Міністерство освіти і науки України

Вінницький національний технічний університет

(повне найменування вищого навчального закладу)

Кафедра програмного забезпечення

(повна назва кафедри)

КУРСОВА РОБОТА

з дисципліни «Комп’ютерна графіка»

(назва дисципліни)

на тему: Розробка статичних та динамічних зображень для веб-сайту «Сергій Корольов»

Члени комісії:

Студента 3 курсу 5ПІ-22б групи

спеціальності 121 – «Інженерія програмного забезпечення» Ревун І.І

(прізвище та ініціали)

Керівник: асистент Завальнюк Є.К.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів: Оцінка: ECTS

Завальнюк Є.К.

(прізвище та ініціали)



(підпис)

Романюк О.Н.

(прізвище та ініціали)

м. Вінниця – 2025

Міністерство освіти і науки України Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Зав. кафедри ПЗ., проф., д.т.н.

О.Н. Романюк

Протокол № 2 від

« 11 » вересня 2025 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу з дисципліни «Комп’ютерна графіка» студенту групи 5ПІ-23б

Ревуну Ільї Івановичу

Тема роботи: Розробка статичних та динамічних зображень для веб-сайту

«Сергій Корольов». Вихідні дані:

1. мова сайту – українська;
2. графічне забезпечення сайту виконати з використанням сучасних веб-технологій (css3, html та ін.);
3. анімаційне забезпечення сайту виконати з використанням відповідних засобів анімації (Adobe Illustrator, Adobe Photoshop, Visual Studio Code, Intellij IDEA та ін.);
4. обґрунтувати вибір колірної гами;
5. обґрунтувати розробку логотипу сайту;
6. багатоваріантний аналіз на всіх стадіях розробки;
7. обґрунтувати стиль викладення матеріалу пояснювальної записки.

Основні складові пояснювальної записки:

* + Титульний лист
  + Індивідуальне завдання
  + Анотація
  + Зміст
  + Вступ
  + 1 Багатоваріантний аналіз розв’язку основної задачі
  + 2 Вибір кольорової гами
  + 3 Розробка логотипу сайту
  + 4 Розробка статичних зображень
  + 5 Розробка динамічних зображень
  + 6 Оцінка обсягів зображень
  + 7 Тестування роботи сайту
  + 8 Розробка програмного застосунку
  + Висновки
  + Список використаних джерел

Назва програмного застосунку: «Реалізація неітеративної сферично-кутової інтерполяції векторів вздовж рядка растеризації».

Дата видачі « 12 » вересня 2025 р. Керівник: Є.К. Завальнюк

Завдання отримав: І.І. Ревун

АНОТАЦІЯ

У роботі розглянуто процес створення та інтеграції статичних і динамічних графічних зображень для вебсайту, присвяченого життю та діяльності Сергія Павловича Корольова — видатного конструктора ракетно-космічних систем і ключової фігури світової космонавтики. Основний акцент зроблено на розробці авторських графічних елементів різних типів (векторних і растрових ілюстрацій, контурних зображень, PNG-матеріалів та GIF-анімацій), які застосовано на сторінках сайту та адаптовано до його тематичного та стилістичного оформлення. Створені візуальні матеріали формують цілісне графічне середовище й підсилюють інформаційно-просвітницький характер ресурсу.

Окремо в роботі представлено програмну компоненту, що присвячена реалізації методу неітеративної сферично-кутової інтерполяції векторів. Результати можуть бути використані для подальших досліджень, навчальних демонстрацій або прикладних задач, що потребують точного інтерполювання орієнтацій та напрямних векторів.

ЗМІСТ

ВСТУП......................................................................................................................5 1 БАГАТОВАРІАНТНИЙ АНАЛІЗ РОЗВ’ЯЗКУ ОСНОВНОЇ ЗАДАЧІ .......... 7 2 ВИБІР КОЛЬОРОВОЇ ГАМИ .......................................................................... 13 3 РОЗРОБКА ЛОГОТИПУ САЙТУ .................................................................... 16 4 РОЗРОБКА СТАТИЧНИХ ЗОБРАЖЕНЬ........................................................ 20 5 РОЗРОБКА ДИНАМІЧНИХ ЗОБРАЖЕНЬ..................................................... 34 6 ОЦІНКА ОБСЯГІВ ЗОБРАЖЕНЬ................................................................... 37 7 ТЕСТУВАННЯ РОБОТИ САЙТУ................................................................... 40 8 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ............................................... 43 ВИСНОВКИ.......................................................................................................... 48 СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ............................................................. 50 Додаток А. СТАТИЧНІ ЗОБРАЖЕННЯ............................................................. 51 Додаток Б. ДИНАМІЧНЕ ЗОБРАЖЕННЯ ......................................................... 57 Додаток В. ЛІСТИНГ ВЕБСТОРІНКИ INDEX.HTML ..................................... 65 Додаток Г. ЛІСТИНГ ПРОГРАМНОГО ЗАСТОСУНКУ.................................. 72

ВСТУП

Комп’ютерна графіка є динамічною галуззю сучасних цифрових технологій, що охоплює створення, обробку та відтворення візуальної інформації за допомогою спеціалізованих інструментів. Її застосування охоплює широкий спектр сфер — від освіти й мультимедіа до інженерії, наукових досліджень та космічних проєктів — що підкреслює її універсальність і важливість у формуванні візуального мислення та цифрової культури.

Одним із актуальних напрямів використання комп’ютерної графіки є розробка вебресурсів, у яких графічні матеріали відіграють ключову роль у структуризації та поданні інформації. Завдяки застосуванню зображень, ілюстрацій та анімацій вебсайти набувають виразності, стають зручнішими у сприйнятті та виконують не лише інформативну, а й візуально-естетичну функцію.

Особливий інтерес становлять ресурси, присвячені видатним постатям науки й техніки, оскільки вони поєднують інформаційну та просвітницьку складові. Сергій Павлович Корольов — провідний інженер і конструктор ракетно-космічних систем — є однією з ключових фігур розвитку світової космонавтики. Представлення його життєвого шляху, наукових напрацювань та спадщини у вигляді багатосторінкового вебресурсу потребує ретельно створених графічних елементів, що відповідають тематичному та стильовому спрямуванню.

Метою роботи є створення графічних елементів (статичних зображень, контурних ілюстрацій, PNG-матеріалів та динамічних GIF-анімацій) для тематичного вебсайту про Сергія Корольова та їх інтеграція у структуру вебресурсу.

Об’єктом дослідження є процес створення та застосування статичних і динамічних графічних матеріалів у вебсередовищі.

Предметом дослідження є методи та засоби комп’ютерної графіки, що використовуються під час розробки графічних елементів сайту та реалізації математичних алгоритмів обробки візуальних даних.

У межах роботи додатково створено програмну компоненту, присвячену реалізації методу неітеративної сферично-кутової інтерполяції векторів, що застосовується у задачах комп’ютерної графіки та візуалізації орієнтацій.

1 БАГАТОВАРІАНТНИЙ АНАЛІЗ РОЗВ’ЯЗКУ ОСНОВНОЇ ЗАДАЧІ

Головною задачею роботи є створення графічного наповнення інформативного вебсайту, присвяченого життю та діяльності Сергія Павловича Корольова. Для забезпечення цілісного візуального стилю, високої якості зображень та оптимальної швидкодії вебресурсу необхідно обґрунтовано обрати типи графіки та засоби для їх створення.

У межах розділу проводиться багатоваріантний аналіз типів графіки, визначення їхніх властивостей, переваг і недоліків, а також аналіз інструментів, які можуть бути використані для реалізації графічного наповнення сайту.

1.1. Вибір типу графіки

Для наповнення вебсайту застосовуються два основних типи комп’ютерної графіки: растрова та векторна. Кожен із них виконує різну функцію у структурі сайту, а тому їх вибір потребує окремого обґрунтування.

Растрова графіка базується на наборі пікселів, тому найкраще підходить для реалістичних матеріалів, фотографій і текстур. Вона дозволяє точно передати дрібні деталі та складні колірні переходи, але втрачає якість при масштабуванні та потребує оптимізації розміру файлу для вебформатів.

Векторна графіка складається з математично описаних ліній і кривих. Вона забезпечує бездоганне масштабування, чіткість контурів та малу вагу файлів. Саме тому її доцільно використовувати для схем, іконок, лінійних портретів і декоративних елементів сайту. Зокрема, поширеним векторним стилістичним прийомом є стиль контурних ліній.

1.2. Застосування растрової графіки

Загалом, у роботі обраний комбінований підхід: растрова графіка обрана для тих елементів, у яких пріоритетом є реалістичність або наявність природної текстури. Вона буде використана для:

* історичних фотографій Сергія Корольова;
* документальних кадрів запусків ракет;
* атмосферних фонів та текстур;
* GIF-анімацій.

Її застосування обмежується тими матеріалами, де втрата якості при масштабуванні не є значущою, або де важливо зберегти реальність зображення.

1.3. Застосування векторної графіки

Векторна графіка обрана для створення елементів, яким необхідна чіткість контурів і можливість масштабування без втрати якості. У межах сайту вона доцільна для:

* портретних лінійних ілюстрацій;
* іконок і декоративних елементів;
* схем та силуетів ракет;
* стилізованих космічних лінійних композицій.

Тут уточнюється: чи створюються файли у векторному форматі (SVG), чи використовується векторний інструментарій для побудови зображень, що потім експортуються у PNG.

У роботі обраний комбінований підхід: векторні інструменти застосовуються для побудови чітких контурів, але готові файли зберігаються переважно у растровому форматі PNG з прозорістю. У тих випадках, де потрібна повна векторність (наприклад, прості іконки), може використовуватися формат SVG.

1.4. Вибір інструментів графічної розробки

У цьому підрозділі лише обираються інструменти; детальне використання подано в іншому розділі.

1.4.1. Canva

Canva була обрана як інструмент для підготовки:

* GIF-анімацій;
* простих композицій;
* фонових колажів;
* попередніх ескізів сторінок.

Її переваги полягають у простоті створення анімацій, наявності шаблонів та швидкому експорту GIF. Недоліки — обмежена точність ручного малювання та слабкий контроль над векторними кривими.

1.4.2. Photopea

Photopea обрано як основний засіб опрацювання зображень. Це растровий редактор, що підтримує шарову структуру, маски та більшість функцій, подібних до Adobe Photoshop. Важливо, що Photopea також має векторні інструменти (Pen Tool, Shape Tool), що дозволяє створювати контурні рисунки.

Ключові характеристики Photopea:

* растровий редактор з підтримкою векторного інструментарію;
* робота у браузері без встановлення програм;
* підтримка PSD, PNG, JPG, SVG;
* експорт як у растрові, так і у векторні формати.

Аналоги та чому вони гірші

1. GIMP — не підтримує векторні інструменти на рівні редакторів After Effects чи Photoshop; менш зручний інтерфейс;
2. Pixlr — менше функцій, обмежена підтримка шарів;
3. PhotoScape X — орієнтований на фільтри, а не на професійне редагування;
4. Krita — чудовий для цифрового малювання, але не для створення PNG-іконок чи векторизації.

У підсумку Photopea забезпечує оптимальний баланс між доступністю, функціональністю та підтримкою форматів.

1.5. Підсумковий вибір стилю графіки

Для проєкту обрано комбінований підхід, у якому:

* векторні інструменти застосовуються для побудови чітких контурних ілюстрацій;
* растрові формати використовуються для фінальних PNG-іконок і всіх фото;
* GIF-анімації створюються у Canva;
* векторизація за потреби виконується засобами Photopea.

Висновок

Отже, для створення графічного наповнення вебсайту обрано такі засоби та підходи:

* Photopea — як основний растровий редактор із векторними інструментами;
* Canva — для GIF-анімацій та композицій;
* растрова графіка — для фото та GIF;
* векторні інструменти — для схем, іконок та лінійних портретів;
* комбінований стиль — для збереження точності й водночас адаптації під вебформати.

1.6. Вибір мови та середовища для програмної реалізації

Для технічної розробки проєкту необхідно розділити стек технологій на дві складові: верстка вебсторінок та розробка програмного модуля.

1.6.1. Веброзробка

HTML5 та CSS3 Для створення структури та стилізації вебсайту обрано класичний стек технологій:

1. HTML5 (HyperText Markup Language) — використовується для розмітки контенту, створення семантично правильної структури сторінок, інтеграції зображень та мультимедіа.
2. CSS3 (Cascading Style Sheets) — застосовується для візуального оформлення, адаптації під різні екрани, налаштування шрифтів, кольорів (згідно з обраною палітрою) та анімаційних ефектів.

Середовище розробки: Visual Studio Code (VS Code) Для написання коду HTML та CSS обрано редактор VS Code. Це легкий, швидкий та розширюваний редактор коду. Він підтримує підсвічування синтаксису, автодоповнення (Emmet), а також має вбудований термінал. Ключовою перевагою є наявність розширення Live Server, яке дозволяє переглядати зміни на сайті в реальному часі без необхідності постійного оновлення сторінки вручну.

1.6.2. Програмна компонента

Java та JavaFX Для реалізації інтерактивної програмної частини проєкту (окремого модуля або десктопного додатку) обрано об'єктно-орієнтовану мову програмування Java. Java – це надійна, кросплатформова мова зі строгою типізацією, що забезпечує стабільність роботи програми.

JavaFX – це сучасна платформа для створення графічних інтерфейсів користувача (GUI) на Java. Вона дозволяє створювати багаті інтерфейси з використанням FXML розмітки та CSS-стилізації, що концептуально схоже на веброзробку, але для десктопних систем.

Середовище розробки: IntelliJ IDEA Для роботи з Java обрано інтегроване середовище розробки (IDE) IntelliJ IDEA від JetBrains. Це найпотужніше середовище для Java-розробників. Воно надає інструменти для інтелектуального аналізу коду, швидкого рефакторингу та зручної відладки (debugging). Ключовою перевагою є відмінна інтеграція з JavaFX та системами збірки (Maven/Gradle), що значно пришвидшує процес розробки програмного забезпечення.

2 ВИБІР КОЛЬОРОВОЇ ГАМИ

Колір є одним із ключових інструментів у створенні емоційного та візуального враження на користувача. Для сайту, присвяченого Сергію Корольову та розвитку космічної галузі, вибір кольорової гами відіграє особливо важливу роль, адже саме колір здатен передати атмосферу космосу, глибини всесвіту та технологічного прогресу.

Правильно підібрана кольорова палітра підсилює зміст сторінок, допомагає користувачу краще сприймати інформацію, формує цілісний стиль і акцентує увагу на ключових елементах. Контраст між темним фоном та яскравими акцентами забезпечує хорошу читабельність і додає дизайну сучасності.

Основні принципи, враховані при виборі кольорів:

1. Контраст. На сайті використано темні відтінки синього як фон — вони асоціюються з космосом, нічним небом і глибиною. Жовтий колір виступає контрастним акцентом, що привертає увагу до важливих елементів (кнопок, заголовків, декоративних деталей).
2. Гармонія. Використання близьких за відчуттям синіх тонів створює цілісність і плавність дизайну. Темний і глибокий синій формує атмосферу серйозності й науковості, а світліші сині відтінки можуть використовуватися для підзаголовків або декору.
3. Психологія кольору.

Синій традиційно символізує науку, точність, технічність, спокій та глибину — що ідеально підходить для теми космічних досліджень.

Жовтий додає відчуття енергії, відкриттів, інновацій та динаміки. Він підсилює увагу та спрямовує погляд користувача на головні інформаційні блоки.

1. Теплі й холодні кольори. Холодна синя гама балансується теплим жовтим — це створює емоційний баланс та не дає сайту виглядати надто “холодним” або “технічним”.
2. Технічний підбір палітри (Adobe Color). Для забезпечення математично точної гармонії відтінків було використано інструмент Adobe Color. Обрано правило комплементарної гармонії, при якому поєднуються кольори, розташовані на протилежних сторонах колірного кола. Задавши базовий «космічний» синій колір, система автоматично визначила ідеальний контрастний жовтий відтінок. Це дозволило отримати пару, яка гарантує максимальну виразність елементів інтерфейсу на темному тлі.

Результат генерації колірної схеми в середовищі Adobe Color та фінальні коди наведено на рисунку 2.1.

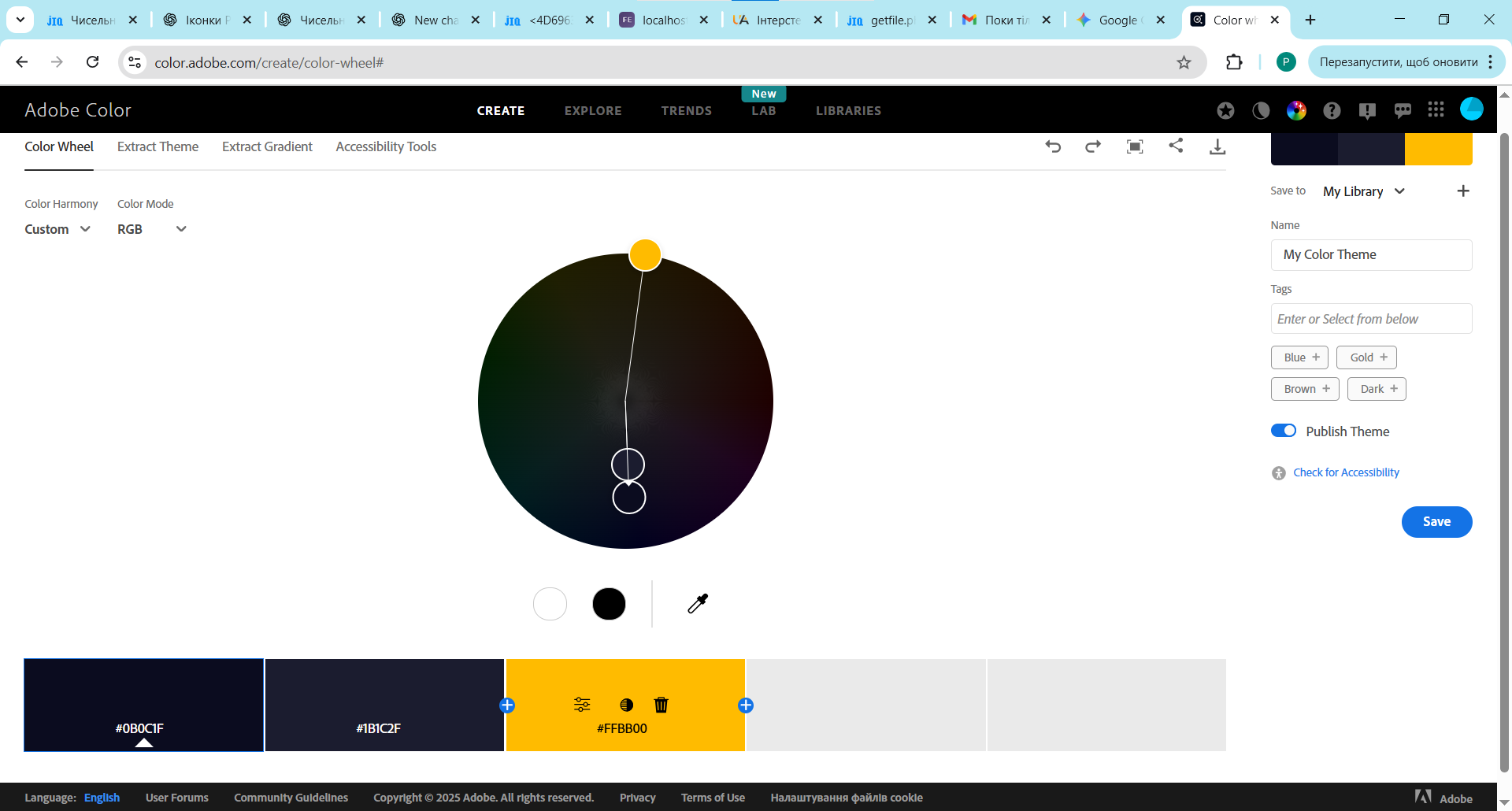


Рисунок 2.1 — Основні кольори сайту

Затверджена кольорова схема включає наступні коди:

1. Темно-синій (#0b0c1f). Використовується як фон заголовку сторінки. Це найтемніший відтінок, що створює ефект глибини.
2. Світло-синій (#1b1c2f). Загальний фон, хедер. Він підтримує загальну синю композицію, не відволікаючи увагу. Використовується як базовий колір, що формує атмосферу «космосу», та забезпечує максимальну контрастність із текстом.
3. Жовтий (#ffbb00). Основний акцентний колір. Згенерований як комплементарний до синього фону. Використовується для кнопок, важливих елементів, маркерів, окремих частин графіки. Створює візуальний контраст і допомагає користувачу швидше орієнтуватися.

Висновок.  
Отже, ретельно підібрана кольорова гама забезпечує не лише естетичну привабливість сайту, а й підтримує його функціональність, зручність сприйняття та тематичну єдність. Комплементарне поєднання темно-синього та жовтого ефективно передає атмосферу космічних досліджень і підкреслює ключові інформаційні елементи, роблячи сайт сучасним, гармонійним та зрозумілим для користувача.

3 РОЗРОБКА ЛОГОТИПУ САЙТУ

Логотип є важливим елементом айдентики вебсайту, оскільки він формує перше враження про проєкт і допомагає користувачу асоціювати сайт із його тематикою. Якісний логотип має бути простим, впізнаваним, доречним до змісту сайту та добре відтворюватися у різних розмірах і форматах. У комп’ютерній графіці логотипи умовно поділяють на два основні типи: іконічні (символьні) та комбіновані, що поєднують графічний елемент із текстовим.

Іконічні логотипи використовують символ або графічний елемент без тексту. Такі логотипи легко сприймаються, швидко запам’ятовуються і добре працюють у цифровому середовищі. Вони передають зміст через форму та образ — наприклад, схематична ракета, планета чи силует об’єкта. Комбіновані логотипи, своєю чергою, додають текстову частину, зазвичай назву компанії чи проєкту. Це дозволяє одночасно передати і візуальний образ, і вербальний зміст, але робить логотип дещо складнішим за структурою.

У межах створення сайту про Сергія Корольова було проаналізовано обидва підходи. Оскільки сайт присвячений космічній тематиці та особистості головного конструктора ракетно-космічних систем, найбільш доречним рішенням став іконічний логотип. Такий тип дозволяє передати космічну символіку без зайвих деталей і зробити логотип максимально універсальним для використання у хедері, favicon, меню та мобільній версії сайту.

Етап 1. Створення контуру логотипу

На першому етапі було створено базовий контур логотипу у вигляді стилізованої ракети та пускової вежі. Для побудови форми використовувалися прості геометричні елементи, зокрема прямокутники, еліпси та трикутники. Контур логотипу було намальовано за допомогою інструментів Canva та Photopea, що дозволило швидко отримати чіткі форми та рівномірні лінії. Основний акцент був зроблений на мінімалізмі та впізнаваності силуету. Результат показано на рисунку 3.1.

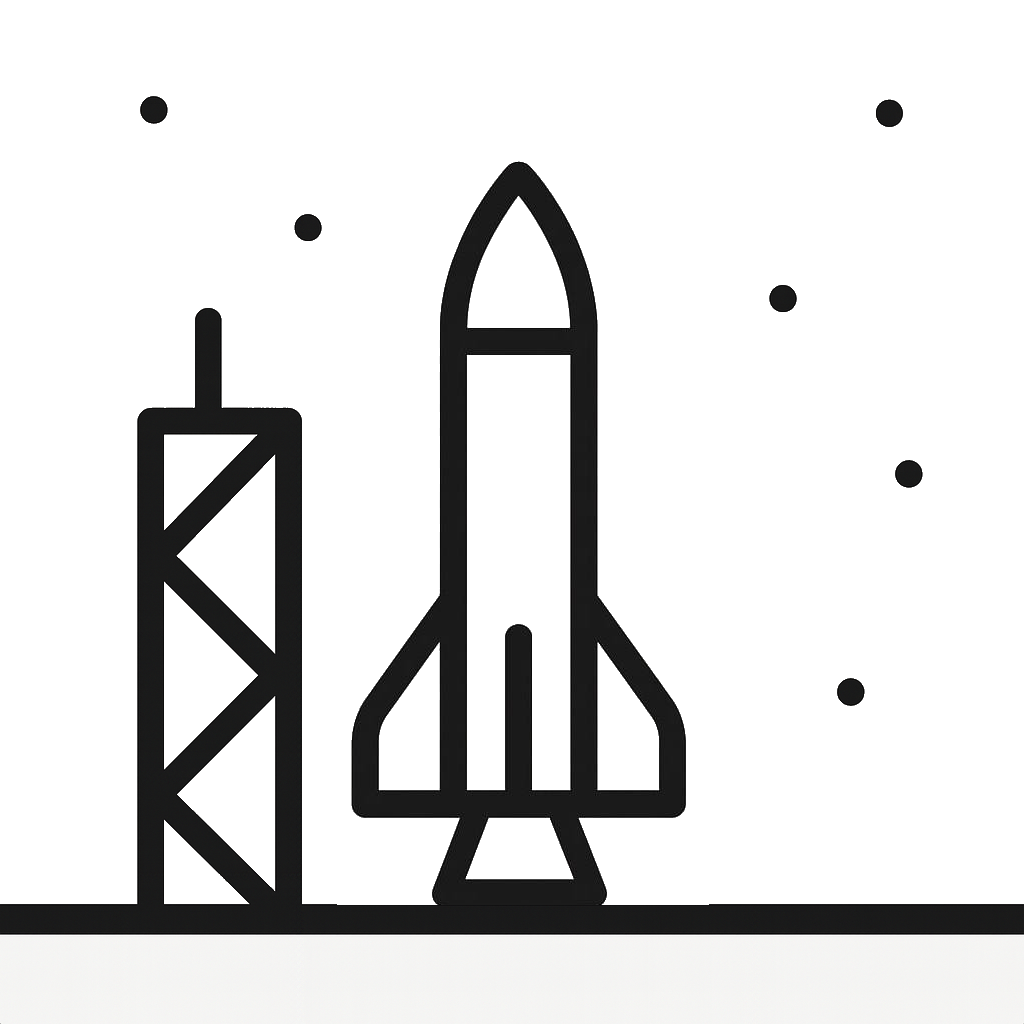


Рисунок 3.1 – Контур логотипу (початковий етап)

Етап 2. Підбір кольорів і заливка елементів

Після створення контуру було розпочато роботу над заповненням логотипу кольором. За допомогою інструмента «Піпетка» в Canva були вибрані основні відтінки, що відповідають загальній кольоровій гамі сайту:

* глибокий синьо-графітовий (#1f2a35) для фону;
* молочно-білий (#f2f2f2) для корпусу ракети;
* темно-сірий (#2a2a2a) для технічних деталей та обведення;
* світлі акцентні точки для зірок на тлі.

Таке поєднання кольорів створює відчуття космічності, технологічності та мінімалізму. Фон підсилює тематику нічного неба, а біла ракета контрастує та виділяє основний символ. Кольорова схема відповідає загальному дизайну сайту та гармонійно інтегрується у шапку та головну сторінку.

Готовий логотип показано на рисунку 3.2.



Рисунок 3.2 – Готовий логотип сайту

Висновок до розділу

Отже, у процесі розробки було створено простий, стильний та тематично відповідний іконічний логотип. Він вдало відображає космічну спрямованість сайту, пов’язану з діяльністю Сергія Корольова, та відповідає принципам комп’ютерної графіки — лаконічність, впізнаваність, контрастність та гармонія кольору.

4 РОЗРОБКА СТАТИЧНИХ ЗОБРАЖЕНЬ

У сучасному цифровому середовищі статичні графічні зображення відіграють важливу роль у візуальній комунікації. При розробці вебсайтів зазвичай використовуються два типи графіки: растрова та векторна.

Растрова графіка складається з пікселів і дозволяє передавати складні переходи кольорів, тіні та фотореалістичні ефекти, проте втрачає якість при масштабуванні. Найпопулярніші формати: JPEG, PNG. Векторна графіка будується на математичних формулах (лініях, кривих), що дозволяє змінювати розмір зображення без втрати якості. Стандартний формат: SVG.

Для створення візуальних елементів у межах даної роботи було використано онлайн-редактор Photopea. Цей інструмент підтримує роботу як з растровими, так і з векторними об’єктами, що дозволило застосувати гібридний підхід. Хоча більшість форм (ракети, планети) створювалися векторними інструментами (Pen Tool, Shape Tool) для забезпечення чіткості контурів, кінцеві зображення були експортовані у растровий формат PNG. Цей вибір зумовлений необхідністю використання складних ефектів (світіння, розмиття, текстурні пензлі) та прозорого фону, що найкраще реалізується саме в форматі PNG.

У межах проєкту розроблено серію графічних елементів на космічну тематику: планета Земля, комета, ракета Р-7, супутники, портрет Корольова та фонові композиції.

4.1 Створення графічного зображення «Планета Земля»

Першим етапом було створення базової форми Землі. За допомогою інструмента Ellipse Tool було намальовано коло, що слугує основою планети.

Додавання континентів. Інструмент Pen Tool використано для побудови нерівних контурів континентів. Кожен елемент створено на окремому шарі для подальшої зручної обробки (рис. 4.1).



Рисунок 4.1 – Контур планети Земля

Заливка та деталі. Використано інструменти Fill, Gradient і Stroke для додавання кольорів, тіней та легкого блиску на поверхні Землі. Завдяки цьому зображення набуло глибини та об’єму (рис. 4.2).



Рисунок 4.2 – Зафарбоване зображення планети Земля

4.2 Створення графічного зображення «Комета»

Комета була створена не як вектор, а як шарове растрове зображення, що складається з кількох частин:

* основне ядро (Ellipse Tool + градієнтна заливка);
* хвіст комети (Brush Tool з м’якими краями);
* ефекти руху (Blur → Motion Blur)(рис.4.3).

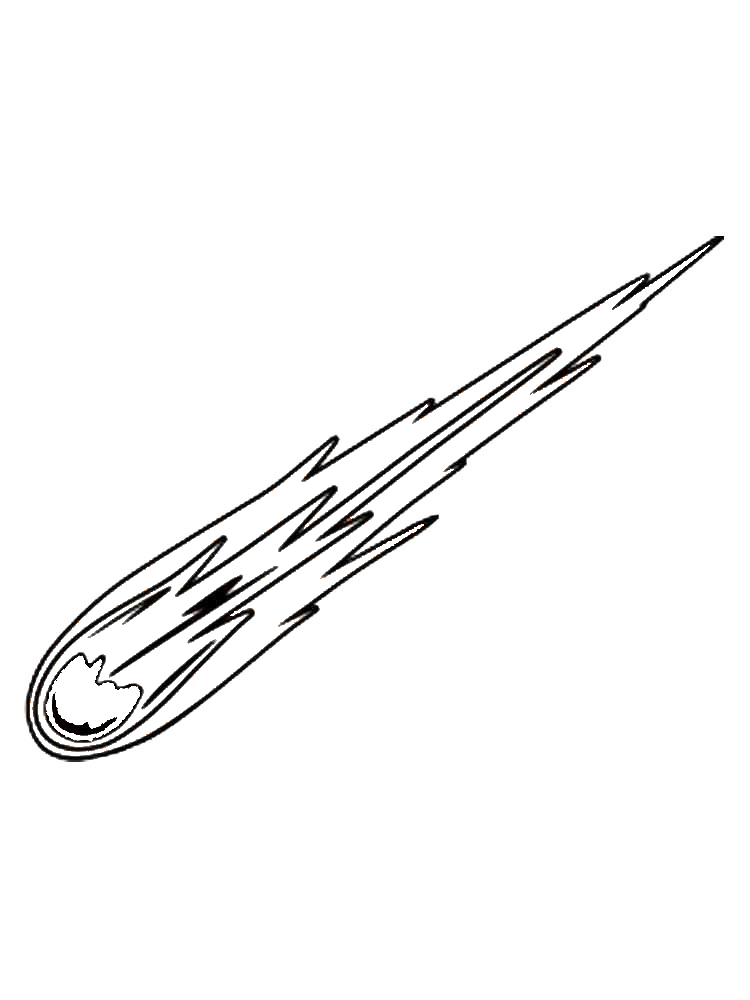


Рисунок 4.3 – Контур комети

Комета містить кілька напівпрозорих шарів, які накладаються один на одного для створення ефекту світіння та руху (рис. 4.4).

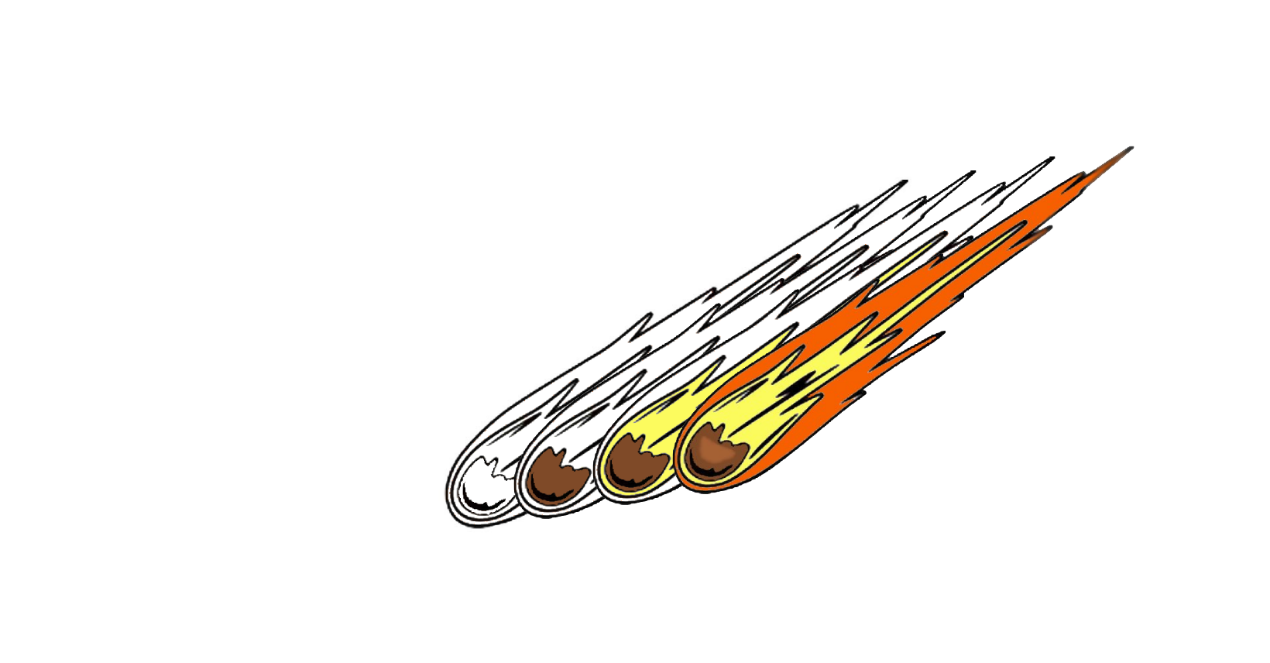


Рисунок 4.4 – Шарове зображення комети

4.3 Створення графічного зображення «Ракета Р-7»

Створення ракети розпочато з побудови основного контуру за допомогою Pen Tool. Ракета складається з центрального корпусу та бічних блоків.

Деталізація. За допомогою Shape Tool та Pen Tool додано:

* сопла;
* вікна;
* обводку;

декоративні елементи корпусу.

Заливка. Кожна частина має власну заливку та тіні, що створює реалістичний вигляд ракети (рис. 4.8).

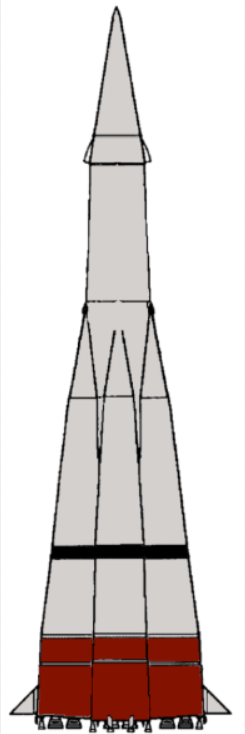


Рисунок 4.8 – Зафарбоване зображення ракети Р-7

4.4 Створення графічного зображення «Супутник-1»

Основою супутника є сфера, створена інструментом Ellipse Tool. Металевий блиск сформовано через градієнтні заливки та тінювання (рис. 4.9).

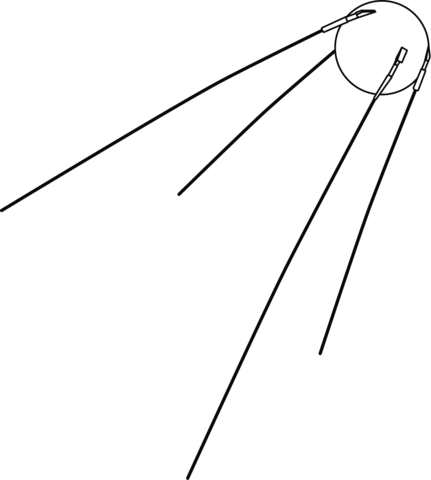


Рисунок 4.9 – Контур супутника Супутник-1

Антени виконані інструментом Pen Tool та мають легку обводку для підсилення контрасту. Зображення є стилізованою інтерпретацією супутника, наближеною до історичного вигляду (рис. 4.10).

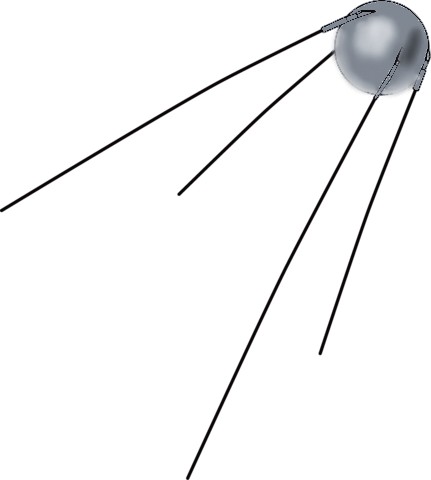


Рисунок 4.10 – Зафарбоване зображення супутника Супутник-1

4.5 Створення графічного зображення «Новий супутник»

Новий супутник складається з кількох геометричних форм:

* основний корпус (Rectangle Tool + обведення);
* сонячні панелі (Polygon Tool);
* антени та кріплення (Pen Tool) (рис. 4.11).

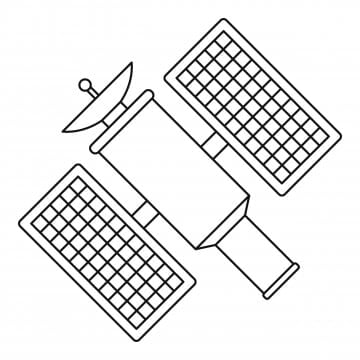


Рисунок 4.11 – Контур сучасного супутника

Застосовано шарові ефекти: Inner Shadow, Outer Glow, Gradient Overlay — для додання технічного вигляду (рис. 4.12).

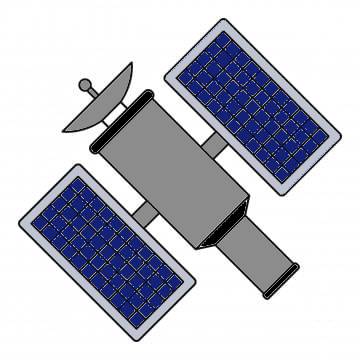


Рисунок 4.12 – Зафарбоване зображення сучасного супутника

4.6 Портрет Сергія Корольова

Портрет створено у стилі лінійного малюнка. За допомогою інструмента Brush Tool із твердим краєм накреслено контурні лінії голови, рис обличчя та одягу (рис. 4.13).



Рисунок 4.13 – Контур портрета Сергія Корольова

Фінальний штрих — зафарбовування (рис. 4.14).



Рисунок 4.14 – зафарбований портрет Сергія Корольова

4.7 Космос із Землею, орбітою та супутником

Це найкомплексніше зображення, яке поєднує елементи, створені раніше:

космічний фон (Brush Tool, маленький розмір, білий колір — імітація зоряного неба);

* Земля (готовий окремий графічний шар);
* орбіта (Ellipse Tool з білою обводкою);
* супутник-1 (вставлений із попередньо створеного шару).

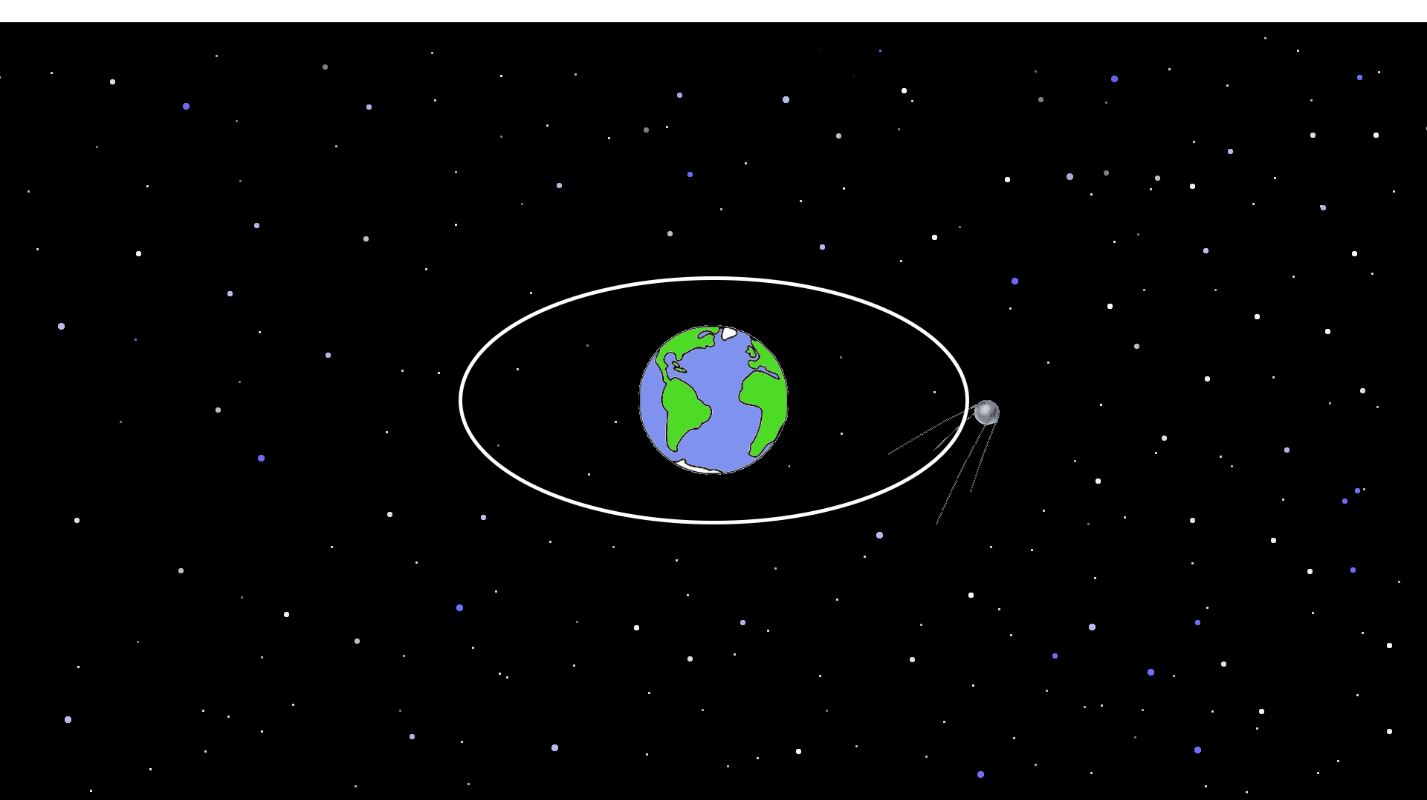
Для глибини фон містить кілька шарів з різною прозорістю (рис. 4.15).  


Рисунок 4.15 – Колаж з Землею, космосом, Супутником-1 та орбітою

4.8 Фон «Космос із Землею»

Це спрощена версія попереднього зображення без супутника та без орбіти. Основні елементи — стилізована Земля та зорі, створені інструментом Brush Tool (рис. 4.16).



Рисунок 4.16 – Колаж з Землею та космосом

4.9 Контурне зображення «Ракета»

Ракета виконана у стилі контурної графіки. Використано чіткі векторні лінії (Pen Tool + Stroke). Форма ракети максимально спрощена, що робить її придатною для інфографіки (рис. 4.17, рис. 4.18).

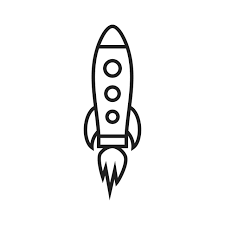


Рисунок 3.17 – Контур зображення ракети

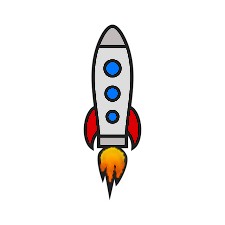


Рисунок 4.18 – Зафарбоване зображення ракети

4.10 Космічний шатл

Шатл змодельовано за допомогою поєднання Pen Tool та Shape Tool. Корпус створено як кілька складених фігур, після чого додано:

* обведення;
* затемнення під крилами;
* вікна;
* реактивні елементи (рис. 4.19).

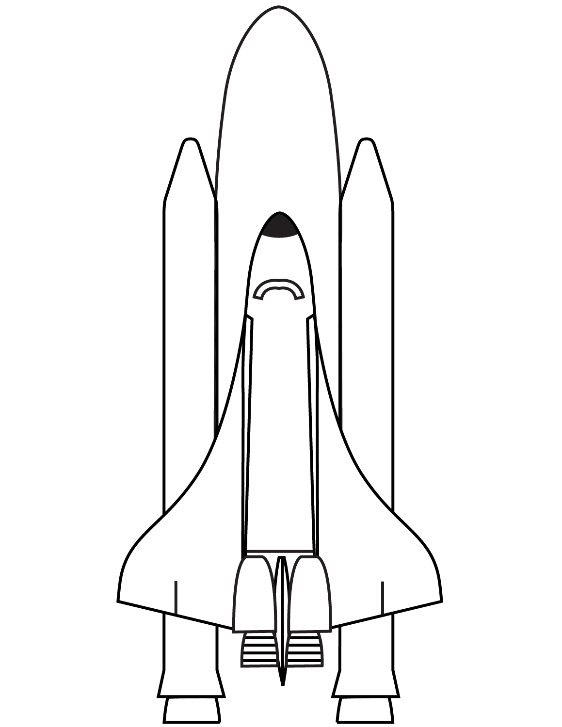


Рисунок 4.19 – Контур космічного шатла

Фінальний результат — чисте технічне зображення (рис. 4.20).

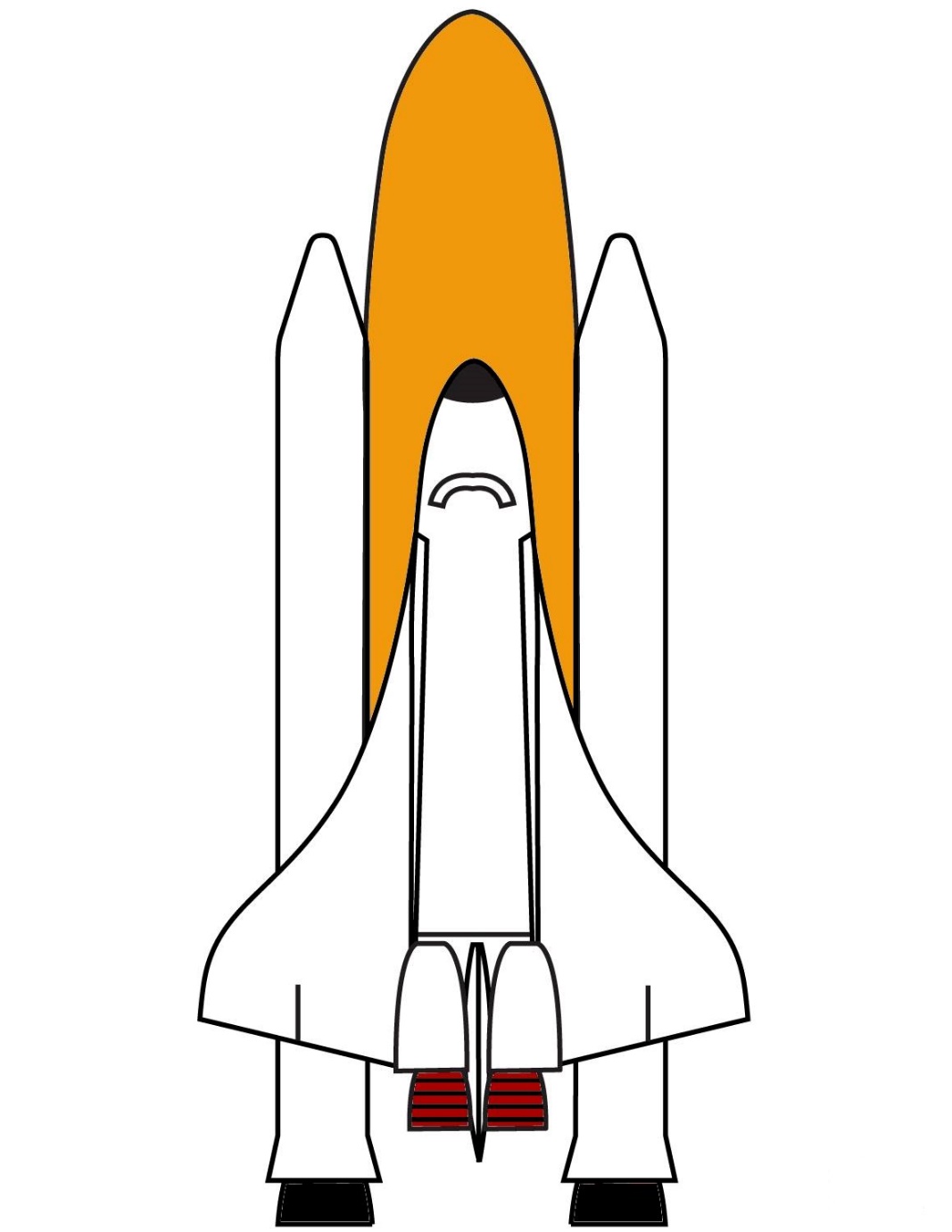


Рисунок 4.20 – зафарбоване зображення косміного шатла

Висновок до розділу 4

Отже, у четвертому розділі було розроблено пакет графічних матеріалів, необхідних для візуального оформлення вебсайту. Використання векторних інструментів для побудови форм та растрових ефектів для фіналізації дозволило отримати якісні зображення з прозорим фоном у форматі PNG, які гармонійно вписуються у темну стилістику сайту. Повний перелік розроблених графічних елементів наведено у Додатку А.

5 РОЗРОБКА ДИНАМІЧНИХ ЗОБРАЖЕНЬ

Динамічні зображення, або анімації, представляють собою важливий елемент еволюції візуальних технологій, що дозволяє оживити статичний інтерфейс вебсайту. Розробка таких зображень обумовлена необхідністю привернення уваги користувача та візуалізації складних процесів. Розвиток динамічних зображень тісно пов'язаний із прогресом у сфері вебтехнологій та збільшенням пропускної здатності мереж, що дозволяє використовувати графіку високої якості без затримок у завантаженні.

Основними видами анімацій, що використовуються у веброзробці, є: 2D-векторна анімація, кадрова (GIF) анімація та 3D-анімація.

2D-векторна анімація — це форма анімації, яка використовує математичні описи кривих для створення руху об'єктів. Її перевагою є малий розмір файлів та чіткість на будь-яких екранах.

3D-анімація представляє технічно складний підхід, що використовує тривимірні моделі та сцени. Вона забезпечує високий рівень реалізму та глибини, що активно застосовується у презентаційних матеріалах космічної тематики.

Кадрова анімація (Frame-by-Frame) — це класичний метод, де ілюзія руху створюється шляхом швидкої зміни послідовності статичних зображень (кадрів). Саме цей метод є найбільш доцільним для створення легких анімованих елементів у форматі GIF для вебсайтів.

У процесі роботи над динамічним зображенням для сайту було використано графічне середовище Canva. Застосовуючи принцип «одна сторінка — один кадр», було розроблено сюжетну анімацію зустрічі ракети з кометою. Процес створення кадрів та налаштування часових інтервалів показано на рисунку 5.1.

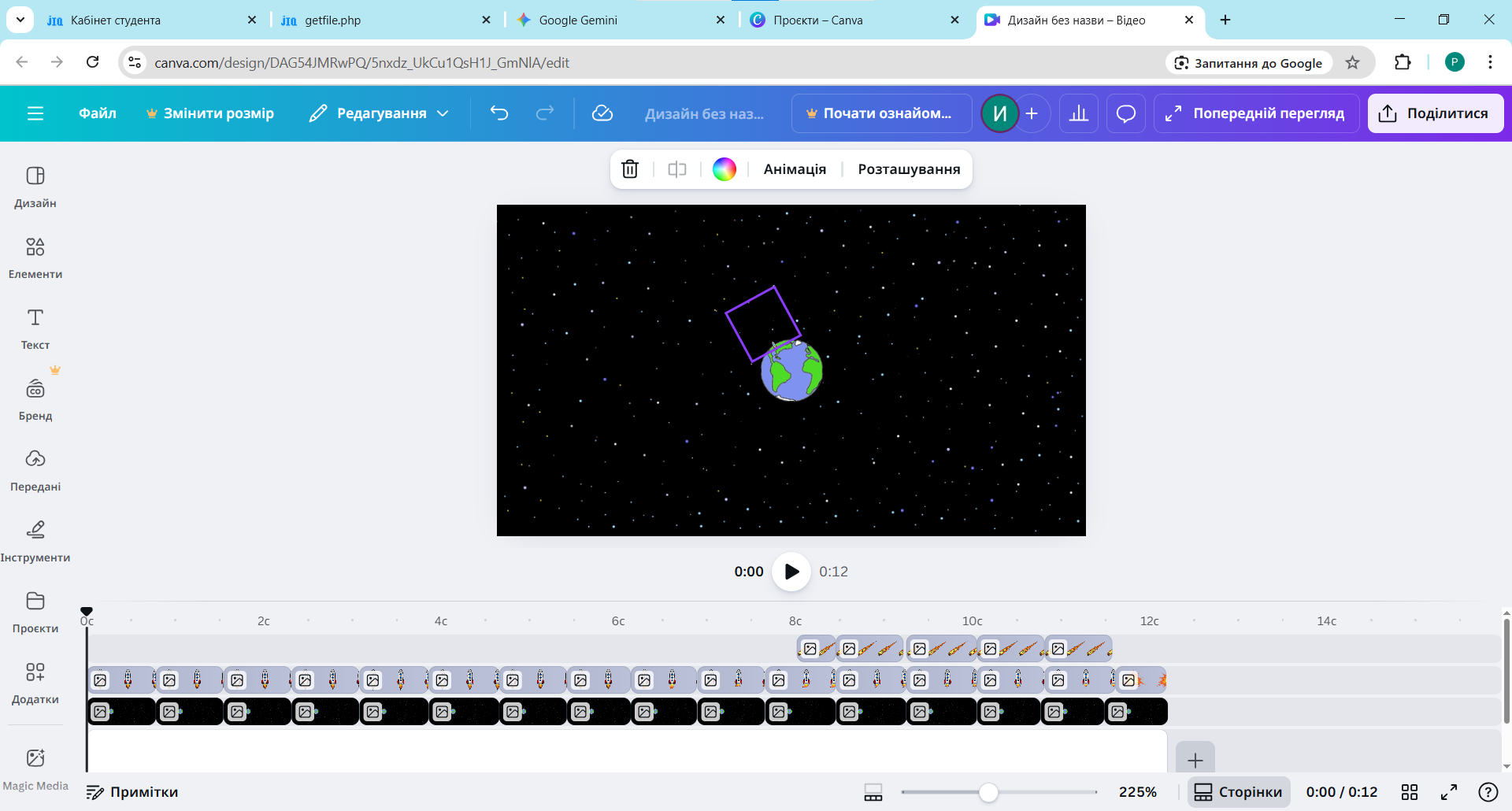


Рисунок 5.1 – Інтерфейс розробки анімації у середовищі Canva

Для реалізації плавності руху було налаштовано часову шкалу (Timing). Кожен кадр демонструється протягом короткого інтервалу (0.1–0.2 с), що забезпечує динамічність сцени (див. рисунок 5.2).



Рисунок 5.2 – Налаштування часових інтервалів між кадрами

Сюжет анімації розбито на логічні етапи, кожен з яких відповідає за певну частину руху об'єктів. Але сама ідея анімації у тому що ракетами можна буде збивати об’єкти що летять в сторону Землі і можуть загрожувати людству. Детальний опис розробки кадрів наведено нижче:

1. Етап масштабування (Кадри 1–6). Перші 6 кадрів відповідають за ефект наближення ракети. На кожному наступному кадрі розмір об'єкта «Ракета» пропорційно збільшується,створюючи ілюзію, що вона летить з пускового майданчика на Землі до глядача.
2. Етап прямолінійного руху (Кадри 7–10). Наступні 4 кадри демонструють розворот. Ракета, досягнувши максимального розміру, починає зміщуватися вздовж вертикальної осі вгору. Далі вона розвертається в ліву сторону з нахилом вниз.
3. Етап зближення з кометою (Кадри 10–15). У кадрі №11 у нижньому лівому куті сцени з'являється новий об'єкт — «Комета». Протягом 4 кадрів відбувається одночасний рух двох об'єктів назустріч один одному по діагоналі: комета рухається вгору-вправо, а ракета — вниз-вліво. Швидкість зміни позицій збільшена для імітації стрімкості події.
4. Етап зіткнення (Кадр 16). Останній кадр фіксує момент контакту об'єктів. Замість ракети та комети відображається графічний елемент «Вибух» (спалах), що займає центральну частину композиції, символізуючи фінал сюжету.

Повна розкадровка створеної анімації продемонстрована в Додатку Б.

6 ОЦІНКА ОБСЯГІВ ЗОБРАЖЕНЬ

У сучасному вебдизайні, особливо для сайтів з насиченою візуальною складовою, як-от проєкт про космос та авіацію, оптимізація графічного контенту є критично важливим етапом розробки. Високоякісні зображення ракет, деталізовані схеми та глибокі фони зоряного неба можуть мати значний обсяг, що без належної обробки призведе до повільного завантаження сторінок та погіршення взаємодії з користувачем (UX).

Кількість та вага зображень безпосередньо впливають на швидкість рендерингу вебсайту. Враховуючи, що користувачі можуть переглядати сайт про Сергія Корольова з мобільних пристроїв або через лімітований інтернет-зв'язок, важливо знайти баланс між естетикою космічної тематики та технічною продуктивністю.

Вибір формату зображення визначає співвідношення між обсягом файлу та якістю відтворення. У проєкті були проаналізовані формати JPEG (для фотографій), PNG (для схем та об’єктів з прозорим фоном) та GIF (для анімацій). Механізм стиснення дозволив значно зменшити вагу файлів без видимої втрати чіткості, що є критичним для технічних креслень та текстур.

Для забезпечення швидкого завантаження вебсайту та зменшення навантаження на сервер було застосовано комплексний підхід до оптимізації, який включає такі методи:

1. Зміна роздільної здатності (Resizing). Оригінальні зображення часто мають надлишкову роздільну здатність (наприклад, 4K). Для вебсайту всі зображення були зменшені до розмірів, які фактично відображаються на екрані (наприклад, іконки до 64x64 пікселів, фото в галереї — до 1200 пікселів по ширині).
2. Кадрування (Cropping). У багатьох зображеннях (наприклад, «Ракета», «Супутник») було обрізано зайвий прозорий простір навколо об'єкта. Це дозволило зменшити розмір файлу та спростити верстку.
3. Використання інструментів компресії. Для автоматичної оптимізації без втрати візуальної якості використовувалися онлайн-сервіси (наприклад, TinyPNG) та вбудовані алгоритми експорту редактора Photopea. Це дозволило зменшити вагу PNG-файлів у середньому на 40–60%.
4. Lazy Loading (Відкладене завантаження). Для великих фонових зображень та галереї передбачено використання технології Lazy Loading [8]. Це дозволяє завантажувати графіку лише тоді, коли користувач доскролює до відповідної секції, що значно пришвидшує первинне відображення сторінки.

Після застосування методів оптимізації було проведено оцінку кінцевих розмірів файлів. Результати аналізу окремих графічних елементів наведено в таблиці 6.1.

Таблиця 6.1 – Оцінка розмірів графічних зображень

|  |  |  |
| --- | --- | --- |
| **Назва графічного зображення** | **Тип файлу** | **Розмір графічного зображення, кб** |
| r7.png | PNG | 46,9 |
| ракета1png.jpg | JPG | 6,05 |
| rocket.jpg | JPG | 54,8 |
| zemlya1.png | PNG | 123 |
| suputnyk-2.jpg | JPG | 19,8 |
| Untitled(1).png | PNG | 35,4 |
| sputnik-1.jpg | JPG | 14,9 |
| Дизайн без назви (1).gif | GIF | 108 |
| колаж.png | PNG | 39,8 |
| logo.png | PNG | 24,2 |
| korolov.jpg | JPG | 71,9 |

Щоб оцінити загальне навантаження на пропускну здатність каналу при відвідуванні різних розділів вебсайту, було просумовано вагу всіх графічних елементів для кожної сторінки. Зведені дані представлені в таблиці 6.2.

Таблиця 6.2 – Розміри зображень на сторінках сайту

|  |  |  |
| --- | --- | --- |
| **Назва сторінки** | **Кількість зображень** | **Загальний розмір графічних зображень, кб** |
| Головна сторінка |  |  |
| Біографія С.П. Корольова | 1 | 71,9 |
| Ракета Р7 | 3 | 143,15 |
| Супутник | 4 | 140,1 |
| Досягнення | 2 | 158,4 |

Висновок до розділу 6

Отже, у цьому розділі було проаналізовано принципи та методи оптимізації графічного контенту. Завдяки використанню сучасних форматів стиснення, правильному кадруванню та налаштуванню роздільної здатності вдалося досягти оптимального розміру файлів. Як видно з таблиці 6.2, загальний обсяг сторінок не перевищує критичних значень, що гарантує швидке завантаження сайту та стабільну роботу анімаційних ефектів навіть при повільному інтернет-з'єднанні.

7 ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ ТА ВЕБСАЙТУ

Сьогодні успіх інформаційного ресурсу залежить не лише від його дизайну та змістовного наповнення, а й від стабільності роботи та коректного відображення на різних пристроях. Для забезпечення якості розробленого вебсайту та програмного модуля було використано комплексний підхід до тестування, що включає перевірку функціональності, сумісності (кросбраузерності) та продуктивності.

Метою етапу тестування є виявлення та усунення можливих помилок у HTML-розмітці, CSS-стилях та логіці роботи Java-додатку перед фінальною здачею проєкту.

7.1 Функціональне тестування Це основний етап, спрямований на перевірку того, чи виконує система покладені на неї завдання. У межах вебсайту було протестовано:

Навігація: коректність роботи гіперпосилань між сторінками (Головна, Біографія, Галерея, Контакти). Перевірено відсутність «битих» посилань.

Медіаконтент: правильність відображення зображень (формати PNG, JPG) та відтворення GIF-анімацій (політ ракети).

Інтерактивні елементи: реакція кнопок та меню на наведення курсору (hover-ефекти), що були реалізовані за допомогою CSS.

У межах програмного модуля (Java/JavaFX) перевірено:

Запуск додатку та ініціалізація графічного інтерфейсу.

Коректність роботи логічних алгоритмів.

7.2 Тестування на сумісність (UI/UX Testing) Оскільки користувачі можуть переглядати інформацію про Сергія Корольова з різних пристроїв, було проведено перевірку верстки:

Кросбраузерність: сайт коректно відображається у браузерах Google Chrome, Mozilla Firefox та Microsoft Edge. Стилі CSS (Flexbox/Grid) інтерпретуються однаково, зберігаючи структуру макета.

Адаптивність: перевірено читабельність шрифтів та розташування блоків на екранах з різною роздільною здатністю. Темно-синій фон та жовті акценти залишаються контрастними та розбірливими за будь-яких умов.

7.3 Тестування продуктивності Цей етап дозволив оцінити, наскільки ефективно були проведені заходи з оптимізації графіки (описані у Розділі 6).

Швидкість завантаження: завдяки стисненню зображень та чистому HTML-коду сторінки завантажуються миттєво.

Плавність анімації: векторна та GIF-анімація відтворюється без затримок (лагів) навіть на слабких ПК, що підтверджує правильність вибору форматів.

Результат фінальної перевірки головної сторінки розробленого ресурсу продемонстровано на рисунку 7.1.

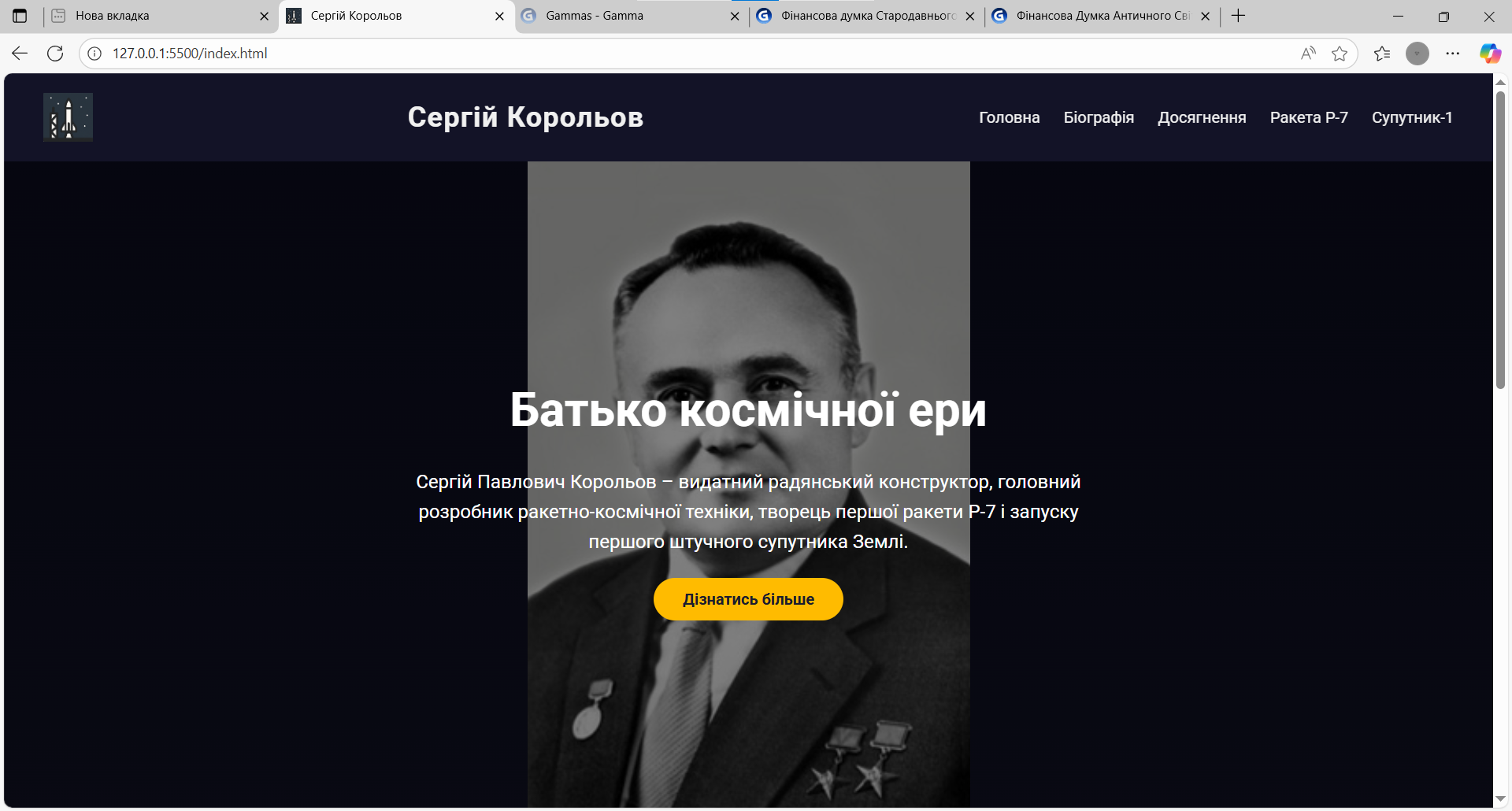


Рисунок 7.1 – Головна сторінка розробленого вебсайту

Підсумкове тестування було проведено за такими контрольними критеріями:

Відповідність дизайну: Графічне оформлення повністю відповідає затвердженій «космічній» гамі (синій/жовтий) та передає атмосферу епохи Корольова.

Коректність функціонування: Усі переходи між сторінками працюють, Java-додаток виконує свої функції без критичних помилок.

Валідність коду: HTML та CSS код структуровано згідно зі стандартами, що забезпечує легку підтримку проєкту в майбутньому.

Висновок до розділу 7

Отже, було проведено повний цикл тестування розробленого програмного продукту. Сайт та додаток успішно пройшли перевірку за всіма критеріями: вони працюють стабільно, швидко та коректно відображають інформацію. Повний лістинг програмного коду (HTML, CSS, Java) наведено в Додатку В.

8 РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ

Під час роботи над курсовою роботою було розроблено програмний застосунок, який виконує алгоритм неітеративної сферично-кутової інтерполяції векторів. Застосунок було розроблено на мові програмування Java, з використанням бібліотеки для розробки користувацького інтерфейсу та роботи з тривимірною графікою JavaFX. Програмний код було написано в редакторі коду IntelliJ IDEA.

Неітеративна сферично-кутова інтерполяція — це метод знаходження проміжних значень між двома векторами на поверхні сфери, який дозволяє отримати плавний рух по найкоротшій дузі великого кола. Цей підхід є альтернативою лінійній інтерполяції, яка не враховує кривизну сфери, та класичному методу SLERP.

Процес інтерполяції базується на математичній моделі, що використовує ортогоналізацію векторів. Основна формула розрахунку інтерпольованого вектора N(t) виглядає наступним чином:

Де:

— початковий нормалізований вектор;

— повний кут між початковим та кінцевим векторами;

t — параметр інтерполяції (змінюється від 0 до 1);

— допоміжний ортогональний вектор, що лежить у площині обертання.

Вектор розраховується шляхом вилучення проекції кінцевого вектора на початковий (процес ортогоналізації) та наступної нормалізації:

Приклад роботи алгоритму візуалізовано на рисунку 8.1.

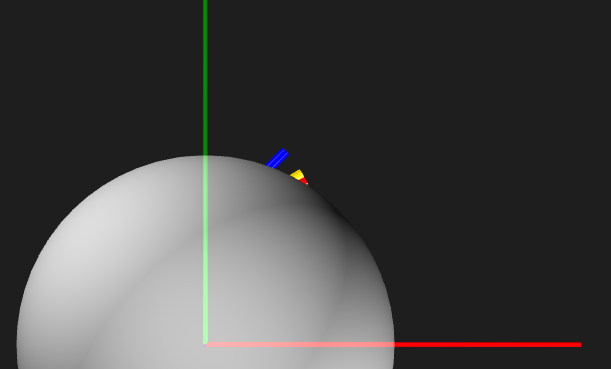


Рисунок 8.1 – Візуалізація сферично-кутової інтерполяції у розробленому застосунку

Вхідні дані програмного застосунку:

− координати початкового вектора ();

− координати кінцевого вектора ;

− параметр часу t (змінюється автоматично в процесі анімації).

Інтерфейс застосунку реалізовано у вигляді 3D-сцени, яка містить наступні елементи:

Сфера — напівпрозора геометрична фігура, що позначає простір інтерполяції.

Осі координат (X, Y, Z) — кольорові циліндри для орієнтації у просторі.

Вектори — початковий (червоний) та кінцевий (синій) вектори, між якими відбувається інтерполяція.

Інтерпольований вектор — золотий вектор, що динамічно змінює своє положення відповідно до алгоритму.

Також реалізовано можливість обертання камери навколо сцени за допомогою миші для детального огляду траєкторії руху. Інтерфейс застосунку показано на рисунку 8.2.



Рисунок 8.2 – Інтерфейс програмного застосунку

Вихідні дані:

− динамічне зображення руху вектора по дузі великого кола сфери.

Малювання виконується за допомогою методу sphericalAngularInterpolation(), реалізованого в класі Vector3D. Основна ідея алгоритму полягає у переході від лінійної комбінації векторів до тригонометричного представлення у новому базисі, утвореному векторами та . Це дозволяє уникнути ітеративних наближень та обчислювати точне положення вектора для будь-якого моменту часу t за одну операцію.

Алгоритм роботи програми включає наступні кроки:

* нормалізація вхідних векторів;
* обчислення кута через скалярний добуток;
* побудова ортогонального вектора ;
* застосування формули обертання для поточного значення t;
* візуалізація отриманого вектора на 3D-сцені.

Алгоритм роботи програми для побудови інтерполяції показано у вигляді блок-схеми на рисунку 8.3.

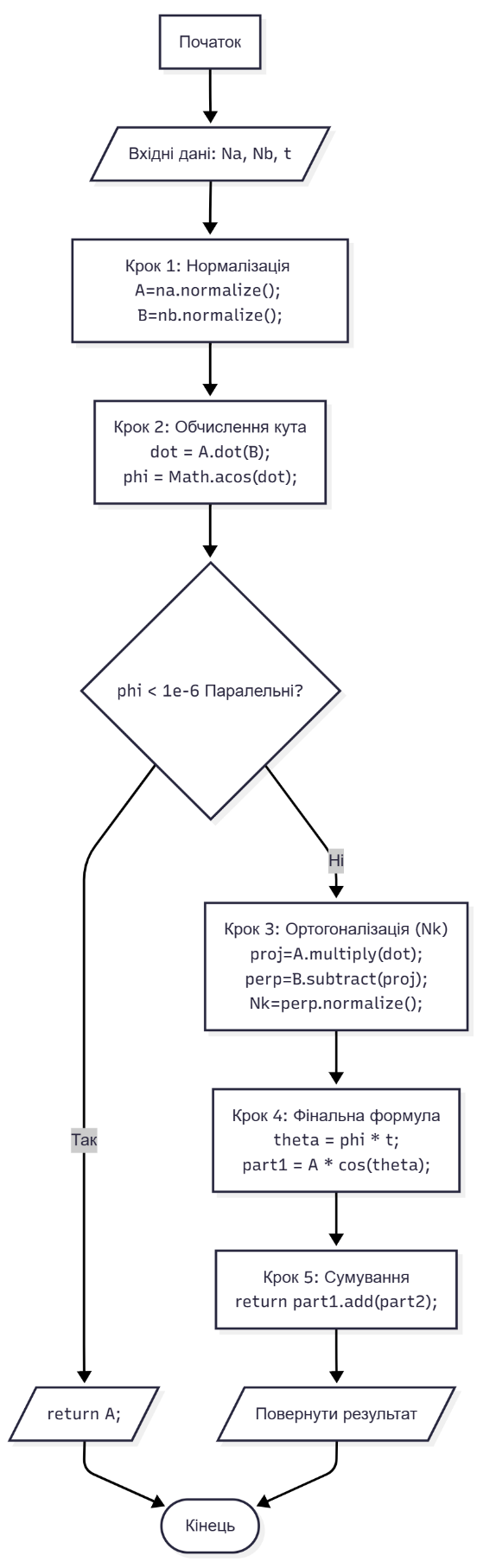


Рисунок 8.3 – Блок-схема алгоритму неітеративної сферично-кутової інтерполяції

Отже, програмний застосунок для демонстрації неітеративної сферично-кутової інтерполяції було реалізовано на мові програмування Java з використанням бібліотеки JavaFX для створення тривимірної графіки. Застосунок дозволяє наочно продемонструвати коректність роботи математичної моделі, забезпечуючи плавний рух вектора по сфері без спотворень швидкості та траєкторії.

В процесі розробки було вивчено та реалізовано аналітичний метод інтерполяції, який, на відміну від лінійних методів, зберігає одиничну довжину вектора на всьому шляху руху. Лістинг коду наведено в додатку Г.

ВИСНОВКИ

У ході виконання курсової роботи було проаналізовано теоретичні основи, класифікацію засобів та видів комп'ютерної графіки. Розглянуто об’єкт, предмет дослідження та досягнуто поставленої мети розробки інформаційного ресурсу.

На основі проведеного аналізу для графічного наповнення сайту було обрано комбінований підхід. Поєднання векторних контурів для чіткості форм та растрових ефектів для реалістичності дозволило досягти високої якості візуальних матеріалів. Було обґрунтовано вибір інструментальних засобів: графічного редактора Photopea для створення основних зображень та сервісу Canva для розробки анімацій.

Особливу увагу приділено колористиці проєкту. За допомогою колірного кола Іттена та інструменту Adobe Color було підібрано комплементарну кольорову гаму, що складається з темно-синіх відтінків (атмосфера космосу) та контрастного жовтого кольору (акценти). Таке рішення забезпечило сучасний вигляд сайту та високу читабельність тексту.

Практичним результатом роботи стала розробка вебсайту, присвяченого життю та діяльності Сергія Павловича Корольова. Для ресурсу створено повний набір статичних зображень (портрети, ракети, супутники, фонові композиції) та динамічну анімацію польоту ракети. Проведено оптимізацію графічного контенту (стиснення, вибір форматів PNG/JPG), що забезпечило швидке завантаження сторінок без втрати якості. Верстка сайту виконана з використанням технологій HTML5 та CSS3, проведено тестування на кросбраузерність та адаптивність.

У рамках програмної частини курсової роботи розроблено застосунок мовою програмування Java з використанням бібліотеки JavaFX. Програма реалізує та візуалізує алгоритм неітеративної сферично-кутової інтерполяції векторів. Створене програмне забезпечення дозволяє наочно демонструвати плавний рух об'єкта по дузі сфери, підтверджуючи коректність використаної математичної моделі. Для застосунку розроблено інтуїтивно зрозумілий графічний інтерфейс та блок-схему алгоритму.

Отже, поставлені задачі виконано в повному обсязі: створено цілісний, естетично привабливий та функціональний вебсайт про історію космонавтики, а також працездатний програмний модуль для роботи з тривимірною векторною графікою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТОК А. СТАТИЧНІ ЗОБРАЖЕННЯ



Рисунок 1 – Планета Земля



Рисунок 2 – Колаж комети, Землі та космосу

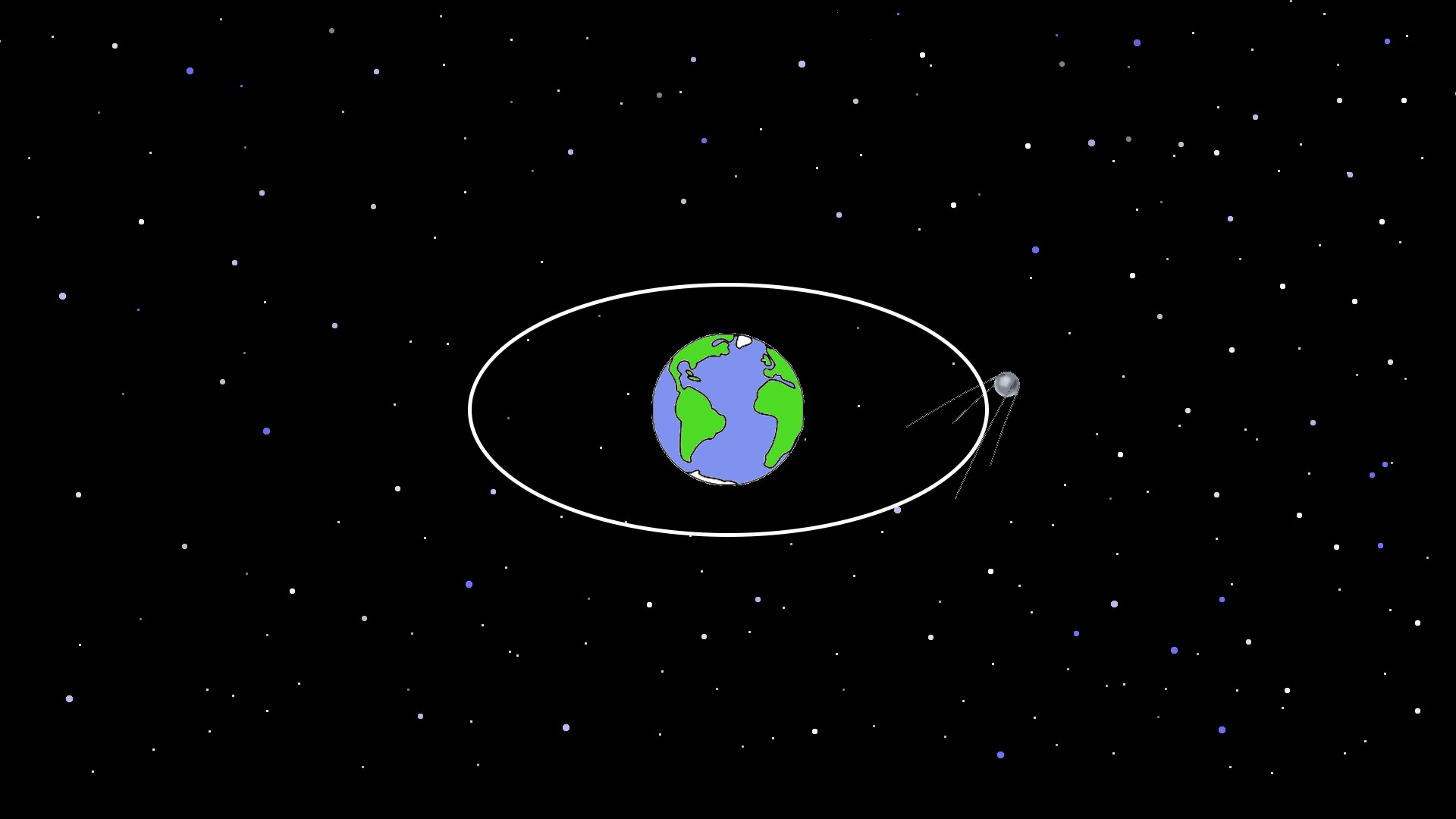


Рисунок 3 – Планета Земля, орбіта супутника, супутника та космос



Рисунок 4 – Планета Земля та космос

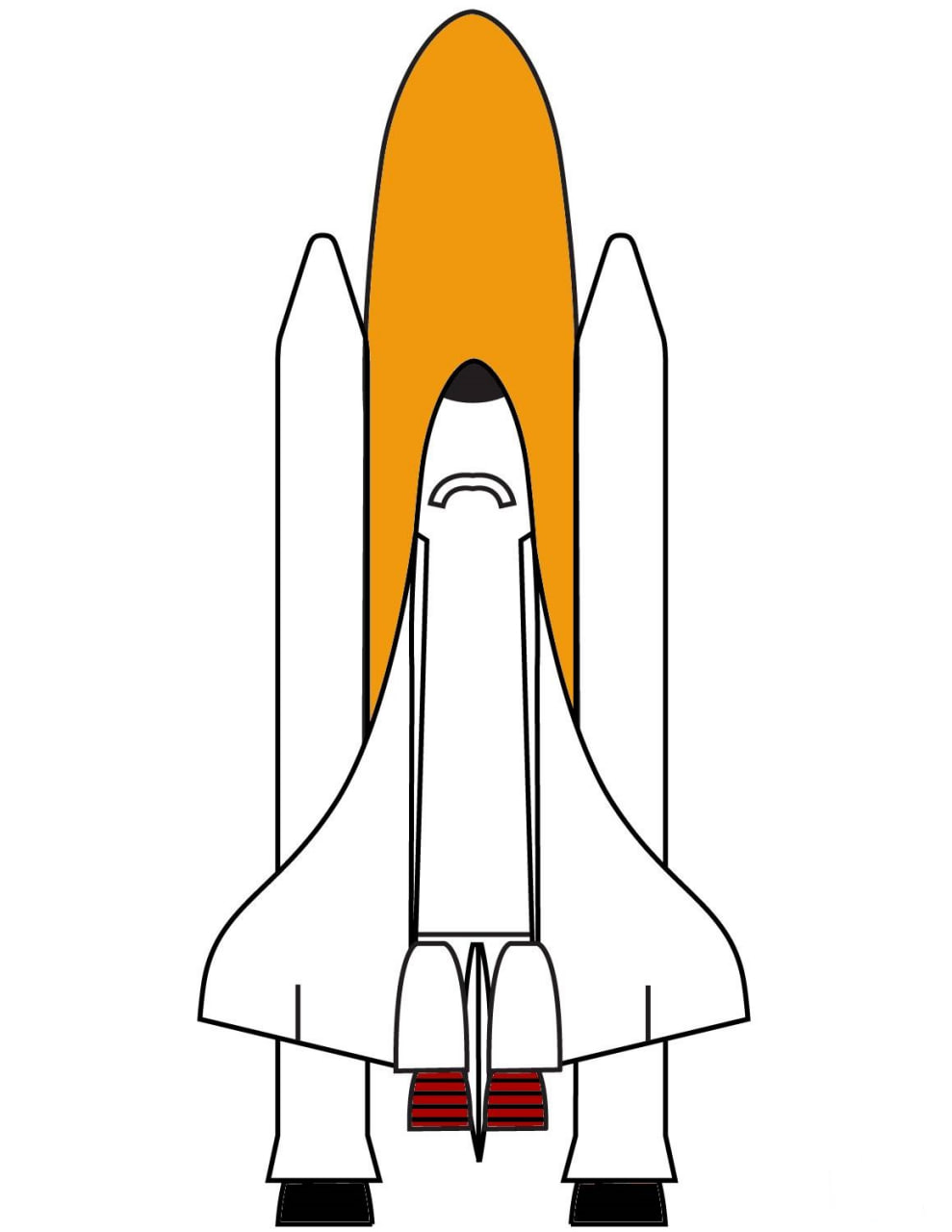


Рисунок 5 – Сучасний космічний шатл

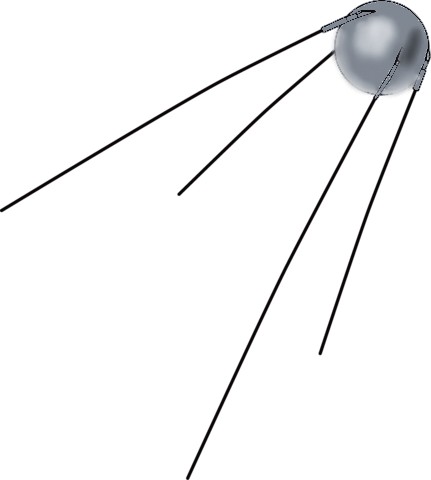


Рисунок 6 – Супутник «Супутник-1»

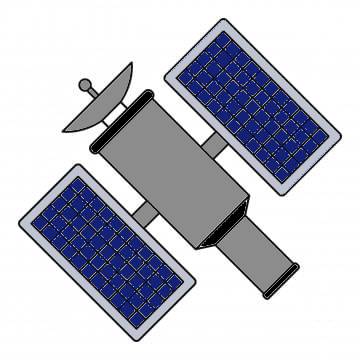


Рисунок 7 – Сучасний супутник

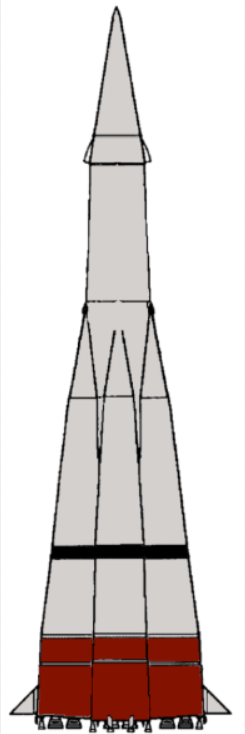


Рисунок 8 – Ракета «Р-7»



Рисунок 9 – Комета



Рисунок 10 – Логотип сайту



Рисунок 11 – Портрет Корольова

ДОДАТОК Б. ДИНАМІЧНЕ ЗОБРАЖЕННЯ



Рисунок 1 – Перший кадр

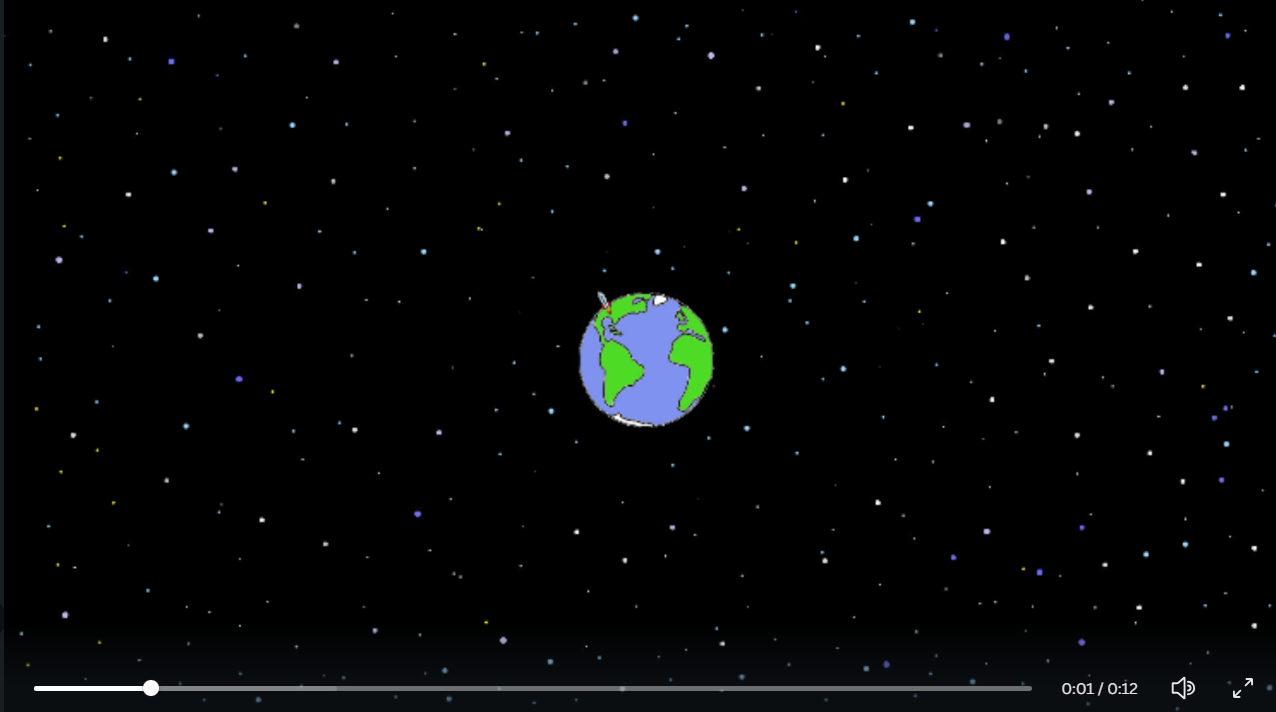


Рисунок 2 – Другий кадр



Рисунок 3 – Третій кадр



Рисунок 4 – Четвертий кадр

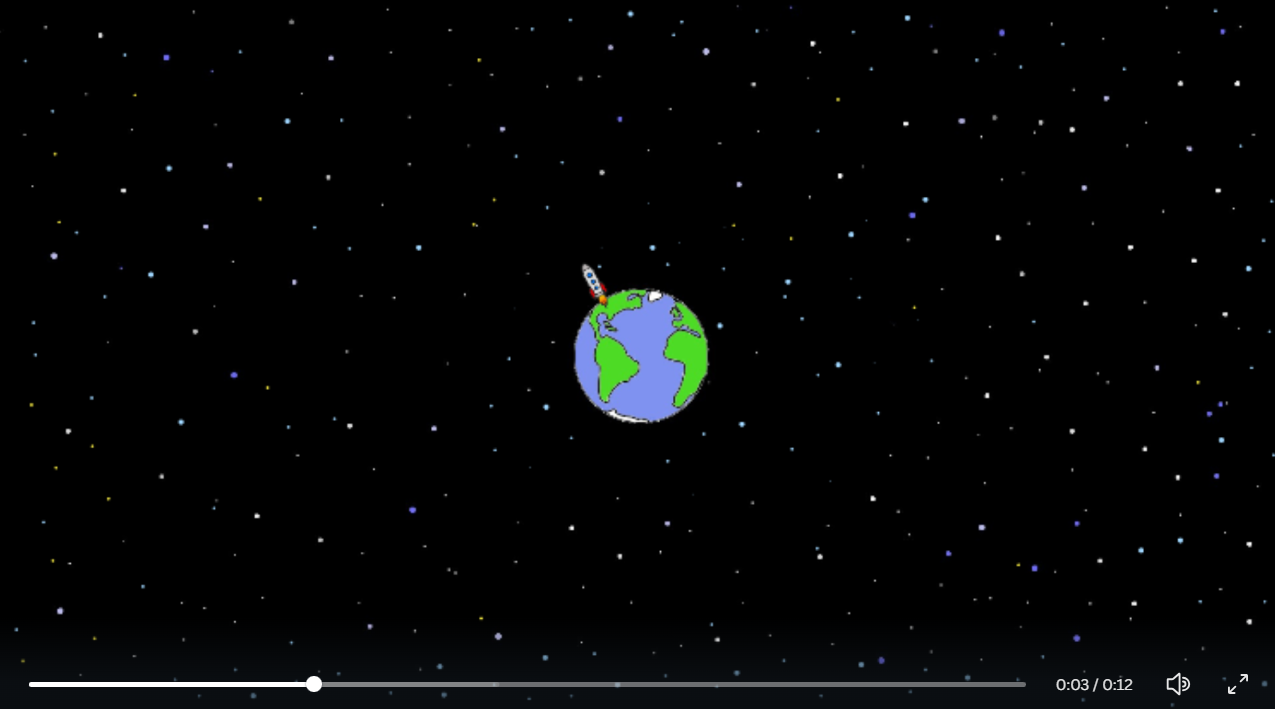


Рисунок 5 – П’ятий кадр

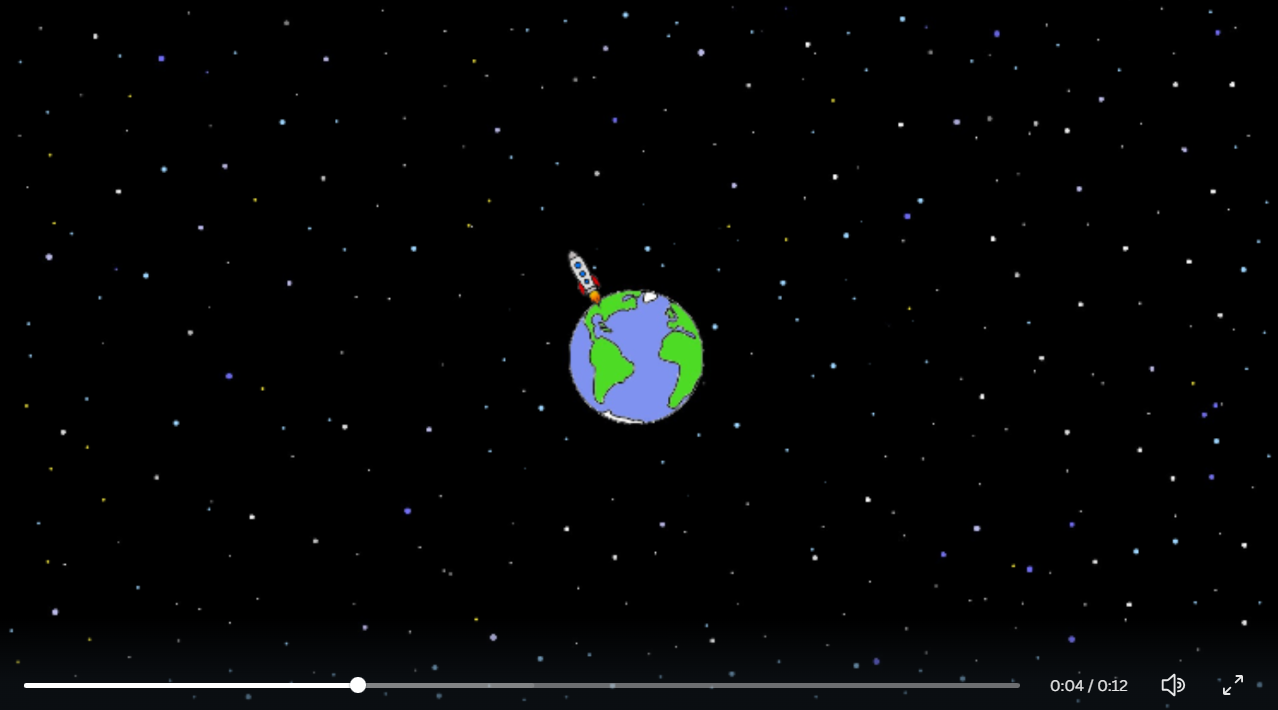


Рисунок 6 – Шостий кадр



Рисунок 7 – Сьомий кадр

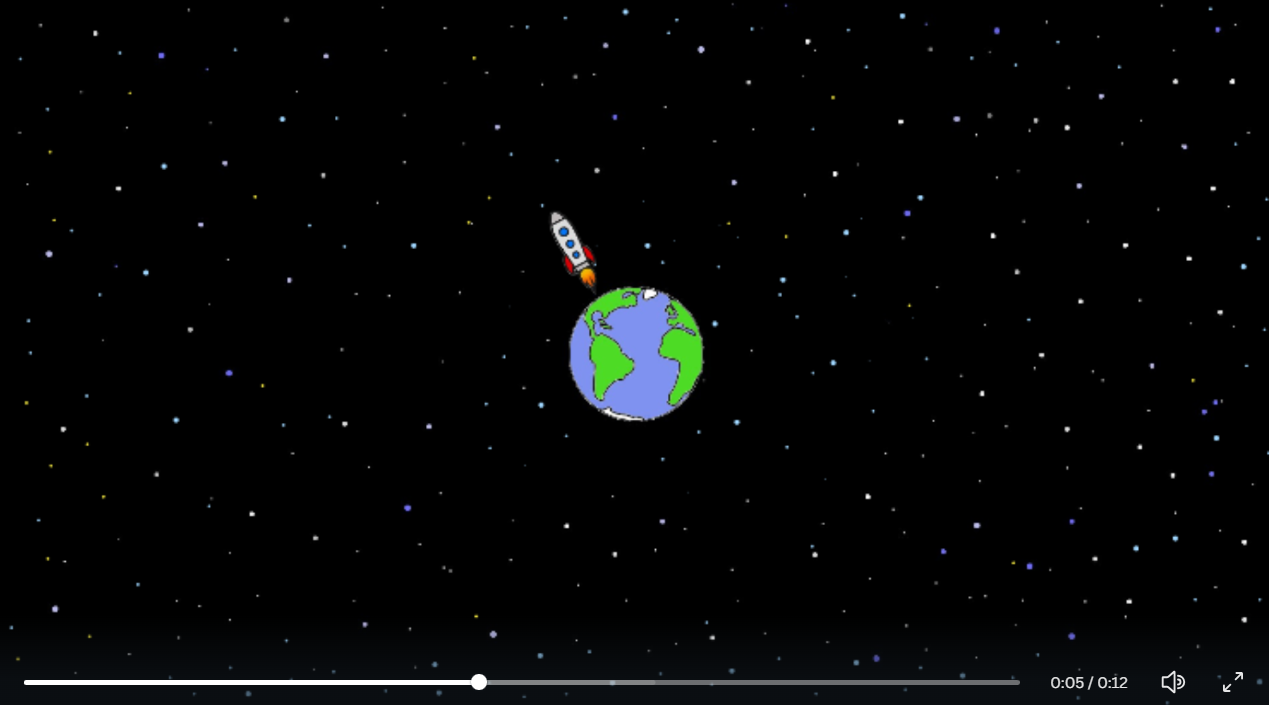


Рисунок 8 – Восьмий кадр



Рисунок 9 – Дев’ятий кадр



Рисунок 10 – Десятий кадр



Рисунок 11 – Одинадцятий кадр

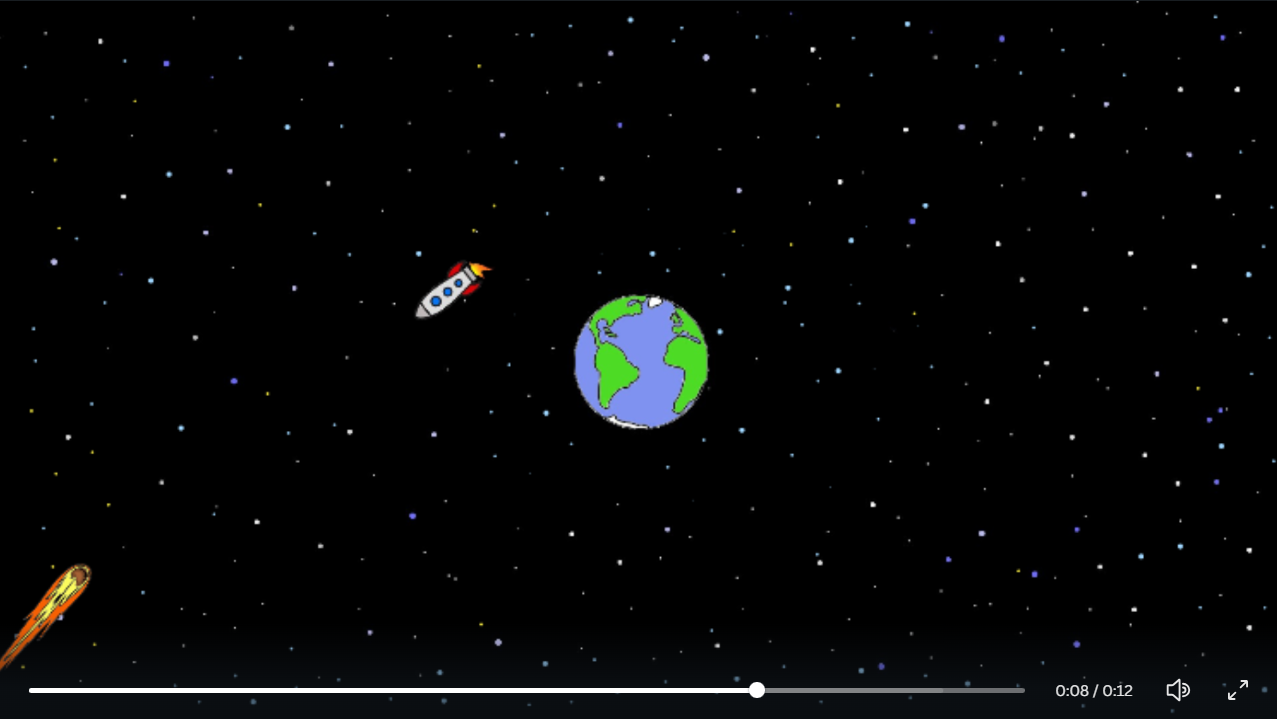


Рисунок 12 – Дванадцятий кадр

Рисунок 13 – Тринадцятий кадр

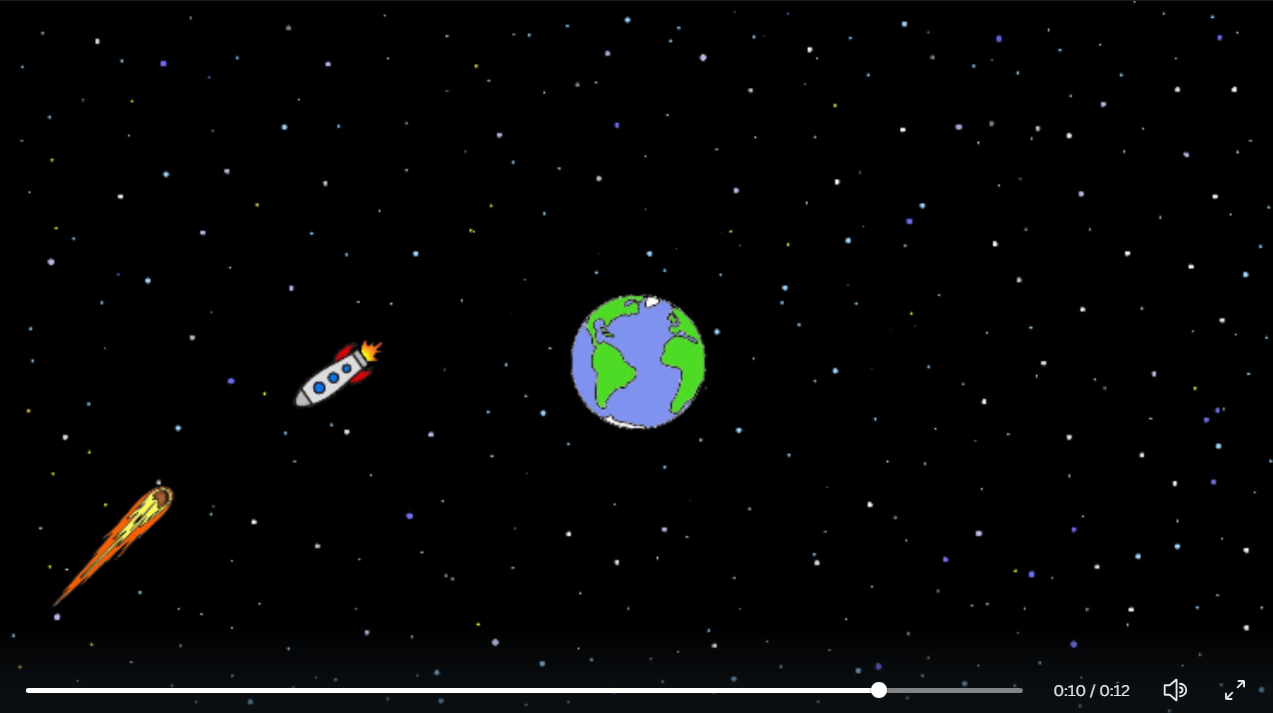


Рисунок 14 – Чотирнадцятий кадр

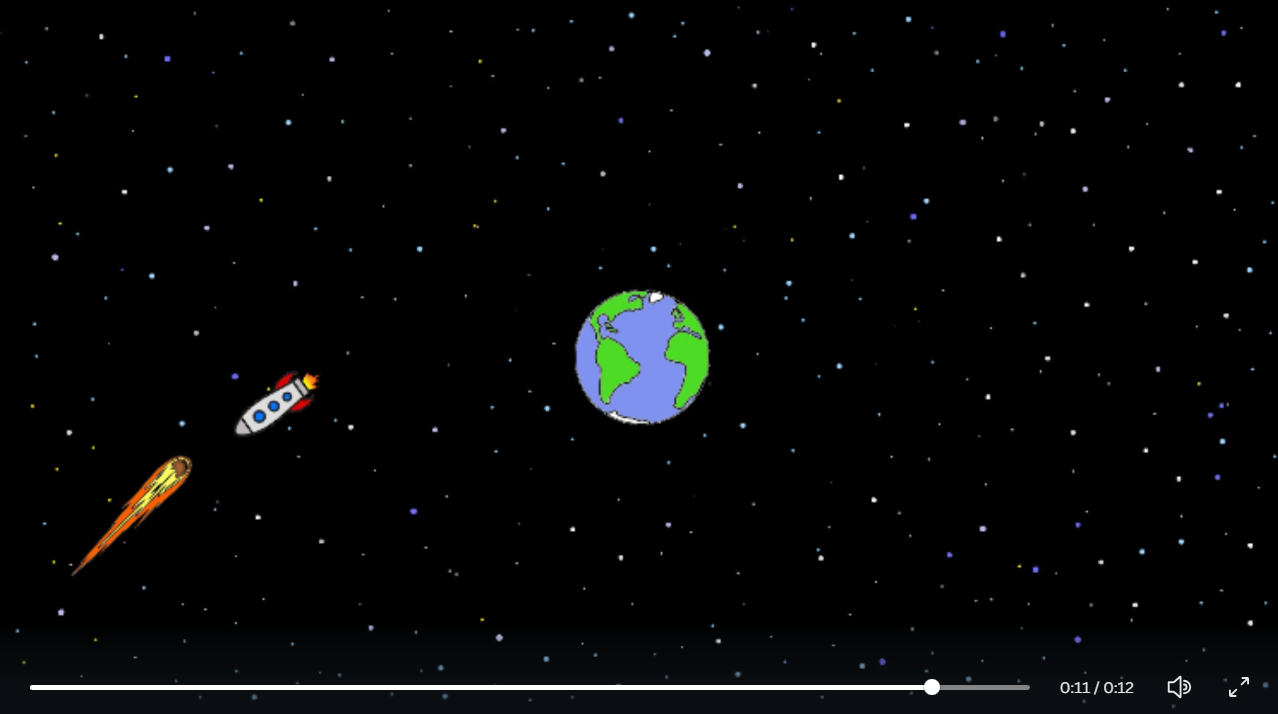


Рисунок 15 – П’ятнадцятий кадр



Рисунок 16 – Шістнадцятий кадр

ДОДАТОК В. ЛІСТИНГ ВЕБ СТОРІНКИ INDEX.HTML

<!DOCTYPE html>

<html lang="uk">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Сергій Корольов</title>

    <link rel="icon" type="image/png" href="../images/logo.png" sizes="32x32">

    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">

    <style>

        /\* ======= Загальні стилі ======= \*/

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }

        body {

            font-family: 'Roboto', sans-serif;

            background: linear-gradient(to bottom, #0b0c1f, #1b1c2f);

            color: #f0f0f0;

            line-height: 1.6;

        }

        a {

            text-decoration: none;

            color: inherit;

        }

        /\* ======= Хедер ======= \*/

        header {

            background-color: rgba(20, 20, 40, 0.9);

            padding: 20px 40px;

            display: flex;

            justify-content: space-between;

            align-items: center;

            position: sticky;

            top: 0;

            z-index: 100;

        }

        header h1 {

            font-size: 1.8rem;

            font-weight: 700;

            letter-spacing: 1px;

        }

        nav a {

            margin-left: 20px;

            font-weight: 500;

            transition: color 0.3s;

        }

        nav a:hover {

            color: #ffbb00;

        }

        /\* ======= Головний блок ======= \*/

        .hero {

            display: flex;

            flex-direction: column;

            align-items: center;

            justify-content: center;

            height: 90vh;

            text-align: center;

            background: url('../images/Korolov.jpeg') no-repeat center center/contain;

            position: relative;

        }

        .hero::after {

            content: "";

            position: absolute;

            top: 0;

            left: 0;

            width: 100%;

            height: 100%;

            background-color: rgba(0, 0, 0, 0.5);

        }

        .hero-content {

            position: relative;

            z-index: 1;

            color: #fff;

        }

        .hero h2 {

            font-size: 3rem;

            margin-bottom: 20px;

        }

        .hero p {

            font-size: 1.2rem;

            max-width: 700px;

            margin: 0 auto 30px auto;

        }

        .hero .btn {

            padding: 12px 30px;

            background-color: #ffbb00;

            color: #1b1c2f;

            font-weight: 700;

            border-radius: 25px;

            transition: transform 0.3s;

        }

        .hero .btn:hover {

            transform: scale(1.05);

        }

        /\* ======= Основний контент ======= \*/

        main {

            padding: 60px 20px;

            max-width: 1200px;

            margin: 0 auto;

        }

        section {

            margin-bottom: 60px;

        }

        section h3 {

            font-size: 2rem;

            margin-bottom: 20px;

            border-bottom: 2px solid #ffbb00;

            display: inline-block;

            padding-bottom: 5px;

        }

        section p {

            font-size: 1.1rem;

            margin-bottom: 15px;

        }

        /\* ======= Футер ======= \*/

        footer {

            background-color: #101021;

            padding: 20px 0;

            text-align: center;

            font-size: 0.9rem;

            color: #aaa;

        }

        @media(max-width:768px) {

            .hero h2 {

                font-size: 2.2rem;

            }

            .hero p {

                font-size: 1rem;

            }

        }

    </style>

</head>

<body>

    <header>

        <img src="../images/logo.png" alt="Logo" style="width: 50px;">

        <h1>Сергій Корольов</h1>

        <nav>

            <a href="index.html">Головна</a>

            <a href="biography.html">Біографія</a>

            <a href="achievements.html">Досягнення</a>

            <a href="r7.html">Ракета Р-7</a>

            <a href="sputnik.html">Супутник-1</a>

        </nav>

    </header>

    <section class="hero">

        <div class="hero-content">

            <h2>Батько космічної ери</h2>

            <p>Сергій Павлович Корольов – видатний радянський конструктор, головний розробник ракетно-космічної техніки,

                творець першої ракети Р-7 і запуску першого штучного супутника Землі.</p>

            <a href="biography.html" class="btn">Дізнатись більше</a>

        </div>

    </section>

    <main>

        <section id="biography">

            <h3>Біографія</h3>

            <p>Сергій Корольов народився 1907 року в Житомирі. Він отримав освіту інженера і присвятив життя розвитку

                ракетобудування та космічних досліджень.</p>

        </section>

        <section id="achievements">

            <h3>Досягнення</h3>

            <p>Корольов очолював проєкти з запуску першого супутника Землі, першого космонавта і створення ракети Р-7.

                Його робота заклала основу сучасної космонавтики.</p>

        </section>

        <section id="r7">

            <h3>Ракета Р-7</h3>

            <p>Р-7 – перша міжконтинентальна балістична ракета, яка також слугувала основою для запуску супутників та

                космічних кораблів.</p>

        </section>

        <section id="sputnik1">

            <h3>Супутник-1</h3>

            <p>Перший штучний супутник Землі, запущений 4 жовтня 1957 року, відкрив еру космічних досліджень і став

                символом технологічного прориву.</p>

        </section>

    </main>

    <footer>

        &copy; 2025 Сергій Корольов. Усі права захищені.

    </footer>

</body>

</html>

ДОДАТОК Г. ЛІСТИНГ ПРОГРАМНОГО ЗАСТОСУНКУ

HelloApplication.java

package org.example.demo;  
  
import javafx.animation.Timeline;  
import javafx.animation.KeyFrame;  
import javafx.application.Application;  
import javafx.scene.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.paint.PhongMaterial;  
import javafx.scene.shape.\*;  
import javafx.scene.transform.Rotate;  
import javafx.scene.transform.Translate;  
import javafx.stage.Stage;  
import javafx.util.Duration;  
  
public class HelloApplication extends Application {  
  
 // Вектори для завдання  
 private final Vector3D START\_VECTOR = new Vector3D(1.0, 0.5, 0.0).normalize();  
 private final Vector3D END\_VECTOR = new Vector3D(0.0, 1.0, 1.0).normalize();  
  
 private static final double RADIUS = 150;  
 private static final double VECTOR\_LENGTH = RADIUS \* 1.2;  
  
 // Золотий вектор, що буде рухатись  
 private final Cylinder interpolatedVector = createVectorShape(Color.GOLD);  
  
 // Група для 3D світу  
 private final Group world = new Group();  
  
 // Змінні для керування мишею  
 private double anchorX, anchorY;  
 private double anchorAngleX = 0;  
 private double anchorAngleY = 0;  
 private final Rotate rotateX = new Rotate(0, Rotate.X\_AXIS);  
 private final Rotate rotateY = new Rotate(0, Rotate.Y\_AXIS);  
  
 @Override  
 public void start(Stage primaryStage) {  
 // 1. Додаємо об'єкти у світ  
 world.getChildren().addAll(createAxes(), createSphere());  
 world.getChildren().add(createVectorShape(START\_VECTOR, Color.RED)); // Початок  
 world.getChildren().add(createVectorShape(END\_VECTOR, Color.BLUE)); // Кінець  
 world.getChildren().add(interpolatedVector); // Рухомий вектор  
  
 // 2. Додаємо світло (щоб бачити об'єм)  
 PointLight light1 = new PointLight(Color.WHITE);  
 light1.setTranslateX(-500); light1.setTranslateY(-500); light1.setTranslateZ(-500);  
  
 PointLight light2 = new PointLight(Color.WHITE);  
 light2.setTranslateX(500); light2.setTranslateY(500); light2.setTranslateZ(-500);  
  
 world.getChildren().addAll(light1, light2);  
  
 // 3. Центруємо світ посередині вікна (400, 300)  
 Group root = new Group(world);  
 world.getTransforms().addAll(  
 new Translate(400, 300, 0),  
 rotateX,  
 rotateY  
 );  
  
 // 4. Камера  
 Camera camera = new PerspectiveCamera(false);  
  
 // 5. Сцена з темним фоном  
 Scene scene = new Scene(root, 800, 600, true, SceneAntialiasing.BALANCED);  
 scene.setFill(Color.rgb(30, 30, 30));  
 scene.setCamera(camera);  
  
 // Ініціалізація керування та анімації  
 initMouseControl(scene);  
 setupAnimation();  
  
 primaryStage.setTitle("Non-Iterative Spherical-Angular Interpolation (Formula 12)");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 // --- Математика візуалізації ---  
  
 private void setupAnimation() {  
 Timeline timeline = new Timeline();  
  
 // Рух від 0 до 1  
 for (double t = 0.0; t <= 1.0; t += 0.01) {  
 double finalT = t;  
 timeline.getKeyFrames().add(new KeyFrame(Duration.seconds(2.0 \* t), e -> {  
 // !!! ТУТ ГОЛОВНА ЗМІНА: ВИКЛИК НОВОГО МЕТОДУ ЗГІДНО З ЗАВДАННЯМ !!!  
 Vector3D v = Vector3D.sphericalAngularInterpolation(START\_VECTOR, END\_VECTOR, finalT);  
 setVectorRotation(interpolatedVector, v);  
 }));  
 }  
  
 // Рух назад від 1 до 0  
 for (double t = 1.0; t >= 0.0; t -= 0.01) {  
 double finalT = t;  
 timeline.getKeyFrames().add(new KeyFrame(Duration.seconds(4.0 - 2.0 \* t), e -> {  
 Vector3D v = Vector3D.sphericalAngularInterpolation(START\_VECTOR, END\_VECTOR, finalT);  
 setVectorRotation(interpolatedVector, v);  
 }));  
 }  
  
 timeline.setCycleCount(Timeline.INDEFINITE);  
 timeline.play();  
 }  
  
 // --- Допоміжні методи побудови сцени ---  
  
 private void initMouseControl(Scene scene) {  
 scene.setOnMousePressed(event -> {  
 anchorX = event.getSceneX();  
 anchorY = event.getSceneY();  
 anchorAngleX = rotateX.getAngle();  
 anchorAngleY = rotateY.getAngle();  
 });  
 scene.setOnMouseDragged(event -> {  
 rotateX.setAngle(anchorAngleX - (anchorY - event.getSceneY()));  
 rotateY.setAngle(anchorAngleY + (anchorX - event.getSceneX()));  
 });  
 }  
  
 private Node createSphere() {  
 Sphere sphere = new Sphere(RADIUS);  
 PhongMaterial material = new PhongMaterial(Color.WHITE);  
 material.setDiffuseColor(Color.rgb(200, 200, 200, 0.3)); // Напівпрозора  
 sphere.setMaterial(material);  
 return sphere;  
 }  
  
 private Cylinder createVectorShape(Color color) {  
 // Циліндр: радіус 4, довжина VECTOR\_LENGTH  
 Cylinder cylinder = new Cylinder(4, VECTOR\_LENGTH);  
 PhongMaterial mat = new PhongMaterial(color);  
 mat.setSpecularColor(Color.WHITE);  
 cylinder.setMaterial(mat);  
 // Зсуваємо, щоб початок був у (0,0,0)  
 cylinder.setTranslateY(-VECTOR\_LENGTH / 2.0);  
 return cylinder;  
 }  
  
 private Node createVectorShape(Vector3D v, Color color) {  
 Cylinder cylinder = createVectorShape(color);  
 setVectorRotation(cylinder, v);  
 return cylinder;  
 }  
  
 private void setVectorRotation(Cylinder cylinder, Vector3D v) {  
 Vector3D initial = new Vector3D(0, 1, 0);  
 double dot = initial.dot(v);  
 // clamp values  
 if (dot < -1) dot = -1;  
 if (dot > 1) dot = 1;  
  
 double angle = Math.acos(dot);  
  
 Vector3D axis = new Vector3D(  
 initial.y \* v.z - initial.z \* v.y,  
 initial.z \* v.x - initial.x \* v.z,  
 initial.x \* v.y - initial.y \* v.x  
 );  
  
 if (axis.magnitude() < 1e-6) {  
 axis = new Vector3D(1, 0, 0);  
 } else {  
 axis = axis.normalize();  
 }  
  
 angle = Math.toDegrees(angle);  
  
 cylinder.getTransforms().clear();  
 cylinder.getTransforms().add(new Rotate(angle, axis.x, axis.y, axis.z));  
 }  
  
 private Node createAxes() {  
 Group axes = new Group();  
 final double LEN = 300;  
  
 Cylinder x = new Cylinder(2, LEN);  
 x.setMaterial(new PhongMaterial(Color.RED));  
 x.setRotationAxis(Rotate.Z\_AXIS); x.setRotate(90); x.setTranslateX(LEN / 2);  
  
 Cylinder y = new Cylinder(2, LEN);  
 y.setMaterial(new PhongMaterial(Color.GREEN));  
 y.setTranslateY(-LEN / 2);  
  
 Cylinder z = new Cylinder(2, LEN);  
 z.setMaterial(new PhongMaterial(Color.BLUE));  
 z.setRotationAxis(Rotate.X\_AXIS); z.setRotate(90); z.setTranslateZ(LEN / 2);  
  
 axes.getChildren().addAll(x, y, z);  
 return axes;  
 }  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
}  
  
  
Vector3D.java

package org.example.demo;  
  
public class Vector3D {  
 public final double x, y, z;  
  
 public Vector3D(double x, double y, double z) {  
 this.x = x;  
 this.y = y;  
 this.z = z;  
 }  
  
 // --- Базові операції ---  
  
 public Vector3D normalize() {  
 double magnitude = Math.sqrt(x \* x + y \* y + z \* z);  
 if (magnitude < 1e-9) return new Vector3D(0, 0, 0);  
 return new Vector3D(x / magnitude, y / magnitude, z / magnitude);  
 }  
  
 public double dot(Vector3D other) {  
 return this.x \* other.x + this.y \* other.y + this.z \* other.z;  
 }  
  
 public Vector3D add(Vector3D other) {  
 return new Vector3D(this.x + other.x, this.y + other.y, this.z + other.z);  
 }  
  
 public Vector3D subtract(Vector3D other) {  
 return new Vector3D(this.x - other.x, this.y - other.y, this.z - other.z);  
 }  
  
 public Vector3D multiply(double scalar) {  
 return new Vector3D(this.x \* scalar, this.y \* scalar, this.z \* scalar);  
 }  
  
 /\*\*  
 \* Реалізація методу з Вашої статті (Формула 12).  
 \* Неітеративна сферично-кутова інтерполяція.  
 \* N(t) = Na \* cos(phi \* t) + Nk \* sin(phi \* t)  
 \*/  
 public static Vector3D sphericalAngularInterpolation(Vector3D na, Vector3D nb, double t) {  
 // 1. Нормалізація вхідних векторів (Na, Nb)  
 Vector3D A = na.normalize();  
 Vector3D B = nb.normalize();  
  
 // 2. Розрахунок косинуса кута між ними  
 double dot = A.dot(B);  
 // Захист від похибок (щоб не вийти за межі -1..1)  
 if (dot < -1.0) dot = -1.0;  
 if (dot > 1.0) dot = 1.0;  
  
 // Кут phi (повний кут між векторами)  
 double phi = Math.acos(dot);  
  
 // Якщо вектори майже збігаються, повертаємо A  
 if (Math.abs(phi) < 1e-6) {  
 return A;  
 }  
  
 // 3. Знаходження вектора Nk (ортогональний вектор у площині обертання)  
 // Nk = (Nb - Na \* cos(phi)) / ||...||  
 // Це процес ортогоналізації (Грама-Шмідта)  
 Vector3D projection = A.multiply(dot); // Проекція B на A  
 Vector3D perpendicular = B.subtract(projection); // Перпендикулярна складова  
 Vector3D Nk = perpendicular.normalize(); // Нормуємо, щоб отримати Nk  
  
 // 4. Застосування формули (12)  
 // t змінюється від 0 до 1, тому поточний кут theta = phi \* t  
 double theta = phi \* t;  
  
 Vector3D part1 = A.multiply(Math.cos(theta));  
 Vector3D part2 = Nk.multiply(Math.sin(theta));  
  
 return part1.add(part2);  
 }  
 // Обчислення довжини вектора (Саме цього методу не вистачало)  
 public double magnitude() {  
 return Math.sqrt(x \* x + y \* y + z \* z);  
 }  
}