Vector зображень було прийнято використовувати

XmlPullParser, так як даний інструмент є частиною стандартної бібліотеки Android SDK і призначений для обробки документів XML , а також забезпечує гнучкість при їх обробці. Він надає можливість послідовно проходити елементами XML - документа і витягувати необхідну інформацію. Такий підхід дозволяє уникнути завантаження всього документа на згадку і, отже, знижує споживання ресурсів програми.

У разі парсингу SVG/Vector зображень XmlPullParser використовується для отримання інформації про шлях (Path) та атрибутів зображення. Ця інформація потім використовується для формування об'єкта класу SVGParsedDrawable / ParsedVectorDrawable, який містить усі необхідні атрибути для обробки даних, відображення та анімування зображення на екрані пристрою.

SVGParsedDrawable і ParsedVectorDrawable є класами даних, які використовуються для зберігання інформації про SVG/Vector зображення після їх парсингу.

Також у програмному рішенні існує прошарок для валідації, що перевіряє коректність даних, які будуть надалі записані в класи SVGParsedDrawaЬIe та ParsedVectorDrawable. На рис. 2.5 можна відображено роботу модуля.

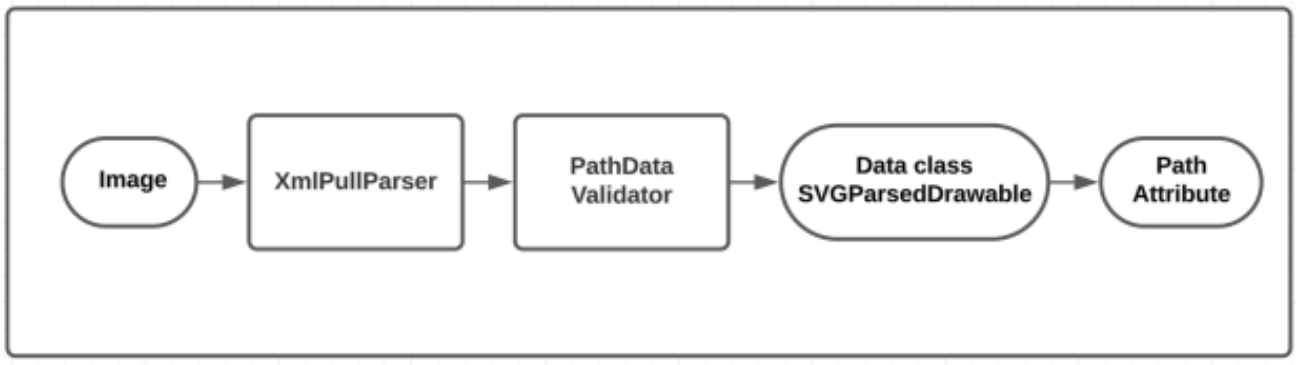

Рисунок 2.5 – Діаграма програмного модуля роботи із зображеннями

2.4 Path атрибут

2.4.1 Опис Path атрибута

Path атрибут у SVG та Vector файлах є одним з основних елементів, які визначають форму зображення. Він є набором команд та параметрів, які зі свого боку задають контури та криві лінії зображення, а також складається з букв та цифр, що визначають команди та параметри. Атрибут може бути використаний для створення різних форм, від простих ліній та кіл до складних кривих та багатокутників.

Цей атрибут містить набір команд, які можуть бути абсолютними або залежними, залежно від того, чи задаються координати точок щодо початку координатної системи зображення або відносно попередньої точки.

Наприклад, команда "M" (Move to) означає перехід до нової точки. Її абсолютна версія "M 10 20" означає перехід до точки з координатами (10, 20) щодо початку координатної системи зображення, а залежна команда "m 10 20" означає перехід до точки з координатами (10, 20) щодо попередньої точки.

Аналогічно абсолютна команда "L 30 40" означає перехід щодо початку координатної системи зображення, а залежна команда "1 30 40" означає малювання лінії щодо попередньої точки.

Команди "C" (Cubic Bezier curve), "S" (Smooth cubic Bezier curve to) та "Q" (Quadratic Bezier curve) означають малювання кривих Безьє різних типів. Наприклад, команда "C 10 20, 30 40, 50 60" означає малювання кривої Безьє третього порядку з двома контрольними точками (10, 20) і (30, 40) та кінцевою точкою (50, 60). На рис. 2.6 можна побачити залежність кривої від розташування контрольних точок.

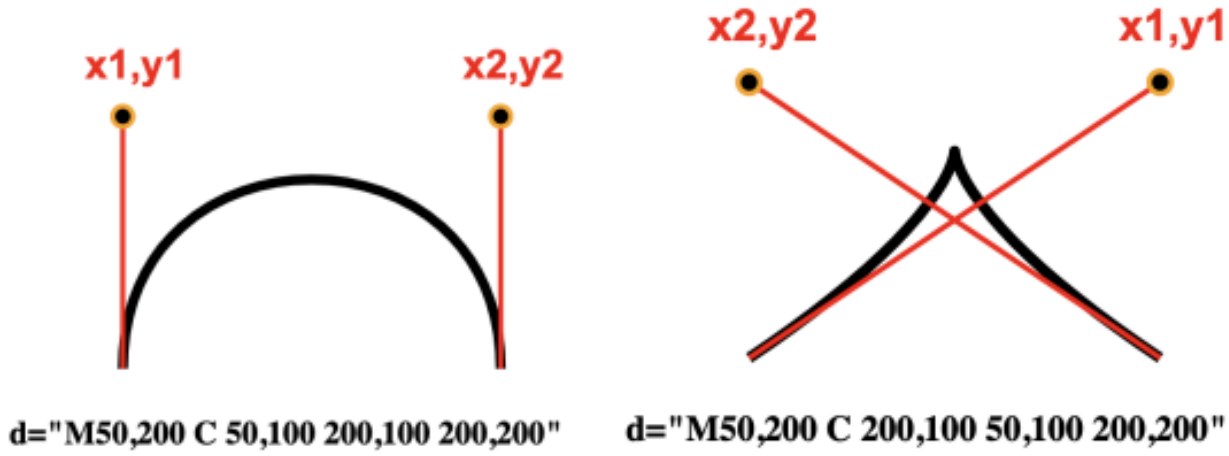

Рисунок 2.6 – Квадратична крива Безьє

Команда "Z" (Close path) означає закриття контуру, тобто малювання лінії від поточної точки до початкової точки контуру.

Команда "A" (Elliptical Arc) означає малювання дуги еліпса. Ця команда вимагає завдання кількох параметрів, таких як радіуси еліпса, кут повороту та інші.

При обробці атрибуту Path не можна нехтувати типом використовуваних команд і слід правильно інтерпретувати їх значення, а вибір між абсолютними та залежними командами залежить від конкретного завдання та необхідного результату.

На рис. 2.7 можна побачити приклад атрибуту Path та його складові частини.

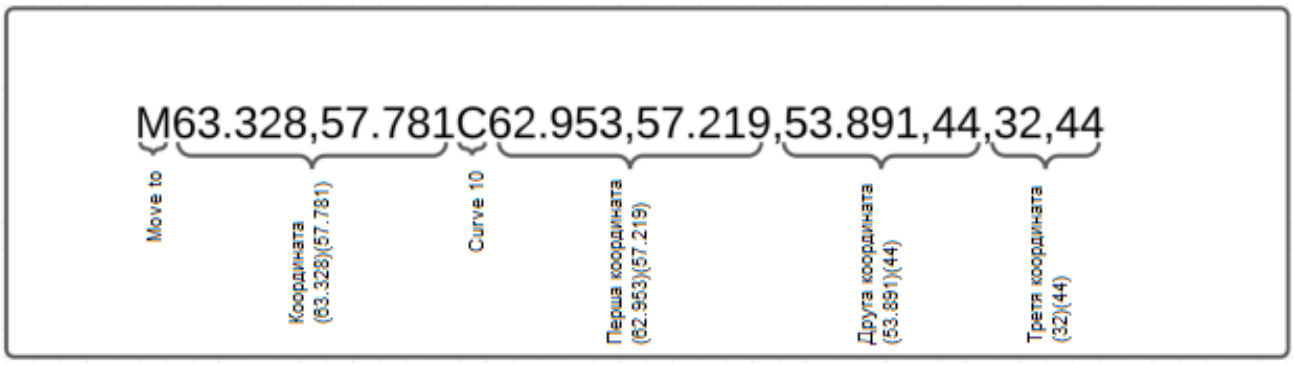

Рисунок 2.7 – Складові атрибуту Path

2.4.2 Реалізація "парсеру" для Path атрибуту

Для того, щоб використовувати складові частини Path атрибута, було необхідно розробити парсер, який зміг би перетворити даний атрибут на специфічні координати, які можуть бути використані надалі для промальовки та анімації зображень користувача.

Було вирішено розробити три окремі парсери зі своїми функціоналами в залежності від необхідних завдань.

Перший парсер (parsedPath) перетворює кожну координату залежно від її типу (абсолютна чи залежна) і використовується у разі, коли потрібно отримати набір координат для подальшої обробки чи аналізу. parsedPath не виконує жодних додаткових перетворень чи розрахунків.

Другий інструмент, absolutePathCoords, як і попередній, обробляє кожну точку в залежності від її типу і перетворює будь-які не абсолютні координати на абсолютні. Таким чином, виходить набір координат, що підходить для відображення зображення в будь-якому місці та масштабі. Цей інструмент корисний, коли потрібно точно визначити координати на площині для подальшої обробки або анімації зображення.

Наступний парсер (scaledAbsolutePathCoords) був реалізований на основі парсерів parsedPath і absolutePathCoords. Він перетворює кожну координату в залежності від її типу, як це робить parsedPath, всі не абсолютні координати він прораховує, а також скалює, використовуючи параметри актуального екрану. Такий підхід може гарантувати отримання набору координат, що підходить для відображення зображення на будь-якому екрані з правильним масштабом та позиціонуванням. Щоб використовувати цей інструмент було виправдано, необхідно використовувати його тільки для відображення зображення різних видах пристроїв, наприклад, якщо програма повинна працювати не тільки на мобільних девайсах, але й на планшетах/розумному годиннику.

На рис. 2.8 представлено узагальнену діаграму реалізації парсерів.

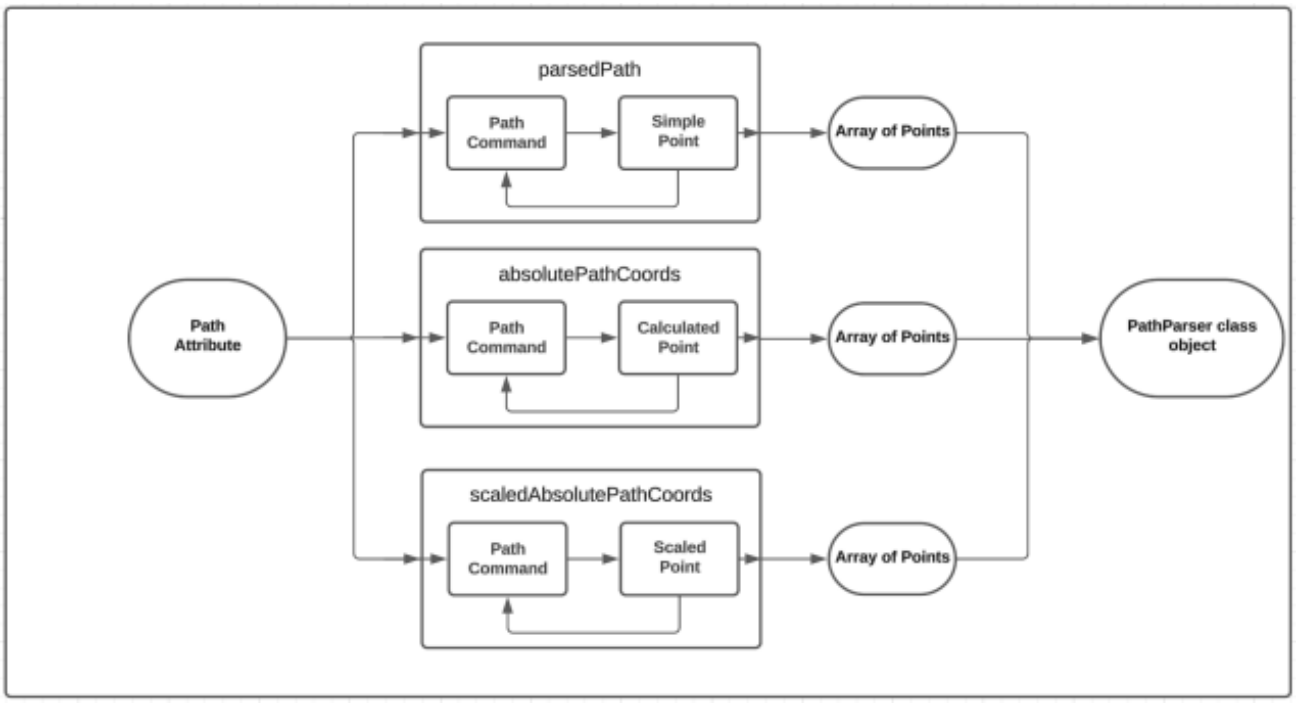

Рисунок 2.8 – Діаграма роботи модуля роботи з атрибутом Path

2.5 Обробка та розрахунок координат за допомогою структур кривих Безьє

Наступним етапом після створення парсерів для Path була розробка певного парсера для координат, який зможе зробити зображення більш плавним, використовуючи формули кривих Безьє.

Формули Безьє - це математичні функції, які дозволяють описувати криві та контури, використовуючи набір контрольних точок.

Парсер координат за допомогою формул Безьє дозволяє перетворити специфічний набір координат, отриманий з атрибуту Path, на набір контрольних точок, які можуть бути використані для створення плавних кривих і контурів і всього зображення.

PathBezier використовує алгоритм, суть якого полягає в проходженні набору специфічних координат, отриманих з попередніх парсерів, і визначенні типу кожної з них. Якщо певна координата є координатою для побудови кривої, далі вона передається в клас BezierCurves, де вже застосовуються відповідні

формули кривих Безьє для обчислення координат контрольних точок та створення кривих залежно від їх типу.

Для цього парсера обробляються лише криві типу:

– "A" (Elliptical Arc);
– "C" (Cubic Bezier curve);
– "S" (Smooth cubic Bezier curveto);
– "Q" (Quadratic Bezier curve);
– "T" (Smooth quadratic Bezier curveto).

Інструмент PathBezier дає змогу ефективно обробляти складні форми кривих, які описані у SVG та Vector файлах, для їхнього майбутнього використання при створенні зображень з високою якістю. Цей інструмент також може бути використаний не тільки в області застосування цього програмного засобу, а в цілому для отримання будь-яких даних для розробки певних проектів.

На рис. 2.9 наведено схему роботи цього модуля.

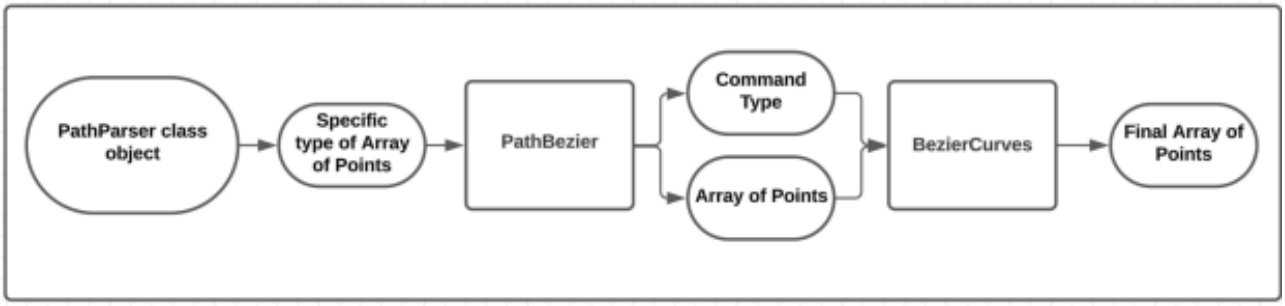

Рисунок 2.9 – Діаграма роботи модуля перетворення координат з допомогою формул кривої Безьє

2.6 Відображення статичного зображення

Для відображення на екрані статичного зображення було необхідно реалізувати механізм його відображення за отриманими координатами. Даний механізм повинен забезпечувати можливість відображення зображення на екрані

пристрою з використанням координат, отриманих за допомогою парсерів .

Таким чином, був розроблений конструктор класу DrawPath, що генерує об'єкт View, який може відображатися на екрані пристрою користувача.

Як вхідні параметри цього конструктора можуть бути використані такі дані:

– координати, котрі отримані за допомогою парсерів для атрибуту Path;

– ширина та висота зображення, що визначають розміри області, в якій буде відображено зображення;

– колір зображення;

– атрибут Path, за допомогою якого можна відобразити зображення.

У конструкторі класу DrawPath використовувалися класи Canvas та Paint для малювання зображення, а також був перевизначений метод onDraw для відображення зображення на екрані пристрою. Для малювання на поверхні екрана були завантажені координати Canvas, а всі стилі, кольори та інші параметри малювання були визначені в Paint.

Для розташування зображення на екрані пристрою був використаний метод onMeasure. Даний метод викликається системою визначення розмірів View, яке буде відображено на екрані пристрою. У ньому були встановлені розміри зображення відповідно до розмірів, отриманих за допомогою парсерів. На рис. 2.10 можна побачити узагальнену схему роботи цього модуля.

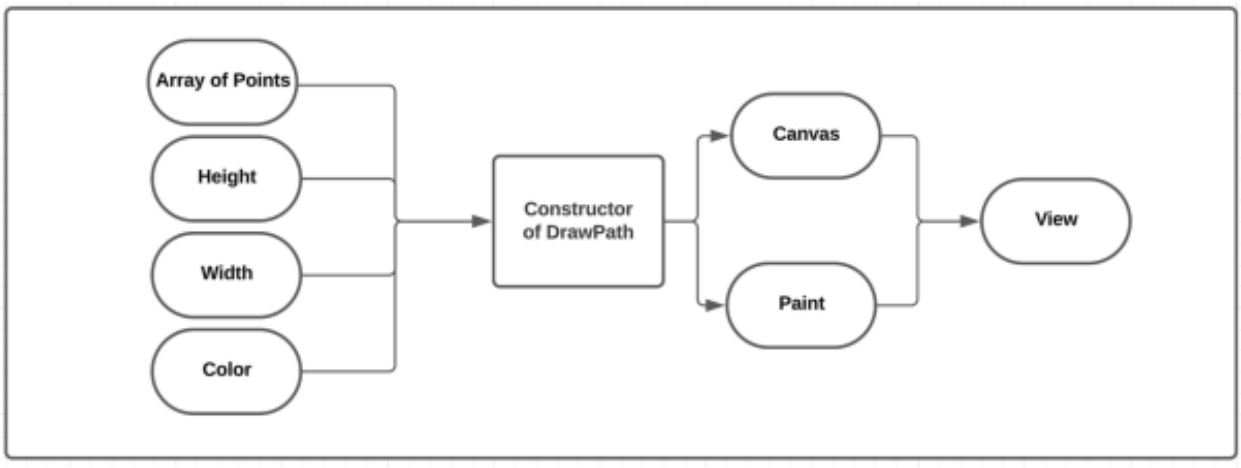

Рисунок 2.10 – Діаграма роботи модуля відтворення статичного зображення

2.7 Динамічна анімація зображення

Для реалізації динамічної анімації, використаний як основа схожий механізм відображення - DrawPath, створений конструктор класу Animation, який повертає анімовану View, що відображається на екрані користувача.

На вхід конструктора можна подавати такі параметри:

– координати, які визначають положення та форму анімованого об'єкта;

– ширину та довжину, що визначають розміри анімованого об'єкта;

– колір об'єкта;

– атрибут Path визначення форми об'єкта;

– деякі атрибути анімації, які залежать від типу анімації, наприклад, тривалість, швидкість, напрямок і т.д.

Крім того, у бібліотеці також надано різні типи анімацій, які можна використовувати залежно від потреб та вимог проекту.

1. Лінійна анімація (linear animation) - проста анімація, яка змінює значення якості об'єкта з постійною швидкістю. Цей тип анімації корисний для створення простих та гладких переходів між станами об'єкта.

2. Нелінійна анімація (Non-linear animation) - це анімація для складніших і динамічніших переходів між станами об'єкта, що змінює значення властивості об'єкта зі змінною швидкістю, наприклад прискорення чи уповільнення.

3. Формальна анімація (Shape animation) – анімація, яка здатна змінити розміри об'єкта.

4. Шляхова анімація (Path animation) - цей тип анімації пов'язані з рухом об'єкта певним шляхом, наприклад рух кривою, чи переміщення між точками.

Дані типи анімацій були реалізовані з використанням різних підходів та інструментів, щоб забезпечити надійність, продуктивність, масштабованість об'єкта. Крім того, були надані різні налаштування та параметри для кожного типу анімації, щоб дозволити розробникам ефективно використовувати їх у своїх проектах.

За допомогою перевизначення методу OnTouchEvent було реалізовано обробку торкання екрана користувачем, а також оброблено деякі анімації, пов'язані з торканням екрана, наприклад, зміна кольору або розміру об'єкта при натисканні на нього.

Крім того, був створений метод SpecificAnimationMethod, в якому реалізовані різні алгоритми та підходи для кожного типу анімації, а також реалізований механізм, який дозволяє розробнику обробляти індивідуальні параметри та логіку анімованого об'єкта.

Всі типи анімацій були реалізовані з використанням різних підходів, так для формальної анімації використано алгоритми, пов'язані зі зміною форми об'єкта – матричні перетворення. Для дорожньої анімації використаний підхід, пов'язаний з рухом об'єкта за певним шляху - інтерполяція кривих Безьє.

Таким чином, реалізовано систему створення анімацій, яка дозволяє розробнику анімувати об'єкти та використовувати їх у своїх проектах. На рис. 2.11 можна побачити схему цього модуля.

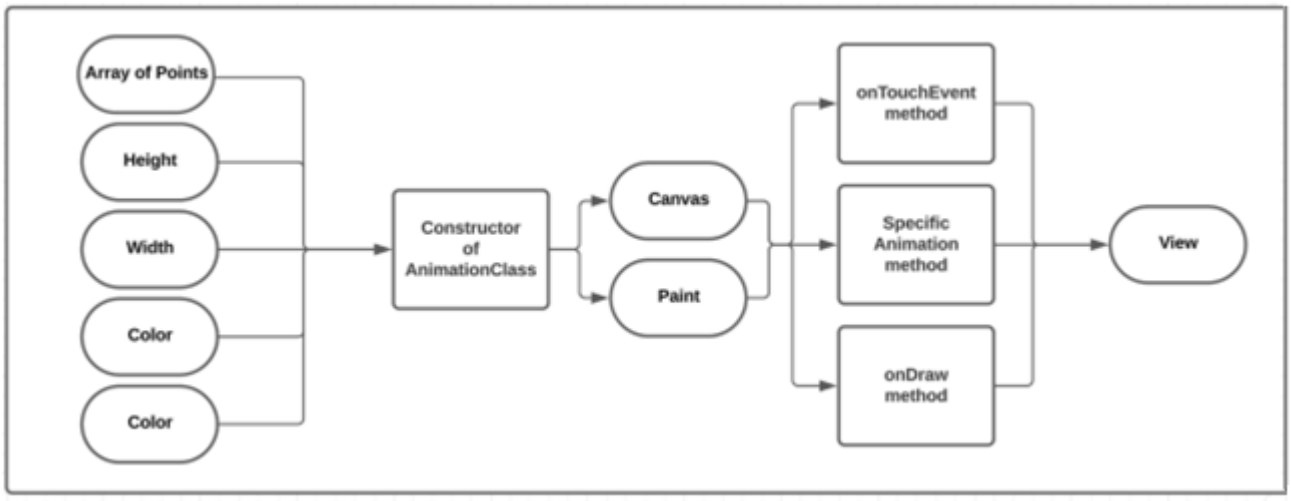

Рисунок 2.11 – Діаграма роботи модуля промальовки анімованого зображення

3 РЕЗУЛЬТАТИ РОБОТИ

3.1 Тестування розробленого рішення

Бібліотека була протестована та показала непогані результати порівняно з іншими альтернативними рішеннями.

Для тестування були використані різні підходи та визначений інструмент для оцінки продуктивності – Systrace [26], щоб забезпечити масштабованість. Systrace забезпечує застосування політики системних викликів для програм, обмежуючи доступ програм до системи. Політика генерується в інтерактивному режимі. Операції, які не охоплюються цією політикою, викликають тривогу, дозволяючи користувачеві уточнити поточну налаштовану політику.

Також для читання та аналізу файлів трасування, зібраних за допомогою Systrace, був використаний інструмент Perfetto UI (рис. 3.1) [27]. Perfetto дозволяє переглядати та аналізувати сліди в браузері. Він підтримує кілька різних форматів трасування, включаючи формат трасування perfetto proto та застарілий формат трасування json.

Рисунок 3.1 – Логотип Perfetto UI

Рис. 3.2 демонструє метрики порівняльного аналізу. На ньому можна побачити, що інструмент показує хороший результат відносно інших інструментів та способів малювання зображень.

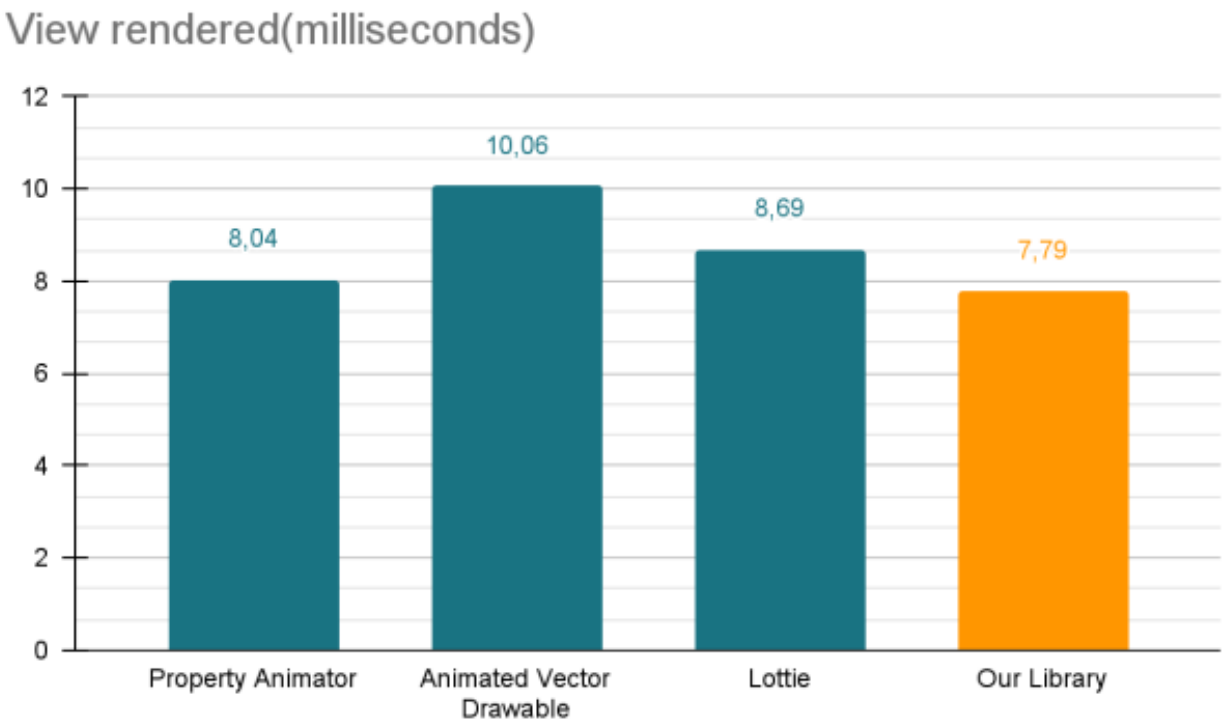

Рисунок 3.2 – Діаграма показників продуктивності інструментів

3.2 Документування бібліотеки

Для забезпечення повної та чіткої документації бібліотеки використано кілька підходів:

– коментування коду: додано коментарі, щоб забезпечити чітке та коротке пояснення того, що робить код, які аргументи приймають функції, що повертають тощо;

– використання документуючих рядків: включені рядки (docstrings) у функції, класи та модулі, щоб пояснити, як використовувати елементи коду;

– створення README -файлу: розроблено докладний README- файл, який містить короткий опис бібліотеки, її функціональні можливості, інструкції з встановлення та використання, а також приклади коду для роботи з бібліотекою.

Дані підходи дозволяють розробникам зрозуміти функціональні можливості бібліотеки та мінімізувати час, необхідний її освоєння.

Як приклад документації у коді можна розглянути функцію "cubicCurve" на рис. 3.3.

```
/**
  * Ця функція обчислює та повертає масив точок, які утворюють
  кубічну криву Безьє.
  *    Крива визначається початковою точкою, двома точками
  підтримки та кінцевою точкою.
 * @param Startpoint Початкова точка кривої.
 *     @param FirstSupportPoint Перша точка підтримки кривої.
 *     @param SecondSupportPoint Друга точка підтримки кривої.
 *     @param endPoint Остання точка кривої.
 *     @return Масив точок, які утворюють кубічну криву Безьє.
 */
fun cubicCurve(
   startPoint: Point, firstSupportPoint:
Point, secondSupportPoint: Point,
endPoint: Point ) : Array<Point> { ... }
```

Рисунок 3.3 – Лістинг функції "cubicCurve"

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Класифікація шкідливих та небезпечних виробничих факторів

Шкідливий виробничий фактор – небажане явище, яке супроводжує виробничий процес і вплив якого на працюючого може призвести до погіршення самопочуття, зниження працездатності, захворювання, виробничо зумовленого чи професійного, і навіть смерті, як результату захворювання. Небезпечний виробничий фактор – небажане явище, яке супроводжує виробничий процес і дія якого за певних умов може призвести до травми або іншого раптового погіршення здоров'я працівника (гострого отруєння, гострого захворювання) і навіть до раптової смерті [28].

Поділ несприятливих чинників виробничого середовища на шкідливі та небезпечні зумовлене різним характером їх дії на людський організм, тим, що вони потребують різних заходів та засобів для боротьби з ними та профілактики викликаних ними ушкоджень, а також рядом причин організаційного характеру. В той же час між шкідливими та небезпечними виробничими факторами інколи важко провести чітку межу. Один і той же чинник може викликати травму і профзахворювання (наприклад, високий рівень іонізуючого або теплового випромінювання може викликати опік або навіть призвести до миттєвої смерті, а довготривала дія порівняно невисокого рівня цих же факторів – до хвороби; пилинка, що потрапила в око, спричиняє травму, а пил, що осідає в легенях, – захворювання, що зветься пневмоконіоз). Через це всі несприятливі виробничі чинники часто розглядаються як єдине поняття – небезпечний та шкідливий виробничий фактор (НШВФ) [29]. За своїм походженням та природою дії всі НШВФ можна поділити на 5 груп: фізичні, хімічні, біологічні, психофізіологічні та соціальні До фізичних НШВФ відносяться машини та механізми або їх елементи, а також вироби, матеріали, заготовки тощо, які рухаються або обертаються; конструкції, які руйнуються; системи, устаткування або елементи обладнання, які знаходяться під підвищеним тиском; підвищена запиленість та загазованість

повітря; підвищена або понижена температура повітря, поверхонь приміщення, обладнання, матеріалів; підвищені рівні шуму, вібрації, ультразвуку, інфразвуку; підвищений або понижений барометричний тиск та його різкі коливання; підвищена та понижена вологість; підвищена швидкість руху та підвищена іонізація повітря; підвищений рівень іонізуючих випромінювань; підвищене значення напруги в електричній мережі; підвищені рівні статичної електрики, електромагнітних випромінювань; підвищена напруженість електричного, магнітного полів; відсутність або нестача світла; недостатня освітленість робочої зони; підвищена яскравість світла; понижена контрастність; прямий та віддзеркалений блиск; підвищена пульсація світлового потоку; підвищені рівні ультрафіолетової та інфрачервоної радіації; гострі крайки, зачипки, шершавість на поверхні заготовок, інструментів та обладнання; розташування робочого місця на значній висоті відносно землі (підлоги); слизька підлога; невагомість.

Хімічні НШВФ:

- за характером дії на організм людини поділяються на токсичні, задушливі, наркотичні, подразнюючі, сенсибілізуючі, канцерогенні, мутагенні та такі, що впливають на репродуктивну функцію;

- за шляхами проникнення в організм людини поділяються на такі, що потрапляють через: 1) органи дихання; 2) шлунково-кишковий тракт; 3) шкіряні покриви та слизова оболонка;

- які перебувають у різному агрегатному стані: 1)твердому 2)газоподібному 3)рідкому.

Біологічні НШВФ – це: - патогенні мікроорганізми (бактерії, віруси, рикетсії, спірохети, грибки, найпростіші) та продукти їхньої життєдіяльності; - макроорганізми (тварини та рослини) та продукти їхньої життєдіяльності. До психофізіологічних НШВФ відносяться фізичні (статичні та динамічні) перевантаження і нервово-психічні перевантаження (розумове перенапруження, перенапруження аналізаторів, монотонність праці, емоційні перевантаження). 6 Соціальні НШВФ – це неякісна організація роботи, понаднормова робота, змушеність праці в колективі з поганими відносинами між його членами, соціальна

ізольованість з відривом від сім'ї, зміна біоритмів, незадоволеність роботою, фізична та/або словесна образа та її ризик, насильство та його ризик. Один і той же НШВФ за природою своєї дії може належати водночас до різних груп.

4.2 Вплив вібрації на людину

Вібрація - це механічні коливання пружних тіл або коливальні рухи механічних систем. Для людини вібрація є видом механічного впливу, який має негативні наслідки для організму [28].

Причиною появи вібрації є неврівноважені сили та ударні процеси в діючих механізмах. Створення високопродуктивних потужних машин і швидкісних транспортних засобів при одночасному зниженні їх матеріалоємності неминуче призводить до збільшення інтенсивності і розширення спектру вібраційних та віброакустичних полів. Цьому сприяє також широке використання в промисловості і будівництві високоефективних механізмів вібраційної та віброударної дії.

Дія вібрації може приводити до трансформування внутрішньої структури і поверхневих шарів матеріалів, зміни умов тертя і зносу на контактних поверхнях деталей машин, нагрівання конструкцій. Через вібрацію збільшуються динамічні навантаження в елементах конструкцій, стиках і сполученнях, знижується несуча здатність деталей, ініціюються тріщини, виникає руйнування обладнання. Усе це приводить до зниження строку служби устаткування, зростання імовірності аварійних ситуацій і зростання економічних витрат. Вважають, що 80% аварій в машинах і механізмах здійснюється внаслідок вібрації. Крім того, коливання конструкцій часто є джерелом небажаного шуму. Захист від вібрації є складною і багатоплановою в науково-технічному та важливою у соціально-економічному відношеннях проблемою нашого суспільства [29].

Вплив вібрації на людину залежить від її спектрального складу, напрямку дії, прикладення, тривалості впливу, а також від індивідуальних особливостей людини.

При оцінці вібраційного впливу потрібно враховувати, що коливальні процеси притаманні живому організму. В основі серцевої діяльності і кровообігу та біострумів мозку лежать ритмічні коливання. Внутрішні органи людини можна розглядати як коливальні системи з пружними зв'язками. Частоти їх власних коливань лежать у діапазоні 3..6 Гц. Частоти власних коливань плечового пояса, стегон і голови щодо опорної поверхні (положення стоячи) складають 4...6 Гц, голови щодо плечей (положення сидячи) 25...30 Гц.

При впливі на людину зовнішніх коливань (хитавиці, струсів, вібрації) відбувається їхня взаємодія з внутрішніми хвильовими процесами, виникнення резонансних явищ. Так, зовнішні коливання частотою менш 0,7 Гц утворюють хитавицю і порушують у людини нормальну діяльність вестибулярного апарата. Інфразвукові коливання (менш 16 Гц), впливаючи на людину, пригнічують центральну нервову систему, викликаючи почуття тривоги, страху. При певній інтенсивності на частоті 6..7 Гц інфразвукові коливання, втягуючи у резонанс внутрішні органи і систему кровообігу, здатні викликати травми, розриви артерій, тощо [29].

Вібрація, що діє на людину, має широкий діапазон – від десятих часток одного до декількох тисяч Гц. Характерними ознаками шкідливого впливу вібрації на людину є можливі зміни у функціональному стані: підвищена втома, збільшення часу моторної реакції, порушення вестибулярної реакції. Медичними дослідженнями встановлено, що вібрація є подразником периферичних нервових закінчень, розташованих на ділянках тіла людини, що сприймають зовнішні коливання. Адекватним фізичним критерієм оцінки її впливу на організм людини є коливальна енергія, що виникає на поверхні контакту, а також енергія, поглинена тканинами і передана опорно-руховому апарату та іншим органам. У результаті впливу вібрації виникають нервовосудинні розлади, ураження кістково-суглобної та інших систем організму. Відзначаються, наприклад, зміни функції щитовидної залози, сечостатевої системи, шлунково-кишкового тракту. Так, медичні дослідження показали , що у працюючих в умовах вібрації відбуваються значні зміни кістковосуглобної системи, які виражаються у функціональній перебудові

кісткової тканини, регіональному остеопорозі, кістовидних утвореннях у кістках, асептичному некрозі кісток, хронічних переломах. Відзначається, що терміни виникнення змін у кістках у працівників вібраційних професій коливається в межах від 6-8 місяців до 2-5 років.

Шкідливість вібрації збільшується при одночасному впливі на людину таких факторів, як знижена температура, підвищений шум, запиленість повітря, тривала статична напруга тощо. Сучасна медицина розглядає виробничу вібрацію як могутній стрес-фактор, що має негативний вплив на психомоторну працездатність, емоційну сферу і розумову діяльність, підвищує ймовірність виникнення різних захворювань і нещасних випадків. Особливо небезпечний тривалий вплив вібрації для жіночого організму. Широкий комплекс патологічних відхилень, викликаний впливом вібрації на організм людини, кваліфікується як віброзахворювання [28].

Вібрація як фізичний чинник виробничого середовища спостерігається в металообробній, гірничодобувній, металобудівній, машинобудівній, авіаційній та інших галузях народного господарства. Джерелом вібрації можуть бути різні механізми, вібраційне устаткування, віброінструменти, акустичні системи, транспортні та сільськогосподарські машини.

Загальна вібрація поділяється на транспортну вібрацію, яка діє на людину на робочих місцях в транспортних засобах (трактори сільськогосподарські та промислові, самохідні сільськогосподарські машини (комбайни), тягачі, грейдери ті інші); транспортно-технологічну вібрацію, яка діє на людину на робочих місцях машин з обмеженою рухливістю (екскаватори, крани промислові та будівельні, гірничі комбайни, транспорт виробничих приміщень та інші) та технологічну вібрацію, яка діє на людину на робочих місцях стаціонарних машин чи передається на робочі, де немає джерел вібрації (верстати та метало-деревообробне, пресувально-ковальське обладнання, ливарні машини, електричні машини, насосні агрегати та вентилятори, обладнання для буріння свердловин, бурові верстати, машини для тваринництва, очищення та сортування зерна (у тому числі сушарні), обладнання промисловості будматеріалів (крім бетоноукладачів), установки хімічної та нафтохімічної промисловості та інші.

Оператори машин, які зазнають у процесі трудової діяльності впливу вібрації, підлягають попереднім та періодичним медичним оглядам відповідно до Порядку проведення медичних оглядів працівників певних категорій, затвердженого Наказом МОЗ України від 21.05.2007 р. №246. Обов'язкові попередні (під час прийняття на роботу) та періодичні (протягом трудової діяльності) медичні огляди дозволять визначити стан здоров'я працівника та можливість виконання без погіршення стану здоров'я професійних обов'язків, своєчасно виявити ранні ознаки хронічного професійного захворювання, забезпечує динамічне спостереження за станом здоров`я в умовах дії шкідливих та небезпечних факторів і трудового процесу, вирішує питання щодо можливості продовжувати роботу в умовах дії шкідливих та небезпечних факторів і трудового процесу [28].

За результатами періодичних медичних оглядів роботодавець забезпечує проведення відповідних оздоровчих заходів Заключного акта у повному обсязі та усуває причини, що призводять до професійних захворювань. Організовує проведення лабораторних досліджень умов праці на робочих місцях та вживає заходів до усунення небезпечних і шкідливих для здоров'я виробничих факторів.

До роботи операторами машин допускаються особи не молодші 18 років, які пройшли попередній медичний огляд, мають відповідну кваліфікацію та ознайомлені з характером впливу вібрації на організм.

ВИСНОВКИ

У рамках кваліфікаційної роботи було реалізовано бібліотеку для оптимізації процесу використання SVG/Vector зображень, їх анімацію та промальовку за допомогою формул кривих Безьє.

Були виконані всі поставлені завдання:

– вивчено структуру формул кривих Безьє;

– вивчено інструментарій, який необхідний для створення та анімування зображень на платформі Android;

– реалізовано "парсери" SVG/Vector зображень;

– реалізовано промальовування даних статичних зображень;

– реалізовано їх різні анімації за формулами кривим Безьє;

– забезпечено документування та тестування бібліотеки для оптимізації процесу використання SVG/Vector зображень, їх анімації та промальовування за допомогою формул кривих Безьє.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ImageView. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/ widget/ImageView (Дата звернення: 16.02.2024).

2. SurfaceView. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/ view/SurfaceView (Дата звернення: 16.02.2024).

3. Kotlin. [Електронний ресурс] – Режим доступу до ресурсу: https://kotlinlang.org/ (Дата звернення: 7.03.2024).

4. SVG Tutorial. [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/graphics/ svg_intro.asp (Дата звернення: 7.03.2024).

5. Vector drawables overview. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/ develop/ui/views/graphics/vector-drawable-resources (Дата звернення: 7.03.2024).

6. ObjectAnimator. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/ android/animation/ObjectAnimator (Дата звернення: 7.03.2024).

7. Internet Archive. The Wayback Machine. [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/ (Дата звернення: 03.03.2024).

8. Animation. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/topics/animation (Дата звернення: 29.02.2024).

9. Крива Безьє. [Електронний ресурс] – Режим доступу до ресурсу: https://uk.javascript.info/bezier-curve (Дата звернення: 21.02.2024).

10. Криві Безьє. Основні поняття та властивості кривих Безьє. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathros.net.ua/kryvi-bezje.html (Дата звернення: 07.06.2024).

11. View. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/view/ View (Дата звернення: 16.02.2024).

12. OpenGL ES. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/develop/ui/views /graphics/opengl/about-opengl/ (Дата звернення: 17.02.2024).

13. Smoliková H. Introduction to computer animation and its possible educational applications. [Електронний ресурс] – Режим доступу: https://www.researchgate.net/publication/259211673_Introduction_to_computer_animation_and_its_possible_educational_applications (Дата звернення: 01.03.2025).

14. Animated Vector Drawable. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/android/graphics/drawable/AnimatedVectorDrawable (Дата звернення: 29.02.2024).

15. SVG to Vector Drawable. [Електронний ресурс] – Режим доступу до ресурсу: https://svg2vector.com/ (Дата звернення: 29.02.2024).

16. Adobe. [Електронний ресурс] – Режим доступу до ресурсу: https://www.adobe.com/ (Дата звернення: 01.03.2024).

17. V.Lea. A better tool for cubic-bezier() easing. [Електронний ресурс] – Режим доступу до ресурсу: https://cubic-bezier.com/ (Дата звернення: 02.03.2024).

18. Android Animation Example. [Електронний ресурс] – Режим доступу до ресурсу: https://www.digitalocean.com/ community/tutorials/android-animation-example (Дата звернення: 02.03.2024).

19. AnimatedSvgView. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/jaredrummler/ AnimatedSvgView (Дата звернення: 03.03.2024).

20. Leonids. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/plattysoft/Leonids (Дата звернення: 03.03.2024).

21. ExplosionField. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/tyrantgit/ExplosionField (Дата звернення: 03.03.2024).

22. NineOldAndroids. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/JakeWharton/ NineOldAndroids (Дата звернення: 03.03.2024).

23. AndroidViewAnimations. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/daimajia/ AndroidViewAnimations (Дата звернення: 03.03.2024).

24. CubicBezierEasing. [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/reference/ kotlin/androidx/compose/animation/core/ CubicBezierEasing (Дата звернення: 21.02.2024).

25. CurvedBottomSheet. [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/TayfunCesur/ CurvedBottomSheet (Дата звернення: 08.03.2024).

26. Systrace - Interactive Policy Generation for System Calls. [Електронний ресурс] – Режим доступу до ресурсу: http://www.citi.umich.edu/u/provos/systrace/ (дата звернення: 08.05.2025).

27. System profiling, app tracing and trace analysis // Perfetto. [Електронний ресурс] – Режим доступу до ресурсу: https://ui.perfetto.dev// (Дата звернення: 01.04.2024)

28. Яремко З. М. Безпека життєдіяльності: Навч. посіб. Львів., 2005. 301 с.

29. Желібо Є. П. Заверуха Н.М., Зацарний В.В. Безпека життєдіяльності. Навчальний посібник. К.: Каравела, 2004. 328 с.

ДОДАТКИ

Додаток А. Диск з роботою