Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Кафтан Микита Олегович*
(ПІБ)

академічної групи *122-18-2*
(шифр)

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

освітньої програми *Комп'ютерні науки*
(назва освітньої програми)

на тему: *Розробка додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Спірінцев В.В.*			
розділів:				
спеціальний	*доц. Спірінцев В.В.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент				
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-18-2* *Кафтан М.О.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch*

затверджена наказом ректора НТУ «ДП» від №

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ *доц.Спірінцев В.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Кафтан М.О.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 94 с., 41 рис., 2 табл., 3 дод., 20 джерел.

Об'єкт розробки: навчальний комп’ютерний застосунок.

Мета кваліфікаційної роботи: розробка додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch.

У вступі розглядається аналіз та сучасний стан проблеми, конкретизується мета кваліфікаційної роботи та галузь її застосування, наведено обґрунтування актуальності теми та уточнюється постановка завдання.

У першому розділі проведено аналіз предметної галузі, визначено актуальність завдання та призначення розробки, розроблена постановка завдання, задані вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі виконано аналіз існуючих рішень, обрано вибір платформи для розробки, виконано проектування і розробка програми, наведено опис алгоритму і структури функціонування програми, визначені вхідні і вихідні дані, наведені характеристики складу параметрів технічних засобів, описаний виклик та завантаження програми, описана робота програми.

В економічному розділі визначено трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Практичне значення роботи полягає в створенні навчальної програми для дітей з метою вивчення й навчання роботи з графічним редактором Adobe Illustrator.

Актуальність теми кваліфікаційної роботи визначається великим попитом на подібні розробки, через те, що сьогодні у навчальних закладах з метою активізації пізнавальної діяльності учня потрібно використовувати усі можливі мультимедійні технології для проведення уроків, практичних і самостійних занять.

Список ключових слів: ДОДАТОК, ОПЕРАЦІЙНА СИСТЕМА, ГРАФІЧНИЙ РЕДАКТОР, ВЕКТОРНА ГРАФІКА, ПРОГРАМА, АЛГОРИТМ, НАЛАГОДЖЕННЯ, ТЕСТУВАННЯ, ПРОЄКТУВАННЯ.

ABSTRACT

Explanatory note: 94 pp., 41fig., 2 table, 3 appendix, 20 sources.

Object of development: educational computer application.

Purpose of the qualification work: Development of an application for teaching children to work with the graphic editor Adobe Illustrator based on Scratch.

The introduction considers the analysis and current state of the problem, specifies the purpose of the qualification work and the field of its application, provides a justification for the relevance of the topic and clarifies the problem.

In the first section the analysis of the subject area is carried out, the urgency of the task and the purpose of development are determined, the task statement is developed, the requirements to the software implementation, technologies and software are set.

The second section analyzes existing solutions, selects a platform for development, designs and develops the program, describes the algorithm and structure of the program, determines the input and output data, provides characteristics of the parameters of technical means, describes the call and download program, describes the work programs.

The economic section determines the complexity of the developed information system, calculates the cost of work to create a program and calculates the time for its creation.

The practical significance of the work is to create a curriculum for children to learn and learn to work with the graphic editor Adobe Illustrator.

The relevance of the topic of qualification work is determined by the great demand for such developments, due to the fact that today in educational institutions in order to enhance the cognitive activity of students need to use all possible multimedia technologies for lessons, practical and independent classes.

List of keywords: APP, OPERATING SYSTEM, GRAPHIC EDITOR, VECTOR GRAPHICS, PROGRAM, ALGORITHM, ADJUSTMENT, TESTING, TESTING, DESIGN.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ - програмне забезпечення;

ОС - операційна система;

ПК - персональний комп'ютер;

ТЗ - технічне завдання;

ООП - об’єктно орієнтоване програмування;

HDD - hard disk drive;

ОЗП - оперативний запам'ятовуючий пристрій;

SSD - solid-state drive.

ВСТУП

Навчання є основною складовою процвітання людства. В наш час є безліч шкіл, гуртків, навчальних відеоматеріалів. Також, за останні 10 років, інформаційні технології наклали свій великий відбиток. У початкових та середніх класах, більше й частіше починають знайомити з комп'ютером. Ми бачимо, що діти навчаються працювати з гаджетами раніше, ніж правильно писати і грамотно говорити. Цю особливість потрібно опановувати, задля їх найшвидшого розвитку і надання правильного, точного напрямку в галузі інформаційних технологій. На додачу до цього, все навчання конкретизовано на індивідуалізації та оптимізації, як шкільної, так і інших освіт.

Задля активізації та зацікавленості дітей, в начальних закладах використовуються мультимедійні технології. Тож було вирішено розробити застосунок для навчання дітей графічному редактору. Така програма повинна бути простою та цікавою для юних дизайнерів. За допомогою таких застосунків, дитина зможе швидко зорієнтуватися в виборі програми для навчання, розвитку і подальшої, можливо, роботи.

Актуальність цієї роботи полягає в тому, що кожна дитина, може спробувати себе в новому напрямку розвитку. Так як аналогів на ринку майже немає, передбачається велика зацікавленість в цьому продукті. Для простоти в використанні, в ньому не повинно бути забагато зайвих деталей та функцій. В застосунку реалізована ідея інтерактивного уроку, що дозволяє дитині почувати себе більш впевнено, ніж як на першому уроці в групі.

Цей застосунок може бути використаним у різних навчальних закладах для наочного показу навчальної програми, профорієнтування та зацікавленні нових учнів до нової сфери діяльності.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Загальні відомості з предметної галузі

Навчання дітей є одною з найважливіших умов процвітання людства. Всі розуміють, якщо навчатися чомусь змалечку, то подальший розвиток в обраній сфері буде набагато простішим. На даний момент часу, абсолютно всі діти навчаються програмувати на «Scratch». Тож застосунок, який буде зроблений на його базі не буде вводити дитину у ступор при вигляді нової, незрозумілої програми. Також, дітям дуже важко сприймати відеоматеріали та відеоуроки, в першу чергу через недостачу усидливості та уваги.

При пошуку аналогів, було виявлено, що їх немає, тож цей застосунок унікальний, та повинен зайняти нову нішу в навчанні дітей.

Тож було вирішено розробити застосунок, який допоможе зробити перші кроки в сфері графічного дизайну. При виборі графічного редактору, вибір впав на Adobe Illustrator. Є кілька основних причин, чому вибір саме векторного графічного редактору є максимально правильним:

- якісне масштабування у будь-який бік;
- збільшення чи зменшення об'єктів проводиться збільшенням чи зменшенням відповідних коефіцієнтів у математичних формулах;
- векторна графіка залежить від розширення, тобто, може бути показана у різноманітних вихідних пристроях з різною роздільною здатністю без втрати якості;
- за допомогою векторної графіки можна вирішити багато художньо-графічних завдань;
- векторна графіка також може вирішити багато художньо-графічних завдань;
- цей вид комп'ютерної графіки дозволяє проводити будь-які геометричні побудови;

- редагування контуру може бути застосоване для роботи над лінійним малюнком, дизайном виробів зі скла, кераміки та інших пластичних матеріалів.

Але найголовніше – це те що графічний дизайн може вирішує безліч різних завдань за допомогою кольорів, форм, зображень, композицій та типографії. Вирішувати будь-які завдання одним способом або інструментом неможливо, тому є кілька видів графічного дизайну. Зазвичай дизайнери спеціалізуються на одному виді, але сьогодні потрібно бути гнучким та вникати у всі галузі сфери.

Як ми бачимо, реклама стимулює більшість людей купляти продукцію у розпіарених компаній. Але яким чином залучити до свого бізнесу більше клієнтів (без урахування різниці продукції)? На допомогу приходить якісна і стилізована реклама, яка в свою чергу створена завдяки графічним дизайнерам. Вирішувати будь-які завдання одним способом або інструментом неможливо, тому є кілька видів графічного дизайну. Зазвичай дизайнери спеціалізуються на одному виді, але сьогодні потрібно бути гнучким та вникати у всі галузі сфери.

Графічний дизайн - засіб візуальної комунікації. Якщо сказати простіше - це вираз ідей, смислів та цінностей через образи, зображення, шрифти, відео тощо.

Сам по собі графічний дизайн є багатогранною сферою діяльності. Діяльність полягає в проектуванні комунікацій між споживачем та виробником. В більшості випадків дизайнери знають, що потрібно споживачу. Адже формування цілісної картини ідейного образу і реалізація її в житті – дуже важка робота. За достатньо великий час існування графічного дизайну, саме художники і конструктори мають достатній досвід у інтерпретації повідомлень від замовника.

Тож, графічний дизайн став новим шляхом в розвитку бізнесу. І як показує час, художників у світі не так і багато. Тому навчання дизайнерів ще змалку є дуже потрібною справою. Виховання гарного смаку дуже добре

сказується на формуванні юних експертів цієї справи. Добрий смак є, що далі? Потрібні практичні навички малювання і розробки зображень, плакатів, банерів.

Ці практичні навички можна отримати шляхом проходження курсів з графічного дизайну. На цих курсах ви отримаєте навички щодо теорії кольору, композиції, володіння різними растровими та векторними графічними редакторами.

Зараз максимально популярні та потрібні саме векторні ілюстрації, адже якість зображення, при різному масштабуванні не змінюється. Тож мій проект націлений саме на навчання роботі в векторному графічному редакторі Adobe Illustrator із залученням дітей до проходження повного курсу по комп'ютерній графіці в школі програмування.

Adobe Illustrator (Adobe Systems) – векторний графічний редактор, використовуємий в різних цілях: реклама, розкадровки, журнали, плакати. Програма має широкий вибір інструментів для макетування та малювання. Підтримує понад 20 форматів файлів.

1.2. Призначення розробки та галузь застосування

В наш час відкривається все більше шкіл, метою яких є навчання дітей до 18 років. Частіше за все такі школи спеціалізуються на вивченні комп'ютера. Бувають абсолютно різні курси: від комп'ютерної грамотності і до глибокого вивчення одної з мов програмування.

Ці школи потрібні як і для дітей, так і для батьків. Вони завжди знають, що їх дитина не тільки розважається, проводячи час в одній групі з однолітками, а ще й здобуває цінний досвід с однієї зі сфер комп'ютерних наук.

Але є і негативний зворотній бік. Дуже багато часу займають пробні уроки, так звані «Майстер-класи». Тож розробка проекту, що проводить дитині короткий екскурс по програмі навчання, дуже доцільна.

Як об'єкт впровадження розроблюваного додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch розглядається застосунок, який містить зручний інтерфейс для розглядання та засвоєння інформації дітям.

Аналізуючи вимоги та враховуючи специфіку та досвід навчання дітей, було зроблено висновок про те, що обов'язково буде реалізовано в проекті:

- інтуїтивно-зрозумілий інтерфейс, щоб легко було розібратися та швидко перейти до навчання;
- можливість запуску окремих частин програми, для простого перегляду інформації ще раз, для кращого засвоєння.

Головна сторінки, яка б першою поставала перед користувачем немає, це зроблено для того, щоб у дитини не було можливості «втекти» від навчання та без зволікань почати.

Розроблена інформаційна система призначена для:

– об’єднання, безпосередньо, навчання й розваг, завдяки музичному супроводу та жартам, вбудованим в програму;
– економії робочого часу викладачів;
– економії коштів на розробку програми для проведення пробних уроків;
– наочного показу майбутнім студентам результатів проходження курсу;
– проведення екскурсу по навчальній програмі;
– наочний показ майбутнім студентам результатів проходження курсу;
– підвищення ефективності роботи з дітьми, за рахунок того, що вони одразу будуть знати, що і як вони будуть вивчати.

Така програма може використовуватись як викладачами і школами програмування для ознайомлення студентів з курсом, так і батьками для профорієнтації дітей.

1.3. Підстава для розробки

Відповідно до ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

- ОПП за спеціальністю 122 «Комп'ютерні науки»;
- навчальний план та графік навчального процесу;
- наказ ректора Національного технічного університету «Дніпровська політехніка» № _____від____.____.2022 р;
- завдання на дипломний проект на тему «Розробка додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch».

1.4. Постановка завдання

Мета кваліфікаційної роботи «Розробка додатку для навчання дітей роботі з графічним редактором Adobe Illustrator на базі Scratch» – аналіз технологій та методів для розробки програм на базі візуального подійно-орієнтованого середовища програмування Scratch, аналіз та детальне вивчення всіх функцій і можливостей векторного графічного редактору Adobe Illustrator, та розробка застосунку для ознайомлення та навчання дітей з ним.

Для досягнення потрібного результату, потрібно виконати наступні пункти:

- ознайомитись с методичними вказівками та навчальною програмою;
- ознайомитись с літературою по методикам і методам навчання дітей;
- детально розібратися з роботою використовуваного програмного забезпечення;
- виставити правильні й чіткі потреби до застосунку;
- створити програмну реалізацію проекту.

По виконанню роботи, застосунок повинен вирішувати такі задачі:

- утримання інтересу у дитини;
- допомога в інсталяції програмного забезпечення;
- ознайомлення дитини с програмою;
- надання перших вказівок по роботі з застосунком;
- схиляння дитини до подальшого навчання.

Процес користування розроблюваним застосунком повинен бути легким і зрозумілим. У застосунку повинні бути розважальні моменти. Тривалість «чистої» роботи повинна бути не більше 15 хвилин, для того, щоб дитина не засиджувалася за комп'ютером і щоб процес першого уроку не набрид. В процесі роботи програми потрібно ввести розповідь про інтерфейс та інструменти.

На етапі створення застосунку, потрібно використовувати виразні кольори з чітко окресленою стилістикою програми Adobe Illustrator. Використовувати ілюстрації з методички та презентацій до уроку. Використання несумісних кольорів суворо заборонено. Обов'язкове використовування анімацій.

Задля зацікавлення дитини повинен бути введений розповідач. Монолог повинен бути простим, живим, але з невеликим використанням технічної лексики. Він повинен розповісти про те, як користуватись застомунком, задіяні інструменти, про структуру і алгоритм роботи з ілюстраціями та перспективи роботи графічним дизайнером

При озвучуванні застосунку, голос для тексту потрібен бути чітким, гучним та зрозумілим. Задля створення хорошої атмосфери допускається використовування приємної музики, яка буде наводити думки на робочий лад. Заборонено використання вбудованої музики, тільки реальні композиції без тексту та різких змін темпу та гучності

Після закінчення роботи застосунку повинно озвучитись питання про подальше навчання. Та обов'язково запропонувати вибрати інший рисунок для розбору.

При тестуванні застосунку необхідно перевірити абсолютно всі варіанти переходів по робочому матеріалу. Провести тестування всіх варіантів застосунку(для браузеру, для ОС Windows та для Scratch). Більше всього уваги потрібно виділити на озвучку. Текст повинен читатися по мірі проходження кожного етапу.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення мети, поставленої в роботі в додатку, що розробляється, повинні бути реалізовані:

- комфортний перегляд інформації щодо навчання;
- можливість запуску окремих частин програми;
- приємний звуковий супровід;
- інтуїтивно-зрозумілий інтерфейс;
- візуально красива картинка;
- пошук потрібного матеріалу по ключовим словам;
- керування застосунком повинно здійснюватись клавіатурою та мишею.

1.5.1. Вимоги до інформаційної безпеки

Інформаційна безпека — практика запобігання несанкціонованому доступу, використанню, розкриттю, спотворенню, змінам, дослідженням, запису або знищенню інформації. Це універсальне поняття застосовується незалежно від форми, яку можуть набувати дані.

За класифікацією, загрози інформаційної безпеки можуть бути розділеними за різними ознаками:

- загрози конфіденційності;
- цілісності;

- доступності.

Конкретно по розроблюємому застосунку можна сказати, що загроз конфіденційності, в оригінальній програмі, немає.

Можна виділити такі вимоги до інформаційної безпеки:

- цілісність даних;
- захист від неавторизованого редагування;
- доступність інформації для всіх користувачів.

Частіше за все, надійність роботи таких програмних засобів залежить від надійності операційної системи, на якій буде працювати застосунок.

1.5.2. Вимоги до складу та параметрів технічних засобів

Для запуску застосунку, технічні засоби повинні відповідати наступним мінімальним вимогам, що наведені в табл. №1.

Таблиця 1.1

Мінімальні вимоги до технічних засобів

ОС:	Підтримка файлів формату .exe, або наявність браузеру
Розрядність:	підтримка 64 або 86 розрядного процесору та операційної системи
Процесор:	Pentium® 4 або AMD Athlon 1.5 ГГц
Відеокарта:	Intel HD Graphics
Оперативна пам’ять:	2 ГБ
Місце на жорсткому диску:	Не менше 2 ГБ
Наявність браузеру	+

Рекомендована конфігурація для запуску застосунку:

Таблиця 1.2

Рекомендована конфігурація для запуску застосунку

ОС:	Підтримка файлів формату .exe, або наявність браузеру
Розрядність:	підтримка 64 або 86 розрядного процессору та операційної системи
Процесор:	Intel core I3 2110 або AMD Athlon 2.3 ГГц
Оперативна пам’ять:	4 ГБ
Відеокарта:	Intel HD Graphics 2000
Місце на жорсткому диску:	2 ГБ
Наявність браузеру	+

Наведені вище технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче зазначених, розроблений програмний виріб буде функціонувати відповідно до вимог щодо надійності, швидкості обробки даних і безпеки.

1.5.3. Вимоги до інформаційної та програмної сумісності

Програма повинна бути сумісною з ОС Microsoft Windows 7 та її пізнішими версіями.

Розробку проводити візуально-блочному подієво-орієнтованому середовищі програмування Scratch.

В результаті розробки, застосунок має мати декілька варіантів:

- програма, яка запускається в браузері;
- програма, яка запускається на ОС Windows;
- файл програми, для запуску в середі розробки Scratch.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

2.1. Функціональне призначення програми

Застосунок «Додаток для навчання дітей Adobe illustrator на базі Scratch» призначений для:

1. Ознайомлення дітей-студентів з векторним графічним редактором «Abode Illustrator», щоб дитина змогла обрати для себе бажаний курс без великого вкладення часу на дорогу до школи програмування та на пробний урок.
2. Оптимізації робочих годин викладача курсу «Комп'ютерна графіка. Векторна графіка Adobe Illustrator». Завдяки йому викладач має більше часу на підготування власних проектів для показу дітям і заохочування їх приєднатися до курсу.
3. Зменшення витрат школи на заробітну плату викладачів за час проведення вступного (пробного) уроку.
4. Спрощення батьків в ознайомленні з програмним матеріалом.
5. Полегшення реклами типу «сарафанне радіо» (бо ще немає жодної школи, яка б використовувала такий спосіб залучення до навчання).
6. Показ прикладу випускного, на наступний розбір проекту для курсу «Візуальне програмування Scratch».

На відміну від звичайних вступних відеокурсів для навчання з різних графічних редакторів, у цьому застосунку використано багато функцій та можливостей середовища розробки Scratch. Що в свою чергу такий великий проект викликає у студентів бажання не тільки перейти до іншого курсу, а ще й самостійно, чи з викладачем зробити подібний по складності проект.

Додаток розроблений для відкривання його в браузері та окремо на різних ОС з підтримкою файлів формату .EXE. Реалізовано на платформі Scratch, за допомогою вбудованого блочного середовища розробки.

Кожен «контрольний пункт» застосунку може запускатись за допомогою ключових слів. Які, в свою чергу будуть показуватись на робочому екрані програми. Також кожен етап спроектовано таким чином, щоб інформації на них не було забагато. Зроблено це для того, щоб учень міг в любий момент відлучитися від роботи і продовжити її на місці, де до цього зупинився.

2.2. Опис застосованих математичних методів

Для даної програми була розроблена та введена для використання функція (рис.2.1) для посимвольного друкування тексту віртуального помічника, для легкості читання та ефекту «присутності» його в циклі програми. Цей метод зчитує кількість символів в рядку і друкує ці символи через певний проміжок часу, тим самим створюючи ефект друкування тексту.

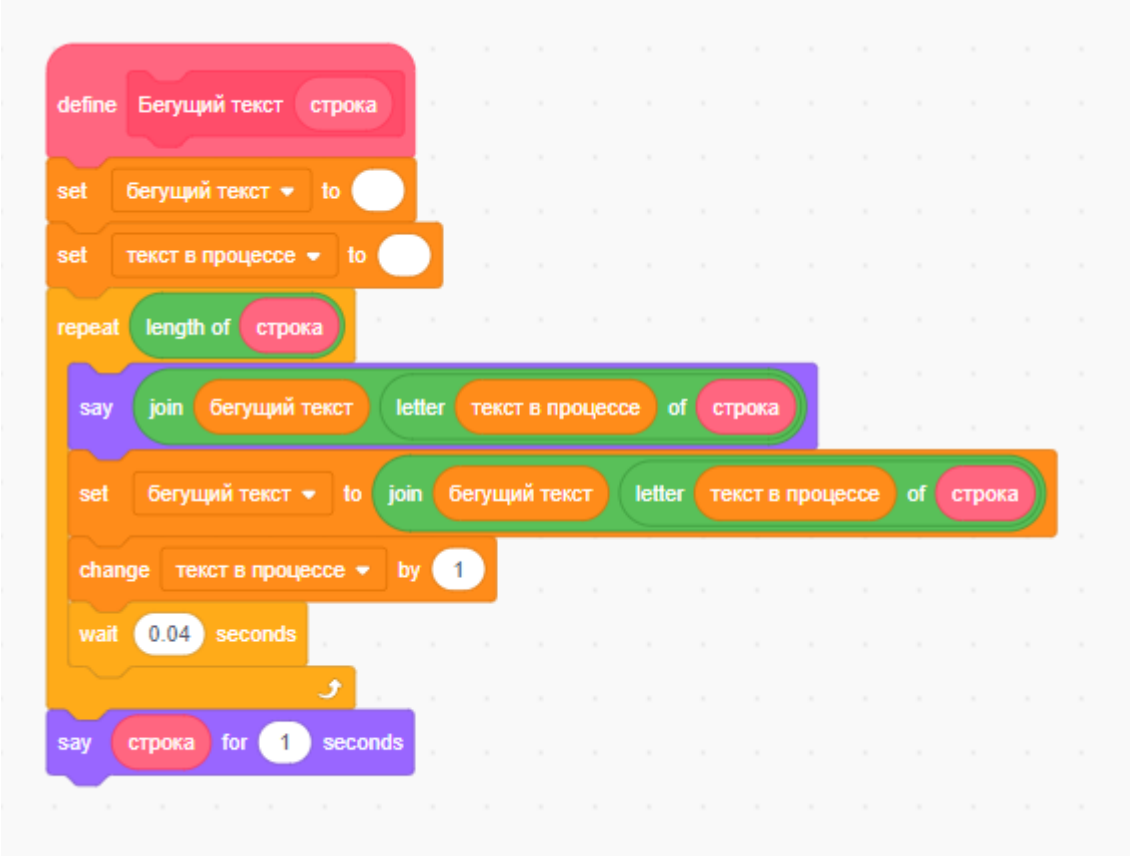

Рис. 2.1 Функція для тексту

Сенс цієї функціїї полягає в друкуванні тексту. Таким чином, сприймання інформації стає набагато простішим, ніж було без цієї функції.

2.3. Опис використаної архітектури та шаблонів проектування

В цьому застосунку використовується подієво-орієнтована архітектура програмування прописувалася поведінка для програми у разі будь-яких подій. Подією у системі вважається зміна її стану. Кожна подія запускає якийсь процес. Система, керована подіями, зазвичай містить два компоненти: джерела подій та його споживачі. Типів подій зазвичай теж два: ініціалізуюча подія та подія, на яку реагують споживачі.

Якщо класу потрібне сповіщення про якусь подію, розробник реалізує так званого слухача — ActionListener і дописує його до об'єкта, який ця подія може згенерувати.

Оскільки програми складаються з великої кількості асинхронних модулів (які не мають інформації про реалізацію один одного), їх легко масштабувати. Такі системи збираються як конструктор - прописувати залежності не потрібно, достатньо реалізувати новий модуль. Додатково асинхронна модель дозволяє досягти високої продуктивності програм.

Але є й мінуси такої архітектури - асинхронна натура таких програм ускладнює налагодження. Одна подія може запускати кілька ланцюжків дій. Якщо таких ланцюжків буде багато, то зрозуміти, що саме спричинило збій, може бути важко. Для вирішення проблеми доводиться опрацьовувати складні умови обробки помилок. Звідси випливає проблема з журналуванням - логи важко структурувати. Але ця проблема дуже легко вирішується банальним структуруванням коду.

Як повторювана архітектурна конструкція, використовувався базовий шаблон проектування – делегування. В цьому шаблоні об’єкт зовні виконує деяку поведінку, але насправді передає відповідальність за виконання цієї поведінки іншому пов’язаному об’єкту. Цей шаблон є фундаментальною

абстракцією і з нього реалізовані інші шаблони. Але шаблону делегування в цьому проекті достатньо.

2.4. Опис використаних технологій та мов програмування

Для розробки цього додатку було вирішено використовувати візуально-блочне подієво-орієнтоване середовище програмування Scratch.

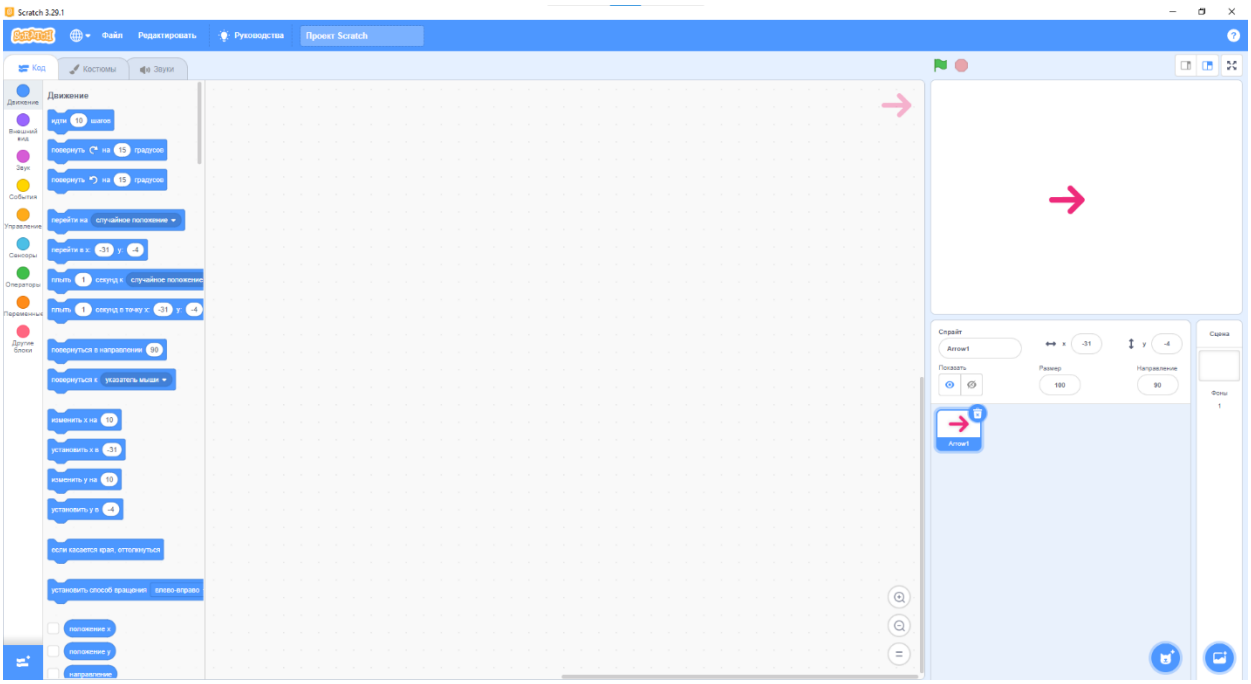

Рис.2.2. Інтерфейс платформи Scratch

Це пов’язано с декількома факторами, а саме:

- Scratch є одним з найпростіших середовищ програмування, що робить його одним з найкращих для старту дітей в IT. Однак в цьому проекті використовуються важкі функції та методи рядків, що робить його цікавим не тільки для тих, хто хоче навчатися векторній графіці, а й майбутнім розробникам програмного забезпечення;
- Scratch використовують для підготовки випускників загальноосвітніх шкіл, і це нам дуже допоможе в їх же навчанні роботи зі складними алгоритмами, при збереженні інтересу до програмування;
- тестування програми може проводитися при залучанні самих студентів школи, що залучить їх одразу до двох направлень в галузі IT;

- а найголовніше те, що весь код Scratch можливо змінювати без великих зусиль. Тож для того щоб зробити таку програму під інший курс, потрібно витратити набагато менше часу ніж писати її з нуля.

Дуже гнучка система дозволяє використовувати будь які референтні зображення та програми для вирішення своїх питань у проектуванні та розробці застосунку. Але головна перевага - це те що програма пишеться не завдяки коду, який може бути складним або нудним для дітей, а різнобарвними блоками. Різноманітність кольорів зацікавлює дитину будь якої вікової групи. Саме таким чином дитина знайомиться с основами і суттю програмування, вчиться розбивати задачі на складові, які набагато простіші. Також, вони засвоюють важкі математичні концепції, такі як змінювані координати або випадкові числа. Створювати функції, методи, ази ООП.

Це все робить проект, який розроблявся для юних дизайнерів, дуже корисним і потрібним для інших студентів. Тож розробка на Scratch є одним із варіантів уразити багато направлень в навчанні, всього одним проектом.

При розробці в Scratch завжди є можливість завжди протестувати свій застосунок не виходячи з редактору, та навіть не проводячи контрольне збереження. При цьому є один мінус, який може відбити бажання у дитини продовжувати розробку. Це відсутність автозбереження, тому не варто нехтувати сповіщеннями про незбережений файл. Краще зайвий раз зберегти, ніж потім жалкувати про марне проведення часу і кропітливу роботу.

Від самого початку Scratch позиціонував себе як середовище для навчання програмуванню дітей, але чому не піти далі і не розробити програму для навчання дітей чомусь новому. Цей застосунок найбільше підійде тим, хто вже знає що таке подієво-орієнтоване програмування в Scratch. Як показує досвід і практика у викладанні – всі діти хоча б раз чули про Scratch. І багато з них розробляли свої ігри або мультфільми у ньому. Так що використання саме цього середовища обумовлено тим, щоб викликати почуття ностальгії, чогось дуже рідного. І тим самим отримати більше позитивних проявів та відгуків з їх боку.

У Scratch є ряд змінних властивостей кожного елементу програми, що можна налаштувати саме під потрібний проект. Це допомагає у налагодженні взаємозв’язків між частинами програми і в цілому, зробити проект візуально і функціонально привабливим.

Також тут можлива реалізація своїх функцій та скриптів, для реалізації специфічних алгоритмів. Тим самим розширюючи функціонал програми в декілька разів. Є можливість розробки власних скриптів, класів, методів і прив’язування їх одразу до декількох об’єктів.

Всі компоненти об’єктів запускають власні події, які в свою чергу запускають інші події. Це дуже зручно для покрокового програмування і простого доповнення та розширення програми.

Adobe Illustrator — це графічний векторний редактор і програма для дизайну, розроблена і продана компанією Adobe Inc. Приклад інтерфейсу наведений нижче на рис.2.3.

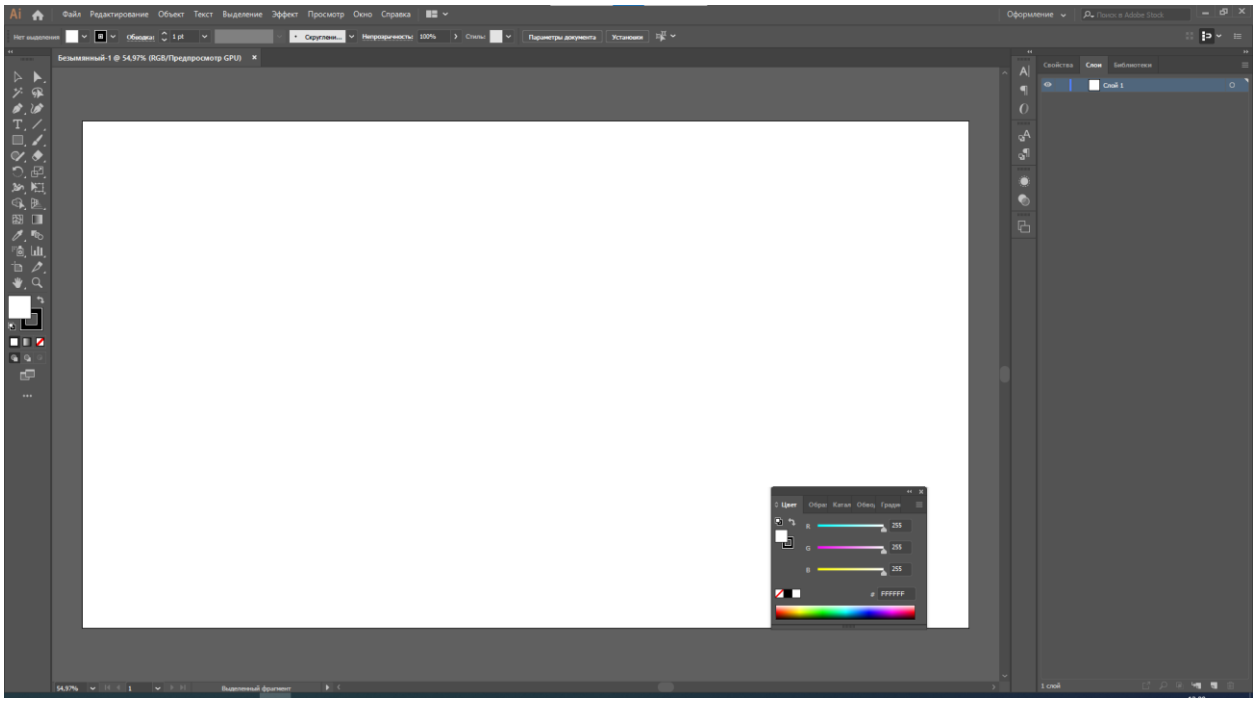

Рис.2.3. Інтерфейс графічного редактору Adobe Illustrator

Разом з Creative Cloud був випущений Illustrator CC. Ця 17 версія стала одним з переломних моментів компанії, бо вона була першою, яка продавалася лише в моделі сервісу на основі підписки, так само як і все інше програмне

забезпечення Creative Cloud, яке раніше називалося Creative Suite. Як частина Creative Cloud, ця версія принесла покращення в цій темі. Найпомітніші з них:

- синхронізація кольорів
- синхронізація шрифтів
- синхронізація налаштувань програми
- збереження документів у хмарі
- інтеграція з Behance (творча мережа спільної роботи)
- та інші функції, такі як новий штрих, сумісний інструмент типів, зображення в пензлях, вилучення CSS та пакування файлів.

Також плюсом буде сумісність з графічним редактором для технічних ілюстрацій Inkscape, форматом якого є SVG, який підтримується Adobe Illustrator. А в свою чергу Inkscape може експортувати файли в форматах які добре розпізнаються Adobe Illustrator.

Бічна панель з різноманітними інструментами, яка з’являється зліва на екрані для вибору, створення та керування об’єктами чи ілюстраціями в Illustrator. Ці інструменти можна розділити на підтипи наступним чином: малювання, введення тексту, зміна форми, розрізання та вирізання, символіка, переміщення та масштабування, а також робота з графіками. Деякі інструменти мають маленький трикутник у нижньому правому куті значка панелі інструментів. Маленький трикутник має можливість переглядати або розгортати деякі приховані інструменти, утримуючи кнопку миші на трикутнику, або натиснувши праву кнопку миші.

Також існують мобільні версії застосунку для користувачів Android та iOS Разом з Illustrator, його зараз продає Adobe через Creative Cloud. Малюнки, зроблені за допомогою програми Illustrator Draw, можна експортувати до настільних програм Adobe Illustrator.

Adobe Illustrator — це популярна програма для редагування векторної графіки та проектування, яка в основному маніпулює векторною графікою, яку використовують як художники, так і графічні дизайнери для створення векторних зображень. Це, безумовно, найпростіший спосіб генерувати та

редагувати масштабований вміст для відтворення на різноманітних носіях будь-якого розміру. Ми можемо створювати дизайни з файлами невеликого розміру, які можна роздрукувати у високій якості. Найчастіше він використовується для створення логотипів компаній, ілюстрацій, графіків, діаграм, карикатур із реальними фотографіями, будь-якої рекламної продукції та іншого рекламного використання або навіть особистої роботи, як у друкованому, так і в цифровому вигляді. Створювати та редагувати будь-які об’єкти з Illustrator легко. Ми можемо створити будь-що, як у малюнках від руки. Імпорт фотографій — це майстерний хід у цій програмі, який працює як керівництво для відстеження та перефарбовування певного об’єкта, перетворюючи його на художній твір, що дає схожий вигляд малюнків від руки. У деяких країнах є навіть навчальні інститути з Adobe Illustrator, які пропонують навчання на професійному рівні та охоплюють усі важливі поняття від базового до просунутого рівня.

Як правило, файли експортуються з Illustrator для однієї з двох цілей: для друку або для викладення в Інтернет. Буде суттєва різниця між файлом, оптимізованим для друку, і файлом, який використовується в Інтернеті, і в кінцевому підсумку це може мати величезний вплив на кінцеву якість роботи. Ідеальним форматом файлу для друку документа, створеного в Illustrator, є PDF. PDF-файли будуть більшими, ніж багато інших варіантів, але також нададуть принтеру більше інформації для друку. Цей файл також не втратить якість з часом і використанням.

З такою кількістю програмного забезпечення, доступного на ринку, нелегко вибрати правильний. Вибравши Illustrator, дизайнери можуть створювати ілюстрації з плавним вирівнюванням, малюючи ідеальні для пікселя фігури. Illustrator має власні плагіни, які допомагають перетворити порожню веб-сторінку в блискучу веб-сторінку. Його функції та версія Creative Cloud роблять його ідеальним програмним забезпеченням для графічного дизайну. Будучи частиною Creative Cloud, ви можете використовувати програму як на комп’ютері, так і на Mac. Цей графічний

редактор достатньо часто оновлюється та допрацьовується. Завдяки всім перерахованим вище перевагам, та актуальності вибір саме Adobe illustrator є більш ніж доцільним.

Розроблюємий застосунок повинен повноцінно працювати на всіх ОС на в браузері. Основні використуємі характеристики:

- багатозадачність та многопоточність. Scratch підтимує многопоточність, тож задля хорошої швидкодії нам потрібна ОС яка це підтримує. На даний момент часу майже всі збірки систем можуть функціонувати з синхронною загрузкою потоків процесора;
- апаратно-незалежне програмування. Всі проекти Scratch можуть можуть працювати незалежно від конкретно обраної ОС. Це все завдяки драйверам. Тож не має потреби піклуватися про апаратуру та їх сумісність з ОС. Програма створена на одному «залізі» буде безпроблемно працювати на іншому;
- віконна середа. Майже в усіх програмах є хоча б одне вікно, яке має стандартний вигляд і містить обов’язкові елементи. І цей застосунок не є виключенням.

2.5. Опис структури системи та алгоритмів її функціонування

2.5.1. Розробка концепції застосунку

Концепцією застосунку є інтерактивний додаток, за допомогою якого дитина матиме змогу навчитися базовій роботі в графічному редакторі Adobe Illustrator.

При запуску програми, користувача зустрічає віртуальний помічник, який буде супроводжувати протягом всього навчання. При подальшій роботі програми дитина навчиться, як зробити свій перший проект. Від установлення Adobe illustrator і до збереження готового результату в різних форматах для різних цілей. На вибір дається дві роботи на розбір. Також їй буде

запропоновано продовжити навчання у одній з ІТ шкіл. По закінченню роботи застосунку, буде пораджено вибрати ще один проект для розбору для подальшого навчання та закріплення матеріалу.

Програма забезпечує можливість легко додавати нові проекти, змінювати хід розбору та його пояснення.

2.5.2. Розробка алгоритму застосунку

Наступним етапом після планування є розробка. При розробці алгоритму необхідно визначити послідовність дій, які потрібно програти для досягнення потрібного результату. Сам алгоритм не є складним, але потрібно врахувати дуже багато факторів при конструюванні порядку програвання пунктів програми. Ось декілька з них:

- комфортний перегляд інформації;
- утримання інтересу у дитини на психологічному рівні;
- зацікавлення дитини в подальшому розвитку у цій сфері;
- підтримання хорошого настрою протягом навчання;

Результатом етапу розробки цього алгоритму є докладний опис алгоритму та(або) його блок-схема.

Алгоритм, що реалізує логіку застосунку, виглядає наступним чином:

1. Запуск програми.
2. Ознайомлення з програмою.
3. Можливість повернення до любого з пунктів програми.
4. Далі проходить сам учбовий процес, який в свою чергу ділиться на підпункти:

4.1. Установка.

4.2. Налаштування.

4.3. Інтерфейс

4.4. Інструменти

4.5. Вибір рисунку

4.6. Рисунок 1

4.7. Рисунок 2

4.8. Рисунок 3

4.9. Збереження проекту

5. Пропозиція закріплення знань

6. Анонс навчання на курсі

7. Вихід з програми

Умовна блок-схема даного алгоритму наведена на рис.2.4.

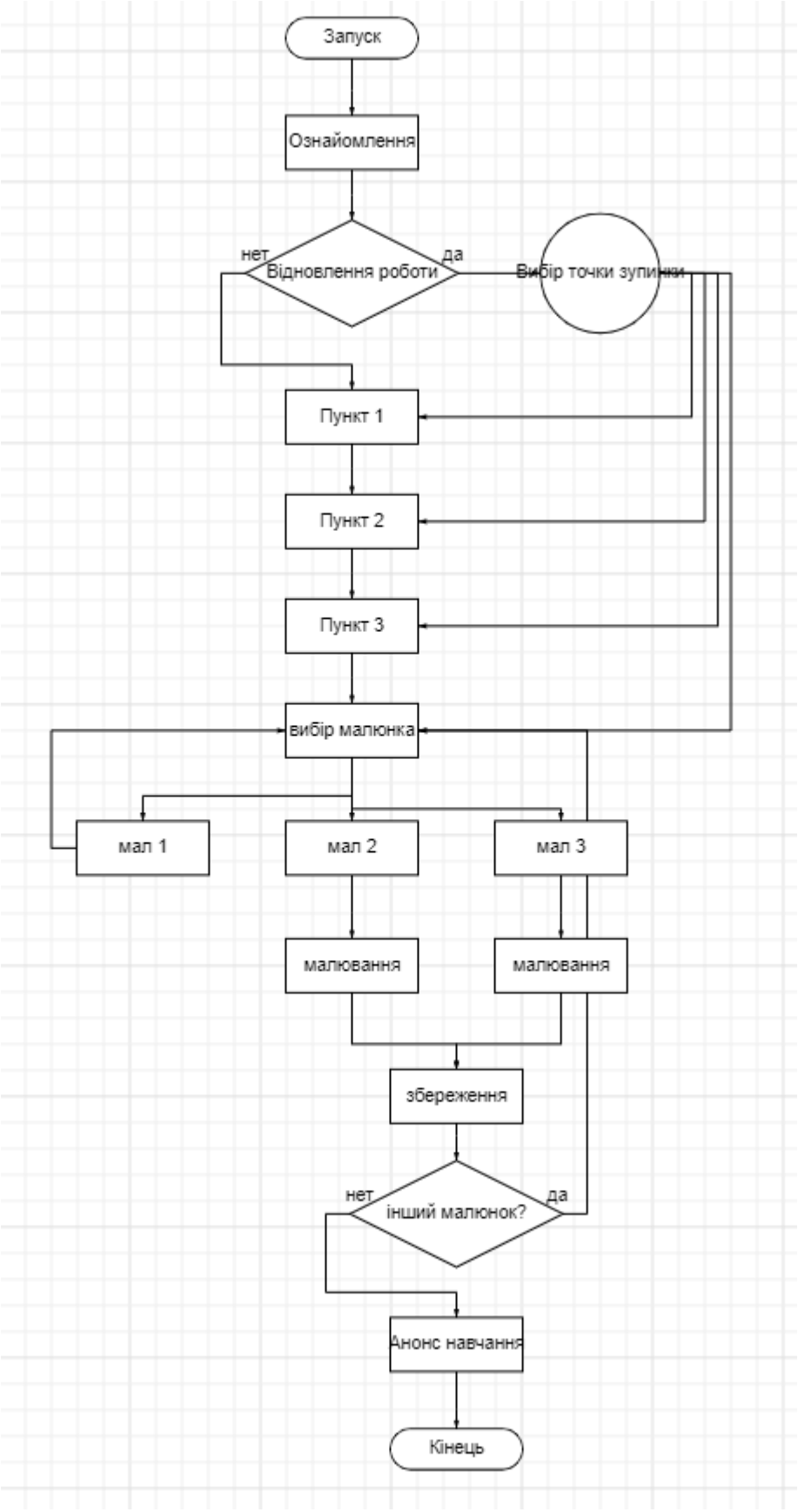

Рис. 2.4. Умовна блок-схема алгоритму застосунку

2.5.3. Опис структури та роботи меню

Єдина форма меню має такі заголовки:

- «Установка»;
- «Настройка»;
- «Интерфейс»;
- «Рисуем колу»;
- «Рисуем суши»;
- «Инструменты».

Меню паузи не було реалізовано за його непотрібністю. Дитина завжди може натиснути на конку «Меню» і продовжити з того місця, де зупинився. Схема меню зображена на рис. 2.5.

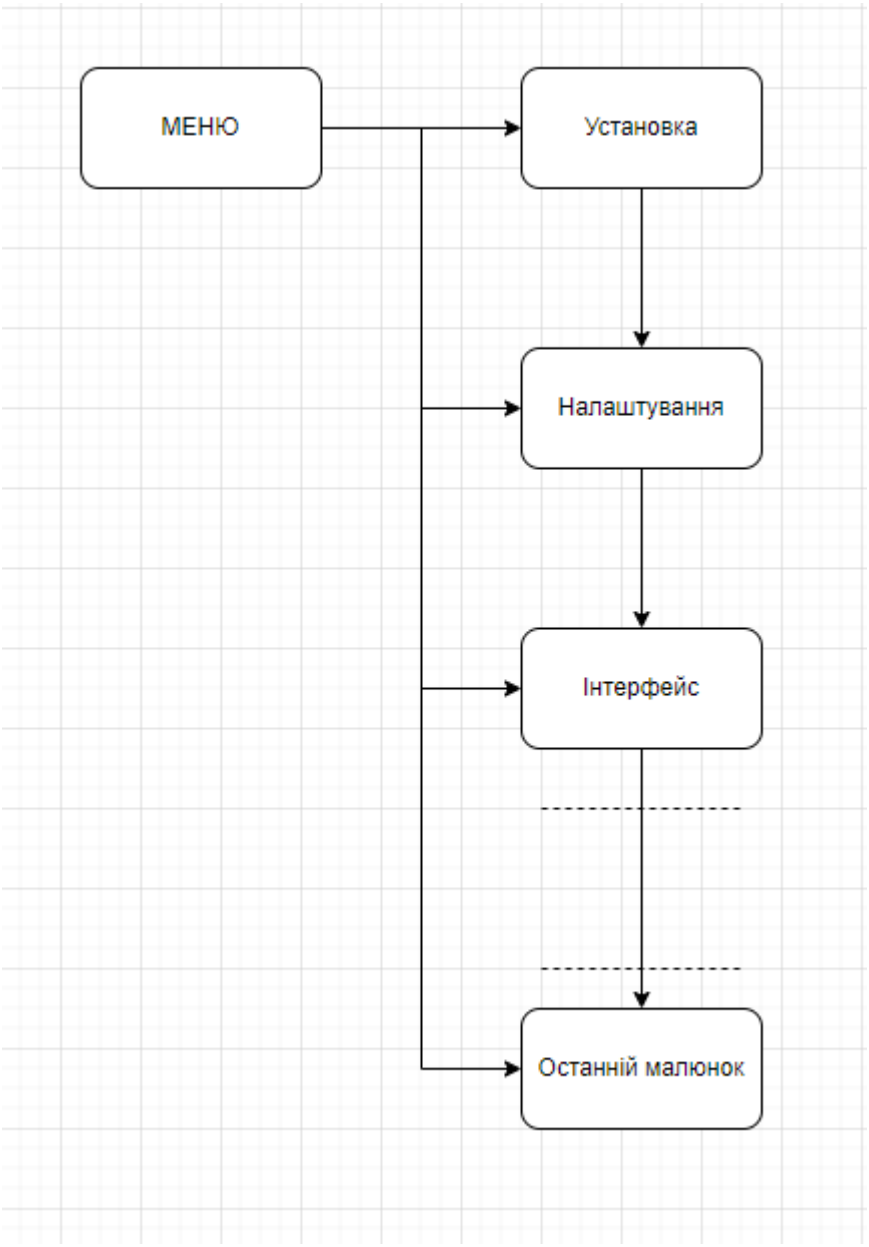

Рис 2.5 Схема роботи головного меню

2.5.4. Програмна реалізація компонентів програми

2.5.4.1. Реалізація графічної частини

Вся графічна складова Scratch – це спрайти. Це об'єкт, який виконує будь-які дії в проекті. Окрім виконання команд, вони вміють пересуватися. Про пересування трохи пізніше. У цих спрайтів є багато налаштувань, такі як:

- положення на координтаній сітці проекту;
- розмір;

- направлення руху;
- напрям погляду;
- видимість на екрані;
- положення в різних пластах проекту, відносно інших спрайтів.

Для реалізації графічних об’єктів у середовищі Scratch є спеціальне вікно-редактор, в якому можна як зробити спрайт власноруч, так і просто відредагувати вже готовий.

Графічний редактор Scratch – інструмент створення зображень у Scratch. Більшість прогресивних скретчерів створюють власні спрайти, костюми і фони. Ці зображення можуть бути використані для різних цілей. Це одна з відмінних функцій Scratch від інших середовищ програмування, що не включають вбудований графічний редактор.

Графічний редактор поділяється на векторний та растровий редактори. Векторний графічний редактор відрізняється від растрового тим, що не має пікселів у зображенні та векторна картинка виглядає красиво у будь-якому розмірі. Приклад інтерфейсу цього редактору наведений на Рис 2.6.

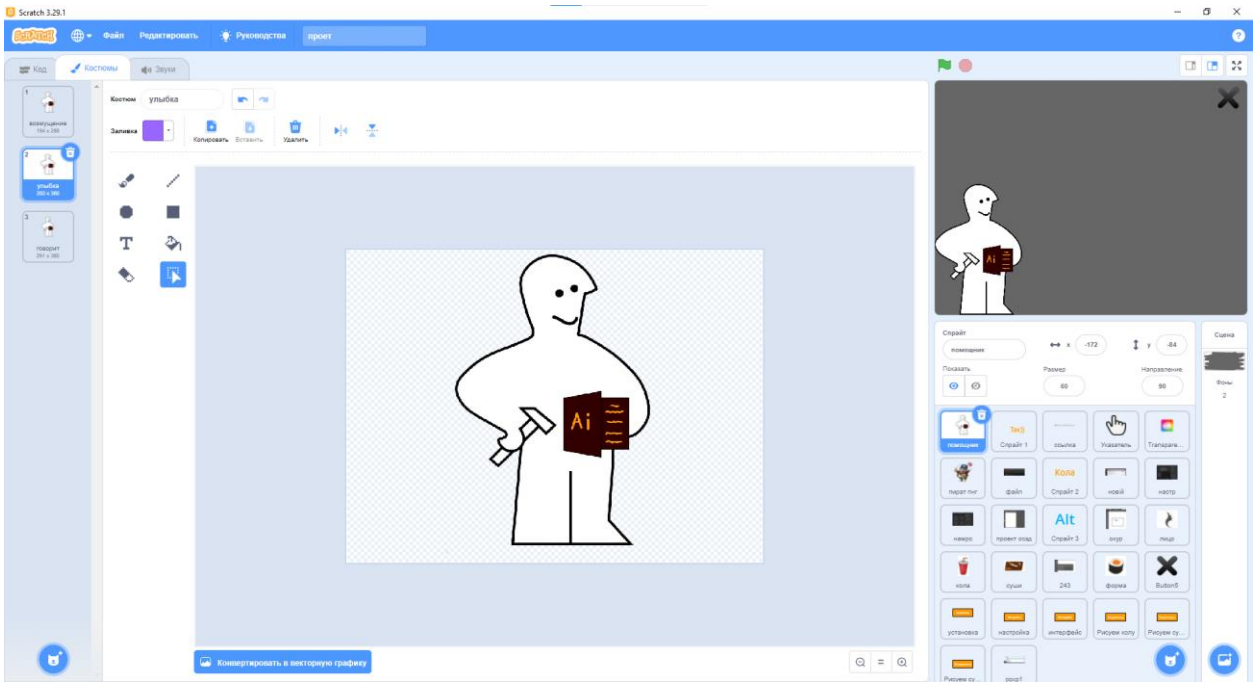

Рис. 2.6 Вбудований графічний редактор Scratch

Тут є декілька інструментів, функції та призначення яких наведено нижче:

 Пензель. Інструмент «Пензель» (Brush tool) є основним інструментом малювання. Він працює як традиційний інструмент малювання, наносить колір штрихами.

 Лінія. Інструмент «Лінія» дозволяє намалювати на полотні лінію за двома точками. Лінії можна малювати як векторні фігури, контури або пікселі.

  Круг і квадрат. Прості інструменти для створення стандартних фігур.

 Текст. Інструмент «Текст» дозволяє вбудовувати текст в виділену область. Також є можливість вибору шрифту, розміру, кольору та контуру.

 Заливка. Це інструмент, який дозволяє заповнити кольором (залити, зафарбувати) вибраний об'єкт, частину зображення чи шар. В Scratch для заливки передбачені інструменти «Заливка» та «Градієнт».

 Гумка. За допомогою інструменту гумка, можна випрявляти деякі частини малюнку. За своєю дією цей інструмент аналогічний шкільній гумці, для стирання олівця.

 Вибір зони. Цей інструмент використовується для вибору деяких частин малюнку для подальшого переносу на інше місце, або для копіювання в інші спрайти.

Ці інструменти допомагають у редагуванні вже готового спрайту, але яких може бути недостатньо для розробки власних персонажів с нуля.

Саме тому, для створення нових спрайтів було вирішено використовувати векторний графічний редактор Adobe Illustrator. Який, в свою чергу, і використовується для навчання у цьому застосунку. У цій програмі

працювати набагато простіше й ефективніше. В ньому вибір інструментів набагато більший, та відрізняється від редактору в Scratch.

Інструменти Adobe Illustrator:

- Selection Tool. Використовується для вибору будь-якого формату або об'єкта у вікні документа;
- Direct Selection Tool. Цей інструмент допомагає правити впорядковані розділи існуючої форми до форми нового формату;
- Magic Wand Tool. Може бути використаний в виборі форм та об’єктів сі схожими атрибутами. Цей інструмент є сприятливим для економії вашого часу. Замість вибору однакових форм один за одним, ви можете просто обрати їх за один клік;
- Lasso Tool. Це ще одна версія інструмента вибору. Це інструмент також домогає в приміщенні більше, ніж одиного об'єкту. Клацніть на інструменті Lasso і клацніть інструменти, щоб зробити те, що ви хочете. Цей інструмент є тільки іншим варіантом для вибору елементів;
- Pen Tool. Це інструмент, що найчастіше використовується в ілюстраторі. Спочатку він може здатися вам складним у використанні, але надалі ви будете із задоволенням користуватися цим інструментом. Інструмент «Перо» використовується для малювання шляхом створення контурів. Контури - це лінії з двома кінцевими точками, які називаються якорями, а якорі мають дві ручки, які допомагають змінювати форму об'єкта. Коли клікните правою кнопкою миші на інструменті «Перо» на панелі інструментів, ви побачите інші інструменти;
- Add Anchor Point Tool (Додати точку прив'язки) (+): Щоб додати опорні точки до контуру, виділіть контур, потім натисніть на інструмент «Перо» і підведіть вказівник до контуру, поруч із вказівником з'явиться знак плюс (+), натисніть, щоб встановити точку на контурі;
- Delete Anchor Point Tool (Видалити опорну точку) (-): Щоб видалити будь-яку опорну точку на контурі, виконайте ту саму процедуру, що й при додаванні опорної точки, але різниця в тому, що при наведенні

вказівника на існуючу опорну точку поруч із вказівником з'явиться знак мінус , натисніть , щоб видалити опорну точку;

- Curvature Tool. Цей інструмент знаходиться на панелі інструментів поруч із інструментом «Перо». Цей інструмент допомагає надати контуру плавну криву. За допомогою інструмента «Кривізна» можна створити будь-яку форму;
- Type Tool. Цей інструмент використовується для додавання тексту до зображення. У цьому інструменті є підінструменти, які можна побачити, клацнувши правою кнопкою миші на значку "Type". За допомогою інструмента «Type» можна набрати текст в один рядок або абзац. До складу інструменту «Type» входять такі інструменти:
  - Інструмент «Область»;
  - Інструмент «Тип за контуром»;
  - Інструмент «Вертикальний тип»;
  - Інструмент «Вертикальний тип області»;
  - Інструмент «Вертикальний тип на контурі»;
  - Інструмент «Дотик».

Та ще порядком двадцяти інших інструментів, які зроблені на основі попередніх.

Окрім ілюстрацій персонажів, у додатку використовуються знімки окремих частин робочого екрану Adobe Illustrator. Ці скріншоти робляться по мірі малювання запропонованих малюнків, або по мірі потреби, для пояснення роботи інструментів. Вони, в свою чергу робилися за допомогою вбудованого в ОС Windows застосунку, який можна визвати гарячими клавішами «Shift+Win+S»

Для розробки анімацій було вирішено використовувати вбудовані можливості Scratch. Реалізована вона завдяки циклічному збільшенню об’єкта, та плавному переміщенні по координатній сітці. Реалізація однієї з анімацій наведена на рис.2.7.

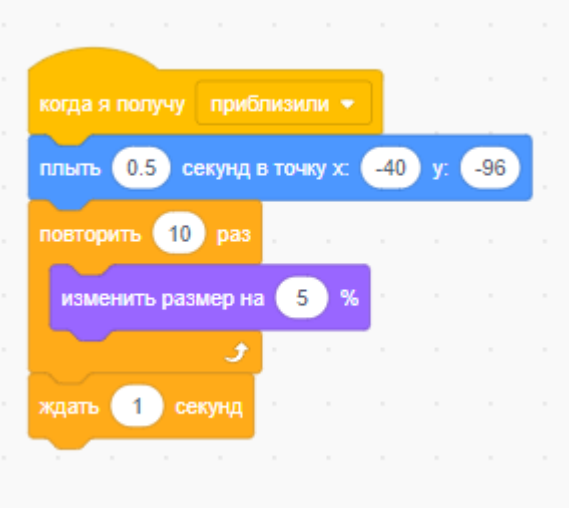

Рис. 2.7. Приклад анімації збільшення та переміщення

Анімація кнопок меню була зроблена шляхом зміни кольорового забарвлення самої кнопки. При наведенні на одну з кнопок, вона буде змінювати колір. Блок-схема алгоритму зміни костюму показано на рис.2.8.

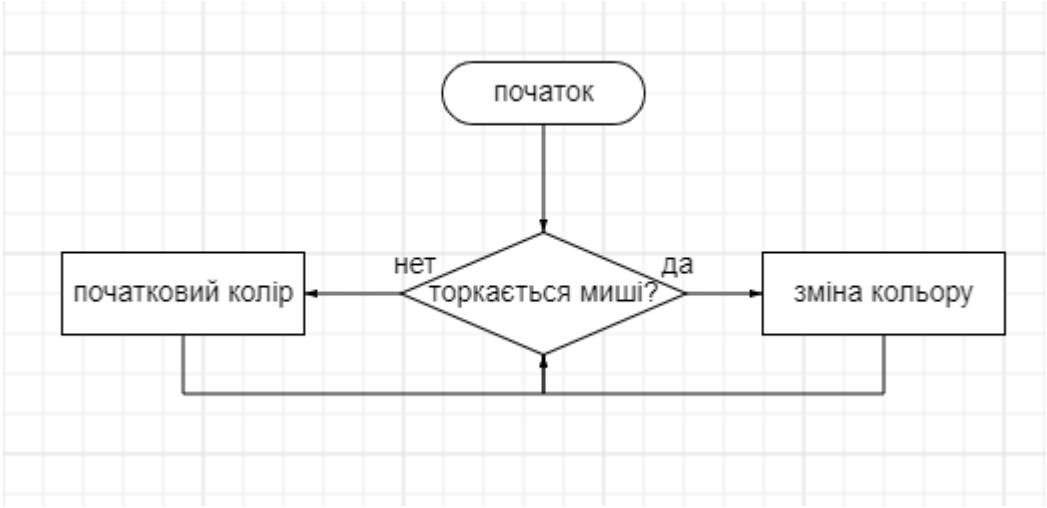

Рис. 2.8. Блок-схема алгоритму зміни костюму

Приклад роботи кнопок наведено на рис.2.9.

Рис. 2.9. Зміна кольору кнопки при наведенні миші на неї

Також для додавання живності нашому помічнику, було вирішено зробити декілька образів виразу обличчя. Ці образи наведені на рис.2.10.

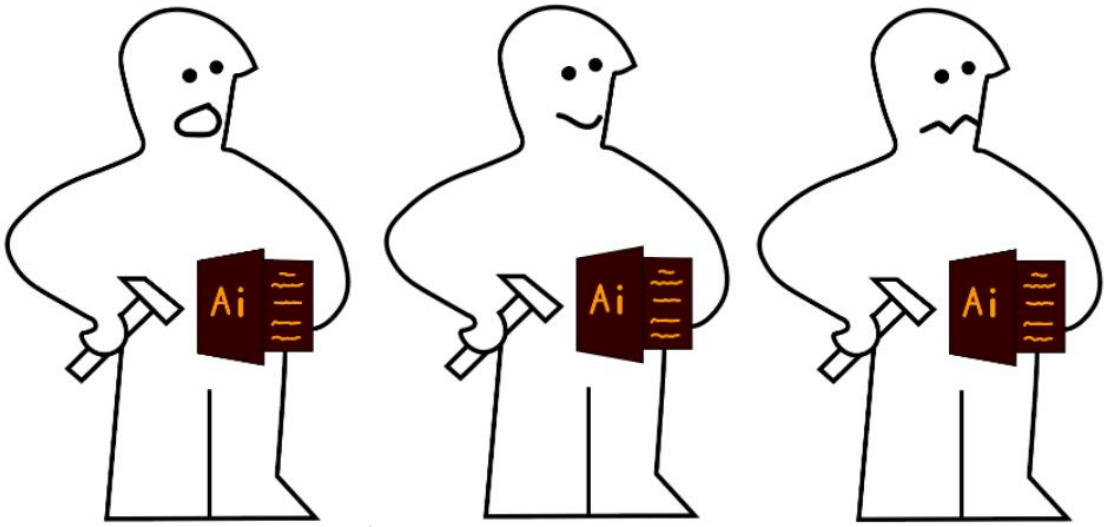

Рис. 2.10. Вирази обличчя віртуального помічника

Ці варіації також були зроблені за допомогою графічного редактору Adobe Illustrator. Було визначено, що для віртуального помічника потрібно використовувати прості форми, малу кольорову насиченість та простоту, для того щоб він не відволікав користувача від навчального процесу. Як прототип, був використаний образ, створений дизайнером Грахам Робсоном.

Як головну кольорову гаму, була використана палітра робочої зони Abode Illustrator. Це зроблено для того, щоб використання застосунку із ключовою програмою було комфортним і без різкої зміни кольорів.

2.5.4.2. Опис основних компонентів

В середовищі програмування Scratch кожен об’єкт представляє собою спрайт, який зберігає атрибути початкових компонентів. Так, наприклад, стандартні компоненти любого спрайту, які неможливо видалити:

- Назва
- Положення на координатній сітці
- Видимість
- Розмір
- Направлення

Приклад налаштувань компонентів одного спрайту наведено на рис.2.11.

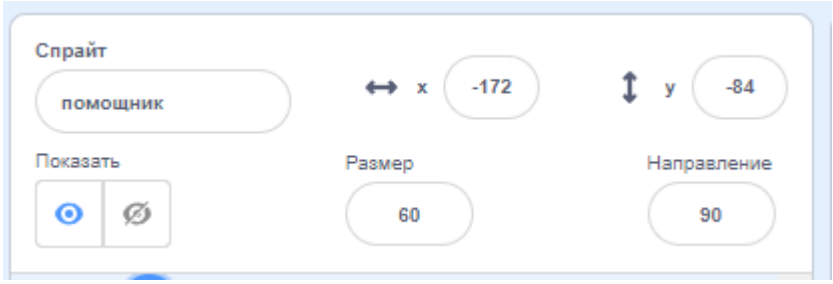

Рис. 2.11. Компоненти спрайту

Він зберігає параметри назви, розміру, направлення і положення об’єкту на координатній сітці відносно середини екрану застосунку. Зміну цих параметрів ми проводимо завдяки робочим командам-блокам, наведених на рис.2.12.

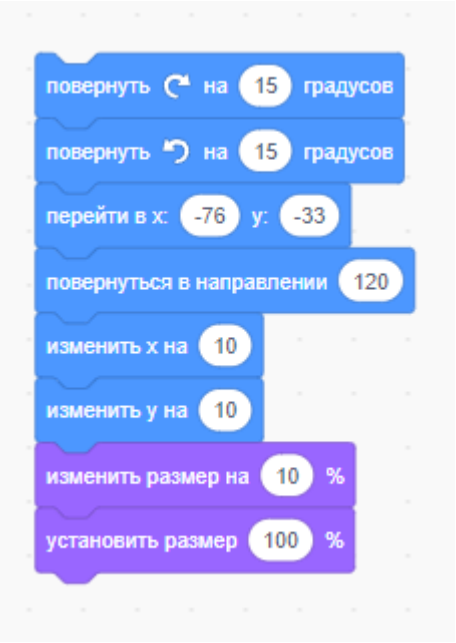

Рис. 2.12. Зміна налаштувань об’єкту

Також в проекті є звуковий супровід. Ці композиції були взяті з відкритого доступу. У цих композицій відсутні авторські права, тож їх використання без дозволу творця. Музика була підібрана таким чином, щоб не відволікати користувача від навчального процесу. У цій музиці немає тексту, різких перепадів гучності та чітко вираженого ритму. Алгоритми програвання музики надано нижче на рис. 2.13 - 2.14.

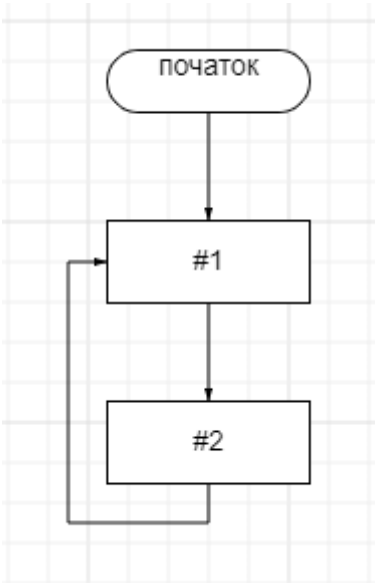

Рис. 2.13. Схема роботи музики

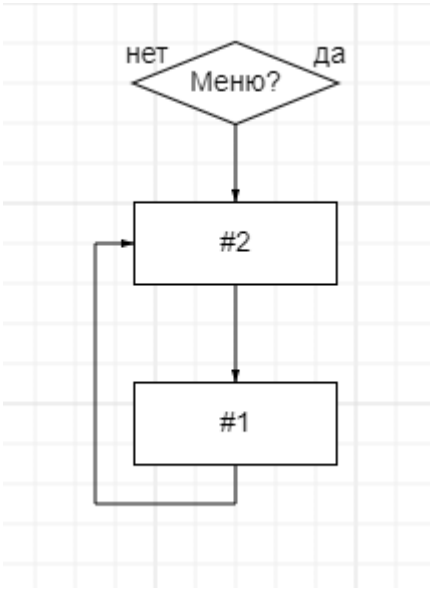

Рис. 2.14. Схема роботи музики при вході до головного меню

У середовищі програмування Scratch музика відображається так, як показано на рис. 2.15.

Рис. 2.15. Музика в Scratch

Реалізація музичного супроводу в блоках (рис. 2.16).

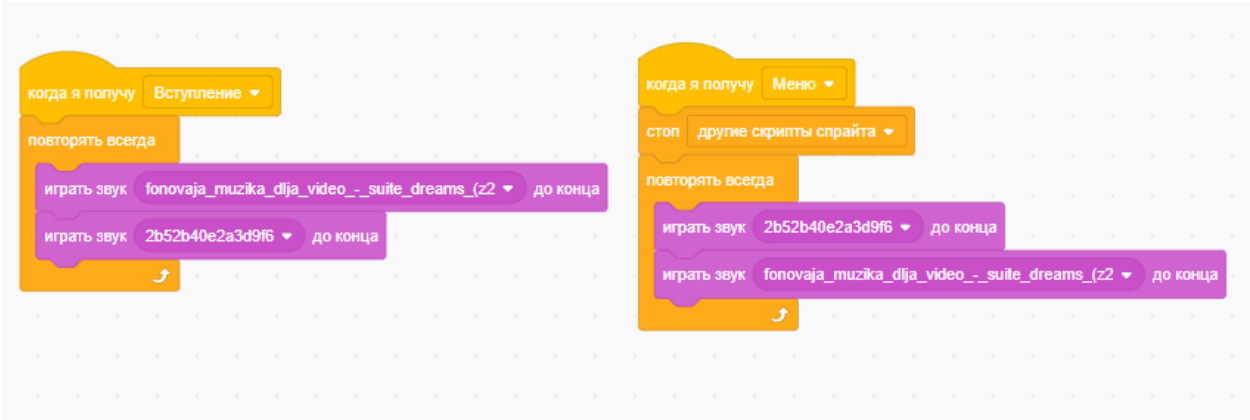

Рис. 2.16. Музика в блоках Scratch

Для озвучування написаного тексту, використовується синтез мови, вбудований в Scratch (рис. 2.17).

Синтез мови - у широкому сенсі - відновлення форми мовного сигналу за його параметрами, у вузькому призначенні - формування мовного сигналу за друкованим текстом. Частина штучного інтелекту. Синтезом мови перш за все називається все, що пов'язане зі штучним виробництвом людської мови.

У Scratch використовується параметричний синтез. Тут мовний сигнал представляється набором невеликого числа параметрів, що безперервно змінюються. Параметричний синтез доцільно застосовувати у тих випадках, коли набір повідомлень обмежений та змінюється не надто часто. Якість параметричного синтезу є достатньо високою. Однак, параметричний синтез не може застосовуватися для довільних, заздалегідь не заданих повідомлень.

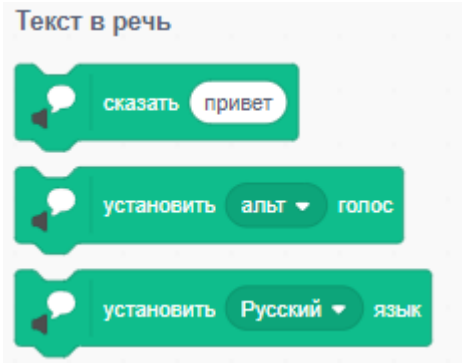

Рис. 2.17. Блоки мовного синтезу

В цьому додатку можна змінювати тембр голосу, та його мову. У даному застосунку використовується жіночий голос. Для озвучування були спеціальні блоки з підготовленим текстом (рис. 2.18).

Для того, щоб текст на екрані не розбігався з текстом, що промовляється, було вирішено вручну проставити паузи між цитатами віртуального помічника.

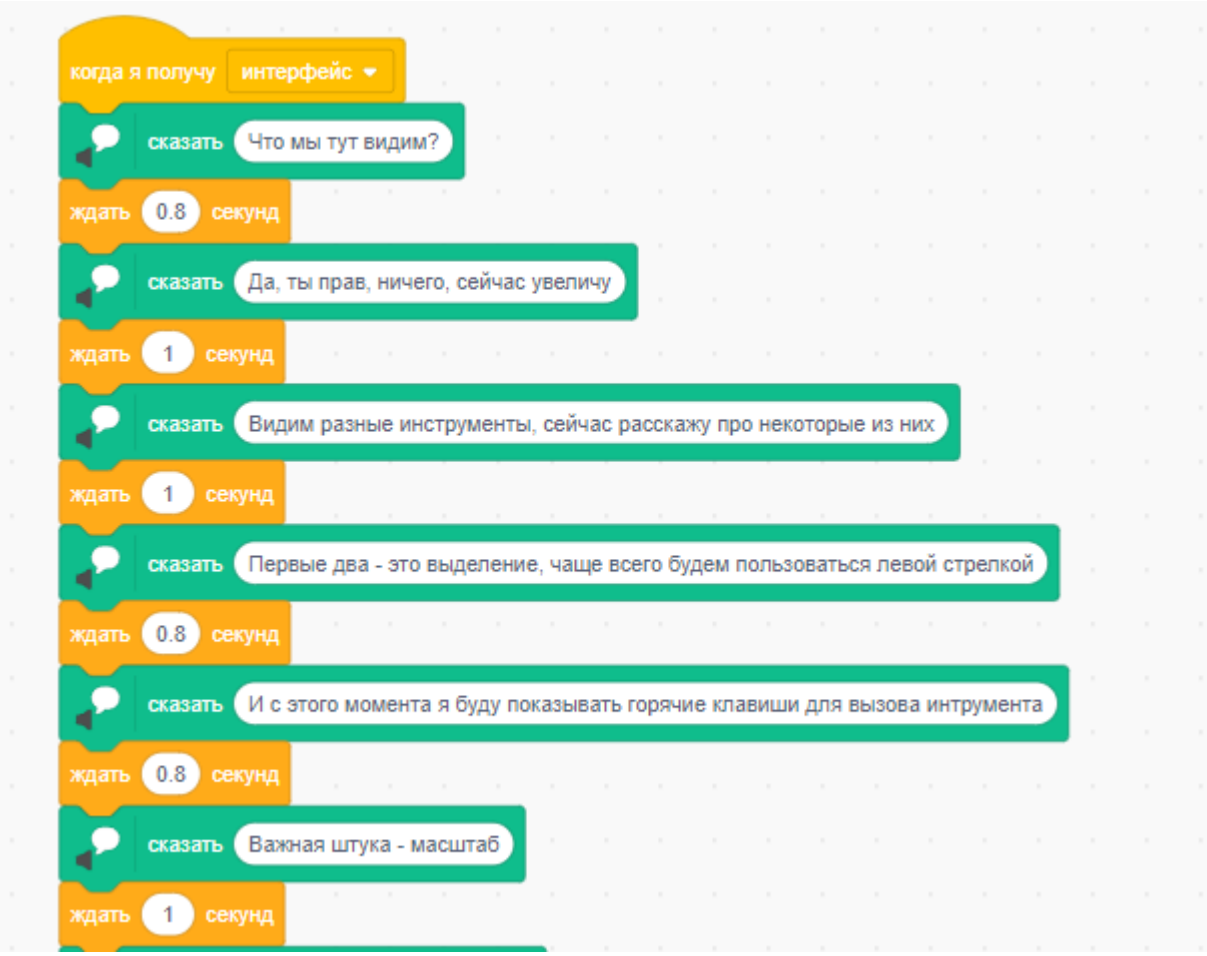

Рис. 2.18. Блоки синтезу тексту в голос з паузами

2.5.4.3. Опис роботи подій

Для того щоб розроблюєма програма працювала правильно та без дублювання вище описаного звуку і тексту, що з’являється, Використовувалися безліч подій. У більшості випадків, запуск нової події зупиняє роботу попередньої. Але для того щоб весь застосунок, з усіма додатковими засобами працювала коректно, розробилися додаткові події, які запускають окремі засоби для розваги та підтримання гарного настрою дитини. Наприклад – пірат (рис. 2.19).

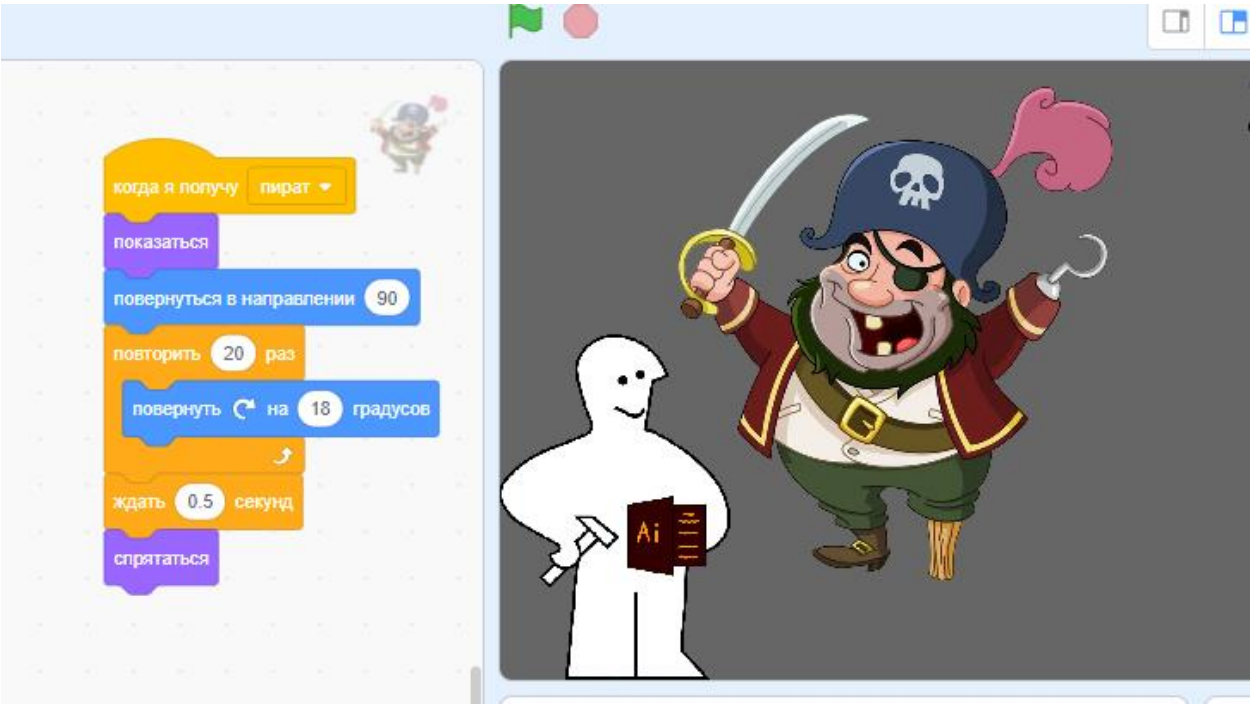

Рис. 2.19. Реалізація події «Пірат»

Повний список подій, в алфавітному порядку, приведено нижче:

- «Вступление»
- «Выбор колы»
- «Выбор лица»
- «Выбор рисунка»
- «Выбор суши»
- «Закруглить края»
- «Инструменты»
- «Инструменты выделения 2»
- «Инструменты выделения 3»
- «Инструменты выделения 4»
- «Инструменты выделения 5»
- «Инструменты выделения 6»
- «Интерфейс»
- «Интерфейс 2»
- «Кола»
- «Конец»
- «Меню»

- «Настройка»
- «Настройка слово»
- «Новый файл»
- «Параметры файла»
- «Первый показ иструментов»
- «Первый проект»
- «Пират»
- «Приблизили»
- «Пример слова»
- «Проект создан»
- «Проценты»
- «Работа»
- «Разрешение»
- «Рисунок»
- «Слово интерфейс»
- «Слои»
- «Сохранение»
- «Ссылка»
- «Суши»
- «Удаление надписей»
- «Установка»
- «Установка слово»
- «Alt»
- «Creative»
- «V»

За допомогою такої кількості подій, реалізація лінійного алгоритму програми, та перехід по ключовим словам, стали набагато простішими. Та саме з реалізацією подій було зроблене головне меню (рис.2.20) з переходами по учбовому матеріалу.

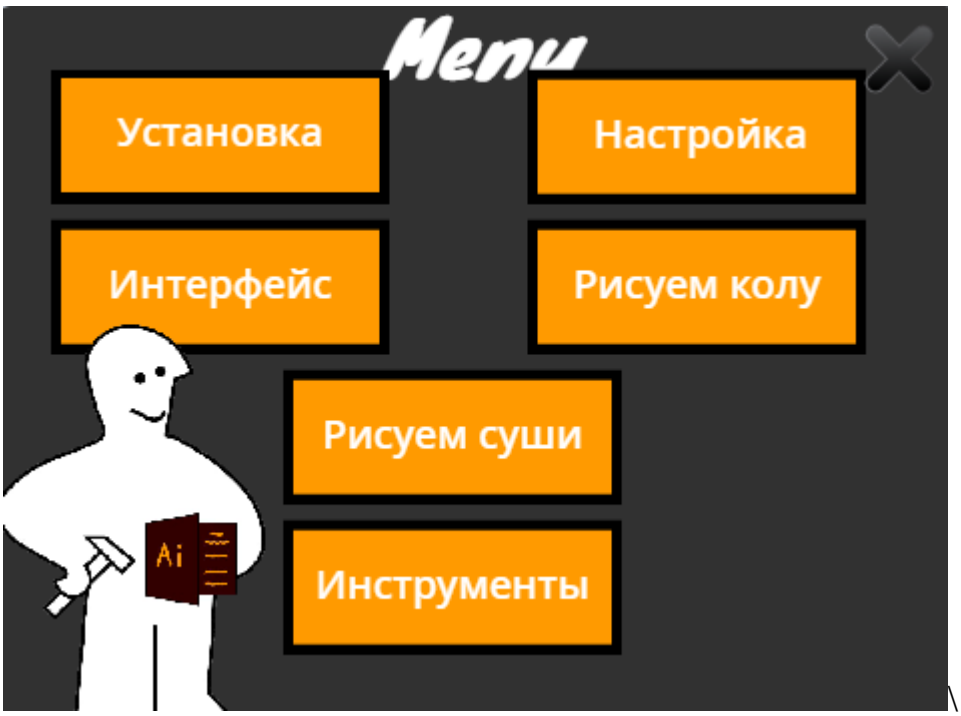

Рис. 2.20. Головне меню

2.6. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними розроблюємого застосунку є положення миші(для кнопок в головному меню) та ключові слова, які будуть введені у спеціальне поле (рис. 2.21).

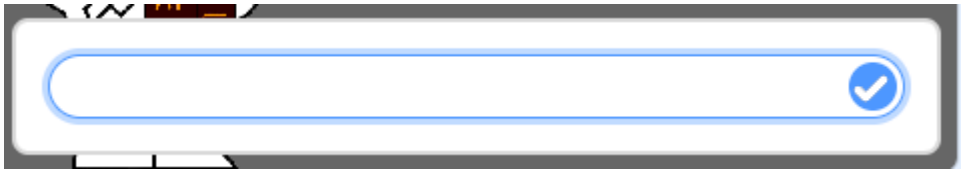

Рис. 2.21. Поле для введення ключового слова

Вихідні дані розроблюємого застосунку є текстова та графічна інформація, яка відображається в програмі Scratch, зображення, які користувач отримує від застосунку при продовженні його роботи.

2.7. Опис розроблювальної системи

2.7.1. Використані технічні засоби.

Для розробки додатку для навчання дітей Adobe illustrator на базі Scratch був використаний ПК з наступними характеристиками:

- Процесор: Intel Core i3-9100f 4 ядра3.6 GHz;
- ОЗП: 16 GB RAM;
- Відеокарта: Rx 570 4Gb;
- Монітор: 24" 1920x1080 пікселів;
- Клавіатура та комп’ютерна миша;
- ОС: Windows 10;
- Накопичувач: M2 SSD 512Gb

Тестування проводилось на декількох комп’ютерах з різними характеристиками. Таке тестування обумовлене різністю комп’ютерів дітей, які будуть використовувати застосунок. Характеристики тестувального ком’ютеру №1:

- Процесор: Intel Core i3-2110 2 ядра 3.1 GHz;
- ОЗП: 8 GB RAM;
- Відеокарта: Radeon hd 5570 1Gb;
- Монітор: 24" 1920x1080 пікселів;
- Клавіатура та комп’ютерна миша;
- ОС: Windows 10;
- Накопичувач: HDD 320 gb

Характеристики тестувального ком’ютеру №2:

- Процесор: Intel Pentium n6000 3.3 GHz;
- ОЗП: 4 GB RAM;
- Відеокарта: Intel UHD Graphics;
- Монітор: 13" 1280x720 пікселів;
- Вбудована клавіатура та комп’ютерна миша;

- ОС: Windows 7;
- Накопичувач: HDD 320 gb

Характеристики тестувального ком’ютеру №3:

- Процесор: AMD a10-8770;
- ОЗП: 2 GB RAM;
- Відеокарта: Radeon hd 5570 1Gb;
- Монітор: 18" 1280x720 пікселів;
- Клавіатура та комп’ютерна миша;
- ОС: Windows 7;
- Накопичувач: HDD 500 gb

2.7.2. Використані програмні засоби.

Застосунок для навчання дітей Adobe illustrator на базі Scratch був розроблений у візуально-блочному подієво-орієнтованому середовищі програмування Scratch, використовуючи вбудоване програмне середовище.

Для створення одного виконуваного файлу був використаний браузерний застосунок TurboWarp, який заархівував всі залежності та файли ресурсів в один файл, та зробило можливим запускати застосунок без середовища програмування Scratch.

Застосунок сумісний з будь-якою ОС, на яких можлива інсталяція браузеру та(або) середовища програмування Scratch v2.0 і вище.

2.7.3. Виклик та завантаження програми

Спосіб виклику та запуску є стандартним для файлів в ОС Windows, Linux та ін. Застосунок не потребує додаткової установки в систему та може бути завантажений та викликаний з носія, таких як флешка, диск або жорсткий диск.

Для запуску в браузері потрібно відкрити офіційний сайт Scratch та відкрити файл формату sb3 за допомогою браузерного додатку.

Для запуску з середовища програмування Scratch, попередньо встановленому на комп'ютер, потрібно просто відкрити файл з проектом в системі. Це робиться за допомогою кнопки «файл» - «запустити з комп'ютера» (рис. 2.22).

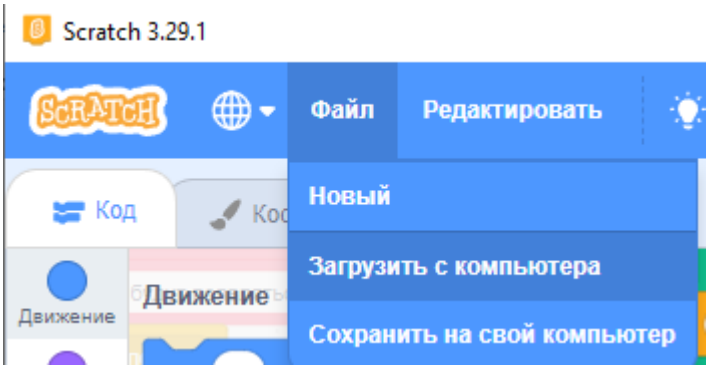

Рис. 2.22. Запуск додатку через Scratch

2.7.4. Опис інтерфейсу користувача

При запуску застосунку, користувач побачить віртуального помічника який одразу почне розповідати про функціонування програми (рис.2.23).

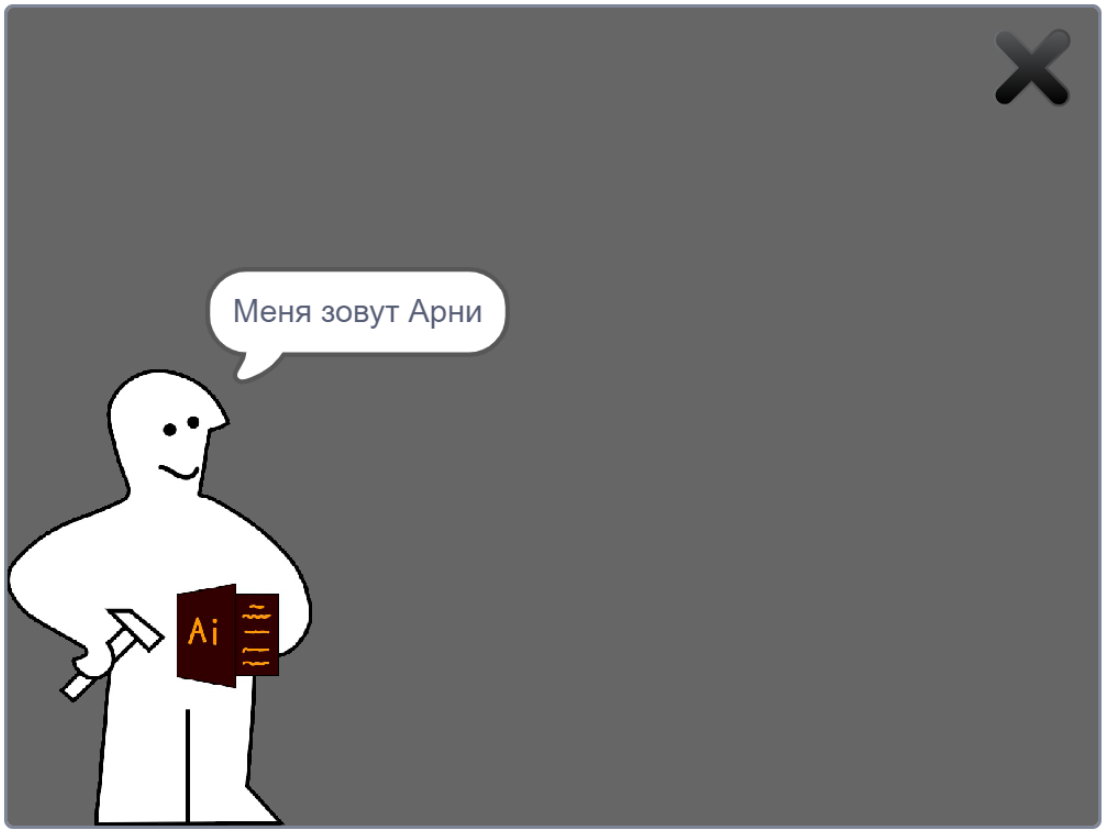

Рис. 2.23. Вигляд програми, при запуску

По мірі програвання застосунку, користувач побачить ключові слова. Перше, пробне повідомлення, покажеться ближче до середини екрану. Колір слів був обраний саме помаранчевим, задля створення та підтримання атмосфери та кольорового забарвлення графічного редактору Adobe Illustrator. Друге слово, на відміну від першого, буде розташовано у місці, яке одразу впадає в очі, але не буде надокучати, а саме зверху справа (рис. 2.24). Їх можна прибрати, натиснувши кнопку «пробіл».

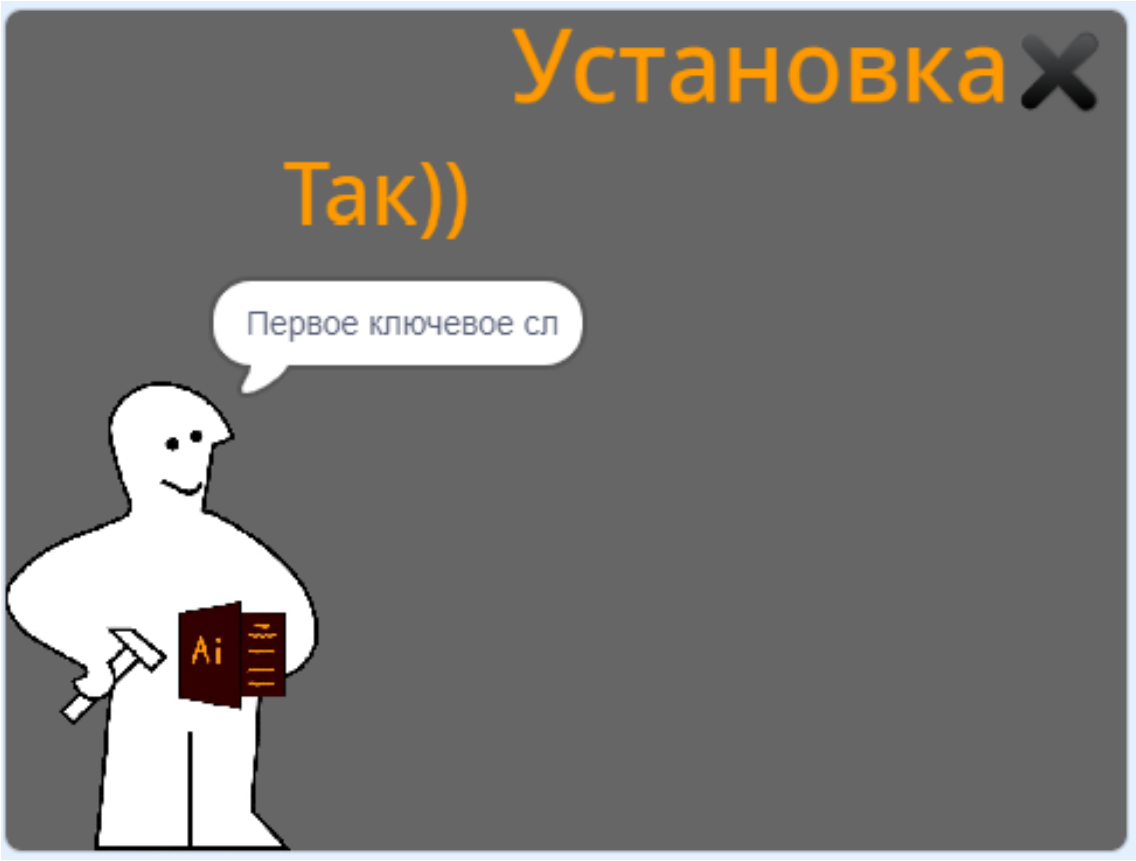

Рис. 2.24. Вигляд положення ключових слів

Ключові моменти роботи програми наведено нижче на рис.2.25-2.30.

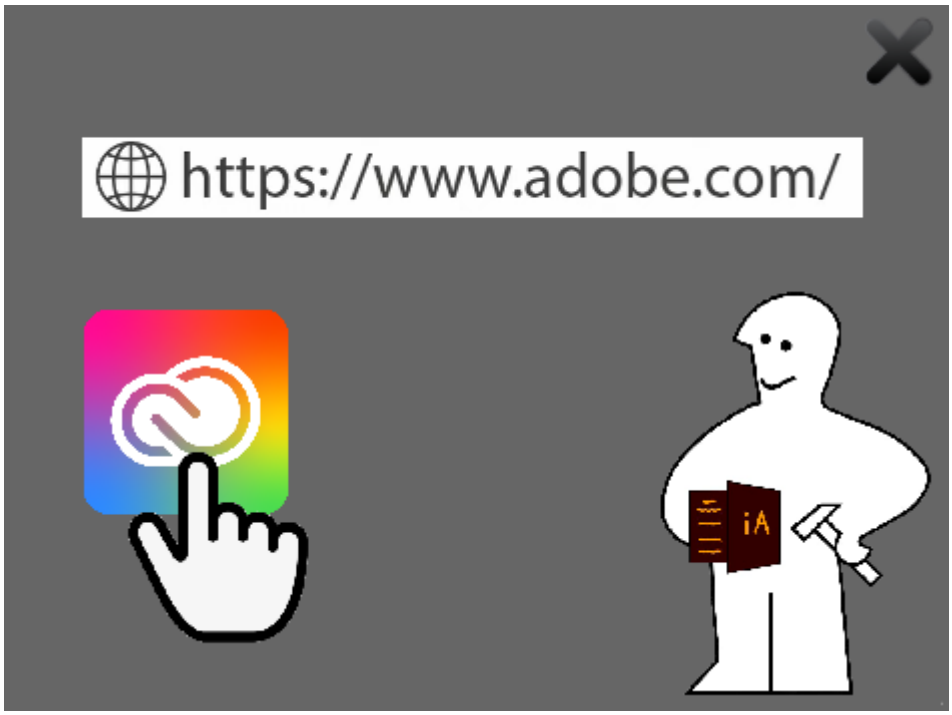

Рис. 2.25. Установка

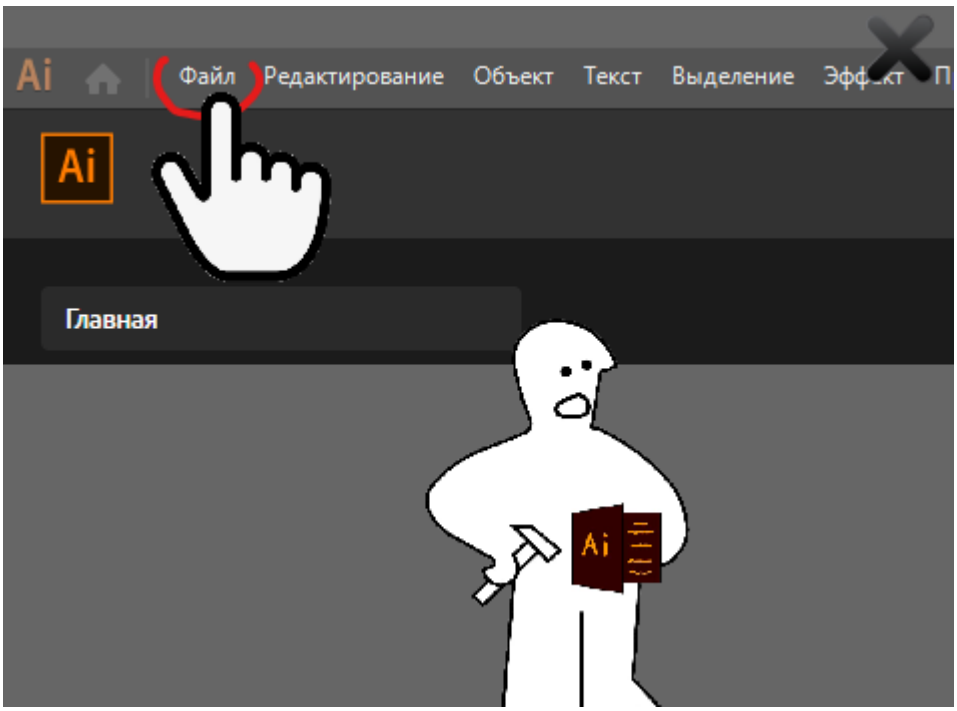

Рис. 2.26. Створення файлу

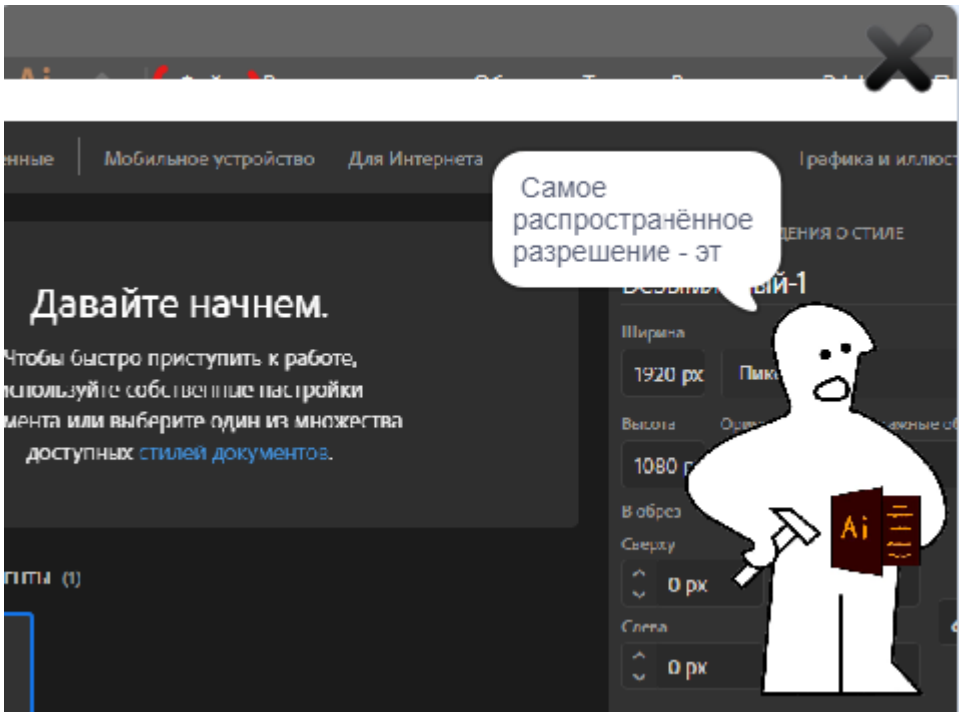

Рис. 2.27. Створення файлу продовження

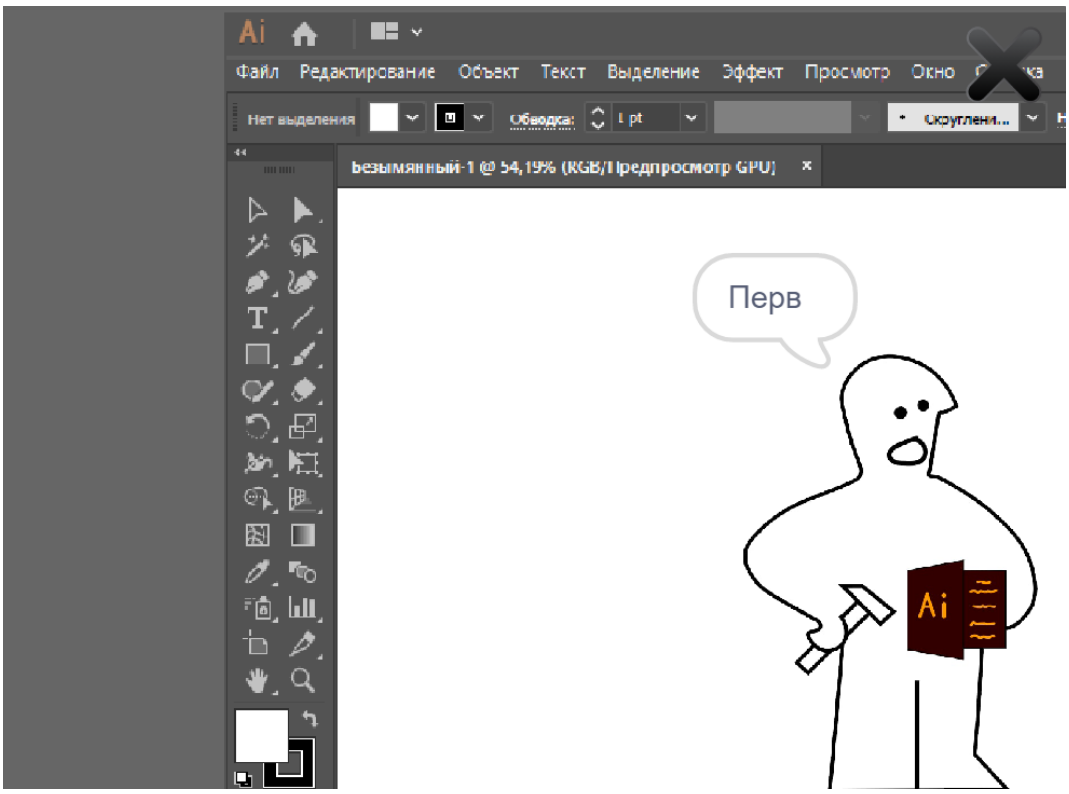

Рис. 2.28. Розбір інструментів

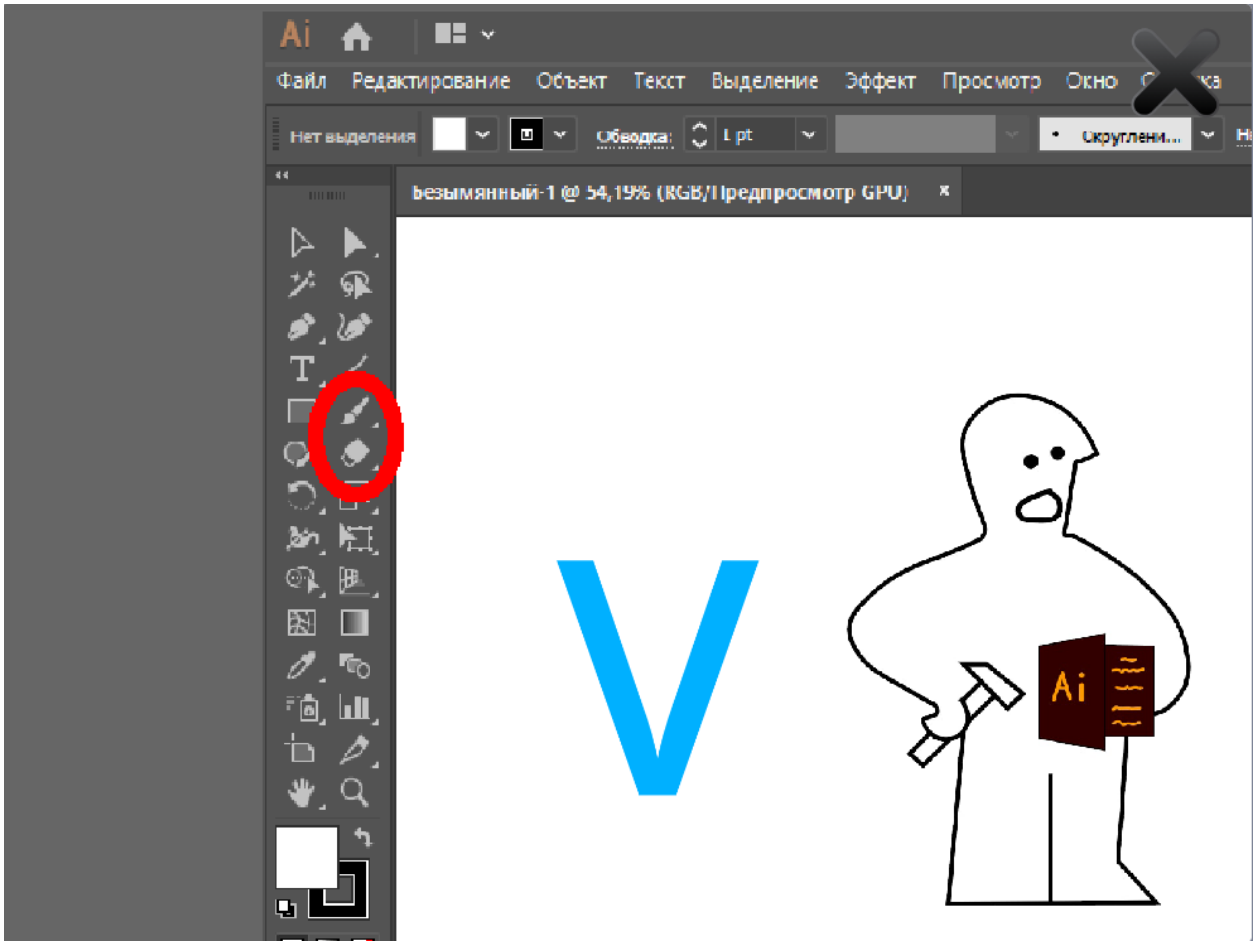

Рис. 2.29. Розбір інструментів з показом клавіші виклику

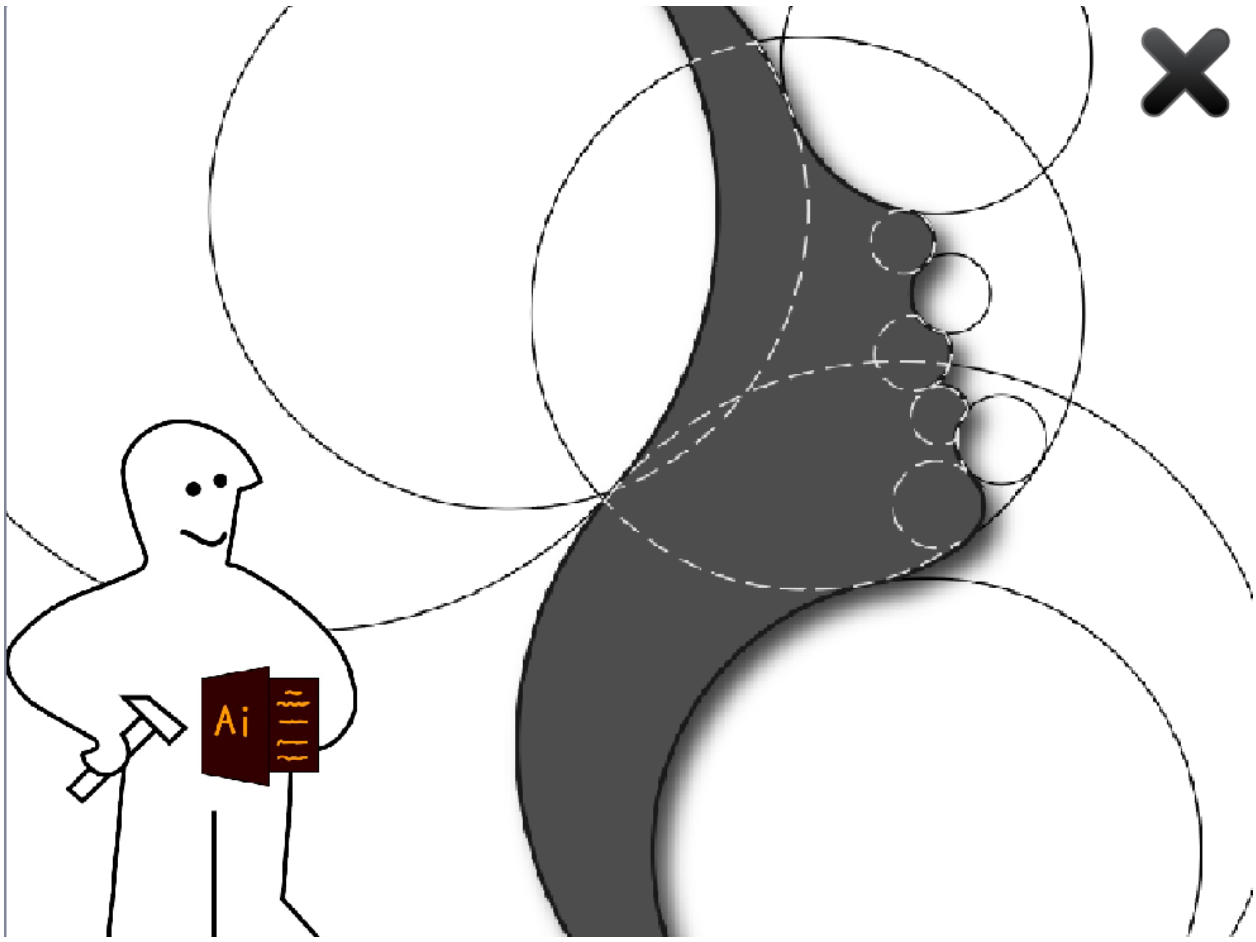

Рис. 2.30. Демонстрація прикладів рисунків

У цьому застосунку існує тільки два положення екрану: головне меню, та робочий стан. У робочому стані, застосунок буде програватися до моменту вибору малюнку або переходу в меню. Приклад вибору малюнку наведено на рис. 2.31.

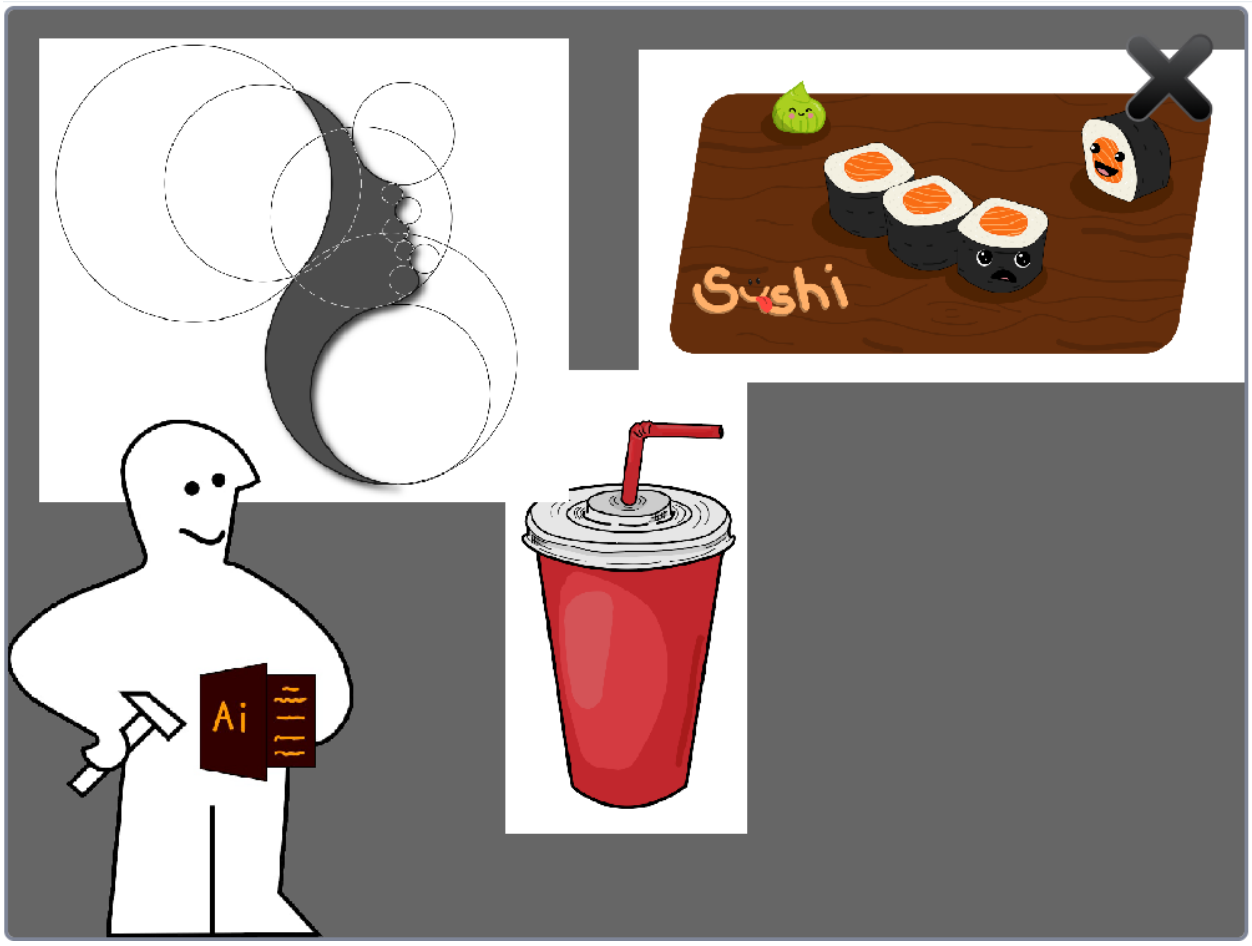

Рис. 2.3.1 Вибір малюнка

Після вибору починається гайд по розробці і збереженню проекту (рис. 2.32 – 2.40)

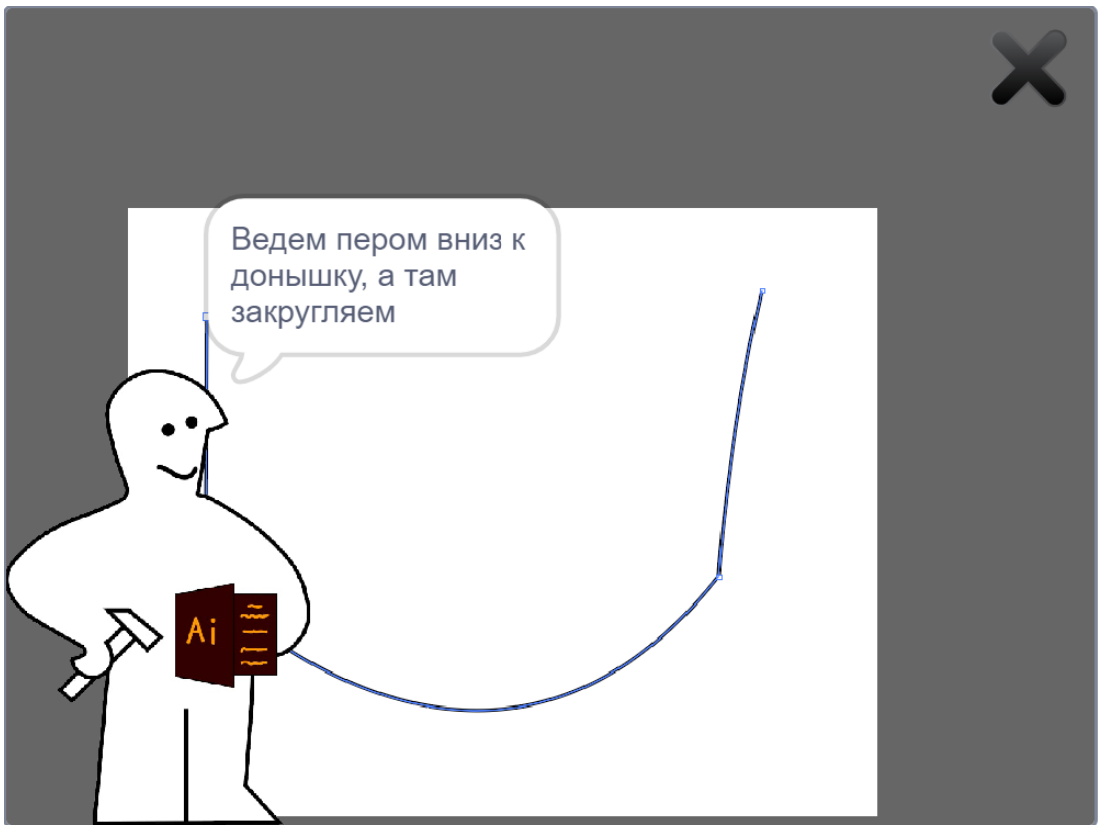

Рис. 2.32. Робота з пером

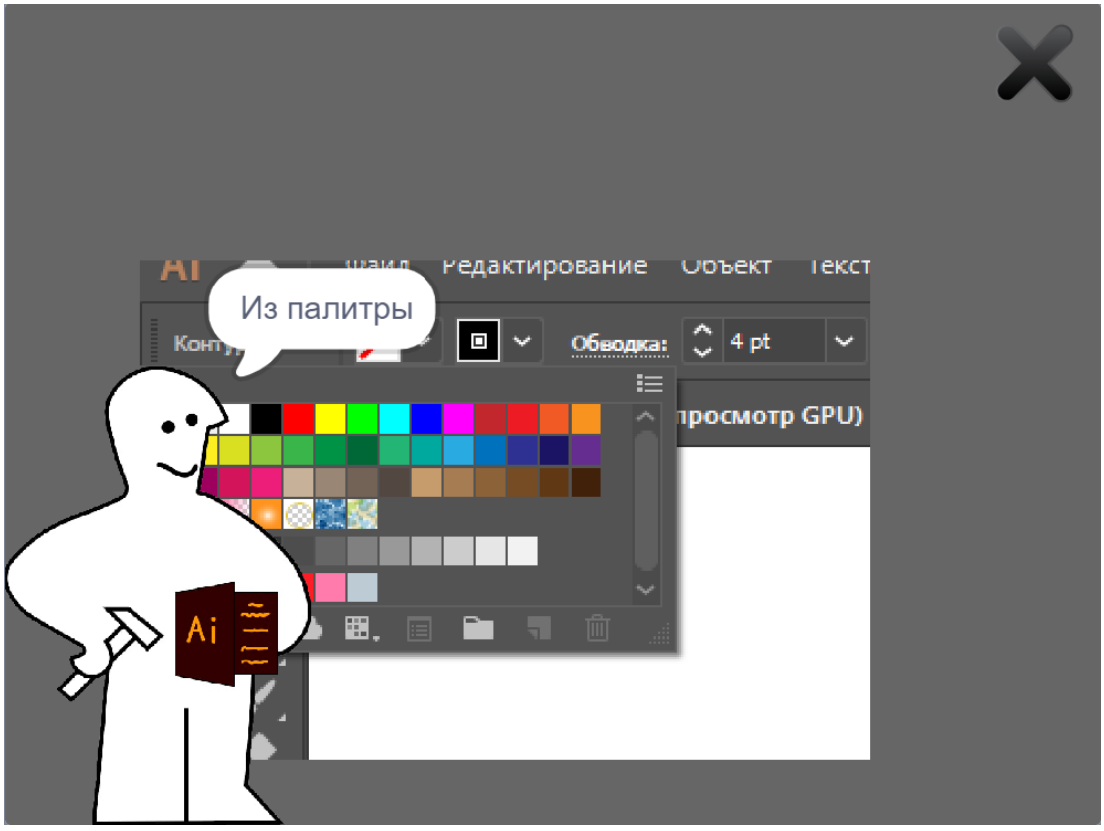

Рис. 2.33. Вибір кольору з палітри

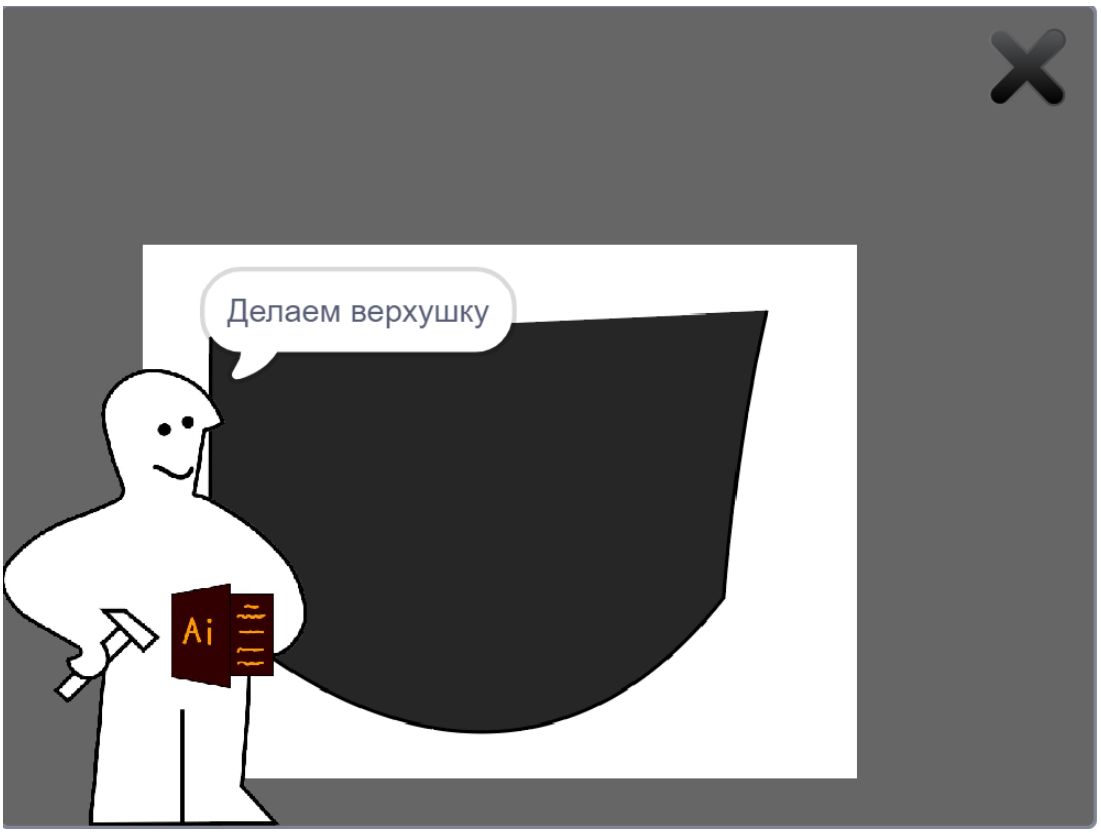

Рис. 2.34. Результат вибору кольору

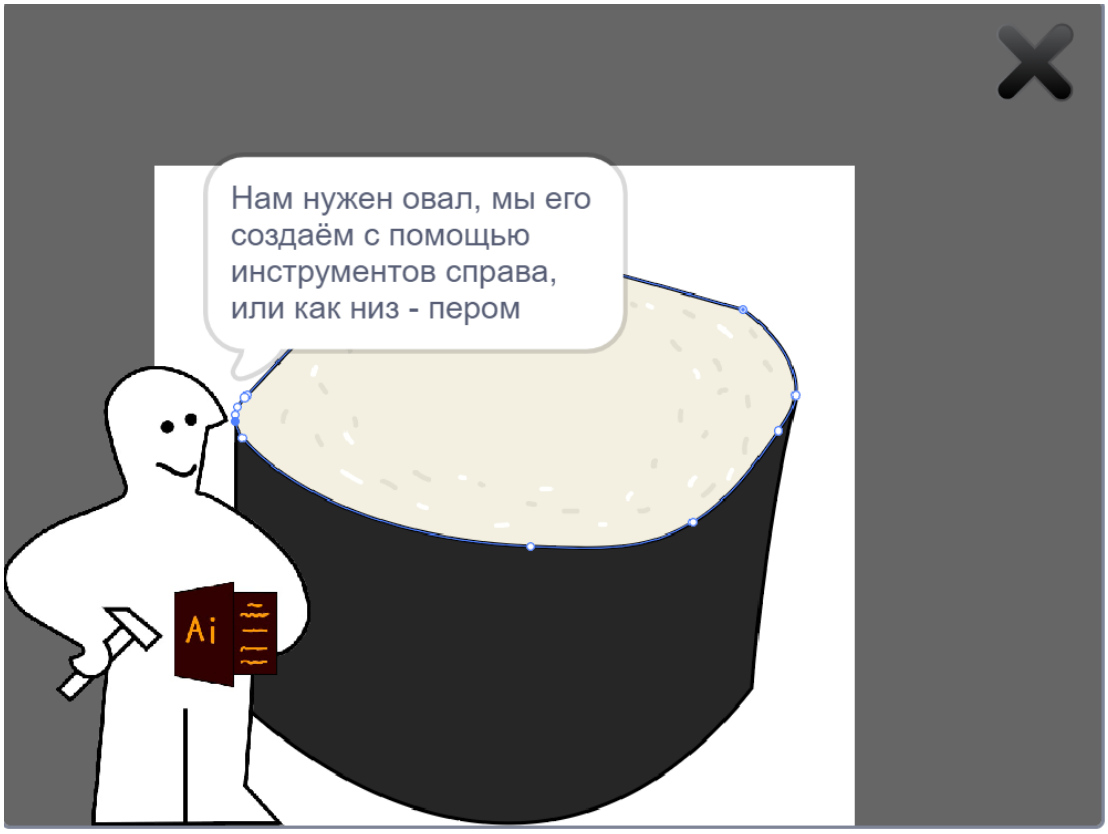

Рис. 2.35. Створення верхівки рисунку

Рис. 2.36. Створення риби

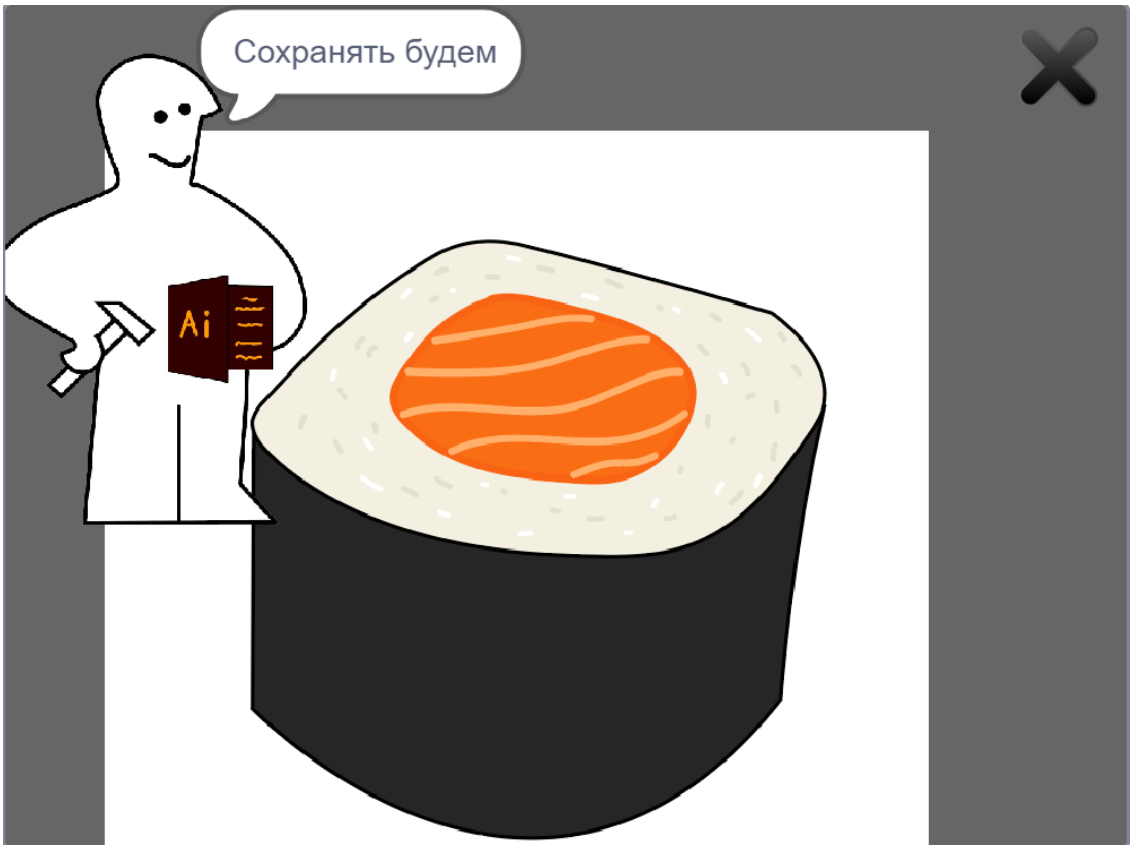

Рис.2.37. Створення деталей на рисунку

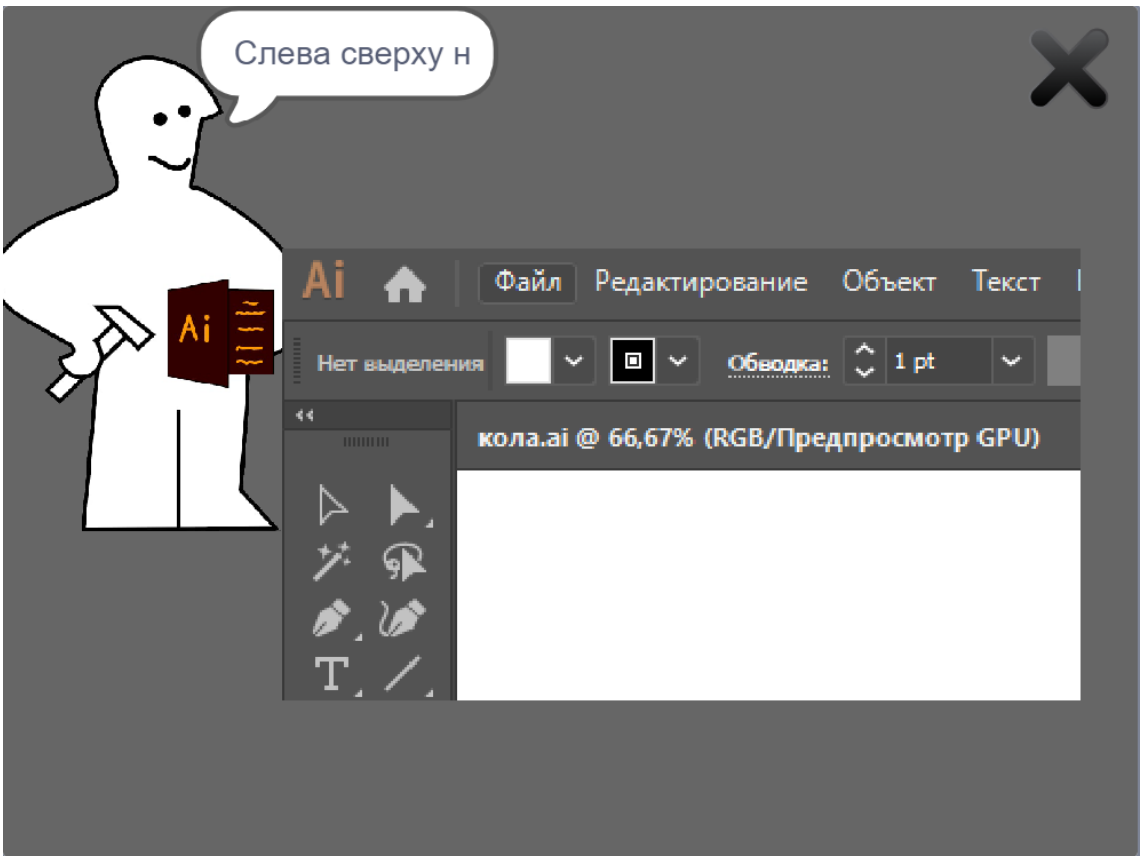

Рис. 2.38. Збереження файлу

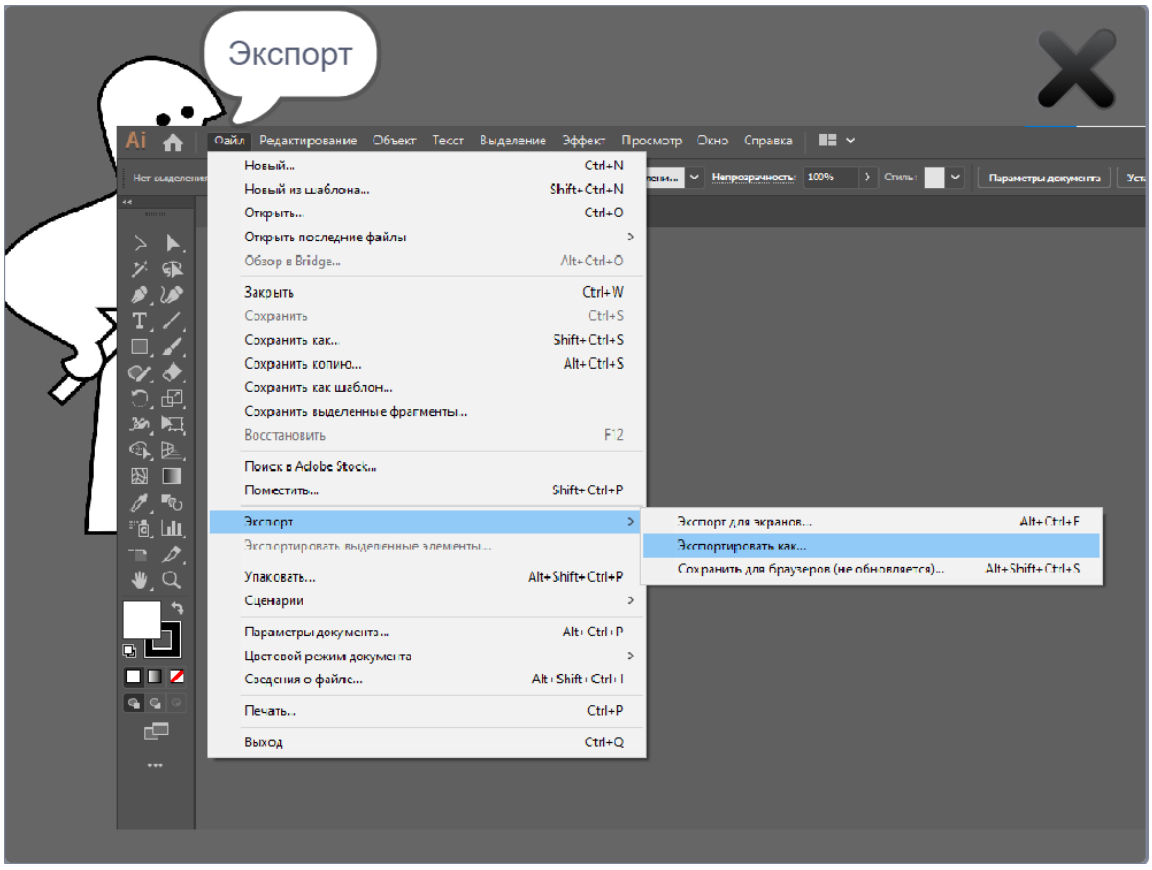

Рис. 2.39 .Продовження збереження

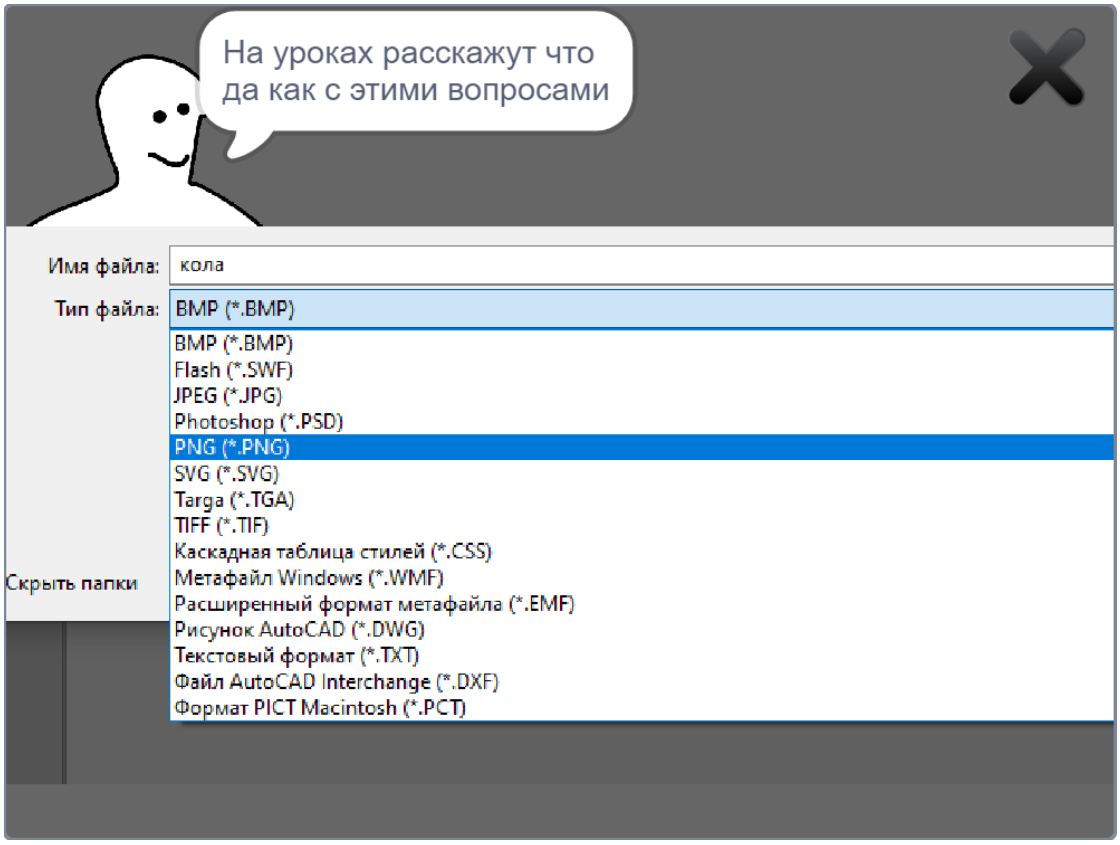

Рис. 2.40. Вибір формату збереження

Протягом всієї роботи застосунку, користувач буде бачити кнопку переходу в головне меню, у вигляді хрестику, з анімацією при наведенні та натисканні (рис. 2.41).

Рис. 2.41. Вигляд анімації наведення та натискання

Після натискання на одну з кнопок, програма продовжить програш з виділеного місця, зупинивши всі попердні події, функціх та скрипти, але продовживши програвання музики. Також при натисканні на помічника, ми побачимо плашку для введення кодового слова (рис.2.21).

Функціонал цього вікна такий же, як і меню. Він був доданий до програми виключно задля різноманіття програми та для збереження інтересу дитини, шляхом наявності додаткових функцій.

Вихід з програми виконується шляхом закривання вікна застосунку.

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

3.1. Визначення тромісткості розробки програмного забепечення

Вихідні дані розробки програмного забзпечення:

а) передбачувана кількість операторів – 1000;

б) коефіцієнт складності програми – 1,3;

в) коефіцієнт кореляції програми в ході її розробки – 0,06;

г) середня годинна заробітня плата програміста, грн/год – 45;

д) коефіцієнт кваліфікації програміста, обумовлений від стажу – 1,1;

е) вартість машино-годин ЕОМ, грн/год – 4,2;

ж) Коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі (1,2...1,5) - 1,2.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ , людино-годин,} \qquad (3.1)$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 60 людино-годин);

t_u – витрати праці на дослідження алгоритму рішення задачі,

t_a – витрати праці на розробку блок-схеми алгоритму,

t_n – витрати праці на програмування по готовій блок-схемі,

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ,

$t_д$ – витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = qC(1+p), \text{ людино-годин,} \qquad (3.2)$$

де q – передбачуване число операторів,

С – коефіцієнт складності програми,

р – коефіцієнт корекції програми в ході її розробки.

$$Q=1000*1,73*(1+0,06) = 1\ 740,4$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t = \frac{Q*B}{(75..85)*k}, \text{ людино-години,} \qquad (3.3)$$

де *B,* яке дорівнює 1,2 - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі,

k, яке дорівнює 1,2, - коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t = \frac{1740,4*1,2}{82*1,1} = 25,13, \text{ людино-годин.}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t = \frac{Q}{(20..25)*k}; \qquad (3.4)$$

$$t = \frac{1740}{22*1,1} = 71,9, \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t = \frac{Q}{(20..25)*k}; \quad (3.5)$$

$$t = \frac{1740}{21*1,1} = 78,73, \text{ людино-годин;}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t = \frac{Q}{(4..5)*k}; \quad (3.6)$$

$$t = \frac{1740}{5*1,1} = 285,24, \text{ людино-годин,}$$

– за умови комплексного налагодження завдання;

$$t^k_{отл} = 1,4*t_{отл}; \quad (3.7)$$

$t^k_{отл}$ = 1,4*285,24 = 399,33, людино годин.

Витрати праці на підготовку документації:

$$t_д = t_{дp} + t_{до}, \quad (3.8)$$

де $t_д$ – трудомісткість підготовки матеріалів і рукопису:

$$t = \frac{Q}{(15..20)*k}; \quad (3.9)$$

$$t = \frac{1740}{20*1.1} = 79,09, \text{ людино-годин}$$

де $t_{до}$ – трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75* t_{др}; \quad (3.10)$$

$t_{до}$ = 0,75 * 79,09 = 59,31, людино-годин.

$t_д$ = 79,09 + 59,31 = 138,4, людино-годин.

Отримаємо трудомісткість розробки програмного забезпечення:

$t_д = 60 + 25,13 + 71,9 + 78,73 + 285,24 + 138,4 = 659,4$, людино-годин.

У результаті ми розрахували, що в загальній кількості необхідно 659,4 людино-годин для розробки даного програмного забезпечення.

3.2 Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (3.11)$$

де $З_{ЗП}$ – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t * С_{ПР}, \text{ грн.} \quad (3.12)$$

де t – загальна трудомісткість людино-годин,

$С_{ПР}$ – середня годинна заробітня плата програмісту, грн/година. Середня заробітня плата розробника Scratch становить 250$(1.5$ в годину). Що у перерахунку на грн/год становить 45 грн/год.

$$З_{ЗП} = 45 * 659,4 = 29\,673, \text{ грн.}$$

$З_{МВ}$ - вартість машинного часу, необхідно для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} * С_{МЧ}, \text{ грн.} \quad (3.13)$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$C_{МЧ}$ - вартість машино-години ЕОМ, грн/год., дорівнює 4,2.

$$З_{МВ} = 285{,}24 * 4{,}2 = 1\ 198, \text{ грн.}$$

$$К_{ПО} = 29\ 673 + 1\ 198 = 30\ 871 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k * F_p}, \text{ мес,} \qquad (3.14)$$

де B_k – число виконавців,

F_p – місячний фонд робочого часу (при 40 годинному робочому тижні F_p = 176 годин).

$$T = \frac{659{,}4}{1*176} = 3{,}72 \text{ міс.}$$

Висновок: час розробки даного програмного забезпечення складає 659,4 людино-годин. Таким чином, очікувана тривалість розробки складе 3,72 місяця при 40 годинному робочому тижні (місячний фонд робочого часу 176 годин), а витрати на створення програмного забезпечення складатимуть 30 871 грн.

ВИСНОВКИ

Метою даної кваліфікаційної роботи є розробка додатку для навчання дітей Adobe illustrator на базі Scratch. Додаток був протестований та перевірений на баги і швидкодію командою розробників Scratch Спеціалізованої школи № 134 гуманістичного навчання та виховання, під керівником учня 5-бі класу Кафтан М.О.

Проект запускається на будь-якій операційній системі з підтримкою формату файлу EXE, та у будь-якому браузері. Застосунок не є вимогливим до апаратних прискорювачів, так як запускається і функціонує майже на кожному ПК.

Застосунок був спроектований та розроблений для навчальних цілей. Має високу лояльність до зміни наповнення учбового матеріалу. Його було створено за допомогою мови середовища програмування Scratch. Дизайн розроблений і виконаний у графічному редакторі Adobe Illustrator.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. «Як навчають у фінляндії» – Тимоті Вокер 2022р. – 256с.
2. Методичні рекомендації до виконання кваліфікаційних робіт бакалаврів напряму підготовки 122 «Комп’ютерні науки» галузі знань 12 Інформаційні технології/, Л.М. Коротенко , О.С. Шевцова; Нац. гірн. ун-т. – Д : ДВНЗ НГУ, 2017. – 50 с.
3. Методичні вказівки з виконання економічного розділу в дипломних проектах студентів спеціальності “Комп’ютерні системи” / Укладачі О.Г. Вагонова, Нікітіна О.Б. Н.Н. Романюк – Дніпропетровськ: Національний гірничий університет. – 2013. – 23с.
4. Дональд Ервін Кнут. Майстерність програмування. Том 4А. Комбінаторні алгоритми, часть 1. – М.: «Віл'ямс», 2019. – 960с.
5. «Искусство объяснять» – Ли ЛеФевер – 2018р. – 264с.
6. «Scratch для дітей. Веселий вступ до програмування з іграми, малюнками, фактами і математикою» – Маджед Марджі – 2018р. – 192с.
7. Scratch [Електронний ресурс]: https://scratch.mit.edu/
8. Adobe illustrator wiki [Електронний ресурс]: https://adobe.fandom.com/wiki/Adobe_Illustrator
9. Adobe illustrator [Електронний ресурс]: https://www.adobe.com/products/illustrator.html
10. Adobe illustrator [Електронний ресурс]: https://www.wikidata.org/wiki/Q215016
11. Scratch wiki [Електронний ресурс]: https://en.scratch-wiki.info/
12. Scratch mit media lab [Електронний ресурс]: https://mitmedialabscratch.fandom.com/wiki/Scratch_Wiki
13. «Психологія впливу» – Роберт Чалдіні – 2009р. – 272с.
14. «Анімація на Scratch. Програмування для дітей.» – Йохан Алудден, Федеріко Вал'ясінді, Федеріка Гамбел – 2018р. – 128с.

15. «Думай повільно… вирішуй швидко» – Даниель Канеман – 2011р. – 499с.
16. «Програмування для дітей. Створюй відеоігри за допомогою Scratch» – Перекладач: О.Григорович – 128стр. – 2019р.
17. «Вибухова історія людства» – Юля Смаль – 128стр. – 2021р.
18. «Ігри в Scratch для дітей» – Павел Трофимов – 184стр. – 2022р.
19. «ScratchJR для юних програмістів» – 97стр. – 2020р.
20. «Scratch і Arduino для юних программістів и конструкторів» – 177стр. – 2018р.

ДОДАТОК А

КОД ПРОГРАМИ

```
const noop = () => null

window.Scratch = {
  get vm () {
    return window.vm
  },
  get renderer () {
    return window.vm.runtime.renderer
  },
  get audioEngine () {
    return window.vm.runtime.audioEngine
  },
  get bitmapAdapter () {
    return window.vm.runtime.v2BitmapAdapter
  },
  get videoProvider () {
    return window.vm.runtime.ioDevices.video.provider
  }
}

const CLOUD_PREFIX = '\u2601 '
window.setCloud = (name, value) => {
  vm.postIOData('cloud', {
    varUpdate: {
      name: CLOUD_PREFIX + name,
      value
    }
  })
}
```

```
function postError (err) {
  setCloud('eval error', err.toString())
}

class CloudProvider {
  constructor (options) {
    this._serverUrl = options.cloud.serverUrl
    this._specialBehaviours = options.cloud.specialBehaviours
    this._options = options

    this._ws = null

    this.createVariable = noop
    this.renameVariable = noop
    this.deleteVariable = noop

    this._handleMessage = event => {
      event.data.split('\n').forEach(message => {
        if (message) {
          const { name, value } = JSON.parse(message)
          vm.postIOData('cloud', {
            varUpdate: { name, value }
          })
        }
      })
    }
    this._handleOpen = () => {
      this._sendData({ method: 'handshake' })
    }
    this._handleClose = () => {
      setTimeout(() => this._openConnection(), 500)
    }
```

```
  this.handleUrlChange = () => {
    setCloud('url', window.location.href)
  }
  if (this._specialBehaviours) {
    window.addEventListener('hashchange', this.handleUrlChange)
    window.addEventListener('popstate', this.handleUrlChange)
    // Paste output
    window.addEventListener('paste', event => {
      setCloud(
        'pasted',
        (event.clipboardData || window.clipboardData).getData('text')
      )
    })
  }

  if (this._serverUrl) {
    this._openConnection()
  }
}

_openConnection () {
  try {
    this._ws = new WebSocket(this._serverUrl)
  } catch (error) {
    console.warn(error)
    return
  }
  this._ws.onmessage = this._handleMessage
  this._ws.onopen = this._handleOpen
  this._ws.onclose = this._handleClose
}
```

```
_sendData (data) {
  data.user = this._options.username
  data.project_id = this._options.cloud.projectId
  this._ws.send(JSON.stringify(data) + '\n')
}

updateVariable (name, value) {
  if (this._specialBehaviours) {
    let matched = true
    if (name === CLOUD_PREFIX + 'eval') {
      try {
        Promise.resolve(eval(value))
          .then(output => {
            setCloud('eval output', output)
          })
          .catch(postError)
      } catch (error) {
        postError(error)
      }
    } else if (name === CLOUD_PREFIX + 'open link') {
      try {
        window.open(value, '_blank')
      } catch (error) {
        postError(error)
      }
    } else if (name === CLOUD_PREFIX + 'redirect') {
      window.location = value
    } else if (name === CLOUD_PREFIX + 'set clipboard') {
      try {
        navigator.clipboard.writeText(value).catch(postError)
      } catch (error) {
```

```
      postError(error)
    }
  } else if (name === CLOUD_PREFIX + 'set server ip') {
    this._cloudHost = value
    if (this._ws) {
      this._ws.onclose = noop
      this._ws.close()
    }
    this._openConnection()
  } else if (name === CLOUD_PREFIX + 'username') {
    this._options.username = value
    vm.postIOData('userData', { username: value })
  } else {
    matched = false
  }
  if (matched) {
    return
  }
}
if (
  !this._serverUrl ||
  (this._specialBehaviours &&
    name.startsWith(CLOUD_PREFIX + 'local storage'))
) {
  try {
    localStorage.setItem('[s3] ' + name, value)
  } catch (error) {
    postError(error)
  }
} else {
  this._sendData({ method: 'set', name, value })
}
```

```
  }

  requestCloseConnection () {
    if (
      this._ws &&
      this._ws.readyState !== WebSocket.CLOSING &&
      this._ws.readyState !== WebSocket.CLOSED
    ) {
      this._ws.onclose = noop
      this._ws.close()
    }
  }
}

// Based on
// https://github.com/LLK/scratch-gui/blob/7b658c60c7c04055e575601a861195fe6c9933f3/src/lib/video/camera.js
// https://github.com/LLK/scratch-gui/blob/7b658c60c7c04055e575601a861195fe6c9933f3/src/lib/video/video-provider.js
class VideoProvider {
  constructor (width, height) {
    this._dimensions = [width, height]
    this.mirror = true
    this._frameCacheTimeout = 16
    this._video = null
    this._track = null
    this._workspace = []
  }

  get video () {
    return this._video
  }
```

```
enableVideo () {
  this.enabled = true
  return this._setupVideo()
}

disableVideo () {
  this.enabled = false
  if (this._singleSetup) {
    this._singleSetup
      .then(this._teardown.bind(this))
      .catch(err => this.onError(err))
  }
}

_teardown () {
  if (this.enabled === false) {
    requestStack.pop()
    const disableTrack = requestStack.length === 0
    this._singleSetup = null
    this._video = null
    if (this._track && disableTrack) {
      this._track.stop()
    }
    this._track = null
  }
}

getFrame ({
  dimensions = this._dimensions,
  mirror = this.mirror,
  format = 'image-data',
```

```
  cacheTimeout = this._frameCacheTimeout
 }) {
  if (!this.videoReady) {
   return null
  }
  const [width, height] = dimensions
  const workspace = this._getWorkspace({
   dimensions,
   mirror: Boolean(mirror)
  })
  const { videoWidth, videoHeight } = this._video
  const { canvas, context, lastUpdate, cacheData } = workspace
  const now = Date.now()

  if (lastUpdate + cacheTimeout < now) {
   if (mirror) {
    context.scale(-1, 1)
    context.translate(width * -1, 0)
   }

   context.drawImage(
    this._video,
    0,
    0,
    videoWidth,
    videoHeight,
    0,
    0,
    width,
    height
   )
```

```
    context.setTransform(1, 0, 0, 1, 0, 0)
    workspace.lastUpdate = now
  }

  if (!cacheData[format]) {
    cacheData[format] = { lastUpdate: 0 }
  }
  const formatCache = cacheData[format]

  if (formatCache.lastUpdate + cacheTimeout < now) {
    if (format === 'image-data') {
      formatCache.lastData = context.getImageData(0, 0, width, height)
    } else if (format === 'canvas') {
      formatCache.lastUpdate = Infinity
      formatCache.lastData = canvas
    } else {
      console.error(`video io error - unimplemented format ${format}`)
      formatCache.lastUpdate = Infinity
      formatCache.lastData = null
    }

    formatCache.lastUpdate = Math.max(
      workspace.lastUpdate,
      formatCache.lastUpdate
    )
  }

  return formatCache.lastData
}

onError (error) {
  console.error('Unhandled video io device error', error)
```

```
}

_setupVideo () {
 if (this._singleSetup) {
  return this._singleSetup
 }

 if (requestStack.length === 0) {
  this._singleSetup = navigator.mediaDevices.getUserMedia({
   audio: false,
   video: {
    width: { min: width, ideal: (480 * width) / height },
    height: { min: height, ideal: 480 }
   }
  })
  requestStack.push(streamPromise)
 } else if (requestStack.length > 0) {
  this._singleSetup = requestStack[0]
  requestStack.push(true)
 }
 this._singleSetup
  .then(stream => {
   this._video = document.createElement('video')

   try {
    this._video.srcObject = stream
   } catch (_error) {
    this._video.src = window.URL.createObjectURL(stream)
   }
   this._video.play()
   this._track = stream.getTracks()[0]
   return this
```

```
    })
    .catch(error => {
      this._singleSetup = null
      this.onError(error)
    })

  return this._singleSetup
}

get videoReady () {
  if (!this.enabled || !this._video || !this._track) {
    return false
  }
  const { videoWidth, videoHeight } = this._video
  return (
    typeof videoWidth === 'number' &&
    typeof videoHeight === 'number' &&
    videoWidth > 0 &&
    videoHeight > 0
  )
}

_getWorkspace ({ dimensions, mirror }) {
  let workspace = this._workspace.find(
    space =>
      space.dimensions.join('-') === dimensions.join('-') &&
      space.mirror === mirror
  )
  if (!workspace) {
    workspace = {
      dimensions,
      mirror,
```

```
        canvas: document.createElement('canvas'),
        lastUpdate: 0,
        cacheData: {}
      }
      workspace.canvas.width = dimensions[0]
      workspace.canvas.height = dimensions[1]
      workspace.context = workspace.canvas.getContext('2d')
      this._workspace.push(workspace)
    }
    return workspace
  }
}

const fullscreenBtn = document.getElementById('fullscreen-btn')
const exitFullscreen =
  document.exitFullscreen ||
  document.msExitFullscreen ||
  document.mozCancelFullScreen ||
  document.webkitExitFullscreen
const requestFullscreen =
  document.body.requestFullscreen ||
  document.body.msRequestFullscreen ||
  document.body.mozRequestFullScreen ||
  document.body.webkitRequestFullscreen
function isFullscreen () {
  return (
    document.fullscreenElement ||
    document.mozFullScreenElement ||
    document.webkitFullscreenElement ||
    document.msFullscreenElement
  )
}
```

```
function toggleFullscreen () {
  if (isFullscreen()) {
    exitFullscreen.call(document)
  } else {
    requestFullscreen.call(document.body)
  }
}
fullscreenBtn.addEventListener('click', () => {
  fullscreenBtn.blur()
  toggleFullscreen()
})
function handleFullscreenChange () {
  if (isFullscreen()) {
    document.body.classList.add('fullscreen')
  } else {
    document.body.classList.remove('fullscreen')
  }
}
document.addEventListener('fullscreenchange', handleFullscreenChange)
document.addEventListener('mozfullscreenchange', handleFullscreenChange)
document.addEventListener('webkitfullscreenchange', handleFullscreenChange)
document.addEventListener('msfullscreenchange', handleFullscreenChange)

window.init = async ({ width, height, ...options }) => {
  window.vm = new window.NotVirtualMachine(width, height)
  vm.setCompatibilityMode(options.fps)
  vm.setTurboMode(options.turbo)
  vm.requireLimits(options.limits, { fencing: options.fencing })

  const storage = new ScratchStorage()
  const AssetType = storage.AssetType
  if (options.assets.project) {
```

```
  storage.addWebStore([AssetType.Project], () => options.assets.project)
  storage.addWebStore(
    [AssetType.ImageVector, AssetType.ImageBitmap, AssetType.Sound],
    ({ assetId, dataFormat }) => options.assets[`${assetId}.${dataFormat}`]
  )
}
const progress = document.getElementById('loading-progress')
if (options.loadingProgress) {
  const _load = storage.webHelper.load
  let total = 0,
    complete = 0
  storage.webHelper.load = function (...args) {
    const result = _load.call(this, ...args)
    total += 1
    const percentage = (complete / total) * 100
    progress.dataset.progress = percentage.toFixed(2) + '%'
    progress.style.setProperty('--progress', percentage + '%')
    result.then(() => {
      complete += 1
      const percentage = (complete / total) * 100
      progress.dataset.progress = percentage.toFixed(2) + '%'
      progress.style.setProperty('--progress', percentage + '%')
    })
    return result
  }
}
vm.attachStorage(storage)

function resize () {
  const rect = canvas.getBoundingClientRect()
  renderer.resize(rect.width, rect.height)
  if (options.stretchStage) {
```

```
    monitorWrapper.style.transform = `scaleX(${rect.width /
      width}) scaleY(${rect.height / height})`
  } else {
    monitorWrapper.style.transform = `scale(${rect.height / height})`
  }
}
const monitorWrapper = document.getElementById('monitors')
const canvas = document.getElementById('stage')
const renderer = new window.ScratchRender(
  canvas,
  -width / 2,
  width / 2,
  -height / 2,
  height / 2
)
resize()
vm.attachRenderer(renderer)

vm.attachAudioEngine(new window.AudioEngine())
vm.attachV2BitmapAdapter(
  new ScratchSVGRenderer.BitmapAdapter(null, null, width, height)
)
vm.setVideoProvider(new VideoProvider(width, height))
vm.start()

if (options.extensionCount > 0) {
  const OldWorker = window.Worker
  window.Worker = class extends OldWorker {
    // deno-lint-ignore constructor-super
    constructor (...args) {
      if (args[0].endsWith('extension-worker.js')) {
        if (options.extensionWorker.url) {
```

```
          super(options.extensionWorker.url, ...args.slice(1))
        } else {
          super(
            URL.createObjectURL(
              new Blob([options.extensionWorker.script], {
                type: 'text/javascript'
              })
            ),
            ...args.slice(1)
          )
        }
      } else {
        super(...args)
      }
    }
  }
}

/* https://github.com/LLK/scratch-gui/blob/develop/src/containers/stage.jsx#L176-L300 */
const getEventXY = e => {
  if (e.touches && e.touches[0]) {
    return { x: e.touches[0].clientX, y: e.touches[0].clientY }
  } else if (e.changedTouches && e.changedTouches[0]) {
    return { x: e.changedTouches[0].clientX, y: e.changedTouches[0].clientY }
  }
  return { x: e.clientX, y: e.clientY }
}
let mouseDown = false
let mouseDownPosition = null
let mouseDownTimeoutId = null
let isDragging = false
let dragId = null
```

```
let dragOffset = null
function cancelMouseDownTimeout () {
  if (mouseDownTimeoutId !== null) {
    clearTimeout(mouseDownTimeoutId)
  }
  mouseDownTimeoutId = null
}
// https://github.com/LLK/scratch-gui/blob/develop/src/containers/stage.jsx#L337-L366
function handleStartDrag () {
  if (dragId || !mouseDownPosition) return
  const drawableId = renderer.pick(mouseDownPosition.x, mouseDownPosition.y)
  if (drawableId === null) return
  const targetId = vm.getTargetIdForDrawableId(drawableId)
  if (targetId === null) return
  const target = vm.runtime.getTargetById(targetId)
  if (!target || !target.draggable) return
  target.goToFront()
  const drawableData = renderer.extractDrawable(
    drawableId,
    mouseDownPosition.x,
    mouseDownPosition.y
  )
  vm.startDrag(targetId)
  isDragging = true
  dragId = targetId
  dragOffset = drawableData.scratchOffset
}
const accumulative = { x: 0, y: 0 }
function postIfPointerLocked (e, isDown) {
  if (
    document.pointerLockElement === canvas ||
    document.mozPointerLockElement === canvas
```

```
  ) {
    accumulative.x += e.movementX
    accumulative.y += e.movementY
    vm.postIOData('mouse', {
      isDown,
      x: accumulative.x + width / 2,
      y: accumulative.y + height / 2,
      canvasWidth: width,
      canvasHeight: height
    })
    return true
  } else {
    return false
  }
}
function postMouse (event, isDown) {
  const { x, y } = getEventXY(event)
  const rect = canvas.getBoundingClientRect()
  const mousePosition = { x: x - rect.left, y: y - rect.top }
  vm.postIOData('mouse', {
    isDown,
    ...mousePosition,
    canvasWidth: rect.width,
    canvasHeight: rect.height
  })
  return { mousePosition, rect }
}
function handleMouseMove (event) {
  if (postIfPointerLocked(event)) return
  const { mousePosition, rect } = postMouse(event)
  if (mouseDown && !isDragging) {
    const distanceFromMouseDown = Math.hypot(
```

```
      mousePosition.x - mouseDownPosition.x,
      mousePosition.y - mouseDownPosition.y
    )
    if (distanceFromMouseDown > 3) {
      cancelMouseDownTimeout()
      handleStartDrag()
    }
  }
  if (mouseDown && isDragging) {
    const nativeSize = renderer.getNativeSize()
    vm.postSpriteInfo({
      x:
        (nativeSize[0] / rect.width) * (mousePosition.x - rect.width / 2) +
        dragOffset[0],
      y: -(
        (nativeSize[1] / rect.height) * (mousePosition.y - rect.height / 2) +
        dragOffset[1]
      ),
      force: true
    })
  }
}
function handleMouseDown (event) {
  mouseDown = true
  vm.postIOData('keyboard', {
    key: 'Mouse' + event.which,
    isDown: true
  })
  if (postIfPointerLocked(event, true)) return
  mouseDownPosition = postMouse(event, true).mousePosition
  mouseDownTimeoutId = setTimeout(handleStartDrag, 400)
  event.preventDefault()
```

```
  if (!document.body.classList.contains('asking')) {
    window.focus()
  }
}
function handleMouseUp (event) {
  cancelMouseDownTimeout()
  mouseDown = false
  mouseDownPosition = null
  if (isDragging) {
    vm.stopDrag(dragId)
    isDragging = false
    dragOffset = null
    dragId = null
  }
  vm.postIOData('keyboard', {
    key: 'Mouse' + event.which,
    isDown: false
  })
  if (postIfPointerLocked(event, false)) return
  postMouse(event, false)
}
document.addEventListener('mousemove', handleMouseMove)
document.addEventListener('mouseup', handleMouseUp)
document.addEventListener('touchmove', handleMouseMove)
document.addEventListener('touchend', handleMouseUp, { passive: false })
canvas.addEventListener('wheel', event => {
  vm.postIOData('mouseWheel', event)
  event.preventDefault()
})
window.addEventListener('resize', resize)
canvas.addEventListener('contextmenu', event => {
  event.preventDefault()
```

```
})
if (options.pointerLock) {
  canvas.requestPointerLock =
    canvas.requestPointerLock || canvas.mozRequestPointerLock
  canvas.addEventListener('click', () => {
    canvas.requestPointerLock()
  })
}

/**
 * Maps e.key to e.code. For example, Shift -> ShiftRight. This is because the
 * keyup event doesn't fire when you let go of one shift if the other shift
 * key is still held down.
 */
const keyToCode = new Map()
function postKey (event, isDown) {
  const key = !event.key || event.key === 'Dead' ? event.keyCode : event.key
  vm.postIOData('keyboard', {
    key: key,
    isDown
  })
  vm.postIOData('keyboard', {
    key: 'code_' + event.code,
    isDown
  })
}
document.addEventListener('keydown', event => {
  if (event.target !== document && event.target !== document.body) return
  const prevCode = keyToCode.get(event.key)
  if (prevCode) {
    if (prevCode !== event.code) {
      vm.postIOData('keyboard', {
```

```
        key: 'code_' + prevCode,
        isDown: false
      })
    }
  }
  postKey(event, true)
  keyToCode.set(event.key, event.code)
  if (event.keyCode === 32 || (event.keyCode >= 37 && event.keyCode <= 40)) {
    event.preventDefault()
  }
  if (
    event.key === 'f' &&
    (event.ctrlKey || event.metaKey) &&
    !event.shiftKey &&
    !event.altKey
  ) {
    toggleFullscreen()
    event.preventDefault()
  }
})
document.addEventListener('keyup', e => {
  postKey(e, false)
  keyToCode.delete(e.key)
  if (e.target !== document && e.target !== document.body) {
    e.preventDefault()
  }
})
vm.postIOData('keyboard', {
  key: 'HTMLifier',
  isDown: true
})
```

```
const question = document.getElementById('question')
const askBox = document.getElementById('answer')
vm.runtime.addListener('QUESTION', questionData => {
  // null means the question was interrupted by stop script block
  if (questionData === null) {
    document.body.classList.remove('asking')
  } else {
    document.body.classList.add('asking')
    question.textContent = questionData
    askBox.value = ''
    askBox.focus()
  }
})
askBox.addEventListener('keydown', event => {
  if (event.key === 'Enter') {
    // The asking class must be removed first because the VM may start the
    // next question immediately inside the .emit call
    document.body.classList.remove('asking')
    vm.runtime.emit('ANSWER', askBox.value)
  }
})

const getVariable = (targetId, variableId) => {
  const target = targetId
    ? vm.runtime.getTargetById(targetId)
    : vm.runtime.getTargetForStage()
  return target.variables[variableId]
}
const monitorStates = {}
vm.runtime.addListener('MONITORS_UPDATE', monitors => {
  monitors.forEach((record, id) => {
    const {
```

```
    value,
    visible,
    mode,
    x,
    y,
    width,
    height,
    params,
    opcode,
    spriteName,
    sliderMin,
    sliderMax,
    isDiscrete,
    targetId
  } = record

  if (!monitorStates[id]) {
    const label = document.createElement('span')
    label.className = 'monitor-label'
    const name = params.VARIABLE || params.LIST || opcode
    label.textContent = spriteName ? `${spriteName}: ${name}` : name

    const value = document.createElement('span')
    value.className = 'monitor-value'

    const monitor = document.createElement('div')
    monitor.className = 'monitor ' + mode
    monitor.style.left = x + 'px'
    monitor.style.top = y + 'px'
    monitor.append(label, value)

    monitorStates[id] = { monitor, valueElem: value, wasVisible: true }
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	**Опис**
Пояснювальні документи	
Диплом_Кафтан.doc	Пояснювальна записка до кваліфікаційної роботи. Документ Word
Диплом_Кафтан.pdf	Пояснювальна записка до кваліфікаційної роботи у форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і відкомпільовану програму
Презентація	
Презентація_Кафтан.pptx	Презентація кваліфікаційної роботи